A Rational Interpolation Technique To Approximate The Time-Dependent Matrix Exponential

STEPHEN STUBBERUD
ORINCON Corporation
9363 Towne Centre Dr.
San Diego, CA 92121
United States of America

Abstract - While the computation of the matrix exponential is known to be dubious, the computed value is used widely in system analysis. This is especially true for the time-dependent matrix exponential. As the systems that we model become more complex, the order of the systems, and thus the order of the matrix exponential become larger. Even with today's increases in processing power, the computation of numerous values to visualize or analyze the system can tax computational output to point where a degradation in performance can be seen by the user. Often great accuracy is not needed especially in the visualization and interpolations can be used. In this paper, we present a rational interpolation method to estimate the time-dependent matrix exponential that can also estimate the error bound of the interpolation. We clearly note that, while this method utilizes significantly less computational resources than actual computation, it is dependent on the actual computational method chosen for its performance.

Keywords – matrix exponential, rational interpolation, polynomial approximation, computational efficiency, Taylor series, finite difference

1 Introduction

Some engineering software tools provide automated calculation of points to provide visualization of the system being analyzed. Also, in some analysis only a rough estimate of the system behavior is needed or even accurate. Even with today's processing power, our systems can experience a computational bottleneck. Often this is a result of the increase in the complexity of the systems being modeled. Function evaluation for large scale models can tie up precious computer resources, especially during the project design and testing phases, when repeated calculations are required. In the case of GUIs, many calculations are required quickly in order to detail, in near real-time, interesting sections of a graph. In both cases, full computer accuracy (i.e., all digits of a double precision number) is not usually required and may be sacrificed to some degree to increase computational speed. To alleviate this problem the development of faster, yet still accurate, computational techniques for various matrix-valued functions is still being examined.

In previous works by the author and his colleagues (see [4], [5], [6], and [9]), a rational interpolation technique was developed for the frequency response of the state-space system modeled by

$$E\dot{x} = Ax + Bu$$
$$y = Cx$$

where the matrix E may be either singular or nonsingular. During this research, we theorized that this technique could be adapted to evaluate the time dependent version of the matrix exponential, e^{At}.

The matrix exponential's theoretical use appears throughout control theory [8]. It has often been used to analyze systems modeled by the first-order state-space model

$$\dot{x} = Ax + Bu$$
$$y = Cx$$

where either the solution vector *x(t)* or a multiple thereof is sought. Another prominent development, the so-called hump of $\left\| e^{At} \right\|_2$, which describes the trade-off between the speed of a system and its

transient behavior, has been noted to be of some interest [2].

While its theoretical importance is great, there does not exist a best numerical method to compute the matrix exponential [7]. However, its explicit computation usually can be avoided. In the few cases where computation of the matrix exponential is required, it is for the time-dependent matrix exponential, e^{At} , for a large set defining the values of t. In such cases, the acceptable methods for computing the matrix exponential involve substantial computations.

The principal development in this paper is a rational interpolation method to compute the time-dependent matrix exponential in both a fast and reliable manner. We note here that this technique has been devised to be applied over subintervals of the region defined by the time set of t. In this way, we can avoid potential numerical instabilities caused by large-order interpolations, and new computational bottlenecks, which often accompany larger systems, are avoided.

2 Interpolation Development

The basic idea of this method is based on the simple Taylor series approximation

$$G(t+h) = T_0 + T_1 h + \cdots + T_k h^k + E_k \quad (1)$$

but considered in the general interpolation form with *k+1* interpolation points :

$$G(t+h) = G_0 + G_1(h-h_0) + \cdots + G_k(h-h_0)(h-h_1)\cdots(h-h_{k-1}) + E_k \quad (2)$$

is approximately $O(kn_A^2)$ floating-point operations (flops) for each value of *h*. The cost of computing each coefficient matrix is approximately the same as evaluating *G* by the method that would normally be preferred. Thus, for a large number of values in the set of *t*, a significant computational savings can be made over the acceptable computational methods, which normally require $O(n_A^3)$ flops.

2.1 The Polynomial Interpolation Algorithm for the Matrix Exponential

The first coefficient matrix G_0 of the k^{th}-order polynomial approximation (3) is equal to $e^{A(t+h_0)}$. In order to compute the coefficient matrices $G_1,\ldots,G_k$, we shall employ finite differences of the function

$$M = e^{At}. \quad (4)$$

The first order difference is defined by

$$M[h_0,h_1] = \frac{M(h_1) - M(h_0)}{h_1 - h_0} \quad (5)$$

while higher-order differences are defined recursively by

$$M[h_0,h_1,\ldots,h_k] = \frac{M[h_1,\ldots,h_k] - M[h_0,\ldots,h_{k-1}]}{h_1 - h_0} \quad (6)$$

The k^{th}-order interpolation approximation can then be written as

$$P_k(h) = M(h_0) + M[h_0,h_1](h-h_0) + \cdots + M[h_0,h_1,\ldots,h_k](h-h_0)\cdots(h-h_{k-1}) \quad (7)$$

with the interpolation error

$$E_k = M[h_0,h_1,\ldots,h_k,h]\prod_{i=0}^{k}(h-h_i). \quad (8)$$

Although finite differences have a certain elegance to their formulation, they can encounter numerical inaccuracies due to subtraction of near-equal-valued quantities. An extreme example of this is the case in which all of the interpolation points are the same. In theory, the first-order difference is exactly the first derivative of *M*, but numerically it is useless. Fortunately, the differences of the matrix exponential can be expressed in matrix product forms which avoid these cancellation problems as the following lemma and theorem show. A similar development for the resolvent function, , can be found in [3]. We present the first-order difference separately because an interesting development occurs which affects the hypothesis for the development of the product form of the higher-order differences.

Lemma 1 *Let*

$$G(h) = e^{A(t+h)}$$

and, without loss of generality, $h_0 = 0$ *Then*

$$G(h_1) - G(h_0) = e^{At}(e^{Ah_1} - I). \quad (9)$$

Proof:

$$\begin{aligned} G(h_1) - G(h_0) &= e^{A(t+h_1)} - e^{At} \\ &= e^{At}e^{Ah_1} - e^{At}I \\ &= e^{At}(e^{Ah_1} - I). \end{aligned}$$

Q.E.D.

Notice that in (9) there is not a term to cancel the step difference which is the denominator of the finite difference formula. For higher-order differences, the simplicity and elegance that led us to a product form for the finite difference would be ruined by the necessity of computing the lowest common denominator of each lower-order finite difference contained in the higher-order finite difference. However, if the step sizes are chosen to be uniform with respect to each other, the denominator does not cause any of these algebraic problems. The following theorem extends Lemma 1 to the higher-order finite differences using uniform steps to create a closed-form solution to the denominator.

Theorem 2 *Let*

$$G(h) = e^{A(t+h)},$$
$$h = h_i - h_{i-1}, \qquad i = 1,\ldots,n$$

and

$$h_0 = 0.$$

Then

$$G[h_0, h_1, \ldots, h_k] = \frac{e^{At}(e^{Ah} - I)^k}{k!h^k}. \quad (10)$$

Proof: Letting $k = 2$, we get

$$G[h_0, h_1, h_2] = \frac{G[h_1, h_2] - G[h_0, h_1]}{2h}.$$

Then by the preceding lemma

$$\begin{aligned} G[h_0, h_1, h_2] = & \\ & \left[\frac{e^{A(t+h)}(e^{Ah} - I) - e^{At}(e^{Ah} - I)}{h}\right]/2h \\ &= \frac{(e^{A(t+h)} - e^{At})(e^{Ah} - I)}{2h^2} \\ &= \frac{e^{At}(e^{Ah} - I)(e^{Ah} - I)}{2h^2} \\ &= \frac{e^{At}(e^{Ah} - I)^2}{2h^2}. \end{aligned}$$

Thus the result is true for $k = 2$. Now assume that it is true for $k = n$, and we will prove that it is then true for $k = n + 1$.

$$\begin{aligned} G[h_0, h_1, \ldots, h_{n+1}] & \\ &= \frac{G[h_1, h_2, \ldots, h_{n+1}] - G[h_0, h_1, \ldots, h_n]}{h_{n+1} - h_0} \\ &= \frac{e^{A(t+h)}(e^{Ah} - I)^n - e^{At}(e^{Ah} - I)^n}{(n+1)!\,h^{n+1}} \\ &= \frac{(e^{A(t+h)} - e^{At})(e^{Ah} - I)^n}{(n+1)!h^{n+1}} \\ &= \frac{e^{At}(e^{Ah} - I)^{n+1}}{(n+1)!h^{n+1}}. \end{aligned}$$

Since the result is true for $k = n + 1$, by induction the theorem is proved.

Q.E.D.

From this theorem, the polynomial interpolation formula (7) becomes

$$\begin{aligned} P_k(h) = e^{At} &+ \frac{e^{At}(e^{Ah} - I)}{h_s}(h - h_0) + \cdots \\ &+ \frac{e^{At}(e^{Ah} - I)^k}{(k)!\,h_s^k}(h - h_0)(h - h_1)\cdots(h - h_{k-1}) \end{aligned} \quad (11)$$

where

$$h_s = h_i - h_{i-1}, \qquad i = 1,\ldots,k,$$

and

$$t = t_0 + h$$

where is the initial time of the interval on which we are interpolating. However, for the error equation (8) there does not exist such an elegant form.

The question then arises whether series (11) converge. If the step size were allowed to become zero, the series would degenerate into the Taylor series expansion for the matrix exponential, which, although can be numerically inaccurate, does converge. The following theorem we shall determine the conditions of the interpolation step size for convergence of (11) for a nonzero interpolation step size.

Theorem 3 *For any stable system, an h_s can be found such that the infinite series*

$$P_k(h) = e^{At} + \frac{e^{At}(e^{Ah} - I)}{h_s}(h - h_0) + \cdots$$
$$+ \frac{e^{At}(e^{Ah} - I)^k}{(k)!\,h_s^k}(h - h_0)(h - h_1)\cdots(h - h_{k-1})$$
$$+ \cdots$$

is always convergent

Proof: Since the other terms cancel each other or are constant, all we need to show the maximum eigenvalue is less than one or that the following condition of the spectral radius is true:

$$\rho\left(e^{Ah} - I\right) < 1.$$

This implies $\max_{\lambda \in \Lambda(A)} \left| e^{\lambda h_s} - 1 \right| < 1$.

We rewrite the equation as

$$\left| e^{\lambda_{\max} h_s} - 1 \right| < 1$$
$$\leftrightarrow \left| e^{(r_m + \theta_m j) h_s} - 1 \right| < 1$$
$$\leftrightarrow \left(e^{2 r_m h_s} - 2 e^{r_m h_s} \cos(\theta_m h_s) + 1 \right)^{1/2} < 1$$
$$\leftrightarrow e^{2 r_m h_s} - 2 e^{r_m h_s} \cos(\theta_m h_s) + 1 < 1$$
$$\leftrightarrow e^{2 r_m h_s} - 2 e^{r_m h_s} \cos(\theta_m h_s) < 0$$
$$\leftrightarrow e^{r_m h_s} \left(e^{r_m h_s} - 2\cos(\theta_m h_s) \right) < 0$$
$$\leftrightarrow e^{r_m h_s} - 2\cos(\theta_m h_s) < 0$$
$$\leftrightarrow 2\cos(\theta_m h_s) > e^{r_m h_s}$$
$$\Rightarrow -\pi/2 < \theta h_s < \pi/2$$

With this condition upon θ and the stability criterion, we can see that we can always find a step size small enough to meet the inequality.

Q.E.D.

This shows that the interpolation routine is numerically feasible.

2.2 Exponential Interpolation Algorithm

To begin, we choose the interpolation interval. Next, decide what order interpolation to use. The step size h_s between the interpolation points which are to be spread uniformly across the interval are then computed. Then compute the initial matrix exponential

$$C = e^{At_o}$$

Follow this by the computation of the interpolating matrix exponential

$$H = e^{Ah_s}$$

Next compute the interpolation coefficient matrices.

$$X_i = \left(e^{Ah_s} - I\right) X_{i-1} / (i h_s)$$

where $X_o = I$. Premultiply each X_i by *C*.

Sum the approximation series

$$M(h) = G_o + G_1(h - h_0) + \cdots$$
$$+ G_k(h - h_0)(h - h_1)\cdots(h - h_{k-1})$$

3 Numerical Example

A tenth-order example matrix was generated randomly using MATLAB. The real Schur form of the matrix was found. Those eigenvalues with real

parts that were unstable (positive) were made stable by multiplying them by a negative one. We looked at the interpolation of the region around the hump of the 2-norm, [0,2]. We computed an interpolation and the actual matrix exponential for each set of data. We tried several different permutations of interpolation point spacing, number of interpolation points and regions over which we computed the approximation compared to the interpolation points. Results are shown for each.

Figure 1 shows the results of when five interpolation points of spacing 0.2 are used. The error of the interpolation is shown in Figure 2. Figure 3 shows that the same interpolation scheme does well beyond the hump until the change in dynamic behavior takes place. We have seen that creating a second interpolation region for the tail is good for the rest of the curve. Figure 4 shows the result of using three interpolation points of 0.3 spacing. The interpolation error is seen in Figure 5. A larger region with more interpolation points, eight, separated by 0.3 is shown in Figure 6. These results indicate that the interpolation routine works well even beyond the region of the interpolation points. As with the results in the frequency response version of the routine, significant changes in the dynamics indicate where new interpolation points are needed.

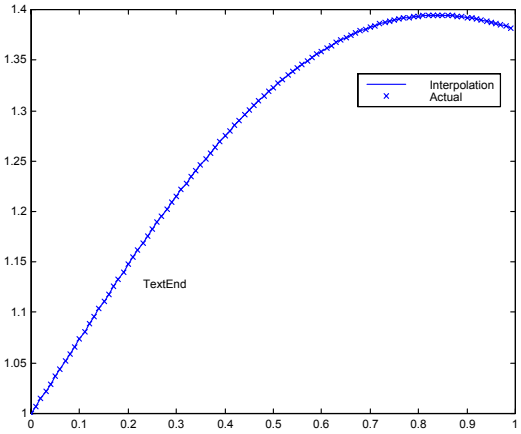

Figure 1: Interpolation of the region [0,1] with five points is visually accurate.

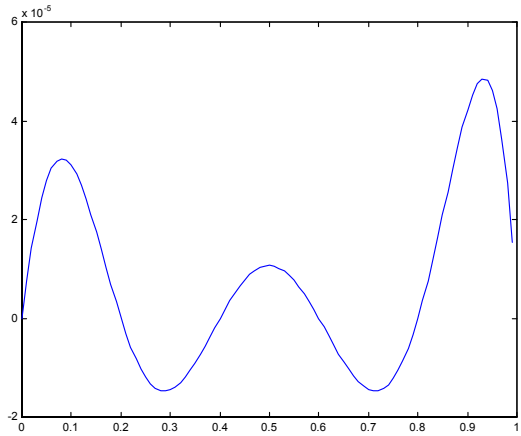

Figure 2: At an order of 10^{-6} the actual interpolation error is small

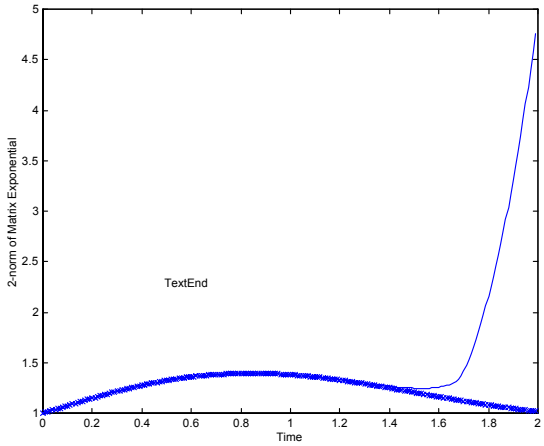

Figure 3: Beyond the region of the hump, the interpolation grows without bound. It is outside its region of approximation.

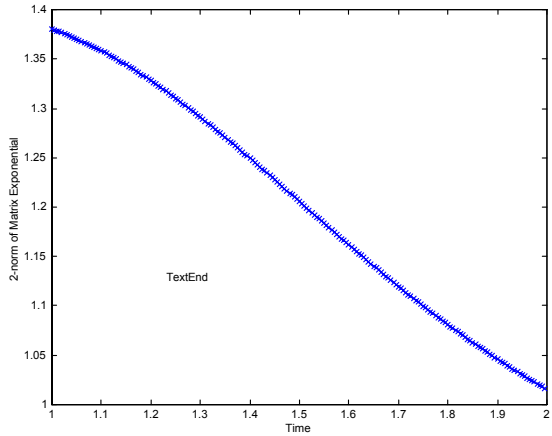

Figure 4: Three Interpolation points do well.

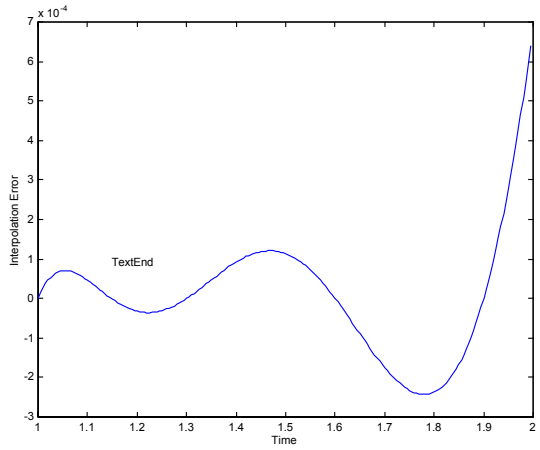

Figure 5: The interpolation error grows at the tail of the region which is outside the interpolation points.

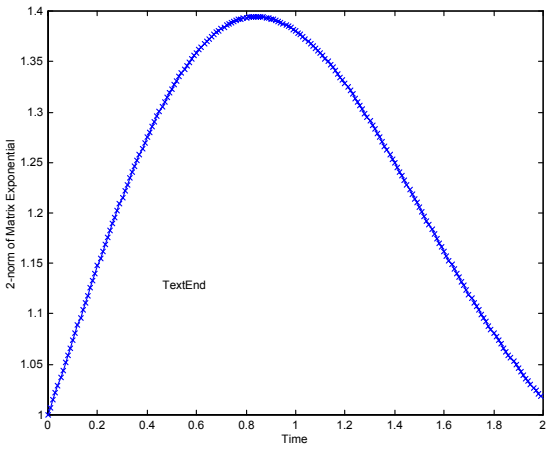

Figure 6: Interpolation across the entire hump region is attainable with eight interpolation points across the region.

4 Conclusion

In this paper an interpolation routine for approximating the time-dependent matrix exponential was developed. While the calculation of the matrix exponential should be avoided, when the need arises this technique can evaluate it accurately and in a fraction of the time required by conventional method. While the approach is sound more research and development is required to make this a use product in software design systems. A method to estimating the interpolation error is of highest priority; followed by the development of an autonomous version of this technique to be readily.

References

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

[2] Hewer, G. A. and C.S. Kenney, "The Sensitivity of the Stable Lyapunov Equation, "SIAM J. of Control Opt., 26(1988), pp. 321-344.

[3] Kato, T., Perturbation Theory for Linear Operators, 2nd Edition, Springer-Verlag, Hidelberg, 1982.

[4] Kenny, C.S., A.J. Laub, and S.C. Stubberud, "Frequency Response via Rational Interpolation," Trans. Auto. Control, Vol. 38, No. 8, pp. 1203 - 1213, August 1993.

[5] Kenney, C.S., S.C. Stubberud, and A.J. Laub, "A Rational Interpolation Method to Compute Frequency Response," Fifth Annual NASA/NSF/DOD Workshop on Aerospace Computational Control, Santa Barbara, Calif., August 1992.

[6] Kenney, C.S., S.C. Stubberud and A.J. Laub, "Frequency Response via Rational Interpolation," Proceedings of the 1992 IEEE Control Systems Society Symposium on Computer-Aided Control System Design, pp. 188-195, Napa, Calif. March, 1992.

[7] Moler, C.B. and C. F. Van Loan, "Nineteen Dubious Ways to Compute the Exponential of a Matrix," Siam Review, 20(1978), pp. 801-836.

[8] Santina, M.S., A.R. Stubberud, and G.H. Hostetter, *Digital Control System Design 2nd Edition,* Saunders College Publishing, Fort Worth, Texas, 1994.

[9] Stubberud, S.C., A.J. Laub, and C.S. Kenney, "Computation of Frequency Response of Descriptor Systems by Rational Interpolation," Book Chapter, Control and Dynamic Systems Series, C.T. Leondes (ed.), vol. 56, pp. 267-301, 1993.