Approximation of matrix operators applied to multiple vectors ⋆

Marlis Hochbruck and Jörg Niehoff

Mathematisches Institut, Heinrich-Heine-Universität Düsseldorf, Universitätsstr. 1, D–40225 Düsseldorf, Germany

Abstract

In this paper we propose a numerical method for approximating the product of a matrix function with multiple vectors by Krylov subspace methods combined with a QR decomposition of these vectors. This problem arises in the implementation of exponential integrators for semilinear parabolic problems. We will derive reliable stopping criteria and we suggest variants using up- and downdating techniques. Moreover, we show how Ritz vectors can be included in order to speed up the computation even further. By a number of numerical examples, we will illustrate that the proposed method will reduce the total number of Krylov steps significantly compared to a standard implementation if the vectors correspond to the evaluation of a smooth function at certain quadrature points.

Key words: Krylov subspace methods, shift and invert Lanczos process, projection method, matrix exponential function, matrix functions, restarts, error analysis, QR decomposition, multiple right-hand sides, exponential integrators

1 Introduction

Recently, exponential integrators for semilinear parabolic problems have attracted a lot of interest [5–9,20,21,25,28,33]. The main computational effort for the implementation of these methods is the computation or approximation of the product of a matrix function with a vector. In parabolic problems, the matrix stems from the spatial discretization of an elliptic operator. If this discretization is based on finite differences or finite elements, the resulting matrix

⋆ This work is supported by the Deutsche Forschungsgemeinschaft.
Email addresses: marlis@am.uni-duesseldorf.de (Marlis Hochbruck), niehoff@am.uni-duesseldorf.de (Jörg Niehoff).

is large and sparse. The most popular method for approximating such matrix operators is by Krylov subspace methods [2–4,11–13,15,16,19,23,24,32,34,36]. A more recent development for symmetric (or sectorial) matrices is to use rational approximations [26,29,30,37,38]. A major difficulty with general matrix functions is the lack of a residual. Therefore, it is challenging to reuse information from previous time steps and most of the methods proposed for linear systems cannot be generalized to this application.

In this paper we consider the approximation of

$$z_i := f(\tau A)g_i, \qquad g_i = g(t + c_i\tau), \qquad i = 1, \ldots, n \tag{1}$$

where g is a smooth function, A is a square (symmetric or nonsymmetric) real matrix, $\tau > 0$ denotes the time step of some numerical integration scheme, and $c_1, \ldots, c_n$ are given quadrature nodes. For simplicity, we restrict ourselves to real problems, but everything can be generalized to complex problems with only obvious modifications.

Such problems arise in the implementation of exponential Runge-Kutta or multistep methods. Note that within these integrators the vector g_i depends on previous vectors $g_1, \ldots, g_{i-1}$, so block methods which use all vectors in the very beginning are not applicable. A standard implementation would be to run the Lanczos process or the Arnoldi process for each of the n vectors $g_1, \ldots, g_n$ separately, hence ignoring previously computed information.

The idea to reuse information from previous computations is to compute an orthonormal basis $q_1, \ldots, q_n$ of the vectors $g_1, \ldots, g_n$ and to compute bases of the Krylov subspaces with respect to q_i instead of g_i. We will prove that the entries of the upper triangular factor of the QR decomposition of the matrix build from $g_1, \ldots, g_n$ have entries of magnitude $O(\tau^{i-1})$ in the ith row. This can be exploited by the fact that the tolerance for stopping the ith iteration can be enlarged by a factor of order $O(\tau^{1-i})$ compared to the tolerance of the first iteration, so that fewer and fewer iterations are required. For the solution of linear systems with multiple right-hand sides, a similar approach was proposed by Fischer [14] and Lötstedt and Nilsson [27]. However, since we cannot access residual vectors for general matrix functions, we have to modify their algorithms, in particular we need a much more detailed analysis than in [14,27] to construct reliable and efficient stopping criteria.

We would like to stress that the methods we consider in this paper can be used with standard Krylov subspace methods (Lanczos, Arnoldi, Chebyshev methods, etc.) as well as with rational variants, e.g. the shift and invert methods (for symmetric or sectorial matrices) from [29,38]. However, one could also use other methods than Krylov subspace methods to approximate the matrix operator.

The paper is organized as follows: in Section 2 we present the basic idea and some theoretical results which are required to motivate the procedure and to construct stopping criteria. Section 3 deals with practical aspects of the implementation such as restarting and up- and downdating of the QR decomposition. We also show how to incorporate approximate Ritz vectors into the QR decomposition. In Section 4 we present a number of numerical experiments which show that the new method can speed up a standard implementation of exponential integrators such that up to 80–90% of the overall computational time can be saved. Moreover, we give comparisons to standard (non exponential) integrators for stiff problems.

2 Orthogonalization procedure

In the following we assume that $g : \mathbb{R} \to \mathbb{R}^N$ is a sufficiently smooth function so that the constants

$$M_i := \max_{\xi \in [t+c_1\tau, t+c_n\tau]} \|g^{(i)}(\xi)\|, \qquad i = 0, \ldots, n$$

are reasonably bounded. $\tau > 0$ denotes the step size (e.g. of some time integration scheme) and

$$0 \le c_1 < \ldots < c_n$$

are given quadrature nodes.

The motivation for the proposed algorithm is based on the following theorem.

Theorem 1 *The QR decomposition*

$$[g_1, \ldots, g_n] = QR, \qquad Q = [q_1, \ldots, q_n], \quad R = \{r_{ik}\}_{i,k=1}^n$$

satisfies

$$|r_{ik}| \le C_i M_{i-1} \tau^{i-1} \qquad i \le k, \tag{2}$$

(and $r_{ik} = 0$, for $i > k$), for $i, k = 1, \ldots, n$. The constants C_i only depend on the nodes c_i but are independent of g and τ.

Remark The theorem states that the entries of the ith row of the upper triangular factor R of the QR decomposition are of magnitude $O(\tau^{i-1})$.

Proof Without loss of generality we assume $c_1 = 0$. Obviously, $r_{ik} = q_i^T g_k$ for $i \le k$, so that we can bound

$$|r_{1k}| = |q_1^T g_k| \le \|g(t + c_k\tau)\| \le M_0, \qquad k = 1, \ldots, n. \tag{3}$$

In the following we consider $i \ge 2$. The polynomial p_{i-2} of degree $i-2$ which interpolates g in the nodes $t + c_j\tau$, $j = 1, \ldots, i-1$ can be written in Lagrange

form as

$$p_{i-2}(\sigma) = \sum_{j=1}^{i-1} \ell_j(\sigma) g_j. \tag{4}$$

Obviously, we have

$$q_i^T p_{i-2}^{(k)}(\sigma) = 0, \qquad i = 2, \ldots, n, \qquad k \geq 0.$$

Let $E_{i-2}(\sigma) := g(\sigma) - p_{i-2}(\sigma)$ be the interpolation error.

For $k > i$, Taylor expansion yields for some $\theta \in (c_i, c_k)$

$$\begin{aligned}
|r_{ik}| &= |q_i^T g(t + c_k\tau)| \\
&\leq \|E_{i-2}(t + c_k\tau)\| \\
&= \|\sum_{j=0}^{i-2} \frac{(c_k - c_i)^j}{j!} \tau^j E_{i-2}^{(j)}(t + c_i\tau) + \frac{(c_k - c_i)^{i-1}}{(i-1)!} \tau^{i-1} E_{i-2}^{(i-1)}(t + \theta\tau)\| \\
&\leq C_i M_{i-1} \tau^{i-1}
\end{aligned}$$

by Lemma 1 below and

$$\|E_{i-2}^{(i-1)}(t + \theta\tau)\| = \|g^{(i-1)}(t + \theta\tau)\|.$$

For $k = i$ the same bound is derived directly from Lemma 1. $\square$

The following Lemma is probably not new but to the best of our knowledge the result is not given in the literature in the form required for our purposes. We therefore present it here with a complete proof.

Lemma 1 *Let p_n be the polynomial interpolating g in the nodes $a = x_1 < \ldots < x_{n+1} < b$. Then the interpolation error $E_n = p_n - g$ satisfies for $x_{n+1} < x \leq b$*

$$\|E_n^{(k)}(x)\| \leq C_{n,k} M_{n+1}(x)(x - x_1)^{n+1-k}, \qquad k = 0, \ldots, n,$$

where $M_j(x) = \max_{\xi \in [x_1, x]} \|g^{(j)}(\xi)\|$ and

$$C_{n,k} \leq \frac{1}{(n+1-k)!} + kn \left(\frac{\Delta}{\delta}\right)^{n-1},$$

with $\delta := \min_{j=1}^{n} |x_{j+1} - x_j|$ and $\Delta := |x_{n+1} - x_1|$.

Proof Due to $x_n < x < b$, the interpolation error can be written as

$$E_n(x) = \frac{1}{n!} \int_a^b g^{(n+1)}(\sigma) \kappa_n(x, \sigma) \mathrm{d}\sigma$$

where

$$\kappa_n(x,\sigma) = (x-\sigma)_+^n - \sum_{j=1}^{n+1}(x_j-\sigma)_+^n \ell_j(x),$$

denotes the Peano kernel

$$(x-\sigma)_+ = \begin{cases} x-\sigma, & x \geq \sigma, \\ 0, & x < \sigma. \end{cases}$$

For $k = 0, \ldots, n$, the kth derivative of the error is

$$\begin{aligned}
\|E_n^{(k)}(x)\| &= \|\frac{1}{n!}\int_a^x g^{(n+1)}(\sigma)\frac{\partial^k}{\partial x^k}\kappa_n(x,\sigma)\mathrm{d}\sigma\| \\
&\leq \frac{M_{n+1}(x)}{n!}\int_a^x \left|\frac{\partial^k}{\partial x^k}\left((x-\sigma)^n - \sum_{j=1}^{n+1}(x_j-\sigma)_+^n\ell_j(x)\right)\right|\mathrm{d}\sigma \\
&= \frac{M_{n+1}(x)}{n!}\int_a^x \left|\frac{n!}{(n-k)!}(x-\sigma)^{n-k} - \sum_{j=1}^{n+1}(x_j-\sigma)_+^n\ell_j^{(k)}(x)\right|\mathrm{d}\sigma \\
&\leq M_{n+1}(x)\left(\frac{(x-a)^{n+1-k}}{(n+1-k)!} + \sum_{j=2}^{n+1}|\ell_j^{(k)}(x)|\frac{(x_j-a)^{n+1}}{(n+1)!}\right).
\end{aligned}$$

Because of

$$\begin{aligned}
|\ell_j^{(k)}(x)| &= \left|\prod_{\substack{m=1\\l\neq j}}^{n+1}\frac{1}{x_j-x_m}\;\frac{\partial^k}{\partial x^k}\prod_{\substack{l=1\\l\neq j}}^{n+1}(x-x_l)\right| \\
&= \left|\prod_{\substack{m=1\\l\neq j}}^{n+1}\frac{1}{x_j-x_m}\sum_{j_1=1}^{n+1}\cdots\sum_{j_k=j_{k-1}+1}^{n+1}k!\prod_{\substack{l=1\\l\neq j,l\neq j_1,\ldots,l\neq j_k}}^{n+1}(x-x_l)\right| \\
&\leq \prod_{\substack{m=1\\l\neq j}}^{n+1}\frac{1}{|x_j-x_m|}\;k!\,\frac{(n+1)!}{(k-1)!}\,(x-x_1)^{n-k},
\end{aligned}$$

we get with $a = x_1$

$$\begin{aligned}
\|E_n^{(k)}(x)\| &\leq M_{n+1}(x)(x-x_1)^{n+1-k}\Big(\frac{1}{(n+1-k)!} + \sum_{j=2}^{n+1}k\prod_{\substack{l=1\\l\neq j}}^{n+1}\frac{(x_j-x_1)^n}{|x_j-x_l|}\Big) \\
&\leq M_{n+1}(x)(x-x_1)^{n+1-k}\Big(\frac{1}{(n+1-k)!} + \sum_{j=2}^{n+1}k\prod_{\substack{l=2\\l\neq j}}^{n+1}\frac{(x_j-x_1)^{n-1}}{|x_j-x_l|}\Big) \\
&\leq \left(\frac{1}{(n+1-k)!} + kn\left(\frac{\Delta}{\delta}\right)^{n-1}\right)M_{n+1}(x)(x-x_1)^{n+1-k}.
\end{aligned}$$

This proves the lemma. □

The idea how to exploit the QR decomposition for the approximation of $z_i = f(\tau A)g_i$ is summarized in Algorithm 1. Here and in the following we use Matlab notation to refer to submatrices. In (5) one can use any stopping criterion which is suitable for the particular application, for instance the criterion proposed by Saad [34] or by van den Eshof and Hochbruck [38].

Algorithm 1 Orthogonal projection method

for $i = 1, \ldots, n$ **do**
  Compute the QR decomposition $[g_1, \ldots, g_i] = QR$.
  Compute (Krylov) approximations $\widetilde{w}_i$ to $w_i = f(\tau A)q_i$ such that

$$\|w_i - \widetilde{w}_i\| \leq \mathrm{tol}_i. \tag{5}$$

  Compute $\widetilde{z}_i = [\widetilde{w}_1, \ldots, \widetilde{w}_i]R_{1:i,i}$ as approximation to $z_i = f(\tau A)g_i$.
end for

3 Practical aspects

In this section we discuss some details of the implementation which are necessary to provide an efficient and reliable algorithm. For the computation of the QR decomposition we propose to use reorthogonalization as described by Daniel, Gragg, Kaufman, and Stewart [10], if loss of orthogonality is detected.

3.1 Stopping criterion

An important issue is to determine the tolerances tol_i in (5) such that

$$\|z_i - \widetilde{z}_i\| \leq \mathrm{tol}, \qquad i = 1, \ldots, n \tag{6}$$

is satisfied. Since we do not construct Krylov bases with respect to A and g_i, we cannot use standard stopping criteria here. By definition of the QR decomposition we have

$$\|z_i - \widetilde{z}_i\| \leq \sum_{k=1}^{i} \|w_k - \widetilde{w}_k\| \cdot |r_{k,i}|, \qquad i = 1, \ldots, n.$$

Therefore,

$$\|w_k - \widetilde{w}_k\| \leq \frac{\mathrm{tol}}{i|r_{k,i}|}, \qquad k = 1, \ldots, n, \qquad i = k, \ldots, n$$

is sufficient to guarantee (6) for $i = 1, \ldots, n$. However, since the vectors g_i, $i \geq 2$ are not available in the beginning but can only be computed after

the previous iterations are completed, $r_{k,n}$ is unknown at the time where we need to decide upon stopping the iteration. Fortunately, the analysis in the Section 2 shows that all entries of the kth row of the factor R are of the same magnitude so that – up to constants depending on the function g and the quadrature nodes only – we have

$$|r_{k,i}| \approx |r_{k,n}|, \qquad k = 1, \dots, n, \qquad i = k, \dots, n.$$

This motivates to define

$$\mathrm{tol}_i := \frac{\mathrm{tol}}{n|r_{i,i}|} \tag{7}$$

Due to $r_{i,i} = O(\tau^{i-1})$, this means that the tolerance is enlarged by a factor of $1/\tau$ in each QR step, so that we need fewer and fewer Krylov steps while adding more vectors to the decomposition.

3.2 Updating, downdating, restarts

Algorithm 1 becomes inefficient if the number of vectors used in the QR decomposition becomes too large. The computational cost for the construction of the QR decomposition as well as for the linear combinations to compute the approximations z_i will become prohibitively expensive. Moreover, even though we use reorthogonalization, loss of orthogonality due to roundoff will occur.

The first idea to overcome these difficulties is to restart the algorithm after k vectors. This will limit the storage requirements and the computational cost at the price of throwing away all previously computed information. Alternatively, we suggest to up- and downdate the QR decomposition, so that after the starting phase we always keep k orthogonal vectors, cf. Golub and van Loan [17].

We will now describe the procedure in more detail. Assume we have finished k steps of Algorithm 1, so we have computed Q, R, and approximations $\widetilde{w}_1, \dots, \widetilde{w}_k$ from which we can obtain $\widetilde{z}_1, \dots, \widetilde{z}_k$. We start to downdate the QR decomposition by eliminating q_1. This can be done by computing a QR decomposition of the last $k-1$ columns of R,

$$R \begin{bmatrix} 0 \\ I \end{bmatrix} = H \widehat{R}_{k-1}, \qquad H \in \mathbb{R}^{k,k-1}, \quad \widehat{R}_{k-1} \in \mathbb{R}^{k-1,k-1}.$$

where H is an orthogonal upper Hessenberg matrix and $\widehat{R}_{k-1}$ is upper triangular. This yields

$$[g_2, \dots, g_k] = \widehat{Q}_{k-1} \widehat{R}_{k-1}, \qquad \widehat{Q}_{k-1} = QH \in \mathbb{R}^{N,k-1}.$$

Next we orthogonalize g_{k+1} against $\widehat{Q}_{k-1}$:

$$[g_2, \ldots, g_{k+1}] = \widehat{Q}_k \widehat{R}_k, \qquad \widehat{Q}_k \in \mathbb{R}^{N,k}, \quad \widehat{R}_k \in \mathbb{R}^{k,k}.$$

If we define

$$v_i = f(\tau A)\widehat{q}_i, \qquad i = 1, \ldots k,$$

then we have by construction

$$[v_1, \ldots, v_{k-1}] = [w_1, \ldots, w_k]H.$$

Approximations to these vectors are taken from already available approximations via

$$[\widetilde{v}_1, \ldots, \widetilde{v}_{k-1}] = [\widetilde{w}_1, \ldots, \widetilde{w}_k]H$$

and an approximation to v_k is computed by starting a new Krylov process with respect to A and $\widehat{q}_k$.

As in the previous section, it is essential to stop the iteration correctly. Due to H being orthogonal upper Hessenberg and $h_{j+1,j}\widehat{r}_{j,j} = r_{j+1,j+1}$, we have for $j = 1, \ldots, k-1$

$$\begin{aligned} \|v_j - \widetilde{v}_j\| &= \|\sum_{i=1}^{j+1}(w_i - \widetilde{w}_i)h_{ij}\| \\ &\leq \sum_{i=1}^{j} \mathrm{tol}_i + \mathrm{tol}_{j+1}|h_{j+1,j}| \\ &= \sum_{i=1}^{j} \mathrm{tol}_i + \frac{\mathrm{tol}}{k|\widehat{r}_{j,j}|} \end{aligned}$$

From Theorem 1 we have $|\widehat{r}_{j,j}| = O(\tau^{1-j})$, so that

$$\|v_j - \widetilde{v}_j\| \leq \frac{\mathrm{tol}}{k} O(\tau^{1-j}), \qquad j = 1, \ldots, k-1.$$

For v_k we can proceed as in the previous section and use

$$\|v_k - \widetilde{v}_k\| \leq \widehat{\mathrm{tol}}_k := \frac{\mathrm{tol}}{k|\widehat{r}_{k,k}|}$$

as stopping criterion. These criteria ensure (6) for all $i = 1, \ldots, k+1$. Moreover, we are in the same situation as before the up- and downdating step, which means that we can use the same criteria for the next step.

In numerical experiments we found it necessary to restart this up- and downdating procedure after a certain number of steps. We illustrate the effect by the following example.

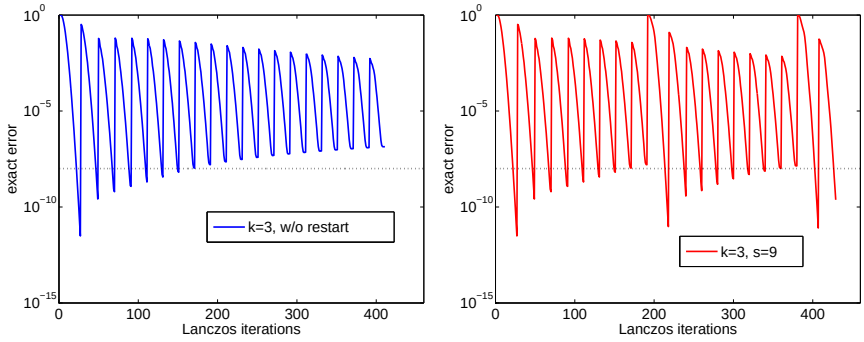

Fig. 1. Effect of restarting the up- and downdating procedure, cf. Example 3.1.

Example 3.1. Let A be a diagonal matrix of dimension $N = 10^5$ with diagonal entries uniformly distributed in the interval $[-400, 0]$ and let be $b_1 = Au$ with a normalized random vector $u \in \mathbb{R}^N$. We choose

$$b_i = A(b_{i-1} + hz_{i-1}), \qquad i = 2, \ldots, n$$

with $z_i = e^A b_i$, $h = 1/n$ and $n = 20$. This recursion resembles 20 time steps of the exponential Euler method. We approximate $e^A b_i$, $i = 1, \ldots, n$ to the accuracy $\text{tol} = 10^{-8}$ with the Lanczos method by using the up- and downdating variant of the QR decomposition with $k = 3$. The left picture of Figure 1 shows the error versus the number of Lanczos iterations without restarts. For the right picture we restarted after $s = 9$ approximations. The dotted line corresponds to the required accuracy tol. The example clearly shows that restarting is necessary. $\diamond$

3.3 Including Ritz vectors

It might be interesting to exploit the knowledge of an almost A invariant subspace into the computation. For instance, in exponential integrators for parabolic problems, f is related to an exponential function. Then the solution $f(\tau A)g_i$ is dominated by eigenvectors corresponding to eigenvalues with largest real part. This motivates to approximate the eigenvectors of these eigenvalues by Ritz vectors extracted from the Krylov subspace with respect to A and g_1. The additional work required for the computation of the Ritz vectors is negligible. However, since the subspace spanned by these Ritz vectors is not exactly A invariant, these results have to be included in the computation in a clever way.

We propose to extract normalized Ritz vectors $u_1, \ldots, u_l$ corresponding to the l Ritz values with largest real part and add these vectors to the QR decomposition,

$$[u_1, \ldots, u_l, g_1, \ldots, g_k] = QR,$$

where the g vectors can be added one after the other. Note that the entries of the upper part of R, $R_{1:l,:}$, are $O(1)$, since in general, the Ritz vectors cannot be interpreted as evaluations of a smooth function. The entries of the submatrix of the bottom right block of R, i.e. $R_{l+1:l+k,l+1:l+k}$, are $O(\tau^{i-1})$ in the ith row due to Theorem 1.

The idea is now to compute the approximations

$$\widetilde{z}_1 \approx z_1 = f(\tau A)g_1, \qquad \|z_1 - \widetilde{z}_1\| \leq \frac{\text{tol}}{l+k}, \tag{8a}$$

$$\widetilde{v}_i \approx v_i = f(\tau A)u_i, \qquad \|v_i - \widetilde{v}_i\| \leq \frac{\text{tol}}{(l+1)(l+k)}, \qquad i = 1, \ldots, l \tag{8b}$$

$$\widetilde{w}_i \approx w_i = f(\tau A)q_{l+i}, \quad \|w_i - \widetilde{w}_i\| \leq \frac{\text{tol}}{(l+k)|r_{l+i,l+i}|}, \quad i = 2, \ldots, k \tag{8c}$$

by constructing Krylov bases with respect to A and g_1, $u_1, \ldots, u_l$ and $q_{l+2}, \ldots, q_{l+i}$. Note that we do not compute a Krylov subspace with respect to A and q_{l+1} since we can recover

$$w_1 = f(\tau A)q_{l+1} = [v_1, \ldots, v_l, z_1]R^{-1}_{1:l+1,l+1}$$

from already computed vectors. The approximation

$$\widetilde{w}_1 = [\widetilde{v}_1, \ldots, \widetilde{v}_l, \widetilde{z}_1]R^{-1}_{1:l+1,l+1}$$

then satisfies

$$\|w_1 - \widetilde{w}_1\| \lesssim l\,\frac{\text{tol}}{(l+1)(l+k)} + \frac{\text{tol}}{(l+k)|r_{l+1,l+1}|} \approx \frac{\text{tol}}{l+k},$$

where $\lesssim$ means that the inequality holds up to constants. For $S = R^{-1}_{1:l,1:l}R_{1:l,l+1:l+k}$ we have

$$z_i = f(\tau A)g_i = [v_1, \ldots, v_l]S_{:,i} + [w_1, \ldots, w_i]R_{l+1:l+i,l+i}.$$

The approximations

$$\widetilde{z}_i = [\widetilde{v}_1, \ldots, \widetilde{v}_l]S_{:,i} + [\widetilde{w}_1, \ldots, \widetilde{w}_i]R_{l+1:l+i,l+i}$$

then satisfy

$$\|z_i - \widetilde{z}_i\| \lesssim l\,\frac{\text{tol}}{l+k} + \sum_{j=1}^{i} \frac{\text{tol}}{(l+k)|r_{l+j,l+j}|}\,|r_{l+j,l+i}|.$$

From Theorem 1 we have $|r_{l+j,l+i}| \approx |r_{l+j,l+j}|$ so that (6) is satisfied (up to constants).

In our practical algorithm we compute Ritz vectors only in the very first time step. We stop the Krylov process when the stopping criterion (8a) is satisfied.

The number of actually used Ritz vectors is determined automatically by the following strategy: we compute $l_{\max}$ eigenpairs of the Hessenberg matrix (e.g. $l_{\max} \leq 10$) and consider the residual norms of the corresponding Ritz pairs. These residual norms can be computed efficiently from the eigenpairs of the Hessenberg matrix without computing the Ritz vectors itself, cf. [1, Section 4.4 and 7.5]. Then we use all Ritz pairs, which have a residual norm less than a moderate tolerance. In our experiments, we found 0.1 sufficient. Note that we do not extend the Krylov space to improve the accuracy of the Ritz pairs.

Up- and downdating is only performed with the vectors $g_1, \ldots, g_n$. We will restart this up- and downdating procedure after s vectors g_j have been included in the QR decomposition.

After running a large number of experiments we suggest to use k between 3 and 5, if the application does not lead to a natural choice. For instance, if the application is an multistep or multistage exponential integrator, than the number of stages is a possible candidate for k. Moreover, we suggest to choose s as a multiple of k, for instance $s = 2k, 3k, 4k$.

4 Numerical experiments

We consider the semilinear parabolic problem

$$\frac{\partial U}{\partial t}(x,t) - \Delta U = \frac{1}{1+U(x,t)^2} + \Phi(x,t) \tag{9}$$

for $x \in [0,1]^d$ and $t \in [0,1]$, subject to homogeneous Dirichlet boundary conditions. The source function Φ is chosen in such a way that the exact solution of the problem is

$$U(x,t) = e^t \prod_{i=1}^{d} x_i(1-x_i).$$

The spatial discretization is done by finite differences with N interior grid points in each space dimension. This results in the following system of ordinary differential equations

$$\frac{\partial u}{\partial t}(t) = Au + \rho(t,u), \qquad A \in \mathbb{R}^{N^d,N^d}.$$

Note that due to $\|A\| \approx (N+1)^2$, the problem is stiff. The matrix A is symmetric, negative definite, so that we can apply the symmetric Lanczos process or its shift and invert variant [29,38] as underlying Krylov method.

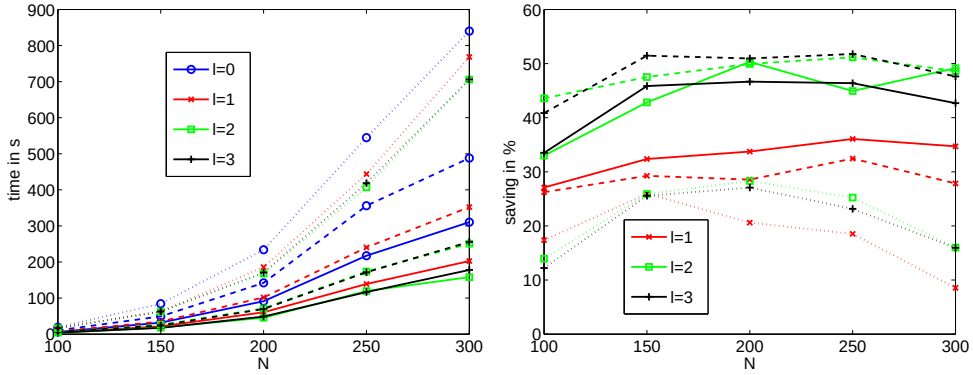

Fig. 2. Cpu time in seconds (left) and savings in percent (right) versus N for different numbers l of included Ritz vectors, and time steps $\tau = 0.1$ (solid), $\tau = 0.05$ (dashed), $\tau = 0.01$ (dotted). Details are given in Example 4.1.

To illustrate the performance of our method, we have chosen the simplest possible exponential integrator, namely the exponential Euler method

$$u_{n+1} = u_n + \tau\varphi(hA)\Big(Au_n + \rho(t_n, u_n)\Big), \qquad n = 0, 1, 2, \ldots, \tag{10}$$

where $\varphi(z) = (e^z - 1)/z$. In each time step, a product of the matrix function with a vector has to be approximated. This approximation was computed up to a tolerance of $\mathrm{tol} = 10^{-6}$.

Example 4.1. In order to demonstrate the benefit of using Ritz vectors, we study the effect of incorporating a fixed number l of Ritz vectors into the decomposition. This is done on the test problem in dimension $d = 2$ with the exponential Euler scheme for different values of N and τ. The results for the standard Lanczos algorithm as underlying Krylov method are presented in Figure 2. The diagram on the left shows the cpu time versus N, the diagram on the right the saving compared to the case that no Ritz vectors are used ($l = 0$). The solid lines correspond to $\tau = 0.1$, the dashed lines to $\tau = 0.05$, and the dotted lines to $\tau = 0.01$. In this example, using two or three Ritz vectors was most efficient, saving about 40%–50% of the computational time. ◇

In our experiments we found that one cannot benefit from incorporating Ritz vectors into the decomposition if the shift and invert Lanczos process is used as the underlying Krylov process. This is not surprising, since the motivation behind the shift and invert Lanczos process was to approximate the dominant eigenspaces fast and thus an additional gain cannot be expected.

From now on, all experiments involving the standard Krylov process use the automatic selection strategy to determine the optimal number l of used Ritz vectors.

Example 4.2. We consider the test problem in dimension $d = 2$ with dif-

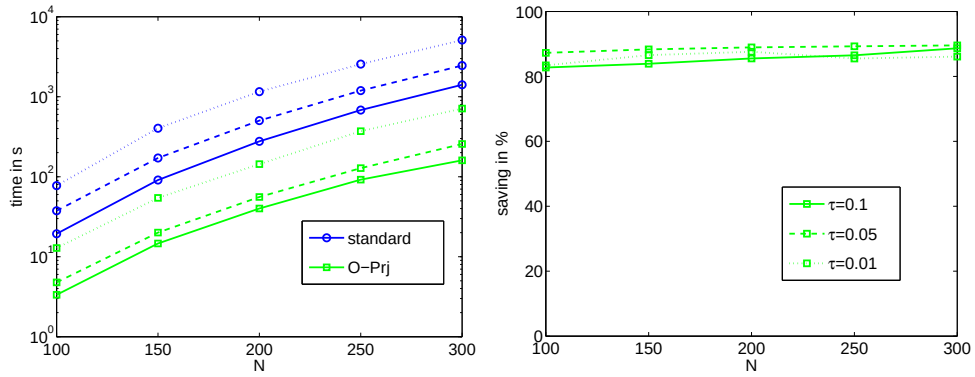

Fig. 3. Cpu time in seconds (left) and saving in percent (right) versus the number of grid points N in each direction of the new algorithm compared to a standard implementation for Example 4.2 for time steps $\tau = 0.1$ (solid), $\tau = 0.05$ (dashed), $\tau = 0.01$ (dotted).

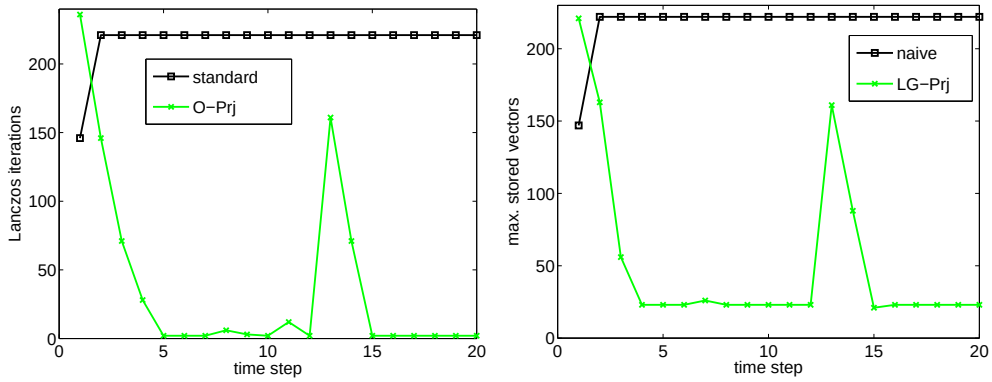

Fig. 4. Number of Lanczos iterations versus time steps (left), maximum number of vectors which have to be kept in memory (right) for $N = 200$ and $\tau = 0.05$ for Example 4.2.

ferent numbers of spatial grid points N. The time integration is done with the exponential Euler method with three different time steps. The standard implementation starts a new Krylov process for approximating the product of $\varphi(\tau A)$ with a vector in each time step. In Figure 3 we compare the standard implementation with our new method where we set $k = 4$ and we restart after $s = 12$ vectors g_j have been included in the QR decomposition using up- and downdating. The picture on the left-hand side shows time steps versus cpu time. On the right-hand side we illustrate how much cpu time we save compared to the standard implementation. Here, the solid line corresponds to $\tau = 0.1$, the dashed line to $\tau = 0.05$, and the dotted line to $\tau = 0.01$. In this example, we save between 80% and 90% of the computational time compared to a standard implementation.

For $N = 200$ and $\tau = 0.05$ the left diagram in Figure 4 shows the number of Lanczos iterations required in each time step. The moderately larger number of Lanczos iterations in the first time step is due to the Krylov processes

for approximating $v_i = \varphi(\tau A)u_i$, for the Ritz vectors u_i, $i = 1, \ldots, l$. The peak at time step 13 is due to the restart. It is not as high as the one in the first step, because we keep the Ritz vectors in the QR decomposition. In the right diagram, we consider the amount of vectors which have to be stored in each time step. It turns out that the orthogonal projection technique does not require more storage than the standard implementation. $\diamond$

Example 4.3. This example is identical to Example 4.2, except that the shift and invert Lanczos iteration [29,38] was chosen as underlying Krylov method (and hence no Ritz vectors were included). The shifted linear systems are solved directly by LU decomposition with reordering. The results are shown in Figures 5 and 6. Comparing the timings with those of Example 4.2, one observes that shift and invert Lanczos is significantly faster. This is due to the fact that only very few Krylov iterations are necessary to achieve the desired accuracy, see Figure 6 (left diagram). As a consequence, the savings are not as pronounced as in the previous example, but still we save between 20% and 60%. Note that this example was done also with larger values of N. $\diamond$

We have done similar comparisons for the Krogstad method [25], which is a 4-stage exponential Runge-Kutta method. It has been shown to be of stiff order four for this particular example. We refer to [20] for a detailed error analysis of this and more general exponential Runge-Kutta methods for semilinear parabolic problems. However, since these comparisons led to almost identical pictures as for the exponential Euler method, we decided not to present them here. More experiments can be found in [31].

Next we will show that implementations of exponential integrators using the new algorithm are indeed competitive to standard integrators for parabolic problems in this example. As competitors, we have used the following time integration schemes:

- ode15s, the standard stiff Matlab solver modified such that reordering of A is applied to speed up the solution of the linear systems.
- radau5, a Matlab implementation of the Fortran code Radau5 by Hairer and Wanner [18] with the same reordering as for ode15s.
- rkc, a Matlab implementation of the Fortran code of the Runge-Kutta-Chebyshev method by Sommeijer, Shampine, and Verwer [35].

The exponential integrators used are the Krogstad method [25] and a three-stage Rosenbrock exponential integrator proposed by Hochbruck, Ostermann and Schweitzer [22]. All methods except the Krogstad method are equipped with adaptive time stepping. For Krogstad's method, no reliable error estimation is known, so we used it with constant time steps such that the achieved accuracy at the final time $t = 1$ was of the same order than for the other schemes.

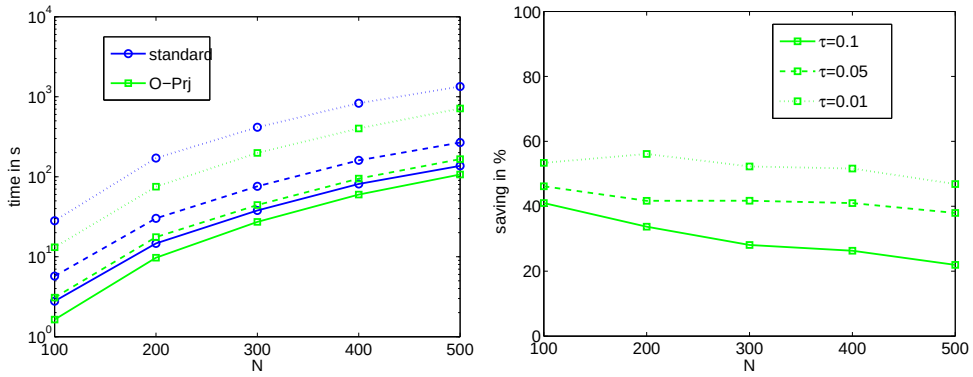

Fig. 5. Cpu time in seconds (left) and saving in percent (right) versus the number of grid points N in each direction of the new algorithm compared to a standard implementation for Example 4.3 for time steps $\tau = 0.1$ (solid), $\tau = 0.05$ (dashed), $\tau = 0.01$ (dotted). The underlying Krylov method is a shift and invert Lanczos process.

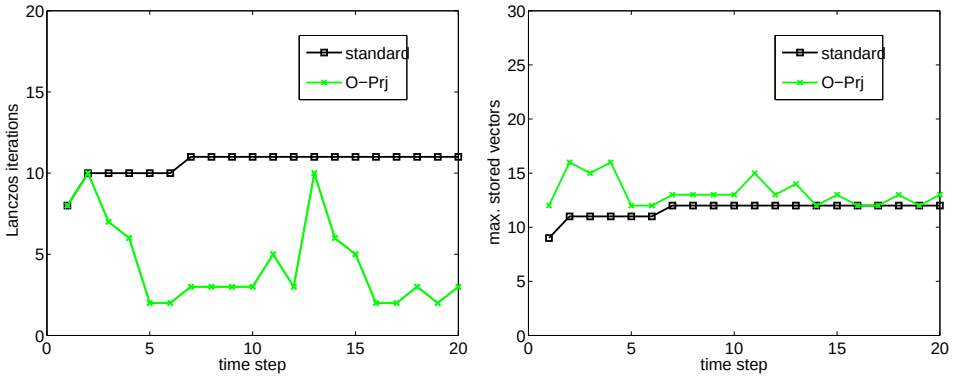

Fig. 6. Number of shift and invert Lanczos iterations versus time steps (left), maximum number of vectors which have to be kept in memory (right) for $N = 500$ and $\tau = 0.05$ for Example 4.3.

For Krogstad's method, two of the inner stages yield approximations to the solution at the same node. We might still apply Theorem 1 but the upper bound (2) turns out to be quite pessimistic in this case. However, the heuristic criteria still give reasonable results.

Example 4.4. We solved the test example in dimension $d = 2$. The underlying Krylov method for the exponential integrators was the shift and invert Lanczos method with orthogonal projection with $k = 4$ including restarts after $s = 12$ vectors. We did not include Ritz vectors into the decomposition. For the solution of the linear systems, we used the same reordering as for the standard ode solvers. This enabled us to present fair comparisons. The error at $t = 1$ was about 10^{-5} for all ode solvers. For Krogstad's method we used $\tau = 0.1$. In Figure 7 we observe that the results for Krogstad's method and for the Rosenbrock exponential integrator are similar to those of ode15s for $N = 400$. For $N = 700$, the Krogstad method took only 555 seconds, the Rosenbrock

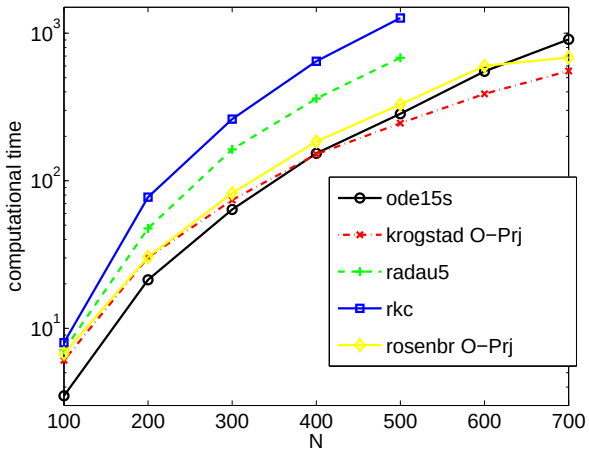

Fig. 7. Cpu times for the solution of the 2d test problem (Example 4.4) for different N with different integrators and with shift and invert Lanczos process for the exponential integrators.

integrator 685 seconds, and ode15s more than 900 seconds. Radau5 and RKC were less efficient in this example. $\diamond$

This example illustrates that exponential integrators have the potential to outperform standard stiff solvers even if fast diagonalization of A (e.g. by fft) is not applicable.

In the next example, we show that for 3d problems, which are generally too large to apply the shift and invert Lanczos process, standard Krylov methods still give good results.

Example 4.5. In this example we set $d = 3$ and we chose a standard Lanczos process as underlying Krylov method using the same parameters as in Example 4.4 but this time we also included Ritz vectors. The codes radau5 and ode15s turned out to be inefficient for $N > 30$. The implementation of the exponential integrators are nearly as good as the RKC code. $\diamond$

Acknowledgments

We would like to thank our former colleague Jasper van den Eshof for fruitful discussions. We are grateful to an anonymous referee for posing interesting questions which motivated us to suggest a new strategy for the selection of Ritz vectors.

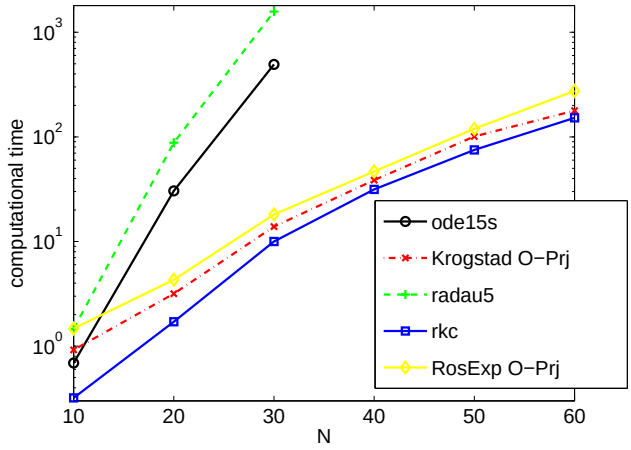

Fig. 8. Cpu times for the solution of the 3d test problem (Example 4.5) for different N with different integrators and with standard Lanczos process for the exponential integrators.

References

[1] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe (eds.), Templates for the solution of algebraic eigenvalue problems, Software, Environments, and Tools, SIAM, Philadelphia, 2000.

[2] L. Bergamaschi, M. Caliari, A. Martínez, M. Vianello, Comparing Leja and Krylov approximations of large scale matrix exponentials, in: V. N. Alexandrov, G. D. van Albada, P. M. A. Sloot, J. Dongarra (eds.), Computational Science — ICCS 2006, vol. 3994 of Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 2006.

[3] L. Bergamaschi, M. Caliari, M. Vianello, The ReLPM exponential integrator for FE discretizations of advection-diffusion equations, in: M. Bubak, G. D. v. Albada, P. M. A. Sloot, J. Dongarra (eds.), Computational Science — ICCS 2004, vol. 3039 of Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 2004.

[4] L. Bergamaschi, M. Vianello, Efficient conputation of the exponential operator for large, sparse, symmetric matrices, Numer. Linear Algebra Appl. 7 (2000) 27–45.

[5] H. Berland, B. Owren, B. Skaflestad, B-series and order conditions for exponential integrators, SIAM J. Numer. Anal. 43 (4) (2005) 1715–1727.

[6] H. Berland, B. Skaflestad, Solving the nonlinear Schrödinger equation using exponential integrators, Tech. rep., Norwegian University of Science and Technology, Trondheim (2005).

[7] M. P. Calvo, C. Palencia, A class of explicit multistep exponential integrators for semilinear problems, Numer. Math. 102 (3) (2005) 367–381.

[8] E. Celledoni, A. Marthinsen, B. Owren, Commutator-free lie group methods, Future Generation Computer Systems 19 (3) (2003) 341–352.

[9] S. M. Cox, P. C. Matthews, Exponential time differencing for stiff systems, J. Comput. Phys. 176 (2) (2002) 430–455.

[10] J. Daniel, W. Gragg, L. Kaufman, G. Stewart, Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization, Math. Comp 30 (1976) 772–795.

[11] V. Druskin, A. Greenbaum, L. Knizhnerman, Using nonorthogonal Lanczos vectors in the computation of matrix functions, SIAM J. Sci. Comp. 19 (1) (1998) 38–54.

[12] V. Druskin, L. Knizhnerman, Krylov subspace approximation of eigenpairs and matrix functions in exact and computer arithmetic, Numer. Linear Algebra Appl. 2 (3) (1995) 205–217.

[13] M. Eiermann, O. Ernst, A restarted Krylov subspace method for the evaluation of matrix functions, SIAM J. Numer. Anal. 44 (6) (2006) 2481–2504.

[14] P. Fischer, Projection techniques for iterative solution of $A\underline{x} = \underline{b}$ with successive right-hand sides., Comp. Meth. Appl. Mech. Eng. 163 (1998) 193–204.

[15] A. Frommer, V. Simoncini, Matrix functions, Tech. rep., Dipartimento di Matematica, Bologna (2006).

[16] E. Gallopoulos, Y. Saad, Efficient solution of parabolic equations by Krylov approximation methods, SIAM J. Sci. Statist. Comput. 13 (1992) 1236–1264.

[17] G. H. Golub, C. F. V. Loan, Matrix Computation, John Hopkins University Press, 1996, third Edition.

[18] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems, 2nd ed., Springer-Verlag, New York, 1996.

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

[20] M. Hochbruck, A. Ostermann, Explicit exponential Runge–Kutta methods for semilinear parabolic problems, SIAM J. Numer. Anal. 43 (3) (2005) 1069–1090.

[21] M. Hochbruck, A. Ostermann, Exponential Runge–Kutta methods for parabolic problems, Appl. Num. Math. 53 (2–4) (2005) 323–339.

[22] M. Hochbruck, A. Ostermann, J. Schweitzer, Exponential integrators of Rosenbrock-type, in: Oberwolfach Reports, vol. 18/2006, Mathematisches Forschungsinstitut Oberwolfach, 2006.

[23] L. A. Knizhnerman, Calculation of functions of unsymmetric matrices using Arnoldi's method, Comput. Math. Math. Phys. 31 (1992) 1–9.

[24] L. A. Knizhnerman, Error bounds in Arnoldi's method: The case of a normal matrix, Comput. Math. Math. Phys. 32 (1992) 1199–1211.

[25] S. Krogstad, Generalized integrating factor methods for stiff PDEs, J. Comput. Phys. 203 (1) (2005) 72–88.

[26] L. Lopez, V. Simoncini, Analysis of projection methods for rational funciton approximation to the matrix exponential, SIAM J. Numer. Anal. 44 (2006) 613–635.

[27] P. Lötstedt, M. Nilsson, A minimal residual interpolation method for linear equations with multiple right hand sides, SIAM J. Sci. Comput. 25 (6) (2004) 2126–2144.

[28] B. Minchev, Exponential integrators for semilinear problems, phd thesis, Tech. rep., University of Bergen, Norway (2004).

[29] I. Moret, P. Novati, RD rational approximations of the matrix exponential, BIT 44 (2004) 595–615.

[30] I. Moret, P. Novati, Interpolation functions of matrices on zeros of quasi-kernel polynomials, Numer. Linear Algebra Appl. 11 (2005) 337–353.

[31] J. Niehoff, Projektionsverfahren zur Approximation von Matrixfunktionen mit Anwendungen auf die Implementierung exponentieller Integratoren, Ph.D. thesis, Mathematisches Institut, Heinrich-Heine-Universität Düsseldorf (2006).

[32] B. Nour-Omid, Applications of the Lanczos algorithm, Comput. Phys. Comm. 53 (1989) 157–168.

[33] A. Ostermann, M. Thalhammer, W. M. Wright, A class of explicit exponential general linear methods, BIT 46 (2) (2006) 409–432.

[34] Y. Saad, Analysis of some Krylov subspace approximations to the matrix exponential operator, SIAM J. Numer. Anal. 29 (1) (1992) 209–228.

[35] B. Sommeijer, L. Shampine, J. Verwer, RKC: an explicit solver for parabolic PDEs, J. Comp. Appl. Math. 88 (1991) 315–326.

[36] D. E. Stewart, T. S. Leyk, Error estimates for Krylov subspace approximations of matrix exponentials, J. Comput. Appl. Math. 72 (1996) 359–406.

[37] L. N. Trefethen, J. A. C. Weidemann, T. Schmelzer, Talbot quadratures and rational approximations, BIT 46 (2006) 653–670.

[38] J. van den Eshof, M. Hochbruck, Preconditioning Lanczos approximations to the matrix exponential, SIAM J. Sci. Comp. 27 (4) (2006) 1438–1457.