REPORTS IN INFORMATICS

ISSN 0333-3590

Computing analytic matrix functions for a class of exponential integrators

Borislav V. Minchev

REPORT NO 278 **June 2004**

Department of Informatics

UNIVERSITY OF BERGEN

Bergen, Norway

This report has URL http://www.ii.uib.no/publikasjoner/texrap/ps/2004-278.ps

Reports in Informatics from Department of Informatics, University of Bergen, Norway, is available at http://www.ii.uib.no/publikasjoner/texrap/.

Requests for paper copies of this report can be sent to:

Department of Informatics, University of Bergen, Høyteknologisenteret,
P.O. Box 7800, N-5020 Bergen, Norway

Computing analytic matrix functions for a class of exponential integrators

Borislav V. Minchev *

Department of Informatics, University of Bergen,
Thormhlensgate 55, N-5020 Bergen, Norway

Abstract

Different methods for numerically stable computation of functions which are closely related to the exponential function are discussed. Such functions appear in the format of the most often used exponential integrators. A generalization of the method base on the tridiagonal reduction is proposed. The new approach, allows to compute all the functions included in the format of a general exponential integrator, in the case when the arguments are symmetric (Hermitian) matrices. Complexity of the considered methods is derived. Some practical issues regarding variable step size implementations as well as the main advantages and disadvantages are of the different methods are also discussed.

Key words: exponential integrators, exponential time differencing, Block Schur–Parlett algorithm, Krylov subspaces, Cauchy integral approach

1 Introduction

In the last few years there has been a lot of renewed interest in the construction of so-called exponential integrators for solving stiff parabolic partial differential equations. These integrators, as there name suggests, use the exponential function and often functions which are closely related to the exponential function, inside the numerical method. We refer to [20] for an overview of different methods based on this idea.

At the moment it is still unclear whether or not the exponential integrators are fully competitive with the existing numerical methods for solving stiff problems. Especially in the case when a variable step size is used. The main concern is that numerical implementation of an exponential integrator might be simply to expensive. A great variety of fast algorithms for computing the matrix exponential operator or its action to a given vector are available in the literature. We refer to [21] and the reference within for a comprehensive review. The aim of this note is to focus on different numerical techniques for computing the rest of the functions used in the format of the exponential integrators, and to discuss the advantages and disadvantages of the different techniques.

The most commonly used functions in the format of an exponential integrator have the following explicit form

$$\phi^{[1]}(z) = \frac{e^z - 1}{z}, \qquad \phi^{[i+1]}(z) = \frac{\phi^{[i]}(z) - \frac{1}{i!}}{z}, \quad \text{for} \quad i = 2, 3, \ldots. \tag{1.1}$$

*email: Borko.Minchev@ii.uib.no, http://www.ii.uib.no/∼borko

We mention that other $\phi^{[i]}$ functions can also be used in the formulation of some exponential integrators (see [18]). However, in most of the known cases, they are closely related with the above defined functions, which are often referred to as the exponential time differencing (ETD) $\phi^{[i]}$ functions [3, 20]. In this note we restrict our considerations only to the case of ETD $\phi^{[i]}$ functions.

Straightforward implementation of (1.1) suffers, for small z, from cancelation errors. An illustration of this phenomena, for the first function $\phi^{[1]}$, is given in [13, Table 2.1]. The cancelation errors are even more sever for the second and the next $\phi^{[i]}$ functions. In practice, the arguments of the functions in (1.1) are matrices of the form $A = \gamma hL$, where $\gamma \in \mathbb{R}$, h is the length of step size and $L \in \mathbb{R}^{d\times d}$ is a discretized linear operator. Usually, the size d of L is very large, thus we need fast, reliable and numerically stable algorithm for computing the functions form (1.1).

The note is organized as follows: We first consider in Section 2 methods based on Schur decomposition followed by higher order rational Chebyshev or Padé approximation to the function $\phi^{[i]}$. Next in Section 3 we present a technic, which allows us to approximate the action of each $\phi^{[i]}$ function on a given vector by means of a projection process onto a small Krylov subspace. In Section 4 we discuss method based on Cauchy integral formula and comment its implementation in some special cases. Finally we summarize the main advantages and disadvantages of the discussed methods in Section 5.

2 Decomposition methods

In the heart of all decomposition methods is the similarity transformation

$$A = SBS^{-1}, \tag{2.1}$$

where B is such that $\phi^{[i]}(B)$ is easy to compute. Then

$$\phi^{[i]}(A) = S\phi^{[i]}(B)S^{-1}. \tag{2.2}$$

The choice of the matrices S and B in (2.1) usually involves two conflicting tasks: Make B close to diagonal so that $\phi^{[i]}(B)$ is easy to compute and make S well conditioned so that errors in evaluating $\phi^{[i]}(B)$ are not magnified. Thus it is natural to consider methods based on Schur decomposition so that S is unitary and therefore well conditioned. Next we present a general algorithm first proposed in [4] which employs the block form of the Parlett recurrence and does not impose any restrictions on the argument matrix A.

2.1 Block Schur-Parlett algorithm

The first step of the algorithm is to compute the Schur decomposition $A = QTQ^*$, where Q is unitary and T is upper triangular. This can be achieved by the standard QR algorithm [8]. We now need to calculate the matrices $F^{[i]} = \phi^{[i]}(T)$. Assuming that the diagonal elements $f_{kk}^{[i]}$ of $F^{[i]}$ are already known, we can compute the hole $F^{[i]}$ using the scalar Parlett recurrence [22]. This is exactly how the MATLAB 6.5 (R13) built in function `funm` works. The problem with this approach is that it fails to produce correct results, in floating point arithmetic, when the eigenvalues of T are equal or close to each other.

To avoid this problem the authors in [4] propose to reorder T in to block upper triangular matrix $\widetilde{T}$. The diagonal blocks $\widetilde{T}_{kk}$ of $\widetilde{T}$ are construct in such a way that the eigenvalues of each diagonal block are "close" to each other and the distinct diagonal blocks have "sufficiently distinct" eigen values. To define the meaning of "close" and "sufficiently distinct" the authors introduce a special parameter in the format of the method. Since $\widetilde{T}$ is block upper diagonal, so is $\widetilde{F}^{[i]} = \phi^{[i]}(\widetilde{T})$. From

the commutativity relation $\widetilde{F}^{[i]}\widetilde{T} = \widetilde{T}\widetilde{F}^{[i]}$ follows the following block form of the Parlett recurrence

$$\widetilde{T}_{kk}\widetilde{F}^{[i]}_{kl} - \widetilde{F}^{[i]}_{kl}\widetilde{T}_{ll} = \widetilde{F}^{[i]}_{kk}\widetilde{T}_{kl} - \widetilde{T}_{kl}\widetilde{F}^{[i]}_{ll} + \sum_{j=k+1}^{l-1}\left(\widetilde{F}^{[i]}_{kj}\widetilde{T}_{jl} - \widetilde{T}_{kj}\widetilde{F}^{[i]}_{jl}\right), \tag{2.3}$$

where $\widetilde{T} = (\widetilde{T}_{kl})$ and $\widetilde{F}^{[i]} = \left(\widetilde{F}^{[i]}_{kl}\right)$. If the diagonal blocks $\widetilde{F}^{[i]}_{kk}$ are already evaluated, the above relation allows us to compute the whole matrix $\widetilde{F}^{[i]}$ by solving the *Sylvester equation* (2.3) with respect to $\widetilde{F}^{[i]}_{kl}$. The imposed requirements for the eigenvalues of $\widetilde{T}_{kk}$ guaranteed that the equation (2.3) is nonsingular and well conditioned [4]. Thus what we still need to specify is how to compute the diagonal blocks $\widetilde{F}^{[i]}_{kk}$. In [4] the authors suggest to use a suitable Taylor series truncation or Padé approximation. This are local approximations which are very accurate near the origin, but may suffer in accuracy a way form it. In addition the accuracy of such approaches depends of the norm of $\widetilde{F}^{[i]}_{kk}$ and usually, when the norm is large, they require scaling and squaring strategy [21]. For the first function $\phi^{[1]}$ analogous formula, similar to the well-known fundamental property of the exponential function, is proposed in [10]. It is based on the equalities $\phi^{[1]}(2z) = (e^z + 1)\phi^{[1]}(z)/2$ and $e^{2z} = e^z e^z$. In general for $i > 1$ we do not have a simple generalization of the above formulas and thus an alternative approach for computing $\widetilde{F}^{[i]}_{kk}$ can be to use higher order Chebyshev (uniform) rational approximation [27]. We discuss this type of approximations in details in Subsection 2.2.

Next we summarize the overall block Schur-Parlett algorithm for computing $\phi^{[i]}(A)$ for a general matrix A and $i = 1, 2, \ldots$.

Algorithm 2.1 *(Block Schur-Parlett algorithm)*

- *Compute the Schur decomposition* $A = QTQ^*$.
- *Reorder* T *in to block upper triangular matrix* $\widetilde{T}$.
- *Compute* $\phi^{[i]}(\widetilde{T}_{kk})$ *for all diagonal blocks* $\widetilde{T}_{kk}$.
- *Find* $\phi^{[i]}(\widetilde{T})$ *by solving the Silvester equation (2.3).*
- *Compute* $\phi^{[i]}(A) = Q\phi^{[i]}(\widetilde{T})Q^*$.

The cost of Algorithm 2.1 depends from the eigenvalue distribution of A, and it is between $28d^3$ and $d^4/3$ flops (see [4]) for each i, where d is the dimensionality of A.

The advantages of this method are that it is numerically stable and works without any restrictions on the structure of the matrix A. The main disadvantage is its higher computational cost, which makes it applicable only if integrators with fix step size are used, so that all $\phi^{[i]}$ functions must be computed only ones, in the beginning of integration process. Thus the above method provides a benchmark for the computational cost of a method.

In the case when A is real symmetric (or complex Hermitian) a cheaper method, also based on the Schur decomposition, can be constructed. In the next subsection we generalize the approach proposed in [15] for computing the matrix exponential operator and later extended in [16] for computing the first function $\phi^{[1]}$, to a general algorithm for computing all the functions from (1.1).

2.2 Tridiagonal reduction

A common first step in the computation of the eigenvalues and the eigenvectors of a *symmetric* matrix A is to use a tridiagonal reduction of the form $A = Q\mathsf{T}Q^T$, where Q is orthogonal and T is symmetric tridiagonal. Such a representation of

A can be obtained by using Hauseholder reflections [8]. Thus according to (2.2) the computation of $\phi^{[i]}(A)$ requires the value $\phi^{[i]}(\mathsf{T})$. In [15, 16] efficient numerical algorithms for computing the exponential and the $\phi^{[1]}$ function, based on the above tridiagonal reduction, followed by Chebyshev (uniform) rational approximation are developed. Here we utilize this approach and propose how it could be generalized, so that each of the functions from (1.1) can be efficiently computed.

We first consider, how to find Chebyshev rational approximation to e^{T}, for a given tridiagonal matrix T. The key idea is to compute the largest eigenvalue λ_1 of T (for example by bisection method) and then make use of the following obvious equality $e^{\mathsf{T}} = e^{\lambda_1} e^{\mathsf{T}-\lambda_1 I}$, where I is $d \times d$ identity matrix. The eigenvalues of $\mathsf{T}-\lambda_1 I$ are always located in the interval $(-\infty, 0]$. This allows us to approximate $e^{\mathsf{T}-\lambda_1 I}$ by rational function $R_p(z) = N_p(z)/D_p(z)$ (where N_p and D_p are polynomials of z of degree p), which minimizes the maximum error for approximating e^z on $(-\infty, 0]$. An approximation to e^{T} is then obtained by

$$e^{\mathsf{T}} \approx e^{\lambda_1} R_p(\mathsf{T} - \lambda_1 I). \tag{2.4}$$

What we gain in this way is that, regardless of the spectrum of T, we can always use a rational Chebyshev approximation to $e^{\mathsf{T}-\lambda_1 I}$ in the interval $(-\infty, 0]$, which has one and same coefficients. If we choose to represent R_p via its partial fraction expansion (see [7])

$$R_p(z) = \alpha_0^{(p)} + \sum_{j=1}^{p} \frac{\alpha_j^{(p)}}{z - \theta_j^{(p)}}, \tag{2.5}$$

the coefficients $\alpha_j^{(p)}$ and the poles $\theta_j^{(p)}$ of R_p can be computed once and for all. To achieve standard double precision (64-bit floating point numbers), it is sufficient to choose $p = 14$. Since the coefficients and the poles appears in complex conjugate pairs, for even p, it is enough to add just the first $p/2$ terms in the sum in (2.5), and then double the real part of the result. This in fact leads to significant computational savings, since it halves the number of matrix inversions in the corresponding formula

$$R_p(\mathsf{T} - \lambda_1 I) = \alpha_0^{(p)} I + \sum_{j=1}^{p} \alpha_j^{(p)} \left[\mathsf{T} - (\lambda_1 + \theta_j^{(p)}) I\right]^{-1}. \tag{2.6}$$

The values of $\alpha_j^{(p)}$ and $\theta_j^{(p)}$ for $p = 14$ and 16 are listed in [16, Table 2].

Here we have chosen *diagonal* rational approximation R_p, that is, the numerator N_p has the same degree as the denominator D_p. We note however, that alternative strategies (e.g. using L-stable approximations) might also be considered without altering the principle of the method.

Before to consider how the approximation (2.4) can be used to approximate any of the functions $\phi^{[i]}$ form (1.1), we mention that, once we have the largest eigenvalue λ_1 of T, we can always calculate a Chebyshev rational approximation to $\phi^{[i]}(z)$ on the interval $(-\infty, \lambda_1]$ and then used it to approximate $\phi^{[i]}(\mathsf{T})$. However, this approach is rather unpractical since it highly depends form the value λ_1. Therefore we need to recompute the coefficients of the approximation for every different λ_1, which is quite a costly task.

Alternative approach for computing $\phi^{[i]}(\mathsf{T})$, in the case when the largest eigenvalue of T belongs to the interval $(-\infty, 0]$, is to replace e^{T} in the definition of each of the functions $\phi^{[i]}$ by its Chebyshev rational approximation (2.4). Thus for example the first function $\phi^{[1]}$ can be approximated by

$$\phi^{[1]}(\mathsf{T}) \approx \mathsf{T}^{-1} \left[\left(e^{\lambda_1} \alpha_0^{(p)} - 1\right) I + e^{\lambda_1} \sum_{j=1}^{p} \alpha_j^{(p)} \left[\mathsf{T} - (\lambda_1 + \theta_j^{(p)}) I\right]^{-1} \right].$$

The above formula should not be used when λ_1 is positive. The problem in this case is that for each of the functions $\phi^{[i]}$, we can not just shift the argument by $\lambda_1 I$, since the relation between $\phi^{[i]}(\mathsf{T})$ and $\phi^{[i]}(\mathsf{T} - \lambda_1 I)$ is not that simple like for the exponential function.

A different approach for approximating $\phi^{[1]}(\mathsf{T})$, in the case when $\lambda_1 > 0$, is proposed in [16]. It is based on the observation that for

$$B = \begin{bmatrix} \mathsf{T} & I \\ 0 & 0 \end{bmatrix},$$

the exponential of B is given by

$$e^B = \begin{bmatrix} e^{\mathsf{T}} & \phi^{[1]}(\mathsf{T}) \\ 0 & I \end{bmatrix}. \tag{2.7}$$

Therefore, applying the approximation (2.4) with R_p given by (2.6) and $\mathsf{T} = B$, we obtain

$$e^B = e^{\lambda_1} e^{B-\lambda_1 I} \approx e^{\lambda_1} \left\{ \alpha_0^{(p)} I + \sum_{j=1}^{p} \alpha_j^{(p)} \left[B - (\lambda_1 + \theta_j^{(p)}) I \right]^{-1} \right\}. \tag{2.8}$$

Equating the entries in positions (1,2) of (2.7) and (2.8) leads to the following approximation for $\phi^{[1]}(\mathsf{T})$

$$\phi^{[1]}(\mathsf{T}) \approx e^{\lambda_1} \sum_{j=1}^{p} \frac{\alpha_j^{(p)}}{\lambda_1 + \theta_j^{(p)}} \left[\mathsf{T} - (\lambda_1 + \theta_j^{(p)}) I \right]^{-1}.$$

We have found that, for λ_1 positive, the same idea can be easily generalized to the case when approximation to $\phi^{[i]}(\mathsf{T})$ for $i > 1$ is needed. To approximate the function $\phi^{[2]}(\mathsf{T})$ we take the matrix B to be of the form

$$B = \begin{bmatrix} \mathsf{T} & 0 & I \\ 0 & 0 & 0 \\ 0 & I & 0 \end{bmatrix}.$$

It is easy to see, for example by direct computation, that its exponential is given by

$$e^B = \begin{bmatrix} e^{\mathsf{T}} & \phi^{[2]}(\mathsf{T}) & \phi^{[1]}(\mathsf{T}) \\ 0 & I & 0 \\ 0 & I & I \end{bmatrix}.$$

As before, based on formulas (2.4) and (2.6), we obtain the following approximation

$$\phi^{[2]}(\mathsf{T}) \approx e^{\lambda_1} \sum_{j=1}^{p} \frac{\alpha_j^{(p)}}{\left(\lambda_1 + \theta_j^{(p)}\right)^2} \left[\mathsf{T} - (\lambda_1 + \theta_j^{(p)}) I \right]^{-1}.$$

Approximations for the rest of the functions $\phi^{[i]}$, for $\lambda_1 > 0$, can be computed by straight-forward generalization of the above process.

In the next algorithm we summarize the method based on the tridiagonal reduction for computing any of the functions $\phi^{[i]}(A)$ for a symmetric matrix A and $i = 1, 2, \ldots$.

Algorithm 2.2 *(Tridiagonal Reduction)*

- *Calculate a symmetric tridiagonal reduction* $A = Q\mathsf{T}Q^T$.
- *Find the largest eigenvalue* λ_1 *of* T.
- *Compute* $\phi^{[i]}(\mathsf{T})$ *by Chebyshev rational approximation with respect to the value of* λ_1.
- *Calculate* $\phi^{[i]}(A)$ *by* $\phi^{[i]}(A) = Q\phi^{[i]}(\mathsf{T})Q^T$.

The coast of the algorithm, for each i is $\mathcal{O}(\frac{4}{3}d^3 + d + d^2 + 2d^3)$, where d is the dimensionality of A. Therefore the total number of operations for computing each of the functions $\phi^{[i]}$ is $\mathcal{O}(\frac{10}{3}d^3)$. In the case of constant step size, once we have computed the first function $\phi^{[1]}$, we can reduce the work for computing the other $\phi^{[i]}$ functions by reusing the tridiagonal decomposition.

If the matrix A is symmetric and tridiagonal, the above algorithm requires only $\mathcal{O}(d^2)$ operations, since its first and last step are not needed. In this case instead of computing all the $\phi^{[i]}$ functions once, before the integration to begin, and then apply them at every step to a different vectors v, we can repeatedly compute the action of each $\phi^{[i]}$ on its corresponding vector v. The total number of operations in this case can still do not exceed the work required to compute the matrix-vector product $\phi^{[i]}$ times v, especially when $d \gg p$. This is because the inverse matrices in the third step of the algorithm are replaced by solutions of linear systems with tridiagonal coefficient matrices. In general, algorithms design to solve such kind of systems require $\mathcal{O}(d)$ operations, which leads to $\mathcal{O}(pd)$ flops needed to compute the action $\phi^{[i]}v$. This is to be compared with $\mathcal{O}(d^2)$ operations needed for a matrix-vector product. In [17], several direct method for solving linear systems of equations are presented. Some of the algorithms proposed there, can be easily adopted to the tridiagonal case considered here. We will comment more on this in Section 4.

When A is tridiagonal, the above approach allows to preserve the total number of operations, needed to implement an exponential integrator, approximately the same even when a variable step size strategy is used. The advantage of the Algorithm 2.2 is that it is less expensive than Algorithm 2.1. Its disadvantages are that it is applicable only in the case when the matrix A is symmetric (Hermitian) and that it can not be used with variable step size strategy, unless in the case when A is tridiagonal.

We next consider a method which is entirely based on the idea of approximating the action of each of the functions $\phi^{[i]}$ on a given state vector v. This approach is useful for implementations employing a variable step size strategy.

3 Krylov subspace approximations

Since the mid-eighties, the idea to use the Krylov subspace approximations to the action of the evolution operators is studied from many authors. A short and definitely incomplete list of publications on this topic includes [5, 6, 7, 9, 10, 23, 24]. The convergence properties of the action of the matrix exponential operator are investigated in [5, 7, 24]. Later in [9] a sharper error estimates are derived. It is also shown that, unless a good preconditioner is not available, the Krylov approximation to $e^A v$ converges faster than its corresponding approximation to the solution of the linear system $(I - A)x = v$. An approximation to the action of the first function $\phi^{[1]}$, based on the Krylov subspace approximation techniques, was considered in [24] and later studied in [9]. It was found that it obeys the same error bounds as the approximation to the matrix exponential operator. This provided the initial motivation, in [10], for developing a Rosenbrock-like exponential integrators.

The main idea of the Krylov subspace techniques is to approximately project the action of the evolution operator $\phi^{[i]}(A)$ on a state vector $v \in \mathbb{C}^d$, to a small

Krylov subspace

$$K_{\mathsf{m}} \equiv \text{span}\{v, Av, \ldots, A^{\mathsf{m}-1}v\}.$$

Usually, even for a relatively small $\mathsf{m} \ll d$, an accurate approximation can be obtained [24]. Thus the approach is to approximate the action of $\phi^{[i]}(A)$ by the action of $\phi^{[i]}$ applied to the projection of A on the smaller subspace K_{m}.

It is convenient to choose an orthogonal basis $V_{\mathsf{m}} = [v_1, v_2, \ldots, v_{\mathsf{m}}]$ of K_{m}. It can be generated by the Arnoldi algorithm, with $v_1 = v/||v||_2$ as an initial vector.

Algorithm 3.1 *(Arnoldi)*

Compute $v_1 = v/||v||_2$.
for $j = 1, 2, \ldots, m$ **do**
  for $i = 1, 2, \ldots, j$ **do**
    $h_{i,j} = (Av_j, v_i)$,
  end
  $w = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$,
  $h_{j+1,j} = ||w||_2, \quad v_{j+1} = w/h_{j+1,j}$,
end.

Alternatively, the Lanczos algorithm for generating a *biorthogonal* basis on the subspace K_{m}, can also be used (see [24]).

Let H_{m} be the $\mathsf{m} \times \mathsf{m}$ upper Hessenberg matrix consisting of the coefficients $h_{i,j}$. Since the Algorithm 3.1 is just a modified Gram-Schmidt process, the following relation holds

$$AV_{\mathsf{m}} = V_{\mathsf{m}} H_{\mathsf{m}} + h_{\mathsf{m}+1,\mathsf{m}} v_{\mathsf{m}+1} e_{\mathsf{m}}^T, \tag{3.1}$$

where e_i denotes the i^{th} unit vector in $\mathbb{R}^{\mathsf{m}}$. Using the fact that V_{m} is orthogonal from the above relation follows that $V_{\mathsf{m}}^T A V_{\mathsf{m}} = H_{\mathsf{m}}$. Therefore H_{m} represents the orthogonal projection of A to the subspace K_{m}, with respect to the basis V_{m}. Similarly $V_{\mathsf{m}} V_{\mathsf{m}}^T \phi^{[i]}(A)v$ is the projection of $\phi^{[i]}(A)v$ on K_{m}, that is the closest approximation to $\phi^{[i]}(A)v$ from K_{m}. If $\beta \equiv ||v||_2$ then $v = \beta v_1$ and since $v_1 = V_{\mathsf{m}} e_1$, we have

$$\begin{aligned} \phi^{[i]}(A)v = V_{\mathsf{m}} V_{\mathsf{m}}^T \phi^{[i]}(A)v &= \beta V_{\mathsf{m}} V_{\mathsf{m}}^T \phi^{[i]}(A) v_1 \\ &= \beta V_{\mathsf{m}} V_{\mathsf{m}}^T \phi^{[i]}(A) V_{\mathsf{m}} e_1 \ . \end{aligned} \tag{3.2}$$

From computational point of view, the above formula is not useful, since it still involves operations with the big matrix A. To avoid this, the idea is to replace the term $V_{\mathsf{m}}^T \phi^{[i]}(A) V_{\mathsf{m}}$ in (3.2) by suitable approximation. Note that $V_{\mathsf{m}}^T \phi^{[i]}(A) V_{\mathsf{m}}$ is a $\mathsf{m} \times \mathsf{m}$ matrix. By induction, from (3.1), one can prove [24, Lemma 3.1] that

$$p_{\mathsf{m}-1}(A)v = \beta V_{\mathsf{m}} p_{\mathsf{m}-1}(H_{\mathsf{m}}) e_1, \text{ for all polynomials } p_{\mathsf{m}-1} \text{ of degree } \leq \mathsf{m} - 1.$$

Therefore it is natural to approximate $V_{\mathsf{m}}^T \phi^{[i]}(A) V_{\mathsf{m}}$ by $\phi^{[i]}(H_{\mathsf{m}})$. From (3.2) we obtain

$$\phi^{[i]}(A)v \approx \beta V_{\mathsf{m}} \phi^{[i]}(H_{\mathsf{m}}) e_1. \tag{3.3}$$

The approximation (3.3) can also be derived from the standard Krylov approximation to the solution of linear system of equations [23] and the Cauchy integral formula (see [9]). The advantage of using (3.3) is that instead of working with the original large matrix A we use its orthogonal approximation H_{m}, which has much smaller dimension. The action $\phi^{[i]}(A)v$ is then computed in $\mathcal{O}(\mathsf{m}d)$ operations by using only matrix-vector multiplications between elements with the original large size d. Thus, when $\mathsf{m} \ll d$ the cost of computing the expression $\beta V_{\mathsf{m}} \phi^{[i]}(H_{\mathsf{m}}) e_1$ is usually much less then the cost needed to compute $\phi^{[i]}(A)v$.

The computation of $\phi^{[i]}(H_{\mathsf{m}})e_1$ requires $\mathcal{O}(\mathsf{m}^3)$ operations in general and only $\mathcal{O}(\mathsf{m}^2)$ if the matrix H_{m} is tridiagonal (e.g. A is symmetric). It can be done by using Chebyshev rational approximation, evaluated by partial fraction expansion (see Subsection 2.2).

The main computational cost of an exponential integrator, using Krylov subspace approximation technique, comes from the repeated application of Algorithm 3.1. At every step we need to construct several bases of Krylov subspaces with respect to one and same matrix A and different vectors v. In general Algorithm 3.1 requires $\mathcal{O}(\mathsf{m}d^2)$ operations. We mention also that it assumes exact arithmetic is used. In practice, round off errors and cancelations might cost loss of the orthogonality between the vectors v_i. A significant improvement in the performance can be achieved by using double orthogonalization [14]. In addition a convergence criterion is needed to determine the value of m that gives a sufficiently accurate approximation. Thus we conclude that the above approach is preferable in the case when the number of the needed Krylov bases can be significantly reduced. Efficient exponential integrators based on this *reduced* idea are developed in [10]. The advantage of the Krylov subspace approximation technique is that, implementations based on variable step size strategy, do not increase the total computational cost of the integrator.

4 Cauchy integral approach

The last approach for computing the functions $\phi^{[i]}$ or their action on a given vector v, which we consider, is introduced in [13] and it is based on the Cauchy integral formula

$$\phi^{[i]}(A) = \frac{1}{2\pi i}\int_{\Gamma_A} \phi^{[i]}(\lambda)(\lambda I - A)^{-1}d\lambda, \tag{4.1}$$

where Γ_A is a contour in the complex plane that encloses the eigenvalue of A, and it is also well separated from 0. It is practical to choose the contour Γ_A to be a circle centered on the real axis. In this way when A is real, based on the symmetry, one can evaluate the integral only on the upper half of the circle and then double the real part of the result. To approximate the integral in (4.1) the authors in [13] propose to use the trapezoid rule, which converges exponentially [26]. Therefore we obtain the following approximation

$$\phi^{[i]}(A) \approx \frac{1}{k}\sum_{j=1}^{k} \lambda_j \phi^{[i]}(\lambda_j)(\lambda_j I - A)^{-1}, \tag{4.2}$$

where k is the number of the equally spaced points λ_j along the contour Γ_A. Usually, values of $k = 32$ or $k = 64$, are sufficient to insure correct computations.

For problems with diagonal matrix A it is beneficial to choose the contour Γ_A to be, in addition, a circle centered at A. Thus (4.2) simply reduces to the `mean` of $\phi^{[i]}$ over the equally spaced points along Γ_A (or again just half of them for a real A). When the matrix A is non-diagonal the cost for computing an approximation to the functions $\phi^{[i]}$ increases. This is due to the number of the matrix inverses involved in (4.2). In the case of a constant step size, the total impact on the computational time is still small, since all the $\phi^{[i]}$ functions can be evaluated only once before the integration to begin. However, in the case of variable step size, direct application of (4.2) leads to significant increase in the computational work. To gain more inside how formula (4.1) can be effectively used, in the case of variable step size, we recall that the matrix $A = \gamma h L$, where $\gamma \in \mathbb{R}$, h is the step size and L is the discretized linear operator. Note that every time when we need to change the step size h, it is enough to change only the parameter γ.

The idea now is to represent $\phi^{[i]}(\gamma h L)$ in such a way that the number of the matrix inverses used in (4.1), respectively in (4.2), is independent of γ. If we choose

a suitable contour Γ, such that for all different values of γ, which appear in the integration process, it encloses the eigenvalues of γhL and $\gamma\Gamma$ is well separated from 0 then we can compute each of the functions $\phi^{[i]}$ by the following formula

$$\phi^{[i]}(A) = \phi^{[i]}(\gamma hL) = \frac{1}{2\pi i}\int_{\Gamma} \phi^{[i]}(\gamma\lambda)(\lambda I - hL)^{-1} d\lambda. \tag{4.3}$$

As before, approximating the integral in (4.3) by the trapezoid rule, we get

$$\phi^{[i]}(A) \approx \frac{1}{k}\sum_{j=1}^{k} \lambda_j \phi^{[i]}(\gamma\lambda_j)(\lambda_j I - A)^{-1}, \tag{4.4}$$

where now λ_j are the equally spaced points along the contour Γ. The above formula allows to reduce the computational work needed to evaluate the functions $\phi^{[i]}$ or their action on a given vector v, in the case when a variable step size is used. The main advantage comes form the fact that the inverse matrices in (4.4) no longer depend of γ. Thus in any case we have to compute only k (or $k/2$) matrix inverses.

In the case when the matrix L arises from a finite difference approximation to a second order partial differential operator, we can benefit from its sparse block structure. Similar to the idea presented in Subsection 2.2, in this case, we can also evaluate the action of the function $\phi^{[i]}$ to a given vector v. If the number of operations required does not exceed $\mathcal{O}(d^2)$-the cost of a matrix-vector product then we obtain a competitive method. Note that in general the matrix $\phi^{[i]}(\gamma hL)$ does not retain the sparse block structure of L. Thus what we need is an efficient method for solving special sparse block linear systems of equations. When the matrix L is symmetric (Hermitian) and positive definite the most favourable methods are *preconditioned conjugate gradient* and *multigrid methods* [12]. They are based on the idea of approximating the solution by iterative procedure and usually require $\mathcal{O}(d)$ operations. The problem with this methods is that they require a good preconditioner or a rather complex algorithm with considerable overhead to organize the computations. In addition, we note that the matrices in (4.4) are symmetric (Hermitian) only if $\lambda_j \in \mathbb{R}$.

Alternatively, direct methods for solving linear systems of equations can be used. In order to be competitive, such methods are specially design to take advantage from the sparse block structure of the coefficient matrix. In general, the structure of L, depends from the dimensionality of the problem, the type of the approximation and the boundary conditions imposed. When L is block tridiagonal, two methods are from practical importance: the Buneman variant of *cyclic reduction* [1] and the decomposition method based on the *fast Fourier transform* (FFT) [2]. Both of these methods compute the solution in $\mathcal{O}(d\log_2 n)$ operations, where n is the block size of L. Combination of the above two methods known as Fourier analysis-cyclic reduction (FACR) is proposed in [11]. The asymptotic operation count for this method is reduced to $\mathcal{O}(d\log_2\log_2 n)$ (see [25]). A restriction for all of this methods is that n should be power of two or a composite of small primes.

In [17], method for solving tridiagonal block Teoplitz linear systems of equations, which does not impose any restrictions on n is proposed. The method is based on a modified LU factorization and it is fully applicable to all the matrices in (4.4). The restriction for positive definiteness of the coefficient matrix is introduced in order to make possible, comparison between this method and some classical techniques. In [19] the same idea is generalized to the case when the coefficient matrix has pentadiagonal block circulant structure. The complexity of this methods is $\mathcal{O}(d^2)$. Significant computational savings can be achieved by using the techniques of solving linear systems with multiple right hand sides. Since the methods are based on the LU decomposition idea, we can factorize the k ($k/2$) coefficient matrices arising

from (4.4) once and for all and then use only a back-substitution formulas to find their action on a given vector v. Thus, for $k \ll d$ we obtain methods which can be implemented with variable step size, without to increase the total computational work.

5 Concluding remarks

We summarize the main advantages and disadvantages of the different methods for computing the functions $\phi^{[i]}$ or their actions on a vector, presented in this note.

- Methods based on Algorithm 2.1 are expensive and not suitable for implementations using variable step size. Their main advantage is that they do not place any restrictions on the coefficient matrix.
- Methods using Krylov subspace approximation techniques are suitable for implementations involving variable step size, but are applicable only in the case when the number of the needed Krylov bases can be significantly reduced.
- Methods based on the Cauchy integral formula are suitable for both constant and variable step size implementations. Particularly cheep methods, in the case of variable step size, can be obtained if the the coefficient matrix has a sparse block structure. Alternatively for tridiagonal matrices Algorithm 2.2 can be used.

Acknowledgments

This work has been partially supported by Norwegian Research Council through the GI4PDE project, contract number 142955/431.

References

[1] O. Buneman, *A compact non-iterative Poisson solver*, Rep. 294, Stanford University Institute for Plasma Research, 1969.

[2] B. Buzbee, G. Golub, and C. Nielson, *On direct methods for solving Poisson's equation*, SIAM J. Numer. Anal. **7** (1970), 627–656.

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

[4] P. Davies and N. Higham, *A Schur-Parlett algorithm for computing matrix functions*, SIAM J. Matrix Anal. Appl. **25(2)** (2003), 464–485.

[5] V. L. Druskin and L. A. Knizhnerman, *Error bounds in the simple Lanczos procedure for computing functions of symmetric matrices and eigenvalues*, Comput. Maths. Math. Phys. **7** (1991), 20–30.

[6] _____, *Krylov subspace approximations of eigenpairs and matrix functions in exact an computer arithmetic*, Numer. Lin. Alg. Appl. **2** (1995), 205–217.

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

[8] G. Golub and C. Van Loan, *Matrix computations*, Johns Hopkins University Press, Baltimore, 1996, 3td ed.

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

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

[11] R. Hockney, *The potential calculation and some applications*, Methods of Computational Physics **7** (1969), 136–211.

[12] C. Johnson, *Numerical solution of partial differential equations by the finite element methods*, Cambridge University press, 1987.

[13] A.-K. Kassam and L.N. Trefethen, *Fourth order time stepping for stiff pdes*, SIAM J. Sci. Comp. **to appear**.

[14] R. Lehoucq, D. Sorensen, and C. Yang, *ARPACK user's guide*, SIAM, 1998.

[15] Y. Y. Lu, *Exponentials of symmetric matrices through tridiagonal reductions*, Linear Algebra Appl. **279** (1998), 317–324.

[16] ———, *Computing a matrix function for exponential integrators*, J. Comp. and Appl. Math. **161(1)** (2003), 203–216.

[17] B. Minchev, *Some algorithms for solving special tridiagonal block teoplitz linear systems*, J. Comp. and Appl. Math **156** (2003), 179–200.

[18] ———, *Exponential integrators for semilinear problems*, Ph.D. thesis, University of Bergen, Department of Informatics, 2004.

[19] B. Minchev and I. Ivanov, *A method for solving hermitian pentadiagonal block circulant systems of linear equations*, Lect. Notes Comput. Sci., Springer **2907** (2004), 481–488.

[20] B. Minchev and William Wright, *A review of exponential integrators*, in preparation, University of Bergen, 2004.

[21] C. Moler and C. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Review **45(1)** (2003), 3–49.

[22] B.N. Parlett, *A recurrence among the elements of functions of triangular matrices*, Linear Algebra Appl. **14** (1976), 117–121.

[23] Y. Saad, *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comp. **37** (1981), 105–126.

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

[25] P. Swarztrauber, *The methods of cyclic reduction, Fourier analysis and the FACR algorithms for the discrete solution of the Poisson's equation on a rectangle*, SIAM Review **19** (1977), 490–501.

[26] L. N. Trefethen, *Spectral methods in* MATLAB, SIAM, 2000.

[27] R. S. Varga, *On higher order stable implicit methods for solving parabolic differential equations*, J. Math. and Phys **XL** (1961), 220–231.