A Generalized Horner Algorithm for the Computation of Integrals Involving the Matrix Exponential

Charles Van Loan

Department of Computer Science, Cornell University
Ithaca, New York 14853

One way of computing the weighting matrices in the optimal regulator problem is to exponentiate a certain block upper triangular matrix. A complete analysis of this technique was given in an earlier paper by the author when diagonal Padé approximants are used to estimate the necessary matrix exponential. A new method for evaluating matrix polynomials suggests that Taylor approximation may be preferable for reasons of time and storage.

In [3] we gave a new algorithm for computing the integrals

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

$$Q(\Delta) = \int_0^\Delta e^{A^T s}\, Q_c e^{As}\, ds$$

$$M(\Delta) = \int_0^\Delta e^{A^T s}\, Q_c H(s)\, ds$$

$$W(\Delta) = \int_0^\Delta H(s)^T\, Q_c H(s)\, ds$$

where A, B, and Q_c have dimension nxn, nxp, and nxn respectively and, in addition, Q_c is symmetric positive semi-definite. The technique is based on the fact that if

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

(the column widths are n, n, n, p)

and

$$e^{Ct} = \begin{bmatrix} \hat{F}_1(t) & \hat{G}_1(t) & \hat{H}_1(t) & \hat{K}_1(t) \\ 0 & \hat{F}_2(t) & \hat{G}_2(t) & \hat{H}_2(t) \\ 0 & 0 & \hat{F}_3(t) & \hat{G}_3(t) \\ 0 & 0 & 0 & \hat{F}_4(t) \end{bmatrix}$$

then

$$H(t) = \hat{G}_3(t)$$

$$Q(t) = \hat{F}_3(t)^T \hat{G}_2(t)$$

$$M(t) = \hat{F}_3(t)^T \hat{H}_2(t)$$

$$W(t) = [B^T \hat{F}_3(t)^T \hat{K}_3(t)] + [B^T \hat{F}_3(t)^T \hat{K}_1(t)]^T$$

If $t = \Delta/2^j$ $(j \geq 0)$, then the integrals $H(\Delta)$, $Q(\Delta)$, $M(\Delta)$, and $W(\Delta)$ can be obtained through repeated application of the doubling formulae

$$W(2t) = 2W(t) + H(t)^T M(t) + M(t)^T H(t) + H(t)^T Q(t) H(t)$$

$$M(2t) = M(t) + e^{A^T t}[Q(t)H(t) + M(t)]$$

$$Q(2t) = Q(t) + e^{A^T t}\, Q(t)\, e^{At}$$

$$H(2t) = H(t) + e^{At}\, H(t)$$

$$e^{2At} = e^{At}\, e^{At}$$

To formulate an algorithm based upon these relationships, it is necessary to estimate e^{Ct}. For this purpose, it is well known [2] that Padé approximation works well provided $\|Ct\|$ is small enough. Here, as elsewhere in this article, we use the Frobenius norm:

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

By "small enough" we mean that $j \geq 0$ is chosen such that

$$\| Ct \| = \| C \| \Delta / 2^j \leq 1/2$$

Denoting the (p,q) Padé approximant to e^z by $R_{pq}(z)$ where

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

and

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

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

we obtain the following

Algorithm 1

1) Let j be the smallest non-negative integer for which

$$\| C \| \Delta / 2^j \leq 1/2$$

and set $t_0 = \Delta / 2^j$.

2) Choose p and q (see below) and compute

$$R_{pq}(Ct_0) = \begin{bmatrix} F_1(t_0) & G_1(t_0) & H_1(t_0) & K_1(t_0) \\ 0 & F_2(t_0) & G_2(t_0) & H_2(t_0) \\ 0 & 0 & F_3(t_0) & G_3(t_0) \\ 0 & 0 & 0 & F_4(t_0) \end{bmatrix}$$

3) Set

$$F_0 = F_3(t_0)$$

$$H_0 = G_3(t_0)$$

$$Q_0 = F_3(t_0)^T G_2(t_0)$$

$$M_0 = F_3(t_0)^T H_2(t_0)$$

$$W_0 = [B^T F_3(t_0)^T K_1(t_0)] + [B^T F_3(t_0)^T K_1(t_0)]^T$$

and for k=0,1,...,j-1 compute

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

$$M_{k+1} = F_k^T [Q_k H_k + M_k]$$

$$Q_{k+1} = Q_k + F_k^T Q_k F_k$$

$$H_{k+1} = H_k + F_k H_k$$

$$F_{k+1} = F_k F_k$$

4) $F = F_j$, $H = H_j$, $Q = Q_j$, $M = M_j$, $W = W_j$ are then the respective approximations to $e^{A\Delta}$, $H(\Delta)$, $Q(\Delta)$, $M(\Delta)$ and $W(\Delta)$.

The truncation errors associated with these approximations were bounded in [3] for the case p=q ; i.e. diagonal Padé approximation. The key to the analysis was the following result from [2]:

$$R_{pq}(Y) = e^{Y+E}$$

where

$$\| E \| \leq 8 \| Y \|^{p+q+1} \frac{p!q!}{(p+q)!(p+q+1)!}$$

under the assumption that $\| Y \| \leq 1/2$.

This "inverse error analysis" can be used in an identical fashion to establish the following inqualities for tne case when general Padé approximants are used in the above algorithm:

$$\| F - e^{A\Delta} \| \leq \varepsilon \Delta \Theta(\Delta) e^{\varepsilon\Delta}$$

$$\| H - H(\Delta) \| \leq \varepsilon \Delta \Theta(\Delta) e^{\varepsilon\Delta} [1 + \alpha\Delta/2]$$

$$\| Q - Q(\Delta) \| \leq \varepsilon \Delta \Theta(\Delta)^2 e^{2\varepsilon\Delta} [1 + \alpha\Delta]$$

$$\| M - M(\Delta) \| \leq \varepsilon \Delta \Theta(\Delta)^2 e^{2\varepsilon\Delta} [1 + \varepsilon + \alpha\Delta]^2$$

$$\| W - W(\Delta) \| \leq \varepsilon \Delta \Theta(\Delta)^2 e^{2\varepsilon\Delta} [2 + 3(\alpha+\varepsilon)\Delta]^3$$

where

$$\varepsilon = 2^{3-(p+q)} \| C \| \frac{p!q!}{(p+q)!(p+q+1)!}$$

$$\theta(\Delta) = \max_{0 \leq s \leq \Delta} \| e^{As} \|$$

$$\alpha = \max\{ \| B \|, \| Q_c \| \}$$

Methods for estimating the factor $\theta(\Delta)$ are given in [3] and the net result is that the parameters p and q can be chosen such that the above error bounds are less than some prescribed tolerance.

The reason that we only considered diagonal Padé approximants in [3] is because for a fixed amount of work, they give greater accuracy than the off-diagonal approximants. (For the precise meaning of this statement, see [2].) However, a new method for evaluating matrix polynomials has led us to conclude that Taylor approximation (i.e. $q = 0$) is preferable for the problem at hand.

The matrix polynomial algorithm to which we are referring is discussed in detail in [4] and amounts to a generalization of Horner's rule. To illustrate the key idea in a concrete setting, suppose we want to compute

$$p(Z) = \sum_{k=0}^{16} c_k Z^k$$

where Z is an $n \times n$ matrix. This can be accomplished in $15n^3$ operations and $2n^2$ storage using Horner's rule:

$S := c_{16} Z + c_{15} I$

For $k = 14,13,\ldots,1,0$

  $S := ZS + c_k I$

whereupon $S = p(Z)$. However, an algebraic manipulation shows that

$$p(Z) = c_{16}(Z^4)^4 + \sum_{k=0}^{3} (Z^4)^k [c_{4k+3} Z^3 + c_{4k+2} Z^2 + c_{4k+1} Z + c_{4k} I]$$

Therefore, once Z^2, Z^3, and Z^4 are are known, $p(Z)$ can be computed as follows:

$S := c_{16} Z^4 + c_{15} Z^3 + c_{14} Z^2 + c_{13} Z + c_{12} I$

For $k=2,1,0$

  $S := (Z^4)S + c_{4k+3} Z^3 + c_{4k+2} Z^2 + c_{4k+1} Z + c_{4k} I$

Overall, we see that $6n^3$ operations are required in contrast to the count of $15n^3$ associated with Horner's rule.

In it's general form, this approach to matrix polynomial evaluation leads to an algorithm requiring $O[\sqrt{d}\, n^3]$ operations where d is the degree of the polynomial and n is the dimension of the matrix. (See [4] for precise workcounts.) Obviously, the bigger d is, the greater the improvement over the traditional Horner approach which requires $(d-1)n^3$ operations. However, in its raw form, the generalized Horner scheme requires a lot of storage. For instance, in the $d = 16$ example above, arrays are required for S, A, A^2, A^3, and A^4. In general, for matrix polynomials of degree d, $O[\sqrt{d}\, n^2]$ storage is needed - a serious drawback.

Fortunately, there is a way to arrange the computation so that this storage requirement is greatly reduced. The idea is to compute $p(Z)$ "column-at-a-time" and once again we resort to the $d = 16$ case for purposes of illustration. Letting e_j denote the j-th column of the identity, we have

Algorithm 2

$Y := Z^2$; $Y := Y^2$

For $j = 1,\ldots,n$

  $y_0 := e_j$; $y_1 := Zy_0$;

  $y_2 := Zy_1$; $y_3 := Zy_2$; $s = c_{16} e_j$

  For $k = 3,2,1,0$

    $s := Ys + \sum_{i=0}^{3} c_{4k+i} y_i$

  $f_j := s$

whereupon

$$p(Z) = \sum_{i=0}^{16} c_k Z^k = [f_1|f_2|....|f_n]$$

A careful (multiplicative) operation count reveals that p(Z) requires $7n^3$ operations. (Products like Ze_j and Ye_j are "free".) Moreover, only three nxn arrays are needed : for Z, Z^4 , and p(Z) .

Now these may seem like small inconsequential savings. However, in our problem, Z = C , a (3n+p)x(3n+p) matrix. and so reductions in work and storage predicted by these little shortcuts has a magnified effect. In the remainder of this paper, we shall discuss the specialization of Algorithm 2 to the problem of evaluating $R_{pq}(Ct_0)$ as required in step 3 of Algorithm 1. The specialization exploits the block structure of C and the fact that only certain submatrices of $R_{pq}(Ct_0)$ are needed to compute the integrals we're after. Moreover, we shall restrict ourselves to the case p=16 , q=0, i.e., Taylor approximation of degree 16. We do this because (a) the value of ε above is then - about $10^{-18}\|C\|$ which is adequate for most purposes and (b) more economies can be squeezed from Algorithm 2 if we apply it to $R_{16,0}(Ct_0)$ rather than the numerator and denominator matrices of the roughly equivalent approximants $R_{p,16-p}(Ct_0)$ $(1 \le p \le 15)$.

To this end we first realize that for $k \ge 1$, the matrix $(Ct_o)^{2^k}$ has the form:

$$(Ct_0)^{2^k} = \begin{bmatrix} X_k^T & U_k & Z_k & Y_k \\ O & X_k^T & R_k & V_k \\ O & O & X_k & D_k \\ O & O & O & O \end{bmatrix} \begin{matrix} \} n \\ \} n \\ \} n \\ \} p \end{matrix}$$

$$n \quad n \quad n \quad p$$

Since we are interested in the application of Algorithm 2 to the matrix polynomial $R_{16,0}(Ct_0)$, we compute $(Ct_0)^4$ as follows:

$$Q_c := t_0 Q_c \; ; \; A := t_0 A \; ; \; B := t_0 B$$

$$Y_1 := 0 \; ; \; Z_1 := Q_c \; ; \; U_1 := -2A^T$$

$$V_1 := Q_c B$$

$$R_1 := -A^T Q_c + (A^T Q_c)^T$$

$$D_1 := AB$$

$$X_1 := A^2$$

$$Y_2 := U_1 V_1 + Z_1 D_1$$

$$Z_2 := X_1^T Z_1 + U_1 R_1 + (X_1^T Z_1)^T$$

$$U_2 := 2U_1 X_1^T$$

$$V_2 := X_1 V_1 + R_1 D_1$$

$$R_2 := (X_1^T R_1) - (X_1^T R_1)^T$$

$$D_2 := X_1 D_1$$

$$X_2 := X_1 X_1$$

Now we continue in Algorithm 2 for the purpose of computing selected portions of $R_{16,0}(Ct_0)$. For example, the last p columns of this matrix are given by

$$\begin{bmatrix} K_1(t_0) \\ H_2(t_0) \\ G_3(t_0) \\ I \end{bmatrix} \begin{matrix} \} n \\ \} n \\ \} n \\ \} p \end{matrix}$$

The top three submatrices are involved in the computation of $H(t_0)$, $M(t_0)$, and $W(t_0)$ and may be found as follows:

For j = 1,...,p

$$y_0 := \begin{bmatrix} 0 \\ 0 \\ 0 \\ e_j \end{bmatrix} \quad y_1 := Cy_0 = \begin{bmatrix} 0 \\ 0 \\ Be_j \\ 0 \end{bmatrix}$$

$$y_2 := Cy_1 = \begin{bmatrix} 0 \\ Q_c(Be_j) \\ A(Be_j) \\ 0 \end{bmatrix} \; , \; y_3 := Cy_2 = \text{etc.}$$

$$s := \frac{1}{16!} y_0$$

For k = 3,2,1,0

$$s := [C^4]s + \sum_{i=0}^{3} \frac{1}{(4k+i)!} y_i$$

$$f_j := s$$

whereupon

$$\begin{bmatrix} K_1(t_0) \\ H_2(t_0) \\ G_3(t_0) \\ I \end{bmatrix} = [f_1|f_2|\cdots|f_n]$$

In the above, the product $[C^4]s$ has the form

$$\begin{bmatrix} X_2^T & U_2 & Z_2 & Y_2 \\ 0 & X_2^T & R_2 & V_2 \\ 0 & 0 & X_2 & D_2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} \begin{matrix} \} n \\ \} n \\ \} n \\ \} p \end{matrix}$$

Note that the subvector s_4 is always a multiple of some column of the $p \times p$ identity. (We mention these things simply to alert the reader to the kinds of economies possible in computations.)

Our final task is to compute a selected portion of columns $2n+1$ to $3n$ of the matrix $R_{16,0}(Ct_0)$:

$$\begin{bmatrix} H_1(t_0) \\ G_2(t_0) \\ F_3(t_0) \\ 0 \end{bmatrix} \begin{matrix} \} n \\ \} n \\ \} n \\ \} n \end{matrix}$$

In particular, we see that both $F_3(t_0)$ and $G_2(t_0)$ are required for the computation of the integrals $H(t_0)$, $Q(t_0)$, $M(t_0)$, and $W(t_0)$. Denoting $\hat{C}$ by

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

we see that Algorithm 2 can be put to the task of computing $F_3(t_0)$ and $G_2(t_0)$ as follows

For $j = 1,\ldots,n$

$Y_0 := \begin{bmatrix} 0 \\ e_j \end{bmatrix}$; $Y_1 := \hat{C}Y_0 = \begin{bmatrix} Q_c e_j \\ Ae_j \end{bmatrix}$

$Y_2 := \hat{C}Y_1 = \begin{bmatrix} -A^T(Qe_j) + Q_c(Ae_j) \\ A(Ae_j) \end{bmatrix}$

$Y_3 := \hat{C}Y_2 =$ etc.

$s := \frac{1}{16!} Y_0$

For $k = 3,2,1,0$

$s := [\hat{C}^4]s + \sum_{i=0}^{3} \frac{1}{(4k+i)!} Y_i$

$f_j := s$

whereupon

$$\begin{bmatrix} G_2(t_0) \\ F_3(t_0) \end{bmatrix} = [f_1|f_2|\ldots|f_n]$$

We call attention to the fact that the products $[\hat{C}^4]s$ have the form

$$\begin{bmatrix} X_2^T & R_2 \\ 0 & X_2 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \begin{matrix} \} n \\ \} n \end{matrix}$$

With these computations we have completed our calculations of the relevant portions of the matrix $R_{16,0}(Ct_0)$. Now in many applications, only a subset of the integrals $H(\Delta)$, $Q(\Delta)$, $M(\Delta)$, and $W(\Delta)$ are required. When this is the case, the above algorithm can be further streamlined. For example, if all but the integral $W(\Delta)$ is desired, then we effectively work with a C matrix of the form

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

and take care to delete all the above computations which a specific to $W(\Delta)$. For

example, we would not bother to compute U_2, Z_2, and Y_2. Similarly, if only $H(\Delta)$ is wanted we work with

$$C = \begin{bmatrix} A & B \\ O & O \end{bmatrix}$$

while if only $Q(\Delta)$ is desired we set

$$C = \begin{bmatrix} -A^T & Q_c \\ O & A \end{bmatrix}$$

The following table summarizes the work and storage requirements of Algorithm 1 when $(p,q) = (16,0)$ and when Algorithm 2 is invoked to compute $R_{16,0}(Ct_0)$. By way of comparison, we have included the corresponding figures for the case when $p,q) = (7,7)$ and no slick versions of Horner's rule is invoked during the computation of the numerator and denominator matrices. (Hardly any savings would result because "7" is too low a degree to substantially speed up Horner's rule.) The reason for using the (7,7) approximant is that it is lowest order diagonal approximant for which $\varepsilon \leq 10^{-18}\,\|C\|$, the value of ε associated with $R_{16}(Ct_0)$.

We see from the Table that the new approach requires less storage and generally less work. Because Taylor approximation is easier to program and leads to shorter codes, we argue that this new approach is marginally better.

References

[1] E.S.Armstrong and A.K.Caglayan, "An Algorithm for the Weighting Matrices in the Sample-Data Optimal linear Regulator Problem", NASA Langley Res.Cent.,HAmpton, VA., NASA TN D-8372,1976.

[2] C.B.Moler and C.Van Loan, "Nineteen Dubious Ways to Compute the Exponential of a Matrix", SIAM Review, October, 1978.

[3] C.Van Loan, "Computing Integrals Involving the Matrix Exponential", IEEE TRans.Auto.Contr., AC-23, 1978, 395- 404.

[4] C.VanLoan, "A Note on the Evaluation of Matrix Polynomials", Cornell Computer Science Tech. Report TR78-353, 1978.

Matrices Computed	Storage New:	Storage Old:	Multiplicative Operations New:	Multiplicative Operations Old:
F	$3n^2$	$4n^2$	$(7+j)n^3$	$(7+j)n^3$
F,H	$3n^2+ 2np$	$4n^2+ 4np$	$(7+j)n^3 + (7+j)n^2p$	$(7+j)n^3 + (7+j)n^2p$
F,Q	$6n^2$	$8n^2$	$(19 + \frac{5}{2}j)n^3$	$(21+\frac{5}{2}j)n^3$
F,H,Q,M	$6n^2+ 5np$	$8n^2+ 7np$	$(19+\frac{5}{2}j)n^3 + (20+3j)n^2p$	$(21+\frac{5}{2}j)n^3 +(21+3j)n^2p$
F,H,Q,M,W	$8n^2+ 7np$	$11n^2+10np$	$(21+\frac{5}{2}j)n^3 + (31+\frac{11}{2}j)n^2p$	$(28+\frac{5}{2}j)n^3 + (30+\frac{11}{2}j)n^2p$

The figures associated with the "old" algorithm ,i.e. Algorithm 1 with $(p,q) = (7,7)$, were derived by substituting q=7 into Table I in [3]. The work and storage assessments associated with the "new" algorithm, i.e. Algorithm 1 with $(p,q)= (16,0)$ and use of Algorithm 2 in the evaluation of $R_{16,0}(Ct_0)$, can be deduced by carefully going over the above calculations.