Memorandum 1973 (January 2012). ISSN 1874-4850. Available from: http://www.math.utwente.nl/publications
Department of Applied Mathematics, University of Twente, Enschede, The Netherlands

A block Krylov subspace time-exact solution method for linear ODE systems*

In memory of Gene H. Golub, on occasion of his 20th birthday

M. A. Botchev†

January 13, 2012

Abstract

We propose a time-exact Krylov-subspace-based method for solving linear ODE (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 SVD (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. Since 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 new method, as compared to an exponential time integrator with Krylov subspace matrix function evaluations.

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

AMS MSC2010 subject classification: 65F60, 65L05, 65M20, 65M22, 65F30

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

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

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

*Preliminary version of this report has appeared in arXiv.org: http://arxiv.org/abs/1109.5100

†Department of Applied Mathematics, University of Twente, P.O. Box 217, 7500 AE Enschede, the Netherlands, mbotchev@na-net.ornl.gov.

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

The time step size in explicit time integration methods can often be unacceptably small, for instance, due to the stiffness of the ODE system or due to a locally refined spatial mesh. In this case implicit time integration is of interest. Since recently, a lot of research has been carried out on the so-called exponential time integration schemes, see a recent comprehensive survey [15]. These are time integration schemes involving the matrix exponential and related matrix functions. The interest in exponential time integration is due to the new, challenging applications [20, 19, 16] as well as to the recent progress in techniques to compute actions of matrix functions for large matrices (see e.g. [30, 6, 25, 7, 12, 5, 29, 26, 14, 8, 1]).

In some applications in which explicit time integration is by far inefficient, a gain with implicit or exponential time integration is not guaranteed [28, 32]. 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 are well below the costs of the explicit time integrator times 100. In such situations, the gain is not guaranteed because a linear solver in the implicit time integrator can be quite expensive. In these problems, there is thus only a small niche to for 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 summer 2011 [27]. 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 (see e.g. [15] for a definition). This approach has two restrictions. First, a Taylor polynomial approximation is likely to deteriorate for large time intervals. Second, 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 $g(t)$ [10], time dependent boundary conditions (see 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 SVD (singular value decomposition) is described in Section 2. Section 3 is devoted to the new block Krylov subspace method, Section 4 presents numerical experiments and, finally, Section 5 draws conclusions.

2 Truncated SVD 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 $0 = t_1 < t_2 < \cdots < t_{s-1} < t_s = T$ on the time interval $[0, T]$. The polynomial approximation is based on the truncated SVD (singular value decomposition) 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

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

be the thin SVD [11, 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 $\Sigma \in \mathbb{R}^{m\times m} = \mathrm{diag}(\sigma_1, \ldots, \sigma_m)$ 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 2-norm and Frobenius norm of the error $\tilde{G} - G$ holds [11, Section 2.5.3]:

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

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 $u_1, \ldots, u_m$:

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

where v_{ij} are the entries of the unitary matrix V. Following the approach of [4], we consider the coefficients of these linear combinations, namely $\sigma_j v_{ij}$, $j = 1, \ldots, m$, 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 done in [4]) or by piecewise interpolation polynomials. The latter approach is followed in this paper. More specifically, we approximate

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

where piecewise polynomials $p_j(t)$, $j = 1, \ldots, m$, are obtained by cubic spline interpolation of the functions $f_j(t)$ in points t_i. Packing the polynomials $p_j(t)$, $j = 1, \ldots, m$, in one polynomial vector function $p(t) = (p_1(t), \ldots, p_m(t))^T$, 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 $t_1, \ldots, t_s$ (typically, it is sensible to use Chebyshev of Leja points for interpolation). Second, by the number of terms m in the SVD truncation (4) and, finally, by the piece wise polynomial interpolation in (5). The first two sources of the error are by far dominant but can be easily controlled when the approximation is constructed (see [4] 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 new 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 new methods.

3.1 EBK: exponential block Krylov method

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

$$r_k(t) \equiv -Ay_k(t) - y_k'(t) + Up(t).$$

This residual concept (well known in the ODE literature [9, 23, 19, 17]) can be used as a stopping criterion and for restarting in Krylov subspace methods for matrix exponential [2]. 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

$$r_0(t) = Up(t). \tag{8}$$

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

$$y_k(t) = y_0(t) + \xi_k(t).$$

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

$$\begin{cases} \xi' = -A\xi + r_0(t), \\ \xi(0) = 0, \qquad\qquad t \in [0, T], \end{cases} \tag{9}$$

Note that if $\xi_k(t)$ solves (9) exactly then $y_k(t)$ is the sought-after exact solution of (7). It is natural to solve (9) 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 (see e.g. [31, 22]). 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 $V_i \in \mathbb{R}^{n\times m}$, and 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 [31, 22]

$$AV_{[k]} = V_{[k+1]}H_{[k+1,k]} = V_{[k]}H_{[k,k]} + V_{k+1}H_{k+1,k}E_k^T, \tag{10}$$

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 (9) can be computed as

$$\xi_k(t) = V_{[k]}u(t), \tag{11}$$

where $u(t)$ solves the projected IVP

$$\begin{cases} u'(t) = -H_{[k,k]}u(t) + E_1p(t), \\ u(0) = 0, \qquad\qquad t \in [0, T], \end{cases} \tag{12}$$

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

$$E_1p(t) = V_{[k]}^T r_0(t) = V_{[k]}^T V_1 p(t).$$

Using (8), (10) and (12), we see that for the exponential residual $r_k(t)$ of the solution $y_k(t)$ holds

$$\begin{aligned} r_k(t) &= -Ay_k - y_k' + Up(t) = -Ay_0 - y_0' - AV_{[k]}u(t) - V_{[k]}u'(t) + Up(t) = \\ &= r_0(t) - AV_{[k]}u(t) - V_{[k]}u'(t) = \\ &= r_0(t) - (V_{[k]}H_{[k,k]} + V_{k+1}H_{k+1,k}E_k^T)u(t) - V_{[k]}u'(t) = \\ &= r_0(t) - V_{[k]}(H_{[k,k]}u(t) + u'(t)) - V_{k+1}H_{k+1,k}E_k^T u(t) = \\ &= r_0(t) - V_{[k]}E_1 p(t) - V_{k+1}H_{k+1,k}E_k^T u(t) = -V_{k+1}H_{k+1,k}E_k^T u(t). \end{aligned} \tag{13}$$

A similar expression for the exponential residual is obtained in [2] for a non-block Krylov subspace method. There are two important messages relation (13) 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 (8), 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 relation (8) holds with $U := V_{k+1}$ and $p(t) := -H_{k+1,k}E_k^T u(t)$. The just described correction with k block Krylov iterations can then be repeated, which results in a restarted block Krylov subspace method for solving (7).

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

3.2 An extension to rational Krylov subspaces

Being essentially a matrix exponential method, the new block Krylov method can be generalized to rational Krylov subspaces. We now show how this can be done for the shift-and-invert (SaI) Krylov subspace methods for the matrix exponential [21, 29]. 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 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 (10) transforms into

$$(I + \gamma A)^{-1}V_{[k]} = V_{[k+1]}\tilde{H}_{[k+1,k]} = V_{[k]}\tilde{H}_{[k,k]} + V_{k+1}\tilde{H}_{k+1,k}E_k^T, \tag{14}$$

where the notation of (10) 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

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

where I is the $km \times km$ identity matrix. In practice it is convenient to use relation (14) rewritten as [29, formula (4.1)]

$$AV_{[k]} = V_{[k]}H_{[k,k]} - R_k, \qquad R_k = \frac{1}{\gamma}(I + \gamma A)V_{k+1}\tilde{H}_{k+1,k}E_k^T\tilde{H}_{[k,k]}^{-1}. \tag{15}$$

There are essentially two simple modifications needed in the EBK method to adapt to the SaI subspace. First, the small projected problem (12) should be adjusted and, second, the expression for the residual (13) should be changed. Rewriting IVP (7) in an equivalent form involving the matrix $(I+\gamma A)^{-1}$ and forming its Galerkin projection onto the SaI Krylov subspace colspan$(V_{[k]})$, we arrive at the SaI projected IVP (cf. (12))

$$\begin{cases} \tilde{u}'(t) = -H_{[k,k]}\tilde{u}(t) + \tilde{H}_{[k,k]}^{-1}E_1 p(t), \\ \tilde{u}(0) = 0, \qquad\qquad t \in [0, T], \end{cases} \tag{16}$$

which yields the SaI Krylov subspace correction (11) with $u(t) = \tilde{H}_{[k,k]}\tilde{u}(t)$.

We now derive an expression for the SaI exponential residual. Proceeding as in derivation (13), we have

$$\begin{aligned} r_k(t) &= r_0(t) - AV_{[k]}u(t) - V_{[k]}u'(t) = \\ &= r_0(t) - [V_{[k]}H_{[k,k]} - R_k]u(t) - V_{[k]}u'(t) = \\ &= r_0(t) - V_{[k]}[H_{[k,k]}u(t) + u'(t)] + R_k u(t) = \\ &= r_0(t) - V_{[k]}[H_{[k,k]}\tilde{H}_{[k,k]}\tilde{u}(t) + \tilde{H}_{[k,k]}\tilde{u}'(t)] + R_k u(t) = \\ &= r_0(t) - V_{[k]}\tilde{H}_{[k,k]}[H_{[k,k]}\tilde{u}(t) + \tilde{u}'(t)] + R_k u(t) = \\ &= r_0(t) - V_{[k]}\tilde{H}_{[k,k]}[\tilde{H}^{-1}_{[k,k]}E_1 p(t)] + R_k u(t) = r_0(t) - V_{[k]}E_1 p(t) + R_k u(t) = \\ &= R_k u(t) = \frac{1}{\gamma}(I + \gamma A)V_{k+1}\tilde{H}_{k+1,k}E_k^T\tilde{H}^{-1}_{[k,k]}u(t), \end{aligned} \tag{17}$$

where we used the Arnoldi decomposition (15) and the fact that the matrices $\tilde{H}_{[k,k]}$ and $H_{[k,k]}$ commute.

Again, just as for the relation (13), the message provided by (17) is two-fold. 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.

3.3 A block method for the second order ODE systems

The just introduced block Krylov method can be easily adapted to the second order IVP

$$\begin{cases} y'' = -Ay + g(t), \\ y(0) = v,\ y'(0) = w, \qquad\qquad t \in [0, T], \end{cases} \tag{18}$$

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 done by the introducing a new variable $\tilde{y}(t) \equiv y(t) - v - tw$, similarly to the transformation from (1) to (2). Again, for simplicity of the presentation, we omit the $\tilde{\cdot}$ in the transformed IVP. The approximation procedure $g(t) \approx Up(t)$ of the (transformed) $g(t)$ then leads to a problem

$$\begin{cases} y'' = -Ay + Up(t), \\ y(0) = 0,\ y'(0) = 0, \qquad\qquad t \in [0, T]. \end{cases} \tag{19}$$

Galerkin projection of (19) onto the block Krylov subspace results in a projected IVP

$$\begin{cases} u''(t) = -H_{[k,k]}u(t) + E_1 p(t), \\ u(0) = u'(0) = 0, \qquad\qquad t \in [0, T]. \end{cases} \tag{20}$$

The only remaining modification with respect to the EBK scheme concerns the residual update. Proceeding as in (13), we obtain:

$$\begin{aligned} r_k(t) &= -Ay_k - y_k'' + Up(t) = -Ay_0 - y_0'' - AV_{[k]}u(t) - V_{[k]}u''(t) + Up(t) = \\ &= r_0(t) - AV_{[k]}u(t) - V_{[k]}u''(t) = \\ &= r_0(t) - (V_{[k]}H_{[k,k]} + V_{k+1}H_{k+1,k}E_k^T)u(t) - V_{[k]}u''(t) = \\ &= r_0(t) - V_{[k]}(H_{[k,k]}u(t) + u''(t)) - V_{k+1}H_{k+1,k}E_k^T u(t) = \\ &= r_0(t) - V_{[k]}E_1 p(t) - V_{k+1}H_{k+1,k}E_k^T u(t) = -V_{k+1}H_{k+1,k}E_k^T u(t). \end{aligned} \tag{21}$$

We refer to the just derived method as EBK2, exponential block Krylov method for the second order ODE systems.

3.4 Implementation of the EBK 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 $g(t) \approx Up(t)$.
2. Set $y_0(t) := 0$ and $r_0(t) := Up(t)$. 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 (10):
   compute V_{k+1} and the block column k of $H_{[k+1,k]}$,
   $AV_{[k]} = V_{[k+1]}H_{[k+1,k]} = V_{[k]}H_{[k,k]} + V_{k+1}H_{k+1,k}E_k^T$.
   (b) Find solution $u(t)$ of the projected IVP (12) approximately,
   compute residual with (13): $r_k(t) := -V_{k+1}H_{k+1,k}E_k^T u(t)$.
   (c) if $k =$ `restart` or $\|r_k(t)\|$ is small enough
   solve the projected IVP (12) 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

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 above, 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 $\|r_k(T)\|$ is not enough. For example $\|r_0(T)\|$ turns out to 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 due to a restart or satisfied stopping criterion. In EBK 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(1)*1e3,1e-2])`.

In EBK2, using the stiff MATLAB solver `ode15s` for the nonstiff second order projected IVP (20) 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.

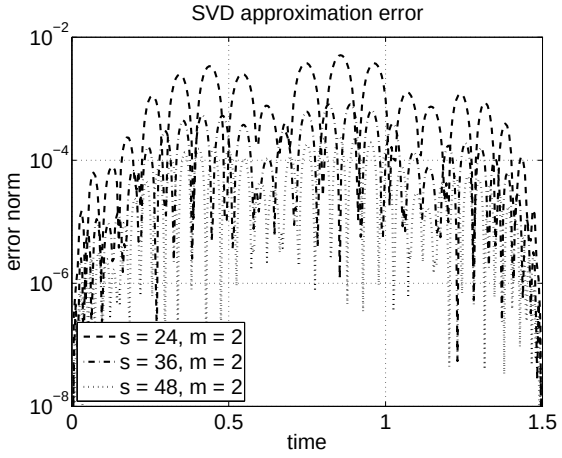

Figure 1: Relative error norm of the SVD approximation error for test problem 1. Mesh 102×102 is used. The time averaged errors on these plots are: `2.5e-04` for $s = 24$ (dashed line), `4.0e-05` for $s = 36$ (dash-dotted line), `1.2e-05` for $s = 48$ (dotted line).

4 Numerical experiments

In this section we present several numerical experiments with the new 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 $\frac{1}{2}(v_1 u_x + v_2 u_y) + \frac{1}{2}((v_1 u)_x + (v_2 u)_y)$, which is possible because the velocity field is divergence free. The convection terms yield an exactly skew-symmetric matrix when discretized in this form [18]. 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 new EBK scheme. Indeed, note that $g(t)$ is, for any t, an element of a two-dimensional subspace $\mathrm{span}\{v, Av\}$, 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 new 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 (see e.g. [13]). Since 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. [15]). Furthermore, having only the φ_1 function allows to employ the EXPOKIT package [24] 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 see e.g. [3]. 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 a comprehensive recent survey [15].

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 gets 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 1. The new EBK 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 gets 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 1, 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 [29]. However, since 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:

$$\begin{cases} u_{tt} = \Delta u, \quad (x,y) \in \Omega \equiv (0,1)^2, \\ u(x,y,0) = 0, \quad u_t(x,y,0) = 0, \\ u\big|_{x=0} = u_{\mathrm{b}}(y,t), \quad u\big|_{\partial\Omega\setminus\{x=0\}} = 0, \end{cases} \tag{22}$$
$$u_{\mathrm{b}}(y,t) = \sin(2\pi t) e^{-100\left(y - \frac{1}{2}\left(1 + \frac{1}{4}\sin(2\pi t)\right)\right)^2}.$$

A time snapshot of the solution for this problem is plot in Figure 2. The standard five-point discretization of (22) yields a second order IVP (18) where $g(t)$ contains the contributions from the 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

Table 1: 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	22400	7.3e-04
EE2/EXPOKIT($\tau = 1.5/200$)	35	38400	1.8e-04
EE2/EXPOKIT($\tau = 1.5/400$)	70	76800	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	22400	7.4e-04
EE2/EXPOKIT($\tau = 1.5/200$)	1096	38400	1.8e-04
EE2/EXPOKIT($\tau = 1.5/400$)	2086	76800	4.6e-05

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 (22). As an illustration, the SVD approximation error, computed in the same way as for test problem 1, is plot in Figure 3.

With this test problem, we compare the new EBK2 scheme (presented in Section 3.3) against the EE2/EXPOKIT scheme. The results are obtained for the final time $T = 0.5$ and `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 large real Ritz values, which are spurious since the eigenvalues the equivalent first order system matrix are purely imaginary. These spurious Ritz values can be both negative or positive and thus harmful for the solution accuracy of the projected IVP. However, they do not seem to harm the EXPOKIT which relies on the Padé approximations of the projected matrix functions.

The results of the test runs of EBK2 and EE2/EXPOKIT are given in Table 2. In the tests, EBK2 is restarted every 20 Krylov steps. Thus, the total Krylov dimension size is at most $20 \times m$, i.e. 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 new 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 [8].

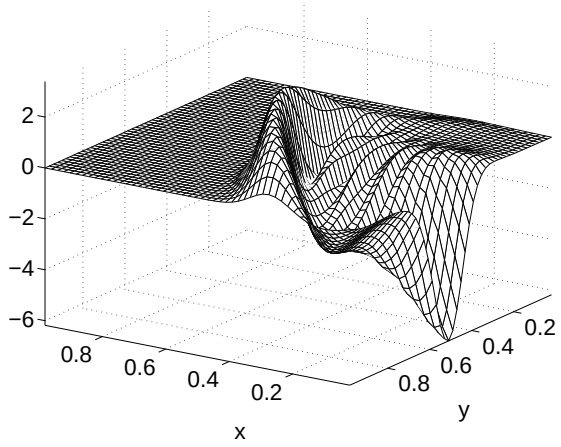

Figure 2: Solution of test problem 2 at $t = 0.5$

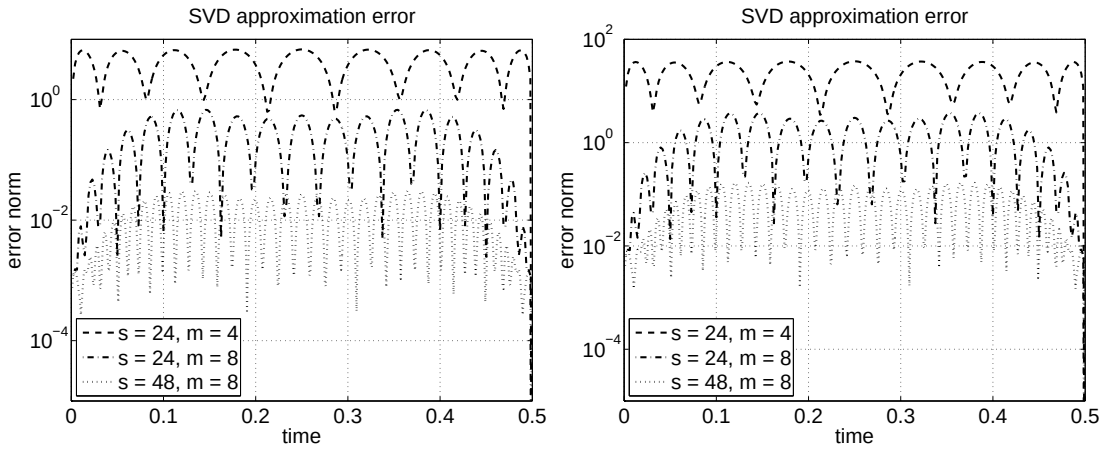

Figure 3: Relative error norm of the SVD approximation error for test problem 2 on meshes 51×51 (left) and 101×101 (right). Note different scales of the y-axes.

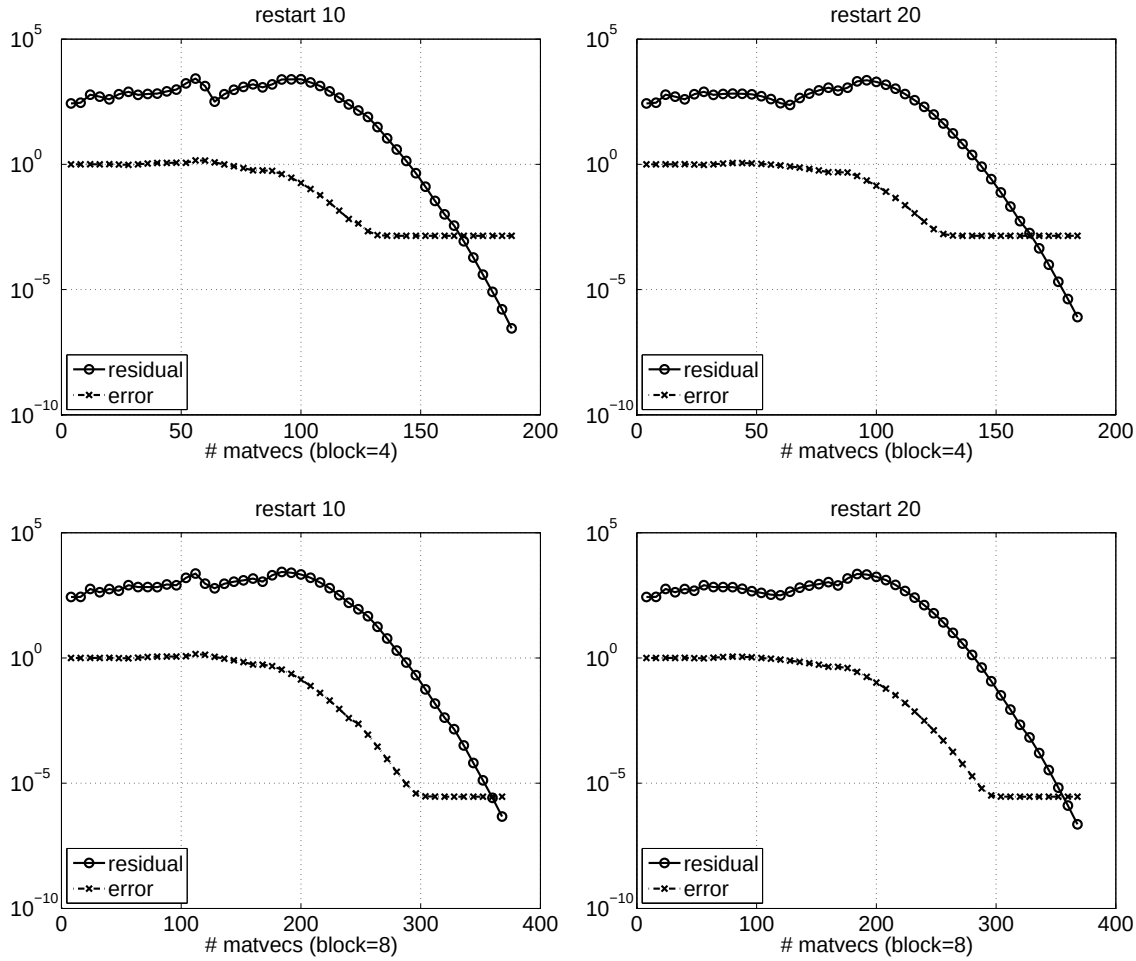

Figure 4: Convergence plots of the new EBK2 scheme restarted every 10 or 20 steps (left/right plots), with either $s = 24$, $m = 4$ or $s = 48$, $m = 8$ (top/bottom plots)

Table 2: 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	19296	2.9e-04
EE2/EXPOKIT($\tau = 0.5/200$)	18	38496	7.3e-05
EE2/EXPOKIT($\tau = 0.5/400$)	35	76800	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	19200	3.2e-04
EE2/EXPOKIT($\tau = 0.5/200$)	95	38400	8.0e-05
EE2/EXPOKIT($\tau = 0.5/400$)	192	76800	2.0e-05

5 Conclusions

A new 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.

References

[1] A. H. Al-Mohy and N. J. Higham. Computing the action of the matrix exponential, with an application to exponential integrators. *SIAM J. Sci. Comput.*, 33(2):488–511, 2011. `http://dx.doi.org/10.1137/100788860`.

[2] M. A. Botchev. Residual, restarting and Richardson iteration for the matrix exponential. Memorandum 1928, Department of Applied Mathematics, University of Twente, Enschede, November 2010. `http://eprints.eemcs.utwente.nl/18832/`.

[3] M. A. Botchev. Residual, restarting and Richardson iteration for the matrix exponential, revised. *ArXiv e-prints*, Dec. 2011. `http://arxiv.org/abs/1112.5670`.

[4] M. A. Botchev, G. L. G. Sleijpen, and A. Sopaheluwakan. An SVD-approach to Jacobi-Davidson solution of nonlinear Helmholtz eigenvalue problems. *Lin. Algebra Appl.*, 431:427–440, 2009. `http://dx.doi.org/10.1016/j.laa.2009.03.024`.

[5] V. Druskin, A. Greenbaum, and L. Knizhnerman. Using nonorthogonal Lanczos vectors in the computation of matrix functions. *SIAM J. Sci. Comput.*, 19(1):38–54, 1998. Special issue on iterative methods (Copper Mountain, CO, 1996).

[6] V. L. Druskin and L. A. Knizhnerman. Two polynomial methods of calculating functions of symmetric matrices. *U.S.S.R. Comput. Maths. Math. Phys.*, 29(6):112–121, 1989.

[7] V. L. Druskin and L. A. Knizhnerman. Krylov subspace approximations of eigenpairs and matrix functions in exact and computer arithmetic. *Numer. Lin. Alg. Appl.*, 2:205–217, 1995.

[8] M. Eiermann, O. G. Ernst, and S. Güttel. Deflated restarting for matrix functions. *SIAM J. Matrix Anal. Appl.*, 32(2):621–641, 2011. `http://dx.doi.org/10.1137/090774665`.

[9] W. H. Enright. Continuous numerical methods for ODEs with defect control. *J. Comput. Appl. Math.*, 125(1-2):159–170, 2000. Numerical analysis 2000, Vol. VI, Ordinary differential equations and integral equations.

[10] A. V. Gelber, E. P. Shurina, and M. I. Epov. An application of finite element method for solving the problem of modeling non-stationary electromagnetic fields of defectoscope. In *International Conference on Computational Mathematics. Part I, II*, pages 427–431. ICM&MG Pub., Novosibirsk, 2002.

[11] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore and London, third edition, 1996.

[12] M. Hochbruck and C. Lubich. On Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 34(5):1911–1925, Oct. 1997.

[13] M. Hochbruck, C. Lubich, and H. Selhofer. Exponential integrators for large systems of differential equations. *SIAM J. Sci. Comput.*, 19(5):1552–1574, 1998.

[14] M. Hochbruck and J. Niehoff. Approximation of matrix operators applied to multiple vectors. *Math. Comput. Simulation*, 79(4):1270–1283, 2008.

[15] M. Hochbruck and A. Ostermann. Exponential integrators. *Acta Numer.*, 19:209–286, 2010.

[16] K. J. in 't Hout and J. A. C. Weideman. A contour integral method for the Black-Scholes and Heston equations. *SIAM J. Sci. Comput.*, 33(2):763–785, 2011.

[17] J. Kierzenka and L. F. Shampine. A BVP solver that controls residual and error. *JNAIAM J. Numer. Anal. Ind. Appl. Math.*, 3(1-2):27–41, 2008.

[18] L. A. Krukier. Implicit difference schemes and an iterative method for solving them for a certain class of systems of quasi-linear equations. *Sov. Math.*, 23(7):43–55, 1979. Translation from Izv. Vyssh. Uchebn. Zaved., Mat. 1979, No. 7(206), 41–52 (1979).

[19] C. Lubich. *From quantum to classical molecular dynamics: reduced models and numerical analysis*. Zurich Lectures in Advanced Mathematics. European Mathematical Society (EMS), Zürich, 2008.

[20] A. Mielke, editor. *Analysis, modeling and simulation of multiscale problems*. Springer-Verlag, Berlin, 2006.

[21] I. Moret and P. Novati. RD rational approximations of the matrix exponential. *BIT*, 44:595–615, 2004.

[22] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Book out of print, 2000. `www-users.cs.umn.edu/~saad/books.html`.

[23] L. F. Shampine. Solving ODEs and DDEs with residual control. *Appl. Numer. Math.*, 52(1):113–127, 2005.

[24] R. B. Sidje. EXPOKIT. A software package for computing matrix exponentials. *ACM Trans. Math. Softw.*, 24(1):130–156, 1998. `www.maths.uq.edu.au/expokit/`.

[25] H. Tal-Ezer. Spectral methods in time for parabolic problems. *SIAM J. Numer. Anal.*, 26(1):1–11, 1989.

[26] H. Tal-Ezer. On restart and error estimation for Krylov approximation of $w = f(A)v$. *SIAM J. Sci. Comput.*, 29(6):2426–2441 (electronic), 2007.

[27] H. Tal-Ezer. 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. `http://matrix.inm.ras.ru/mmma-2011/`.

[28] G. Tóth, R. Keppens, and M. A. Botchev. Implicit and semi-implicit schemes in the Versatile Advection Code: numerical tests. *Astronomy and Astrophysics*, 332:1159–1170, 1998.

[29] J. van den Eshof and M. Hochbruck. Preconditioning Lanczos approximations to the matrix exponential. *SIAM J. Sci. Comput.*, 27(4):1438–1457, 2006.

[30] 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.*, 18:249–263, 1987.

[31] H. A. van der Vorst. *Iterative Krylov methods for large linear systems*. Cambridge University Press, 2003.

[32] J. G. Verwer and M. A. Botchev. Unconditionally stable integration of Maxwell's equations. *Linear Algebra and its Applications*, 431(3–4):300–317, 2009.