Journal of Applied Analysis and Computation
Volume 7, Number 4, November 2017, 1402–1416
Website:http://jaac-online.com/
DOI:10.11948/2017085

APPROXIMATION OF THE LINEAR COMBINATION OF φ-FUNCTIONS USING THE BLOCK SHIFT-AND-INVERT KRYLOV SUBSPACE METHOD*

Dongping Li [3,4] and Yuhao Cong[1,2,†]

Abstract In this paper, we develop an algorithm in which the block shift-and-invert Krylov subspace method can be employed for approximating the linear combination of the matrix exponential and related exponential-type functions. Such evaluation plays a major role in a class of numerical methods known as exponential integrators. We derive a low-dimensional matrix exponential to approximate the objective function based on the block shift-and-invert Krylov subspace methods. We obtain the error expansion of the approximation, and show that the variants of its first term can be used as reliable a posteriori error estimates and correctors. Numerical experiments illustrate that the error estimates are efficient and the proposed algorithm is worthy of further study.

Keywords Matrix exponential, exponential integrators, block shift-and-invert Krylov subspace, a posteriori error estimates.

MSC(2010) 65L05, 65F10, 65F30.

1. Introduction

In this paper, we are concerned with numerical methods for approximating the linear combination of the action of exponential-type functions of the type

$$\varphi_0(A)b_0 + \varphi_1(A)b_1 + \varphi_2(A)b_2 + \cdots + \varphi_p(A)b_p, \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$ usually arises from the discretization of unbounded sectorial operators and $b_i \in \mathbb{R}^n$ for $0 \le i \le p$. These φ_j-functions are defined as

$$\varphi_0(z) = e^z, \quad \varphi_j(z) = \int_0^1 e^{(1-\theta)z}\frac{\theta^{j-1}}{(j-1)!}d\theta, \quad j \ge 1, \tag{1.2}$$

†the corresponding author. Email address: yhcong@shu.edu.cn (Y. Cong)
[1]Department of Mathematics, Shanghai University, Shanghai, 200444, China
[2]Department of Mathematics, Shanghai Customs College, Shanghai, 201204, China
[3]Department of Mathematics, Changchun Normal University, Changchun, 200234, China
[4]Mathematics and Science College, Shanghai Normal University, Shanghai, 130032, China
*The article was supported by National Natural Science Foundation of China (11471217) and Natural Science Foundation of Changchun Normal University.

which satisfy the recursive relation

$$\varphi_j(z) = z\varphi_{j+1}(z) + \frac{1}{j!}, \quad j \geq 0. \tag{1.3}$$

Such problems play a key role in a class of numerical methods called exponential integrators for solving the autonomous semi-linear problems of the form

$$y'(t) = Ay(t) + N(y(t)), \quad y(t_0) = y_0. \tag{1.4}$$

According to the variation-of-constants formula, the exact solution of (1.4) at time

$$t_{n+1} = t_n + h_n, \quad n = 0, 1, \cdots,$$

can be denoted as

$$y(t_n + h_n) = e^{Ah_n} y(t_n) + h_n \int_0^1 e^{Ah_n(1-s)} N\big(y(t_n + sh_n)\big) ds. \tag{1.5}$$

Then, the exponential integrators can be constructed by approximating the nonlinear terms $N\big(y(t_n + sh_n)\big)$ in (1.5) by an interpolating polynomial, which involves the linear combination of matrix-vector products of type (1.1). A full overview of this topic can be found in [19, 24].

As mentioned above, the efficient computation of the matrix exponential and related functions of matrices acting on certain vectors is the core of exponential integrators. In the past few years, many methods have been proposed for computing such functions of matrices; see, e.g., [14, 33]. For a comprehensive review of these techniques, we refer to the review [25] and the recent book [20].

In many applications, the matrix A is large and sparse so that it is prohibitive to directly compute $\varphi_j(A)$ and then form the product with vector b_j. In this case Krylov subspace methods are often used and significantly reduce computational cost; see, e.g., [5, 8, 18, 29, 31] and the references given therein. In this setting, our main interest is the case of ill conditioned matrices A with a large norm typically arising from the space discretization of stiff time-dependent parabolic PDEs. The widely used standard Krylov subspace methods often turn out to be inefficient for this type of matrices because of slow convergence and large memory requirement. Two popular strategies are presented to overcome this disadvantage. The first one is the restarted Krylov subspace methods [10, 12, 36]. However, these methods may slow down the convergence speed and even diverge in some cases. The other approach is to apply rational Krylov subspace methods to approximate the object functions; see, e.g., [11, 16, 23, 27]. One such method is to use the well-known shift-and-invert (SI) Krylov subspace methods, which has been investigated independently in [27] and [11] for the computation of the matrix exponential. Further application and analysis can be found in [6, 9, 28, 38]. It has been proved in [13, 15] that the SI Krylov subspace methods as a preconditioning technique for φ_k-functions ($k \geq 0$) can obtain grid-independent convergence under some reasonable assumptions and accelerate the convergence of the standard Krylov subspace method.

The aim of this paper is to describe how a block version of Eshof and Hochbruck [11] can be employed for the linear combinations of the action of φ-functions of the form (1.1). We also provide some reliable a posteriori estimates and effective corrected schemes from a practical viewpoint. We are not here to thoroughly explore all

of the numerical analysis questions associated with the proposed algorithm. However, typical numerical experiments exhibit that the proposed algorithm has faster convergence speed and the error estimates can capture correctly the characteristics of the true error, meaning it is worthy of further study.

This paper is organized as follows. In Section 2 we first review some useful results to be used, and then introduce the block SI Krylov subspace algorithm. Section 3 present the exact error expansion, which is then exploited to produce reliable a posteriori error estimates and some slightly more accurate corrected schemes. In Section 4 we present some numerical experiments to demonstrate the effectiveness of the a posteriori error estimates and illustrate the benefits of our algorithm. Finally, some concluding remarks are given in Section 5.

Throughout the paper, e_i denotes the i-th coordinate vector with appropriate size. Let I be the identity and 0 be the zero matrix or vector whose dimension are clear from the context. Let E_i be an $mp\times p$ matrix which consists of the $(i-1)p+1$-th to the ip-th columns of the $mp \times mp$ identity matrix. $\|\cdot\|$ always denotes the vector 2-norm or its induced matrix norm. Standard MATLAB notations are used whenever necessary.

2. The block shift-and-invert (SI) Krylov subspace approximation to (1.1)

We start by recalling a useful result to be used in this section. In [2], Al-Mohy and Higham observed that the representation (1.1) can be represented exactly in terms of a single exponential of an $(n+p)\times(n+p)$ matrix, i.e.,

$$\sum_{k=0}^{p}\varphi_k(A)b_k = [I_n,\ 0]e^{\tilde{A}}\begin{pmatrix} b_0 \\ e_p \end{pmatrix}, \tag{2.1}$$

where $\tilde{A} = \begin{pmatrix} A & \mathcal{B} \\ 0 & J \end{pmatrix} \in \mathbb{C}^{(n+p)\times(n+p)}$, $\mathcal{B} = [b_p, b_{p-1}, \cdots, b_1] \in \mathbb{C}^{n\times p}$, and $J = \begin{pmatrix} 0 & I_{p-1} \\ 0 & 0 \end{pmatrix} \in \mathbb{C}^{p\times p}$. Thus the evaluating of (1.1) can be replaced by approximating the action of the matrix exponential on a vector. However, the matrix $\tilde{A}$ is usually large and sparse such that the explicit computation of this matrix exponential is still not a trivial task. This paper tries to employ the block SI Krylov subspace algorithm to reduce the dimension of matrix A and solve the reduced approximation to (2.1) exactly.

The following result proved in [32] establish a useful relation between the representation (1.1) and the exact solution of linear differential equations with polynomial inhomogeneity, which also provides the platform for developing the block SI Krylov subspace algorithm.

Lemma 2.1. *The non-autonomous linear initial value problem*

$$y'(t) = Ay(t) + \sum_{j=0}^{p-1}\frac{t^j}{j!}b_{j+1}, \quad y(0) = b_0, \tag{2.2}$$

has the exact solution

$$y(t) = \varphi_0(tA)b_0 + \sum_{j=0}^{p-1} t^{j+1}\varphi_{j+1}(tA)b_{j+1}, \tag{2.3}$$

where the functions φ_i are defined in (1.2).

On taking $t = 1$, the representation (2.3) reduces to (1.1). This means one can obtain an approximation of (1.1) by solving the IVP (2.2). By using the equivalence relation between computing the representation (1.1) and solving the above systems of ODEs, we can derive a low-dimensional matrix exponential to approximate the matrix functions (1.1) by projecting the solution of the IVP (2.2) onto a block Krylov subspace generated by a SI matrix. The idea of using Krylov subspace techniques to reduce large-scale systems of ODEs is not new but has been used often, see, e.g., [3, 4, 7, 22, 30]. In [6], Botchev has applied the block SI Krylov subspace methods to solve systems of ODEs of the form

$$y' = Ay + g(t). \tag{2.4}$$

The IVP (2.2), as a special form of (2.4), can be solved naturally by the method. Here, we wish to fully take advantage of the structure of IVP (2.2) and obtain a numerical solution to (1.1) based on block SI Krylov subspace method such that some interesting properties can be maintained.

An IVP with nonzero initial vector can always be converted to one with zero initial vector by replacing the state vector $y(t)$ by $y(t) - y_0$. For convenience, we consider the non-autonomous linear IVP (2.2) with zero initial vector.

Let $B = [b_1, b_2, \cdots, b_p]$ and $P(t) = [1, t, \frac{t^2}{2!}, \cdots, \frac{t^{p-1}}{(p-1)!}]^T$, then IVP (2.2) with zero initial vector can be rewritten in the following form

$$y'(t) = Ay(t) + BP(t),\ y(0) = 0. \tag{2.5}$$

Premultiply IVP (2.5) by matrix $\hat{A} = (I + \gamma A)^{-1}$ to get

$$\hat{A}y'(t) = \frac{(I - \hat{A})}{\gamma}y(t) + \hat{A}BP(t),\ y(0) = 0, \tag{2.6}$$

where $\gamma \in \mathbb{C}\backslash\{z|z\lambda + 1 \neq 0, \lambda \in \Lambda(A)\}$ is a suitably chosen nonzero parameter.

Let $B = V_1 R$ be the economy-size QR decomposition of B, thus V_1 is an $n \times p$ column orthogonal matrix, and R is a $p \times p$ upper triangular matrix. Applying m steps of the modified block Arnoldi process [35, Algorithm 6.23] to $\mathcal{K}_m(\hat{A}, V_1)$, we obtain the recursion relation

$$\hat{A}\mathcal{V}_m = \mathcal{V}_m\mathcal{H}_m + V_{m+1}H_{m+1,m}E_m^T = \mathcal{V}_{m+1}\bar{\mathcal{H}}_m, \tag{2.7}$$

where the columns of $\mathcal{V}_m$ form an orthonormal basis of $\mathcal{K}_m(\hat{A}, V_1)$, $\bar{\mathcal{H}}_m = (H_{ij})$ is a $(m+1)p \times mp$ block upper-Hessenberg matrix with $p \times p$ blocks H_{ij} as its elements, and $\mathcal{H}_m$ is obtained by deleting the last row block of $\bar{\mathcal{H}}_m$. Algorithm 1 is a brief description of the block Arnoldi process to $\mathcal{K}_m(\hat{A}, V_1)$. Once A is symmetric, this process can be replaced by the symmetric block Lanczos algorithm (see Algorithm 2).

Algorithm 1 Block Arnoldi process to $\mathcal{K}_m(\hat{A}, V_1)$

Choose a $n \times p$ unitary matrix V_1
for $j = 1, 2, \cdots, m$, **do**
  Solve the linear system $(I + \gamma A)W_j = V_j$
  for $i = 1, 2, \cdots, j$, **do**
    $H_{ij} = V_i^T W_j$
    $W_j = W_j - V_i H_{ij}$
  end for
  Compute the QR decomposition $W_j = V_{j+1} H_{j+1,j}$
end for

Algorithm 2 Symmetric block Lanczos process to $\mathcal{K}_m(\hat{A}, V_1)$

Choose a $n \times p$ unitary matrix V_1
for $j = 1, 2, \cdots, m$, **do**
  Solve the linear system $(I + \gamma A)W_j = V_j$
  $H_{jj} = V_j^T W_j$
  $W_j = W_j - V_{j-1} H_{j,j-1} - V_j H_{jj}$
  Compute the QR decomposition $W_j = V_{j+1} H_{j+1,j}$
end for

Provided $\mathcal{H}_m$ is invertible and let $\mathcal{T}_m = \frac{(\mathcal{H}_m^{-1} - I)}{\gamma}$, the relation (2.7) can be rewritten as

$$\mathcal{V}_m \mathcal{T}_m = A\mathcal{V}_m + \frac{(I + \gamma A)}{\gamma} V_{m+1} H_{m+1,m} E_m^T \mathcal{H}_m^{-1}. \tag{2.8}$$

We project the solution of systems (2.6) onto the block Krylov subspace $\mathcal{K}_m(\hat{A}, V_1)$. Then, $y(t)$ can be approximated by

$$y(t) \approx \mathcal{V}_m u(t), \tag{2.9}$$

where $u(t) \in \mathbb{R}^{mp}$ is a low-dimensional vector function to be determined.

By substituting the approximation for $y(t)$ directly into equation (2.6), the residual associated with this approximation is

$$r_m(t) = \hat{A}\mathcal{V}_m u'(t) - \frac{(I - \hat{A})}{\gamma} \mathcal{V}_m u(t) - \hat{A}BP(t), \quad u(0) = 0. \tag{2.10}$$

By analogy with the idea of weighted residual, imposing the orthogonality condition $\mathcal{V}_m^T r_m = 0$ and left-multiplying equation (2.10) by $\mathcal{V}_m^T$, we arrive at the reduced-order system

$$\mathcal{H}_m u'(t) = \frac{(I - \mathcal{H}_m)}{\gamma} u(t) + \mathcal{H}_m E_1 RP(t), \ u(0) = 0, \tag{2.11}$$

which may be written in the simplified form

$$u'(t) = \mathcal{T}_m u(t) + E_1 RP(t), \ u(0) = 0. \tag{2.12}$$

It is obvious that the reduced systems (2.12) preserve the original structure of the systems (2.5). Thus its exact solution can be similarly represented by a linear

combination of the action of φ-function on vectors, by identity (2.1), which in turn can be denoted as

$$u(t) = [I_{mp},\ 0]e^{t\tilde{\mathcal{T}}_m}e_{p(m+1)}, \tag{2.13}$$

where

$$\tilde{\mathcal{T}}_m = \begin{pmatrix} \mathcal{T}_m & \mathcal{W} \\ 0 & J \end{pmatrix}, \quad J = \begin{pmatrix} 0 & I_{p-1} \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{p\times p},$$

and $\mathcal{W}$ is a flipped matrix of E_1R in the left-right direction. Then we arrive at

$$y_m(t) = \mathcal{V}_m u(t) = [\mathcal{V}_m,\ 0]e^{t\tilde{\mathcal{T}}_m}e_{p(m+1)}. \tag{2.14}$$

The original problem—which is equivalent to computing $[I_n,\ 0]e^{\tilde{A}}\begin{pmatrix} b_0 \\ e_p \end{pmatrix}$ with $\tilde{A} \in R^{(n+p)\times(n+p)}$—has been replaced by evaluating $[\mathcal{V}_m,\ 0]e^{\tilde{\mathcal{T}}_m}e_{p(m+1)}$ with $\tilde{\mathcal{T}}_m \in R^{(m+1)p\times(m+1)p}$ and $mp \ll n$. For the reduced problem, the matrix exponential can be solved by a direct method such as the scaling-and-squaring algorithms [1, 17]. The above discussion is summarized below as Algorithm 3.

Algorithm 3 The block SI Krylov subspace algorithm for the matrix function (2.3)

1: Given: A, $B = [b_0, b_1, \cdots, b_p]$ and t
2: $b_1 := b_1 + Ab_0$
3: Compute the QR decomposition: $B(:, 2 : p+1) = V_1R$
4: **for** $m = 1, 2, \cdots$ until convergence **do**
5: Compute Arnoldi/Lanczos decomposition:
$(I+\gamma A)^{-1}\mathcal{V}_m = \mathcal{V}_m\mathcal{H}_m + V_{m+1}H_{m+1,m}E_m^T$ of $\mathcal{K}_m((I+\gamma A)^{-1}, V_1)$
6: Compute $\mathcal{T}_m = \frac{(\mathcal{H}_m^{-1}-I)}{\gamma}$
7: Form $\tilde{\mathcal{T}}_m$: $\mathcal{W}(:, i) = (E_1R)(:, p-i+1), \quad i = 1, 2, \cdots p$, and

$$\tilde{\mathcal{T}}_m := \begin{pmatrix} \mathcal{T}_m & \mathcal{W} \\ 0 & J_p \end{pmatrix}$$

8: Compute $u(t) = [I_m,\ 0]e^{t\tilde{\mathcal{T}}_m}e_{p(m+1)}$
9: Compute $y_m(t) = \mathcal{V}_m u(t) + b_0$
10: **end for**

3. A posteriori error estimates and corrected schemes

In actual calculation, it is important to provide a possible simple and reliable stopping criterion to determine whether or not the approximation y_m has the required accuracy. To this purpose we first consider the explicit expression for the error.

Premultiply equation (2.12) by $\mathcal{V}_m$ to get

$$y_m'(t) = \mathcal{V}_m\mathcal{T}_m\mathcal{V}_m^T y_m(t) + V_1RP(t),\ y_m(0) = 0. \tag{3.1}$$

Using the relation (2.8), we see that our approximation satisfies

$$y'_m(t) = Ay_m(t) + \frac{(I+\gamma A)}{\gamma} V_{m+1} H_{m+1,m} E_m^T \mathcal{H}_m^{-1} \mathcal{V}_m^T y_m(t) + V_1 RP(t),\ \ y_m(0) = 0. \tag{3.2}$$

Then, the error $\varepsilon_m(t) := y(t) - y_m(t)$ as a function of t satisfies the initial value problem

$$\varepsilon'_m(t) = A\varepsilon_m(t) - \frac{(I+\gamma A)}{\gamma} V_{m+1} H_{m+1,m} E_m^T \mathcal{H}_m^{-1} \mathcal{V}_m^T y_m(t),\ \ \varepsilon_m(0) = 0. \tag{3.3}$$

By using the variation-of-constants formula, the exact solution of (3.3) can be explicitly denoted as

$$\varepsilon_m(t) = -\frac{(I+\gamma A)}{\gamma} \int_0^t e^{(t-s)A} V_{m+1} H_{m+1,m} E_m^T \mathcal{H}_m^{-1} \mathcal{V}_m^T y_m(s) ds. \tag{3.4}$$

The above derivation is exactly the same as in [5, 6]. However, the evaluation of (3.4) is difficult to measure in practice as it requires to construct a quadrature approximation to the integral. The following theorem presents an alternative to the error by reformulating the expression (3.4) as an expansion in terms of products between φ-functions of matrices and vectors.

Theorem 3.1. *The error $\varepsilon_m(t)$ satisfies the following expansion:*

$$\varepsilon_m(t) = -\frac{t(I+\gamma A)}{\gamma} \sum_{j=0}^{\infty} (tA)^j V_{m+1} H_{m+1,m} E_m^T (\gamma \mathcal{T}_m + I)[I_{mp},\ 0] \varphi_{j+1}(t\tilde{\mathcal{T}}_m) e_{p(m+1)}. \tag{3.5}$$

Proof. Inserting the expression (2.14) and the Taylor expansion of $e^{(t-s)A}$ into (3.4), we obtain

$$\begin{aligned}
\varepsilon_m(t) &= -\frac{(I+\gamma A)}{\gamma} \int_0^t \sum_{i=1}^{\infty} \frac{(t-s)^i A^i}{i!} V_{m+1} H_{m+1,m} E_m^T (\gamma \mathcal{T}_m + I)[I_{mp},\ 0] e^{s\tilde{\mathcal{T}}_m} e_{p(m+1)} ds \\
&= -\frac{(I+\gamma A)}{\gamma} \sum_{i=1}^{\infty} A^i V_{m+1} H_{m+1,m} E_m^T (\gamma \mathcal{T}_m + I)[I_{mp},\ 0] \int_0^t \frac{(t-s)^i}{i!} e^{s\tilde{\mathcal{T}}_m} ds \cdot e_{p(m+1)}.
\end{aligned}$$

Since

$$\int_0^t \frac{(t-s)^i}{i!} e^{s\tilde{\mathcal{T}}_m} ds = t^{i+1} \int_0^1 \frac{\theta^i}{i!} e^{t(1-\theta)\tilde{\mathcal{T}}_m} d\theta,$$

we arrive at the conclusion. □

In fact, many approximations for matrix functions based on Krylov subspace methods have the similar error expansion of form (3.5), including the Arnoldi/Lanczos approximation to $\varphi_p(A)b$, $p \geq 0$, the SI Krylov subspace method to $e^A b$ and the restarted Krylov subspace approximation to $f(A)b$ and so on, see, e.g., [10,11,31,34] and the references given therein. Following the classical proposal in [34], see also [21] for further theoretical analysis, a practical stopping criterion without involving any multiplications with the matrix A is presented as follows

$$\varepsilon_m^1 = \frac{t}{\gamma} \parallel V_{m+1} H_{m+1,m} E_m^T (\gamma \mathcal{T}_m + I)[I_{mp},\ 0] \varphi_1(t\tilde{\mathcal{T}}_m) e_{p(m+1)} \parallel. \tag{3.6}$$

Substituting $\varphi_1(t\tilde{\mathcal{T}}_m)$ by $e^{t\tilde{\mathcal{T}}_m}$, we may obtain a slightly rough estimate without having to compute an extra φ_1-function

$$\varepsilon_m^2 = \frac{t}{\gamma} \parallel V_{m+1}H_{m+1,m}E_m^T(\gamma\mathcal{T}_m + I)[I_{mp},\ 0]e^{t\tilde{\mathcal{T}}_m}e_{p(m+1)} \parallel . \tag{3.7}$$

These error estimates can be embedded into Algorithm 3 to terminate Arnoldi/Lanczos process once the required accuracy has been achieved. We have tested the performance of them using a suite of test matrices. Numerical results show that the two a posteriori estimates follow the actual error closely. For the sake of brevity, we only choose two frequently used matrices to discuss in Section 4.

We also use the first term of the error expansion (3.5) or its variants as a corrector. Typical corrected schemes include

$$y_m^1(t) = y_m(t) - \frac{t(I+\gamma A)}{\gamma}V_{m+1}H_{m+1,m}E_m^T(\gamma\mathcal{T}_m + I)[I_{mp},\ 0]\varphi_1(t\tilde{\mathcal{T}}_m)e_{p(m+1)}, \tag{3.8}$$

$$y_m^2(t) = y_m(t) - \frac{t}{\gamma}V_{m+1}H_{m+1,m}E_m^T(\gamma\mathcal{T}_m + I)[I_{mp},\ 0]\varphi_1(t\tilde{\mathcal{T}}_m)e_{p(m+1)} \tag{3.9}$$

and

$$y_m^3(t) = y_m(t) - \frac{t}{\gamma}V_{m+1}H_{m+1,m}E_m^T[I_{mp},\ 0]\varphi_1(t\tilde{\mathcal{T}}_m)e_{p(m+1)}. \tag{3.10}$$

These corrected schemes provide the solutions in an enlarged block Krylov subspace of dimension $(m+1)p$ by involving the information of V_{m+1} and can be expected to achieve higher precision.

Let

$$\tilde{V}_{m+1} = (I+\gamma A)V_{m+1}, \quad T_{m+1}^1 = \begin{pmatrix} \tilde{\mathcal{T}}_m & 0 \\ -\frac{1}{\gamma}H_{m+1,m}E_m^T(\gamma\mathcal{T}_m + I)[I_{mp},\ 0] & 0 \end{pmatrix},$$

and

$$T_{m+1}^2 = \begin{pmatrix} \tilde{\mathcal{T}}_m & 0 \\ -\frac{1}{\gamma}H_{m+1,m}E_m^T[I_{mp},\ 0] & 0 \end{pmatrix}.$$

Then, a direct calculation shows that these corrected schemes can be rewritten in the following condensed form

$$y_m^1(t) = [\mathcal{V}_m, \tilde{V}_{m+1}]\begin{pmatrix} I_{mp} & 0 & 0 \\ 0 & 0 & I_p \end{pmatrix} e^{tT_{m+1}^1}e_{p(m+1)}, \tag{3.11}$$

$$y_m^2(t) = \mathcal{V}_{m+1}\begin{pmatrix} I_{mp} & 0 & 0 \\ 0 & 0 & I_p \end{pmatrix} e^{tT_{m+1}^1}e_{p(m+1)} \tag{3.12}$$

and

$$y_m^3(t) = \mathcal{V}_{m+1}\begin{pmatrix} I_{mp} & 0 & 0 \\ 0 & 0 & I_p \end{pmatrix} e^{tT_{m+1}^2}e_{p(m+1)}, \tag{3.13}$$

respectively. The above corrected schemes avoid involving evaluate the φ_1-function and can be measured directly by MATLAB's expm.

4. Numerical experiments

In this section we present several numerical experiments to demonstrate error bounds derived and show the efficiency of our algorithm over two state-of-the-art numerical algorithms. All experiments are carried out in MATLAB R2012a on a laptop with 2.6 GHz Intel Core i5 processor and RAM 6 GB.

In the first experiment, we shall compare the actual errors of Algorithm 3 with the a posteriori error estimates and test the efficiency of the corrected schemes. In the other two experiments, we compare our code with two existing codes so-called phiv and phipm on various large sparse matrices. The first code phiv is from EXPOKIT [31], which evaluates $\varphi_0(tA)b_0 + \varphi_1(tA)b_1$ using Krylov subspace projection techniques. The codes are directly available from http://www.maths.uq.edu.au/expokit/. The second code phipm of Niesen and Wright [29] computes the linear combination of form (1.1), which is available from http://www1.maths.leeds.ac.uk/~jitse/software.html. We run both codes with their default parameters and the variable convergence tolerance in our experiments. As was done in [11], we pick the shift $\gamma = -0.1$ in the block SI Arnoldi/Lanczos algorithm. The linear systems arising in the block SI Arnoldi/Lanczos process can either be solved directly or with a preconditioned iterative method, e.g., see [11]. Here we use a sparse LU factorization to solve the linear systems. In most cases, we compare Algorithm 3 with phipm.

We make use of the relative error

$$Error = \frac{\|y - y_m\|_2}{\|y\|_2} \tag{4.1}$$

to measure the stopping criteria of all numerical methods, where y_m and y are the approximation and the "exact" solution, respectively. Unless otherwise stated, the "exact" solutions are obtained by using the MATLAB's build-in functions ode45 or ode15s with small relative and absolute tolerances to compute the corresponding systems of ODEs. For a small or medium sized matrix T, we use MATLAB's build-in function expm to compute the matrix exponential e^T. The code expm is an implementation of the scaling-and-squaring method [17].

Experiment 1. This experiment is a variation of one from Jia and Lv [21]. There are two test matrices in the experiment. The first test matrix is a diagonal matrix of size 1001 taken with equally spaced diagonal entries in the interval [-1, 0]. The second test matrix comes from the seven-point stencil finite difference method discretization of a 3D convection-diffusion problem, which can be represented as the Kronecker product form

$$A = I_n \otimes [I_n \otimes C_1] + [B \otimes I_n + I_n \otimes C_2] \otimes I_n$$

of dimension $N = n^3$. Here,

$$B = tridiag(1, -2, 1),\ C_j = tridiag(1 + \mu_j, -2, 1 - \mu_j),\ j = 1, 2,$$

where $\mu_j = \tau_j h/2$. This nonsymmetric matrix is a popular test matrix since it has known eigenvalues. For a detailed description of the spectral properties, we refer to [26]. As in [26], we choose $h = 1/16$ and $\tau_1 = 96$, $\tau_2 = 128$, which leads to $N = 3375$ and $\mu_1 = 3$, $\mu_2 = 4$.

We want to compute $\varphi_0(tA)b_0 + \varphi_1(tA)b_1 + \cdots + \varphi_5(tA)b_5$ with $t = 1, 400$ for the first test matrix and $t = 0.1, 20$ for the second one, where $b_i, i = 0, 1, \cdots, 5$ are generated randomly by MATLAB function rand for each matrix. We define relative a posterior error estimates as

$$\varepsilon_m^1 := \frac{\varepsilon_m^1}{\|y\|_2} \tag{4.2}$$

and

$$\varepsilon_m^2 := \frac{\varepsilon_m^2}{\|y\|_2}, \tag{4.3}$$

and compare them with the true relative error (4.1). Figures 1-4 report the numerical results.

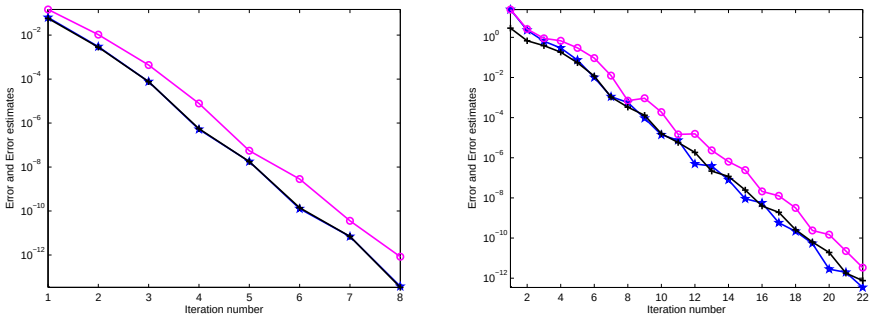

Figure 1. Experiment 1: The actual relative error (+) and the relative error estimates (3.6) ($\star$) and (3.7) (o) versus iteration steps for the first test matrix. Left picture: t=1 and right picture: t=400.

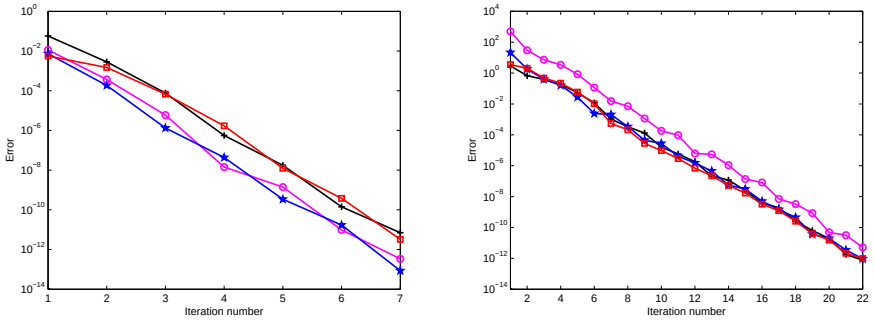

Figure 2. Experiment 1: The relative errors of standard method (+) and corrected methods (3.11) (o), (3.12) ($\star$) and (3.13) ($\square$) versus iteration steps for the first test matrix. Left picture: $t = 1$ and right picture: $t = 400$.

Figure 1 plots the curves of the true relative error of Algorithm 3 and the two relative error estimates (4.2) and (4.3) for the first test matrix. The behaviour of the corrected schemes is plotted in Figure 2. Similar results for the second test matrix are shown in Figure 3 and Figure 4. From these figures we see that Algorithm 3 achieve high precision of about 10^{-12} in all cases. Both error estimates turn out to be effective in all cases, which follow well the behavior of the actual error.

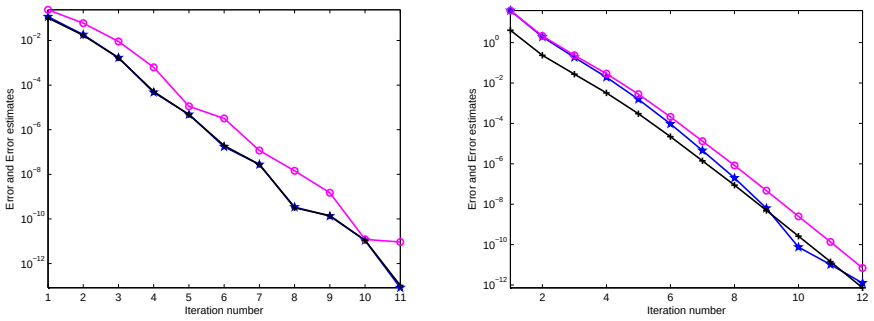

Figure 3. Experiment 1: The actual relative error (+) and the relative error estimates (3.6) ($\star$) and (3.7) (o) versus iteration steps for the second test matrix. Left picture: $t = 0.1$ and right picture: $t = 20$.

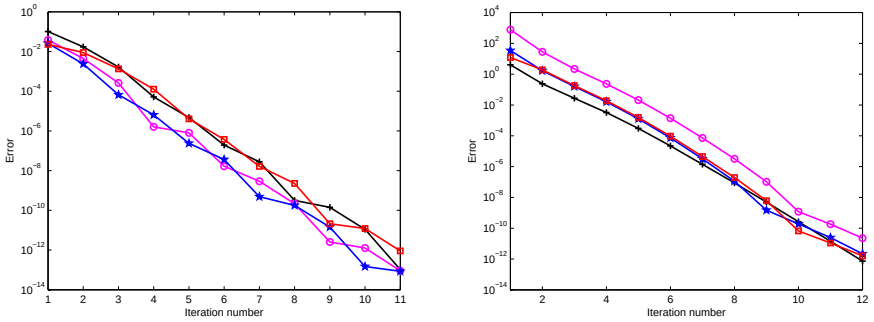

Figure 4. Experiment 1: The relative errors of standard method (+) and corrected methods (3.11) (o), (3.12) ($\star$) and (3.13) ($\square$) versus iteration steps for the second test matrix. Left picture: $t = 0.1$ and right picture: $t = 20$.

Particularly, the error estimate (3.6) is sharper than (3.7) and is indistinguishable from the true error for the two test matrices with small t.

In most cases, we observe that the corrected schemes are more accurate than the standard one. Furthermore, we notice that when $t = 1$ for the first test matrix and $t = 0.1$ for the second, the corrected schemes (3.11) and (3.12) are more accurate than (3.13). However, as t increases, the scheme (3.11) deteriorates, which can be attributed to this representation involving tA.

The following experiments try to show competitiveness of Algorithm 3 over two popular algorithms for computing (1.1). For each problem, the efficiency of an algorithm is measured in terms of CPU time (CPU) and the relative error (Error). The CPU time is computed by MATLAB functions tic and toc. Each test has been run three times to use the minimum speedup.

Experiment 2. In this experiment, we compare the performance of Algorithm 3 and phipm by evaluating $\varphi_0(A)b_0 + \varphi_1(A)b_1 + \cdots + \varphi_p(A)b_p$ for $p = 5, 10$. The experiment is performed on four different test matrices with order 10,000. For each matrix and p, the vectors b_i, $i = 0, 1, \cdots, p$, with elements from the uniform [0,1] distribution and generated by code rand. The first three matrices are generated by MATLAB build-in functions wilkinson and gallery. The first test ma-

trix is -wilknson(10000), which is a symmetric, tridiagonal matrix with Wilkinson's eigenvalues. The second matrix is generated by gallery('lesp',10000). It returns an unsymmetric, tridiagonal matrix with real, negative and sensitive eigenvalues. The third matrix is constructed by -2500×gallery('poisson',99) in MATLAB, which is a block tridiagonal matrix and arises from a multiple of the standard finite difference discretization of the 2D Laplacian. The final matrix is taken from [5, Experiment 6.2], which is the negative of the standard 5-point discretization of a 2D convection-diffusion operator with $Pe = 100$. It is an unsymmetric matrix with $49,600$ nonzero elements.

Table 1 lists the numerical results. We see from Table 1 that Algorithm 3 performs much better than phipm. For $p = 5$, the two methods deliver required accuracy, but Algorithm 3 outperforms phipm in terms of the CPU time. While for $p = 10$, phipm fails to convergence (f.c), but our algorithm succeeds. The table also shows that the increase of p has no significant effect on the CPU time of Algorithm 3.

Table 1. The CPU and the relative errors of phipm and Algorithm 3 on four matrices.

p	Algorithm	-wilknson(10000)		gallery('lesp',10000)		-2500gallery('poisson',99)		con-diff-pe100	
		Error	Time	Error	Time	Error	Time	Error	Time
5	Algorithm 3	7.47e-12	0.27	2.26e-11	0.16	1.02e-11	0.18	3.92e-13	0.65
	phipm	8.11e-12	0.28	1.19e-11	7.95	1.80e-11	0.69	7.85e-12	1.15
10	Algorithm 3	9.60e-11	0.42	1.83e-11	0.26	5.27e-12	0.28	9.34e-13	1.33
	phipm	f.c		f.c		f.c		f.c	

Experiment 3. This experiment uses essentially the same tests as [37]. There are two symmetric positive test matrices, which are directly available from the University of Florida Sparse Matrix Collection. The matrices and problem details are

- The first matrix, 1138bus arising from power system networks, is of order $n = 1138$ with $nnz = 2596$ nonzero elements.
- The second matrix, PresPoisson arising from computational fluid dynamics problems, is of order $n = 14822$ with $nnz = 715804$ nonzero elements.

We evaluate $\varphi_0(-tA)b_0 + \varphi_1(-tA)b_1 + \cdots + \varphi_p(-tA)b_p$ with $p = 1, 5$ at each t =1,10,100,1000 for the first matrix, and t =100,1000,10000,100000 for the second one. As in Experiments 1 and 2, for each test matrix A and p, vectors b_i, $i = 0, 1, \cdots, p$ are randomly generated by MATLAB build-in function rand.

The codes adopted are phiv, phipm and Algorithm 3 for $p = 1$ and phipm and Algorithm 3 for $p = 5$. For the second matrix at $t = 100000$, the MATLAB build-in functions ode45 and ode15s are infeasible to compute the corresponding ODEs. As a compromise, we regard the approximations obtained by phipm with a small tolerance as the "exact" solutions. Tables 2 and 3 report the numerical results of the test runs. The results in Tables 2 and 3 show that Algorithm 3 is the fastest, in some cases by a considering margin, especially when t is large. Moreover, one observes that the CPU time for Algorithm 3 has little changes as t increase, but the CPU time for the other algorithms increase dramatically as t become large.

Table 2. CPU and the relative errors of phiv, phipm and Algorithm 3 for $1138bus$.

p	Methods	t=1		t=10		t=100		t=1000	
		Error	Time	Error	Time	Error	Time	Error	Time
1	phiv	2.14e-13	0.44	1.49e-13	2.82	4.49e-12	14.60	3.92e-10	64.20
	phipm	2.15e-13	0.19	1.58e-13	0.81	3.54e-12	3.02	1.36e-10	13.65
	Algorithm 3	3.09e-13	0.07	7.14e-13	0.07	9.32e-12	0.06	1.41e-10	0.07
5	phipm	5.22e-12	0.18	2.63e-12	0.77	4.24e-11	6.76	5.94e-10	75.19
	Algorithm 3	3.35e-12	0.16	4.60e-12	0.12	1.87e-11	0.08	9.29e-10	0.07

Table 3. CPU and the relative errors of phiv, phipm and Algorithm 3 for $PresPoisson$.

p	Methods	t=100		t=1000		t=10000		t=100000	
		Error	Time	Error	Time	Error	Time	Error	Time
1	phiv	9.75e-14	1.37	2.37e-14	6.81	4.96e-13	46.37	3.11e-12	304.90
	phipm	1.74e-14	0.72	5.95e-13	3.15	1.20e-13	15.65	4.80e-12	76.17
	Algorithm 3	7.55e-14	0.86	2.92e-13	0.83	9.86.1e-13	0.86	3.92e-12	0.62
5	phipm	8.69e-14	0.70	5.95e-13	3.19	8.55e-12	14.42	4.63e-12	79.40
	Algorithm 3	8.83e-14	0.85	2.92e-13	0.80	7.34e-12	0.83	5.08e-12	0.61

5. Conclusion

The aim of this paper is to apply the block SI Krylov subspace methods to the linear combination of φ-functions acting on vectors. We have proposed two practical a posteriori error estimates and established three compact corrected schemes. We have numerically confirmed the effectiveness of the numerical methods and error estimates.

For future work, we plan to present a comprehensive convergence and stability analysis of algorithms presented. A theoretical interpretation of two a posteriori error estimates should be further taken into account. Another interesting question is to develop restarted iterative methods based on residual.

References

[1] A. Al-Mohy and N. J. Higham, *A new scaling and modified squaring algorithm for matrix functions*, SIAM J. Matrix Anal. Appl., 2009, 31(3), 970–989.

[2] A. Al-Mohy and N. J. Higham, *Computing the action of the matrix exponential, with an application to exponential integrators*, SIAM J. Sci. Comput., 2011, 33(2), 488–511.

[3] A. C. Antoulas and D. C. Sorensen, *Approximation of large-scale dynamical systems: an overview*, Int. J. Appl. Math. Comput. Sci, 2001, 11(5), 1093–1121.

[4] A. C. Antoulas, *Approximation of large-scale dynamical Systems*, SIAM, Philadelphia, 2009.

[5] M. Botchev, V. Grimm, and M. Hochbruck, *Residual, restarting, and Richardson iteration for the matrix exponential*, SIAM J. Sci. Comput., 2013, 35(3), 1376–1397.

[6] M. A. Botchev, *A block Krylov subspace time-exact solution method for linear ordinary differential equation systems*, Numer. Linear Algebra Appl., 2013, 20(4), 557–574.

[7] E. Celledoni and I. Moret, *A Krylov projection method for systems of ODEs*, Appl. Numer. Math., 1997, 24(2), 365–378.

[8] Y. H. Cong and D. P. Li, *Block Krylov subspace methods for approximating the linear combination of φ-functions arising in exponential integrators*, Comput. Math. Appl., 2016, 72(4), 846–855.

[9] V. Druskin, C. Lieberman and M. Zaslavsky, *On adaptive choice of shifts in rational Krylov subspace reduction of evolutionary problems*, SIAM J. Sci. Comput., 2010, 32(5), 2485–2496.

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

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

[12] A. Frommer, S. Güttel and M. Schweitzer, *Efficient and stable Arnoldi restarts for matrix functions based on quadrture*, SIAM J. Matrix Anal. Appl., 2014, 35(2), 661–683.

[13] T. Göckler and V. Grimm, *Convergence Analysis of an Extended Krylov Subspace Method for the Approximation of Operator Functions in Exponential Integrators*, SIAM J. Numer. Anal., 2013, 51(4), 2189–2213.

[14] T. Göckler, *Rational Krylov subspace methods for φ-functions in exponential integrators,* PhD thesis, Karlsruhe Institute of Technology (KIT), 2014.

[15] V. Grimm, *Resolvent Krylov subspace approximation to operator functions*, BIT Numer. Math., 2012, 52(3), 639–659.

[16] S. Güttel, *Rational Krylov method for operator functions,* Ph.D. thesis, Fakultät füt Mathematik und Informatik der Technischen Universität Bergakademie Freiberg, 2010.

[17] N. J. Higham, *The Scaling and Squaring Method for the Matrix Exponential Revisited*, SIAM J. Matrix Anal. Appl., 2005, 26(4), 1179–1193.

[18] M. Hochbruck and C. Lubich, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 1997, 34(5), 1912–1925.

[19] M. Hochbruck,and A. Ostermann, *Exponential Integrators*, Acta Numer., 2010, 19(19), 209–286.

[20] N. J. Higham, *Functions of matrices: theory and computation*, SIAM, Philadelphia, 2008.

[21] Z. Jia and H. Lv, *A posteriori error estimates of Krylov subspace approximations to matrix functions*, Numer. Algor., 2015, 69(1), 1–28.

[22] H. M. Kim and R. R. Craig, Jr, *Structural dynamics analysis using an unsymmetric block Lanczos algorithm*, Int. J. for Numer. Meth. Eng., 1988, 26(10), 2305–2318.

[23] L. Knizhnerman and V. Simoncini, *A new investigation of the extended Krylov subspace method for matrix function evaluations*, Numer. Linear Algebra Appl., 2010, 17(4), 615–638.

[24] B. V. Minchev and W. M. Wright, *A review of exponential integrators for first order semi-linear problems*, Tech. report 2/05, Department of Mathematics, NTNU, 2005.

[25] C. Moler, C. V. Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Review, 2003, 20(4), 3–49.

[26] I. Moret and P. Novati, *An interpolatory approximations of the matrix exponential based on Faber polynomials*, J. Comput. Appl. Math., 2001, 131(1–2), 361–380.

[27] I. Moret and P. Novati, *RD-rational approximations of the matrix exponential*, BIT, 2004, 44(3), 595–615.

[28] I. Moret and M. Popolizio, *The restarted shift-and-invert Krylov method for matrix function*, Numer. Linear Algebra Appl., 2014, 21(1), 68–80.

[29] J. Niesen and W. M. Wright, *Algorithm 919: A Krylov subspace algorithm for evaluating the phi- functions appearing in exponential integrators*, ACM Trans. Math. Software, 2012, 38(3), 1–19.

[30] B. Nour-Omid, *Application of the Lanczos algorithm*, Comput. Phy. Comm, 1989, 53(1–3), 157–168.

[31] R. B. Sidje, *Expokit: A software package for computing matrix exponentials*, ACM Trans. Math. Software, 1998, 24(1), 130–156.

[32] B. Skaflestad and W. M. Wright, *The scaling and modified squaring method for matrix functions related to the exponential*, Appl. Numer. Math., 2009, 59(3), 783–799.

[33] T. Schmelzer, *The fast evaluation of matrix functions for exponential integrators,* PhD thesis, University of Oxford, 2007.

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

[35] Y. Saad, *Iterative Methods for sparse linear systems*, 2nd edition, SIAM, Philadelphia, 2003.

[36] H. Tal-ezer, *On restart and error estimation for Krylov approximation of* $\omega = f(A)v$, SIAM J. Sci. Comput., 2007, 29(6), 2426–2441.

[37] G. Wu, H. Pang and J. Sun, *Preconditioning the Restarted and Shifted Block FOM Algorithm for Matrix Exponential Computation*, Mathematics, 2014, 1–24.

[38] Q. Ye, *Error bounds for the Lanczos methods for approximating matrix exponentials*, SIAM. J. Numer Anal., 2013, 51(1), 68–87.