Computing $exp(-\tau A)b$ by the Filtered Conjugate Residual Algorithm [*]

Bernard N. Sheehan [†] Yousef Saad [‡]

August 1, 2006

Abstract

This paper discusses a class of Filtered Conjugate Residual Algorithms (FCR) as a way to compute the product of the exponential of a matrix by an arbitrary vector. These methods utilize implicitly expansions of the exponential function in a basis of orthogonal polynomials. FCR methods based on Laguerre, Hermite, and Chebyshev polynomials, are described and their performances are compared. The paper discusses how scaling and staging can affect convergence.

Keywords: Conjugate Residual, Filtered Conjugate Residual, Polynomial Filtering, Exponential Propagation, Orthogonal Polynomials

1 Introduction

The problem of calculating expressions of the form $exp(-\tau A)b$, $A \in R^{n\times n}$ being a non-negative definite matrix and $b \in R^n$ 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

$$\dot{y} = -Ay, \quad y(0) = b \tag{1}$$

at time τ. Equation (1), in turn, arises out of finite difference or finite element discretizations of thermal problems (A symmetric) or convection-diffusion and vibration problems (A non-symmetric). Similarly, very *stiff* systems of the form (1) arise in predicting the time-evolution of electrical circuits. Another occurrence of (1) is in computing the transient solution of Markov chains [22].

The calculation of a matrix exponential times a vector is a treacherous task; see [17] for a survey of potential difficulties. Many methods have been proposed [10, 4, 9, 14, 16, 6]. For very large, sparse matrices, perhaps the preferred method [12, 11, 21, 13, 7, 18, 8] is to use

[*]This work was supported by NSF grants DMS 0510131 and DMS 0528492 and by the Minnesota Supercomputing Institute

[†]Computer Science & Engineering, University of Minnesota, Twin Cities. `sheehan@cs.umn.edu`

[‡]Computer Science & Engineering, University of Minnesota, Twin Cities. `saad@cs.umn.edu`

Lanczos or Arnold iterations to obtain a matrix $V_m \in R^{n\times m}$ whose columns span the Krylov subspace $span\{b, Ab, \cdots, A^{m-1}\}$, and then to write

$$exp(-\tau A)b \approx V_m exp(-\tau H_m)\beta e_1 \tag{2}$$

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

The recent paper [20] showed how to apply a filtered conjugate residue-like algorithm (FCR) to problems as diverse as regularization in graphics, information retrieval, and in electronic structure calculations. Here, we take up the question whether a FCR-type algorithm can also be usefully applied to exponential propagation. The family of techniques we examine in this paper are based on the expansion of $exp(t)$ as a series of orthogonal polynomials. Bergamaschi *et al.* has recently proposed a technique for exponential propagation that uses Chebyshev polynomials [3, 2]; that this technique appears to be competitive with (2) adds interest to the question whether other choices for 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 [3, 2]. 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. As will be seen, this advantage is somewhat mitigated by the requirement to pre-scale the matrix and proceed in stages.

2 Orthogonal Expansions for $exp(-\tau t)$

One can approximate $exp(-\tau A)v$ by using either rational or polynomial approximations to $exp(-\tau t)$. Methods based or rational approximations require inversion of large sparse matrices, and are not considered here. To avoid inversion, we therefore consider methods based on approximating $exp(-\tau t)$ by a polynomial $p(t)$. Then, in place of $exp(-\tau A)b$, we compute $p(A)b$ by a series of matrix-vector mutliplies using A. As background to this strategy, we briefly consider in this section how to expand $exp(-\tau t)$ in series of

- Generalized Laguerre polynomials
- Hermite polynomials
- Chebyshev polynomials.

These polynomials can be thought of as being generated from the sequence $\{1, t, t^2, \dots\}$ by a Gram-Schmidt procedure using an appropriate inner product.

2.1 Generalized Laguerre Polynomials

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

$$\langle p,q\rangle_L = \int_0^\infty t^\alpha e^{-t} p(t)q(t)dt \tag{3}$$

The expansion of the exponential function $exp(-\tau t)$ in terms of laguerre polynomials is know to be [15, p. 90]

$$e^{-\tau t} = (\tau+1)^{-\alpha-1}\sum_{n=0}^{\infty}\left(\frac{\tau}{\tau+1}\right)L_n^\alpha(t), \qquad 0<t<\infty. \tag{4}$$

By truncating the above summation to only m terms an m^{th} degree polynomial will be obtained that will approximate $exp(-\tau t)$ over some interval.

2.2 Hermite Polynomials

The Hermite Polynomials $H_n(t),\ n=0,1,2,\ldots,$ are orthogonal with respect to the inner-product

$$\langle p,q\rangle_H = \int_{-\infty}^\infty e^{-t^2} p(t)q(t)dt \tag{5}$$

In terms of these orthogonal polynomials the exponential $exp(-\tau t)$ has the following expansion [15, p. 74]

$$e^{-\tau t} = e^{\tau^2/4}\sum_{n=0}^{\infty}\frac{(-1)^n\tau^n}{2^n n!}H_n(t), \qquad -\infty<t<\infty. \tag{6}$$

2.3 Chebyshev Polynomials

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

$$\langle p,q\rangle_T = \int_{-1}^{+1}\frac{p(t)q(t)dt}{\sqrt{1-t^2}}. \tag{7}$$

In this case, it is useful to give the expansion of the exponential that is fitted in a general interval with half-width l_1, and centered at l_2, rather than just the interval $[-1,1]$. The expansion of $exp(-\tau t)$ in terms of Chebyshev polynomials over the intervals $[l_2-l_1, l_2+l_1]$ is given by [1, section 9.6]

$$e^{-\tau t} = \sum_{n=0}^{\infty} a_n T_n\left(\frac{t-l_2}{l_1}\right), \qquad l_2-l_1<t<l_2+l_1. \tag{8}$$

where

$$\begin{aligned} a_0 &= exp(-\tau l_2)I_0(-\tau l_1) & (9)\\ a_k &= 2exp(-\tau l_2)I_k(-\tau l_1), \quad k>1, & (10)\end{aligned}$$

$I_k(t)$ being the modified Bessel function of the first kind and l_1 and l_2 being 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 [15, 1].

2.4 Rate of Convergence

To give the reader a sense of how well these expansions fit the exponential $z(t) = exp(-\tau t)$, we plot partial sums for (4), (6), and (8) using several values of τ and different numbers of terms. Fig. (1) shows the convergence performance for a moderately slow exponential, $z(t) = exp(-t)$ ($\tau = 1$) with $m = 7$ terms summed. Fig. 2 shows convergence for a somewhat faster decaying exponential, $z(t) = exp(-2.5t)$ ($\tau = 2.5$) using a partial sum of $m = 10$ terms. Finally, Fig. 3 shows how the three orthogonal expansions behave for a decaying oscillation $z(t) = exp(-0.2t)cos(3t)$ ($\tau = 0.2 + 3j$) with $m = 15$ terms. In all cases the Chebyshev expansions were over the interval $[l_2 - l_1, l_2 + l_1] = [-1, 7]$. The generalized Laguerre expansions all used $\alpha = 0$.

A number of observations about the rate of convergence of these expansions can be made from these plots. The expansions with $L_n^\alpha(t)$, $H_n(t)$, and $T_n(t)$ all performed well for the moderately decaying exponential, see Fig. 1. However, the interval over which the fit is good varies depending on the family of orthogonal polynomials used. For Laguerre polynomials, the region of good fit is anchored at zero and gradually grows toward the right as more terms are added in the expansion. For Hermite polynomials, the fit is concentrated in a narrower region around zero; again, as one adds more terms, the region of better fit gradually expands. Finally, for Chebyshev polynomials, the region where the polynomial fits $exp(-\tau t)$ is explicitly under the user's control, being $[l_2 - l_1, l_2 + l_1]$; adding more terms does not expand the fit region but does improves the quality of fit within the region.

Fig. 2 and (3) indicate how problems might arise with these orthogonal expansions. Specifically, the middle picture on the left in Fig. 2 indicates that Hermite polynomials may have numerical difficulties with rapidly decaying exponentials. In the picture one sees how the polynomial expansion is 'working very hard' to fit the steeply rising part of the exponential to the left of zero, and consequently does a poor job fitting the curve for t greater than zero.

To further examine this phenomena, we plot in Fig. 4 the error

$$\left\| e^{-\tau t} - e^{\tau^2/4} \sum_{n=0}^{m-1} \frac{(-1)^n \tau^n}{2^n n!} H_n(t) \right\| \tag{11}$$

versus m, the number of terms included in the partial sum. The plot used $\tau = 3$ and $t = 0.5$. The rate of convergence is very poor, and many terms need to be included in the partial sum before the fit is close. Evidently, if we wish to use Hermite expansions to evaluate $exp(-\tau A)b$, we must ensure that $\tau\lambda_{max}$ (λ_{max} being the largest eigenvalue of A) is not very large (say less than 2).

Laguerre polynomials pose a different problem, as indicated in Fig. 3: these polynomials seem to struggle more when fitting rapidly oscillating exponentials. This behavior is exemplified by the first picture on the left in Fig. 3, which shows that in this example the rate of convergence for Laguerre polynomials is considerably worse than that for Hermite or Chebyshev polynomials. One must go up to 50 terms in the partial sum—see Fig. 5—to get a fit comparable to the Hermite or Chebyshev expansions with only 15 terms. This suggests that Laguerre polynomials may not be well suited for evaluating $exp(-\tau A)b$ when A has eigenvalues with large imaginary parts.

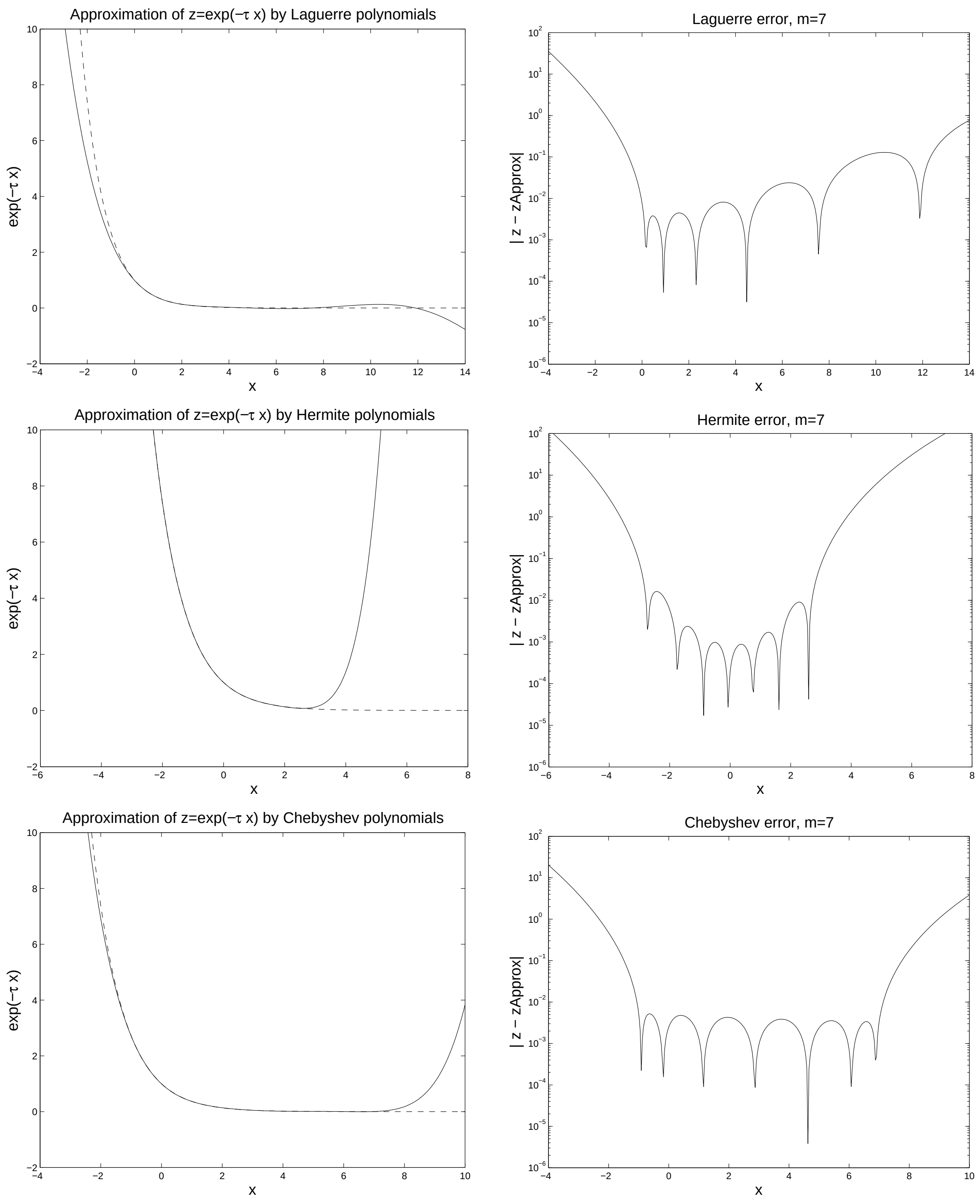

Figure 1: Fit of $z = e^{-\tau t}, \tau = 1$, with $z_{Approx} = \sum_{n=0}^{6} c_n L_n^0(t)$ (top), $z_{Approx} = \sum_{n=0}^{6} c_n H_n(t)$ (middle), and $z_{Approx} = \sum_{n=0}^{6} c_n T_n(t), l_1 = 4, l_2 = 3$ (bottom). Pictures on left plot $e^{-\tau t}$ and polynomials; pictures on right show error $\|e^{-\tau t} - z_{Approx}\|$ versus t.

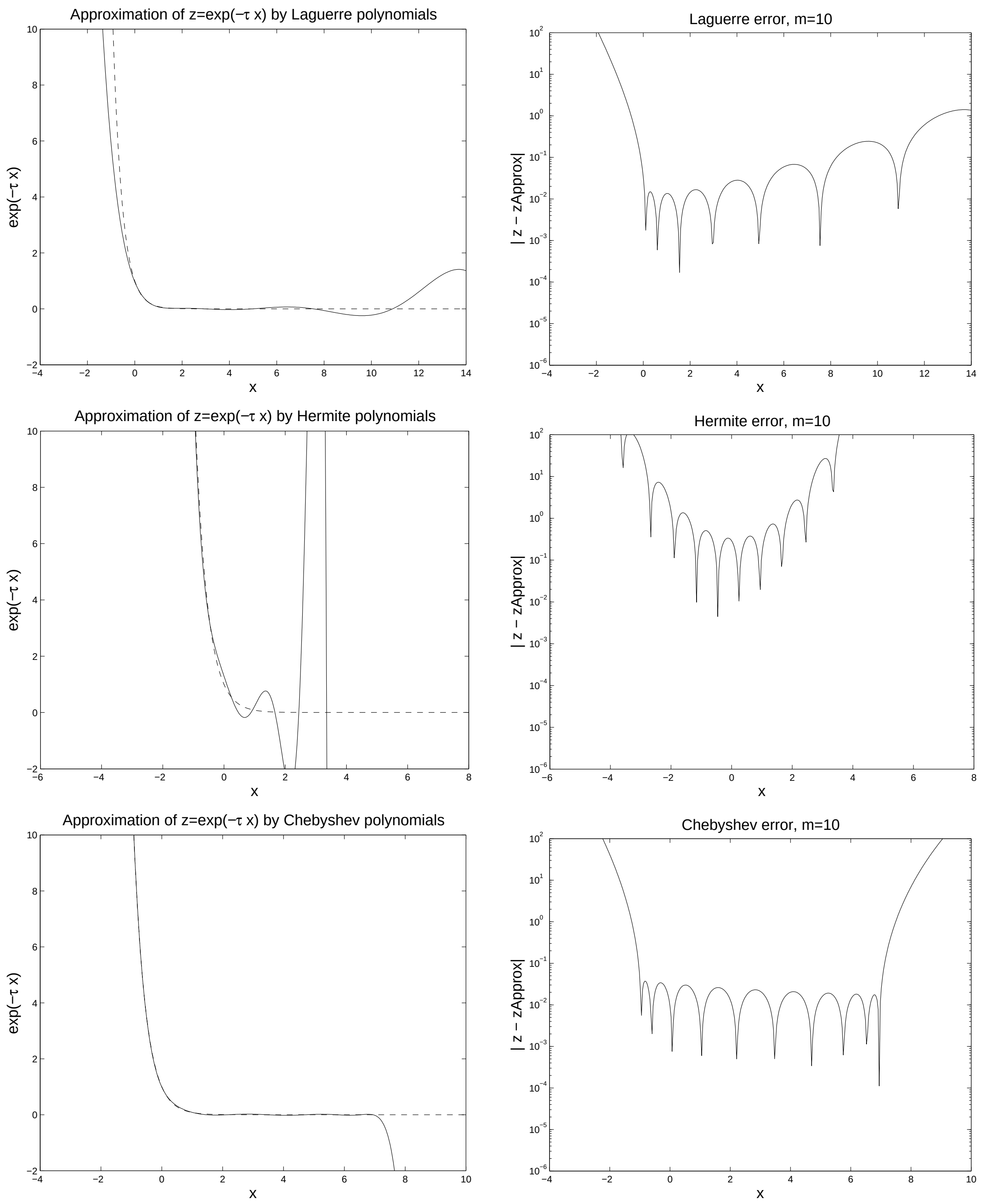

Figure 2: Fit of $z = e^{-\tau t}, \tau = 2.5$, with $z_{Approx} = \sum_{n=0}^{9} c_n L_n^0(t)$ (top), $z_{Approx} = \sum_{n=0}^{9} c_n H_n(t)$ (middle), and $z_{Approx} = \sum_{n=0}^{9} c_n T_n(t), l_1 = 4, l_2 = 3$ (bottom). Pictures on left plot $e^{-\tau t}$ and polynomials; pictures on right show error $\|e^{-\tau t} - z_{Approx}\|$ versus t.

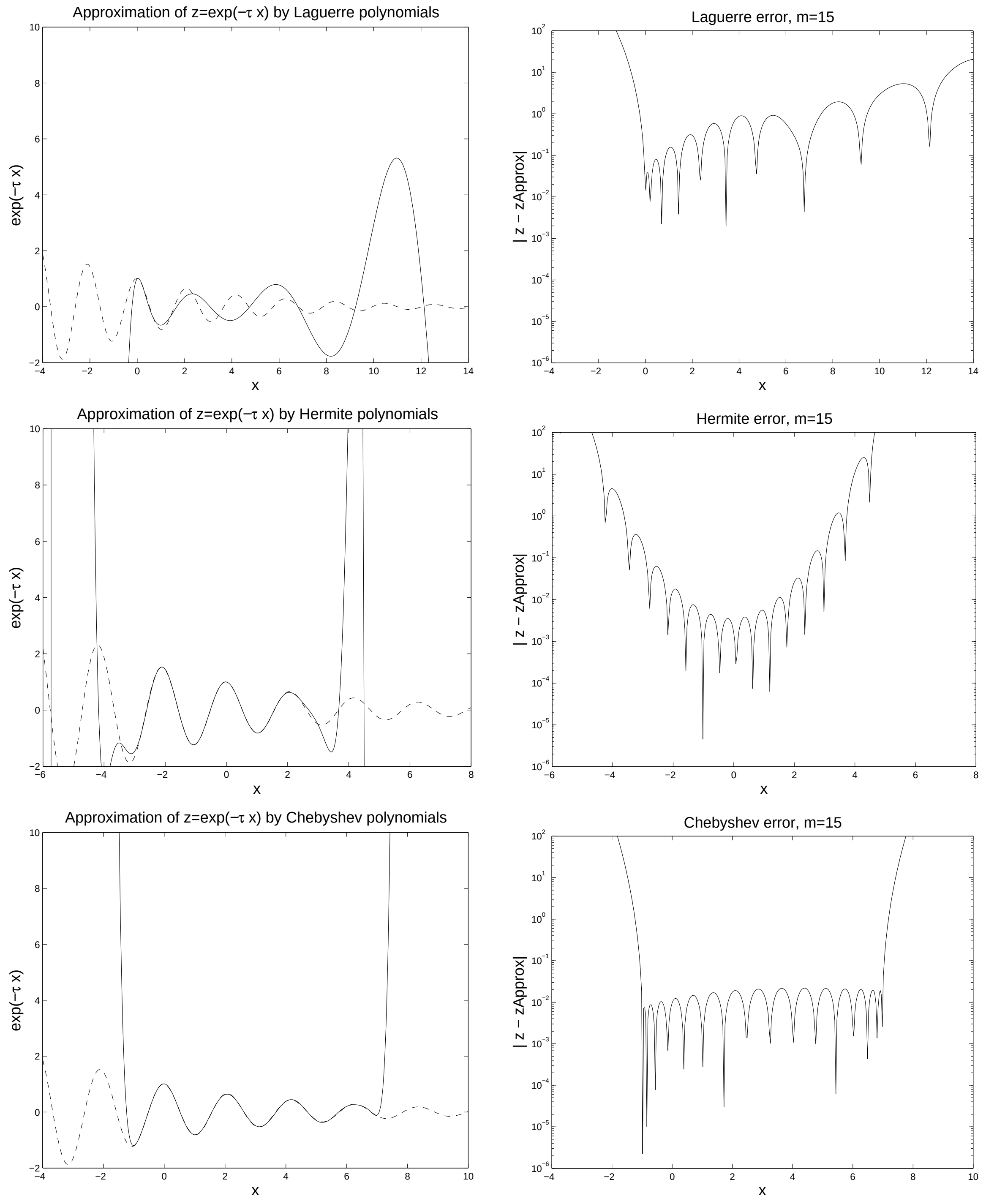

Figure 3: Fit of $z = Re\left[e^{-\tau t}\right], \tau = 0.2 + 3j$, with $z_{Approx} = \sum_{n=0}^{14} c_n L_n^0(t)$ (top), $z_{Approx} = \sum_{n=0}^{14} c_n H_n(t)$ (middle), and $z_{Approx} = \sum_{n=0}^{14} c_n T_n(t), l_1 = 4, l_2 = 3$ (bottom). Pictures on left plot $Re\left[e^{-\tau t}\right]$ and polynomials; pictures on right show error $\|Re\left[e^{-\tau t}\right] - z_{Approx}\|$ versus t.

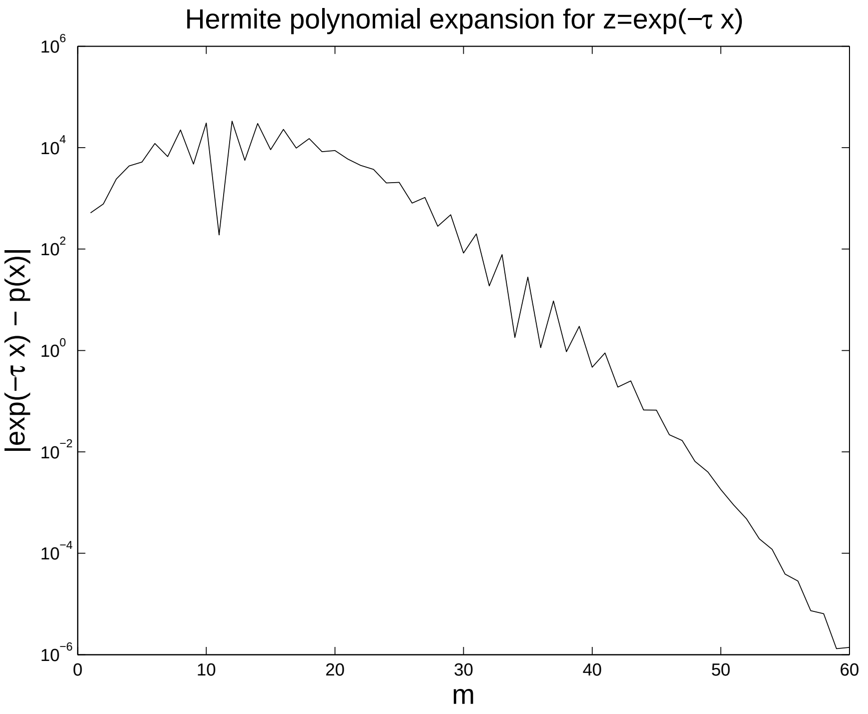

Figure 4: Error between $exp(-3t)$ and $p(t) = \sum_{n=0}^{m-1} c_n H_n(t)$ versus number of terms m in partial sum. Error is evaluated at $t = 0.5$ and expansion uses equation (6) with $\tau = 3$.

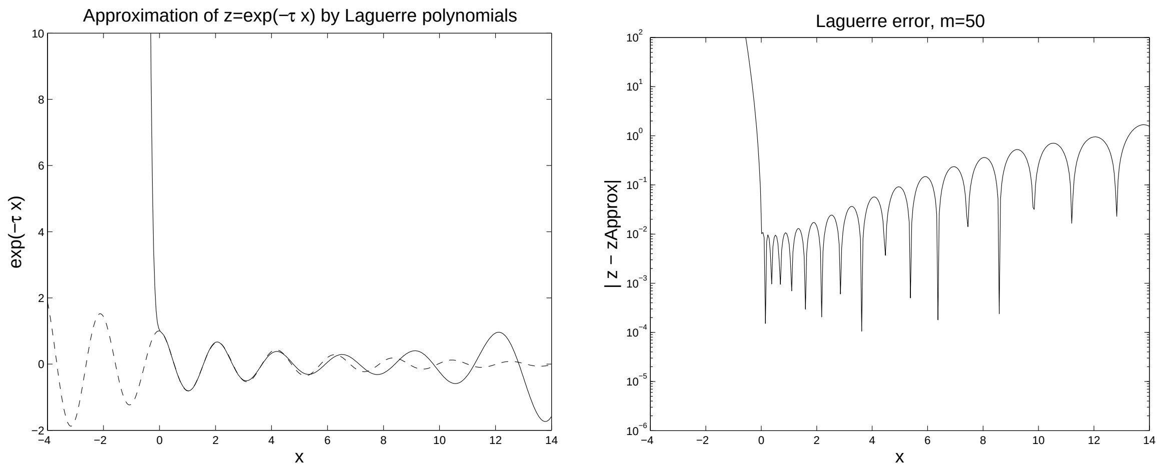

Figure 5: Fit of $z = Re\left[e^{-\tau t}\right], \tau = 0.2 + 3j$, with $z_{Approx} = \sum_{n=0}^{49} c_n L_n^0(t)$ (50 terms). Left is plot of $Re\left[e^{-\tau t}\right]$ and polynomials; right is error $\|Re\left[e^{-\tau t}\right] - z_{Approx}\|$ versus t.

3 Computing $exp(-\tau A)b$ Using Orthogonal Polynomials

All of the above families of orthogonal polynomials satisfy recurrence relations of the form

$$\beta_{n+1}P_{n+1}(t) = (t - \alpha_n)P_n(t) - \beta_{n-1}P_{n-1}(t). \tag{12}$$

These three-term recurrences for the generalized Laguerre, Hermite, and Chebyshev polynomials are given by

$$\begin{aligned} -(n+1)L^{\alpha}_{n+1}(t) &= (t - \alpha - 2n - 1)L^{\alpha}_{n}(t) + (n+\alpha)L^{\alpha}_{n-1}(t) & (13)\\ (1/2)H_{n+1}(t) &= tH_n(t) - nH_{n-1}(t) & (14)\\ T_{n+1}(t) &= 2tT_n(t) - T_{n-1}(t) & (15)\end{aligned}$$

together with the following starting values,

$$\begin{aligned} L^{\alpha}_0(t) &= 1, \quad L^{\alpha}_1(t) = 1 + \alpha - t & (16)\\ H_0(t) &= 1, \quad H_1(t) = 2t & (17)\\ T_0(t) &= 1, \quad T_1(t) = t. & (18)\end{aligned}$$

These recurrences allow one to easily generate successive members of these orthogonal families.

Assuming $exp(-\tau t)$ has the following expansion in terms of orthogonal polynomials $P_n(t)$ (see (4), (6), and (8)),

$$e^{-\tau t} = \sum_{n=0}^{\infty} c_n P_n(t), \tag{19}$$

the following algorithm can be used to compute $exp(-\tau A)b$:

Algorithm **3.1** *$exp(-\tau A)b$ by Orthogononal Polynomials*

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

The algorithm uses (12) in line 4 and (19) in lines 2 and 5. If A is an $n \times n$ matix with $Nz(A)$ non-zeros, then Algorithm 3.1 requires a storage of $4n$ numbers (the 4 vectors z, p_{j-1}, p_j, p_{j+1}) and entails a computational cost of about $9n + 2Nz(A)$ operations per pass through the for-loop (step 7 can be carried out in $O(2)$ operations by redirecting pointers rather than copying data).

4 Filtered Conjugate Residual-type Algorithm

An alternative way of computing $exp(-\tau A)b$ 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:

$$\langle p, q\rangle_w = \int_a^b p(t)q(t)w(t)dt \tag{20}$$

$w(t) \geq 0$ being some weight function. In addition, the updates to x_j use a different coefficient α_j than the coefficient $\tilde{\alpha}_j$ used to update $\tilde{r}_j$. Inputs to the algorithm are a filter function ψ, an inner-product $\langle p, q\rangle_w$, a matrix A and a vector b:

Algorithm **4.1** *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 $p_j, \tilde{r}_j, x_j$ are the corresponding sequences of vectors

$$p_j = \pi_j(A)r_0 \tag{21}$$
$$\tilde{r}_j = \tilde{\rho}_j(A)r_0 \tag{22}$$
$$x_{j+1} = x_0 + s_{j+1}(A)r_0 \tag{23}$$

where $r_0 = b - Ax_0$.

We interject one implementation detail. To get FCR to work with Hermite polynomials, one can use the inner-product

$$\langle p, q\rangle_{H'} = \int_{-\infty}^{\infty} e^{-(t-t_s)^2} p(t)q(t)dt \tag{24}$$

in place of (5). This stratagem ensures that $\langle\tilde{\rho}_j, \lambda\tilde{\rho}_j\rangle \neq 0$ in line 3 when $j = 0$ and $\tilde{\rho}_0 = 1$. For simplicity, we take $t_s = 1$.

The Filtered Conjugate Residue algorithm satisfies the following properties:

Proposition 4.1 *The solution vector x_{j+1} computed at the j-th step of Algorithm 4.1 is of the form $x_{j+1} = x_0 + s_{j+1}(A)r_0$, where s_j is the j-th degree polynomial:*

$$s_{j+1}(\lambda) = \alpha_0\pi_0(\lambda) + \cdots + \alpha_j\pi_j(\lambda) \ . \tag{25}$$

The polynomials π_j and the auxiliary polynomials $\tilde{\rho}_j(\lambda)$ satisfy the orthogonality relations,

$$\langle \lambda\pi_j(\lambda), \lambda\pi_i(\lambda)\rangle_w = \langle \lambda\tilde{\rho}_j(\lambda), \tilde{\rho}_i(\lambda)\rangle_w = 0 \quad \textit{for} \quad i \neq j \ . \tag{26}$$

In addition, the filtered residual polynomial $\psi(\lambda) - \lambda s_j(\lambda)$ minimizes $\|\psi - \lambda s(\lambda)\|_w$ among all polynomials s of degree $\leq j-1$.

The proof is given in [20] for a general inner product.

It is worth remarking that the π_j's generated by algorithm 4.1 are the orthogonal polynomials on $[a,b]$ associated with the weight $t^2w(t)$; this result follows from (26). When the FCR algorithm with Laguerre polynomials $L_n^\alpha(t)$ is used, for example, then the π_j's are the generalized Laguerre polynomials $L_n^{\alpha+2}(t)$.

Corollary 4.1 *If $\phi = 1-\psi$, then the polynomial $\zeta_j(\lambda) = 1 - \lambda s_j$ minimizes $\|\phi - \zeta_j\|_w$ among all polynomials $p_j(\lambda) \in \mathcal{P}_j$ satisfying $p_j(0) = 1$.*

Note that the FCR algorithm has a cost per loop of 3 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), 3 polynomial inner-products, and one polynomial multiply by λ. 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 λ-polynomial multiply will also incur a cost of $6j$ (by employing the 3-term recurrence relation for the orthogonal polynomials to express $\lambda\pi_{j+1}$ in the orthogonal polynomial basis). In short, the over-all cost per loop is $6n + Nz(A) + 16j$, j being the loop index and $Nz(A)$ the number of non-zeros in A. To add a termination test like $\|x_{j+1} - x_j\| < \epsilon$, analogous to line 6 in algorithm (3.1), would increase the loop-cost by another $2n$. FCR requires storage of $4n + 3j$ (the 4 vectors $x_j, \tilde{r}_j, Ap_j$, and p_j, and coefficients for the 3 polynomials $\tilde{\rho}_j, \pi_j, \lambda\pi_j$). Our bookkeeping suggest that (4.1) may be marginally faster but requires marginally more memory than (3.1), provided j remains small compared to n.

To use FCR to compute $exp(-\tau A)b$, let $\phi = exp(-\tau\lambda)$. Then, with $x_0 = 0$, apply the FCR algorithm to get $x_j = \sum_{k=0}^{j-1} \alpha_k\pi_k(A)b$. Finally, compute $z_j = b - Ax_j = (I - As_j(A))b = \zeta_j(A)b$ as an approximation for $\phi(A)b = exp(-\tau A)b$. If one wants the current estimate for $exp(-\tau A)b$ at each step in the iteration, one can replace line (5) in algorithm (4.1) by

$$z_{j+1} = z_j - \alpha Ap_j. \tag{27}$$

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

$$\begin{aligned} \|exp(-\tau A)b - z_j\| &= \|\phi(A)b - \zeta_j(A)b\|_2 & (28)\\ &= \|V(\phi(\Lambda) - \zeta_j(\Lambda))V^Tb\|_2 & (29)\\ &\leq \max_i |\phi(\lambda_i) - \zeta_j(\lambda_i)| \cdot \|V^Tb\|_2 & (30)\end{aligned}$$

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

5 Scaling and Staging

The FCR algorithm involves calculation of vector sequences $s_j(A)b$, $\tilde{\rho}_j(A)b$, and $\pi_j(A)b$, which involve polynomials in A. If the polynomials $s_j(\lambda), \tilde{\rho}_j(\lambda)$, or $\pi_j(\lambda)$ become very large in absolute value at any of the eigenvalues of A, there could be significant loss in accuracy and even numerical overflow during computation of successive polynomials in the sequence.

One strategy to reduce the risk of numerical overflow is to scale A by a number $|a| \approx \|A\|$ and then to break the calculation of the exponential into n_{stage} stages, where $n_{stage} \approx \tau|a|$:

ALGORITHM **5.1** *Scaled and Staged Calculation of* $exp(-\tau A)b$

```
1. $\hat{A} = A/|a|$
2. $\hat{\tau} = \tau|a|/n_{stage}$
3. $z_0 = b$
4. For $t = 1 : n_{stage}$
5.     $z_t = exp(-\hat{\tau}\hat{A})z_{t-1}$
6. EndFor
```

One way to view the above algorithm is from the angle of solving systems of differential equations. Computing $exp(-\tau A)b$ is equivalent to solving the differential equation $y' = -\tau Ay$ for $t \in [0,1]$ with the initial condition $y_0 = b$. Instead of attempting to solve this differential equation in a single step, the above algorithm splits the interval [01] into n_{stage} subintervals and solves the differential equation in n_{stage} steps.

The reasoning here is that, in the first place, scaling A by $|a|$ moves the spectrum of $\hat{A}$ (assumed to be symmetric non-negative definite) into an interval $[0, |a|]$ near $[0,1]$; second, the act of subdividing the calulation into n_{stage} stages ensures that $\hat{\tau} \approx 1$, which should help our orthogonal polynomial approximation for $\phi(\lambda) = exp(-\hat{\tau}\lambda)$ converge after a relatively few number of terms in the partial sum. The condition $\hat{\tau} \approx 1$ means we are closer to situation in Fig. 1 than that in Fig. 2.

6 Numerical Results

In this section we apply the FCR approach described above to some sample calculations of $exp(-\tau A)b$.

6.1 Stiff Symmetric Matrix

As our first example, consider the computation of $exp(-A)b$ ($\tau = 1$) when A is the matrix

$$A = V^T diag\left\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^2\right\} V. \tag{31}$$

V is a randomly chosen orthogonal matrix, and $b = [1, 1, \cdots, 1]^T$. This problem highlights the effect of the relatively large eigenvalue $\lambda_{max} = 10^2$ on the convergence rate of various FCR methods, with and without scaling. The condition number of A is $\kappa_A = 10^8$.

A linear system of differential equations like (1) that has a very wide range of time constants (eigenvalues) is called a *stiff* system. Such systems are difficult to integrate by explicit

	FCR Method	$\lvert a\rvert$	n_{stage}	Iters	Time(s)	$\lVert\phi - z_{Iters}\rVert$
1	Laguerre	1.0	1	*	*	*
2	Laguerre	$\lVert A\rVert_1$	1	4333	22.101	4.6e-13
3	Laguerre	$\lVert A\rVert_1$	10	4470	3.4541	1.8e-12
4	Laguerre	$\lVert A\rVert_1$	100	5200	2.1109	3.0e-12
5	Hermite	1.0	1	*	*	*
6	Hermite	$\lVert A\rVert_1$	1	*	*	*
7	Hermite	$\lVert A\rVert_1$	10	*	*	*
8	Hermite	$\lVert A\rVert_1$	100	2300	1.3026	3.0e-13
9	Chebyshev	1.0	1	66	0.047	1.3e-13
10	Chebyshev	1.0	10	220	0.1863	1.7e-12
11	Chebyshev	1.0	100	1000	1.1619	2.1e-11

Table 1: FCR method applied to matrix A given by (31). Asterisks in a row indicate the method did not converge. For the Chebyshev cases, $[l_2 - l_1, l_2 + l_1] = [0, \|A\|_1]$.

integration methods (e. g., Forward Euler), often requiring a very small time-step to avoid instability. Because exponential propagation can be considered equivalent to integrating a system like (1), our choice of matrix (31) tests the ability of the FCR method to integrate a stiff system of ODEs.

Results for this problem are given in Table 1. From the table we see that FCR with the Chebyshev inner product converged the fastest; the method converged without scaling or staging. FCR with Laguerre polynomials was the second best performer; here, scaling by $\|A\|_1$ was necessary for the method to converge, and staging the calculation by $n_{stage} = 10$ and $n_{stage} = 100$ materially improved the speed of the calcuation. Since A is a relatively small matrix ($8x8$), the cost of the calculation for such large iteration counts is dominated by the cost of the inner-products and the polynomial updates. Finally, FCR with Hermite polynomials did not converge unless A was scaled by $\|A\|_1$ and the calculation was broken into sufficiently many stages ($n_{stage} = 100$); however, when these accommodations were made, the method converged faster than the Laguerre method with the same scaling and staging.

6.2 Non-symmetric Matrix

As a second example, consider the computation of $exp(-\tau A)b$ when A is the unsymmetric matrix

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 10 \\ 0 & 0 & -10 & 2 \end{pmatrix} \tag{32}$$

and, as before, $\tau = 1$ and $b = [1, 1, \cdots, 1]^T$. The eigenvalues of A are $\{+i, -i, 2 + 10i, 2 - 10i\}$.

Results for this case are given in Table 2. For this non-symmetric A, all methods required scaling and staging. The FCR with Hermite polynomials did best, followed by FCR with

	FCR Method	$\lvert a\rvert$	n_{stage}	Iters	Time(s)	$\|\phi - z_{Iters}\|$
1	Laguerre	1.0	1	*	*	*
2	Laguerre	$\|A\|_1$	1	*	*	*
3	Laguerre	$\|A\|_1$	5	435	0.1861	3.7e-10
4	Laguerre	$\|A\|_1$	20	548	0.2574	1.8e-10
5	Hermite	1.0	1	*	*	*
6	Hermite	$\|A\|_1$	1	*	*	*
7	Hermite	$\|A\|_1$	5	175	0.0790	2.0e-11
8	Hermite	$\|A\|_1$	20	296	0.1948	1.7e-11
9	Chebyshev	1.0	1	*	*	*
10	Chebyshev	1.0	5	*	*	*
11	Chebyshev	1.0	20	200	0.2399	2.5e-10

Table 2: FCR method applied to non-symmetric matrix A given by (32). Asterisks in a row indicate the method did not converge. As before, for the Chebyshev cases, $[l_2 - l_1, l_2 + l_1] = [0, \|A\|_1]$.

Laguerre polynomials.

6.3 3D Diffusion-Convection Equations

As a third example, we consider the problem of exponential propagation for a discretization of the system

$$u_t = u_{xx} + u_{yy} - \beta u_x - \gamma u_y, \quad (x, y) \in \Omega. \tag{33}$$

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

$$\begin{aligned}\frac{du_{ij}}{dt} = \frac{1}{\delta x^2}\Bigg\{ & 4u_{i,j} - \left(1 - \frac{\beta\delta x}{2}\right) u_{i+1,j} - \left(1 + \frac{\beta\delta x}{2}\right) u_{i-1,j} \\ & - \left(1 - \frac{\gamma\delta y}{2}\right) u_{i,j+1} - \left(1 + \frac{\gamma\delta y}{2}\right) u_{i,j-1} \Bigg\}.\end{aligned} \tag{34}$$

Taking $\beta\delta x/2 = 0.2$ and $\gamma\delta y/2 = 0.4$ and stamping the righthand side of (34) yields a $250,000 \times 250,000$ non-symmetric A. The b vector is chosen as the initial shape $u_{i,j}(0) = x_i(1 - x_i)y_j(1 - y_j)$.

The data in Table 3 show clearly that, for this problem at least, performance of the various FCR methods depends strongly on the choice of the scaling parameter $|a|$ and number of stages, n_{stage}. For a given choice of n_{stage}, the Chebyshev FCR algorithm generally converges faster than the Hermite FCR method, which in turn converges faster than the Laguerre FCR method, although it is not as robust. This general behavior seems to hold in this example

[1]A rather inefficient version of the Lanczos algorithm was used in that $V_m exp(-\tau T_m)\beta e_1$ in the algorithm is computed at each iteration; this was done to match the availability of $z_m = \zeta_m(A)b$ at each step for FCR methods.

	FCR Method	$\|a\|$	n_{stage}	Iters	Time(s)	$\|\phi - z_{Iters}\|$
1	Laguerre	0.09	1	*	*	*
2	Laguerre	0.1	1	16	0.929	4.9e-12
3	Laguerre	0.2	1	17	0.986	6.0e-13
4	Laguerre	1.0	1	43	2.272	6.0e-13
5	Laguerre	$\|A\|_1$	1	254	12.885	6.0e-13
6	Laguerre	$\|A\|_1$	4	296	14.766	6.0e-13
7	Laguerre	$\|A\|_1$	8	344	17.658	6.0e-13
8	Laguerre	$\|A\|_1$	10	370	19.195	6.0e-13
9	Laguerre	$\|A\|_1$	100	1200	71.979	6.0e-13
10	Hermite	0.2	1	*	*	*
11	Hermite	1.0	1	20	1.1254	6.0e-13
12	Hermite	$\|A\|_1$	1	*	*	*
13	Hermite	$\|A\|_1$	4	124	6.507	6.0e-13
14	Hermite	$\|A\|_1$	8	160	8.829	6.0e-13
15	Hermite	$\|A\|_1$	10	180	10.091	6.0e-13
16	Hermite	$\|A\|_1$	100	800	52.494	6.0e-13
17	Chebyshev	1.0	1	20	1.084	6.0e-13
18	Chebyshev	1.0	4	52	2.857	6.0e-13
19	Chebyshev	1.0	8	88	4.947	6.0e-13
20	Chebyshev	1.0	10	100	5.695	6.0e-13
21	Chebyshev	1.0	100	600	37.232	6.0e-13
18	Lanczos[1]	–	1	31	8.9487	–

Table 3: FCR method applied to diffusion-convection matrix A given by (34). Asterisks in a row indicate the method did not converge. As before, for the Chebyshev cases, $[l_2-l_1, l_2+l_1] = [0, \|A\|_1]$. Errors are computed against the output of the Lanczos algorithm, row 21 in the table.

at least for the standard choice of $|a| = \|A\|_1$ for Laguerre/ Hermite methods and interval $[l_2 - l_1, l_2 + l_1] = [0, \|A\|_1]$ for Chebyshev method.

It is interesting to note that choices of scale $|a|$ less than 1.0 can actually accelerate convergence of the Laguerre FCR algorithm. The fastest convergence occurs when $|a| \approx .1$, which is at the edge of instability: setting $|a| = 0.09$ causes the method to diverge. One observes that for this choice of $|a|$ the incremental change $\|x_{j+1} - x_j\|_2$ becomes quite small very quickly and then begin to slowly and inexorably grow:

$iter = 100 \quad \|x - xprev\| = 1.4417e - 05$
$iter = 200 \quad \|x - xprev\| = 0.061399$
$iter = 300 \quad \|x - xprev\| = 12.1377$
$iter = 400 \quad \|x - xprev\| = 220.6484$
$iter = 500 \quad \|x - xprev\| = 37628.1412$
$iter = 600 \quad \|x - xprev\| = 30367310.0993$

	λ_{max}	$\lvert a\rvert$	n_{stage}	Iters	Time(s)	$\lVert\phi - z_{Iters}\rVert$
1	1	$\lVert A\rVert_1$	1	37	0.0282	1.0e-12
2	10	$\lVert A\rVert_1$	1	261	0.1616	3.9e-13
3	100	$\lVert A\rVert_1$	1	2574	7.8707	1.8e-13
4	100	$\lVert A\rVert_1/10$	1	269	0.1602	3.2e-11
5	100	$\lVert A\rVert_1/10$	4	304	0.1334	3.0e-14
6	1000	$\lVert A\rVert_1$	1	25388	799.8	1.2e-12
7	1000	$\lVert A\rVert_1/10$	1	2574	7.8473	4.1e-12
8	1000	$\lVert A\rVert_1$	40	26120	24.9746	1.2e-12
9	1000	$\lVert A\rVert_1/10$	40	3040	1.0734	1.2e-12

Table 4: FCR method with Laguerre polynomials applied to matrix A given by (35).

$iter = 700 \quad \|x - xprev\| = 6370402512.9629$ etc.

6.4 Further Investigation of $|a|$ and n_{stage}

To further emphasize the role played by scaling and staging, we examine the convergence behavior of $exp(-\tau A)b$ with

$$A = \begin{pmatrix} \lambda_{min} & 0 \\ 0 & \lambda_{max} \end{pmatrix} \tag{35}$$

and $b^T = [1,1]^T$. For out study we choose $\lambda_{min} = 10^{-6}$ and let λ_{max} assume successively the values $1, 10, 100, 1000$. For simplicity, we let $\tau = 1$ and consider only Laguerre polynomials. Results are summarized in Table 4.

Notice that for $\lambda_{max} \geq 100$, we have for all intents and purposes

$$z = exp(-A)b = \begin{pmatrix} e^{-\lambda_{min}} \\ e^{-\lambda_{max}} \end{pmatrix} \approx \begin{pmatrix} e^{-\lambda_{min}} \\ 0 \end{pmatrix}. \tag{36}$$

For $\lambda_{max} = 100$, for example, $e^{-\lambda_{max}} \approx 3.7e - 44$.

Despite the negligible contribution of λ_{max} to the value of $z = exp(A)b$ when $\lambda_{max} \geq 100$, we see from Table 4 that the value of λ_{max} *strongly* affects the rate of convergence of the FCR algorithm. Comparing rows 3 and 7 of the table, for example, we see that while the final answer is essentially unchanged in going from $\lambda_{max} = 100$ to $\lambda_{max} = 1000$, the run time increases from ~ 8 sec to over 800 sec .

A judicious choice of $|a|$ and n_{stage} considerably amerliorates the situation. Qualitatively, $|a|$ can be thought of as moving the spectrum of $\hat{A} = A/|a|$ onto an interval over which the Laguerre polynomials converge fairly quickly. From Fig. 1 and Fig. 2, it seems that the Laguerre expansion for $e^{-\tau t}$ converges reasonably fast on the interval $[0, 10]$; accordingly, a reasonable heuristic is to set $|a| \approx \|A\|_1/10$. This choice of $|a|$ localizes the spectrum of a positive definite $\hat{A}$ within $[0, 10]$. Next, division of the calculation into n_{stage} stages has the advantage of ensuring that $\hat{\tau} = \tau|a|/n_{stage}$ is not too large a number. Appeal being made again to Fig. 2, which shows that the Laguerre expansion for $e^{-2.5t}$ converges fairly quickly over our chosen interval $[0, 10]$, we surmise that choosing n_{stage} such that $\hat{\tau} = \tau|a|/n_{stage} \approx 2.5$ is a reasonable policy. The performance of these heuristics can be judged by comparing rows 3 and 5 and rows 6 and 9 of Table 4.

7 Conclusion

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

All these advantages accrue to Algorithm 3.1 and Algorithm 4.1. They accrue also, however, to explicit integration schemes like forward Euler (FE), which solves (1) by time-stepping with

$$y(t+h) = y(t) - \int_t^{t+h} Ay(\tau)d\tau \approx (I - hA)y(t). \tag{37}$$

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

$$e^{-\tau t} \approx (1 - \tau t/n)^n, \tag{38}$$

where n is the number of time steps taken to integrate from $t = 0$ to $t = \tau$. The left hand side of (38) is stable only if $h = \tau/n$ is taken small enough that $|(1 - \tau t/n)^n| < 1$.

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 $exp(-\tau A)b$ can be thought of as a semi-implicit method for integrating (1). Unfortunately, as we have seen, the FCR method has the property that the number of iterations or 'time-steps' increases with $\tau\lambda_{max}$. Even when the contribution of $e^{-\tau\lambda_{max}}$ to the final answer is entirely negligible, the presence of a large λ_{max} in 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 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 three systems of orthogonal polynomials: Laguerre, Hermite, and Chebyshev. In theory, Chebyshev expansions converge within an ellipse of the complex plane; Laguerre polynomials, within a parabola; Hermite, over the entire plane [5]. This might lead one to expect Hermite polynomials to be the best for exponential propagation, especially with non-symmetric A. Any theoretical advantage Hermite expansions enjoy is somewhat illusory, however, when one considers the adverse impact of finite-precision arthmetic. Because the individual terms in the series (4), (6), and (8) 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 an expansion with finite-precision arithmetic. We saw that the Hermite expansion for $exp(-\tau x)$ converges especially slowly when τ is small, and there is more opportunity for loss of precision and overflow in this situation.

Our tests found that Chebyshev usually performed better than Hermite or Laguerre. Between the latter two, when FCR with Hermite polynomials converged, it usually did so faster than with Laguerre. Contrarily, FCR with Laguerre was more robust than Hermite, converging sometimes when Hermite did not. Although Laguerre and Hermite do not require A's spectrum to be localized rigorously within the interval $[l_2 - l_1, l_2 + l_1]$, as Chebyshev does, one still needs to make sensible choices for scaling parameter $|a|$ and number of stages n_{stage}. In the case of FCR with Laguerre, we proposed the heuristic $|a| \approx \|A\|_1/10$ and $n_{stage} \approx \tau|a|/2.5$.

References

[1] M. Abramowitz and IA Stegun. *Handbook of Mathematical Functions.* Dover, 1992.

[2] Luca Bergamaschi, Marco Caliari, and Marco Vianello. Efficient approximation of the exponential operator for discrete 2d advection-diffusion problems. *Numer. Linear Algebra Appl.*, 10(3):271–289, 2002.

[3] Luca Bergamaschi and Marco Vianello. Efficient computation of the exponential operator for large, sparse, symmetric matrices. *Numer. Linear Algebra Appl.*, 7:27–45, 2000.

[4] Peter M. Bosch, Dennis C. Dietz, and Edward A. Pohl. Choosing the best approach to matrix exponentiation. *Computers and Operations Research*, 26:871–882, 1999.

[5] J. Boyd. *Chebyshev and Fourier spectral methods.* Dover, 2001.

[6] Philip I. Davies and Nicholas J. Higham. A Schur-Parlett algorithm for computing matrix functions. *SIAM J. Matrix Anal. Appl.*, 25(2):464–485, 2003.

[7] V. L. Druskin and L. A. Knizherman. Two polynomial methods for calculating functions of symmetric matrices. *USSR Comput. Maths. Math. Phys.*, 29:112–121, 1989.

[8] Richard A. Friesner, Laurette S. Tuckerman, Bright C. Dornblaser, and Thomas V. Russo. A method for exponential propagation of large systems of stiff nonlinear differential equations. *Journal of Scientific Computing*, 4(4):327–354, 1989.

[9] Ivan P. Gavrilyuk, Wolfgang Hackbusch, and Boris N. Khoromskij. Data-sparse approximation to the operator-valued functions of elliptic operator. *Mathematics of computation*, 73(247):1297–1324, July 2003.

[10] Gene H. Golub and Charles Van Loan. *Matrix Computations*. John Hopkins University Press, second edition, 1989.

[11] M. Hochbruck and C. Lubich. On Krylov subspace approximations to the matrix exponential operator. *SIAM Journal on Numerical Analysis*, 34:1911–1925, 1997.

[12] M. Hochbruck, C. Lubich, and H. Selhofer. Exponential integrators for large systems of differential equations. *SIAM Journal on Scientific Computing*, 19:1552–1574, 1998.

[13] L. A. Knizhnerman. Computations of functions of unsymmetric matrices by means of Arnoldi's method. *J. Comput. Math. and Math. Phys.*, 31(1):5–16, 1991.

[14] J. S. Kole. Solving seismic wave propagation in elastic media using the matrix exponential approach. *Wave Motion*, 2003.

[15] N. N. Lebedev. *Special Functions and Their Applications*. Dover, 1972.

[16] Karl Meerbergen and Miloud Sadkane. Using Krylov approximations to the matrix exponential operator in Davidson's method. *Applied Numerical Mathematics*, 31:331–351, 1999.

[17] C. Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix. *Siam Review*, 20:801–836, Oct 1978.

[18] Yousef Saad. Analysis of some Krylov subspace appproximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 29(1):209–228, Feb 1992.

[19] Yousef Saad. *Iterative methods for sparse linear systems*. PWS, 1996.

[20] Yousef Saad. Filtered conjugate residual-type algorithms with applications. Technical report, Minnesota Supercomputer Institute, University of Minnesota, 2005. To appear, SIAM J. Mat. Anal.

[21] R.B. Sidje. EXPOKIT: Software package for computing matrix exponentials. *ACM Transactions on Mathematical Software*, 24(1):130–156, 1998.

[22] Roger B. Sidje and William J. Stewart. A numerical study of large sparse matrix exponentials arising in Markov chains. *Computational Statistics and Data Analysis*, 1999.