NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2013; **20**:557–574
Published online 28 January 2013 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nla.1865

A block Krylov subspace time-exact solution method for linear ordinary differential equation systems‡

M. A. Botchev*,†

Department of Applied Mathematics and MESA+ Institute for Nanotechnology, University of Twente, P.O. Box 217, 7500 AE Enschede, the Netherlands

SUMMARY

We propose a time-exact Krylov-subspace-based method for solving linear ordinary differential equation systems of the form $y' = -Ay + g(t)$ and $y'' = -Ay + g(t)$, where $y(t)$ is the unknown function. The method consists of two stages. The first stage is an accurate piecewise polynomial approximation of the source term $g(t)$, constructed with the help of the truncated singular value decomposition. The second stage is a special residual-based block Krylov subspace method. The accuracy of the method is only restricted by the accuracy of the piecewise polynomial approximation and by the error of the block Krylov process. Because both errors can, in principle, be made arbitrarily small, this yields, at some costs, a time-exact method. Numerical experiments are presented to demonstrate efficiency of the proposed method. Copyright © 2013 John Wiley & Sons, Ltd.

Received 13 January 2012; Revised 30 November 2012; Accepted 9 December 2012

KEY WORDS: block Krylov subspace methods; matrix exponential; exponential time integration; unconditionally stable time integration; exponential residual; truncated SVD; proper orthogonal decomposition

1. INTRODUCTION AND PROBLEM FORMULATION

Consider initial-value problem (IVP)

$$\begin{cases} y' = -Ay + g(t), \\ y(0) = v, \qquad\qquad t \in [0, T], \end{cases} \tag{1}$$

where $y(t)$ is the unknown vector function, $y : \mathbb{R} \to \mathbb{R}^n$, and the matrix $A \in \mathbb{R}^{n\times n}$, vector function $g : \mathbb{R} \to \mathbb{R}^n$, and vector $v \in \mathbb{R}^n$ are given.

Let $\widehat{y}(t) \equiv y(t) - v$ (meaning that $\widehat{y}(t) = y(t) - v$ for all t). Note that the function $\widehat{y}(t)$ satisfies IVP.

$$\begin{cases} \widehat{y}' = -A\widehat{y} + \widehat{g}(t), \\ \widehat{y}(0) = 0, \qquad\qquad t \in [0, T], \end{cases} \tag{2}$$

where $\widehat{g}(t) \equiv g(t) - Av$. We will assume that the IVP (1) is brought to the equivalent form (2) and, for simplicity, we omit the hat sign $\widehat{\cdot}$ in (2).

Problems of type (2) appear in numerous applications, in particular, in the context of numerical solution of partial differential equations (PDEs) by the method of lines. This means that a

*Correspondence to: M. A. Botchev, Department of Applied Mathematics and MESA+ Institute for Nanotechnology, University of Twente, P.O. Box 217, 7500 AE Enschede, the Netherlands.

†E-mail: mbotchev@na-net.ornl.gov

‡In memory of Gene H. Golub, on occasion of the 80th anniversary of his birthday.

discretization of a PDE in space is followed by a time integration of the resulting ordinary differential equation (ODE) system (2). We are thus interested in problem (2) where A is a large, typically sparse matrix.

A need for efficient, fast and accurate time integration of PDEs is caused by, among other, factors nowadays rather usual coupling of different numerical submodels into one large model. In such a setting, a time integration of a PDE system is merely a substep of the whole solution process. This is the case not only for coupled models but also, for instance, for optimization of systems governed by PDEs [1], and for solution of inverse problems (see a numerical example in Section 4.3). In addition, standard PDE solution settings continue to provide extremely challenging cases for numerical computations, too (e.g., [2]).

The time step size in explicit time integration methods can often be unacceptably small, for instance, because of the stiffness of the ODE system or because of a locally refined spatial mesh. In this case, implicit time integration is of interest. Because recently, a lot of research has been carried out on the so-called exponential time integration schemes, see recent comprehensive surveys [3, 4]. These are time integration schemes involving the matrix exponential and related matrix functions. An attractive feature of exponential time integrators is a combination of excellent stability and accuracy properties, with the latter being usually better than in the standard implicit time integrators [3–6]. The interest in exponential time integration is due to the new, challenging applications [7–9] as well as to the recent progress in Krylov subspace techniques to compute actions of matrix functions for large matrices (e.g., [10–20]). Other methods to compute actions of large matrix functions have also been developed, such as ones based on Chebyshev and Taylor polynomials, for example, [12, 20, 21].

In some applications in which explicit time integration is by far inefficient, a gain with implicit or exponential time integration is not guaranteed [22, 23]. A typical situation is then as follows. Assume a maximum time step size allowed by an explicit scheme is τ_{expl}. One would like to use a time step up to, say, $100\tau_{\text{expl}}$ as a further step size increase would yield an accuracy loss. A reasonable gain with an implicit time integrator is then only possible if its costs per time step are well below the costs of the explicit time integrator times 100. However, such a gain is often not guaranteed because a linear solver in the implicit time integrator can be quite expensive. Thus, in these problems, there is only a small niche for the implicit or exponential time integration.

One way to extend this niche, making exponential time integration more attractive, is explored in this work. Our idea is to exploit the property of exponential time integrators to produce *exact* solutions to IVP (2) in certain situations, for example, if $g(t)$ is a vector polynomial. In fact, this paper is inspired by an interesting talk given by Hillel Tal-Ezer in Moscow in the summer of 2011 [24]. Tal-Ezer proposed to approximate $g(t)$ by a Taylor polynomial, so that an exact solution to (2) can be computed with the help of the so-called φ_k matrix functions (e.g., [3] for a definition). This approach, [3–5, 20], has two restrictions. First, a Taylor polynomial approximation is likely to deteriorate for large time intervals. Second, usually actions of $r + 1$ matrix functions (r being the polynomial degree) have to be computed every time step.

Here, we follow another approach, namely, we cast $g(t)$ into the form $g(t) \approx Up(t)$ where U is a tall skinny $n \times m$ matrix, $m \ll n$. This is followed by a single block Krylov subspace projection. Although an accurate approximation $g(t) \approx Up(t)$ is not possible for long time intervals in all cases, there are several classes of problems where our approach can be very efficient. These include problems with spatially localized sources $g(t)$ [25], time dependent boundary conditions (Section 4.2) and problems where $g(t)$ is a slowly varying function.

The paper is organized as follows. The approximation procedure $g(t) \approx Up(t)$ based on truncated singular value decomposition (SVD) is described in Section 2. Section 3 is devoted to the block Krylov subspace method, Section 4 presents numerical experiments, and finally, Section 5 draws conclusions.

2. TRUNCATED SINGULAR VALUE DECOMPOSITION APPROXIMATION

We now describe the first stage of the method, the truncated SVD piecewise polynomial approximation of the source term $g(t)$. Choose s points $0 = t_1 < t_2 < \ldots < t_{s-1} < t_s = T$ on the time interval

DOI: 10.1002/nla

$[0, T]$. The polynomial approximation is based on the truncated SVD of the matrix

$$\tilde{G} = \begin{bmatrix} g(t_1) & g(t_2) & \dots & g(t_s) \end{bmatrix} \in \mathbb{R}^{n\times s},$$

whose columns are samples $g(t_i)$, $i = 1, \dots s$, of the vector function $g(t)$. More precisely, let

$$\tilde{G} = \tilde{U}\tilde{\Sigma}\tilde{V}^T, \quad \tilde{\Sigma} = \mathrm{diag}(\sigma_1, \dots, \sigma_s) \in \mathbb{R}^{s\times s}, \quad \sigma_1 \geqslant \dots \geqslant \sigma_s \geqslant 0, \tag{3}$$

be the thin SVD [26, Section 2.5.4], where the matrices $\tilde{U} \in \mathbb{R}^{n\times s}$ and $\tilde{V} \in \mathbb{R}^{s\times s}$ have orthonormal columns $u_1, \dots, u_s$ and $v_1, \dots, v_s$, respectively. An approximation to $\tilde{G}$ can be obtained by truncating the SVD as

$$\tilde{G} = \tilde{U}\tilde{\Sigma}\tilde{V}^T = \sum_{i=1}^{s} \sigma_i u_i v_i^T \approx \sum_{i=1}^{m} \sigma_i u_i v_i^T = U\Sigma V^T, \quad m \leqslant s, \tag{4}$$

where $\Sigma \in \mathbb{R}^{m\times m} = \mathrm{diag}(\sigma_1, \dots, \sigma_m)$ and the matrices $U \in \mathbb{R}^{n\times m}$ and $V \in \mathbb{R}^{s\times m}$ are formed by the first m columns of $\tilde{U}$ and $\tilde{V}$, respectively. Note that the approximation sign '$\approx$' in (4) should be replaced by the equality sign if $m = s$. Denote the obtained approximate matrix by $G = U\Sigma V^T$. If follows from (4) that the SVD of $\tilde{G} - G$ is readily available as $\sum_{i=m+1}^{s} \sigma_i u_i v_i^T$. Hence, for the two-norm and Frobenius norm of the error $\tilde{G} - G$ holds [26, Section 2.5.3]:

$$\|\tilde{G} - G\|_2 = \sigma_{m+1}, \quad \|\tilde{G} - G\|_F^2 = \sigma_{m+1}^2 + \dots + \sigma_s^2.$$

Looking at SVD identity (3) columnwise, we see that every sample value $g(t_i)$ of the function $g(t)$ can be approximated by a linear combination of the vectors $u_1, \dots, u_m$:

$$\begin{aligned} g(t_i) &= (\sigma_1 v_{i1})u_1 + (\sigma_2 v_{i2})u_2 + \dots + (\sigma_s v_{is})u_s \\ &\approx (\sigma_1 v_{i1})u_1 + (\sigma_2 v_{i2})u_2 + \dots + (\sigma_m v_{im})u_m, \end{aligned}$$

where v_{ij} are the entries of the unitary matrix V. Following the approach of [27], we consider the coefficients of these linear combinations, namely $\sigma_j v_{ij}$, $j = 1, \dots, m$, as values of some unknown functions $f_j(t)$ at t_i. These functions can be easily approximated, at a low cost (typically $m \ll n$) and with a very high accuracy, by a polynomial best fit (as in [27]) or by piecewise interpolation polynomials. The latter approach is followed in this paper. More specifically, we approximate

$$\begin{aligned} g(t) &\approx f_1(t)u_1 + f_2(t)u_2 + \dots + f_m(t)u_m \\ &\approx p_1(t)u_1 + p_2(t)u_2 + \dots + p_m(t)u_m, \quad t \in [0, T], \end{aligned} \tag{5}$$

where piecewise polynomials $p_j(t)$, $j = 1, \dots, m$, are obtained by cubic spline interpolation of the functions $f_j(t)$ in points t_i.

Other strategies for approximating $g(t)$, such as a least squares polynomial fit, are also possible, for example, [27], where the truncated SVD approximation is employed in the context of a nonlinear eigenvalue problem. In our limited experience, a special care with the polynomial fit is required to avoid a high polynomial degree, and hence, possibly a slight accuracy loss for stringent tolerances. For this reason, we choose here to work with piecewise spline interpolation.

Packing the polynomials $p_j(t)$, $j = 1, \dots, m$, in one polynomial vector function $p(t) = (p_1(t), \dots, p_m(t))^T$, we obtain a polynomial approximation

$$g(t) \approx Up(t). \tag{6}$$

There are three sources contributing to the approximation error here. First, the quality of the approximation is influenced by the choice of the sample points $t_1, \dots, t_s$ (typically, it is sensible to use Chebyshev of Leja points for interpolation). Second, the approximation quality depends on the number of terms m in the SVD truncation (4), and finally, on the piecewise polynomial interpolation in (5). The first two sources of the error are by far dominant, but can be easily controlled when

DOI: 10.1002/nla

the approximation is constructed (see [27] and Section 4), thus giving possibility for an adaptive approximation procedure. With (6), the original initial-value problem (2) takes the form

$$\begin{cases} y' = -Ay + Up(t), \\ y(0) = 0, \qquad\qquad t \in [0, T]. \end{cases} \tag{7}$$

We now introduce a block Krylov subspace method to solve this problem.

3. RESIDUAL-BASED BLOCK KRYLOV SUBSPACE METHODS

In this section, we first present our exponential block Krylov method and then extend it for two situations: acceleration by rational Krylov subspaces and solution of the second-order ODE system $y'' = -Ay + Up(t)$. We also discuss implementation of the proposed methods.

3.1. EBK: exponential block Krylov method

Define residual $r_k(t)$ of an approximate solution $y_k(t)$ of (7) as

$$r_k(t) \equiv -Ay_k(t) - y_k'(t) + Up(t). \tag{8}$$

This residual concept (well known in the ODE literature [8, 28–30]) can be used in Krylov subspace methods for the matrix exponential both as a stopping criterion [15] and for restarting [31, 32]. The methods presented here are based on this residual-based restarting approach.

Choosing the initial guess $y_0(t)$ to be a zero vector function, we see that the corresponding initial residual is

$$r_0(t) = Up(t). \tag{9}$$

The approximate solution $y_k(t)$ at Krylov iteration k is then obtained as

$$y_k(t) = y_0(t) + \xi_k(t), \tag{10}$$

where the vector function $\xi_k(t)$ is the Krylov subspace approximate solution of the correction problem

$$\begin{cases} \xi' = -A\xi + r_0(t), \\ \xi(0) = 0, \qquad\qquad t \in [0, T]. \end{cases} \tag{11}$$

Note that if $\xi_k(t)$ solves (11) exactly, then $y_k(t)$ is the sought-after exact solution of (7). It is natural to solve (11) by projecting it onto a block Krylov subspace defined as

$$\mathcal{K}_k(A, U) \equiv \text{span}\left\{U, AU, A^2U, \ldots, A^{k-1}U\right\},$$

with dimension at most $k \cdot m$. An orthonormal basis for this subspace can be generated by the block Arnoldi or Lanczos process (e.g., [33, 34]). The process produces, after k block steps, matrices

$$V_{[k+1]} = \begin{bmatrix} V_1 & V_2 & \ldots & V_{k+1} \end{bmatrix} \in \mathbb{R}^{n\times(k+1)m}, \qquad H_{[k+1,k]} \in \mathbb{R}^{(k+1)m\times km}.$$

Here $V_i \in \mathbb{R}^{n\times m}$, V_1 is taken to be the matrix U produced by the truncated SVD (4) and $V_{[k+1]}$ has orthonormal columns spanning the Krylov subspace, namely,

$$\text{colspan}(V_{[k]}) = \mathcal{K}_k(A, U).$$

The matrix $H_{[k+1,k]}$ is block upper Hessenberg, with $m \times m$ blocks H_{ij}, $i = 1, \ldots, k+1$, $j = 1, \ldots, k$. The matrices $V_{[k+1]}$ and $H_{[k+1,k]}$ satisfy the block Arnoldi (Lanczos) decomposition [33, 34]

$$AV_{[k]} = V_{[k+1]}H_{[k+1,k]} = V_{[k]}H_{[k,k]} + V_{k+1}H_{k+1,k}E_k^T, \tag{12}$$

DOI: 10.1002/nla

where $H_{k+1,k}$ is the only nonzero block in the last $k+1$ block row of $H_{[k+1,k]}$ and $E_k \in \mathbb{R}^{n\times k}$ is formed by the last m columns of the $km \times km$ identity matrix.

Once the Krylov basis matrix $V_{[k]}$ is built, the Krylov subspace solution $\xi_k(t)$ of (11) can be computed as

$$\xi_k(t) = V_{[k]}u(t), \tag{13}$$

where $u(t)$ solves the projected IVP

$$\begin{cases} u'(t) = -H_{[k,k]}u(t) + E_1 p(t), \\ u(0) = 0, \qquad\qquad t \in [0,T]. \end{cases} \tag{14}$$

Here, $E_1 \in \mathbb{R}^{km\times m}$ is formed by the first m columns of the $km \times km$ identity matrix. Note that

$$E_1 p(t) = V_{[k]}^T r_0(t) = V_{[k]}^T V_1 p(t).$$

We will refer to the just described scheme as EBK, exponential block Krylov method.

Theorem 1
Let $y_k(t)$ be the solution obtained after carrying out k steps of the EBK method. Then, the exponential residual $r_k(t)$ of $y_k(t)$, defined by (8), is given by

$$r_k(t) = -V_{k+1}H_{k+1,k}E_k^T u(t), \tag{15}$$

where V_{k+1}, $H_{k+1,k}$, E_k are introduced in (12) and $u(t)$ is the solution of the projected IVP (14). Furthermore, the error $e_k(t) \equiv y(t) - y_k(t)$, with $y(t)$ being the exact solution of (7), is given by

$$e_k(t) = -\int_0^t \exp(-(t-s)A)V_{k+1}H_{k+1,k}E_k^T u(s)\mathrm{d}s. \tag{16}$$

Proof
Using (9), (12) and (14), we see that for the residual $r_k(t)$, the solution $y_k(t)$ holds

$$\begin{aligned} r_k(t) &= -Ay_k - y_k' + Up(t) = -Ay_0 - y_0' - AV_{[k]}u(t) - V_{[k]}u'(t) + Up(t) = \\ &= r_0(t) - AV_{[k]}u(t) - V_{[k]}u'(t) = \\ &= r_0(t) - \left(V_{[k]}H_{[k,k]} + V_{k+1}H_{k+1,k}E_k^T\right)u(t) - V_{[k]}u'(t) = \\ &= r_0(t) - V_{[k]}\left(H_{[k,k]}u(t) + u'(t)\right) - V_{k+1}H_{k+1,k}E_k^T u(t) = \\ &= r_0(t) - V_{[k]}E_1 p(t) - V_{k+1}H_{k+1,k}E_k^T u(t) = -V_{k+1}H_{k+1,k}E_k^T u(t). \end{aligned}$$

To obtain the expression for the error, we note that

$$e_k'(t) = -Ae_k(t) + r_k(t), \qquad e_k(0) = 0, \tag{17}$$

that is, the exponential residual can be seen as a backward error. Applying a variation of constants formula to the last system, we have

$$e_k(t) = \int_0^t \exp(-(t-s)A)r_k(s)\mathrm{d}s,$$

which yields (16), once the just obtained expression for $r_k(t)$ is substituted. □

Similar expressions for the exponential residual are obtained in [15, 31, 32] for non-block Krylov subspace methods. There are two important messages relation (15) provides. First, the residual can be computed efficiently during the iteration process because the matrices V_{k+1} and $H_{k+1,k}$ are readily available in the Arnoldi or Lanczos process. Second, the residual after k block steps has the same form as the initial residual (9), namely it is a matrix of m orthonormal columns times a time dependent vector function. This allows for a restart in the block Krylov method: set $y_0(t) := y_k(t)$, then

DOI: 10.1002/nla

relation (9) holds with $U := V_{k+1}$ and $p(t) := -H_{k+1,k} E_k^T u(t)$. The correction procedure with k block Krylov iterations can then be repeated, which results in a restarted block Krylov subspace method for solving (7).

The explicit expression for the error, as given by relation (16) in Theorem 1, shows that the residual is a controller of the error and can be used to obtain different error estimates similar to [15, 31].

3.2. EBK/SaI: an extension to rational Krylov subspaces

Being essentially a matrix exponential method, the proposed block Krylov method can be generalized to rational Krylov subspaces. We now show how this can be carried out for the shift-and-invert (SaI) Krylov subspace methods for the matrix exponential [16, 35]. SaI Krylov subspace methods for the matrix exponential converge faster because they emphasize the important small in magnitude eigenvalues in the built up Krylov subspace. More specifically, the Krylov subspace is constructed with respect to the matrix $(I + \gamma A)^{-1}$, with $\gamma > 0$ being a parameter. This means that for every Arnoldi or Lanczos step, a linear system with the matrix $I + \gamma A$ has to be solved. For SaI methods, the block Arnoldi decomposition (12) transforms into

$$(I + \gamma A)^{-1} V_{[k]} = V_{[k+1]} \tilde{H}_{[k+1,k]} = V_{[k]} \tilde{H}_{[k,k]} + V_{k+1} \tilde{H}_{k+1,k} E_k^T, \tag{18}$$

where the notation of (12) is kept and we added the $\tilde{\cdot}$ sign to the matrix $\tilde{H}_{[k,k]}$ to emphasize that it is a projection of the shifted and inverted A (and not anymore of A). A projection $H_{[k,k]}$ of A can be recovered from $\tilde{H}_{[k,k]}$ as

$$H_{[k,k]} = \frac{1}{\gamma} \left(\tilde{H}_{[k,k]}^{-1} - I \right), \tag{19}$$

where I is the $km \times km$ identity matrix. In practice it is convenient to use relation (18) rewritten as [16, formula (4.1)]

$$A V_{[k]} = V_{[k]} H_{[k,k]} - R_k, \qquad R_k = \frac{1}{\gamma} (I + \gamma A) V_{k+1} \tilde{H}_{k+1,k} E_k^T \tilde{H}_{[k,k]}^{-1}. \tag{20}$$

Proceeding as for the EBK scheme (see the previous section), we again choose the initial guess to be zero function, $y_0(t) = 0$, and start the block Arnoldi process with $V_1 = U$. Forming the Galerkin projection of IVP (7) onto the block Krylov subspace (spanned by the columns of $V_{[k]}$), we arrive at the SaI projected IVP, which appears to be identical to the projected IVP (14) for the conventional EBK scheme. The solution $y_k(t)$ after k block steps is computed according to (10) and (13). The essential difference is that the Krylov subspace is now built up for the transformed, shifted-and-inverted A, and hence, the matrix $H_{[k,k]e}$ in (14) is now defined by (19).

We refer to this scheme as EBK/SaI. The following result holds for the residual and error of EBK/SaI.

Theorem 2
Let $y_k(t)$ be the solution obtained after carrying out k steps of the EBK method. Then, the exponential residual $r_k(t)$ of $y_k(t)$, defined by (8), is given by

$$r_k(t) = R_k u(t) = \frac{1}{\gamma} (I + \gamma A) V_{k+1} \tilde{H}_{k+1,k} E_k^T \tilde{H}_{[k,k]}^{-1} u(t), \tag{21}$$

where R_k, V_{k+1}, $H_{k+1,k}$, E_k are from (18) and (20) and $u(t)$ is the solution of the projected IVP (14). Furthermore, the error $e_k(t) \equiv y(t) - y_k(t)$, with $y(t)$ being the exact solution of (7), is given by

$$e_k(t) = \frac{1}{\gamma} (I + \gamma A) \int_0^t \exp(-(t-s)A) V_{k+1} \tilde{H}_{k+1,k} E_k^T \tilde{H}_{[k,k]}^{-1} u(s) \mathrm{d}s. \tag{22}$$

DOI: 10.1002/nla

Proof
We first prove the expression for the residual.

$$\begin{aligned} r_k(t) &= r_0(t) - AV_{[k]}u(t) - V_{[k]}u'(t) = \\ &= r_0(t) - [V_{[k]}H_{[k,k]} - R_k]u(t) - V_{[k]}u'(t) = \\ &= r_0(t) - V_{[k]}\left[H_{[k,k]}u(t) + u'(t)\right] + R_k u(t) = \\ &= r_0(t) - V_{[k]}E_1 p(t) + R_k u(t) = r_0(t) - r_0(t) + R_k u(t) = \\ &= R_k u(t) = \frac{1}{\gamma}(I + \gamma A)V_{k+1}\tilde{H}_{k+1,k}E_k^T \tilde{H}_{[k,k]}^{-1}u(t), \end{aligned}$$

where we use (20) and (14). To obtain the expression for the error, we proceed similarly to the proof of Theorem 1, that is, we consider (17) and apply the variation of constants formula. □

Again, just as for the relation (15), the message provided by (21) is twofold. First, the residual can easily be computed in the block iterative process. Second, the residual can be cast in the form $Up(t)$ by computing the thin QR factorization of $R_k =: QR$ and setting $U := Q$, $p(t) = Ru(t)$. Thus, the residual-based restarting strategy is still possible with the SaI technique. Furthermore, the expression for the error (22), together with (17), justifies the use of the residual for error control and can be used to obtain further insight in the error behavior [16].

3.3. EBK2: a block method for the second-order ODE systems

The EBK method can be easily adapted to the second-order IVP

$$\begin{cases} y'' = -Ay + g(t), \\ y(0) = v,\ y'(0) = w, \qquad\qquad t \in [0, T], \end{cases} \tag{23}$$

where notation is the same as in (1) and $w \in \mathbb{R}^n$ is given. We first transform the problem to an equivalent form with homogeneous initial values. This can be carried out by introducing a new variable $\widehat{y}(t) \equiv y(t) - v - tw$, similarly to the transformation from (1) to (2). Again, for simplicity of the presentation, we omit the $\tilde{\ }$ in the transformed IVP. The approximation procedure $g(t) \approx Up(t)$ of the (transformed) $g(t)$ then leads to a problem

$$\begin{cases} y'' = -Ay + Up(t), \\ y(0) = 0,\ y'(0) = 0, \qquad\qquad t \in [0, T]. \end{cases} \tag{24}$$

We now choose the initial guess $y_0(t)$ to be zero function and form an IVP for the update $\xi_k(t)$, $y_k(t) = y_0(t) + \xi_k(t)$ (cf. (11)). Galerkin projection of this IVP onto the block Krylov subspace spanned by the columns of $V_{[k]}$ results in a projected IVP

$$\begin{cases} u''(t) = -H_{[k,k]}u(t) + E_1 p(t), \\ u(0) = u'(0) = 0, \qquad\qquad t \in [0, T]. \end{cases} \tag{25}$$

The correction $\xi_k(t)$ is then defined as $\xi_k(t) = V_{[k]}u(t)$. For this method, which we call EBK2 for the second-order ODE systems, the following result holds.

Theorem 3
Let $y_k(t)$ be the solution obtained after carrying out k steps of the EBK2 method. Then the exponential residual $r_k(t)$ of $y_k(t)$, defined by $r_k(t) \equiv -Ay_k - y_k'' + Up(t)$, is given by

$$r_k(t) = -V_{k+1}H_{k+1,k}E_k^T u(t), \tag{26}$$

where V_{k+1}, $H_{k+1,k}$, E_k are defined in (18) and $u(t)$ is the solution of the projected IVP (25). Furthermore, the error $e_k(t) \equiv y(t) - y_k(t)$, with $y(t)$ being the exact solution of (24), is given by

$$e_k(t) = -A^{-1/2}\int_0^t \sin((t-s)A^{1/2})V_{k+1}H_{k+1,k}E_k^T u(s)\mathrm{d}s. \tag{27}$$

DOI: 10.1002/nla

Proof
Proceeding as in (15) and using (12) and (25), we obtain:

$$\begin{aligned} r_k(t) &= -Ay_k - y_k'' + Up(t) = -Ay_0 - y_0'' - AV_{[k]}u(t) - V_{[k]}u''(t) + Up(t) = \\ &= r_0(t) - AV_{[k]}u(t) - V_{[k]}u''(t) = \\ &= r_0(t) - \left(V_{[k]}H_{[k,k]} + V_{k+1}H_{k+1,k}E_k^T\right)u(t) - V_{[k]}u''(t) = \\ &= r_0(t) - V_{[k]}\left(H_{[k,k]}u(t) + u''(t)\right) - V_{k+1}H_{k+1,k}E_k^T u(t) = \\ &= r_0(t) - V_{[k]}E_1 p(t) - V_{k+1}H_{k+1,k}E_k^T u(t) = -V_{k+1}H_{k+1,k}E_k^T u(t). \end{aligned} \tag{28}$$

We now derive relation (27). Subtracting the perturbed ODE system $y_k'' = -Ay_k + Up(t) - r_k(t)$ from the ODE system in (24), we see that the error $e_k(t)$ satisfies

$$e_k'' = -Ae_k + r_k(t), \quad e_k(0) = 0, \quad e_k'(0) = 0,$$

which means that the residual $r_k(t)$ can be seen as a backward error for $y_k(t)$. A variation of constants formula (e.g., [36]) yields for the last IVP an expression

$$e_k(t) = \int_0^t A^{-1/2}\sin((t-s)A^{1/2})r_k(s)\mathrm{d}s,$$

which, together with (28) leads to (27). □

Theorem 3 provides an easy-to-compute expression for the residual and a way to restart the block Krylov process: the residual after k steps is of the same form as the initial residual, namely $Up(t)$, $U \in \mathbb{R}^{n\times m}$. In addition, the explicit expression for the error given by Theorem 3 reveals the residual as a legitimate error controller. The estimate can also be used for a more elaborated error analysis, for example, [36, 37].

3.4. Implementation of the exponential block Krylov methods

We now sketch an algorithm for the EBK method presented in Section 3.1. The adjustments for the EBK/SaI and EBK2 schemes are straightforward.

1. Switch to homogeneous initial conditions (see (1),(2)).
   Approximate $g(t) \approx Up(t)$. Set $y_0(t) := 0$.
2. Set $r_0(t) := Up(t)$. Stop if $\|r_0(t)\|$ is small enough.
   Otherwise set $V_1 := U$.
3. main Krylov subspace loop:
   for $k = 1, \ldots,$ `restart`
   (a) Perform step k of the block Arnoldi/Lanczos process (12):
   compute V_{k+1} and the block column k of $H_{[k+1,k]}$.
   (b) Find solution $u(t)$ of the projected IVP (14) approximately,
   compute residual with (15): $r_k(t) := -V_{k+1}H_{k+1,k}E_k^T u(t)$.
   (c) if $k =$ `restart` or $\|r_k(t)\|$ is small enough
   solve the projected IVP (14) accurately,
   update solution $y_k(t) := y_0(t) + V_{[k]}u(t)$
   if $\|r_k(t)\|$ is small enough
   stop
   endif
   if $k =$ `restart`
   $y_0(t) := y_k(t)$, $U := V_{k+1}$, $p(t) := -H_{k+1,k}E_k^T u(t)$
   return to step 2.
   endif
   endfor

DOI: 10.1002/nla

Several remarks are in place. In all the experiments presented in the paper, the samples $g(t_i)$ were computed at the Chebyshev points in $[0, T]$. If desired, the Leja points can also be used. As mentioned earlier, the number of sample points s and the number of SVD terms m can be easily chosen adaptively by checking the SVD error *a posteriori*, after the approximation stage $g(t) \approx Up(t)$ is carried out. Recomputing the SVD approximation for adjusted values of s and m does not lead to a loss in efficiency, because the approximation stage is hardly visible in the total costs and CPU time.

It is important to stop only if $\|r_k(t)\|$ is small enough for *several* values $t \in [0, T]$, checking only $\|r_k(T)\|$ may not be sufficient. For example, $\|r_0(T)\|$ turns out to be exactly zero in the test of Section 4.2. Ideally, one should check the $L_2[0, T]$ integral norm of $\|r_k(t)\|$. Furthermore, note that the projected problem is not solved to a full accuracy most of the time. This is only necessary when the solution is updated because of a restart or satisfied stopping criterion. In EBK and EBK/SaI, the projected IVP is solved with the `ode15s` MATLAB ODE solver. For the numerical experiments presented here, when solving the projected IVP approximately, we set the absolute tolerance `atol` in `ode15s` to one percent of the current residual norm:

```
atol = resid_norm/100;
atol = max([1e-10, abs_tol]);
atol = min([1e-03, abs_tol]);
```

When solving the problem accurately, the `atol` is set `1e-12`. The relative tolerance is in both cases set to `min([atol*1e3,1e-2])`.

In EBK2, using the stiff MATLAB solver `ode15s` for the nonstiff second-order projected IVP (25) is not necessary and can be inefficient. The projected problem is then transformed to an equivalent first-order IVP and solved by the explicit `ode45` MATLAB ODE solver. The tolerances for the projected solver are chosen in the same way as in EBK.

Note that one could also use other (than the general purpose ODE solvers) for solving the projected IVPs. For instance, the exact φ_k-based exponential integrator is employed in [38] provided that the function $E_1 p(t)$ in (14) can be well approximated by a best fit polynomial. To compute the φ_k functions, for example, the EXPINT package can be used [39]. In our case, the function $E_1 p(t)$ is a piecewise polynomial, and we would thus need to apply the exact exponential integrator piecewise. In addition, we would need to choose a proper way to store the computed projected solution $u(t)$ for $t \in [0, T]$, which would probably have to be carried out depending on the chosen accuracy tolerance. All together, this would not be an easy programming task. With the standard ODE solvers in MATLAB, the representation of $u(t)$ for $t \in [0, T]$ is given automatically, on the basis of the given tolerance, and the piecewise polynomial representation of $E_1 p(t)$ can easily be adapted to it at every restart.

4. NUMERICAL EXPERIMENTS

In this section, we present several numerical experiments with the proposed block Krylov subspace method.

4.1. Test 1: convection–diffusion problem

In the first test, we solve IVP (1) where the matrix A stems from the standard five point finite-difference discretization of the two-dimensional convection–diffusion operator

$$L[u] = -(D_1 u_x)_x - (D_2 u_y)_y + Pe(v_1 u_x + v_2 u_y),$$

$$D_1(x, y) = \begin{cases} 10^3 & (x, y) \in [0.25, 0.75]^2, \\ 1 & \text{otherwise}, \end{cases} \qquad D_2(x, y) = \frac{1}{2} D_1(x, y),$$

$$v_1(x, y) = x + y, \qquad v_2(x, y) = x - y,$$

where Pe is the Peclet number. The spatial domain is $[0, 1] \times [0, 1]$ and the L is set to satisfy homogeneous Dirichlet boundary conditions. Before discretization, the convection terms are rewritten as

DOI: 10.1002/nla

$\frac{1}{2}(v_1 u_x + v_2 u_y) + \frac{1}{2}((v_1 u)_x + (v_2 u)_y)$, which is possible because the velocity field is divergence free. The convection terms yield an exactly skew-symmetric matrix when discretized in this form [40]. Building up the matrix A, we scale all its entries with h_1^2, the grid size in x-direction. This means that, if the grid sizes in x and y-directions are the same ($h_1 = h_2$), the diffusion contributions to A are grid-independent and the convection contributions scale with the grid-size as $1/h_1$. The source function $g(t)$ is chosen such that IVP (1) has exact solution $y(t) = \cos(2\pi t)v$ with $v \in \mathbb{R}^n$ taken to be a vector of equal entries and a unit Euclidean norm. The IVP is solved for $t \in [0, T]$, $T = 1.5$.

We emphasize that this first test problem presumably presents an easy case for the EBK scheme. Indeed, note that $g(t)$ is, for any t, an element of a two-dimensional subspace span$\{v, Av\}$, so that there are only two nonzero singular values in the SVD (3) of the matrix containing the samples of $g(t)$. Therefore, we take $m = 2$ in the truncated SVD (4). As an illustration, in Figure 1, we plot the error of the SVD approximation for this test problem. The error is computed as $\|Up(t_j) - g(t_j)\|/g(t_j)\|$ for $10s$ points t_j evenly distributed in $[0, T]$.

It is natural to compare performance of the proposed block Krylov method against another exponential time integration scheme based on Krylov subspace approximations. For this purpose we take the well-known exponentially fitted Euler scheme (e.g., [41]). Because for non-autonomous problems, the scheme has the first-order accuracy only, we apply the scheme with global extrapolation. Although the resulting exponential time integration scheme is second-order accurate, it involves the evaluation of a matrix function $\varphi_1(-\tau A)$ only, with

$$\varphi_1(x) = \frac{\exp(x) - 1}{x},$$

A being the matrix of the IVP (1) and $\tau > 0$ the time step. Doing so, we avoid, for efficiency reasons, using schemes with more than one evaluations of the matrix functions φ_k. (For a definition of φ_k, see, e.g., [3]). Furthermore, having only the φ_1 function allows to employ the EXPOKIT package [42] for computing actions $\varphi_1(-\tau A)$ on a vector. Using EXPOKIT within the globally extrapolated exponentially fitted Euler scheme seems to result in a very competitive exponential time integrator, which we denote by EE2/EXPOKIT. This is because EXPOKIT, at least in our limited experience, is quite robust and efficient for not too large $\tau\|A\|$. For comparisons of modern Krylov subspace matrix exponential solvers against EXPOKIT, for example, [38]. Throughout this paper, we use EXPOKIT with the default value of the Krylov subspace dimension (which is 30) and varied tolerance as indicated in the experiments. For alternative exponential time integrators suitable for large scale problems, see recent surveys [3, 4] and [5].

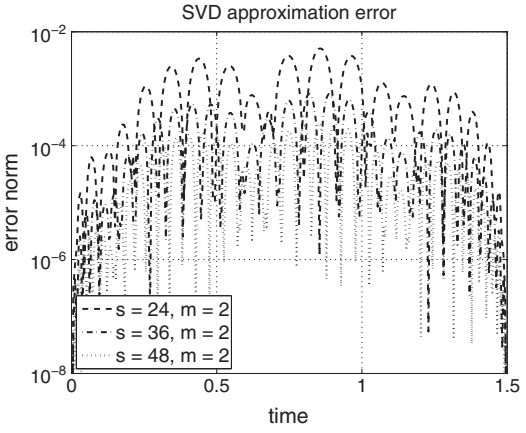

Figure 1. Relative error norm of the singular value decomposition approximation error for test problem 1. Mesh 102×102 is used. The time averaged errors on these plots are: `2.5e-04` for $s = 24$ (dashed line), `4.0e-05` for $s = 36$ (dash-dotted line), `1.2e-05` for $s = 48$ (dotted line).

DOI: 10.1002/nla

Table I. Numerical results for test problem 1. Please note that the CPU time is measured in MATLAB and thus gives only an indication of the actual performance.

Scheme: EBK(s, m) or EE2/EXPOKIT(τ)	CPU time, s	Total # matvecs	Error
	Mesh 102×102, $Pe = 10^3$		
EBK(24, 2)	2.7	196	9.2e−05
EBK(36, 2)	2.5	152	1.6e−05
EBK(48, 2)	2.2	112	4.7e−06
EBK(48, 2), SaI	1.0	"10"	4.7e−06
EE2/EXPOKIT($\tau = 1.5/100$)	20	22,400	7.3e−04
EE2/EXPOKIT($\tau = 1.5/200$)	35	38,400	1.8e−04
EE2/EXPOKIT($\tau = 1.5/400$)	70	76,800	4.6e−05
	Mesh 402×402, $Pe = 10^4$		
EBK(24, 2)	30	328	9.2e−05
EBK(36, 2)	26	272	1.6e−05
EBK(48, 2)	23	212	4.7e−06
EBK(48, 2), SaI	26	"12"	4.7e−06
EE2/EXPOKIT($\tau = 1.5/100$)	671	22,400	7.4e−04
EE2/EXPOKIT($\tau = 1.5/200$)	1,096	38,400	1.8e−04
EE2/EXPOKIT($\tau = 1.5/400$)	2,086	76,800	4.6e−05

In the test runs, the tolerance is set to `toler` $= 10^{-8}$. For EBK, this means that the scheme stops as soon as the exponential residual norm reaches below `toler`. The block Arnoldi process is restarted every 20 block steps (the total Krylov dimension is thus at most $20m = 40$). For EE2/EXPOKIT the chosen tolerance `toler` is given as an input parameter to EXPOKIT every time an action of the φ_1 function has to be computed. The results of the comparison of EBK and EE2/EXPOKIT are presented in Table I. The EBK scheme is a clear winner here in terms of both CPU time, number of matrix–vector multiplications (matvecs) and achieved accuracy. The error is measured as the relative error norm with respect to the known exact solution. As we see, the more accurate the SVD approximation is, the smaller the error becomes and the fewer iterations are needed by EBK to converge. Thus, apparently, an accurate SVD approximation has a regularization effect of the Krylov subspace convergence. Finally, as we see from Table I, the efficiency of the EBK/SaI scheme depends on how fast the linear systems with $(I + \gamma A)$ can be solved. In this two-dimensional test, the systems are solved by the backslash operator, which appears to be faster than computing a sparse LU factorization once and using it every time the system has to be solved. Using an iterative solver is definitely possible and advisable. In fact, an efficient strategy for stopping these inner iterations is proposed in [16]. However, because we have a block Krylov method, the inner iterative solver should accept multiple right hand sides. A natural choice for a linear iterative solver accepting multiple right hand sides would be a block Krylov subspace solver. We are not aware of such solvers available for MATLAB but plan to implement and test them within the EBK framework in the future.

4.2. Test 2: wave equation

Our second test problem is an IVP for a spatially discretized wave equation with time-dependent boundary conditions:

$$\begin{cases} u_{tt} = \Delta u, \quad (x, y) \in \Omega \equiv (0, 1)^2, \\ u(x, y, 0) = 0, \quad u_t(x, y, 0) = 0, \\ u\big|_{x=0} = u_{\mathrm{b}}(y, t), \quad u\big|_{\partial\Omega \setminus \{x=0\}} = 0, \end{cases} \tag{29}$$

$$u_{\mathrm{b}}(y, t) = \sin(2\pi t) e^{-100\left(y - \frac{1}{2}\left(1 + \frac{1}{4}\sin(2\pi t)\right)\right)^2}.$$

A time snapshot of the solution for this problem is plotted in Figure 2. The standard five-point discretization of (29) yields a second-order IVP (23) where $g(t)$ contains the contributions from the

DOI: 10.1002/nla

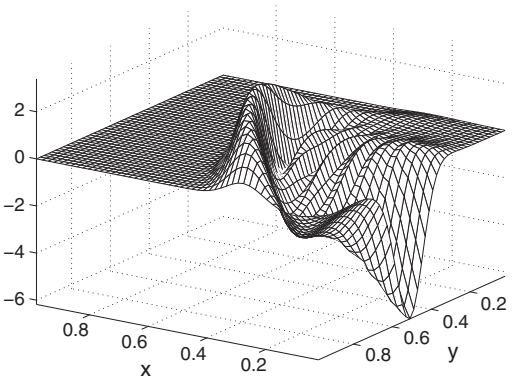

Figure 2. Solution of test problem 2 at $t = 0.5$.

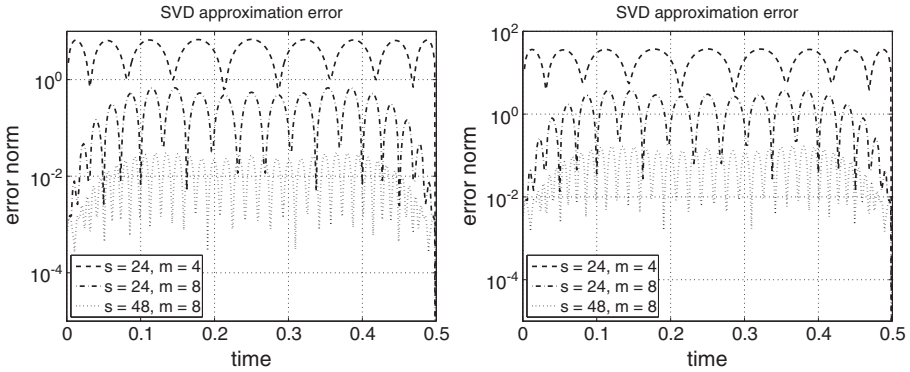

Figure 3. Relative error norm of the singular value decomposition approximation error for test problem 2 on meshes (left) 51×51 and (right) 101×101. Note different scales of the y-axes.

time dependent boundary conditions. Note that the boundary condition function u_{b}, and hence, $g(t)$ can not be represented as a time-dependent scalar function times a constant vector. Thus, there is no reason to assume that the time samples $g(t_i)$, $i = 1, \ldots s$, should span a subspace of a small dimension. The only reason to expect the dimension of this subspace to be restricted is the fact that the function is defined on the domain boundary only. However, as we will see in the experiments, the number of the truncated SVD terms necessary to parametrize the function turns out to be much smaller than the number of boundary degrees of freedom in the spatial discretization of (29). As an illustration, the SVD approximation error, computed in the same way as in the test problem 1, is plotted in Figure 3.

With this test problem, we compare the proposed EBK2 scheme (presented in Section 3.3) against the EE2/EXPOKIT scheme. The results are obtained for the final time $T = 0.5$ and $\texttt{toler} = 10^{-6}$. The EE2/EXPOKIT scheme is applied to an equivalent first-order IVP. This does not give any advantage to the EBK2 scheme: in fact, similar results can be obtained with the EBK scheme applied to the equivalent first-order IVP. However, for EBK, a special care has to be taken to avoid real Ritz values, which are spurious since the eigenvalues of the equivalent first-order system matrix

$$\begin{bmatrix} 0 & I \\ -A & 0 \end{bmatrix}$$

are purely imaginary. These spurious Ritz values can be both negative or positive and thus harmful for the solution accuracy of the projected IVP. The EBK2 does not have this problem because it works directly with the matrix A. The EXPOKIT, which relies on the Padé approximations of the projected matrix functions, does not seem to suffer from working with the first-order system matrix either.

DOI: 10.1002/nla

Table II. Numerical results for test problem 2. Please note that the CPU time is measured in MATLAB and thus gives only an indication of the actual performance.

Scheme: EBK2(s, m) or EE2/EXPOKIT(τ)	CPU time, s	Total # matvecs	Error
	Mesh 51×51		
EBK2(24, 4)	5.5	184	1.4e−03
EBK2(24, 8)	6.5	368	6.5e−05
EBK2(48, 8)	6.8	368	2.9e−06
EBK2(48, 10)	8.3	450	2.9e−06
EE2/EXPOKIT($\tau = 0.5/100$)	9.0	19,296	2.9e−04
EE2/EXPOKIT($\tau = 0.5/200$)	18	38,496	7.3e−05
EE2/EXPOKIT($\tau = 0.5/400$)	35	76,800	1.8e−05
	Mesh 101×101		
EBK2(24, 4)	21	400	1.5e−03
EBK2(24, 8)	24	800	8.2e−05
EBK2(48, 8)	27	800	2.9e−06
EBK2(48, 10)	31	1000	2.9e−06
EE2/EXPOKIT($\tau = 0.5/100$)	47	19,200	3.2e−04
EE2/EXPOKIT($\tau = 0.5/200$)	95	38,400	8.0e−05
EE2/EXPOKIT($\tau = 0.5/400$)	192	76,800	2.0e−05

The results of the test runs of EBK2 and EE2/EXPOKIT are given in Table II. In the tests, EBK2 is restarted every 20 Krylov steps. Thus, the total Krylov dimension size is at most $20 \times m$, that is 200 in this test. The error given in the table is measured as the relative error norm with respect to a reference solution obtained with MATLAB ODE solver `ode15s` run with stringent tolerances. The reference thus contains essentially the same spatial error, and the reported error is merely the time integration error. We see that the EBK2 solver clearly outperforms the exponential integrator EE2/EXPOKIT in terms of CPU time, total number of matvecs and obtained accuracy.

A nice property we observe in the EBK2 scheme is that its convergence does not seem to strongly depend on how often the scheme is restarted. For an illustration, see Figure 4 where the convergence plots are given for restart values 10 and 20, for the 51×51 spatial mesh. This nice feature of the scheme can be apparently explained by the block structure of the scheme: the scheme is restarted with the last block of Krylov vectors rather than with a single vector, as in conventional Krylov subspace methods. This effect is similar to the so-called thick restarts used in Krylov subspace matrix function computations [19].

4.3. Test 3: 3D Maxwell's equations

This test problem comes from electromagnetic imaging in gas-and-oil industry [25, 43] and results from a spatial discretization of Maxwell's equations posed in a three-dimensional (3D) spatial domain. In a typical setting, a probe containing several electric coils is put under the ground [25, 43]. Electric current, switched on for a short time in some of the coils, creates an electromagnetic impulse, whereas the other coils are used to measure the resulting electromagnetic field. This setting, which can be seen as solving the *direct* problem, has to be simulated many times for different known structure of the domain. The goal is then to solve the *inverse* problem, that is to obtain an impression on the structure of the underground domain of interest, based on the measurements of the field in coils. A similar approach can be used for the fault detection in the underground oil-and-gas winning hardware [43].

Maxwell's equations are posed as

$$\begin{aligned} \mu \partial_t \boldsymbol{H} &= -\nabla \times \boldsymbol{E}, \\ \varepsilon \partial_t \boldsymbol{E} &= \nabla \times \boldsymbol{H} - \sigma \boldsymbol{E} + \boldsymbol{J}, \end{aligned} \tag{30}$$

DOI: 10.1002/nla

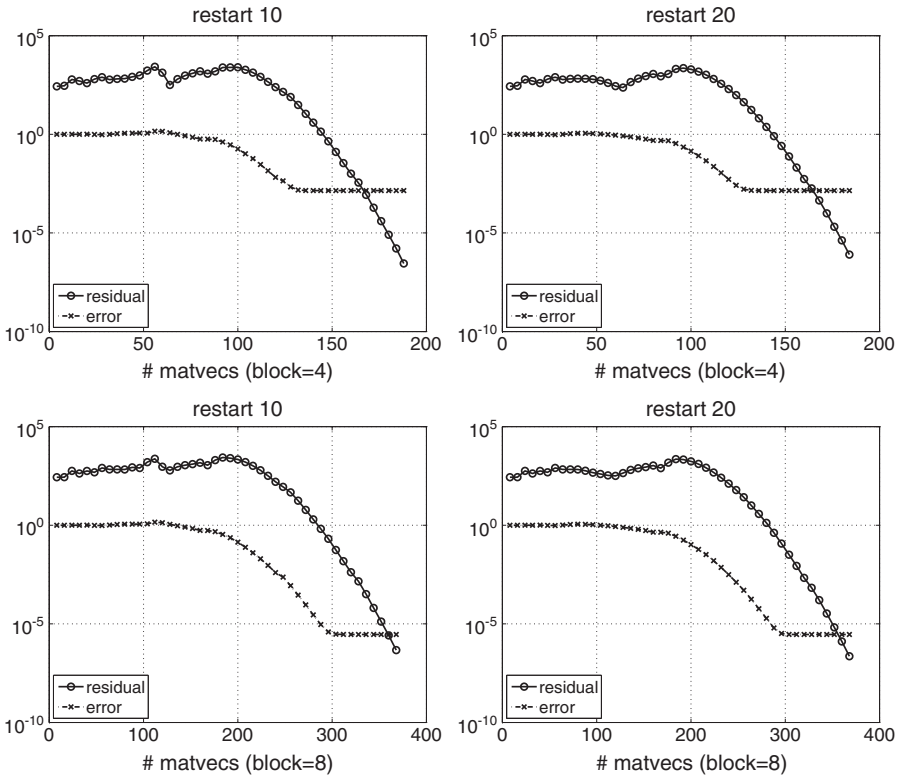

Figure 4. Convergence plots of the EBK2 scheme restarted every 10 or 20 steps (left/right plots), with either $s = 24$, $m = 4$ or $s = 48$, $m = 8$ (top/bottom plots).

where $\boldsymbol{H} = \boldsymbol{H}(x, y, z, t)$ and $\boldsymbol{E} = \boldsymbol{E}(x, y, z, t)$ are unknown vector functions of, respectively, the magnetic and electric fields, $\mu = \mu_0$ is the magnetic permeability, $\varepsilon = \varepsilon_0$ is the electric permittivity[§], $\sigma = \sigma(x, y, z)$ is electric conduction and known $\boldsymbol{J} = \boldsymbol{J}(x, y, z, t)$ is the electric current. The equations are posed in a cubical physical domain $\Omega = [-20, 20]^3$ (the size is given in meters), with the far field boundary conditions (homogeneous Dirichlet). The initial conditions are zero for both fields.

The conductivity σ is piecewise constant and defined as

$$\sigma = \begin{cases} 0.1 & \text{S/m}, \quad x \leqslant 10, \\ 0.001 & \text{S/m}, \quad x > 10. \end{cases} \tag{31}$$

A coil of a square shape is present in the larger subdomain ($x \leqslant 10$) and connects four points with coordinates (x, y, z) being $(-2, -2, 0)$, $(-2, 2, 0)$, $(2, 2, 0)$ and $(2, -2, 0)$. The current source function $\boldsymbol{J}$ (A) is zero everywhere in the domain except on the coil and depends on time (s) as follows.

$$\boldsymbol{J}(x, y, z, t) = \begin{cases} \text{linear growth from 0 to 1}, & 0 \leqslant t \leqslant 10^{-6}, \\ 1 + \sin\left(\frac{\pi}{2}\frac{t-10^{-6}}{10^{-4}}\right), & 10^{-6} < t \leqslant 1.01 \times 10^{-4} \\ \text{linear decay to 0}, & 1.01 \times 10^{-4} < t \leqslant 1.02 \times 10^{-4} \\ 0, & t > 1.02 \times 10^{-4}. \end{cases}$$

A usual dimensionless scaling of the problem and its spatial discretization by the Yee finite differences leads to an IVP of the form (7), where $y(t)$ contains the components of both fields. We use a mesh of size either $20 \times 20 \times 20$ or $40 \times 40 \times 40$, meaning that the problem size is either

[§] μ_0 and ε_0 are respectively the magnetic permeability and electric permittivity of vacuum.

DOI: 10.1002/nla

$n = 55\,566$ or $n = 413\,526$. The source function $g(t)$ contains contributions of the coil source function $\boldsymbol{J}$. Because this function is given as a constant vector (defining the position of the coil in space) times a time-dependent function, there is no need to apply the truncated SVD approximation. The range of the matrix U is then at most two-dimensional ($m = 2$), with one dimension coming from the source function and another from the initial values (provided they are not zero).

The time interval is set to $[0, T]$ with dimensionless $T = 800$, that is soon after the current is switched off (the physical time $t = 1.02 \times 10^{-4}$ s when the coil current reaches zero corresponds to dimensionless $t = 765$). This problem can be seen as a relatively hard test problem, because

(1) it involves rather long time intervals;
(2) it is mildly stiff as the dimensionless scaling brings in a factor of 4800π in the conductivity values;
(3) the resulting matrix A is not symmetric.

The standard Krylov subspace methods, including the EXPOKIT program, fail for the time intervals on which they could be competitive because of a convergence stagnation. For this reason, we apply the EBK/SaI method with the direct sparse LU factorization from the UMFPACK package [44, 45] (provided by MATLAB's function `lu`). The costs for this sparse LU factorization appear to be feasible for the values of τ up to roughly 200 (recall that an LU factorization for the matrix $I + \gamma A$, $\gamma = \tau/10$, has to be built). For larger values of τ, the fill in of the sparse LU factorization grows, making the factorization very inefficient. Therefore, we split the time interval $[0, 800]$ into four subintervals of length 200. An important point is that the sparse LU factorization is computed only once for all the four subintervals.

A standard time integration method used in engineering for this type of problems is a hybrid of the explicit Yee scheme for the curl terms $\nabla \times \boldsymbol{E}$ and $\nabla \times \boldsymbol{H}$ and implicit trapezoidal rule for the remaining terms (conductivity and source terms), for example, [46]. This scheme can be seen as a symplectic composition scheme, and has temporal accuracy of order two. For this reason, we call the scheme CO2, composition order two scheme, see [47] for more details. We test EBK/SaI against the CO2 scheme.

Table III. Numerical results for test problem 3. The error reported in brackets for EBK are measured for the transformed problem (2) with zero initial values. The CPU time is measured in MATLAB and thus gives only an indication of the actual performance.

Scheme	CPU time, s	Error	Krylov dimensions
	Mesh $20 \times 20 \times 20$, $n = 55\,566$		
CO2($\tau = 0.025$)	236	4.8e−07	—
CO2($\tau = 0.0125$)	471	1.2e−07	—
ITR($\tau = 0.5$)	83	9.0e−04	—
ITR($\tau = 0.25$)	163	1.8e−04	—
ITR($\tau = 0.125$)	322	6.1e−06	—
ITR($\tau = 0.0625$)	673	1.5e−06	—
EBK/SaI(`toler` $= 1e-4$)	31	6.3e−04 (1.4e−05)	8, 6, 3, 8
EBK/SaI(`toler` $= 1e-6$)	46	8.8e−08 (1.9e−09)	13, 7, 5, 17
	Mesh $40 \times 40 \times 40$, $n = 413\,526$		
CO2($\tau = 0.0125$)	1,345	1.2e−07	—
CO2($\tau = 0.00625$)	2,687	3.0e−08	—
ITR($\tau = 0.5$)	2,230	1.2e−03	—
ITR($\tau = 0.25$)	4,231	3.8e−04	—
ITR($\tau = 0.125$)	8,581	6.0e−06	—
ITR($\tau = 0.0625$)	16,467	1.5e−06	—
EBK/SaI(`toler` $= 1e-4$)	256	1.8e−04 (3.0e−06)	9, 6, 5, 10
EBK/SaI(`toler` $= 1e-6$)	344	6.3e−08 (1.1e−09)	12, 9, 5, 23

DOI: 10.1002/nla

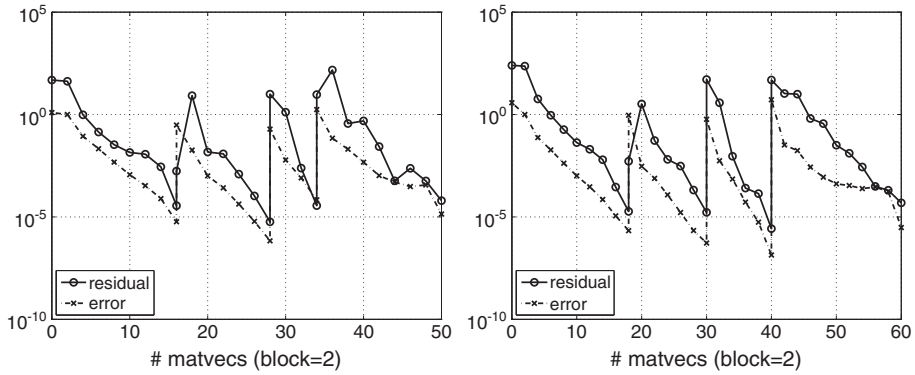

Figure 5. Residual and error convergence plots for the four time steps of EBK/SaI(toler $= 1e$-4) for test problem 3. (Left) Mesh 1, $n = 55\,566$; (Right) mesh 2, $n = 413\,526$.

Because a sparse direct LU factorization is feasible, it is appropriate to include comparisons with an implicit scheme, which relies on such a factorization. Therefore, we include the implicit trapezoidal scheme (ITR) in the tests. ITR is well suited for Maxwell's equations because it is second-order accurate and does not introduce artificial energy dissipation [23]. Just as in the EBK/SaI scheme, the LU factorization for the ITR scheme is computed once before the time stepping process and is used at every time step.

The result of the test runs are presented in Table III. The reported error values are the relative norms $\|y(T) - y_{\text{ref}}(T)\|/\|y_{\text{ref}}(T)\|$, where $y(T)$ is a numerical solution of a certain method and $y_{\text{ref}}(T)$ is the reference solution at the final time. The reference solution is obtained by running the CO2 scheme with very small time steps τ and $\tau/2$ and extrapolating the results. The maximal allowable by stability time steps of the CO2 scheme are roughly $\tau = 0.025$ and $\tau = 0.0125$, respectively on the two meshes. The Krylov dimensions for EBK/SaI are reported in the table for all four time steps.

As we see from Table III, the implicit-explicit CO2 scheme outperforms the fully implicit ITR scheme. This is not a surprise, similar results are reported in [23] for 3D Maxwell's equations with a finite element discretization. The proposed EBK scheme exhibits the best performance of all three schemes, providing both accurate and fast solution. Convergence plots of EBK can be seen in Figure 5.

5. CONCLUSIONS

A block Krylov subspace method is presented for solving linear ODE systems of the form $y' = -Ay + g(t)$ and $y'' = -Ay + g(t)$. In situations where $g(t)$ can be accurately represented by a low rank matrix U times a time dependent function $p(t)$, the method appears to work very well. For the presented test problems, it turns out to be by far more efficient than the second-order accurate exponential time integrator EE2/EXPOKIT.

Further research can be concentrated on efficient updating the approximation $g(t) \approx Up(t)$ to avoid the approximation accuracy loss with a growing time interval. Such an efficient approximation update, possibly combined with a restarting procedure in the block Arnoldi/Lanczos process, should make it possible to extend the approach to nonlinear problems.

ACKNOWLEDGEMENTS

The author thanks Oleg Nechaev for his advice concerning the implementation of the test problem from Section 4.3.

This study was supported by the Russian Federal Program 'Scientific and scientific-pedagogical personnel of innovative Russia', Grant 8500.

DOI: 10.1002/nla

REFERENCES

1. Borzì A, Schulz V. *Computational Optimization of Systems Governed by Partial Differential Equations*. SIAM: Philadelphia, 2011.
2. He X, Funfschilling D, Nobach H, Bodenschatz E, Ahlers G. Transition to the ultimate state of turbulent Rayleigh-Bénard convection. *Physical Review Letters* 2012; **108**(024502).
3. Hochbruck M, Ostermann A. Exponential integrators. *Acta Numerica* 2010; **19**:209–286.
4. Minchev BV, Wright WM. A review of exponential integrators for first order semi-linear problems. *Technical Report 2/05*, Department of Mathematics, NTNU, Norvegian University of Science and Technology, Norway, April 2005. (Available from: http://www.ii.uib.no/~borko/pub/N2-2005.pdf) [Accessed on 15 January 2013].
5. Niesen J, Wright WM. Algorithm 919: a Krylov subspace algorithm for evaluating the φ-functions appearing in exponential integrators. *ACM Transactions on Mathematical Software* 2012; **38**(3):22:1–22:19.
6. Botchev MA, Harutyunyan D, van der Vegt JJW. The Gautschi time stepping scheme for edge finite element discretizations of the Maxwell equations. *Journal of Computational Physics* 2006; **216**:654–686.
7. Mielke A (ed.). *Analysis, Modeling and Simulation of Multiscale Problems*. Springer-Verlag: Berlin, 2006.
8. Lubich C. *From Quantum to Classical Molecular Dynamics: Reduced Models and Numerical Analysis*, Zurich Lectures in Advanced Mathematics. European Mathematical Society (EMS): Zürich, 2008.
9. in 't Hout KJ, Weideman JAC. A contour integral method for the Black-Scholes and Heston equations. *SIAM Journal on Scientific Computing* 2011; **33**(2):763–785.
10. van der Vorst HA. An iterative solution method for solving $f(A)x = b$, using Krylov subspace information obtained for the symmetric positive definite matrix A. *Journal of Computational and Applied Mathematics* 1987; **18**: 249–263.
11. Druskin VL, Knizhnerman LA. Two polynomial methods of calculating functions of symmetric matrices. *USSR Computational Mathematics and Mathematical Physics* 1989; **29**(6):112–121.
12. Tal-Ezer H. Spectral methods in time for parabolic problems. *SIAM Journal on Numerical Analysis* 1989; **26**(1):1–11.
13. Druskin VL, Knizhnerman LA. Krylov subspace approximations of eigenpairs and matrix functions in exact and computer arithmetic. *Numerical Linear Algebra with Applications* 1995; **2**:205–217.
14. Hochbruck M, Lubich C. On Krylov subspace approximations to the matrix exponential operator. *SIAM Journal on Numerical Analysis* Oct 1997; **34**(5):1911–1925.
15. Druskin V, Greenbaum A, Knizhnerman L. Using nonorthogonal Lanczos vectors in the computation of matrix functions. *SIAM Journal on Scientific Computing* 1998; **19**(1):38–54.
16. van den Eshof J, Hochbruck M. Preconditioning Lanczos approximations to the matrix exponential. *SIAM Journal on Scientific Computing* 2006; **27**(4):1438–1457.
17. Tal-Ezer H. On restart and error estimation for Krylov approximation of $w = f(A)v$. *SIAM Journal on Scientific Computing* 2007; **29**(6):2426–2441.
18. Hochbruck M, Niehoff J. Approximation of matrix operators applied to multiple vectors. *Mathematics and Computers in Simulation* 2008; **79**(4):1270–1283.
19. Eiermann M, Ernst OG, Güttel S. Deflated restarting for matrix functions. *SIAM Journal on Matrix Analysis and Applications* 2011; **32**(2):621–641.
20. Al-Mohy AH, Higham NJ. Computing the action of the matrix exponential, with an application to exponential integrators. *SIAM Journal on Scientific Computing* 2011; **33**(2):488–511.
21. De Raedt H, Michielsen K, Kole JS, Figge MT. One-step finite-difference time-domain algorithm to solve the Maxwell equations. *Physical Review E* 2003; **67**:056 706.
22. Tóth G, Keppens R, Botchev MA. Implicit and semi-implicit schemes in the versatile advection code: numerical tests. *Astronomy and Astrophysics* 1998; **332**:1159–1170.
23. Verwer JG, Botchev MA. Unconditionally stable integration of Maxwell's equations. *Linear Algebra and its Applications* 2009; **431**(3–4):300–317.
24. Tal-Ezer H. *Highly Accurate Solution for Time-dependent PDE's*, Lecture on the III International Conference on Matrix Methods in Mathematics and Applications. Institute of Numerical Mathematics of Russian Academy of Sciences: Moscow, June 2011.
25. Gelber AV, Shurina EP, Epov MI. An application of finite element method for solving the problem of modeling non-stationary electromagnetic fields of defectoscope. *International Conference on Computational Mathematics. Part I , II*, ICM&MG Pub., Novosibirsk, 2002; 427–431.
26. Golub GH, Van Loan CF. *Matrix Computations*, Third edn. The Johns Hopkins University Press: Baltimore and London, 1996.
27. Botchev MA, Sleijpen GLG, Sopaheluwakan A. An SVD-approach to Jacobi-Davidson solution of nonlinear Helmholtz eigenvalue problems. *Linear Algebra and its Applications* 2009; **431**:427–440.
28. Enright WH. Continuous numerical methods for ODEs with defect control. *Journal of Computational and Applied Mathematics* 2000; **125**(1–2):159–170. Numerical analysis, 2000, Vol. VI , Ordinary differential equations and integral equations.
29. Shampine LF. Solving ODEs and DDEs with residual control. *Applied Numerical Mathematics* 2005; **52**(1):113–127.
30. Kierzenka J, Shampine LF. A BVP solver that controls residual and error. *Journal of Numerical Analysis, Industrial and Applied Mathematics* 2008; **3**(1–2):27–41.
31. Celledoni E, Moret I. A Krylov projection method for systems of ODEs. *Applied Numerical Mathematics* 1997; **24**(2-3):365–378.

DOI: 10.1002/nla

32. Botchev MA. *Residual, Restarting and Richardson Iteration for the Matrix Exponential*, Memorandum 1928. Department of Applied Mathematics, University of Twente: Enschede, November 2010. (Available from: http://eprints.eemcs.utwente.nl/18832/) [Accessed on 15 January 2013].
33. van der Vorst HA. *Iterative Krylov Methods for Large Linear Systems*. Cambridge University Press: Cambridge, 2003.
34. Saad Y. *Iterative Methods for Sparse Linear Systems*, (2nd edn). SIAM: Philadelphia, 2003. (Available from: www-users.cs.umn.edu/~saad/books.html; http://www.ii.uib.no/%7Eborko/pub/N2-2005.pdf) [Accessed on 15 January 2013].
35. Moret I, Novati P. RD rational approximations of the matrix exponential. *BIT Numerical Mathematics* 2004; **44**:595–615.
36. Hochbruck M, Lubich C. A Gautschi-type method for oscillatory second-order differential equations. *Numerische Mathematik* 1999; **83**:403–426.
37. Grimm V. On error bounds for the Gautschi-type exponential integrator applied to oscillatory second-order differential equations. *Numerische Mathematik* 2005; **100**:71–89.
38. Botchev MA, Grimm V, Hochbruck M. Residual, restarting and Richardson iteration for the matrix exponential. *Technical Report Nr. 12-10*, Department of Mathematics, Karlsruhe Institute of Technology, Karlsruhe, Germany, November 2012. (Available from: www.math.kit.edu/iwrmm/page/preprints/en) [Accessed on 15 January 2013].
39. Berland H, Skaflestad B, Wright WM. EXPINT—a MATLAB package for exponential integrators. *ACM Transactions on Mathematical Software* 2007; **33**(1). (Available from: http://www.math.ntnu.no/num/expint/) [Accessed on 15 January 2013].
40. Krukier LA. Implicit difference schemes and an iterative method for solving them for a certain class of systems of quasi-linear equations. *Soviet Mathematics* 1979; **23**(7):43–55. Translation from Izvestia Vysshih Uchebnyh Zavedenii, Matematika, 1979, No. 7(206), 41–52 (1979).
41. Hochbruck M, Lubich C, Selhofer H. Exponential integrators for large systems of differential equations. *SIAM Journal on Scientific Computing* 1998; **19**(5):1552–1574.
42. Sidje RB. EXPOKIT. A software package for computing matrix exponentials. *ACM Transactions on Mathematical Software* 1998; **24**(1):130–156. (Available from: www.maths.uq.edu.au/expokit/) [Accessed on 15 January 2013].
43. Shurina EP, Gelber AV, Gelber MA, Epov MI. Mathematical modelling of non-stationary electromagnetic fields of defectoscope of casings. *Computational technologies* 2002; **7**(6):114–129. In Russian. (Available from: www.ict.nsc.ru/jct/annotation/346?l=eng) [Accessed on 15 January 2013].
44. Davis TA. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software* 2004; **30**(2):167–195.
45. Davis TA. Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software* 2004; **30**(2):196–199.
46. Rodrigue G, White D. A vector finite element time-domain method for solving Maxwell's equations on unstructured hexahedral grids. *SIAM Journal on Scientific Computing* 2001; **23**(3):683–706.
47. Botchev MA, Verwer JG. Numerical integration of damped Maxwell equations. *SIAM Journal on Scientific Computing* 2009; **31**(2):1322–1346.

DOI: 10.1002/nla