# Computing Integrals Involving the Matrix Exponential

CHARLES F. VAN LOAN

*Abstract*—**A new algorithm for computing integrals involving the matrix exponential is given. The method employs diagonal Padé approximation with scaling and squaring. Rigorous truncation error bounds are given and incorporated in a Fortran subroutine. The computational aspects of this program are discussed and compared with existing techniques.**

## I. Introduction

LET $A$, $B$, and $Q_c$ be real matrices of dimensions $n\times n$, $n\times p$, and $n\times n$, respectively. Assume that $Q_c$ is symmetric ($Q_c^T = Q_c$) and positive semidefinite ($x^TQ_cx \geq 0$). In this paper, we present a new method for computing the integrals

$$H(\Delta)=\int_0^\Delta e^{As}B\,ds \tag{1.1}$$

$$Q(\Delta)=\int_0^\Delta e^{A^Ts}Q_ce^{As}\,ds \tag{1.2}$$

$$M(\Delta)=\int_0^\Delta e^{A^Ts}Q_cH(s)\,ds \tag{1.3}$$

$$W(\Delta)=\int_0^\Delta H(s)^TQ_cH(s)\,ds. \tag{1.4}$$

The need for computing these integrals arises in several applications, notably the optimal sampled-data regulator problem [1], [3].

The method we shall propose involves a) computing the exponential of a certain block triangular matrix and b) combining various submatrices of the result to obtain (1.1)–(1.4). To illustrate the basic idea, if

$$\exp\left(\begin{bmatrix} -A^T & Q_c \\ 0 & A \end{bmatrix}\Delta\right)=\begin{bmatrix} F_2(\Delta) & G_2(\Delta) \\ 0 & F_3(\Delta) \end{bmatrix}$$

then it can be shown that

$$Q(\Delta)=F_3(\Delta)^TG_2(\Delta).$$

In Section II, we establish this result and other, analogous expressions for $H(\Delta)$, $M(\Delta)$, and $W(\Delta)$.

In order to harness these results in a practical algorithm, it is necessary to have an effective means for computing matrix exponentials. The algorithm we formulate in Section III uses diagonal Padé approximation with repeated squaring. These approximations have proven successful when just $e^{A\Delta}$ is desired and fortunately their attractive features carry over to this new application.

In Section IV, we derive truncation error bounds for our computed versions of (1.1)–(1.4). These bounds can then be used in practice to select approximations of appropriate accuracy. A Fortran subroutine called Padé has been implemented which does this. Some computational aspects of this program are discussed in Section V. Finally, in Section VI we assess our algorithm relative to the work others have done in this area.

## II. Theoretical Results

In this section we relate the integrals (1.1)–(1.4) to the exponential of a certain block triangular matrix. We first prove a general result about such exponentials.

*Theorem 1:* Let $n_1$, $n_2$, $n_3$, and $n_4$ be positive integers, and set $m$ to be their sum. If the $m\times m$ block triangular matrix $C$ is defined by

$$C=\begin{bmatrix} A_1 & B_1 & C_1 & D_1 \\ 0 & A_2 & B_2 & C_2 \\ 0 & 0 & A_3 & B_3 \\ 0 & 0 & 0 & A_4 \end{bmatrix}\begin{matrix}\}\,n_1\\ \}\,n_2\\ \}\,n_3\\ \}\,n_4\end{matrix}$$

$$\underbrace{\phantom{00}}_{n_1}\ \underbrace{\phantom{00}}_{n_2}\ \underbrace{\phantom{00}}_{n_3}\ \underbrace{\phantom{00}}_{n_4}$$

then for $t \geq 0$

$$e^{Ct}=\begin{bmatrix} F_1(t) & G_1(t) & H_1(t) & K_1(t) \\ 0 & F_2(t) & G_2(t) & H_2(t) \\ 0 & 0 & F_3(t) & G_3(t) \\ 0 & 0 & 0 & F_4(t) \end{bmatrix}$$

where

$$F_j(t)=e^{A_jt}, \qquad j=1,2,3,4$$

$$G_j(t)=\int_0^t e^{A_j(t-s)}B_je^{A_{j+1}s}\,ds, \qquad j=1,2,3$$

$$H_j(t)=\int_0^t e^{A_j(t-s)}C_je^{A_{j+2}s}\,ds + \int_0^t\int_0^s e^{A_j(t-s)}B_je^{A_{j+1}(s-r)} \cdot B_{j+1}e^{A_{j+2}r}\,dr\,ds, \qquad j=1,2$$

Manuscript received June 17, 1977. Paper recommended by E. Polak, Chairman of the Computational Methods and Discrete Systems Committee. This work was partially supported by NSF Grant MCS76-08686.

The author is with the Department of Computer Science, Cornell University, Ithaca, NY 14853.

0018-9286/78/0600-0395$00.75 

and

$$K_1(t)=\int_0^t e^{A_1(t-s)}D_1e^{A_4s}\,ds+\int_0^t\int_0^s e^{A_1(t-s)}\left[C_1e^{A_3(s-r)}B_3+B_1e^{A_2(s-r)}C_2\right]e^{A_4r}\,dr\,ds$$

$$+\int_0^t\int_0^s\int_0^r e^{A_1(t-s)}B_1e^{A_2(s-r)}B_2e^{A_3(r-w)}B_3e^{A_4w}\,dw\,dr\,ds.$$

*Proof:* Since all powers of $C$ have the same block triangular structure, it is clear that $e^{Ct}$ has the form indicated. By equating submatrices in the equation

$$\frac{d}{dt}\left[e^{Ct}\right]=Ce^{Ct}\quad I=e^{Ct}|_{t=0}$$

we are led to the following differential equations:

$$\begin{aligned}
&\dot F_j(t)=A_jF_j(t) && F_j(0)=I, && j=1,2,3,4\\
&\dot G_j(t)=A_jG_j(t)+B_jF_{j+1}(t) && G_j(0)=0, && j=1,2,3\\
&\dot H_j(t)=A_jH_j(t)+B_jG_{j+1}(t)+C_jF_{j+2}(t) && H_j(0)=0, && j=1,2\\
&\dot K_1(t)=A_1K_1(t)+B_1H_2(t)+C_1G_3(t)+D_1F_4(t) && K_1(0)=0.
\end{aligned}$$

The theorem follows by solving these equations, respectively, for $F_j(t)$, $G_j(t)$, $H_j(t)$, and $K_1(t)$. ■

If we apply this theorem to the $(3n+p)\times(3n+p)$ matrix

$$\hat C=\begin{bmatrix}-A^T & I & 0 & 0\\ 0 & -A^T & Q_c & 0\\ 0 & 0 & A & B\\ 0 & 0 & 0 & 0\end{bmatrix}$$

we find

$$e^{\hat Ct}=\begin{bmatrix}\hat F_1(t) & \hat G_1(t) & \hat H_1(t) & \hat K_1(t)\\ 0 & \hat F_2(t) & \hat G_2(t) & \hat H_2(t)\\ 0 & 0 & \hat F_3(t) & \hat G_3(t)\\ 0 & 0 & 0 & \hat F_4(t)\end{bmatrix}$$

where

$$\hat F_3(t)=e^{At}$$

$$\hat G_2(t)=e^{-A^Tt}\int_0^t e^{A^Ts}Q_ce^{As}\,ds$$

$$\hat G_3(t)=\int_0^t e^{A(t-s)}B\,ds$$

$$\hat H_2(t)=e^{-A^Tt}\int_0^t\int_0^s e^{A^Ts}Q_ce^{Ar}B\,dr\,ds$$

$$\hat K_1(t)=e^{-A^Tt}\int_0^t\int_0^s\int_0^r e^{A^Tr}Q_ce^{Aw}B\,dw\,dr\,ds.$$

It turns out that the integrals (1.1)–(1.4) can be expressed in terms of these submatrices of $e^{\hat Ct}$ when we set $t=\Delta$:

$$H(\Delta)=\hat G_3(\Delta) \tag{2.1}$$

$$Q(\Delta)=\hat F_3(\Delta)^T\hat G_2(\Delta) \tag{2.2}$$

$$M(\Delta)=\hat F_3(\Delta)^T\hat H_2(\Delta) \tag{2.3}$$

$$W(\Delta)=\left[B^T\hat F_3(\Delta)^T\hat K_1(\Delta)\right]+\left[B^T\hat F_3(\Delta)^T\hat K_1(\Delta)\right]^T. \tag{2.4}$$

Equations (2.1)–(2.3) follow directly from the definitions (1.1)–(1.3). To verify (2.4), set

$$F(r,w)=e^{A^Tr}Q_ce^{Aw}$$

and notice

$$\int_0^s\int_0^r\left[F(r,w)+F(w,r)\right]dw\,dr=\int_0^s\int_0^s F(r,w)\,dw\,dr.$$

Thus,

$$\begin{aligned}
&\left[B^T\hat F_3(\Delta)^T\hat K_1(\Delta)\right]+\left[B^T\hat F_3(\Delta)^T\hat K_1(\Delta)\right]^T\\
&\quad=\int_0^\Delta\int_0^s\int_0^r B^T\left[e^{A^Tr}Q_ce^{Aw}+e^{A^Tw}Q_ce^{Ar}\right]B\,dw\,dr\,ds\\
&\quad=\int_0^\Delta B^T\left[\int_0^s\int_0^r\left[F(r,w)+F(w,r)\right]dw\,dr\right]B\,ds\\
&\quad=\int_0^\Delta B^T\left[\int_0^s\int_0^s F(r,w)\,dw\,dr\right]B\,ds=W(\Delta).
\end{aligned}$$

## III. Formulation of the Algorithm

If (2.1)–(2.4) are used to compute the integrals (1.1)–(1.4) in practice, then we need a means for estimating $e^{\hat C\Delta}$. One possibility is to use estimates of the form

$$e^{\hat C\Delta}=\left[R_{qq}\left(\frac{\hat C\Delta}{2^j}\right)\right]^{2^j},\qquad q,j\geq 0 \tag{3.1}$$

where $R_{qq}(z)$ is the $(q,q)$ Padé approximant to $e^z$

$$R_{qq}(z)=\frac{\sum_{k=0}^{q}c_kz^k}{\sum_{k=0}^{q}c_k(-z)^k},\quad c_k=\frac{(2q-k)!q!}{(2q)!k!(q-k)!}.$$

The scaling by $2^j$ followed by the repeated squaring greatly enhances the numerical properties of ordinary Padé approximation [7], [9].

It is clear that the approximation in (3.1) has the form

$$\left[R_{qq}\left(\frac{\hat{C}\Delta}{2^j}\right)\right]^{2^j}=\begin{bmatrix} F_1(\Delta) & G_1(\Delta) & H_1(\Delta) & K_1(\Delta) \\ 0 & F_2(\Delta) & G_2(\Delta) & H_2(\Delta) \\ 0 & 0 & F_3(\Delta) & G_3(\Delta) \\ 0 & 0 & 0 & F_4(\Delta) \end{bmatrix}. \tag{3.2}$$

Thus, in accordance with (2.1)–(2.4), we obtain the following approximations to (1.1)–(1.4):

$$H(\Delta)\approx G_3(\Delta)$$
$$Q(\Delta)\approx F_3(\Delta)^T G_2(\Delta)$$
$$M(\Delta)\approx F_3(\Delta)^T H_2(\Delta)$$
$$W(\Delta)\approx\left[B^T F_3(\Delta)^T K_1(\Delta)\right]^T+\left[B^T F_3(\Delta)^T K_1(\Delta)\right].$$

This procedure is extremely easy to implement. All that is involved is a single call to any Padé matrix exponential subroutine followed by some elementary matrix computations. Ward's algorithm, with its complete error analysis, is particularly well suited [9]. For problems of small dimension, this is certainly a justifiable approach. However, in the interest of efficiency, the algorithm we shall detail does not repeatedly square the *matrix* $R_{qq}(\hat{C}\Delta/2^j)$ as suggested by (3.2). Instead, setting $t_0=\Delta/2^j$, we shall estimate $H(t_0)$, $Q(t_0)$, $M(t_0)$, and $W(t_0)$ using submatrices of $R_{qq}(\hat{C}t_0)$, and then repeatedly exploit the doubling formulas:

$$W(2t)=2W(t)+H(t)^T M(t)+M(t)^T H(t)+H(t)^T Q(t)H(t) \tag{3.3}$$
$$M(2t)=M(t)+e^{A^T t}\left[Q(t)H(t)+M(t)\right] \tag{3.4}$$
$$Q(2t)=Q(t)+e^{A^T t}Q(t)e^{At} \tag{3.5}$$
$$H(2t)=H(t)+e^{At}H(t) \tag{3.6}$$
$$e^{2At}=e^{At}e^{At}. \tag{3.7}$$

These formulas follow from definitions (1.1)–(1.4). (See [1] for details.) Summarizing, our algorithm is as follows:

*Algorithm*

1) Set

$$\hat{C}=\begin{bmatrix} -A^T & I & 0 & 0 \\ 0 & -A^T & Q_c & 0 \\ 0 & 0 & A & B \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

and let $j$ be the smallest nonnegative integer such that $\|\hat{C}\Delta\|/2^j\leqslant 1/2$. Set $t_0=\Delta/2^j$.

2) For some $q\geqslant 1$, compute

$$Y_0=R_{qq}\left(\frac{\hat{C}\Delta}{2^j}\right)$$

$$\equiv\begin{bmatrix} F_1(t_0) & G_1(t_0) & H_1(t_0) & K_1(t_0) \\ 0 & F_2(t_0) & G_2(t_0) & H_2(t_0) \\ 0 & 0 & F_3(t_0) & G_3(t_0) \\ 0 & 0 & 0 & F_4(t_0) \end{bmatrix}$$

and set

$$F_0=F_3(t_0)$$
$$H_0=G_3(t_0)$$
$$Q_0=F_3(t_0)^T G_2(t_0)$$
$$M_0=F_3(t_0)^T H_2(t_0)$$
$$W_0=\left[B^T F_3(t_0)^T K_1(t_0)\right]+\left[B^T F_3(t_0)^T K_1(t_0)\right]^T.$$

3) For $k=0,\cdots,j-1$

$$W_{k+1}=2W_k+H_k^T M_k+M_k^T H_k+H_k^T Q_k H_k$$
$$M_{k+1}=M_k+F_k^T\left[Q_k H_k+M_k\right]$$
$$Q_{k+1}=Q_k+F_k^T Q_k F_k$$
$$H_{k+1}=H_k+F_k H_k$$
$$F_{k+1}=F_k^2.$$

4) $F\equiv F_j$, $H\equiv H_j$, $Q\equiv Q_j$, $M\equiv M_j$, and $W\equiv W_j$ are then approximates to $e^{A\Delta}$, $H(\Delta)$, $Q(\Delta)$, $M(\Delta)$, and $W(\Delta)$, respectively.

Here, as everywhere in this paper, $\|\cdot\|$ denotes the Frobenius norm

$$Y=(y_{ij})\quad \|Y\|=\left[\sum_i\sum_j |y_{ij}|^2\right]^{1/2}.$$

Other norms are possible, but the Frobenius norm is convenient for both practical and theoretical reasons.

In the next section we will show

$$\|F-e^{A\Delta}\|\leqslant\epsilon\Delta\theta(\Delta)e^{\epsilon\Delta} \tag{3.8}$$
$$\|H-H(\Delta)\|\leqslant\epsilon\Delta\theta(\Delta)e^{\epsilon\Delta}\left[1+\frac{\alpha\Delta}{2}\right] \tag{3.9}$$
$$\|Q-Q(\Delta)\|\leqslant\epsilon\Delta\theta(\Delta)^2 e^{2\epsilon\Delta}\left[1+\alpha\Delta\right] \tag{3.10}$$
$$\|M-M(\Delta)\|\leqslant\epsilon\Delta\theta(\Delta)^2 e^{2\epsilon\Delta}\left[1+\epsilon+\alpha\Delta\right]^2 \tag{3.11}$$
$$\|W-W(\Delta)\|\leqslant\epsilon\Delta\theta(\Delta)^2 e^{2\epsilon\Delta}4\left[1+1.5(\alpha+\epsilon)\Delta\right]^3 \tag{3.12}$$

where

$$\epsilon=2^{3-2q}\|\hat{C}\|\frac{(q!)^2}{(2q)!(2q+1)!} \tag{3.13}$$
$$\theta(\Delta)=\max_{0\leqslant s\leqslant\Delta}\|e^{As}\| \tag{3.14}$$
$$\alpha=\max\{\|B\|,\|Q_c\|\}. \tag{3.15}$$

From (3.8)–(3.12), we see that the accuracy of the algorithm can be controlled through the selection of $q$. The choice of this integer and other computational details are discussed in Section V.

## IV. Truncation Error Analysis

To establish inequalities (3.8)–(3.12), we first characterize the errors which arise in Step 2 of the algorithm. We do this in Lemmas 1 and 2.

*Lemma 1:* If $Y_0$ is computed according to Step 2 of the algorithm, then

$$Y_0 = e^{(\hat{C}+\hat{E})t_0} \tag{4.1}$$

where

$$\hat{E} = \begin{bmatrix} -E_1^T & E_2 & E_5 & E_7 \\ 0 & -E_1^T & E_3 & E_6 \\ 0 & 0 & E_1 & E_4 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ (partitioned as } \hat{C}\text{).} \tag{4.2}$$

Furthermore,

$$AE_1 = E_1A \tag{4.3}$$

$$(A+E_1)^TE_2 = E_2(A+E_1)^T \tag{4.4}$$

$$\|E_i\| \leqslant \epsilon, \qquad i = 1, \cdots, 7. \tag{4.5}$$

*Proof:* Since $\|\hat{C}t_0\| \leqslant 1/2$, we have

$$R_{qq}(\hat{C}t_0) = e^{(\hat{C}+\hat{E})t_0}$$

$$\hat{C}\hat{E} = \hat{E}\hat{C}$$

$$\|t_0\hat{E}\| \leqslant 8\|\hat{C}t_0\|^{2q+1}\frac{(q!)^2}{(2q)!(2q+1)!} \tag{4.6}$$

by using [7, appendix 1, lemma 4]. By rearranging this last inequality and recalling that $\|\hat{C}t_0\| \leqslant 1/2$, we have

$$\|\hat{E}\| \leqslant \epsilon. \tag{4.7}$$

From this it is easy to establish (4.5), because the Frobenius norm of any submatrix of $\hat{E}$ is less than the Frobenius norm of $\hat{E}$. It is also clear from [7] that $\hat{E}$ has the same block structure as $\hat{C}$

$$\hat{E} = \begin{bmatrix} E_9 & E_2 & E_5 & E_7 \\ 0 & E_8 & E_3 & E_6 \\ 0 & 0 & E_1 & E_4 \\ 0 & 0 & 0 & E_{10} \end{bmatrix}.$$

Now by scrutinizing Steps 1 and 2 of the algorithm, it is clear that

$$F_1(t_0) = F_2(t_0) = R_{qq}(-A^Tt_0)$$

$$= \left[\left[R_{qq}(At_0)\right]^{-1}\right]^T = \left[F_3(t_0)^{-1}\right]^T$$

and

$$F_4(t_0) = R_{qq}(0) = I.$$

On the other hand, the equation $Y_0 = e^{(\hat{C}+\hat{E})t_0}$ coupled with Theorem 1 tells us that

$$F_1(t_0) = e^{(-A^T+E_9)t_0}$$

$$F_2(t_0) = e^{(-A^T+E_8)t_0}$$

$$F_3(t_0) = e^{(A+E_1)t_0}$$

$$F_4(t_0) = e^{(O+E_{10})t_0}$$

and, therefore, $E_8 = E_9 = -E_1^T$ and $E_{10} = 0$. Thus $\hat{E}$ has the structure defined by (4.2).

By equating the (3,3) and (1,2) blocks of (4.6), we see that (4.3) and $A^TE_2 = E_2A^T$ hold. This latter equality implies that $E_2$ commutes with $E_1^T$, since $E_1$ is a function of $A$ [7]. Thus (4.4) is verified, completing the proof of the lemma. ■

*Lemma 2:* If $F_3(t_0)$, $G_2(t_0)$, $G_3(t_0)$, $H_2(t_0)$, and $K_1(t_0)$ are defined by Step 2 of the algorithm, then

$$F_3(t_0) = e^{(A+E_1)t_0}$$

$$G_2(t_0) = e^{-(A+E_1)t_0}\int_0^{t_0} e^{(A+E_1)^Ts}(Q_c+E_3)e^{(A+E_1)s}\,ds$$

$$G_3(t_0) = \int_0^{t_0} e^{(A+E_1)s}(B+E_4)\,ds$$

$$H_2(t_0) = \int_0^{t_0} e^{-(A+E_1)^T(t_0-s)}E_6\,ds$$

$$+ \int_0^{t_0}\int_0^{s} e^{-(A+E_1)^T(t_0-s)}(Q_c+E_3)$$

$$\cdot e^{(A+E_1)(s-r)}(B+E_4)\,dr\,ds$$

$$K_1(t_0) = \int_0^{t_0} e^{-(A+E_1)^T(t_0-s)}E_7\,ds$$

$$+ \int_0^{t_0}\int_0^{s} e^{-(A+E_1)^T(t_0-s)}E_5e^{(A+E_1)(s-r)}(B+E_4)\,dr\,ds$$

$$+ \int_0^{t_0}\int_0^{s} (I+E_2)e^{-(A+E_1)^T(t_0-r)}E_6\,dr\,ds$$

$$+ \int_0^{t_0}\int_0^{s}\int_0^{r} (I+E_2)e^{-(A+E_1)^T(t_0-r)}(Q_c+E_3)$$

$$\cdot e^{(A+E_1)(r-w)}(B+E_4)\,dw\,dr\,ds.$$

*Proof:* From Lemma 1, $Y_0 = e^{(\hat{C}+\hat{E})t_0}$, where

$$(\hat{C}+\hat{E}) = \begin{bmatrix} -(A+E_1)^T & (I+E_2) & E_5 & E_7 \\ 0 & -(A+E_1)^T & (Q_c+E_3) & E_6 \\ 0 & 0 & (A+E_1) & (B+E_4) \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The lemma follows by applying Theorem 1 with $C = \hat{C}+\hat{E}$ and $t = t_0$. ■

We now examine how the errors in Step 2 propagate as the recursions of Step 3 are applied. To simplify the analysis, define $t_k$ by

$$t_k = \frac{\Delta}{2^{j-k}}, \qquad k = 0, 1, \cdots, j.$$

*Lemma 3:* If $F_k$, $H_k$, $Q_k$, and $M_k$ are defined by the algorithm, then

$$F_k = e^{(A+E_1)t_k}$$

$$H_k = \int_0^{t_k} e^{(A+E_1)s}(B+E_4)\,ds$$

$$Q_k = \int_0^{t_k} e^{(A+E_1)^T s}(Q_c+E_3)e^{(A+E_1)s}\,ds$$

$$M_k = \int_0^{t_k} e^{(A+E_1)^T s}E_6\,ds + \int_0^{t_k}\int_0^{s} e^{(A+E_1)^T s}(Q_c+E_3) \cdot e^{(A+E_1)(s-r)}(B+E_4)\,dr\,ds.$$

*Proof:* Using Lemma 2, it can be verified that these equations hold when $k=0$. Assume that they hold for some $k \geqslant 0$. By showing that they hold in the $(k+1)$st case, the Lemma will be proven by induction. To this end we see

$$F_{k+1} = F_k^2 = e^{(A+E_1)2t_k} = e^{(A+E_1)t_{k+1}}$$

$$H_{k+1} = H_k + F_k H_k = \int_0^{t_k} e^{(A+E_1)s}(B+E_4)\,ds + \int_0^{t_k} e^{(A+E_1)(t_k+s)}(B+E_4)\,ds = \int_0^{t_{k+1}} e^{(A+E_1)s}(B+E_4)\,ds.$$

The computations involving $Q_{k+1}$ and $M_{k+1}$ are similar, although more tedious. They are therefore deleted. ∎

We are now in a position to bound the truncation errors of the algorithm. Three inequalities simplify the analysis

$$\|e^{E_1 u}\| \leqslant e^{\epsilon u}, \qquad u \geqslant 0 \tag{4.8}$$

$$\|e^{(A+E_1)u} - e^{Au}\| \leqslant \epsilon u e^{\epsilon u}\|e^{Au}\|, \qquad u \geqslant 0 \tag{4.9}$$

$$\frac{\Delta}{2^j}\epsilon = t_0\epsilon \leqslant \frac{1}{6}. \tag{4.10}$$

Inequality (4.8) follows from (4.5) and the fact that $\|e^{E_1 u}\| \leqslant e^{\|E_1\|u}$. To establish (4.9), take norms in $e^{(A+E_1)u} - e^{Au} = e^{Au}(e^{E_1 u} - I)$, which follows from (4.3). Finally, (4.10) can be deduced from (3.13), and the inqualities $q \geqslant 1$ and $\|C\Delta\|/2^j \leqslant 1/2$.

*Theorem 2:* If $F$ is defined by the algorithm, then

$$\|F - e^{A\Delta}\| \leqslant \epsilon\Delta\theta(\Delta)e^{\epsilon\Delta}.$$

*Proof:* This follows from (4.9) with $u=\Delta$. ∎

*Theorem 3:* If $H$ is defined by the algorithm, then

$$\|H - H(\Delta)\| \leqslant \epsilon\Delta\theta(\Delta)e^{\epsilon\Delta}\left[1 + \frac{\alpha\Delta}{2}\right].$$

*Proof:* According to Lemma 3, $(k=j)$ and the definition of $H(\Delta)$

$$H - H(\Delta) = \int_0^{\Delta}\left[e^{(A+E_1)s} - e^{As}\right]B\,ds + \int_0^{\Delta} e^{(A+E_1)s}E_4\,ds.$$

The theorem follows by taking norms and applying (4.5), (4.8), and (4.9). ∎

*Theorem 4:* If $Q$ is defined by the algorithm, then

$$\|Q - Q(\Delta)\| \leqslant \epsilon\Delta\theta(\Delta)^2 e^{2\epsilon\Delta}\left[1 + \alpha\Delta\right].$$

*Proof:* From Lemma 3, $(k=j)$ and the definition of $Q(\Delta)$,

$$Q - Q(\Delta) = \int_0^{\Delta}\left[e^{(A+E_1)^T s}(Q_c+E_3)e^{(A+E_1)s} - e^{A^T s}Q_c e^{As}\right]ds$$

$$= \int_0^{\Delta}\left[e^{(A+E_1)s} - e^{As}\right]^T Q_c e^{(A+E_1)s} + e^{A^T s}Q_c\left[e^{(A+E_1)s} - e^{As}\right]ds + \int_0^{\Delta} e^{(A+E_1)^T s}E_3 e^{(A+E_1)s}\,ds.$$

The theorem follows by taking norms and using (4.5), (4.8), and (4.9). ∎

*Theorem 5:* If $M$ is defined by the algorithm, then

$$\|M - M(\Delta)\| \leqslant \epsilon\Delta\theta(\Delta)^2 e^{2\epsilon\Delta}\left[1 + \epsilon + \Delta\alpha\right]^2.$$

*Proof:* From Lemma 3, $(k=j)$ and the definition of $M(\Delta)$ we have

$$M - M(\Delta) = \int_0^{\Delta}\int_0^{s} e^{(A+E_1)^T s} \cdot (Q_c+E_3)e^{(A+E_1)(s-r)}(B+E_4)\,dr\,ds + \int_0^{\Delta} e^{(A+E_1)^T s}E_6\,ds - \int_0^{\Delta}\int_0^{s} e^{A^T s}Q_c e^{A(s-r)}B\,dr\,ds$$

$$= \int_0^{\Delta}\int_0^{s}\left[e^{(A+E_1)s} - e^{As}\right]^T Q_c e^{(A+E_1)(s-r)}B\,dr\,ds + \int_0^{\Delta}\int_0^{s} e^{A^T s}Q_c\left[e^{(A+E_1)(s-r)} - e^{A(s-r)}\right]B\,dr\,ds + \int_0^{\Delta}\int_0^{s} e^{(A+E_1)^T s}\left[Q_c+E_3\right]e^{(A+E_1)(s-r)}E_4\,dr\,ds + \int_0^{\Delta}\int_0^{s} e^{(A+E_1)^T s}E_3 e^{(A+E_1)(s-r)}B\,dr\,ds + \int_0^{\Delta} e^{(A+E_1)^T s}E_6\,ds.$$

The theorem follows by taking norms and invoking (4.5), (4.8), and (4.9). ∎

Bounding the error of $W_k$ is considerably more tedious. To simplify matters, we shall first prove the following lemma.

*Lemma 4:* If $W_0$ is defined by the algorithm, then

$$\|W_0 - W(t_0)\| \leqslant \epsilon t_0\theta(t_0)^2\left[\frac{\alpha^3}{2}\left(t_0^3 + t_0^2\right) + \alpha^2\left(t_0^2 + \frac{t_0}{6} + 1.44\right) + \alpha(1.6t_0 + 3)\right].$$

*Proof:* From (2.4), $(\Delta = t_0)$ and the definition of $W_0$ we have

$$W_0 - W(t_0) = B^T\left[F_3(t_0)^T K_1(t_0) - \hat{F}_3(t_0)^T \hat{K}_1(t_0)\right] + \left[F_3(t_0)^T K_1(t_0) - \hat{F}_3(t_0)^T \hat{K}_1(t_0)\right]^T B$$

and thus

$$\|W_0 - W(t_0)\| \leq 2\alpha \|F_3(t_0)^T K_1(t_0) - \hat{F}_3(t_0)^T \hat{K}_1(t_0)\|. \tag{4.11}$$

From Lemma 2,

$$F_3(t_0)^T K_1(t_0) = \epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4$$

where

$$\epsilon_1 = \int_0^{t_0} e^{(A+E_1)s} E_7 \, ds$$

$$\epsilon_2 = \int_0^{t_0}\int_0^{s} e^{(A+E_1)^T s} E_5 e^{(A+E_1)(s-r)}(B+E_4)\, dr\, ds$$

$$\epsilon_3 = (I+E_2)\int_0^{t_0}\int_0^{s} e^{(A+E_1)^T r} E_6 \, dr\, ds$$

$$\epsilon_4 = (I+E_2)\int_0^{t_0}\int_0^{s}\int_0^{r} e^{(A+E_1)^T r}(Q_c + E_3) \cdot e^{(A+E_1)(r-w)}(B+E_4)\, dw\, dr\, ds.$$

(Equation (4.4) is needed to establish the expressions for $\epsilon_3$ and $\epsilon_4$.) By taking norms and using (4.5), (4.8), and (4.10) we have

$$\|\epsilon_1\| \leq 1.2\epsilon t_0 \theta(t_0) \tag{4.12}$$

$$\|\epsilon_2\| \leq 0.72\epsilon(\alpha+\epsilon) t_0^2 \theta(t_0)^2 \tag{4.13}$$

$$\|\epsilon_3\| \leq 0.6\epsilon(1+\epsilon) t_0^2 \theta(t_0). \tag{4.14}$$

From the definition of $\hat{F}_3(t_0)$ and $\hat{K}_1(t_0)$ in Section II, we have

$$\begin{aligned}\epsilon_4 - \hat{F}_3(t_0)^T \hat{K}_1(t_0) &= (I+E_2)\int_0^{t_0}\int_0^{s}\int_0^{r} e^{(A+E_1)^T r}(Q_c+E_3) \cdot e^{(A+E_1)(r-w)}(B+E_4)\, dw\, dr\, ds \\ &\quad - \int_0^{t_0}\int_0^{s}\int_0^{r} e^{A^T r} Q_c e^{Aw} B \, dw\, dr\, ds \\ &= E_2\int_0^{t_0}\int_0^{s}\int_0^{r} e^{(A+E_1)^T r}(Q_c+E_3) \cdot e^{(A+E_1)(r-w)}(B+E_4)\, dw\, dr\, ds \\ &\quad + \int_0^{t_0}\int_0^{s}\int_0^{r} e^{(A+E_1)^T r} E_3 e^{(A+E_1)(r-w)}(B+E_4)\, dw\, dr\, ds \\ &\quad + \int_0^{t_0}\int_0^{s}\int_0^{r} e^{(A+E_1)^T r} Q_c e^{(A+E_1)(r-w)} E_4 \, dw\, dr\, ds \\ &\quad + \int_0^{t_0}\int_0^{s}\int_0^{r} \left[e^{(A+E_1)r} - e^{Ar}\right]^T Q_c e^{(A+E_1)(r-w)} B \, dw\, dr\, ds \\ &\quad + \int_0^{t_0}\int_0^{s}\int_0^{r} e^{A^T r} Q_c \left[e^{(A+E_1)(r-w)} - e^{A(r-w)}\right] B \, dw\, dr\, ds.\end{aligned}$$

By taking norms and using (4.5) and (4.8)–(4.10), we find

$$\|\epsilon_4 - \hat{F}_3(t_0)^T \hat{K}_1(t_0)\| \leq 0.25\epsilon\theta(t_0)^2 \cdot \left[t_0^4\alpha^2 + t_0^3\left[(\alpha+\epsilon)^2 + 2(\alpha+\epsilon)\right]\right].$$

The lemma follows from this result, (4.11)–(4.14), and the fact that

$$\|F_3(t_0)^T K_1(t_0) - \hat{F}_3(t_0)^T \hat{K}_1(t_0)\| \leq \|\epsilon_1\| + \|\epsilon_2\| + \|\epsilon_3\| + \|\epsilon_4 - \hat{F}_3(t_0)^T \hat{K}_1(t_0)\|.$$

*Theorem 6:* If $W$ is defined by the algorithm, then

$$\|W - W(\Delta)\| \leq \epsilon\theta(\Delta)^2 e^{2\epsilon\Delta} 4\left[1 + 1.5(\alpha+\epsilon)\Delta\right]^3.$$

*Proof:* Subtracting the doubling formula

$$W(t_{k+1}) = 2W(t_k) + H(t_k)^T M(t_k) + M(t_k)^T H(t_k) + H(t_k)^T Q(t_k) H(t_k)$$

from

$$W_{k+1} = 2W_k + H_k^T M_k + M_k^T H_k + H_k^T Q_k H_k$$

and taking norms gives

$$\begin{aligned}\|W_{k+1} - W(t_{k+1})\| &\leq 2\|W_k - W(t_k)\| \\ &\quad + 2\|H_k^T M_k - H(t_k)^T M(t_k)\| \\ &\quad + \|H_k^T Q_k H_k - H(t_k)^T Q(t_k) H(t_k)\|. \end{aligned} \tag{4.15}$$

By applying Lemma 1, Lemma 3, (4.8), and (4.9) the following bounds can be derived:

$$\|H_k^T M_k - H(t_k)^T M(t_k)\| \leq \epsilon\theta(t_{k+1})^2 \cdot e^{2\epsilon\Delta} t_k^2 (\alpha+\epsilon)\left[\frac{3}{2}(\alpha+\epsilon)t_k + 1\right]^2$$

and

$$\|H_k^T Q_k H_k - H(t_k)^T Q(t_k) H(t_k)\| \leq \epsilon\theta(t_{k+1})^2 \cdot e^{2\epsilon\Delta} t_k^3 (\alpha+\epsilon)^2\left[3 + t_k(\alpha+\epsilon)\right].$$

It is thus clear from (4.15) that

$$\|W_{k+1} - W(t_{k+1})\| \leq 2\|W_k - W(t_k)\| + \delta_k \tag{4.16}$$

where

$$\delta_k = \epsilon\theta(t_{k+1})^2 e^{2\epsilon\Delta} t_k^2 (\alpha+\epsilon)\left[3(\alpha+\epsilon)t_k + 2\right]^2.$$

A simple induction argument involving (4.16) shows

$$\|W - W(\Delta)\| = \|W_j - W(t_j)\| \leq 2^j \|W_0 - W(t_0)\| + \sum_{k=0}^{j-1} 2^{j-k-1}\delta_k. \tag{4.17}$$

By using elementary properties of geometric series and the fact that $t_k = 2^{k-j}\Delta$, it is easy to verify that

$$\sum_{k=0}^{j-1} 2^{j-k-1}\delta_k \leq \frac{1}{2}\epsilon\theta(\Delta)^2 \cdot e^{2\epsilon\Delta}\Delta^2(\alpha+\epsilon)\left[3(\alpha+\epsilon)\Delta+2\right]^2. \quad (4.18)$$

Now $\|Ct_0\| \leq 0.5$ implies $\alpha t_0 \leq 0.5$ and, since $\theta(t_0) \leq \theta(\Delta)$, we have from Lemma 4

$$2^j \| W_0 - W(t_0)\| \leq 2\epsilon\Delta\theta(\Delta)^2(1+\alpha)^2.$$

The theorem follows by substituting this result together with (4.18) into (4.17). ■

## V. The Subroutine Padé

The method we have presented and analyzed has been implemented in a Fortran subroutine called Padé. In this final section we discuss some of the computational aspects of this program.

In the course of forming $Y_0 = R_{qq}(\hat{C}\Delta/2^j)$, we must compute the matrices

$$\hat{N} = \sum_{k=0}^{q} c_k \left(\frac{\Delta}{2^j}\right)^k \hat{C}^k$$

$$\hat{D} = \sum_{k=0}^{q} c_k \left(\frac{-\Delta}{2^j}\right)^k \hat{C}^k, \qquad (Y_0 = \hat{D}^{-1}\hat{N}).$$

Hence an efficient and compact means of powering the matrix $\hat{C}$ is desirable. It can be shown that for $k \geq 1$

$$\hat{C}^k = \begin{bmatrix} -A^T & I & 0 & 0 \\ 0 & -A^T & Q_c & 0 \\ 0 & 0 & A & B \\ 0 & 0 & 0 & 0 \end{bmatrix}^k = \begin{bmatrix} (-1)^k X_k^T & (-1)^{k-1} X_{k-1}^T & Z_k & Y_k \\ 0 & (-1)^k X_k^T & R_k & V_k \\ 0 & 0 & X_k & D_k \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

where $X_1 = A$, $D_1 = B$, $R_1 = Q_c$, $Z_1 = 0$, $Y_1 = 0$, $V_1 = 0$, and for $k \geq 2$

$$\begin{aligned} Y_k &= -A^T Y_{k-1} + V_{k-1} \\ Z_k &= -A^T Z_{k-1} + R_{k-1} \\ V_k &= -A^T V_{k-1} + Q_c D_{k-1} \\ R_k &= -A^T R_{k-1} + Q_c X_{k-1} \\ D_k &= X_{k-1} B \\ X_k &= A X_{k-1}. \end{aligned}$$

Thus, as $\hat{N}$ and $\hat{D}$ are "built up," we need $n \times n$ arrays $X$, $R$, and $Z$ to store the matrices $X_k$, $R_k$, and $Z_k$, and $n \times p$ arrays $D$, $V$, and $Y$ to store the matrices $D_k$, $V_k$, and $Y_k$.

Next, we show that only a portion of $\hat{N}$ and $\hat{D}$ is actually required by the algorithm. Partition these two matrices as $\hat{C}$

$$\hat{N} = \begin{bmatrix} N_{11} & N_{1i} & N_{13} & N_{14} \\ 0 & N_{22} & N_{23} & N_{24} \\ 0 & 0 & N_{33} & N_{34} \\ 0 & 0 & 0 & I \end{bmatrix}$$

$$\hat{D} = \begin{bmatrix} D_{11} & D_{12} & D_{13} & D_{14} \\ 0 & D_{22} & D_{23} & D_{24} \\ 0 & 0 & D_{33} & D_{34} \\ 0 & 0 & 0 & I \end{bmatrix}.$$

Referring to Step 2 in the algorithm, we must solve for $F_3(t_0)$, $G_3(t_0)$, $G_2(t_0)$, $H_2(t_0)$, and $K_1(t_0)$ in

$$\begin{bmatrix} D_{11} & D_{12} & D_{13} & D_{14} \\ 0 & D_{22} & D_{23} & D_{24} \\ 0 & 0 & D_{33} & D_{34} \\ 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} F_1(t_0) & G_1(t_0) & H_1(t_0) & K_1(t_0) \\ 0 & F_2(t_0) & G_2(t_0) & H_2(t_0) \\ 0 & 0 & F_3(t_0) & G_3(t_0) \\ 0 & 0 & 0 & F_4(t_0) \end{bmatrix} = \begin{bmatrix} N_{11} & N_{12} & N_{13} & N_{14} \\ 0 & N_{22} & N_{23} & N_{24} \\ 0 & 0 & N_{33} & N_{34} \\ 0 & 0 & 0 & I \end{bmatrix}.$$

Noting that $D_{11} = D_{22} = \sum_{k=0}^{q} c_k(-1)^k(-A^T)^k = [\sum_{k=0}^{q} c_k A^k]^T = N_{33}^T$, the working equations for $F_3(t_0)$, $G_3(t_0)$, $G_2(t_0)$, $H_2(t_0)$, and $K_1(t_0)$ thus become

$$\begin{aligned} D_{33}F_3(t_0) &= N_{33} \\ D_{33}G_3(t_0) &= N_{34} - D_{34} \\ N_{33}^T G_2(t_0) &= N_{23} - D_{23}F_3(t_0) \\ N_{33}^T H_2(t_0) &= N_{24} - D_{23}G_3(t_0) - D_{24} \\ N_{33}^T K_1(t_0) &= N_{14} - D_{12}H_2(t_0) - D_{13}G_3(t_0) - D_{14}. \end{aligned}$$

These linear systems can be solved using Gaussian elimination. Since $\|\hat{C}\Delta/2^j\| < 1/2$, it is easy to show that both $D_{33}$ and $N_{33}^T$ are diagonally dominant and, therefore, that no pivoting is necessary [2, p. 152]. Arrays are needed

TABLE I

| CODE | Matrices Computed | Approximate Storage Required | Magnitude of Work (Multiplicative Operations) |
|---|---|---|---|
| 1 | $F$ | $4n^2$ | $[q+j+\frac{1}{3}]n^3$ |
| 2 | $F,H$ | $4n^2+4np$ | $[q+j+\frac{1}{3}]n^3+[q+j]n^2p$ |
| 3 | $F,Q$ | $8n^2$ | $[3q+\frac{5}{2}j+\frac{2}{3}]n^3$ |
| 4 | $F,H,Q,M$ | $8n^2+7np$ | $[3q+\frac{5}{2}j+\frac{2}{3}]n^3+[3q+3j]n^2p$ |
| 5 | $F,H,Q,M,W$ | $11n^2+10np$ | $[4q+\frac{5}{2}j-\frac{5}{6}]n^3+[4q+\frac{11}{2}j+2]n^2p$ |

to form the submatrices $D_{33}$, $D_{34}$, $D_{23}$, $D_{24}$, $D_{12}$, $D_{13}$, $D_{14}$, $N_{33}$, $N_{34}$, $N_{23}$, $N_{24}$, and $N_{14}$. After that, they can be overwritten as the just mentioned linear systems are solved.

The implementation of the doubling formulas is straightforward and does not require any special commentary. Suffice it to say that no additional storage is necessary to execute that portion of the program.

Regarding storage and efficiency, it may be that not all of the matrices $F$, $H$, $Q$, $M$, and $W$ are desired. For example, suppose that $W$ is not wanted. We can effectively compute $F$, $H$, $Q$, and $M$ by working with

$$\begin{bmatrix} -A^T & Q_c & 0 \\ 0 & A & B \\ 0 & 0 & 0 \end{bmatrix}$$

instead of $\hat{C}$. We merely ignore all the computations that are specific to the construction of $W$. The truncation error bounds (3.8)–(3.11) still hold. Similar techniques exist if only $Q$, $H$, or $F$ are wanted.

Through an integer variable CODE, the user can specify certain subsets of the matrices $F$, $H$, $Q$, $M$, and $W$ which are to be computed. This allows for a saving in both storage and execution time, as Table I indicates.

The volume of computation is seen to depend upon the scale parameter $j$ and the degree $q$ of the Padé approximant which is used. The selection of $j$ was described in Section III. The integer $q$ is chosen in accordance with a user specified tolerance TOL. From Theorems 2–6, we know that if TOL$>0$, then $q$ can be picked so

$$\|F-F(\Delta)\| \leqslant \epsilon\Delta e^{\epsilon\Delta}\theta(\Delta) \leqslant \text{TOL}\,\theta(\Delta) \tag{5.1}$$

$$\|H-H(\Delta)\| \leqslant \epsilon\Delta e^{\epsilon\Delta}\left[1+\frac{\alpha\Delta}{2}\right]\theta(\Delta) \leqslant \text{TOL}\,\theta(\Delta) \tag{5.2}$$

$$\|Q-Q(\Delta)\| \leqslant \epsilon\Delta e^{2\epsilon\Delta}[1+\alpha\Delta]\theta(\Delta)^2 \leqslant \text{TOL}\,\theta(\Delta)^2 \tag{5.3}$$

$$\|M-M(\Delta)\| \leqslant \epsilon\Delta e^{2\epsilon\Delta}[1+\epsilon+\alpha\Delta]^2\theta(\Delta)^2 \leqslant \text{TOL}\,\theta(\Delta)^2 \tag{5.4}$$

$$\|W-W(\Delta)\| \leqslant 4\epsilon\Delta e^{2\epsilon\Delta}\left[1+\frac{3}{2}(\alpha+\epsilon)\Delta\right]^3\theta(\Delta)^2 \leqslant \text{TOL}\,\theta(\Delta)^2. \tag{5.5}$$

(The definitions of $\epsilon$, $\theta(\Delta)$, and $\alpha$ are given in (3.13)–(3.15)). In PADE, $q$ is chosen to be the smallest integer so that the appropriate inequalities just mentioned are satisfied. For example, if CODE$=3$, then $H$, $M$, and $W$ are not required, and so $q$ is chosen so only (5.1) and (5.3) hold.

An upper bound is placed on the size of $q$, which depends upon the precision of the machine used. This bound, called QMAX, is the largest value of $q$ for which

$$c_q\left\|\frac{C\Delta}{2^j}\right\|^q \leqslant \frac{q!}{(2q)!}\frac{1}{2^q} \leqslant \beta^{-t}$$

where the mantissas in the floating-point number system used consist of $t$, base $\beta$ digits. When IBM 370 double precision arithmetic is used ($\beta=16$, $t=14$), QMAX$=10$. There is no point in selecting a "$q$" larger than QMAX, for then the truncation errors we are trying to control through $q$ are dominated by rounding errors.

To obtain absolute error bounds, we see from (5.1)–(5.5) that a bound must be obtained for $\theta(\Delta)$. There are many ways that the norm of a matrix exponential can be bounded [8]. The crudest of these

$$\|e^{Au}\| \leqslant e^{\|A\|u}$$

implies $\theta(\Delta) \leqslant e^{\|A\|\Delta}$. However, in the interest of a more realistic estimate of $\theta(\Delta)$, PADE returns the easily computed quantity

$$\text{THETA} = \max_{0\leqslant k\leqslant j}\left\|\left(R_{qq}\left[\frac{A\Delta}{2^j}\right]\right)^{2^k}\right\|.$$

In practice, $\theta(\Delta)$ is usually a small number and therefore knowledge of its exact value is not too critical. Furthermore, one is often interested in relative error bounds, and in this sense TOL is a fair measure. (If $\theta(\Delta)$ is large, then normally the same can be said of $H(\Delta)$, $Q(\Delta)$, $M(\Delta)$, and $W(\Delta)$.)

To give the reader a feel for the speed of PADE, we mention that it took 5 s of IBM 370/168 CPU time to compute $F$, $H$, $Q$, $M$, and $W$ in a problem of dimension $n=30$ and $p=15$. In this test problem, TOL$=10^{-3}$, and the resulting parameters of the approximation were $q=7$ and $j=9$.

As a test of accuracy, PADE was used to compute $F$, $H$, $Q$, $M$, and $W$ when

$$A=\begin{bmatrix} 2 & -8 & -6 \\ 10 & -19 & -12 \\ -10 & 15 & 8 \end{bmatrix} \quad B=\begin{bmatrix} 5 & 1 \\ 1 & 4 \\ 3 & 2 \end{bmatrix} \quad Q_c=\begin{bmatrix} 4 & 1 & 2 \\ 1 & 3 & 1 \\ 2 & 1 & 5 \end{bmatrix}.$$

Because the eigensystem of $A$ is known

$$A=XDX^{-1}, \quad D=\begin{bmatrix} -2 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & -4 \end{bmatrix}, \quad X=\begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 6 \\ -2 & -5 & -5 \end{bmatrix}.$$

$F(\Delta)$, $H(\Delta)$, $Q(\Delta)$, $M(\Delta)$, and $W(\Delta)$ can be computed analytically and are to 10 significant digits:

$$F(\Delta)=\begin{bmatrix} 0.477528143 & -0.522155363 & -0.351058933 \\ 0.855482148 & -0.994523657 & -0.702117866 \\ -0.855482148 & 1.012839296 & 0.720433505 \end{bmatrix}$$

$$H(\Delta)=\begin{bmatrix} 1.999431436 & -3.394449325 \\ 1.148224072 & -6.155423359 \\ -0.166539711 & 7.627949901 \end{bmatrix}$$

$$Q(\Delta)=\begin{bmatrix} 9.934877720 & -11.08568953 & -9.123023900 \\ -11.08568953 & 13.66870748 & 11.50451512 \\ -9.123023900 & 11.50451512 & 10.29179555 \end{bmatrix}$$

$$M(\Delta)=\begin{bmatrix} 3.515982340 & -24.87596341 \\ -2.516164470 & 30.94693518 \\ -1.194242580 & 24.29316617 \end{bmatrix}$$

$$W(\Delta)=\begin{bmatrix} 12.29648659 & -5.373425530 \\ -5.373425530 & 105.9996704 \end{bmatrix}.$$

With TOL $=10^{-3}$, the computed versions of these matrices were found to be correct through the sixth decimal place. (In this example, $q=4, j=7$, and THETA $=4.2$.)

We do not expect the accuracy of our computed matrices to undercut the value of TOL by such amounts in all problems. Indeed, one must be wary of rounding errors which have not been accounted for in our analysis. However, as experience with Ward's Padé scaling and squaring algorithm for matrix exponentials suggests, we can be fairly confident of our error bounds so long as TOL is not in the immediate neighborhood of the machine precision.

## VI. Conclusions

We conclude by contrasting our algorithm with some of the other techniques that have been suggested for computing the various integrals (1.1)–(1.4).

Johnson and Phillips [4] have proposed the computation of $H(\Delta)$ through the formula

$$H(\Delta)=\left[\sum_{k=0}^{m-1}(e^{At})^k\right]H(t), \qquad mt=\Delta$$

the idea being that for small $t$, $e^{At}$ and $H(t)$ can be accurately computed. (Their discussion assumes $B=I$, but it is easy to extend their results for general $B$.) However, if $m=2^j$, their algorithm requires about $2^j[n^3+n^2p]$ operations to compute $H(\Delta)$ from $H(t)$ in contrast to our algorithm where the corresponding figure is only $j[n^3+n^2p]$.

In search for an efficient squaring algorithm, Kallstrom [5] has proposed repeated application of

$$H(2t)=H(t)\left[2I+AH(t)\right].$$

Unfortunately, this formula only holds if $B$ is the identity and, therefore, one has to compute $H(\Delta)$ by applying Kallstrom's formula to the problem

$$\tilde{H}(\Delta)=\int_0^\Delta e^{As}\,ds$$

and then forming $H(\Delta)=\tilde{H}(\Delta)B$. This increases the volume of computation over our algorithm by an amount proportional to $n/P$. We cannot be more precise because the parameters which define Kallstrom's algorithm are empirically determined, making work counts difficult. Furthermore, there are no rigorous bounds on the errors as they propagate during the repeated use of the previously given squaring formula.

By using quadrature rules, another approach to the estimation (1.1)–(1.4) can be derived. In particular, Levis has proposed using Simpson's rule for the computation of $Q(\Delta)$ [6]. This involves the computation of

$$(e^{A^Th})^kQ_c(e^{Ah})^k, \qquad k=0,\cdots,N \quad \text{(even)}$$

where $Nh=\Delta$. Because the error bound for this method involves a term of the form

$$\frac{(\Delta\|A\|)^4}{N^4}$$

it appears that for a given accuracy tolerance, much more computation is required of this method than ours. (The interested reader should compare [6, (19)] with (3.10).)

Armstrong and Caglayan [1] have proposed a Taylor series approach involving term-by-term integration of the series for $H(t_0)$, $Q(t_0)$, $M(t_0)$, and $W(t_0)$ followed by repeated application of the doubling formulas (3.3)–(3.7). It is hard to compare our algorithm with theirs, for although they give error bounds for the computed versions of $H(t_0)$, $Q(t_0)$, $M(t_0)$, and $W(t_0)$, they do not analyze how these errors propagate during the repeated doubling. Nevertheless, we suspect that their algorithm is quite efficient. Compared to ours, it requires about a third less storage. However, it is known that scaling and squaring algorithms with diagonal Padé approximants require

about half the work as their Taylor series counterparts for a given accuracy [7]. Thus we would guess that our algorithm would be quicker. Of course, a detailed computational study should be done to ascertain this. We leave this as a topic for future investigation.

## ACKNOWLEDGMENT

The author wishes to thank E. S. Armstrong for describing the problem discussed in this paper. He also wishes to thank The Institute of Computer Applications in Science and Engineering, Hampton, VA, for making those discussions possible.

## REFERENCES

[1] E. S. Armstrong and A. K. Caglayan, "An algorithm for the weighting matrices in the sampled-data optimal linear regulator problem," NASA Langley Res. Ctr., Hampton, VA, NASA Tech. Note NASA TN D-8372, 1976.
[2] G. Dahlquist and A. Bjork, *Numerical Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1974.
[3] P. Dorato and A. Levis, "Optimal linear regulators: The discrete time case," *IEEE Trans. Automat. Contr*, vol. AC-16, pp. 613–620, Dec. 1971.
[4] J. C. Johnson and C. L. Phillips, "An algorithm for the computation of the integral of the state transition matrix," *IEEE Trans. Automat. Contr.*, vol. AC-16, pp. 204–205, 1971.
[5] C. Kallstrom, "Computing Exp $(A)$ and $\int \text{Exp}(As)ds$," Div. of Automat. Contr., Lund Inst. Technol., Lund, Sweden, Rep. 7309, 1973.
[6] A. H. Levis, "Some computational aspects of the matrix exponential," *IEEE Trans. Automat. Contr.*, vol. AC-14, pp. 410–411, 1969.
[7] C. B. Moler and C. Van Loan, "Nineteen dubious ways to compute the exponential of a matrix," *SIAM Rev.*, to be published.
[8] C. Van Loan, "The sensitivity of the matrix exponential," *SIAM J. Numer. Analysis*, vol. 14, pp. 971–981, 1977.
[9] R. C. Ward, "Numerical computation of the matrix exponential with accuracy estimate," *SIAM J. Numer. Analysis*, vol. 14, pp. 600–610, 1977.

**Charles F. Van Loan** was born in Orange, NJ, in 1947. He received the B.S. degree, the M.A. degree, and the Ph.D degree, all in mathematics, and all from the University of Michigan, Ann Arbor, in 1969, 1970, and 1973, respectively.

In 1974 and 1975 he was a Postdoctoral Research Fellow in the Department of Mathematics at Manchester University, Manchester, England. Since 1975, he has been an Assistant Professor of Computer Science at Cornell University, Ithica, NY. His research interests include numerical analysis, matrix computations, and the history of computing.

Dr. Van Loan is a member of SIAM.