SIAM J. SCI. COMPUT.
Vol. 35, No. 3, pp. A1376–A1397

RESIDUAL, RESTARTING, AND RICHARDSON ITERATION FOR THE MATRIX EXPONENTIAL*

MIKE A. BOTCHEV†, VOLKER GRIMM‡, AND MARLIS HOCHBRUCK‡

Mike A. Botchev devotes this work to the memory of his father.

Abstract. A well-known problem in computing some matrix functions iteratively is the lack of a clear, commonly accepted residual notion. An important matrix function for which this is the case is the matrix exponential. Suppose the matrix exponential of a given matrix times a given vector has to be computed. We develop the approach of Druskin, Greenbaum, and Knizhnerman [*SIAM J. Sci. Comput.*, 19 (1998), pp. 38–54] and interpret the sought-after vector as the value of a vector function satisfying the linear system of ordinary differential equations (ODEs) whose coefficients form the given matrix. The residual is then defined with respect to the initial value problem for this ODE system. The residual introduced in this way can be seen as a backward error. We show how the residual can be computed efficiently within several iterative methods for the matrix exponential. This resolves the question of reliable stopping criteria for these methods. Further, we show that the residual concept can be used to construct new residual-based iterative methods. In particular, a variant of the Richardson method for the new residual appears to provide an efficient way to restart Krylov subspace methods for evaluating the matrix exponential.

Key words. matrix exponential, residual, Krylov subspace methods, restarting, Chebyshev polynomials, stopping criterion, Richardson iteration, backward stability, matrix cosine

AMS subject classifications. 65F60, 65F10, 65F30, 65N22, 65L05

DOI. 10.1137/110820191

1. Introduction.

Matrix functions, and particularly the matrix exponential, have been an important tool in scientific computations for decades (see, e.g., [14, 15, 16, 17, 21, 22, 23]). The lack of a clear notion for a residual for many matrix functions has been a known problem in the iterative computation of matrix functions [3, 14, 44]. Although it is possible to define a residual for some matrix functions such as the inverse or the square root, for many important matrix functions including the matrix exponential, sine, and cosine, no natural notion for residuals seems to exist.

We consider the computation of

$$y = \exp(-A)v \tag{1.1}$$

for a given matrix $A \in \mathbb{C}^{n\times n}$, such that $A + A^*$ is positive semidefinite and a vector $v \in \mathbb{C}^n$. The question is how to evaluate the quality of an approximate solution

$$y_k \approx \exp(-A)v, \tag{1.2}$$

where k refers to the number of steps (iterations) needed to construct y_k. We interpret the vector y as the value of a vector function $y(t)$ at $t = 1$ such that

$$y'(t) = -Ay(t), \quad y(0) = v. \tag{1.3}$$

*Submitted to the journal's Methods and Algorithms for Scientific Computing section January 6, 2011; accepted for publication (in revised form) February 14, 2013; published electronically May 16, 2013. This work was supported by Russian federal program "Scientific and Scientific-Pedagogical Personnel of Innovative Russia" grant 8500 and the Deutsche Forschungsgemeinschaft via RTG 1294.
http://www.siam.org/journals/sisc/35-3/82019.html

†Corresponding author. Department of Applied Mathematics and MESA+ Institute for Nanotechnology, University of Twente, P.O. Box 217, NL-7500 AE Enschede, the Netherlands (mbotchev@nanet.ornl.gov).

‡Institute for Applied and Numerical Analysis, Karlsruhe Institute of Technology, Kaiserstr. 89–93, D-76133 Karlsruhe, Germany (volker.grimm@kit.edu, marlis.hochbruck@kit.edu).

TABLE 1.1
The linear system and matrix exponential residuals. In both cases the sought-after vector is $f(A)v$ with either $f(x) = 1/x$ or $f(x) = \exp(-x)$.

$f(x)$	$1/x$	$\exp(-x)$
Exact solution y	$y = A^{-1}v$	define $y(t) = \exp(-tA)v$, set $y := y(1)$
Residual equation	$Ay = v$	$\begin{cases} y'(t) = -Ay(t), \\ y(0) = v \end{cases}$
Residual for $y_k \approx y$	$r_k = v - Ay_k$	$r_k(t) = -Ay_k(t) - y_k'(t)$
Error ϵ_k	$\epsilon_k = y - y_k$	$\epsilon_k(t) = y(t) - y_k(t)$
Mapping Error $\epsilon_k \to$ residual r_k	$r_k = A\epsilon_k$	$\begin{cases} r_k(t) = \epsilon_k'(t) + A\epsilon_k(t), \\ \epsilon_k(0) = 0 \end{cases}$
Perturbed problem (Backward stability)	$Ay_k = v - r_k$	$\begin{cases} y_k'(t) = -Ay_k(t) - r_k(t), \\ y_k(0) = v \end{cases}$

The exact solution of this initial value problem is given by

$$y(t) = \exp(-tA)v.$$

Assuming now that there is a differentiable vector function $y_k(t)$ such that $y_k(1) = y_k$, we define the residual for $y_k(t) \approx y(t)$ as

$$r_k(t) \equiv -Ay_k(t) - y_k'(t) \tag{1.4}$$

and the error as

$$\epsilon_k(t) \equiv y(t) - y_k(t). \tag{1.5}$$

Obviously, ϵ_k satisfies the initial value problem

$$\epsilon_k'(t) = -A\epsilon_k(t) + r_k(t), \quad \epsilon_k(0) = 0. \tag{1.6}$$

The key point in this residual concept is that $y = \exp(-A)v$ is seen not as a problem on its own but rather as the exact solution formula for the problem (1.3). The latter provides the equation where the approximate solution is substituted to yield the residual. We illustrate this in Table 1.1, where the introduced matrix exponential residual is compared against the conventional residual for a linear system $Ay = v$. As can be seen in Table 1.1, the approximate solution satisfies a perturbed initial value problem, where the perturbation is the residual. Thus, the introduced residual can be seen as a backward error. (See section 4 for residual-based error estimates.) If one is interested in computing the matrix exponential $\exp(-A)$ itself, then the residual can be defined with respect to the matrix initial value problem

$$X'(t) = -AX(t), \quad X(0) = I \tag{1.7}$$

with the exact solution $X(t) = \exp(-tA)$. Checking the norm of $r_k(t)$ in (1.4) is proposed as a possible stopping criterion of Krylov subspace iterations first in [5, 8] and more recently for a similar matrix function in [28].

The contribution of this paper is twofold. First, it turns out that the residual (1.4) can be efficiently computed within several iterative methods for matrix exponential

evaluation. We show how this can be done for some popular Krylov subspace and for Chebyshev polynomial methods for computing $\exp(-A)v$.

Second, we show how the residual notion leads to new algorithms to compute the matrix exponential. The key idea here is to use different approximations to the error ϵ_k by approximating the error equation (1.5) in a suitable way. This either allows us to compute reliable error estimates or to improve the accuracy of the solution.

Two basic Richardson-like iterations are proposed and discussed. When combined with Krylov subspace methods, one of them can be seen as an efficient way to restart the Krylov subspace methods.

The equivalence between problems (1.1) and (1.3) has been widely used in numerical literature and computations. In addition to the work already cited [5, 8, 28], see, e.g., the very first formula in [34] or [21, section 10.1]. Moreover, methods for solving (1.2) are applied to (1.3) (for instance, exponential time integrators [24, 25]) and vice versa [34, section 4]. In [44], van den Eshof and Hochbruck represent the error ϵ_k as the solution of (1.6) and obtain an explicit expression for $\epsilon_k(t)$. This allows them to justify a relative error estimator for their shift-and-invert Lanczos algorithm [44, formula (4.9)]. Although being used, especially in the field of numerical ODEs (see, e.g., [2, 13, 27, 31, 40]), the exponential residual (1.4) does seem to have a potential which has not been fully exploited yet, in particular in matrix computations. Our paper aims at filling this gap.

The paper is organized as follows. Section 2 is devoted to the matrix exponential residual within Krylov subspace methods. In section 3 we show how the Chebyshev iterations can be modified to adopt the residual control. Section 4 presents some simple residual-based error estimates. Richardson iteration for the matrix exponential is the topic of section 5. Numerical experiments are discussed in section 6, and conclusions are drawn in the last section.

Throughout the paper, unless noted otherwise, $\|\cdot\|$ denotes the Euclidean vector norm or the corresponding induced matrix norm.

2. Matrix exponential residual in Krylov subspace methods. Krylov subspace methods have become an important tool for computing matrix functions (see, e.g., [9, 10, 11, 15, 23, 24, 29, 38, 45]). For $A \in \mathbb{C}^{n\times n}$ and $v \in \mathbb{C}^n$ given, the Arnoldi process yields, after k steps, vectors $v_1, \ldots, v_{k+1} \in \mathbb{C}^n$ that are orthonormal in exact arithmetic and span the Krylov subspace $\mathcal{K}_{k+1}(A, v) = \operatorname{span}\{v, Av, \ldots, A^k v\}$ (see, e.g., [17, 39, 46]). If $A = A^*$, the symmetric Lanczos process is usually used instead of the Arnoldi method. Together with the basis vectors v_j, the Arnoldi or Lanczos processes deliver an upper-Hessenberg matrix $\underline{H}_k \in \mathbb{C}^{(k+1)\times k}$, such that the following relation holds:

$$AV_k = V_{k+1}\underline{H}_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T, \qquad V_k^* V_k = I_k, \tag{2.1}$$

where $V_k \in \mathbb{C}^{n\times k}$ has columns $v_1, \ldots, v_k$, $H_k \in \mathbb{C}^{k\times k}$ is the matrix $\underline{H}_k$ without the last row $(0, \ldots, 0, h_{k+1,k})$, $e_k = (0, \ldots, 0, 1)^T \in \mathbb{R}^k$, and I_k denotes the $k \times k$ identity matrix. The first basis vector v_1 is the normalized vector v: $v_1 = v/\beta$, where $\beta = \|v\|$.

2.1. Ritz–Galerkin approximation. An approximation y_k to the matrix exponential $y = \exp(-A)v$ is usually computed as $y_k(1)$ with

$$y_k(t) = V_k u_k(t), \quad u_k(t) \equiv \exp(-tH_k)(\beta e_1), \tag{2.2}$$

where $e_1 = (1, 0, \ldots, 0)^T \in \mathbb{R}^k$. An important property of the Krylov subspace is its scaling invariance: application of the Arnoldi process to tA, $t \in \mathbb{R}$, results in the upper-Hessenberg matrix $t\underline{H}_k$, and the basis vectors $v_1, \ldots, v_{k+1}$ are independent of t.

The approximation y_k can also be obtained via a variational approach, where $y_k \in \mathcal{K}_k(A, v)$ is determined from the condition that the residual r_k defined in (1.4) is orthogonal to $\mathcal{K}_k(A, v)$:

$$0 = V_k^* r_k(t) = V_k^* \big(-y_k'(t) - Ay_k(t)\big), \qquad y_k(t) = V_k u_k(t). \tag{2.3}$$

Inserting (2.1) shows that $u_k(t) : \mathbb{R} \to \mathbb{C}^k$ is the solution of the projected initial value problem

$$u_k'(t) = -H_k u_k(t), \quad u_k(0) = \beta e_1. \tag{2.4}$$

The following simple lemma (cf. [5, formula (4)], [8, formula (29)]) provides an explicit expression for the residual.

LEMMA 2.1. *Let $y_k(t) \approx y(t) = \exp(-tA)v$ be the Krylov subspace approximation given by* (2.2). *Then for any $t \geqslant 0$ the residual $r_k(t)$ for $y_k(t) \approx y(t)$ satisfies*

$$r_k(t) = \beta_k(t) v_{k+1}, \tag{2.5a}$$

where

$$\beta_k(t) = -\beta h_{k+1,k} e_k^T \exp(-tH_k) e_1 = -h_{k+1,k} [u_k(t)]_k. \tag{2.5b}$$

Here, $[u_k(t)]_k$ is the last entry of the vector function $u_k(t)$ defined in (2.2) *and $\beta = \|v\|$.*

Proof. It follows from (2.2) that $y_k'(t) = -V_k H_k \exp(-tH_k)(\beta e_1)$. From the Arnoldi relation (2.1) we have

$$Ay_k(t) = AV_k \exp(-tH_k)(\beta e_1) = (V_k H_k + h_{k+1,k} v_{k+1} e_k^T) \exp(-tH_k)(\beta e_1),$$

which yields the result

$$r_k(t) = -Ay_k(t) - y_k'(t) = \beta_k(t) v_{k+1}$$

with β_k defined in (2.5b). ∎

An immediate conclusion from this lemma is that

$$\|r_k(t)\| = |\beta_k(t)| = |h_{k+1,k} [u_k(t)]_k|.$$

Note that the Krylov subspace approximation (2.2) satisfies the initial condition $y_k(0) = v$ by construction:

$$y_k(0) = V_k(\beta e_1) = \beta v_1 = v.$$

Thus, there is no danger that the residual $r_k(t) = -Ay_k(t) - y_k'(t)$ is small in norm for some $y_k(t)$ approaching a solution of the ODE system $y' = Ay$ with other initial data.

The residual $r_k(t)$ turns out to be closely related to the so-called generalized residual $\rho_k(t)$ defined in [24]. Following [24] (see also [38]), we can write

$$y_k(t) = \beta V_k \exp(-tH_k) e_1 = \frac{1}{2\pi i} \oint_\Gamma e^\lambda V_k (\lambda I + tH_k)^{-1} \beta e_1 \mathrm{d}\lambda,$$
$$y(t) = \exp(-tA)v = \frac{1}{2\pi i} \oint_\Gamma e^\lambda (\lambda I + tA)^{-1} v \mathrm{d}\lambda,$$

where Γ is a closed contour in $\mathbb{C}$ encircling the field of values of $-tA$. Thus, $y_k(t)$ is an approximation to $y(t)$, where the action of the resolvent inverse $x(\lambda,t) = (\lambda I + tA)^{-1}v$ is approximated by k steps of the fully orthogonal method (FOM), which gives the approximation

$$x_k(\lambda,t) = V_k(\lambda I + tH_k)^{-1}\beta e_1.$$

The error can be written as

$$\epsilon_k(t) = y(t) - y_k(t) = \frac{1}{2\pi i}\oint_\Gamma e^\lambda\big(x(\lambda,t) - x_k(\lambda,t)\big)\mathrm{d}\lambda.$$

Since the FOM error $x(\lambda,t) - x_k(\lambda,t)$ is unknown, the authors of [24] replace it by the known FOM residual corresponding to the family of shifted linear systems $(\lambda I + tA)x(\lambda,t) = v$, which is

$$r_k(\lambda,t) = v - (\lambda I + tA)x_k(\lambda,t) = \beta_k(\lambda,t)v_{k+1},$$

where

$$\beta_k(\lambda,t) = -t\beta h_{k+1,k}v_{k+1}e_k^T(\lambda I + tH_k)^{-1}e_1.$$

This leads to the generalized residual

$$\rho_k(t) \equiv \frac{1}{2\pi i}\oint_\Gamma e^\lambda\beta_k(\lambda,t)\mathrm{d}\lambda. \tag{2.6}$$

From (2.5), we obtain

$$\rho_k(t) = t\beta_k(t)v_{k+1} = tr_k(t),$$

hence the generalized residual coincides, up to a factor t, with our matrix exponential residual $r_k(t)$. For the generalized residual, this provides a justification which is otherwise lacking: strictly speaking, there is no reason why the error in the integral expression above can be replaced by the residual. In section 6.4, a numerical test is presented to compare stopping criteria based on $r_k(t)$ and $\rho_k(t)$.

2.2. Shift-and-invert Arnoldi/Lanczos approximations. In the shift-and-invert Arnoldi/Lanczos approximations [18, 19, 35, 44] the Krylov subspace is constructed with respect to the matrix $(I + \gamma A)^{-1}$ with $\gamma > 0$ being a parameter. The Krylov relation for the basis $V_{k+1} \in \mathbb{C}^{n\times(k+1)}$ and the upper-Hessenberg matrix $\underline{\widetilde{H}}_k \in \mathbb{C}^{(k+1)\times k}$ then reads

$$(I + \gamma A)^{-1}V_k = V_{k+1}\underline{\widetilde{H}}_k = V_k\widetilde{H}_k + \widetilde{h}_{k+1,k}v_{k+1}e_k^T, \tag{2.7}$$

where $\widetilde{H}_k \in \mathbb{C}^{k\times k}$ denotes the first k rows of $\underline{\widetilde{H}}_k$. The approximation $y_k(t) \approx \exp(-tA)v$ is then computed as given by (2.2) with H_k defined as

$$H_k = \frac{1}{\gamma}(\widetilde{H}_k^{-1} - I); \tag{2.8}$$

cf. [44]. Relation (2.7) can be rewritten as (cf. formula (4.1) in [44])

$$AV_k = V_kH_k - \frac{\widetilde{h}_{k+1,k}}{\gamma}(I + \gamma A)v_{k+1}e_k^T\widetilde{H}_k^{-1}, \tag{2.9}$$

which leads to the following lemma.

LEMMA 2.2. *Let $y_k(t) \approx y(t) = \exp(-tA)v$ be the shift-and-invert Krylov subspace approximation* (2.2)*, with H_k defined in* (2.8)*. Then for any $t \geqslant 0$ the residual $r_k(t)$ for $y_k(t) \approx y(t)$ satisfies*

$$r_k(t) = \beta_k(t) w_{k+1}, \qquad w_{k+1} = (I + \gamma A) v_{k+1}, \tag{2.10a}$$

where

$$\beta_k(t) = \beta \, \frac{\widetilde{h}_{k+1,k}}{\gamma} \, e_k^T \widetilde{H}_k^{-1} \exp(-tH_k) e_1. \tag{2.10b}$$

Proof. The proof is very similar to that of Lemma 2.1. Instead of the conventional Arnoldi relation (2.1), relation (2.9) should be used. ∎

The residual norm for the shift-and-invert Krylov method then reads

$$\|r_k(t)\| = |\beta_k(t)| \, \|w_{k+1}\|.$$

2.3. Error estimation in Krylov subspace methods. If $y_k(t)$ is the Krylov subspace approximation (2.2) to $y(t) = \exp(-tA)v$, then the error function $\epsilon_k(t)$ defined in (1.5) satisfies the initial value problem (1.6) with the residual $r_k(t)$ given by (1.4).

The key idea is to estimate the error ϵ_k by solving (1.6) approximately by any suitable time integration scheme, for example, by Krylov exponential schemes as discussed, e.g., in [15, section 4] or [24]. The time integration process for solving (1.6) can further be optimized, since by Lemma 2.1, the residual $r_k(t) = \beta_k(t) v_{k+1}$ is a scalar multiple of the $(k+1)$st Arnoldi vector v_{k+1} for all t.

Following an idea by van den Eshof and Hochbruck [44], we propose to approximate ϵ_k by the Galerkin approximation $\widetilde{\epsilon}_k$ with respect to the $(m+k)$th Krylov subspace:

$$\epsilon_k(t) \approx \widetilde{\epsilon}_k(t) = V_{k+m} \delta_k(t), \tag{2.11}$$

where $\widetilde{\epsilon}_k$ satisfies the Galerkin condition

$$V_{k+m}^* \big(\widetilde{\epsilon}_k'(t) + A \widetilde{\epsilon}_k(t) - r_k(t) \big) = 0. \tag{2.12}$$

From the Arnoldi relation (2.1) and by (2.5a), this yields the projected initial value problem

$$\delta_k'(t) = -H_{k+m} \delta_k(t) + \beta_k(t) e_{k+1}, \qquad \delta_k(0) = 0. \tag{2.13}$$

Here, e_{k+1} denotes the $(k+1)$st canonical basis vector in $\mathbb{R}^{k+m}$.

LEMMA 2.3. *Let y_k and y_{k+m} be the kth and the $(k+m)$th Krylov approximation defined in* (2.2)*, respectively. Then the approximated error $\widetilde{\epsilon}_k$ defined in* (2.11) *with δ_k being the solution of* (2.13) *is given by*

$$\widetilde{\epsilon}_k(t) = y_{m+k}(t) - y_k(t). \tag{2.14}$$

Proof. y_k and y_{k+m} satisfy the Galerkin condition (2.3) for k and $k+m$, respectively. From (2.5a) we thus have

$$V_{k+m}^* \big(y_{k+m}' - y_k' + A(y_{k+m} - y_k) \big) = -V_{k+m}^* (r_{k+m} - r_k) = V_{k+m}^* r_k.$$

Using $V_{k+m}^* r_k(t) = \beta_k(t) e_{k+1}$ shows that $y_{m+k} - y_k$ satisfies (2.12), which completes the proof. ∎

Note that (2.14) is just one of the estimates proposed in [44].

The analysis shows that error estimation by the same continued Krylov process is a better option than solving the correction equation (1.6) by a new Krylov process: the latter would mean that we neglect the built-up subspace. In fact, solving the initial value problem (1.6) by another process and then correcting the approximate solution $y_k(t)$ can be seen as a restarting of the Krylov process. We explore this approach further in section 5.

3. Matrix exponential residual for Chebyshev approximations. A well-known method to compute $y_m(t) \approx \exp(-tA)v$ is based on the Chebyshev polynomial expansion (see, for instance, [37, 42]):

$$y_m(t) = P_m(-tA)v = \left[\sum_{k=1}^{m} c_k T_k(-tA) + \frac{c_0}{2} I\right] v. \tag{3.1}$$

Here we assume that the matrix tA can be transformed to have its eigenvalues within the interval $[-1, 1] \subset \mathbb{R}$ (for example, A can be a Hermitian or a skew-Hermitian matrix). Here, T_k is the kth Chebyshev polynomial of the first kind, whose actions on the given vector v can be computed by the Chebyshev recursion

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x), \quad k = 1, 2, \ldots. \tag{3.2}$$

The coefficients c_k can be computed for a large M as

$$c_k = \frac{2}{M} \sum_{j=1}^{M} \exp(\cos(\theta_j)) \cos(k\theta_j), \quad k = 0, 1, \ldots, M, \quad \theta_j = \frac{\pi(j - \frac{1}{2})}{M}, \tag{3.3}$$

which means interpolating $\exp(x)$ at the Chebyshev polynomial roots (see, e.g., [37, section 3.2.3]). This Chebyshev polynomial approximation is used for evaluating different matrix functions in [3].

The well-known Clenshaw algorithm [6] to compute $y_m(t)$ can be modified to provide, along with $y_m(t)$, vectors $y'_m(t)$ and $Ay_m(t)$, so that the exponential residual $r_m(t) \equiv -Ay_m(t) - y'_m(t)$ can be controlled in the course of the iterations. To do this, we use the well-known relations

$$T'_k(x) = kU_{k-1}(x), \tag{3.4}$$

$$xT_k(x) = \frac{1}{2}(T_{k+1}(x) + T_{k-1}(x)), \tag{3.5}$$

$$xU_k(x) = \frac{1}{2}(U_{k+1}(x) + U_{k-1}(x)), \tag{3.6}$$

$$T_k(x) = \frac{1}{2}(U_k(x) - U_{k-2}(x)), \tag{3.7}$$

where $k = 1, 2, \ldots$ and U_k is the kth Chebyshev polynomial of the second kind:

$$U_0(x) = 1, \quad U_1(x) = 2x, \quad U_{k+1}(x) = 2xU_k(x) - U_{k-1}(x), \quad k = 1, 2, \ldots. \tag{3.8}$$

For (3.7) to hold for $k = 1$ we denote $U_{-1}(x) = 0$. From (3.1), (3.4), and (3.6) it follows that

$$\begin{aligned} y'_m(t) &= \left[\sum_{k=1}^{m} \frac{c_k}{t}(-tA)T'_k(-tA)\right] v \\ &= \left[\sum_{k=1}^{m} \frac{c_k k}{2t}(U_k(-tA) + U_{k-2}(-tA))\right] v, \quad m = 1, 2, \ldots. \end{aligned} \tag{3.9}$$

$u_{-2} := -v, \quad u_{-1} := 0, \quad u_0 := v, \quad u_1 := -2t\,Av$
compute c_0
$y := c_0/2\,u_0, \quad y' := 0, \quad \texttt{minusAy} := c_0/(4t)\,u_1$
  for $k = 1, \ldots, N_{\max}$
    $u_2 := -2t\,Au_1 - u_0$
    compute c_k
    $y := y + c_k/2\,(u_1 - u_{-1})$
    $y' := y' + c_k k/(2t)\,(u_1 + u_{-1})$
    $\texttt{minusAy} := \texttt{minusAy} + c_k/(4t)\,(u_2 - u_{-2})$
    $u_{-2} := u_{-1}$
    $u_1 := u_0$
    $u_0 := u_1$
    $u_1 := u_2$
    $\texttt{resnorm} := \|\texttt{minusAy} - y'\|$
    if $\texttt{resnorm} < \texttt{toler}$
      break
    end
  end

FIG. 3.1. *Chebyshev expansion algorithm to compute the vector* $y_{N_{\max}}(t) \approx \exp(-tA)v$. *The input parameters are* $A \in \mathbb{C}^{n\times n}$, $v \in \mathbb{C}^n$, $t > 0$, *and* $\texttt{toler} > 0$. *It is assumed that the eigenvalues* λ *of* tA *satisfy* $-1 \leqslant \lambda \leqslant 1$.

Similarly, from (3.1), (3.5), and (3.7), we obtain

$$\begin{aligned} -Ay_m(t) &= \left[\sum_{k=1}^{m} \frac{c_k}{2t}(T_{k+1}(-tA) + T_{k-1}(-tA)) - \frac{c_0}{2}A\right] v \\ &= \left[\sum_{k=1}^{m} \frac{c_k}{2t}(U_{k+1}(-tA) - U_{k-3}(-tA)) - \frac{c_0}{2}A\right] v, \quad m = 1, 2, \ldots, \end{aligned} \tag{3.10}$$

where we define $U_{-2}(x) = -1$.

These recursions can be used to formulate an algorithm for computing $y_m(t) \approx \exp(-tA)v$ that controls the residual $r_m(t) = -Ay_m(t) - y_m'(t)$; see Figure 3.1. Just like the original Chebyshev recursion algorithm for the matrix exponential, it requires one action of the matrix A per iteration. To be able to control the residual, more vectors have to be stored than in the conventional algorithm: 8 instead of 4.

4. Residual-based error estimates. A different interpretation of the residual $r_k(t)$ defined in (1.4) is to view it as the backward error for $y_k(t)$:

$$y_k'(t) = -Ay_k(t) - r_k(t), \quad y(0) = v, \tag{4.1}$$

is a perturbation of the original problem (1.3). The forward error $\epsilon_k = y - y_k$ solving the initial value problem (1.6) is given by

$$\epsilon_k(t) = \int_0^t \exp\bigl(-(t-s)A\bigr) r_k(s)\mathrm{d}s \tag{4.2}$$

(variation-of-constants formula). This formula can be used to obtain error bounds in terms of the norms of the matrix exponential and the residual; cf. [47].

In the following, we assume the existence of constants $C_A > 0$ and $\omega \geqslant 0$ such that

$$\|\exp(-tA)\| \leqslant C_A \mathrm{e}^{-t\omega}, \qquad t \geqslant 0. \tag{4.3}$$

Remark. For the spectral norm, this bound is satisfied with $C_A = 1$ if the field of values of A is contained in the complex halfplane $\mathbb{C}_\omega := \{z \in \mathbb{C}\colon \operatorname{Re} z \geqslant \omega\}$. In this case, $\omega = -\mu(-A)$, where $\mu(B) = \lambda_{\max}(\frac{1}{2}(B + B^*))$ is the logarithmic norm of matrix B; cf. [7, 21]. If A is diagonalizable, $X^{-1}AX = \Lambda$, then (4.3) holds for an arbitrary matrix norm induced by a vector norm with $C_A = \kappa(X) = \|X\|\,\|X^{-1}\|$ if the spectrum of A is contained in $\mathbb{C}_\omega$.

Our analysis makes use of the so-called φ-functions defined as

$$\varphi_k(z) = \int_0^1 \mathrm{e}^{(1-\theta)z} \frac{\theta^{k-1}}{(k-1)!} \mathrm{d}\theta, \quad k \geq 1. \tag{4.4}$$

These functions satisfy $\varphi_k(0) = 1/k!$ and the recurrence relation

$$\varphi_{k+1}(z) = \frac{\varphi_k(z) - \varphi_k(0)}{z}, \quad \varphi_0(z) = \mathrm{e}^z. \tag{4.5}$$

Assumption (4.3) yields

$$\|\varphi_k(-tA)\| \leq \int_0^1 \|\mathrm{e}^{-t(1-\theta)A}\| \frac{\theta^{k-1}}{(k-1)!} \mathrm{d}\theta \leqslant C_A\, \varphi_k(-t\omega) \leqslant C_A \frac{1}{k!}.$$

LEMMA 4.1. *Let $A \in \mathbb{C}^{n\times n}$ satisfy* (4.3)*. Then for all $t \geqslant 0$, the solution ϵ_k of* (1.6) *is bounded by*

$$\|\epsilon_k(t)\| \leqslant C_A\, t\, \varphi_1(-t\omega)\mu_k, \qquad k = 0, 1, \ldots, \tag{4.6}$$

where

$$\mu_k := \max_{0\leqslant s\leqslant t} \|r_k(s)\|. \tag{4.7}$$

Proof. The variation-of-constants formula (4.2) yields

$$\begin{aligned} \|\epsilon_k(t)\| &\leqslant \int_0^t \|\exp(-(t-s)A) r_k(s)\| \mathrm{d}s \\ &\leqslant C_A t \int_0^1 \exp(-\omega t(1-\theta)) \|r_k(t\theta)\| \mathrm{d}\theta \\ &\leqslant C_A\, t\, \varphi_1(-\omega t) \max_{0\leqslant s\leqslant t} \|r_k(s)\|. \end{aligned} \tag{4.8}$$

This proves the bound. ∎

5. Richardson iteration for the matrix exponential. The notion of the residual allows us to formulate a Richardson method for the matrix exponential.

5.1. Preconditioned Richardson iteration. To motivate our approach, we start by considering the preconditioned Richardson iterative method for solving the linear system $Ax = v$. Given an initial guess $x_0 \in \mathbb{C}^n$ and a preconditioner $M \approx A$, Richardson iteration reads

$$x_{k+1} = x_k + M^{-1}r_k, \qquad r_k = v - Ax_k, \qquad k = 0, 1, \ldots. \tag{5.1}$$

Note that $M^{-1}r_k$ is an approximation to the unknown error $A^{-1}v - x_k = A^{-1}r_k$. The recursion (5.1) for the iterates yields

$$r_{k+1} = r_k - AM^{-1}r_k = (I - AM^{-1})r_k = (M - A)M^{-1}r_k. \tag{5.2}$$

Hence, in general, the convergence will be linear:

$$\|r_k\| \leqslant \|(M - A)M^{-1}\|^k \, \|r_0\| \tag{5.3}$$

if $\|(M - A)M^{-1}\| < 1$.

Analogously to (5.1), one can formulate the Richardson method for the matrix exponential by approximating the solution $\epsilon_k(t)$ of (1.6) by the solution $\widetilde{\epsilon}_k(t)$ of the initial value problem

$$\widetilde{\epsilon}_k'(t) = -M\widetilde{\epsilon}_k(t) + r_k(t), \quad \widetilde{\epsilon}_k(0) = 0, \tag{5.4}$$

where $M \approx A$ is a preconditioning matrix and r_k is defined in (1.4). Finally, the update step of Richardson iteration is defined as

$$y_{k+1}(t) = y_k(t) + \widetilde{\epsilon}_k(t), \qquad k = 0, 1, \ldots. \tag{5.5}$$

Just as for solving linear systems, M has to compromise between the approximation quality $M \approx A$ and the ease of solving (5.4).

In fact, the exponential Richardson method can be seen as a special version of waveform relaxation methods for solving ODEs; see, e.g., [26, 32] and references given there. This is easily seen from (5.4) and (5.5), which can be written in the equivalent form

$$y_{k+1}'(t) + My_{k+1}(t) = (M - A)y_k(t), \qquad y_{k+1}(0) = y_k(0).$$

A possible choice is to use multigrid preconditioning; cf. [32].

From (5.5), (1.4), and (5.4) we have

$$\begin{aligned} r_{k+1} &= -Ay_{k+1} - y_{k+1}' \\ &= -Ay_k - A\widetilde{\epsilon}_k - y_k' - \widetilde{\epsilon}_k' \\ &= (M - A)\widetilde{\epsilon}_k. \end{aligned} \tag{5.6}$$

Taking into account that

$$\widetilde{\epsilon}_k(t) = \int_0^t \exp\bigl(-(t - s)M\bigr)r_k(s)\mathrm{d}s,$$

we obtain the following recursion for the residuals (cf. [33])

$$r_{k+1}(t) = (M - A)\widetilde{\epsilon}_k(t) = (M - A)\int_0^t \exp\bigl(-(t - s)M\bigr)r_k(s)\mathrm{d}s. \tag{5.7}$$

This recurrence should be compared to the corresponding recurrence (5.2) for Richardson iteration for linear systems.

Since $M \approx A$, we now assume that there are constants $C_M > 0$ and $\widetilde{\omega} \geqslant 0$ such that the preconditioning matrix M satisfies

$$\|\exp(-tM)\| \leqslant C_M \mathrm{e}^{-t\widetilde{\omega}}, \qquad t \geqslant 0. \tag{5.8}$$

It is reasonable to assume $\widetilde{\omega} \approx \omega$, where ω is defined in (4.3).

THEOREM 5.1. *If* (5.8) *holds, then the residual* r_k *corresponding to the iteration* (5.4), (5.5) *satisfies* (5.7). *Moreover,* r_k *is bounded by*

$$\|r_k(t)\| \leqslant \Big(C_M \|M - A\|\, t\Big)^k \mathrm{e}^{-t\widetilde{\omega}} \varphi_k(t\widetilde{\omega})\mu_0, \qquad k = 0, 1, \ldots,$$

where $\mu_0 = \max_{0 \leqslant s \leqslant t} \|r_0(s)\|$.

Proof. Solving the recursion (5.7) yields

$$\|r_k(t)\| = \bigg\| \int_0^t (M - A) \exp\big(-(t - s_k)M\big) \int_0^{s_k} (M - A) \exp\big(-(s_k - s_{k-1})M\big) \cdots \int_0^{s_2} (M - A) \exp\big(-(s_2 - s_1)M\big) r_0(s) \mathrm{d}s_1 \ldots \mathrm{d}s_k \bigg\|.$$

Using (5.8) and the definition of the φ-functions (4.4) shows

$$\begin{aligned} \|r_k(t)\| &\leqslant \mu_0 \Big(C_M \|M - A\|\Big)^k \mathrm{e}^{-t\widetilde{\omega}} \int_0^t \int_0^{s_k} \cdots \int_0^{s_2} \mathrm{e}^{s_1\widetilde{\omega}} \mathrm{d}s_1 \ldots \mathrm{d}s_k \\ &\leqslant \mu_0 \Big(C_M \|M - A\|\, t\Big)^k \mathrm{e}^{-t\widetilde{\omega}} \varphi_k(t\widetilde{\omega}) \end{aligned}$$

by using an induction argument and the relation

$$\int_0^s t^k \varphi_k(t\widetilde{\omega})\, \mathrm{d}t = s^{k+1} \varphi_{k+1}(s\widetilde{\omega}), \qquad k = 0, 1, 2, \ldots. \qquad \square$$

Remark. Since $\widetilde{\omega} \geqslant 0$ by assumption, we have

$$t^k \mathrm{e}^{-t\widetilde{\omega}} \varphi_k(t\widetilde{\omega}) = \mathrm{e}^{-t\widetilde{\omega}} \int_0^t \mathrm{e}^{(t-s)\widetilde{\omega}} \frac{s^{k-1}}{(k-1)!} \mathrm{d}s \leqslant \frac{t^k}{k!}, \qquad t \geqslant 0,$$

showing that the convergence is superlinear. If $\|(M - A)M^{-1}\| < 1$, then

$$\|r_k(t)\| \leqslant \|(M - A)M^{-1}\|^k \big(C_M \|M\| t\big)^k \mathrm{e}^{-t\widetilde{\omega}} \varphi_k(t\widetilde{\omega})\mu_0, \qquad k = 0, 1, \ldots.$$

Hence, asymptotically, the iteration for the matrix exponential shows a favorable convergence behavior compared to the linear convergence of Richardson iteration for linear systems.

Remark. Note that it is crucial to consider t in a finite interval, $t \in [0, T]$ with a fixed $T < \infty$, to obtain the superlinear convergence. For instance, if $\widetilde{\omega} > 1$, we have the bound

$$t^k \mathrm{e}^{-t\widetilde{\omega}} \varphi_k(t\widetilde{\omega}) \leqslant \int_0^\infty \mathrm{e}^{-s\widetilde{\omega}} \frac{s^{k-1}}{(k-1)!} \mathrm{d}s = \left(\frac{1}{\widetilde{\omega}}\right)^k$$

showing linear convergence uniformly in $t \in [0, \infty)$. This is in correspondence with many theoretical results for waveform relaxation methods where the emphasis is often on the convergence in the infinite interval $[0, \infty)$ in contrast to our application.

An important practical issue hindering the use of the exponential Richardson iteration is the necessity to store the vectors $r_k(t)$ for different t. To achieve a good accuracy, sufficiently many samples of $r_k(t)$ have to be stored. Our limited experience indicates that the exponential Richardson iteration can be of interest if the accuracy requirements are relatively low, say up to 10^{-5}. In the experiments described in section 6.3 just 20 samples were sufficient to get the residual below tolerance 10^{-4} for a matrix of size $n = 10^4$.

5.2. Krylov restarting via Richardson iteration. The exponential Richardson iteration (5.5) allows for great flexibility in the choice of the approximated error $\widetilde{\epsilon}_k(t) \approx \epsilon_k$. For instance, solving (1.6) by yet another Krylov process is closely related to restarting the matrix exponential; cf. [1, 5, 12, 36].

From Lemmas 2.1 and 2.2 we have

$$r_k(t) = \beta_k(t) w_{k+1},$$

where $w_{k+1} = v_{k+1}$ for the standard Arnoldi process and $w_{k+1} = (I + \gamma A)v_{k+1}$ for shift-and-invert Arnoldi method. We thus propose to approximate

$$\epsilon_k(t) \approx \widetilde{\epsilon}_{k,m}(t) = \widehat{V}_m \delta_{k,m}(t), \tag{5.9}$$

where the columns of $\widehat{V}_m$ are defined via an Arnoldi process for the Krylov subspace $\mathcal{K}_m(A, w_{k+1})$:

$$A\widehat{V}_m = \widehat{V}_{m+1}\underline{\widehat{H}}_m = \widehat{V}_m \widehat{H}_m + \widehat{h}_{m+1,m}\widehat{v}_{m+1}e_m^T, \qquad \widehat{V}_m^*\widehat{V}_m = I_m. \tag{5.10}$$

$\widetilde{\epsilon}_{k,m}(t)$ is determined from the Galerkin condition

$$\widehat{V}_m^*\big(\widetilde{\epsilon}'_{k,m}(t) + A\widetilde{\epsilon}_{k,m}(t) - r_k(t)\big) = 0. \tag{5.11}$$

Similar to the presentation in section 2.3, $\delta_{k,m}(t)$ satisfies

$$\begin{aligned} \delta'_{k,m}(t) &= -\widehat{H}_m \delta_{k,m}(t) + \widehat{V}_m^* r_k(t) \\ &= -\widehat{H}_m \delta_{k,m}(t) + \beta_k(t)\|w_{k+1}\| e_1, \qquad \delta_{k,m}(0) = 0. \end{aligned} \tag{5.12}$$

LEMMA 5.2. *Let $\widetilde{\epsilon}_{k,m}(t) \approx \epsilon_k(t)$ be the Galerkin approximation* (5.9) *with $\delta_{k,m}(t)$ being the solution of* (5.12)*. Then the residual*

$$\widetilde{r}_{k,m}(t) = -\widetilde{\epsilon}'_{k,m}(t) - A\widetilde{\epsilon}_{k,m}(t)$$

is given by

$$\widetilde{r}_{k,m}(t) = -r_k(t) + \widehat{\beta}_m(t)\widehat{v}_{m+1}, \tag{5.13}$$

where

$$\widehat{\beta}_m(t) = -\widehat{h}_{m+1,m} e_m^T \delta_{k,m}(t). \tag{5.14}$$

Proof. Using the Arnoldi relation (5.10) yields

$$\begin{aligned} \widetilde{r}_{k,m}(t) &= -\widehat{V}_m\big(\widehat{H}_m \delta_{k,m}(t) + \widehat{V}_m^* r_k(t)\big) - A\widehat{V}_m \delta_{k,m}(t) \\ &= -r_k(t) + \widehat{\beta}_m(t)\widehat{v}_{m+1} \end{aligned}$$

since $r_k(t) \in \operatorname{span}\{\widehat{v}_1\}$ and thus $\widehat{V}_m\widehat{V}_m^* r_k(t) = r_k(t)$. ∎

Analogously, for the shift-and-invert Krylov process we obtain

$$\widetilde{r}_{k,m}(t) = -r_k(t) + \widehat{\beta}_m(t)\widehat{w}_{m+1}, \qquad \widehat{w}_{m+1} = (I + \gamma A)\widehat{v}_{m+1},$$

where

$$\widehat{\beta}_m(t) = \widehat{\widehat{h}}_{m+1,m}\gamma^{-1}e_m^T\Big(\widehat{\widehat{H}}_m\Big)^{-1}\delta_{k,m}(t),$$

by Lemma 2.2.

For the residual of the $(k+1)$st Richardson iterate (5.5), we thus have

$$r_{k+1} = r_k + \widetilde{r}_{k,m} = \widehat{\beta}_m(t)\widehat{w}_{m+1}, \tag{5.15}$$

where $\widehat{w}_{m+1} = \widehat{v}_{m+1}$ for the standard Krylov process and $\widehat{w}_{m+1} = (I + \gamma A)\widehat{v}_{m+1}$ for the shift-and-invert method. Hence, the residual $r_{k+1}(t)$ is, just as in Lemmas 2.1 and 2.2, a scalar time-dependent function times a constant vector. Since this is true for all iterations $k = 0, 1, 2, \ldots$, the analysis holds for repeated restarts as well.

6. Numerical experiments. All our numerical experiments have been carried out with MATLAB on a Linux and Mac PC. Unless reported otherwise, the initial vector v is taken to be the normalized vector with equal entries. Except for section 6.1, the error reported is the relative error norm with respect to a reference solution computed by the Expokit method [41]. The error reported for Expokit is the error estimate provided by this code.

6.1. Residual in Chebyshev iteration. The following test is carried out for the Chebyshev iterative method with incorporated residual control (see algorithm in Figure 3.1). We compute $\exp(-A)v$, where $v \in \mathbb{R}^n$ is a random vector with mean zero and standard deviation one. In the test, the matrix $A \in \mathbb{R}^{n\times n}$ is diagonal with diagonal entries evenly distributed between -1 and 1. Note that the corresponding ODE is not stiff. The plots of the error and residual norms are presented in Figure 6.1.

6.2. A convection-diffusion problem. In the next several numerical experiments the matrix A is taken to be the standard five-point central difference discretization of the following convection-diffusion operator acting on functions defined on the

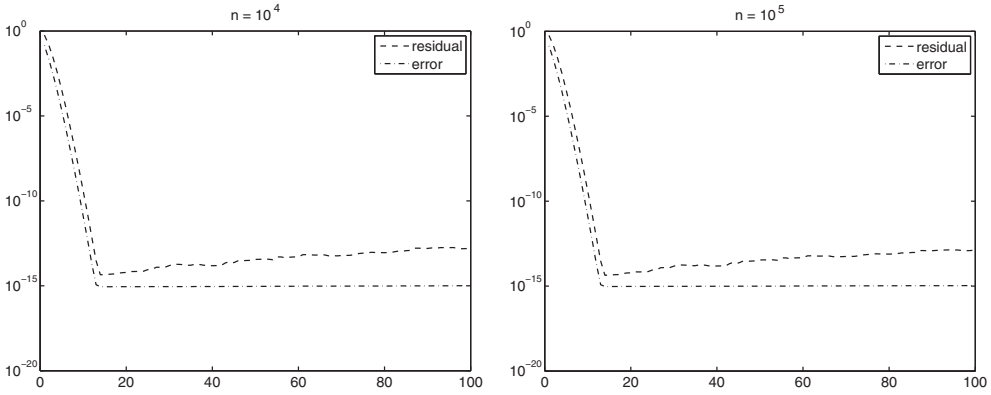

FIG. 6.1. *Residual and true error norms in the Chebyshev algorithm to compute* $y_m \approx \exp(-A)v$ *against iteration number* m *for the example of section* 6.1 *with* $n = 10^4$ *(top) and* $n = 10^5$ *(bottom).*

unit square $(x, y) \in [0, 1]^2$:

$$L[u] = -(D_1 u_x)_x - (D_2 u_y)_y + Pe(v_1 u_x + v_2 u_y),$$
$$D_1(x,y) = \begin{cases} 10^3 & (x,y) \in [0.25, 0.75]^2, \\ 1 & \text{otherwise}, \end{cases} \qquad D_2(x,y) = \frac{1}{2} D_1(x,y),$$
$$v_1(x,y) = x + y, \qquad v_2(x,y) = x - y.$$

To guarantee that the convection terms yield an exactly skew-symmetric matrix, we rewrite the convection terms in the form

$$v_1 u_x + v_2 u_y = \frac{1}{2}(v_1 u_x + v_2 u_y) + \frac{1}{2}((v_1 u)_x + (v_2 u)_y)$$

before discretizing them; cf. [30]. This is possible because the velocity field (v_1, v_2) is divergence free. The operator L is set to satisfy the homogeneous Dirichlet boundary conditions. The discretization is done on a 102×102 or 402×402 uniform mesh, producing an $n \times n$ matrix A of size $n = 10^4$ or $n = 16 \times 10^4$, respectively. The Peclet number varies from $Pe = 0$ (no convection, $A = A^T$) to $Pe = 10^3$, which on the finer mesh means $\|A - A^T\|_1 / \|A + A^T\|_1 \approx 8 \times 10^{-4}$. To get an impression on the field of values of A, which might be instructive regarding the results of sections 4 and 5, we report that for the mesh 402×402 and $Pe = 10^3$ assumption (4.3) is satisfied with $C_A = 1$, $\omega = 0$.

6.3. Exponential Richardson iteration. In this section we apply the exponential Richardson iteration (5.4), (5.5) to compute the vector $\exp(-A)v$ for the convection-diffusion matrices A described in section 6.2. The mesh is taken to be 102×102. As discussed above, to be able to update the residual and to solve the initial value problem (5.4), we need to store the values of $r_k(t)$ for different t spanning the time interval of interest. Too few samples may result in an accuracy loss in the interpolation stage. On the other hand, it can be prohibitively expensive to store many samples. Therefore, in its current form, the method does not seem practical if a high accuracy is needed. On the other hand, it turns out that a moderate accuracy up to 10^{-5} can be reached with relatively few samples (≈ 20).

We organize the computations in the method as follows. The residual vector function $r_k(t)$ is stored as 20 samples. At each iteration, the initial value problem (5.4) is solved by the MATLAB `ode15s` ODE solver, and the values of the right-hand side function $-M\widetilde{\epsilon}_k(t) + r_k(t)$ are interpolated using the stored samples. The `ode15s` solver is run with tolerances determined by the final required accuracy and produces the solution $\widetilde{\epsilon}_k(t)$ in the form of its twenty samples. Then, the solution and residual are updated according to (5.5) and (5.7), respectively.

We have chosen M to be the tridiagonal part tridiag(A) of the matrix A. Assumption (5.8) is satisfied with $C_M = 1$, $\widetilde{\omega} = 0$. Table 6.1 and Figure 6.2 contain results of the test runs. Except for the Richardson method, as a reference we use the Expokit code [41] with the maximal Krylov dimension 100. Note that Expokit provides a much more accurate solution than requested by the tolerance `toler` $= 10^{-4}$. It is rather difficult to compare the total computational work of the Expokit and Richardson methods exactly. We restrict ourselves to the matrix-vector part of the work. In the Richardson method this work consists of the matrix-vector multiplication (matvec) with $M - A$ in (5.7) and the work done by the `ode15s` solver. The matvec with

TABLE 6.1

Performance of the exponential Richardson method for the convection-diffusion test problem of section 6.3 *with* `toler` $= 10^{-4}$, $M = \text{tridiag}(A)$. *The CPU times are measured on a* 3 *GHz Linux PC. We emphasize that the CPU time measurements are made in MATLAB and thus are only an approximate indication of the actual performance.*

	Flops/n, CPU time, s	Matvecs A / steps	LU $I + \alpha M$	Solving $I + \alpha M$	Error
		$Pe = 0$			
Expokit	4590, 2.6	918 matvecs	—	—	1.20e−11
exp. Richardson	2192, 1.7	8 steps	24	176	2.21e−04
		$Pe = 10$			
Expokit	4590, 2.6	918 matvecs	—	—	1.20e−11
exp. Richardson	2202, 1.7	8 steps	29	176	2.25e−04
		$Pe = 100$			
Expokit	4590, 2.6	918 matvecs	—	—	1.20e−11
exp. Richardson	2492, 1.9	9 steps	31	200	4.00e−04

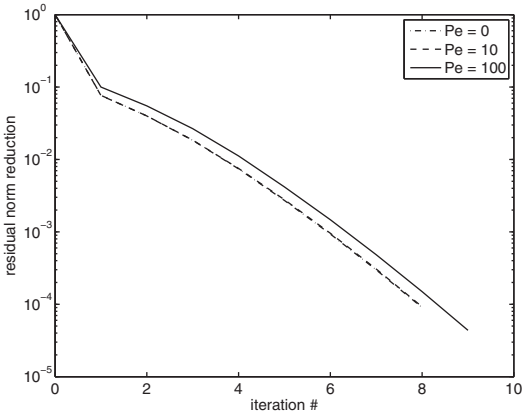

FIG. 6.2. *Convergence history of the exponential Richardson iteration for the numerical example of section* 6.3.

bidiagonal[1] $M - A$ costs about $3n$ flops times 20 samples, in total $60n$ flops[2]. The linear algebra work in `ode15s` is essentially tridiagonal matvecs, LU factorizations, and back/forward substitutions with (possibly shifted and scaled) M. According to [17, section 4.3.1], tridiagonal LU factorization and back/forward substitution require about $2n$ flops each. A matvec with tridiagonal M is $5n$ flops. Thus, in total exponential Richardson costs $60n$ flops times the number of iterations plus $2n$ flops times the number of LU factorizations and back/forward substitutions plus $5n$ flops times the total number of ODE solver steps. The matvec work in Expokit consists of matvecs with pentadiagonal A, which is about $9n$ flops.

From Table 6.1 we see that exponential Richardson is approximately twice as cheap as Expokit. As expected from the convergence estimates, the exponential Richardson iteration converges much faster than the conventional Richardson iteration for solving a linear system $Ax = v$ would do. For these A and v, 8–9 iterations of the conventional Richardson would only give a residual reduction by a factor of ≈ 0.99.

[1] Note M is the tridiagonal part of A which is the standard five-point finite difference discretization of the convection-diffusion operator.

[2] We use definition of flop from [17, section 1.2.4].

6.4. Experiments with Krylov–Richardson iteration. In this section we present some numerical experiments with the Krylov–Richardson method presented in section 5.2. We now briefly describe the other methods to which Krylov–Richardson is compared.

Together with the classical Arnoldi/Lanczos method [10, 15, 23, 38], we have tested the shift-and-invert method of Van den Eshof and Hochbruck [44]. We have implemented the method exactly as described in their paper with a single modification. In particular, in all the tests the shift parameter γ is set to $0.1t_{\text{end}}$, as done in the experiments of [44]. (In general γ should to be chosen depending on the accuracy prescribed [44, Table 3.1].) Furthermore, the relaxed stopping criterion strategy for the inner iterative shift-and-invert solvers is employed. The only thing we have changed is the stopping criterion of the outer Krylov process. To be able to exactly compare the computational work, we can switch from the stopping criterion of Van den Eshof and Hochbruck [44, formula (4.9)] to the residual stopping criterion (see Lemma 2.2). Note that the relaxed strategy for the inner shift-and-invert solver is then also based on the residual norm and not on the error estimate.

Since the Krylov–Richardson method is essentially a restarting technique, it has to be compared with other existing restarting techniques. Note that a number of restarting strategies have recently been developed [1, 12, 20, 36, 43]. We have used the restarting method described in [36]. This choice is motivated by the fact that the method from [36] turns out to be algorithmically very close to our Krylov–Richardson method. In fact, the only essential difference is the handling of the projected problem. In the method [36] the projected matrix H_k built up at every restart is appended to a larger matrix $\widetilde{H}_{*+k}$. There, the projected matrices from each restart are accumulated. Thus, if 10 restarts of 20 steps are done, we have to compute the matrix exponential of a 200×200 matrix. In our method, the projected matrices are not accumulated, so at every restart we deal with a 20×20 matrix. The price to pay, however, is the solution of the small initial value problem (5.12).

In our implementation, at each Krylov–Richardson iteration the initial value problem (5.12) is solved by the `ode15s` ODE solver from MATLAB. To save computational work, it is essential that the solver be called most of the time with a relaxed tolerance. (In our code we set the tolerance to 1% of the current residual norm.) This is sufficient to estimate the residual accurately. Only when the actual solution update takes place (see formula (5.5)) do we solve the projected initial value problem to very high accuracy.

Since the residual time dependence in Krylov–Richardson is given by a scalar function, little storage is needed for the lookup table. Based on the required accuracy, the `ode15s` solver automatically determines how many samples need to be stored. (In our experiments this usually did not exceed 300.) This happens at the end of each restart or when the stopping criterion is satisfied. Further savings in computational work can be achieved by a polynomial fitting: at each restart the newly computed values of the function β_k (see (2.5a)) are approximated by a best-fit polynomial of a moderate degree. (In all experiments the degree was set to 6.) If the fitting error is too large (this depends on the required tolerance), the algorithm proceeds as before. Otherwise, the β_k function is replaced by its best-fit polynomial. This allows a faster solution of the projected initial value problem (5.12) through an explicit formula containing the φ-functions defined in (4.4).

We now present an experiment showing the importance of a proper stopping criterion. We compute $\exp(-5A)v$ for A being the convection-diffusion operator discretized on a uniform mesh 102×102 with $Pe = 100$. The tolerance is set to $\texttt{toler} = 10^{-5}$. We

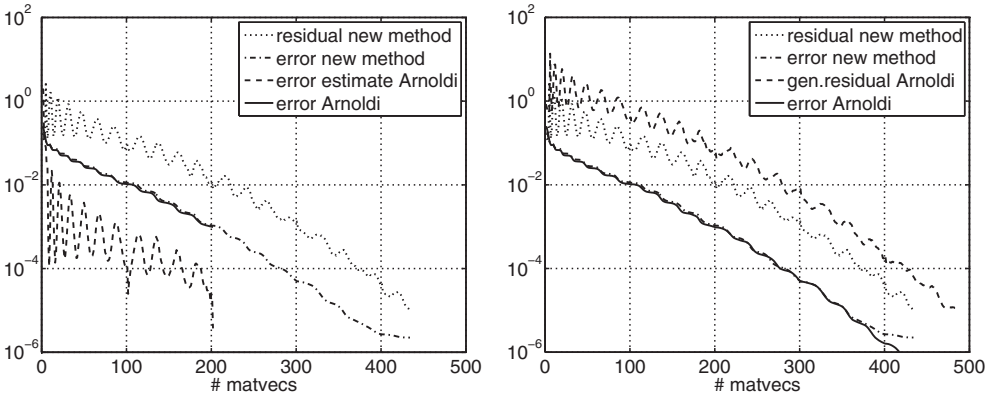

FIG. 6.3. *First numerical example of section* 6.4. *Convergence of the conventional Arnoldi method with two existing stopping criteria and Krylov–Richardson with the residual-based stopping criterion for tolerance* `toler` $= 10^{-5}$. *Left: stopping criterion* [44, formula (4.9)], *Arnoldi stops too early (*201 *matvecs,* 2.6 *s CPU time, error* $1.0e-03$*). Right: generalized residual criterion, Arnoldi stops too late (*487 *matvecs,* 139 *s CPU time, error* $4.9e-08$*). Parameters of the Krylov–Richardson run for both plots (*434 *matvecs,* 11 *s CPU time, error* $2.2e-06$*). The CPU measurements (on a* 3 *GHz Linux PC) are made in MATLAB and thus are only an indication of the actual performance.*

let the usual Arnoldi method, restarted every 100 steps, run with the stopping criterion of [44, formula (4.9)] and with the stopping criterion of [24] based on the generalized residual (2.6). We emphasize that the stopping criterion given by [44, formula (4.9)] is proposed for the Arnoldi method with the shift-and-invert strategy and it works, in our limited experience, very well as soon as shift-and-invert is employed. However, the stopping criteria based on the difference of two consecutive approximations are used in Krylov methods not only with shift-and-invert (see, e.g., [4]) and it is instructive to see possible implications of this. Together with Arnoldi, the Krylov–Richardson method is run with the residual-based stopping criterion. The convergence plots are shown in Figure 6.3. As we see, both existing stopping criteria turn out to be far from optimal in this test. With the residual-based stopping criterion, the Arnoldi method required 438 matvecs and 78 s CPU time to obtain an adequate accuracy of 4.5e−7.

To facilitate a fair comparison between the conventional Arnoldi and the Krylov–Richardson methods, in all the other tests we use the residual-based stopping criterion for both methods. Table 6.2 and Figure 6.4 contain the results of the test runs to compute $\exp(-A)v$ for tolerance `toler` $= 10^{-8}$. We show the results on two meshes for two different Peclet numbers only; the results for other Peclet numbers are quite similar.

The first observation we make is that the CPU times measured in MATLAB seem to favor the Expokit code, disregarding the actual matvec values. We emphasize that when the shift-and-invert strategy is not used, the main computational cost in the three methods, Expokit, Arnoldi, and Krylov–Richardson, are k steps of the Arnoldi/Lanczos process. The differences among the three methods correspond to the rest of the computational work, which is $\mathcal{O}(k^3)$, at least if not too many restarts are made.

The second observation is that the convergence of the Krylov–Richardson iteration is essentially the same as of the classical Arnoldi/Lanczos method. This is not influenced by the restart value or by the shift-and-invert strategy. Theoretically, this is to be expected: the former method applies Krylov for the φ_1 function, the latter for the exponential; for both functions similar convergence estimates hold true, though they are slightly more favorable for the φ_1 function; cf. [23].

TABLE 6.2

Second numerical example of section 6.4. *Results of the test runs of the Krylov–Richardson and Arnoldi with the residual-based stopping criterion. The CPU times are measured on a* 2 *GHz Mac PC (mesh* 102×102*) and on a* 3 *GHz Linux PC (mesh* 402×402*). We emphasize that the CPU time measurements are made in MATLAB and thus are only an approximate indication of the actual performance.*

	Restart/shift-and-invert	Total matvecs or LU actions	CPU time	Error
mesh 102×102, $Pe = 100$				
Expokit	restart 15	1343	2.2	3.61e−09
Arnoldi	restart 15	250	26.4	1.45e−10
New method	restart 15	240	6.7	1.94e−09
Expokit	restart 100	1020	7.6	1.33e−11
Arnoldi	restart 100	167	7.9	1.21e−10
New method	restart 100	168	11.8	1.14e−10
Arnoldi	SaI/GMRES[a]	980 (11 steps)	17.8	3.29e−08
New method	SaI/GMRES[a]	60 (10 steps)	1.7	1.67e−08
Arnoldi	SaI/sparse LU	"11" (10 steps)	1.7	3.62e−09
New method	SaI/sparse LU	"11" (10 steps)	1.8	1.61e−10
mesh 402×402, $Pe = 1000$				
Expokit	restart 15	1445	21	4.36e−09
Arnoldi	restart 15	244	11	1.13e−10
New method	restart 15	254	15	2.62e−09
Expokit	restart 100	1020	69	1.33e−11
Arnoldi	restart 100	202	34	1.06e−10
New method	restart 100	200	35	3.62e−10
Arnoldi	SaI/GMRES[a]	1147 (15 steps)	80	5.68e−08
New method	SaI/GMRES[a]	97 (12 steps)	6.2	1.28e−08
Arnoldi	SaI/sparse LU	"12" (11 steps)	46	3.06e−08
New method	SaI/sparse LU	"13" (12 steps)	50	2.07e−10

[a] GMRES(100) with SSOR preconditioner

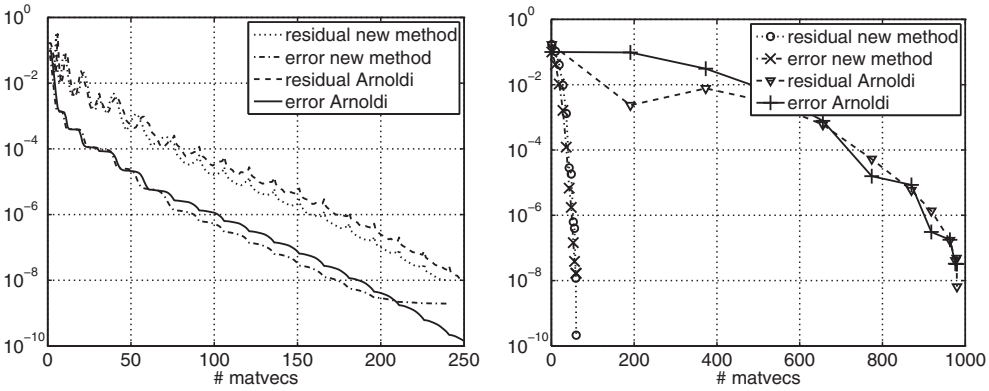

FIG. 6.4. *Second numerical example of section* 6.4 *with* $Pe = 100$. *Convergence plots of the Arnoldi/Lanczos and the new Krylov–Richardson methods on a* 102×102 *mesh. Left: restart every* 15 *steps. Right: shift-and-invert strategy with GMRES. The peaks in the residual plots on the left correspond to the restarts.*

When no shift-and-invert strategy is applied, the gain we have with Krylov–Richardson is twofold. First, a projected problem of much smaller size has to be solved. This is reflected by the difference in the CPU times of Arnoldi and Krylov–Richardson with restart 15 in lines 2 and 3 of Table 6.2: 26.4 s and 6.7 s. Of course, this effect can be less pronounced for larger problems or on faster computers—see

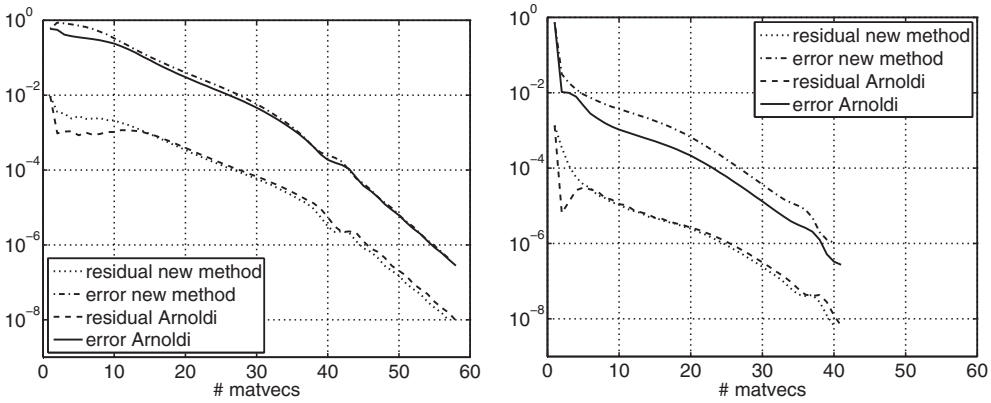

FIG. 6.5. *Convergence plots of the Arnoldi/Lanczos and the new Krylov–Richardson methods for the fourth-order finite volume discretization of the three-dimensional Laplacian with periodic boundary conditions (see section* 6.5*). Left: the starting vector* v *does not satisfy the boundary conditions. Right:* v *is consistent with the boundary conditions.*

the corresponding lines for Arnoldi and Krylov–Richardson with restart 15 on a finer mesh. Second, we have some freedom in choosing the initial vector. (In standard Arnoldi/Lanczos we must always start with v.) This freedom is restricted to the fact that the residual of the initial guess has to have scalar dependence on time. Several variants for choosing the initial vector exist, and we will explore these possibilities in the future.

A significant reduction in total computational work can be achieved when Krylov–Richardson is combined with the shift-and-invert strategy. The gain is then due to the reduction in the number of the inner iterations. (The number of outer iterative steps is approximately the same.) In our limited experience, this is not always the case but typically takes place when, for instance, v and A represent discretizations of a smooth function and a partial differential operator, respectively. Currently, we do not completely understand this behavior. Apparently, the Krylov subspace vectors built in the Krylov–Richardson method constitute more favorable right-hand sides for the inner shift-and-invert solvers to converge. It is rather difficult to analyze this phenomenon, but we will try to do this in the near future.

6.5. Initial vector and Krylov subspace convergence. It is instructive to study the dependence of the Krylov subspace methods on the initial vector v. In particular, if (1.3) stems from an initial boundary value problem and A is a discretized partial differential operator, a faster convergence may take place for v satisfying the boundary conditions of the operator. Note that for the convection-diffusion test problem from the previous section this effect is not pronounced (v did not satisfy boundary conditions), probably due to the jump in the diffusion coefficients. We therefore demonstrate this effect on a simple initial boundary value problem

$$u_t = \Delta u, \qquad u(x, y, z, 0) = u_0(x, y, z), \tag{6.1}$$

posed for $(x, y, z) \in [0, 1]^3$ for the unknown function $u(x, y, z, t)$ obeying periodic boundary conditions. We use a fourth-order finite volume discretization in space from [48] on a regular mesh $40 \times 40 \times 40$ and arrive at an initial value problem (1.3) which we solve for $t = 1000$ by computing $\exp(-tA)v$. In Figure 6.5 convergence of the Krylov–Richardson and Arnoldi/Lanczos methods is illustrated for the starting

vector v corresponding to

$$u_0(x,y,z) = \sin(2\pi x)\sin(2\pi y)\sin(2\pi z) + x(a-x)y(a-y)z(a-z)$$

with $a = 2$ or $a = 1$. In both cases the restart value is set to 100. The second choice $a = 1$ (right plot) agrees with boundary conditions in the sense that u_0 can be periodically extended and leads to a faster convergence. The same effect is observed for the Krylov–Richardson and Arnoldi/Lanczos methods with the shift-and-invert strategy, with a reduction in the number of steps from 12 to 8 or 9. Remarkably, Expokit(100) converges for both choices of v within the same number of steps, 306. Apparently, this is because Expokit splits the given time interval $[0, t_{\text{end}}]$, building a new Krylov subspace for each subinterval.

7. Concluding remarks and an outlook to further research. The proposed residual notion appears to provide a reliable stopping criterion in the iterative methods for computing the matrix exponential. This is confirmed by the numerical tests and the analysis. Furthermore, the residual concept seems to set up a whole framework for a new class of methods for evaluating the matrix exponential. Some basic methods of this class are proposed in this paper. Many new research questions arise. One of them is a comprehensive convergence analysis of the exponential Richardson and the Krylov–Richardson methods. Another interesting research direction is development of other residual-based iterative methods. In particular, one may ask whether the exponential Richardson method (5.4), (5.5) might be used as a preconditioner for the Krylov–Richardson method (5.5). We plan to address this question in future.

Finally, an interesting question is whether the proposed residual notion can be extended to other matrix functions such as trigonometric functions arising in highly oscillatory problems.

Acknowledgments. The first author would like to thank anonymous referees and a number of colleagues, in particular, Michael Saunders, Jan Verwer, and Julien Langou for valuable comments on an earlier version of this paper.

REFERENCES

[1] M. Afanasjew, M. Eiermann, O. G. Ernst, and S. Güttel, *Implementation of a restarted Krylov subspace method for the evaluation of matrix functions*, Linear Algebra Appl., 429 (2008), pp. 2293–2314.

[2] G. Akrivis, C. Makridakis, and R. H. Nochetto, *Galerkin and Runge-Kutta methods: Unified formulation, a posteriori error estimates and nodal superconvergence*, Numer. Math., 118 (2011), pp. 429–456.

[3] M. Benzi and N. Razouk, *Decay bounds and $O(n)$ algorithms for approximating functions of sparse matrices*, Electron. Trans. Numer. Anal., 28 (2007), pp. 16–39.

[4] M. A. Botchev, D. Harutyunyan, and J. J. W. van der Vegt, *The Gautschi time stepping scheme for edge finite element discretizations of the Maxwell equations*, J. Comput. Phys., 216 (2006), pp. 654–686.

[5] E. Celledoni and I. Moret, *A Krylov projection method for systems of ODEs*, Appl. Numer. Math., 24 (1997), pp. 365–378.

[6] C. W. Clenshaw, *Chebyshev Series for Mathematical Functions*, Mathematical Tables 5, Her Majesty's Stationary Office, London, 1962.

[7] K. Dekker and J. G. Verwer, *Stability of Runge-Kutta Methods for Stiff Non-Linear Differential Equations*, North-Holland, Amsterdam, 1984.

[8] V. L. Druskin, A. Greenbaum, and L. A. Knizhnerman, *Using nonorthogonal Lanczos vectors in the computation of matrix functions*, SIAM J. Sci. Comput., 19 (1998), pp. 38–54.

[9] V. L. Druskin and L. A. Knizhnerman, *Two polynomial methods of calculating functions of symmetric matrices*, U.S.S.R. Comput. Math. and Math. Phys., 29 (1989), pp. 112–121.

[10] V. L. Druskin and L. A. Knizhnerman, *Krylov subspace approximations of eigenpairs and matrix functions in exact and computer arithmetic*, Numer. Linear Algebra Appl., 2 (1995), pp. 205–217.
[11] V. L. Druskin and L. A. Knizhnerman, *Extended Krylov subspaces: Approximation of the matrix square root and related functions*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 755–771.
[12] M. Eiermann, O. G. Ernst, and S. Güttel, *Deflated restarting for matrix functions*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 621–641.
[13] W. H. Enright, *Continuous numerical methods for ODEs with defect control*, J. Comput. Appl. Math., 125 (2000), pp. 159–170.
[14] A. Frommer and V. Simoncini, *Matrix functions*, in Model Order Reduction: Theory, Research Aspects and Applications, W. H. A. Schilders, H. A. van der Vorst, and J. Rommes, eds., Springer, New York, 2008, pp. 275–304.
[15] E. Gallopoulos and Y. Saad, *Efficient solution of parabolic equations by Krylov approximation methods*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1236–1264.
[16] F. R. Gantmacher, *The Theory of Matrices*, Vol. 1, AMS Chelsea Publishing, Providence, RI, 1998 (in English).
[17] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.
[18] V. Grimm, *Resolvent Krylov subspace approximation to operator functions*, BIT, 52 (2012), pp. 639–659.
[19] V. Grimm and M. Hochbruck, *Rational approximation to trigonometric operators*, BIT, 48 (2008), pp. 215–229.
[20] S. Güttel, *Rational Krylov Methods for Operator Functions*, Ph.D. thesis, Technischen Universität Bergakademie Freiberg, Freiberg, Germany, 2010.
[21] N. J. Higham, *Functions of Matrices: Theory and Computation*, SIAM, Philadelphia, 2008.
[22] N. J. Higham and A. H. Al-Mohy, *Computing matrix functions*, Acta Numer., 19 (2010), pp. 159–208.
[23] M. Hochbruck and C. Lubich, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 34 (1997), pp. 1911–1925.
[24] M. Hochbruck, C. Lubich, and H. Selhofer, *Exponential integrators for large systems of differential equations*, SIAM J. Sci. Comput., 19 (1998), pp. 1552–1574.
[25] M. Hochbruck and A. Ostermann, *Exponential integrators*, Acta Numer., 19 (2010), pp. 209–286.
[26] J. Janssen and S. Vandewalle, *Multigrid waveform relaxation of spatial finite element meshes: The continuous-time case*, SIAM J. Numer. Anal., 33 (1996), pp. 456–474.
[27] J. Kierzenka and L. F. Shampine, *A BVP solver that controls residual and error*, J. Numer. Anal. Ind. Appl. Math., 3 (2008), pp. 27–41.
[28] L. Knizhnerman and V. Simoncini, *A new investigation of the extended Krylov subspace method for matrix function evaluations*, Numer. Linear Algebra Appl., 17 (2010), pp. 615–638.
[29] L. A. Knizhnerman, *Calculation of functions of unsymmetric matrices using Arnoldi's method*, U.S.S.R. Comput. Math and Math Phys., 31 (1991), pp. 1–9.
[30] L. A. Krukier, *Implicit difference schemes and an iterative method for solving them for a certain class of systems of quasi-linear equations*, Izv. Vyssh. Uchebn. Zaved., Mat., 7 (1979), pp. 41–52 (in Russian).
[31] C. Lubich, *From Quantum to Classical Molecular Dynamics: Reduced Models and Numerical Analysis*, Zurich Lectures in Advanced Mathematics, European Mathematical Society, Zürich, 2008.
[32] C. Lubich and A. Ostermann, *Multi-grid dynamic iteration for parabolic equations*, BIT, 27 (1987), pp. 216–234.
[33] A. Lumsdaine and D. Wu, *Spectra and pseudospectra of waveform relaxation operators*, SIAM J. Sci. Comput., 18 (1997), pp. 286–304.
[34] C. B. Moler and C. F. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Rev., 45 (2003), pp. 3–49.
[35] I. Moret and P. Novati, *RD rational approximations of the matrix exponential*, BIT, 44 (2004), pp. 595–615.
[36] J. Niehoff, *Projektionsverfahren zur Approximation von Matrixfunktionen mit Anwendungen auf die Implementierung exponentieller Integratoren*, Ph.D. thesis, Mathematisch-Naturwissenschaftlichen Fakultät der Heinrich-Heine-Universität Düsseldorf, Düsseldorf, Germany, 2006.
[37] V. S. Ryaben'kii and S. V. Tsynkov, *A Theoretical Introduction to Numerical Analysis*, Chapman & Hall/CRC, Boca Raton, FL, 2007.

[38] Y. Saad, *Analysis of some Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 29 (1992), pp. 209–228.
[39] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, 2003.
[40] L. F. Shampine, *Solving ODEs and DDEs with residual control*, Appl. Numer. Math., 52 (2005), pp. 113–127.
[41] R. B. Sidje, *Expokit. A software package for computing matrix exponentials*, ACM Trans. Math. Software, 24 (1998), pp. 130–156.
[42] H. Tal-Ezer, *Spectral methods in time for parabolic problems*, SIAM J. Numer. Anal., 26 (1989), pp. 1–11.
[43] H. Tal-Ezer, *On restart and error estimation for Krylov approximation of $w = f(A)v$*, SIAM J. Sci. Comput., 29 (2007), pp. 2426–2441.
[44] J. van den Eshof and M. Hochbruck, *Preconditioning Lanczos approximations to the matrix exponential*, SIAM J. Sci. Comput., 27 (2006), pp. 1438–1457.
[45] H. A. van der Vorst, *An iterative solution method for solving $f(A)x = b$, using Krylov subspace information obtained for the symmetric positive definite matrix A*, J. Comput. Appl. Math., 18 (1987), pp. 249–263.
[46] H. A. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, Cambridge, 2003.
[47] J. L. M. van Dorsselaer, J. F. B. M. Kraaijevanger, and M. N. Spijker, *Linear stability analysis in the numerical solution of initial value problems*, Acta Numer., (1993), pp. 199–237.
[48] R. W. C. P. Verstappen and A. E. P. Veldman, *Symmetry-preserving discretization of turbulent flow*, J. Comput. Phys., 187 (2003), pp. 343–368.