Electronic Transactions on Numerical Analysis.
Volume 37, pp. 147-165, 2010.
Copyright © 2010, Kent State University.
ISSN 1068-9613.

ETNA
Kent State University
http://etna.math.kent.edu

COMPUTING $\exp(-\tau \boldsymbol{A})\boldsymbol{b}$ WITH LAGUERRE POLYNOMIALS[*]

BERNARD N. SHEEHAN[†], YOUSEF SAAD[‡], AND ROGER B. SIDJE[§]

Abstract. This paper discusses a method based on Laguerre polynomials combined with a Filtered Conjugate Residual (FCR) framework to compute the product of the exponential of a matrix by a vector. The method implicitly uses an expansion of the exponential function in a series of orthogonal Laguerre polynomials, much like existing methods based on Chebyshev polynomials do. Owing to the fact that orthogonal polynomials satisfy a three-term recurrence, what these series expansion methods offer over other approaches such as Krylov subspace methods lies in the elimination of inner products and the economy in storage since there is no need to compute and keep a set of basis vectors. Compared with Chebyshev polynomials that are orthogonal within a restricted interval and need estimates of the outermost eigenvalues, Laguerre polynomials offer the added feature that they are orthogonal on the half real line, alleviating therefore the need to estimate eigenvalues.

Key words. Conjugate residual, filtered conjugate residual, polynomial filtering, exponential propagation, orthogonal polynomials

AMS subject classifications. 65F50, 65L05, 41A10

1. Introduction.

The problem of calculating expressions of the form $\exp(-\tau \boldsymbol{A})\boldsymbol{b}$, where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is a non-negative definite matrix and $\boldsymbol{b} \in \mathbb{R}^n$ an arbitrary vector, occurs frequently in applications. The problem is equivalent, for example, to that of finding the solution to the system of ordinary differential equations

$$\dot{\boldsymbol{y}} = -\boldsymbol{A}\boldsymbol{y}, \quad \boldsymbol{y}(0) = \boldsymbol{b} \tag{1.1}$$

at time τ. Equation (1.1), in turn, arises out of finite difference or finite element discretizations of thermal problems ($\boldsymbol{A}$ symmetric) or convection-diffusion and vibration problems ($\boldsymbol{A}$ non-symmetric). Similarly, very *stiff* systems of the form (1.1) arise in predicting the time evolution of electrical circuits. Another occurrence of (1.1) is in computing the transient solution of Markov chains [22].

The calculation of a matrix exponential times a vector can be a treacherous task; see [17] for a survey of potential difficulties. Many methods have been proposed [4, 6, 9, 10, 14, 16]. For very large, sparse matrices, perhaps the preferred method [7, 8, 11–13, 18, 21] is to use the Lanczos or Arnold procedure to obtain a matrix $\boldsymbol{V}_m \in \mathbb{R}^{n\times m}$ whose columns span the Krylov subspace $\mathrm{span}\{\boldsymbol{b}, \boldsymbol{A}\boldsymbol{b}, \cdots, \boldsymbol{A}^{m-1}\boldsymbol{b}\}$, and then to write

$$\exp(-\tau \boldsymbol{A})\boldsymbol{b} \approx \boldsymbol{V}_m \exp(-\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1, \tag{1.2}$$

where $\boldsymbol{H}_m$ is the tridiagonal or Hessenberg matrix resulting from the Lanczos or Arnoldi process and $\beta = \|\boldsymbol{b}\|_2$. Since $\boldsymbol{H}_m$ will normally be much smaller than $\boldsymbol{A}$, dense matrix methods such as Padé approximations to $\exp(t)$ can be used to evaluate $\exp(-\tau \boldsymbol{H}_m)$.

Bergamaschi et al. have reported experiments that use expansions of the exponential function in Chebyshev polynomials [2, 3]. The paper convincingly argued that Chebyshev-based methods can give the same accuracy as Krylov techniques using the same polynomial degree, but that they can be much less expensive as they require no inner products. The fact

[*]Received November 11, 2008. Accepted January 4, 2010. Published online May 31, 2010. Recommended by M. Eiermann. This work was supported by NSF grants DMS 0510131 and DMS 0528492 and by the Minnesota Supercomputing Institute.

[†]Mentor Graphics Corporation, Portland, OR 97070 (`bernie_sheehan@mentor.com`).
[‡]Computer Science & Engineering, University of Minnesota, Twin Cities, MN 55455 (`saad@cs.umn.edu`).
[§]Department of Mathematics, University of Alabama, Tuscaloosa, AL 35487 (`roger.b.sidje@ua.edu`).

that this technique appears to be competitive with (1.2) prompted us to ask whether other choices of orthogonal polynomials might be used to good effect.

This work was initially motivated by an intriguing question. The use of Chebyshev polynomials requires some prior knowledge of an interval $[a, b]$ which contains the spectrum of the matrix [2, 3]. In contrast, polynomials that are orthogonal on the half real line (e.g., Laguerre) or the whole line (Hermite) would normally require no bounds. This is an important practical advantage. For example, one can ask the question: *Is it possible to employ a Laguerre series expansion to obtain an inexpensive exponential propagator which bypasses completely both inner products and eigenvalue estimates?* We will show that if used with care, Laguerre expansions combined with a good implementation of the orthogonal expansion can be quite effective. On the other hand, they are as inexpensive as methods based on Chebyshev expansions and do not require accurate eigenvalue estimates. Recall that Chebyshev-based techniques require the knowledge of an interval which is guaranteed to contain all eigenvalues. If the interval fails to contain all eigenvalues the technique will not work. A common remedy is to take a large interval containing all eigenvalues, such as one provided by the Gershgorin theorem, but often this yields poor convergence.

The techniques we examine in this paper are based on the expansion of $e^{-\tau t}$ as a series of Laguerre polynomials. The filtered conjugate residual-like algorithm (FCR) introduced in [20] provides a framework for exploiting least-squares polynomials to solve problems as diverse as regularization in graphics, information retrieval, and in electronic structure calculations. Here, we will use this framework again and show that an FCR-type algorithm can also be usefully applied to exponential propagation. At the same time, we also consider the same framework applied to expansions in Chebyshev polynomials for the purpose of comparison.

2. Computing $\exp(-\tau \boldsymbol{A})\boldsymbol{b}$ with orthogonal polynomials.

It is possible to approximate $\exp(-\tau \boldsymbol{A})\boldsymbol{b}$ by using either rational or polynomial approximations to $\exp(-\tau t)$. Methods based or rational approximations require solving large sparse linear systems of equations and are not considered here. We consider methods based on approximating $\exp(-\tau t)$ by a polynomial $p(t)$. Thus, a polynomial p_m of degree m is found which approximates $\exp(-\tau t)$, and $\exp(-\tau \boldsymbol{A})\boldsymbol{b}$ is approximated by $p_m(\boldsymbol{A})\boldsymbol{b}$. Note that the matrix $p_m(\boldsymbol{A})$ is not computed. Instead, $p_m(\boldsymbol{A})\boldsymbol{b}$ is evaluated by a series of matrix-vector multiplies using $\boldsymbol{A}$.

2.1. Orthogonal expansions for $\exp(-\tau t)$.

As background to this strategy, we briefly consider in this section how to expand $\exp(-\tau t)$ in series of generalized Laguerre polynomials and Chebyshev polynomials.

The generalized Laguerre Polynomials $L_n^\alpha(t)$, $n = 0, 1, 2, \ldots$ and $\alpha > -1$, are orthogonal with respect to the inner product

$$\langle p, q\rangle_\alpha = \int_0^\infty t^\alpha e^{-t} p(t) q(t)\, dt. \tag{2.1}$$

The expansion of the exponential function $e^{-\tau t}$ in terms of Laguerre polynomials is known to be [15, p. 90]

$$e^{-\tau t} = (\tau + 1)^{-\alpha-1} \sum_{n=0}^{\infty} \left(\frac{\tau}{\tau+1}\right)^n L_n^\alpha(t), \qquad 0 < t < \infty. \tag{2.2}$$

By truncating the above summation to only m terms, a polynomial of degree m will be obtained that will approximate $e^{-\tau t}$ over some interval. It is also possible to use expansions in Hermite polynomials, which are orthogonal on the whole real line with respect to the weight e^{-t^2}, but we will not consider this in this paper.

The Chebyshev polynomials $T_n(t)$, $n = 0, 1, 2, \ldots$, are orthogonal over the interval $[-1, 1]$ with respect to the inner product

$$\langle p, q\rangle_T = \int_{-1}^{+1} \frac{p(t)q(t)\,dt}{\sqrt{1-t^2}}. \tag{2.3}$$

In this case, it is useful to give the expansion of the exponential that is fitted in a general interval $[a, b]$, rather than just the interval $[-1, 1]$. The expansion of $e^{-\tau t}$ in terms of Chebyshev polynomials over $[a, b]$ is given by [1, Section 9.6]

$$e^{-\tau t} = \sum_{k=0}^{\infty} a_k T_k\left(\frac{t - l_2}{l_1}\right), \qquad a < t < b \quad \text{with} \tag{2.4}$$

$$a_0 = e^{-\tau l_2} I_0(-\tau l_1), \qquad a_k = 2e^{-\tau l_2} I_k(-\tau l_1), \quad k > 1. \tag{2.5}$$

Here, $I_k(t)$ is the modified Bessel function of the first kind, and $l_1 = (b - a)/2$ and $l_2 = (a + b)/2$ are the semi-width and the midpoint, respectively, of the interval over which the approximation is desired. For further information on orthogonal polynomials, the reader is referred to [1, 15].

2.2. Classical use of expansions in orthogonal polynomials. Standard orthogonal polynomials satisfy recurrence relations of the form

$$\beta_{n+1} P_{n+1}(t) = (t - \alpha_n) P_n(t) - \gamma_n P_{n-1}(t), \; n = 0, 1, \ldots, \tag{2.6}$$

with the convention that for $n = 0$ the term $\gamma_0 p_{-1}$ is zero. Thus, Chebyshev polynomials (of the first kind) satisfy the well-known recurrence $T_{n+1}(t) = 2tT_n(t) - T_{n-1}(t)$ starting with $T_0(t) = 1$ and $T_1(t) = t$. Similarly, the three-term recurrence for the generalized Laguerre polynomials is

$$\begin{aligned} -(n+1)L_{n+1}^{\alpha}(t) &= (t - \alpha - 2n - 1)L_n^{\alpha}(t) + (n + \alpha)L_{n-1}^{\alpha}(t), \quad n = 1, 2, \ldots, \\ L_0^{\alpha}(t) &= 1, \quad L_1^{\alpha}(t) = 1 + \alpha - t. \end{aligned}$$

These recurrences allow one to easily generate successive members of these orthogonal families. Assuming $e^{-\tau t}$ has the following expansion in terms of orthogonal polynomials $P_n(t)$ (see (2.2), and (2.4)–(2.5)),

$$e^{-\tau t} = \sum_{n=0}^{\infty} c_n P_n(t), \tag{2.7}$$

the following algorithm can be used to compute an approximation to $\exp(-\tau \boldsymbol{A})\boldsymbol{b}$:

ALGORITHM 2.1. $\exp(-\tau \boldsymbol{A})\boldsymbol{b}$ *by Orthogonal Expansions*

```
1. $p_0 = P_0(A)b$ ;   $p_1 = P_1(A)b$
2. $z = c_0 p_0 + c_1 p_1$
3. For $j = 1, 2, \ldots,$ Do:
4.     $p_{j+1} = (Ap_j - \alpha_j p_j - \gamma_j p_{j-1})/\beta_{j+1}$
5.     $z = z + c_{j+1} p_{j+1}$
6.     if $|c_{j+1}|\,\|p_{j+1}\| \le \epsilon$, break
7. EndDo
```

The algorithm uses (2.6) in line 4 and (2.7) in lines 2 and 5. If $\boldsymbol{A}$ is an $n \times n$ matrix with $Nz(\boldsymbol{A})$ non-zeros, then Algorithm 2.1 entails a computational cost of about $9n + 2Nz(\boldsymbol{A})$ operations per pass through the for-loop. Two vectors are needed to store $\boldsymbol{p}_j$ and $\boldsymbol{p}_{j-1}$. These are exchanged to the pair $\boldsymbol{p}_j$, $\boldsymbol{p}_{j+1}$ in the next step, which can be carried out in a constant number of operations by redirecting pointers rather than copying data. Another vector is needed to perform the matrix-vector product $\boldsymbol{A}\boldsymbol{p}_j$ and another one to store the solution $\boldsymbol{z}$. This brings the total number of vectors to four.

2.3. Filtered conjugate residual-type algorithm. An alternative way of computing $\exp(-\tau\boldsymbol{A})\boldsymbol{b}$ using orthogonal polynomials is based on the Filtered Conjugate Residual (FCR) algorithm presented in [20]. The algorithm parallels the usual Conjugate Residual algorithm [19] except that constants $\tilde{\alpha}_j$ and β_j are computed using an inner product $\langle\,,\,\rangle_w$ in function space rather than by the usual vector inner product.

In order to present the Conjugate Residual method for $\exp(-\tau A)b$ in a way that is similar to the Conjugate Residual method for linear systems, it is necessary to reformulate the problem slightly. We would like to approximate a function $\psi(t)$ by polynomials of the form $ts_k(t)$ where s_k is a polynomial of degree k. In the usual context of linear systems, $\psi(t) \equiv 1$ and $s_k(\boldsymbol{A})\boldsymbol{b}$ is the approximate solution (assuming $\boldsymbol{x}_0 = \boldsymbol{0}$), while $\boldsymbol{b} - \boldsymbol{A}s_k(\boldsymbol{A})\boldsymbol{b}$ is the residual. For the exponential function we will proceed indirectly by instead approximating $\psi(t) = 1 - e^{-\tau t}$ by a polynomial of the form $ts_k(k)$. Thus, the actual polynomial for $e^{-\tau t}$ is $\rho_k(t) = 1 - ts_k(t)$, which is the usual residual polynomial in the context of linear systems. One notable advantage of this approach compared to that of Algorithm 2.1 is that the approximation will be exact at $t = 0$. In other words, $\rho_k(0) = \exp(-\tau \cdot 0) = 1$. This property may or may not be important depending on the situation. For example, when solving ordinary differential equations, this property is vital as it ensures that the underlying integration scheme is at least first-order accurate.

Consider the (functional) inner product

$$\langle p, q\rangle_w = \int_a^b p(t)q(t)w(t)\,dt,$$

where $w(t) \geq 0$ is some weight function. This inner product induces a weighted L_2-norm $\|p\|_w = \langle p, p\rangle^{1/2}$. The goal is to find the polynomial of the form $ts_k(t)$ which is closest to $\psi(t)$ in the sense of this L_2 norm, i.e., such that

$$\|\psi - ts_k(t)\|_w = \min_{s\in\mathcal{P}_k} \|\psi - ts(t)\|_w.$$

The algorithm described in [20] exploits an analogy with what is known when solving linear systems. The Conjugate Residual (CR) algorithms to minimize $\|\boldsymbol{b} - \boldsymbol{A}s(\boldsymbol{A})\boldsymbol{b}\|_2$, is as follows:

ALGORITHM 2.2. *Conjugate Residual Algorithm*

```
0. Compute $\boldsymbol{r}_0 := \boldsymbol{b}$ (starts with $\boldsymbol{x}_0 = \boldsymbol{0}$), $\boldsymbol{p}_0 := \boldsymbol{r}_0$
2. For $j = 0, 1, \ldots,$ until convergence Do:
3.     $\alpha_j := \langle \boldsymbol{r}_j, \boldsymbol{A}\boldsymbol{r}_j\rangle / \langle \boldsymbol{A}\boldsymbol{p}_j, \boldsymbol{A}\boldsymbol{p}_j\rangle$
4.     $\boldsymbol{x}_{j+1} := \boldsymbol{x}_j + \alpha_j \boldsymbol{p}_j$
5.     $\boldsymbol{r}_{j+1} := \boldsymbol{r}_j - \alpha_j \boldsymbol{A}\boldsymbol{p}_j$
6.     $\beta_j := \langle \boldsymbol{r}_{j+1}, \boldsymbol{A}\boldsymbol{r}_{j+1}\rangle / \langle \boldsymbol{r}_j, \boldsymbol{A}\boldsymbol{r}_j\rangle$
7.     $\boldsymbol{p}_{j+1} := \boldsymbol{r}_{j+1} + \beta_j \boldsymbol{p}_j$
8. EndDo
```

This algorithm generates a set of vectors $\{\boldsymbol{A}\boldsymbol{p}_j\}$ which are orthogonal. Since $\boldsymbol{p}_j$ belongs to the Krylov subspace K_j, one can associate canonically with the sequence $\boldsymbol{p}_j$ a sequence of polynomials π_j, where π_j is of degree j. The relation between $\boldsymbol{p}_j$ and π_j is $\boldsymbol{p}_j = \pi_j(\boldsymbol{A})\boldsymbol{b}$. The sequence of polynomials $\{t\pi_j(t)\}$ is also orthogonal with respect to the discrete inner product $\langle \boldsymbol{p}, \boldsymbol{q}\rangle_b = \langle \pi_p(\boldsymbol{A})\boldsymbol{b}, \pi_q(\boldsymbol{A})\boldsymbol{b}\rangle$, where we have denoted by π_p the polynomial associated with the vector $\boldsymbol{p}$ in K_m.

If we replace the discrete inner product $\langle \pi_p, \pi_q\rangle_b$ with any inner product $\langle p, q\rangle_w$, we will obtain an algorithm which formally constructs a sequence $\{tp_j\}$ of polynomials which are orthogonal with respect to this inner product. The polynomial $ts_k(t)$ which is the closest to ψ it the w-norm sense can then easily be determined:

$$\min_{s\in\mathcal{P}_k} \|\psi - ts(t)\|_w = \|\psi - ts_k(t)\|_w \quad \text{with} \quad s_k(t) = \sum_{i=0}^{k} \frac{\langle \psi, t\pi_i\rangle_w}{\langle t\pi_i, t\pi_i\rangle_w}\pi_i(t).$$

The filtered Conjugate Residual algorithm proposed in [20] computes the two sequence of polynomials $\{\pi_j\}$, $\{\rho_j\}$ associated with the usual CR algorithm. These would normally be the scalars and polynomials needed for the case when $\psi = 1$. They are indicated with a tilde in the following algorithm. In addition, we will need to generate the above quantities which represent the expansions coefficients of the polynomial in the basis $\{t\pi_j\}$. These are denoted by α_j in the algorithm.

Thus, the updates to $\boldsymbol{x}_j$ use a different coefficient α_j than the coefficient $\tilde{\alpha}_j$ used to update $\tilde{\boldsymbol{r}}_j$. Inputs to the algorithm are a filter function ψ, an inner product $\langle p, q\rangle_w$, a matrix $\boldsymbol{A}$, and a vector $\boldsymbol{b}$:

ALGORITHM 2.3. *Filtered Conjugate Residual Polynomials Algorithm*

```
0.  Compute $\tilde{r}_0 := b - Ax_0$, $p_0 := \tilde{r}_0$          $\pi_0 = \tilde{\rho}_0 = 1; s_0 = 0$
1.                                                      Compute $\lambda\pi_0$
2.  For $j = 0, 1, \ldots,$ until convergence Do:
3.      $\tilde{\alpha}_j := \langle \tilde{\rho}_j, \lambda\tilde{\rho}_j\rangle_w / \langle \lambda\pi_j, \lambda\pi_j\rangle_w$
4.      $\alpha_j := \langle \psi, \lambda\pi_j\rangle_w / \langle \lambda\pi_j, \lambda\pi_j\rangle_w$
5.      $x_{j+1} := x_j + \alpha_j p_j$                      $s_{j+1} = s_j + \alpha_j\pi_j$
6.      $\tilde{r}_{j+1} := \tilde{r}_j - \tilde{\alpha}_j A p_j$                 $\tilde{\rho}_{j+1} = \tilde{\rho}_j - \tilde{\alpha}_j\lambda\pi_j$
7.      $\beta_j := \langle \tilde{\rho}_{j+1}, \lambda\tilde{\rho}_{j+1}\rangle_w / \langle \tilde{\rho}_j, \lambda\tilde{\rho}_j\rangle_w$
8.      $p_{j+1} := \tilde{r}_{j+1} + \beta_j p_j$                  $\pi_{j+1} := \tilde{\rho}_{j+1} + \beta_j\pi_j$
9.                                                      Compute $\lambda\pi_{j+1}$
10. EndDo
```

Here $\pi_j(\lambda), \tilde{\rho}_j(\lambda)$, and $s_{j+1}(\lambda)$ are polynomials of degree j and the vectors $\boldsymbol{p}_j$, $\tilde{\boldsymbol{r}}_j$, $\boldsymbol{x}_j$ are the corresponding sequences of vectors

$$\begin{aligned} \boldsymbol{p}_j &= \pi_j(\boldsymbol{A})\boldsymbol{r}_0, \\ \tilde{\boldsymbol{r}}_j &= \tilde{\rho}_j(\boldsymbol{A})\boldsymbol{r}_0, \\ \boldsymbol{x}_{j+1} &= \boldsymbol{x}_0 + s_{j+1}(\boldsymbol{A})\boldsymbol{r}_0, \end{aligned}$$

where $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$.

The solution vector $\boldsymbol{x}_{j+1}$ computed at the jth step of Algorithm 2.3 is of the form $\boldsymbol{x}_{j+1} = \boldsymbol{x}_0 + s_{j+1}(\boldsymbol{A})\boldsymbol{r}_0$, where s_j is the jth degree polynomial:

$$s_{j+1}(t) = \alpha_0\pi_0(t) + \cdots + \alpha_j\pi_j(t)\ . \tag{2.8}$$

The polynomials π_j and the auxiliary polynomials $\tilde{\rho}_j(t)$ satisfy the orthogonality relations [20],

$$\langle t\pi_j(t), t\pi_i(t)\rangle_w = \langle t\tilde{\rho}_j(t), \tilde{\rho}_i(t)\rangle_w = 0 \quad \text{for} \quad i \neq j\ . \tag{2.9}$$

In addition, the filtered residual polynomial $\psi(t) - ts_j(t)$ minimizes $\|\psi - ts(t)\|_w$ among all polynomials s of degree $\leq j - 1$. Note also that if $\phi = 1 - \psi$, then the polynomial $\zeta_j(t) = 1 - ts_j$ minimizes $\|\phi - \zeta_j\|_w$ among all polynomials $p_j(t) \in \mathcal{P}_j$ satisfying $p_j(0) = 1$.

It is worth remarking that the π_j generated by Algorithm 2.3 are the orthogonal polynomials associated with the weight $t^2 w(t)$; this result follows from (2.9). When the FCR algorithm with Laguerre polynomials $L_n^\alpha(t)$ is used, for example, then the π_j are the generalized Laguerre polynomials $L_n^{\alpha+2}(t)$. This observation will be exploited shortly.

Note that the FCR algorithm has a cost per iteration of three vector saxpy operations and one matrix-vector multiply. All other operations are with polynomials and these are usually negligible. They include two polynomial saxpys (the calculation of s_{j+1} on line 5 is not necessary but has been inserted for clarity), three polynomial inner-products, and one polynomial multiply by t. If we write ψ, π_j, and $\tilde{\rho}_j$ using as basis the orthogonal polynomials associated with $\langle\ ,\ \rangle_w$, then we can evaluate a polynomial inner-product at a cost of $2j$; the t-polynomial multiply will also incur a cost of $6j$ (by employing the three-term recurrence relation for the orthogonal polynomials to express $t\pi_{j+1}$ in the orthogonal polynomial basis). In short, the overall cost per loop is $6n + Nz(\boldsymbol{A}) + 16j$, j being the loop index and $Nz(\boldsymbol{A})$ the number of non-zeros in $\boldsymbol{A}$. To add a termination test like $\|\boldsymbol{x}_{j+1} - \boldsymbol{x}_j\| < \epsilon$, analogous to line 6 in Algorithm 2.1, would increase the iteration cost by another $2n$. FCR requires storage of $4n + 3j$ (the four vectors $\boldsymbol{x}_j$, $\tilde{\boldsymbol{r}}_j$, $\boldsymbol{A}\boldsymbol{p}_j$, and $\boldsymbol{p}_j$, and coefficients for the three polynomials $\tilde{\rho}_j, \pi_j, t\pi_j$). Our bookkeeping suggests that Algorithm 2.3 may be marginally faster but requires marginally more memory than Algorithm 2.1, provided j remains small compared to n.

In order to use FCR to compute $\exp(-\tau\boldsymbol{A})\boldsymbol{b}$, let $\phi = \exp(-\tau t)$. Then, with $\boldsymbol{x}_0 = \boldsymbol{0}$, apply the FCR algorithm to get $\boldsymbol{x}_j = \sum_{k=0}^{j-1} \alpha_k \pi_k(\boldsymbol{A})\boldsymbol{b}$. Finally, compute $\boldsymbol{z}_j = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_j = (\boldsymbol{I} - \boldsymbol{A}s_j(\boldsymbol{A}))\boldsymbol{b} = \zeta_j(\boldsymbol{A})\boldsymbol{b}$ as an approximation for $\phi(\boldsymbol{A})\boldsymbol{b} = \exp(-\tau\boldsymbol{A})\boldsymbol{b}$. If one wants the current estimate for $\exp(-\tau\boldsymbol{A})\boldsymbol{b}$ at each step in the iteration, one can replace line (5) in Algorithm 2.3 by

$$\boldsymbol{z}_{j+1} = \boldsymbol{z}_j - \alpha\boldsymbol{A}\boldsymbol{p}_j.$$

If $\boldsymbol{A}$ is symmetric with eigen-decomposition $\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T$, then

$$\begin{aligned}
\|\exp(-\tau\boldsymbol{A})\boldsymbol{b} - \boldsymbol{z}_j\|_2 &= \|\phi(\boldsymbol{A})\boldsymbol{b} - \zeta_j(\boldsymbol{A})\boldsymbol{b}\|_2 \\
&= \|\boldsymbol{V}(\phi(\boldsymbol{\Lambda}) - \zeta_j(\boldsymbol{\Lambda})\boldsymbol{V}^T\boldsymbol{b}\|_2 \\
&\le \max_i |\phi(\lambda_i) - \zeta_j(\lambda_i)| \cdot \|\boldsymbol{b}\|_2.
\end{aligned}$$

The hope is that if $\|\phi(t) - \zeta_j(t)\|_w$ is small over some interval $[a, b]$ containing the spectrum of $\boldsymbol{A}$, then $\max_i |\phi(t_i) - \zeta_j(t_i)|$ will also be small, although it is hard to guarantee this since $\|\quad\|_w$ is only a least squares norm.

The simplest criterion for stopping the algorithm is to measure the difference between two consecutive iterates, so one can stop whenever $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2 \le \|\boldsymbol{x}_k\|\epsilon$, where ϵ is some tolerance.

2.4. Use of modified orthogonal polynomials. As was already mentioned, an important advantage of the procedure presented in the previous section, relative to the straightforward procedure based on orthogonal expansions presented in Section 2.2, is that it matches exactly the exponential at $t = 0$. The FCR procedure is rather general and has been used for ψ functions defined as very general spline functions. One may ask whether or not it is possible to derive an algorithm that is equivalent to Algorithm 2.3, but which resembles Algorithm 2.1. This can be done for the exponential function by exploiting the remark made earlier about the choice of α.

In order to minimize $\|\psi - tq(t)\|_\alpha$ over all polynomials q of degree k, we need a sequence of polynomials of the specific form $tp_i(t)$ which are orthogonal with respect to the inner product $\langle\ ,\ \rangle_\alpha$. At this point we recall an observation made earlier: *the sequence of polynomials* $\{tL_i^{\alpha+2}(t)\}$ *is orthogonal with respect to the inner product* $\langle\ ,\ \rangle_\alpha$. This is because for $i \neq j$,

$$\begin{aligned}\langle tL_i^{\alpha+2}, tL_j^{\alpha+2}\rangle_\alpha &= \int_0^\infty t^\alpha e^{-t} t^2 L_i^{\alpha+2}(t) L_j^{\alpha+2}(t)\,dt \\ &= \int_0^\infty t^{\alpha+2} e^{-t} L_i^{\alpha+2}(t) L_j^{\alpha+2}(t)\,dt = 0.\end{aligned}$$

As a result, the least-squares polynomial approximation to ψ is given by

$$ts_k(t) = \sum_{i=0}^{k} c_i t L_i^{\alpha+2}(t) \quad \text{with} \quad c_i = \frac{\langle \psi,\ tL_i^{\alpha+2}\rangle_\alpha}{\langle tL_i^{\alpha+2},\ tL_i^{\alpha+2}\rangle_\alpha}.$$

The following proposition defines a recurrence relation to compute the coefficients c_i.

PROPOSITION 2.1. *Let* $\psi(t) = 1 - e^{-\tau t}$*, and let the three-term recurrence of the generalized Laguerre polynomials* $L_i^{\alpha+2}$ *be written in the form (2.6). Define* $\sigma_i = (i+\alpha+2)/i$ *for* $i > 0$, $\sigma_0 = 1$*. Then the polynomial of degree* $k+1$ *of the form* $ts(t)$ *which minimizes the norm* $\|\psi - ts(t)\|_\alpha$ *among all polynomials* s *of degree* $\leq k$ *is given by*

$$ts_k(t) = \sum_{i=0}^{k} c_i\, tL_i^{\alpha+2}(t), \tag{2.10}$$

where c_i *satisfies the recurrence relation*

$$c_{i+1} = \frac{1}{\sigma_{i+1}\beta_{i+1}}\left[\delta_{i0} - \frac{\tau}{(1+\tau)^{\alpha+4}} - \alpha_i c_i - \frac{\gamma_i}{\sigma_i} c_{i-1}\right], i = 0, 1, \ldots, \tag{2.11}$$

$$c_0 = \frac{1}{\alpha+2}\left[1 - \frac{1}{(1+\tau)^{\alpha+2}}\right], \tag{2.12}$$

where δ_{ij} *is the Kronecker symbol and we define* $\gamma_0 c_{-1}/\sigma_0 \equiv 0$*.*

Proof. The least-squares polynomial approximation to ψ is given by

$$\begin{aligned} ts_k(t) &= \sum_{i=0}^{k} c_i t L_i^{\alpha+2}(t) \quad \text{with} \quad c_i = \frac{\tilde{c}_i}{d_i} \quad \text{and} \\ \tilde{c}_i &= \int_0^\infty t^\alpha e^{-t}\psi(t)\, tL_i^{\alpha+2}(t)\,dt, \quad d_i = \int_0^\infty t^\alpha e^{-t}(tL_i^{\alpha+2}(t))^2 dt\,. \end{aligned}$$

Consider d_i first and observe that

$$d_i = \int_0^\infty t^{\alpha+2} e^{-t}[L_i^{\alpha+2}(t)]^2 dt = \|L_i^{\alpha+2}(t)\|_{\alpha+2}^2.$$

It is known that

$$\|L_i^\alpha\|_\alpha^2 = \frac{\Gamma(i+\alpha+1)}{i!}.$$

So $d_i = \|L_i^{\alpha+2}(t)\|_{\alpha+2}^2 = \frac{\Gamma(i+\alpha+3)}{i!}$.

Next, we take up the task of calculating $\tilde{c}_i$:

$$\tilde{c}_i = \int_0^\infty t^{\alpha+1} e^{-t} [1 - e^{-\tau t}] L_i^{\alpha+2}(t)\, dt \tag{2.13}$$

$$= \int_0^\infty t^{\alpha+2} e^{-t} \frac{1 - e^{-\tau t}}{t} L_i^{\alpha+2}(t)\, dt\ . \tag{2.14}$$

There are two possible ways to proceed from here. One is to attempt to calculate the above integral. This can be done by successive integration by parts using the expression

$$L_i^\alpha(t) = e^t \frac{t^{-\alpha}}{i!} \frac{d^i}{dt^i} \left(e^{-t} t^{i+\alpha}\right).$$

Define the function $\phi_\tau(t) \equiv (1 - e^{-\tau t})/t$. With this, $\tilde{c}_i$ becomes

$$\tilde{c}_i = \frac{1}{i!} \int_0^\infty \phi_\tau(t) \frac{d^i}{dt^i} \left(e^{-t} t^{i+\alpha+2}\right)\ dt\ .$$

This integral can be evaluated with a succession of integration by parts leading to results involving successive derivatives of $\phi_\tau(t)$. This yields a rather complicated expression for $\tilde{c}_i$.

An alternative is to exploit the recurrence relation for the orthogonal polynomials, which we assume is in the form (2.6). It is more convenient to deal with the scaled quantities c_i directly rather than the unscaled $\tilde{c}_i$. We begin by splitting the calculation into two parts. From (2.13), we have

$$c_i = \frac{1}{d_i} \int_0^\infty t^{\alpha+1} e^{-t} L_i^{\alpha+2}(t)\, dt - \frac{1}{d_i} \int_0^\infty t^{\alpha+1} e^{-(1+\tau)t} L_i^{\alpha+2}(t)\, dt \equiv f_i - g_i.$$

We seek to establish recurrence relations for g_i and f_i separately. Consider the case $i = 0$. For $i = 0$, we have $L_0^{\alpha+2}(t) = 1$, so

$$f_0 = \frac{1}{d_0} \int_0^\infty t^{\alpha+1} e^{-t} dt; \qquad g_0 = \frac{1}{d_0} \int_0^\infty t^{\alpha+1} e^{-(1+\tau)t} dt\ .$$

Using the standard definition of the Gamma function, we note that (for $\beta > 0$),

$$\int_0^\infty e^{-t} t^z dt = \Gamma(z+1); \qquad \int_0^\infty e^{-\beta t} t^z dt = \frac{\Gamma(z+1)}{\beta^{z+1}}.$$

Therefore, recalling that $d_0 = \Gamma(\alpha+3) = (\alpha+2)\Gamma(\alpha+2)$, we have

$$f_0 = \frac{1}{d_0} \Gamma(\alpha+2) = \frac{1}{\alpha+2}\ ; \qquad g_0 = \frac{1}{d_0} \frac{\Gamma(\alpha+2)}{(1+\tau)^{\alpha+2}} = \frac{1}{(\alpha+2)(1+\tau)^{\alpha+2}}\ . \tag{2.15}$$

For a general i we have for f_{i+1},

$$\begin{aligned} d_{i+1} f_{i+1} &= \int_0^\infty t^{\alpha+1} e^{-t} L_{i+1}^{\alpha+2}(t)\, dt \\ &= \frac{1}{\beta_{i+1}} \int_0^\infty t^{\alpha+1} e^{-t} \left[t L_i^{\alpha+2}(t) - \alpha_i L_i^{\alpha+2}(t) - \gamma_i L_{i-1}^{\alpha+2}(t)\right] dt. \end{aligned}$$

Apart from the scaling by β_{i+1}, the first term on the right-hand side,

$$\int_0^\infty t^{\alpha+1} e^{-t} t L_i^{\alpha+2}(t)\, dt = \int_0^\infty t^{\alpha+2} e^{-t} L_i^{\alpha+2}(t)\, dt,$$

is either equal to $\Gamma(\alpha+3) = d_0$ when $i=0$, or to zero for $i>0$ by the orthogonality of the functions $L_i^{\alpha+2}$ and 1. Therefore, we have:

$$d_1 f_1 = \frac{d_0 - \alpha_0 d_0 f_0}{\beta_1}\,, \qquad d_{i+1} f_{i+1} = \frac{-\alpha_i d_i f_i - \gamma_i d_{i-1} f_{i-1}}{\beta_{i+1}} \quad \text{for } i>0\,.$$

As a notational convenience we define $\sigma_i = d_i/d_{i-1}$ and note that

$$\sigma_i = \frac{\Gamma(i+\alpha+3)}{i!}\,\frac{(i-1)!}{\Gamma(i+\alpha+2)} = \frac{i+\alpha+2}{i}\,. \tag{2.16}$$

With this, the above formulas become,

$$f_1 = \frac{1-\alpha_0 f_0}{\sigma_1 \beta_1}\,, \qquad f_{i+1} = -\frac{\alpha_i f_i + \frac{\gamma_i}{\sigma_i} f_{i-1}}{\sigma_{i+1}\beta_{i+1}} \quad \text{for } i>0,$$

which we write as a single formula by using the Kronecker symbol and setting $\gamma_0 f_{-1} \equiv 0$:

$$f_{i+1} = \frac{1}{\sigma_{i+1}\beta_{i+1}}\left[\delta_{i,0} - \alpha_i f_i - \frac{\gamma_i}{\sigma_i} f_{i-1}\right] \quad \text{for } i \geq 0. \tag{2.17}$$

We can proceed in a similar way for g_i:

$$\begin{aligned} d_{i+1} g_{i+1} &= \int_0^\infty t^{\alpha+1} e^{-(1+\tau)t} L_{i+1}^{\alpha+2}(t)\,dt \\ &= \frac{1}{\beta_{i+1}} \int_0^\infty t^{\alpha+1} e^{-(1+\tau)t} [t L_i^{\alpha+2}(t) - \alpha_i L_i^{\alpha+2}(t)(t) - \gamma_i L_{i-1}^{\alpha+2}(t)] dt\,. \end{aligned}$$

Now the (undivided) first term in the right-hand side is

$$\int_0^\infty t^{\alpha+2} e^{-t} e^{-\tau t} L_i^{\alpha+2}(t)\,dt.$$

Apart from a norm scaling factor, this is the ith expansion coefficient of the function $e^{-\tau t}$ in the Laguerre orthogonal sequence $\{L_i^{\alpha+2}\}$, i.e., with respect to the ith degree polynomial. Specifically, from (2.2), we know that

$$\frac{\langle e^{-\tau t}, L_i^{\alpha+2}\rangle_{\alpha+2}}{\langle L_i^{\alpha+2}, L_i^{\alpha+2}\rangle_{\alpha+2}} = \frac{\tau}{\tau+1}(1+\tau)^{-\alpha-3} \rightarrow \langle e^{-\tau t}, L_i^{\alpha+2}\rangle_{\alpha+2} = \frac{\tau d_i}{(1+\tau)^{\alpha+4}}.$$

Therefore,

$$d_{i+1} g_{i+1} = \frac{1}{\beta_{i+1}}\left[\frac{\tau d_i}{(1+\tau)^{\alpha+4}} - \alpha_i d_i g_i - \gamma_i d_{i-1} g_{i-1}\right].$$

Finally, dividing through by d_i and rearranging terms,

$$g_{i+1} = \frac{1}{\sigma_{i+1}\beta_{i+1}}\left[\frac{\tau}{(1+\tau)^{\alpha+4}} - \alpha_i g_i - \frac{\gamma_i}{\sigma_i} g_{i-1}\right]. \tag{2.18}$$

It is now possible to combine (2.18), (2.17), into a single recurrence formula for c_i, recalling that that $c_i = f_i - g_i$:

$$c_{i+1} = \frac{1}{\sigma_{i+1}\beta_{i+1}}\left[\delta_{i0} - \frac{\tau}{(1+\tau)^{\alpha+4}} - \alpha_i c_i - \frac{\gamma_i}{\sigma_i} c_{i-1}\right], i = 0, 1, \ldots,$$

where δ_{ij} is the Kronecker symbol. The recurrence can be started with $i=0$, with the convention that $\gamma_0 c_{-1} \equiv 0$. The initial value c_0 is known from (2.15):

$$c_0 = \frac{1}{\alpha+2}\left[1 - \frac{1}{(1+\tau)^{\alpha+2}}\right].$$

This completes the proof. □

3. Convergence and stability.

3.1. Numerical aspects. In practice, the effectiveness of the algorithms hinges on their stability behavior and the rate at which the underlying orthogonal series expansions converge to the exponential function. Recall at this point that the conventional reservation about the Taylor series stems essentially from the fact that on one hand its convergence may require summing a large number of terms, and on the other hand its sum may involve intermediate quantities of large magnitude and/or alternate sign, introducing cancellation errors and stability issues in finite precision arithmetic. Whether the orthogonal series expansions are numerically robust depends on how well they can withstand such drawbacks. In this section, we use a number of figures to explore these issues. We include the Chebyshev series in the discussion for comparison purposes. To keep the figures and the discussion simple, we set $\tau = 1$ in (2.2) and (2.4), and denote by $p_m(x)$ the partial sum of the series of e^{-x} truncated at length m. In the case of the Laguerre series (with $\alpha = 0$),

$$p_m(x) = \frac{1}{2}\sum_{k=0}^{m} \frac{1}{2^k} L_k(x), \qquad 0 < x < \infty,$$

while in the case of the Chebyshev series,

$$p_m(x) = \sum_{k=0}^{m} a_k T_k\left(\frac{x - l_2}{l_1}\right), \qquad a < x < b,$$

with $l_1 = (b-a)/2$, $l_2 = (a+b)/2$, and the coefficients a_k as given earlier in (2.5). We first focus on the real case, bearing in mind that we can always scale a nonnegative definite matrix and use a time-stepping procedure to confine the eigenvalues to a fixed interval.

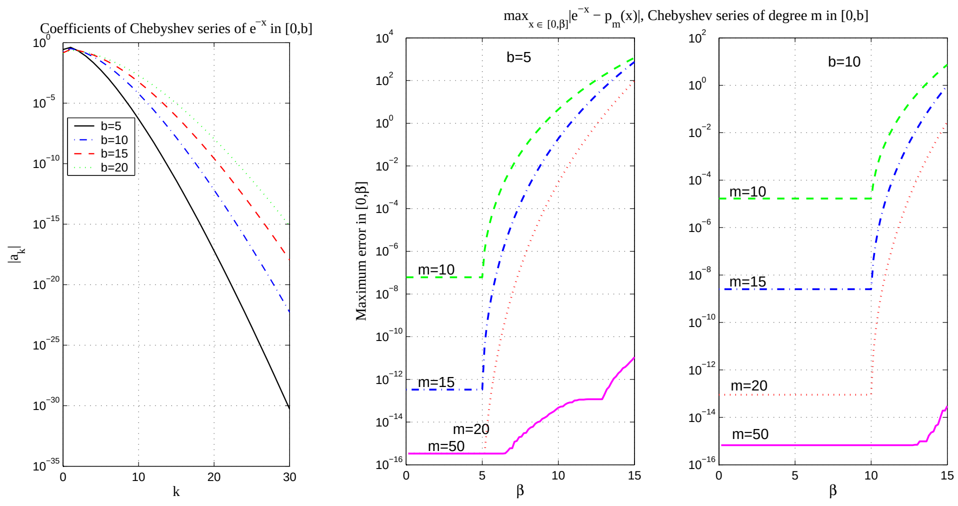

FIG. 3.1. *Coefficients and approximation error of the Chebyshev series.*

To give the reader a sense of how well the Chebyshev series performs, we consider different choices of the target interval $[a, b]$ and plot in Figure 3.1 the evolution of the coefficients a_k. It is seen that the coefficients decrease very rapidly in magnitude, and considering that $\max_{x\in[a,b]} |T_k(x)| \le 1$, we can expect the method to remain stable and converge rapidly for any $x \in [a, b]$. This is indeed what the error curves in the figure show. These observations corroborate [3, 23]. The error curves also remind us that the target interval has to be chosen carefully to suit the problem at hand, because the approximations degrade rapidly outside their intended interval. Thus, in our matrix exponential context, the eigenvalues need to be strictly inside the interval to prevent spurious effects. Increasing the degree m to take in more terms does not expand the fit interval, but does improve the quality of fit within the target interval. All this agrees with the theory, because the Chebyshev series is, by construction, aimed at a (fixed) given interval. Once constructed, the series does well for any point taken in that interval and it gets better as we use more terms in the series. But the series is not effective outside the target interval for which it is built. Targeting a different interval requires building another series, which is why we need an estimate of the spectral interval for the Chebyshev approximation to be effective.

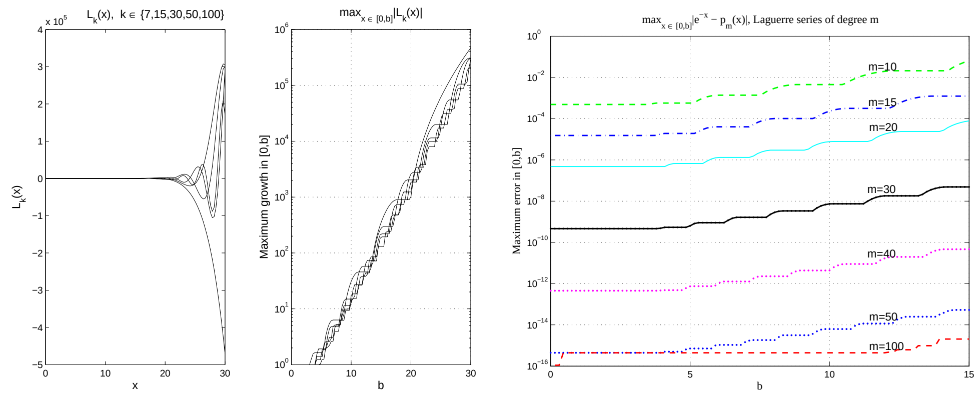

FIG. 3.2. *Laguerre polynomials and approximation error of the Laguerre series.*

In the case of the Laguerre series, an immediate contrast with the Chebyshev series is that the Laguerre polynomials $L_k(x)$ are not bounded by unity, as can be seen on Figure 3.2, even though the coefficients of the Laguerre series are $1/2^{k+1}$. We may wonder if the Laguerre series may be susceptible to numerical difficulties. We see from the figure that this is unlikely because the Laguerre polynomials $L_k(x)$ do not oscillate widely with each other in the interval $[0, 20]$, which is an interval sufficient for practical purposes (considering the time-stepping strategy hinted to earlier and to be further discussed shortly). Moreover, as the figure shows, their maximum growth stays within 10^3 in this interval, so there is no particular concern in double precision arithmetic. Regarding the speed, we see that the rate of convergence of the Laguerre series is less than that of the Chebyshev series, meaning that if a prescribed accuracy is desired, it takes more terms to achieve that accuracy with the Laguerre series. For example, comparing Figure 3.2 with Figure 3.1, one must go up to $m = 40$ terms in the Laguerre series to get a fit comparable to the Chebyshev series with $m = 15$ terms for the interval $[0, 5]$, or $m = 20$ terms for the interval $[0, 10]$. The trade-off when computing the matrix exponential is that the Laguerre series does not need an estimate of the bounds of the spectrum as the Chebyshev series does.

As noted earlier, the FCR-based Laguerre series enforces the exactness of the approximation at $x = 0$. We turn our attention now to comparing it with the classical Laguerre series.

In Figure 3.3, we can see that both series remain very close around the origin, but as we move away, a difference becomes manifest. The onset of the divergence between the two curves moves to the right as more terms are added. Slight differences can also be seen between the maximum error curves in Figure 3.2 and Figure 3.3. For example in the interval

ETNA
Kent State University
http://etna.math.kent.edu

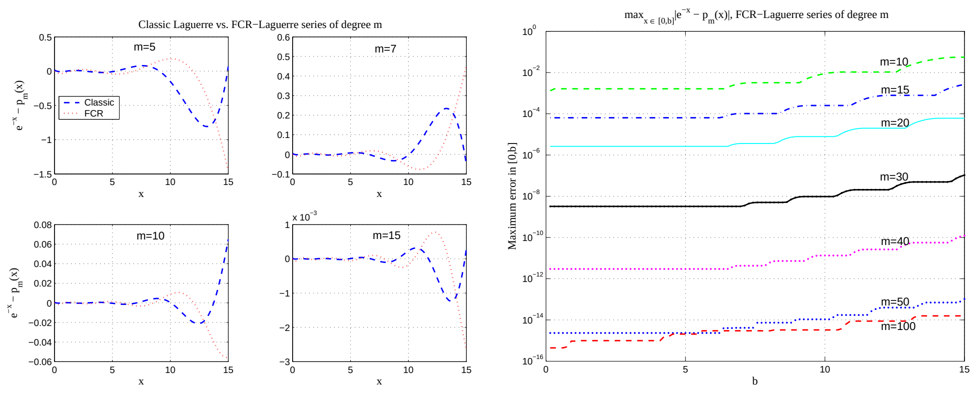

FIG. 3.3. *Variation between the classical Laguerre and FCR-Laguerre, and approximation error of FCR-Laguerre.*

$[0, 5]$, we can see in the former that the maximum error curve for $m = 20$ is slightly under 10^{-6}, whereas in the latter it is slightly over 10^{-6}. The discrepancy is a natural consequence of using different optimality criteria for the expansions. However, the differences are too small to be significant in our context, and we shall use either variant in our experiments.

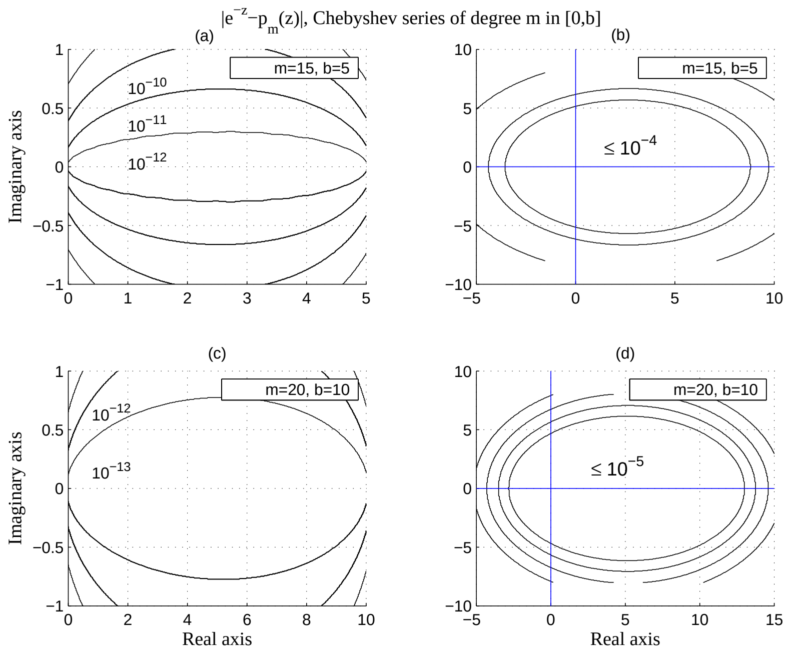

FIG. 3.4. *Approximation error of the Chebyshev series in the complex plane.*

We now consider the case where the matrix involved may have complex eigenvalues. Although the orthogonal series considered in this study are primarily designed to approximate the exponential of a real variable, it is worth exploring how these series will behave if the real variable x is replaced by a complex variable z. This will give us a sense of how the algorithms will behave when used with nonsymmetric matrices that are barely indefinite.

In Figure 3.4 and Figure 3.5, we plot the level curves of the error $|e^{-z} - p_m(z)|$ in

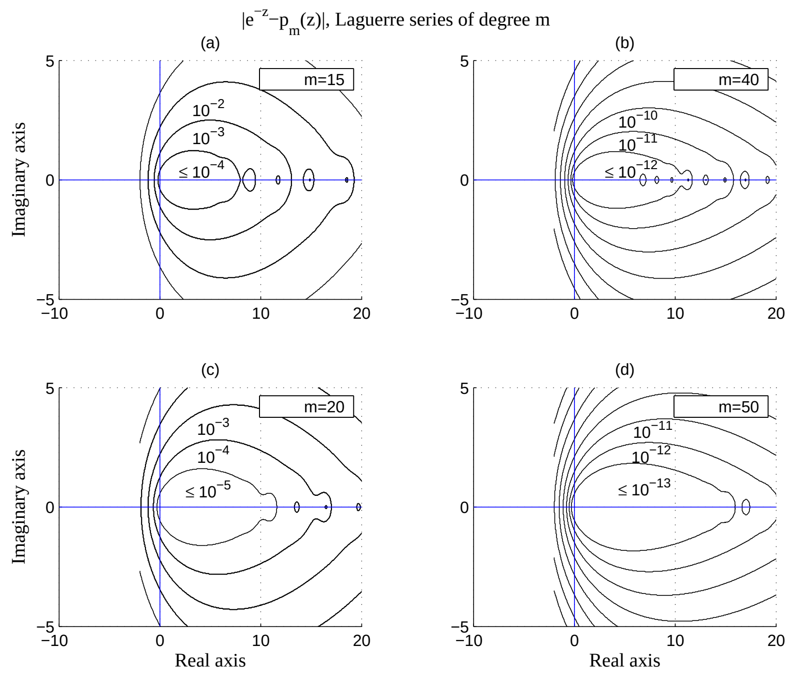

FIG. 3.5. *Approximation error of the Laguerre series in the complex plane.*

the complex plane using $m \in \{15, 20\}$ and $b \in \{5, 10\}$ for the Chebyshev series, and $m \in \{15, 20, 40, 50\}$ for the Laguerre series. In the Chebyshev case in Figure 3.4, where $m \in \{15, 20\}$, the subplots (a) and (c) give the level curves at a finer resolution, whereas the subplots (b) and (d) aggregate the regions where the error is under 10^{-4} and 10^{-5}, respectively. In the Laguerre case in Figure 3.5, the subplots (a) and (c) use $m \in \{15, 20\}$ and aggregate the regions where the error is under 10^{-4} and 10^{-5}, whereas (b) and (d) use the higher degrees $m \in \{40, 50\}$ and aggregate the regions under 10^{-12} and 10^{-13}, respectively. These contour plots allow us to contrast the two methods either from the perspective of equal degree or accuracy. We can see that the observations made in the real case apply also in the complex case. For the Chebyshev series, the region of good fit spreads around the target interval. For the Laguerre series, it is anchored at the origin and gradually grows toward the right as more terms are added to the expansion. In both methods, the region only extends slightly in the imaginary direction, or alternatively, it takes more terms to attain a certain accuracy in the complex case. Of note is that an accuracy of 10^{-4} or 10^{-5} can be achieved by both methods in comparable regions with the same degree $m = 15$ or $m = 20$, respectively. However, the Chebyshev series is appreciably more accurate around the real axis, as is apparent in the subplots (a) and (c) of Figure 3.4. On the whole, neither method is ideally suited for evaluating the matrix exponential at a high accuracy when the matrix has eigenvalues with large imaginary parts, unless a proper scaling is made to shrink the spectrum. This is discussed next.

3.2. Scaling and staging. This strategy is aimed not only at improving the accuracy by confining the spectrum to a more desirable domain for a given degree of the partial sum, but also at avoiding numerical overflow when summing the terms of the series. It consists of scaling the matrix by a number, say n_{stage}, such that $\|\tau \boldsymbol{A}\|/n_{stage} \lesssim b$, where $\|\cdot\|$ is some norm, and b is chosen to target an appropriate interval based on our discussion above. From there, the calculation of $\exp(-\tau \boldsymbol{A})\boldsymbol{b}$ is broken into n_{stage} steps as described below.

ALGORITHM 3.1. *Scaled and Staged Calculation of* $\exp(-\tau \boldsymbol{A})\boldsymbol{b}$

```
1. Choose $n_{stage}$ such that $\|\tau \boldsymbol{A}\|/n_{stage} \lesssim b$
2. $\hat{\tau} = \tau / n_{stage}$
3. $\hat{\boldsymbol{A}} = \hat{\tau}\boldsymbol{A}$
4. $\boldsymbol{z}_0 = \boldsymbol{b}$
5. For $t = 1 : n_{stage}$
6.     $\boldsymbol{z}_t = \exp(-\hat{\boldsymbol{A}})\boldsymbol{z}_{t-1}$
7. EndFor
```

This algorithm simply amounts to a (constant) time-stepping strategy as typically used when solving ordinary differential equations. Here, however, we effectively *scale* the matrix. The implication is that, first, the scaling $\hat{\tau}\boldsymbol{A} = \tau\boldsymbol{A}/n_{stage}$ shrinks the spectrum (with $\boldsymbol{A}$ assumed symmetric nonnegative definite) into an interval $[0, b]$; and second, subdividing the calculations into these n_{stage} stages ensures that each stage uses an orthogonal series approximation for $e^{-\hat{\tau}t}$ that converges after relatively few terms in the partial sum.

4. Numerical results.

In this section we apply the FCR approach described above to some sample calculations of $\exp(-\tau \boldsymbol{A})\boldsymbol{b}$. All the codes are implemented in MATLAB. We run the series until two consecutive estimates satisfy $\|\boldsymbol{z}_{k+1} - \boldsymbol{z}_k\|_2 \leq \mathrm{TOL}$, with preset values of the accuracy parameter TOL. We compare with a Krylov method (the *expv.m* function from the Expokit package [21]), which implements Arnoldi's Full Orthogonalization Method (FOM), and we set 15 as the dimension of the Krylov basis. This is referred to as Krylov(15) and it does not use the staging-and-scaling described earlier since it has its own built-in time-stepping strategy. Keep in mind, therefore, that the variable n_{scale} reported in the tables has no bearing in the Krylov method. We supply the same accuracy parameter TOL for the error control criteria there; see [21] for more details. Since the codes are implemented in MATLAB and are not optimized for speed, exact timings are not decisive. Instead, we report the number of matrix-vector products used. (For the purpose of the experiments, we added a counter to that effect in Expokit's *expv.m* function.) We check the achieved accuracy by reporting $e_{Lag} = \|\boldsymbol{z}_{Kry} - \boldsymbol{z}_{Lag}\|_2$ and $e_{Cheb} = \|\boldsymbol{z}_{Kry} - \boldsymbol{z}_{Cheb}\|_2$, where $\boldsymbol{z}_{Kry}$ is the solution computed by the Krylov method that we use as a reference, while $\boldsymbol{z}_{Cheb}$ and $\boldsymbol{z}_{Lag}$ are the approximations obtained by the Chebyshev and Laguerre methods respectively.

4.1. Dielectric waveguide.

The matrix used here comes from the Matrix Market collection. It results from a finite difference discretization of the Helmholtz equation that governs a dielectric channel waveguide problem, which arises in many integrated circuit applications. The matrix in the example is of order $n = 2048$ with $10,114$ non-zero elements. Its sparsity pattern and spectrum are given in Figure 4.1. The spectrum includes complex eigenvalues, but with small imaginary parts ($|Im(\lambda)| < 10^{-3}$). The spectrum extends slightly past the origin, suggesting we take $[a, b] = [-1, 1]$ in the Chebyshev method. Also, taking $n_{stage} = 1$ is suitable for both the Chebyshev and Laguerre methods. Results for this example are reported in Table 4.1 with $\tau = 1$, $\boldsymbol{b} = (1, \cdots, 1)^T$ and TOL ranging from 10^{-2} to 10^{-6}. We see in this example that Expokit's Krylov(15) detected that one time-step was enough to achieve the desired accuracy. It also appears that the smaller TOL did not induce further computations because the Krylov(15) solution was already more accurate than requested. For the orthogonal expansion methods, we see that Chebyshev performs quite well. Laguerre achieved the desired accuracy, albeit with more iterations. This agrees with our earlier analysis in Section 3.

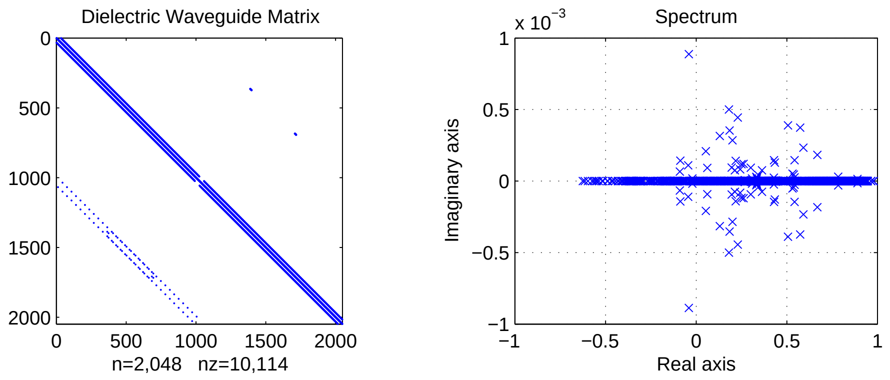

FIG. 4.1. *Sparsity pattern and spectrum of the Waveguide example.*

TOL	Matrix-vector products ($n_{stage} = 1$) Krylov(15)	Laguerre	Chebyshev in $[-1, 1]$	Errors e_{Lag}	e_{Cheb}
10^{-2}	16	16	6	3.3e-03	1.3e-03
10^{-3}	16	19	7	3.9e-04	7.5e-05
10^{-4}	16	22	7	4.1e-05	7.5e-05
10^{-5}	16	26	8	3.5e-06	7.0e-06
10^{-6}	16	30	9	3.8e-07	4.2e-07

TABLE 4.1
Results for the Waveguide example. Here $n_{stage} = 1$ everywhere.

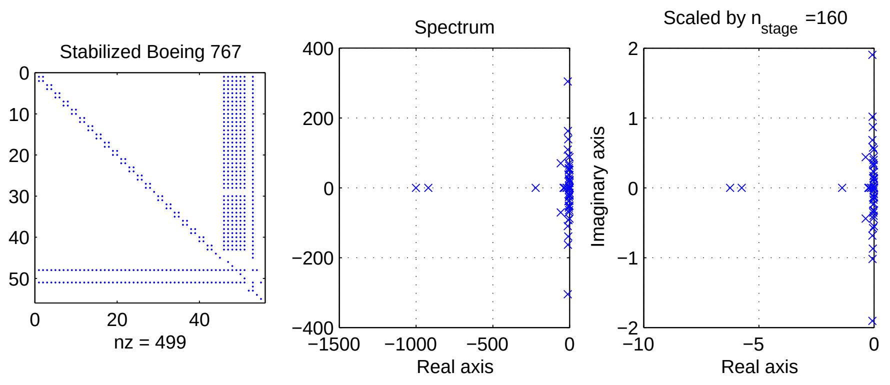

FIG. 4.2. *Sparsity pattern and spectrum of the Boeing 767 example.*

4.2. Boeing 767 matrix. This is an illustrative example in [17]. It arises from modeling a Boeing 767 aircraft with the aim of optimizing its design to suppress flutter of the wings. This involves a nonsmooth, nonconvex optimization method to stabilize a nonsymmetric matrix $\boldsymbol{A}$ that models the aircraft at flutter condition. The resulting stabilized matrix only barely has its spectrum lying in the negative plane. The order is $n = 55$ and the eigenvalue closest to the imaginary axis has $Re(\lambda) = -0.0788$, while the eigenvalue with largest modulus has $|\lambda| = 10^3$. Figure 4.2 depicts the sparsity pattern and the spectrum, showing that this is a challenging example because it has complex eigenvalues with quite large imaginary parts. Even more, these eigenvalues are almost purely imaginary, thus making the problem highly oscillatory and difficult for polynomial methods. In addition, the matrix has $\|\boldsymbol{A}\|_2 = 1.69 \cdot 10^7$ and $\|\boldsymbol{A}\|_1 = 1.6 \cdot 10^7$, so that attempting to infer n_{scale} using either norm would be very restrictive. The figure shows that a scaling value of $n_{scale} = 160$ is suitable to shrink the spectrum to fit our orthogonal expansion methods.

Since $\boldsymbol{A}$ is stable, we do not need to negate it. We set $\boldsymbol{b} = [1, \cdots, 1]^T$ and simply compute $\exp(\tau \boldsymbol{A})\boldsymbol{b}$ using Algorithm 3.1 with, respectively, $\tau = 1, 0.1, 0.01$ and $n_{scale} = 160, 16, 1$. We target two values of TOL, 10^{-3} and 10^{-6}. Results are reported in Table 4.2 with n_{scale} in parentheses. As can be expected, we observe that when $\tau = 1$, Chebyshev and Laguerre struggle to cope with this pathological problem. But the table shows better results with reduced values of τ since they further shrink the spectrum and make the problem more amenable to the polynomial expansion methods.

	Matrix-vector products and n_{stage}			Errors	
$\mathrm{TOL} = 10^{-3}$ τ	Krylov(15)	Laguerre	Chebyshev in $[0, 10]$	e_{Lag}	e_{Cheb}
1	944	5394(160)	2562(160)	9.4e-03	4.4e-02
0.1	144	533(16)	264(16)	2.8e-03	7.9e-03
0.01	48	34(1)	19(1)	2.7e-03	8.4e-04
$\mathrm{TOL} = 10^{-6}$					
1	1552	6880(160)	3218(160)	2.0e-04	2.2e-04
0.1	224	688(16)	339(16)	1.2e-04	1.1e-04
0.01	48	43(1)	24(1)	7.1e-05	6.3e-05

TABLE 4.2
Results for the Boeing example with $\mathrm{TOL} = 10^{-3}, 10^{-6}$ *and* $\tau = 1, 0.1, 0.01$.

4.3. 3D diffusion-convection equations. As a concluding example, we consider the problem of exponential propagation for a discretization of the system

$$u_t = u_{xx} + u_{yy} - \beta u_x - \gamma u_y, \quad (x, y) \in \Omega. \tag{4.1}$$

For our test problem, $\boldsymbol{A}$ is obtained by dividing a square domain Ω into a uniform 500×500 mesh and then applying the standard 5-point diffusion-convection discretization

$$\begin{aligned} \frac{du_{ij}}{dt} = \frac{1}{\delta x^2} \Bigg\{ & 4u_{i,j} - \left(1 - \frac{\beta \delta x}{2}\right) u_{i+1,j} - \left(1 + \frac{\beta \delta x}{2}\right) u_{i-1,j} \\ & - \left(1 - \frac{\gamma \delta y}{2}\right) u_{i,j+1} - \left(1 + \frac{\gamma \delta y}{2}\right) u_{i,j-1} \Bigg\}. \end{aligned} \tag{4.2}$$

Taking $\beta \delta x / 2 = 0.2$ and $\gamma \delta y / 2 = 0.4$ and writing the right hand side of (4.2) as a matrix times a vector $u = u_{i,j}$ yields a $250,000 \times 250,000$ non-symmetric $\boldsymbol{A}$. The $\boldsymbol{b}$ vector is chosen as the initial shape $u_{i,j}(0) = x_i(1 - x_i)y_j(1 - y_j)$.

This matrix has $\|\boldsymbol{A}\|_1 = 8$ and its eigenvalue of largest modulus has $|\lambda| < 8$, while its spectrum does not extend too far in the imaginary direction ($|Im(\lambda)| < 0.5$). Hence, taking $n_{stage} = 1$ is suitable, as well as keeping $[a, b] = [0, 10]$ in the case of the Chebyshev method as done earlier. Results for this example are reported in Table 4.3, with TOL ranging from 10^{-2} to 10^{-6}. We see in the example that, owing to its automatic step size selection mechanism, Expokit's Krylov(15) uses two steps, which is why its number of matrix-vector products is about twice the size of the Krylov basis (the extra products occurred there as part

	Matrix-vector products ($n_{stage} = 1$)			Errors	
TOL	Krylov(15)	Laguerre	Chebyshev in $[0, 10]$	e_{Lag}	e_{Cheb}
10^{-2}	32	20	10	5.4e-05	5.2e-03
10^{-3}	32	24	11	4.8e-06	1.1e-03
10^{-4}	32	28	13	4.0e-07	4.0e-05
10^{-5}	32	31	14	6.0e-08	6.9e-06
10^{-6}	32	35	15	4.7e-09	1.1e-06

TABLE 4.3
Results for the diffusion-convection matrix $\boldsymbol{A}$ given by (4.2). Here $n_{stage} = 1$ everywhere.

of the error estimation). As in the first example, it also appears that the smaller TOL did not induce further computations because the two-step solution here was already more accurate than requested. For the orthogonal expansion methods, we see that Chebyshev performs quite well. Laguerre achieved the desired accuracy with more iterations, as expected from our earlier analysis in Section 3. For both of the methods, choosing n_{stage} is critical to their efficiency. A large n_{stage} would mean that it takes many more steps to completion than necessary, whereas a smaller n_{stage} could mean that the actual polynomial approximation to $\boldsymbol{z}_t = \exp(-\hat{\tau}\boldsymbol{A})\boldsymbol{z}_{t-1}$ in Algorithm 3.1 turns out to be of much higher degree than optimal for the job. An optimal choice is that which shrinks the spectrum into a reasonable domain in a way that ultimately leads to an overall small number of matrix-vector products. This is a problem-dependent issue reminiscent of that arising in ODE solvers where one needs to account for stiffness while attempting to use as few steps as possible.

5. Conclusion.

Matrix exponential algorithms based on approximating $\exp(-\tau t)$ by a suitable polynomial $p(t)$ have several advantages. Because $\boldsymbol{A}$ only occurs in matrix-vector products, algorithms are matrix-free and can be made independent of the data structure chosen for $\boldsymbol{A}$; the user can 'own' $\boldsymbol{A}$'s data structure, so to speak, and does not even have to explicitly store $\boldsymbol{A}$ as a matrix. Further, restricting $\boldsymbol{A}$ to matrix-vector multiplies makes it easy to exploit $\boldsymbol{A}$'s sparsity; no fill-in occurs; and memory usage is capped at $\boldsymbol{A}$'s storage plus a few vectors.

All these advantages accrue to Algorithm 2.1 and Algorithm 2.3. They accrue also, however, to explicit integration schemes like forward Euler (FE), which solves (1.1) by time-stepping with

$$\boldsymbol{y}(t+h) = \boldsymbol{y}(t) - \int_t^{t+h} \boldsymbol{A}\boldsymbol{y}(\tau)d\tau \approx (\boldsymbol{I} - h\boldsymbol{A})\boldsymbol{y}(t). \tag{5.1}$$

The drawback of FE and other explicit integration schemes, of course, is that the time-step h must be taken very small when $\boldsymbol{A}$ is stiff. In the framework of polynomial methods, FE is based on the approximation

$$e^{-\tau t} \approx (1 - \tau t/n)^n\,, \tag{5.2}$$

where n is the number of time steps taken to integrate from $t = 0$ to $t = \tau$. The left hand side of (5.2) is stable only if $h = \tau/n$ is taken small enough that $|(1 - \tau t/n)^n| \leq 1$.

The question arises whether the more sophisticated polynomial methods we have studied in this paper escape the stability issue faced by explicit integration methods when too large a time-step is taken. Using orthogonal polynomials to compute $\exp(-\tau\boldsymbol{A})\boldsymbol{b}$ can be thought of as a semi-implicit method for integrating (1.1).

Unfortunately, as we have seen, the FCR method has the property that the number of iterations or stages increases with $\tau\lambda_{max}$, where λ_{max} is the eigenvalue of largest modulus.

Even when the contribution of $e^{-\tau\lambda_{max}}$ to the final answer is entirely negligible, the presence of a large λ_{max} in $\boldsymbol{A}$'s spectrum forces FCR to iterate more. Thus, semi-implicit methods still suffer from the bane of stiffness.

Viewed differently, all polynomial methods require one to approximate $z(t) = \exp(-\tau t)$ by a polynomial over the interval or region that contains the spectrum of $\boldsymbol{A}$. The *larger* that region is, the more difficult it is (i.e. the higher the degree of polynomial required) to adequately fit $z(t)$ over that region. *Staging* can be thought of as a way to make that region smaller by replacing each eigenvalue λ_i by λ_i/n_{stage}.

With suitable scaling and staging, we have seen that FCR *can* handle quite stiff systems of ODEs (large spread in eigenvalues) as well as exponentials of unsymmetric matrices. Clearly, scaling and staging are an essential part of the FCR method.

We examined the performance of FCR with the Laguerre and Chebyshev systems of orthogonal polynomials and compared them with the Krylov method. In theory, Chebyshev expansions converge within an ellipse in the complex plane; Laguerre expansions, within a parabola; and Hermite expansions, over the entire plane [5]. Because the individual terms in the series (2.2) and (2.4) can be quite large in magnitude but alternate in sign, one may experience significant loss of precision and even numerical overflow as one tries to compute such expansions with finite-precision arithmetic. However, scaling and staging allow us to overcome these difficulties.

In our tests, we found that Chebyshev usually performed better than Laguerre, though this requires that we be able to localize the spectrum to an interval $[a, b]$. Laguerre does not have this requirement, but one still needs to make a sensible choice of n_{stage} for both methods.

REFERENCES

[1] M. ABRAMOWITZ AND I. STEGUN, *Handbook of Mathematical Functions*, Dover, Mineola, New York, 1992.
[2] L. BERGAMASCHI, M. CALIARI, AND M. VIANELLO, *Efficient approximation of the exponential operator for discrete 2d advection-diffusion problems*, Numer. Linear Algebra Appl., 10 (2002), pp. 271–289.
[3] L. BERGAMASCHI AND M. VIANELLO, *Efficient computation of the exponential operator for large, sparse, symmetric matrices*, Numer. Linear Algebra Appl., 7 (2000), pp. 27–45.
[4] P. M. BOSCH, D. C. DIETZ, AND E. A. POHL, *Choosing the best approach to matrix exponentiation*, Comput. Oper. Res., 26 (1999), pp. 871–882.
[5] J. BOYD, *Chebyshev and Fourier Spectral Methods*, Dover, Mineola, New York, 2001.
[6] P. I. DAVIES AND N. J. HIGHAM, *A Schur-Parlett algorithm for computing matrix functions*, SIAM J. Matrix Anal. Appl., 25 (2003), pp. 464–485.
[7] V. L. DRUSKIN AND L. A. KNIZHERMAN, *Two polynomial methods for calculating functions of symmetric matrices*, U.S.S.R. Comput. Math. and Math. Phys., 29 (1989), pp. 112–121.
[8] R. A. FRIESNER, L. S. TUCKERMAN, B. C. DORNBLASER, AND T. V. RUSSO, *A method for exponential propagation of large systems of stiff nonlinear differential equations*, J. Sci. Comput., 4 (1989), pp. 327–354.
[9] I. P. GAVRILYUK, W. HACKBUSCH, AND B. N. KHOROMSKIJ, *Data-sparse approximation to the operator-valued functions of elliptic operator*, Math. Comp., 73 (2003), pp. 1297–1324.
[10] G. H. GOLUB AND C. VAN LOAN, *Matrix Computations*, third ed., John Hopkins University Press, Baltimore, 1996.
[11] M. HOCHBRUCK AND C. LUBICH, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 34 (1997), pp. 1911–1925.
[12] M. HOCHBRUCK, C. LUBICH, AND H. SELHOFER, *Exponential integrators for large systems of differential equations*, SIAM J. Sci. Comput., 19 (1998), pp. 1552–1574.
[13] L. A. KNIZHNERMAN, *Computations of functions of unsymmetric matrices by means of Arnoldi's method*, U.S.S.R. Comput. Math. and Math. Phys., 31 (1991), pp. 5–16.
[14] J. S. KOLE, *Solving seismic wave propagation in elastic media using the matrix exponential approach*, Wave Motion, (2003), pp. 279–293.
[15] N. N. LEBEDEV, *Special Functions and Their Applications*, Dover, Mineola, New York, 1972.

[16] K. MEERBERGEN AND M. SADKANE, *Using Krylov approximations to the matrix exponential operator in Davidson's method*, Appl. Numer. Math., 31 (1999), pp. 331–351.
[17] C. MOLER AND C. VAN LOAN, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Rev., 45 (2003), pp. 3–49.
[18] Y. SAAD, *Analysis of some Krylov subspace appproximations to the matrix exponential operator*, SIAM J. Numer. Anal., 29 (1992), pp. 209–228.
[19] ———, *Iterative Methods for Sparse Linear Systems*, PWS, Boston, 1996. Second edition, SIAM, Philadelphia, 2003.
[20] ———, *Filtered conjugate residual-type algorithms with applications*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 845–870.
[21] R. B. SIDJE, *Expokit: A software package for computing matrix exponentials*, ACM Trans. Math. Software, 24 (1998), pp. 130–156.
[22] R. B. SIDJE AND W. J. STEWART, *A numerical study of large sparse matrix exponentials arising in Markov chains*, Comput. Statist. Data Anal., 29 (1999), pp. 345–368.
[23] D. E. STEWART AND T. S. LEYK, *Error estimates for Krylov subspace approximations of matrix exponentials*, J. Comput. Appl. Math., 72 (1996), pp. 359–369.