Electronic Journal of Differential Equations, Vol. 2014 (2014), No. 79, pp. 1–7.
ISSN: 1072-6691. URL: http://ejde.math.txstate.edu or http://ejde.math.unt.edu
ftp ejde.math.txstate.edu

EXPLICIT EXPRESSIONS FOR THE MATRIX EXPONENTIAL FUNCTION OBTAINED BY MEANS OF AN ALGEBRAIC CONVOLUTION FORMULA

JOSÉ ROBERTO MANDUJANO, LUIS VERDE-STAR

Abstract. We present a direct algebraic method for obtaining the matrix exponential function $\exp(tA)$, expressed as an explicit function of t for any square matrix A whose eigenvalues are known. The explicit form can be used to determine how a given eigenvalue affects the behavior of $\exp(tA)$. We use an algebraic convolution formula for basic exponential polynomials to obtain the dynamic solution $g(t)$ associated with the characteristic (or minimal) polynomial $w(x)$ of A. Then $\exp(tA)$ is expressed as $\sum_k g_k(t)w_k(A)$, where the $g_k(t)$ are successive derivatives of g and the w_k are the Horner polynomials associated with $w(x)$. Our method also gives a number δ that measures how well the computed approximation satisfies the matrix differential equation $F'(tA) = AF(tA)$ at some given point $t = \beta$. Some numerical experiments with random matrices indicate that the proposed method can be applied to matrices of order up to 40.

1. Introduction

The importance of the function e^{tA}, where A is a square matrix and t is a real or complex variable, is well-known. For example, many problems in areas such as Control and Systems Theory, Physics, Biomathematics, Nuclear Sciences, etc., require the solution of a linear system of first-order differential equations with constant coefficients and given initial conditions. Such solution has the form $\mathrm{e}^{tA}C$, where C is a constant vector. The behavior of e^{tA} as a function of t depends in a complex way on the eigenvalues and the entries of A.

In this article we present a method that gives an explicit formula for e^{tA} in terms of linear combinations of functions of the form $t^k \exp(\lambda t)$. Consequently, we can easily compute e^{tA} (and also $\mathrm{e}^{tA}C$) for a large number of values of t.

The explicit expressions can also be used to determine the influence of each eigenvalue in the behavior of the matrix exponential function, and the effect of small perturbations of the eigenvalues. We can also study how the distribution of the eigenvalues in the complex plane determines how accurately e^{tA} can be computed in some given region.

2000 *Mathematics Subject Classification.* 34A30, 65F60, 15A16.
Key words and phrases. Matrix exponential; dynamic solutions; explicit formula; systems of linear differential equations.

Submitted June 20, 2013. Published March 21, 2014.

Our method is a direct algebraic construction of e^{tA} using the entries of A and its eigenvalues. It does not use numerical integration, integral transforms, Padé approximations, orthogonal polynomial series, nor differential equations solvers. We use an algebraic convolution formula for exponential polynomials which is based on the multiplication of basic rational functions.

Our approach is similar to the one used by Luther and Rost [2], but we do not need the inversion of confluent Vandermonde matrices. A straightforward application of our method yields all the explicit formulas obtained by Wu in [9] and many other formulas for matrices with a small number of distinct eigenvalues.

As a consequence of the Cayley-Hamilton theorem, all the powers A^m of a square matrix A can be expressed in terms of the initial powers $I, A, \dots A^n$, for some fixed n. This fact can be used to show that functions of the form $f(tA)$ may be expressed as polynomials in A with coefficients that depend on t. See [7], where this approach is used to study matrix functions $f(tA)$ where f is given by a power series. In particular, for the exponential function we obtained in [7] the formula

$$\mathrm{e}^{tA} = \sum_{k=0}^{n} g_k(t) w_k(A), \tag{1.1}$$

where A is a square matrix, the w_k are the Horner polynomials associated with a polynomial $w(x)$ of degree $n+1$ such that $w(A) = 0$, and the $g_k(t)$ are exponential polynomials that depend only on the eigenvalues of A. Note that $w(x)$ may be the minimal or the characteristic polynomial of A. Note also that formula (1.1) holds when A is replaced by any matrix B that is similar to A, with the same coefficient functions $g_k(t)$. The set $\{w_k(x) : 0 \le k \le n\}$ is often called the *control basis* associated with w.

The function $g_n(t)$, called the *dynamic solution*, is the solution of the scalar differential equation $w(D)y(t) = 0$ with initial conditions $D^k g_n(0) = 0$ for $k = 0, 1, 2, \dots, n-1$ and $D^n g_n(0) = 1$. The other coefficient functions $g_k(t)$ are obtained by repeated differentiation of $g_n(t)$, that is, $g_{n-k}(t) = D^k g_n(t)$ for $0 \le k \le n$. In the present paper we compute the dynamic solution directly from the roots of w using an algebraic formula for the convolution of basic exponential polynomials. If the number of distinct roots and nonzero coefficients of the polynomial w is small, then the numerical errors in the computation are negligible. This is the case of the polynomials considered by Wu in [9] to obtain explicit formulas.

The use of the dynamic solution to solve matrix differential equations was introduced in [5]. For an algebraic approach to convolutions see [8].

If the number of distinct eigenvalues is relatively large then there are some sources of numerical errors. The quality of the computed function $f(tA)$ as an approximation of e^{tA} can be determined by measuring how well it satisfies the differential equation $f'(tA) = Af(tA)$. This is easily done and requires only one additional derivative of g_n and the computation of $f'(tA)$, which is analogous to (1.1).

It is well-known that the computation of the matrix exponential is a difficult numerical problem. See [4] and [1]. Our numerical experiments show that our method works well for matrices A of moderate size, whose characteristic values are known with sufficient accuracy, regardless of the norm of A and the multiplicities of the eigenvalues. In order to determine the accuracy of the computations with the explicit expressions produced by our algorithm we performed some numerical

experiments with random matrices of order up to 40, using a Maple program. Since our method is essentially equivalent to Hermite interpolation of the exponential function at the spectrum of A, the accuracy of the results depend on a complex way on the distribution of the eigenvalues in the complex plane.

It is important to notice that the main objective of our method is to obtain an explicit expression for e^{tA}, where t is a variable. Our method can not be directly compared to algorithms of a different nature that compute e^{A} for a given A, such as the ones based on scaling and squaring, or Padé approximations, which are designed to obtain high accuracy or to handle large matrices for certain restricted sets of matrices.

2. Notation and basic results

In this section we introduce notation that we use in the rest of the paper and present some basic results taken from [6] and [7], where the proofs and more detailed discussion can be found.

Let n be a nonnegative integer and let $w(z) = z^{n+1} + b_1 z^n + \cdots + b_{n+1}$ be a monic polynomial of degree $n+1$ with complex coefficients. The sequence $\{w_k\}$ of *Horner polynomials* associated with w is defined by

$$w_k(z) = z^k + b_1 z^{k-1} + b_2 z^{k-2} + \cdots + b_k, \quad k \geq 0, \tag{2.1}$$

where $b_0 = 1$ and $b_j = 0$ for $j > n+1$. Note that $\{w_k : k \geq 0\}$ is a basis for the vector space of all the polynomials and $\{w_0, w_1, \ldots, w_n\}$ is a basis for the subspace $\mathcal{P}_n$ of all polynomials with degree at most n. This basis is often called the control basis. Note also that $w_{n+1} = w$, $w_{n+1+k}(z) = z^k w(z)$ for $k \geq 0$, and

$$w_{k+1}(z) = z w_k(z) + b_{k+1}, \quad k \geq 0. \tag{2.2}$$

In [6] we proved the following result, that we call the general interpolation formula.

Theorem 2.1. *Let w be as previously defined, let f be a function defined on the multiset of roots of w, and let $p(x)$ be the polynomial in $\mathcal{P}_n$ that interpolates f at the roots of w. Then*

$$p(x) = \Delta_w \left(f(z) w[z, x] \right), \tag{2.3}$$

where Δ_w is the divided difference functional with respect to the roots of w and acts with respect to z, and $w[z, x]$ is the difference quotient

$$w[z, x] = \frac{w(z) - w(x)}{z - x}.$$

A simple computation yields

$$w[z, x] = \sum_{k=0}^{n} w_k(x) z^{n-k}. \tag{2.4}$$

For f as in the previous theorem and a parameter t, define

$$g_k(t) = \Delta_{w(z)} \left(z^{n-k} f(tz) \right), \quad 0 \leq k \leq n. \tag{2.5}$$

Now let A be any square matrix such that $w(A) = 0$. Then we have

$$f(tA) = \Delta_{w(z)} \big(f(tz) w[z, A] \big) = \sum_{k=0}^{n} g_k(t) w_k(A). \tag{2.6}$$

In the simple case in which w has $n+1$ distinct roots $\lambda_0, \lambda_1, \ldots, \lambda_n$ we have

$$g_k(t) = \sum_{j=0}^{n} \frac{\lambda_j^{n-k} f(t\lambda_j)}{w'(\lambda_j)}, \quad 0 \le k \le n. \tag{2.7}$$

If $f(x) = \exp(x)$ then each $g_k(t)$ is an exponential polynomial.

In the general case we have

$$w(z) = \prod_{j=0}^{r} (z - \lambda_j)^{m_j+1}, \tag{2.8}$$

where the λ_j are distinct complex numbers, the m_j are nonnegative integers, and $\sum_j (m_j + 1) = n + 1$. Let us define the basic exponential polynomials

$$\tilde{f}_{x,k}(t) = \frac{t^k}{k!} \mathrm{e}^{xt}, \quad x \in \mathbb{C},\ k \in \mathbb{N}. \tag{2.9}$$

In [7] we proved that the dynamic solution g_n associated with w is given by

$$g_n = \tilde{f}_{\lambda_0,m_0} * \tilde{f}_{\lambda_1,m_1} * \cdots * \tilde{f}_{\lambda_r,m_r}, \tag{2.10}$$

$g_{k-1} = g_k'$ for $1 \le k \le n$, and the convolution of two basic exponential polynomials is given by

$$\tilde{f}_{x,k} * \tilde{f}_{x,m} = \tilde{f}_{x,k+m+1}, \quad x \in \mathbb{C},\ k, m \in \mathbb{N}, \tag{2.11}$$

and

$$\begin{aligned} \tilde{f}_{y,k} * \tilde{f}_{x,m} &= (-1)^{k+1} \sum_{i=0}^{m} \binom{k+i}{i} \frac{\tilde{f}_{x,m-i}}{(y-x)^{k+i+1}} \\ &\quad + (-1)^{m+1} \sum_{j=0}^{k} \binom{m+j}{j} \frac{\tilde{f}_{y,k-j}}{(x-y)^{m+j+1}}, \quad x \ne y. \end{aligned} \tag{2.12}$$

This convolution product coincides with the Duhamel convolution, usually defined by means of integrals, when it is applied to exponential polynomials.

3. Proposed algorithm

Let A be a given square matrix, let w be a monic polynomial such that $w(A) = 0$, let $\lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_r$ be the distinct roots of w, and let $m_j + 1$ be the multiplicity of λ_j for $0 \le j \le r$.

Step 1. Compute the dynamic solution g_n, defined by (2.10), by repeated application of the convolution formula (2.12). In this way we obtain g_n in the form

$$g_n = \sum_{j=0}^{r} \sum_{k=0}^{m_j} \alpha_{j,k}\ \tilde{f}_{\lambda_j,k},$$

where the coefficients $\alpha_{j,k}$ are numbers. Using (2.9) we get $g_n(t)$ as a linear combination of the basic exponential polynomials associated with the roots of w.

Step 2. Obtain the functions g_k by repeated differentiation of g_n; that is, $g_{k-1} = g_k'$ for $k = n, n-1, n-2, \ldots, 1$.

Step 3. Find the matrices $w_k(A)$ using Horner's recurrence relation (2.2).

Step 4. Substitute the scalar functions $g_k(t)$ and the matrices $w_k(A)$ in formula (2.6) to obtain $\exp(tA)$.

Note that computing the $w_k(A)$ is the step with the largest computational cost, since it requires $n - 1$ multiplications by the matrix A. Note also that this step

is independent of steps 1 and 2. If we want to compute $\exp(tA)C$, where C is a vector, then we can compute the vectors $w_k(A)C$ and this requires n matrix-vector multiplications.

Once we have the explicit expression for $\exp(tA)$, to find $\exp(\beta A)$ for a given β, it is sufficient to compute the $n+1$ scalar functions $g_0(\beta), g_1(\beta), \dots, g_n(\beta)$, and then use (2.6), which reduces to the computation of a linear combination of the already known matrices $w_k(A)$. Notice that there are no matrix multiplications involved here. It is clear that we can get any entry of $\exp(tA)$ as an explicit exponential polynomial.

An important property of our algorithm is that it also provides a simple way to estimate the relative error in the computation of $\exp(tA)$ at a point $t = \beta$. Note first that $D_t \exp(tA) = A \exp(tA)$, where D_t denotes differentiation with respect to t. Therefore $\exp(tA)$ satisfies the matrix equation

$$A = \exp(-tA) D_t \exp(tA), \quad t \in \mathbb{R} \tag{3.1}$$

Let $F(t)$ denote the computed right-hand side of (2.6). We measure how well $F(t)$ satisfies (3.1) in the following way. By differentiation of (2.6) with respect to t we get

$$F'(t) = g_0'(t) I + \sum_{k=1}^{n} g_{k-1}(t) w_k(A). \tag{3.2}$$

For a given number β let $B = F(-\beta)F'(\beta)$ and define

$$\delta = \frac{\|B - A\|}{\|A\|}, \tag{3.3}$$

where $\|A\|$ denotes a suitable matrix norm, such as the infinity norm. Then the number δ measures how well $F(t)$ satisfies (3.1) at $t = \beta$. The numerical experiments show that δ is a good approximation of the "true" relative error

$$\mu = \frac{\|F(\beta) - E(bA)\|}{\|E(bA)\|}, \tag{3.4}$$

where $E(bA)$ is produced by the Maple command "MatrixExponential(βA)", using a precision of 100 digits. In most cases we obtain $\delta \geq \mu$.

It is important to detect multiple roots correctly and not deal with them as if they were distinct roots very close to each other, since that would introduce relatively large errors, due to the denominators in (2.12), which are powers of differences of eigenvalues.

4. Numerical results

The computational cost of the proposed algorithm increases with the size of the matrix A, since, in general, the computation of the matrices $w_k(A)$ for $0 \leq k \leq n$ requires $n-1$ multiplications by A, and w is typically the characteristic polynomial of A. Therefore for large matrices A we ca not expect to get high accuracy in the matrices computed with the explicit formula produced by our algorithm, unless A is a sparse matrix and w has a small number of suitably distributed distinct roots. We performed some numerical experiments to determine the range of values of the order of A for which the algorithm produces acceptable results.

We tested our algorithm taking A as a random matrix of order n for a few values of n between 20 and 40, and using different parameters for the random matrix generator which yield different distributions of the eigenvalues.

TABLE 1. Some numerical results

n	D	a	b	$\|A\|$	ρ	$\|\exp(A)\|$	δ	μ
20	50	-4	2	10.7875	4.99504	13.6899	2.54043e-45	2.48411e-45
20	50	-2	4	9.58886	4.70923	173.594	1.80540e-39	1.17495e-39
25	50	-4	2	13.6732	6.43704	29.7808	7.33657e-44	5.09239e-44
25	50	-2	4	13.7619	6.40041	982.736	1.31585e-34	8.66711e-35
30	60	-4	2	15.5971	8.29544	46.79	2.51331e-52	2.05524e-52
30	60	-2	4	14.7011	7.10494	1894.19	4.09793e-40	2.72607e-40
35	64	-4	2	17.0037	9.35093	48.1121	9.91921e-55	6.16559e-55
35	64	-2	4	17.7606	8.70811	8599.8	8.54165e-39	6.12971e-39
40	70	-4	2	21.3238	10.4744	46.52	2.23268e-60	2.04208e-60
40	70	-2	4	19.4641	9.73745	28070.4	8.35698e-40	5.04061e-40
40	70	-1	4	20.4786	14.8176	3.65e+06	4.83707e-30	2.49511e-30

In Table 1 the columns correspond to the following parameters: n is the order of the matrix A, D is the number of digits used in the computations, a and b are the end points of the interval where the random entries are generated using the uniform distribution, $\|A\|$ is the infinity norm of A, ρ is the spectral radius of A, $\|\exp(A)\|$ is the infinity norm of the computed $\exp(A)$, δ and μ are the relative errors defined by (3.3) and (3.4). In these computations we computed the matrix function at $t = 1$. The matrix A is the generated random matrix multiplied by a scaling factor of 0.25. This was done in order to deal with matrices whose norms have a reasonable size. Note that the less accurate results are obtained when the norm of $\exp(A)$ is large. Note also that as n increases it is necessary to increase the number of digits used in the computations in order to obtain acceptable error sizes. The entries of the computed matrix are expressed as explicit linear combinations of (complex) exponentials, for example, an entry obtained in a case with $n = 20$ looks like

$$\begin{aligned}
&0.2745\exp(1.6103t)\cos(0.8201t) - 0.0166\exp(1.6103t)\sin(0.8201t)\\
&- 0.2018\exp(0.5083t)\cos(0.4520t) - 0.1724\exp(0.5083t)\sin(0.4520t)\\
&+ 0.0063\exp(0.8337t)\cos(1.1256t) + 0.1363\exp(0.8337t)\sin(1.1256t)\\
&- 0.9704\exp(-2.1568t)\cos(0.4983t) + 0.5197\exp(-2.1568t)\sin(0.4983t)\\
&+ 0.8632\exp(-1.8023t)\cos(0.2841t) - 1.4200\exp(-1.8023t)\sin(0.2841t)\\
&- 0.0190\exp(0.6122t)\cos(2.4512t) - 0.0273\exp(0.6122t)\sin(2.4512t)\\
&- 0.1654\exp(-1.4406t)\sin(1.4041t) - 0.0217\exp(-1.4406t)\cos(1.4041t)\\
&+ 0.0345\exp(-0.4696t)\cos(2.0036t) + 0.0760\exp(-0.4696t)\sin(2.0036t)\\
&- 0.0075\exp(-0.2615t) + 0.0426\exp(0.9229t) - 0.0193\exp(1.6650t)\\
&+ 0.0186\exp(-0.7649t),
\end{aligned}$$

where the number of digits was truncated to abbreviate the expression. Therefore, using this method for matrices of large order n is not very convenient. In addition, in such cases computing the eigenvalues with the required accuracy is not easy.

In [3] we use another algorithm, where the functions g_k are expressed as truncated Taylor series and the eigenvalues of A are not used.

Since the computation of $\exp(tA)$ is essentially a matrix valued Hermite interpolation at the multiset of eigenvalues, the accuracy of the computed results depends on the distribution of the eigenvalues on the complex plane. This is why the error estimate δ is very useful.

The repeated differentiation to compute the functions $g_k(t)$ and the computation of the matrices $w_k(A)$ are steps that may reduce the accuracy of the computation if there are either eigenvalues or entries of A with large absolute values. In addition, if there are sets of eigenvalues very close to each other then the repeated application of formula (2.12) is another source of errors.

Relatively large errors are obtained when an eigenvalue of multiplicity greater than one is not properly detected and is treated as several distinct eigenvalues that are very close to each other. In that case some of the denominators in (2.12) become very small. Therefore the multiple eigenvalues and their multiplicities should be properly identified, for example, using some root refining algorithm such as Newton's method.

Acknowledgments. This research was partially supported by COFAA and EDD grants from IPN, México.

References

[1] N. J. Higham; *Functions of Matrices, Theory and Computation*, SIAM Publications, Philadelphia, PA, 2008.

[2] U. Luther, K. Rost; *Matrix exponentials and inversion of confluent Vandermonde matrices*, Electronic Transactions on Numerical Analysis, 18 (2004) 91–100.

[3] J. R. Mandujano, L. Verde-Star; *Computation of the matrix exponential using the dynamic solution*, submitted.

[4] C. B. Moler, C. F. Van Loan; *Nineteen dubious ways to compute the exponential of a matrix*, twenty-five years later, SIAM Rev. 45 (2003) 3–49.

[5] J. C. Ruiz Claeyssen, T. Tsukazan; *Dynamic solutions of linear matrix differential equations*, Quart. Appl. Math., 48 (1990) 169–179.

[6] L. Verde-Star; *Divided differences and combinatorial identities*, Stud. Appl. Math., 85 (1991) 215–242.

[7] L. Verde-Star; *Functions of matrices*, Linear Algebra Appl., 406 (2005) 285–300.

[8] L. Verde-Star; *An algebraic approach to convolutions and transform methods*, Advances in Appl. Math., 19 (1997) 117–143.

[9] B. Wu; *Explicit formulas for the exponentials of some special matrices*, Appl. Math. Letters, 24 (2011) 642–647.

José Roberto Mandujano
Escuela Superior de Cómputo, Instituto Politécnico Nacional, U. Zacatenco, México D.F., México
E-mail address: jrmandujano@yahoo.com.mx

Luis Verde-Star
Department of Mathematics, Universidad Autónoma Metropolitana, Iztapalapa, Apartado 55-534, México D.F. 09340, México
E-mail address: verde@xanum.uam.mx