Calcolo (2024) 61:61
https://doi.org/10.1007/s10092-024-00610-3

A μ-mode approach for exponential integrators: actions of φ-functions of Kronecker sums

Marco Caliari[1] · **Fabio Cassini**[1] · **Franco Zivcovich**[2]

Received: 17 January 2024 / Revised: 16 May 2024 / Accepted: 8 August 2024

Abstract
We present a method for computing actions of the exponential-like φ-functions for a Kronecker sum K of d arbitrary matrices A_μ. It is based on the approximation of the integral representation of the φ-functions by Gaussian quadrature formulas combined with a scaling and squaring technique. The resulting algorithm, which we call PHIKS, evaluates the required actions by means of μ-mode products involving exponentials of the *small sized* matrices A_μ, without forming the *large sized* matrix K itself. PHIKS, which profits from the highly efficient level 3 BLAS, is designed to compute different φ-functions applied on the same vector or a linear combination of actions of φ-functions applied on different vectors. In addition, thanks to the underlying scaling and squaring techniques, the desired quantities are available simultaneously at suitable time scales. All these features allow the effective usage of PHIKS in the exponential integration context. In fact, our newly designed method has been tested in popular exponential Runge–Kutta integrators of stiff order from one to four, in comparison with state-of-the-art algorithms for computing actions of φ-functions. The numerical experiments with discretized semilinear evolutionary 2D or 3D advection–diffusion–reaction, Allen–Cahn, and Brusselator equations show the superiority of the proposed μ-mode approach.

Keywords Semilinear evolutionary problems · Kronecker sum · Exponential integrators · φ-functions · μ-mode approach · Tucker operator

Marco Caliari, Fabio Cassini, and Franco Zivcovich have contributed equally to this work.

✉ Marco Caliari
marco.caliari@univr.it

Fabio Cassini
fabio.cassini@univr.it

Franco Zivcovich
franco.zivcovich@gmail.com

[1] Department of Computer Science, University of Verona, Strada Le Grazie, 15, 37134 Verona, Italy

[2] Neurodec, Sophia Antipolis, France

Published online: 21 September 2024

Mathematics Subject Classification 65L05 · 65M20

1 Introduction

We consider a system of Ordinary Differential Equations (ODEs) of the form

$$\begin{cases} \boldsymbol{u}'(t) = K\boldsymbol{u}(t) + \boldsymbol{g}(t, \boldsymbol{u}(t)) = \boldsymbol{f}(t, \boldsymbol{u}(t)), \quad t \in [0, T], \\ \boldsymbol{u}(0) = \boldsymbol{u}_0. \end{cases} \tag{1a}$$

Here, $\boldsymbol{u}\colon [0, T] \to \mathbb{C}^N$ is the unknown, being N the total number of degrees of freedom, $\boldsymbol{g}\colon [0, T] \times \mathbb{C}^N \to \mathbb{C}^N$ is a nonlinear function, and $K \in \mathbb{C}^{N\times N}$ is a large sized matrix which can be written as a Kronecker sum, that is

$$\begin{aligned} K &= A_d \oplus A_{d-1} \oplus \cdots \oplus A_1 = \sum_{\mu=1}^{d} A_{\otimes\mu}, \\ A_{\otimes\mu} &= I_d \otimes \cdots \otimes I_{\mu+1} \otimes A_\mu \otimes I_{\mu-1} \otimes \cdots \otimes I_1. \end{aligned} \tag{1b}$$

Here and throughout the paper $d \in \mathbb{N}$, $\mu = 1, \ldots, d$, I_μ is the identity matrix of size $n_\mu \times n_\mu$, the symbol $\otimes$ denotes the Kronecker product, and, unless otherwise stated, $A_\mu \in \mathbb{C}^{n_\mu \times n_\mu}$ is an arbitrary matrix.

These systems may arise when applying the method of lines to some evolutionary Partial Differential Equations (PDEs), from different fields of science and engineering, defined in a spatial domain $\Omega \subseteq \mathbb{R}^d$ which is the Cartesian product of d intervals. Typical instances are semilinear advection–diffusion, nonlinear Schrödinger, or complex Ginzburg–Landau equations, possibly fractional in space. In dimension $d = 2$, for example, the Laplace operator $\Delta = \partial_{x_1x_1} + \partial_{x_2x_2}$ on a rectangular domain Ω with homogeneous Dirichlet boundary conditions can be discretized as $K = A_2 \oplus A_1 = I_2 \otimes A_1 + A_2 \otimes I_1 \in \mathbb{R}^{N\times N}$, where $A_\mu \in \mathbb{R}^{n_\mu \times n_\mu}$ is the discretization matrix of $\partial_{x_\mu x_\mu}$ with standard second-order finite differences and $N = n_1 n_2$. Notice that several other tensor-product approximation techniques lead to Kronecker sums. We mention higher-order (non)uniform finite differences and lumped mass finite elements (that yield sparse matrices A_μ), or spectral differentiations (that yield dense matrices A_μ). Other type of boundary conditions can be considered, as long as it is possible to write K as a Kronecker sum. In particular, inhomogeneous boundary conditions of Dirichlet or Neumann type can be encapsulated into the nonlinear term $\boldsymbol{g}$.

When system (1) is *stiff*, a prominent way to numerically integrate it in time is by using explicit exponential integrators [1]. These schemes require the action of the exponential and/or of the so-called φ-functions which, for a general matrix $X \in \mathbb{C}^{N\times N}$, are defined as

$$\varphi_\ell(X) = \int_0^1 f_\ell(\theta, X) d\theta, \quad f_\ell(\theta, X) = \frac{\theta^{\ell-1}}{(\ell-1)!} \exp((1-\theta)X), \quad \ell \geq 1. \tag{2}$$

The direct approximation of the matrix φ-functions is feasible only when the size of X is not too large. In this case, the most commonly employed techniques are based on Padé approximations [2], although other rational methods based on the numerical inversion by a quadrature formula of the Laplace transform [3] or polynomial methods based on the truncated Taylor series [4] can be considered. On the other hand, in this manuscript we are interested in matrices of large size. In this case, it is possible to rely on methods which directly compute the action of the matrix φ-functions on a vector, or even their linear combination at once. Among them, Krylov methods [5–7] and other polynomial methods [4, 8–12] are typically employed.

When we consider matrices K which are Kronecker sums (1b), it is possible to express the action of $\exp(K)$ on a vector $\boldsymbol{v}$ by using the Kronecker product of the exponentials of the matrices A_μ. In fact, considering again the two-dimensional case $K = A_2 \oplus A_1$, we have

$$\begin{aligned}\exp(K) &= \exp(I_2 \otimes A_1 + A_2 \otimes I_1) = \exp(I_2 \otimes A_1)\exp(A_2 \otimes I_1)\\ &= (I_2 \otimes \exp(A_1))(\exp(A_2) \otimes I_1) = \exp(A_2) \otimes \exp(A_1),\end{aligned}$$

where we used the commutativity of $I_2 \otimes A_1$ and $A_2 \otimes I_1$, the semigroup property of the exponential function, and the mixed-product property of the Kronecker product. Therefore, we obtain $\exp(K)\boldsymbol{v} = (\exp(A_2) \otimes \exp(A_1))\boldsymbol{v}$. The same result can be accomplished by the more computationally attractive formula $\exp(A_1)\boldsymbol{V}\exp(A_2)^\mathsf{T}$, without computing the Kronecker product $\exp(A_2) \otimes \exp(A_1)$ (see References [13, 14]). Here, $\boldsymbol{V}$ is the matrix of size $n_1 \times n_2$ whose jth column is made by the elements of $\boldsymbol{v}$ from $(j-1)n_1 + 1$ to jn_1, for $j = 1, \ldots, n_2$. This approach can be generalized to the computation of the action of the matrix exponential when K is the Kronecker sum of d *arbitrary* matrices A_μ through the so-called μ-mode product (see the next section for more details). With this technique, it is possible to efficiently implement exponential schemes which require the action of the matrix exponential only, such as some splitting schemes, Lawson methods, and Magnus integrators. Unfortunately, such an elegant approach does not directly apply to the computation of the action of φ-functions, since they do not enjoy the aforementioned semigroup property of the exponential function.

In this paper, we aim at computing actions of φ-functions for a matrix K which is the Kronecker sum of d matrices A_μ using a μ-mode approach, without assembling the matrix K itself. Moreover, since we are interested in the application to exponential integrators which may require more than a single φ-function evaluation, as in the case of exponential Runge–Kutta schemes of high stiff order [15, 16], we will derive an algorithm for the computation of actions of different φ-functions on the *same* vector

$$\{\exp(\tau K)\boldsymbol{v}, \varphi_1(\tau K)\boldsymbol{v}, \varphi_2(\tau K)\boldsymbol{v}, \ldots, \varphi_p(\tau K)\boldsymbol{v}\} \tag{3}$$

at once, as well as for *linear combinations* of actions of φ-functions

$$\exp(\tau K)\boldsymbol{v}_0 + \varphi_1(\tau K)\boldsymbol{v}_1 + \varphi_2(\tau K)\boldsymbol{v}_2 + \cdots + \varphi_p(\tau K)\boldsymbol{v}_p, \tag{4}$$

where τ is the time step size of the integrator. For an efficient computation of the quantities in formula (3) we will use the scaling and modified squaring method proposed in Reference [17], while for the linear combination in formula (4) we will derive a new scaling and squaring procedure. As a byproduct of these techniques, our algorithm can also output the desired quantities at different time scales, a feature of great importance in the implementation of certain exponential integrators.

The remaining part of the paper is structured as follows. In Sect. 2 we briefly describe the founding basis for the new method, i.e., the μ-mode product, the Tucker operator, and its coupling with quadrature formulas for the evaluation of φ-functions. Section 3, the main one, is devoted to the description of the new algorithm, which we call PHIKS (PHI-functions of Kronecker Sums), for the approximation of actions of φ-functions on the same vector and for linear combinations of actions of φ-functions. An important subsection describes a suitable choice of the scaling parameter and of the quadrature formula. In particular, for the latter, we propose an effective closed Gaussian formula. Then, in Sect. 4, we validate our implementation of PHIKS by running several examples in dimensions $d = 3$ and $d = 6$ which involve different φ-functions and their linear combinations. Moreover, we apply the proposed technique to the numerical solution of physically relevant 2D and 3D stiff advection–diffusion–reaction equations, with up to $N = 2 \cdot 100^3$ degrees of freedom, and five different exponential Runge–Kutta integrators of order up to four. Finally, we draw some conclusions in Sect. 5.

2 A μ-mode approach for evolutionary equations in Kronecker form

The founding basis of the technique that we propose in this manuscript is based on the μ-mode approach for the action of the matrix exponential. Due to its importance, we briefly recall here the main concepts, and invite a reader not familiar with the following formalism to check References [18–20] for a thorough explanation with full details. Let us denote by $\boldsymbol{V}$ an order-d tensor of size $n_1 \times \cdots \times n_d$ with elements $v_{i_1 \ldots i_d}$, and by L_μ a matrix of size $n_\mu \times n_\mu$ of elements ℓ^μ_{ij}. Then, the μ-*mode* product of the tensor $\boldsymbol{V}$ with the matrix L_μ, denoted as $\boldsymbol{V} \times_\mu L_\mu$, is the tensor $\boldsymbol{W}$ of size $n_1 \times \cdots \times n_d$ defined elementwise as

$$w_{i_1 \ldots i_d} = \sum_{j_\mu = 1}^{n_\mu} v_{i_1 \ldots i_{\mu-1} j_\mu i_{\mu+1} \ldots i_d} \ell^\mu_{i_\mu j_\mu}.$$

This corresponds to multiply the matrix L_μ onto the μ-*fibers* of the tensor $\boldsymbol{V}$ (i.e., vectors along direction μ which are generalizations to tensors of columns and rows of a matrix). The concatenation of μ-mode products with the matrices $L_1, \ldots, L_d$, that is the tensor $\boldsymbol{W}$ with elements

$$w_{i_1 \ldots i_d} = \sum_{j_d = 1}^{n_d} \cdots \sum_{j_1 = 1}^{n_1} v_{j_1 \ldots j_d} \prod_{\mu = 1}^{d} \ell^\mu_{i_\mu j_\mu},$$

is denoted by $\boldsymbol{V} \times_1 L_1 \times_2 \cdots \times_d L_d$ and referred to as *Tucker operator*. In terms of computational cost, a single μ-mode product requires $\mathcal{O}(Nn_\mu)$ floating point operations, being $N = n_1 \cdots n_d$, and it can be implemented by a single matrix-matrix product. Consequently, the Tucker operator has an overall computational cost of $\mathcal{O}(n^{d+1})$ for the case $n_1 = \ldots = n_d = n$. It can be realized with d calls of level 3 BLAS (Basic Linear Algebra Subprograms [21]), whose highly optimized implementations are available for any kind of modern computer hardware (see, for instance, References [22–24]).

The relation between the Kronecker product and the Tucker operator is given by the following Lemma (the proof can be found, for instance, in Reference [19, Lemma 2.1]).

Lemma 1 *Let $L_\mu \in \mathbb{C}^{n_\mu \times n_\mu}$ be matrices, with $\mu = 1, \ldots, d$, and let $\boldsymbol{v} \in \mathbb{C}^N$, with $N = n_1 \cdots n_d$. Let $\boldsymbol{V} \in \mathbb{C}^{n_1 \times \cdots \times n_d}$ be an order-d tensor such that* $\mathrm{vec}(\boldsymbol{V}) = \boldsymbol{v}$*, where* vec *denotes the operator which stacks by columns the elements of the input tensor. Then, we have*

$$(L_d \otimes L_{d-1} \otimes \cdots \otimes L_1)\boldsymbol{v} = \mathrm{vec}(\boldsymbol{V} \times_1 L_1 \times_2 \cdots \times_d L_d).$$

As observed in the introduction for the case $d = 2$, since K is a Kronecker sum of d matrices, we can similarly write

$$\exp(K)\boldsymbol{v} = (\exp(A_d) \otimes \exp(A_{d-1}) \otimes \cdots \otimes \exp(A_1))\, \boldsymbol{v}$$

which, using the introduced tensor formalism with $L_\mu = \exp(A_\mu)$, can be computed as

$$\exp(K)\boldsymbol{v} = \mathrm{vec}\left(\boldsymbol{V} \times_1 \exp(A_1) \times_2 \cdots \times_d \exp(A_d)\right). \tag{5}$$

The superiority of such an approach to compute the action $\exp(K)\boldsymbol{v}$ has been thoroughly analyzed and highlighted in Reference [18] in the context of the numerical solution of some Schrödinger equations. Also, this technique has been successfully used in Reference [19] for some (linear) advection–diffusion–absorption equations with space dependent coefficients.

Formula (5) can be employed to compute the exact solution of system (1) in the case $\boldsymbol{g}(t, \boldsymbol{u}(t)) \equiv 0$. In general, it is possible to integrate the system by employing an integrator of stiff order one such as exponential Euler

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + \tau\varphi_1(\tau K)\boldsymbol{f}(t_n, \boldsymbol{u}_n), \tag{6a}$$

where $\boldsymbol{u}_n \approx \boldsymbol{u}(t_n)$ and τ is the time step size, constant for simplicity of exposition. Its implementation requires the computation of the action of the φ_1 function on a vector. An equivalent formulation of the scheme (see Reference [15]) is

$$\boldsymbol{u}_{n+1} = \exp(\tau K)\boldsymbol{u}_n + \tau\varphi_1(\tau K)\boldsymbol{g}(t_n, \boldsymbol{u}_n), \tag{6b}$$

and a simple way to evaluate the latter is to compute the action of the exponential of a slightly augmented matrix, that is

$$\exp\left(\begin{bmatrix} \tau K & \tau \boldsymbol{g}(t_n, \boldsymbol{u}_n) \\ 0 \ \dots \ 0 & 0 \end{bmatrix}\right) \begin{bmatrix} \boldsymbol{u}_n \\ 1 \end{bmatrix},$$

and take the first N rows of the resulting vector. The advantage of this approach is that it can be easily generalized to higher order φ-functions [8], whose actions are needed for high stiff order exponential integrators. Indeed, consider the exponential Runge–Kutta scheme of stiff order two ETD2RK [25]

$$\begin{aligned} \boldsymbol{u}_{n2} &= \boldsymbol{u}_n + \tau\varphi_1(\tau K)\boldsymbol{f}(t_n, \boldsymbol{u}_n), \\ \boldsymbol{u}_{n+1} &= \boldsymbol{u}_{n2} + \tau\varphi_2(\tau K)(\boldsymbol{g}(t_{n+1}, \boldsymbol{u}_{n2}) - \boldsymbol{g}(t_n, \boldsymbol{u}_n)), \end{aligned} \tag{7a}$$

or, equivalently,

$$\begin{aligned} \boldsymbol{u}_{n2} &= \exp(\tau K)\boldsymbol{u}_n + \tau\varphi_1(\tau K)\boldsymbol{g}(t_n, \boldsymbol{u}_n), \\ \boldsymbol{u}_{n+1} &= \exp(\tau K)\boldsymbol{u}_n + \tau\varphi_1(\tau K)\boldsymbol{g}(t_n, \boldsymbol{u}_n) + \tau\varphi_2(\tau K)(\boldsymbol{g}(t_{n+1}, \boldsymbol{u}_{n2}) - \boldsymbol{g}(t_n, \boldsymbol{u}_n)). \end{aligned} \tag{7b}$$

For its implementation, we can compute the actions of the two matrix functions $\varphi_1(\tau K)$ and $\varphi_2(\tau K)$ [formulation (7a)]. Alternatively, we can compute two linear combinations of actions of φ-functions of type $\exp(\tau K)\boldsymbol{v}_0 + \varphi_1(\tau K)\boldsymbol{v}_1 + \varphi_2(\tau K)\boldsymbol{v}_2$ [formulation (7b)]. To do the latter, it is again possible to use an augmented matrix approach. In fact, from Theorem 2.1 of Reference [8], we have that the first N rows of the vector

$$\exp\left(c\begin{bmatrix} & \tau K & & \boldsymbol{v}_p & \boldsymbol{v}_{p-1} & \dots & \boldsymbol{v}_2 & \boldsymbol{v}_1 \\ 0 & \dots & 0 & 0 & 1 & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & 0 & \ddots & & \vdots \\ \vdots & & \vdots & \vdots & \vdots & & \ddots & 0 \\ 0 & \dots & 0 & 0 & 0 & \dots & 0 & 1 \\ 0 & \dots & 0 & 0 & 0 & \dots & \dots & 0 \end{bmatrix}\right) \begin{bmatrix} \boldsymbol{v}_0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \quad c \in \mathbb{C},$$

coincide with the vector

$$\exp(c\tau K)\boldsymbol{v}_0 + c\varphi_1(c\tau K)\boldsymbol{v}_1 + c^2\varphi_2(c\tau K)\boldsymbol{v}_2 + \cdots + c^p\varphi_p(c\tau K)\boldsymbol{v}_p.$$

With this technique, the computation of a single linear combination of actions of φ-functions reduces to the action of the exponential of an augmented matrix.

However, in this context it is not possible to directly apply the μ-mode approach for computing the action of the matrix exponential, since the augmented matrix is not anymore a Kronecker sum. The idea is then to take the integral definition of the φ-functions (2) and approximate it by a quadrature rule. By doing so, for the exponential

Euler scheme introduced in formula (6b) we obtain

$$\boldsymbol{u}_{n+1} \approx \exp(\tau K)\boldsymbol{u}_n + \tau \sum_{i=1}^{q} w_i \exp((1-\theta_i)\tau K)\boldsymbol{g}(t_n, \boldsymbol{u}_n),$$

where w_i and θ_i are the quadrature weights and nodes, respectively (see Sect. 3.3 for an effective choice). Since the matrix arguments of the exponential function are Kronecker sums, it is now possible to employ a μ-mode approach for the efficient evaluation of its action. In the next section, we give more details and extend this idea to higher order φ-functions and integrators.

3 Approximation of φ-functions of a Kronecker sum

In this section, by using the tools presented in Sect. 2, we describe in detail how to approximate the action of single φ-functions on the same vector and the linear combination of actions of φ-functions, which are the two tasks addressed by the algorithm PHIKS. For instance, a ν-stage explicit exponential Runge–Kutta integrator [1] with time step size τ is defined by

$$\begin{aligned}
\boldsymbol{u}_{ni} &= \exp(c_i\tau K)\boldsymbol{u}_n + c_i\tau\varphi_1(c_i\tau K)\boldsymbol{g}(t_n, \boldsymbol{u}_n) + \tau\sum_{j=2}^{i-1} a_{ij}(\tau K)\boldsymbol{d}_{nj} \\
&= \boldsymbol{u}_n + c_i\tau\varphi_1(c_i\tau K)\boldsymbol{f}(t_n, \boldsymbol{u}_n) + \tau\sum_{j=2}^{i-1} a_{ij}(\tau K)\boldsymbol{d}_{nj}, \qquad 2 \le i \le \nu, \\
\boldsymbol{u}_{n+1} &= \exp(\tau K)\boldsymbol{u}_n + \tau\varphi_1(\tau K)\boldsymbol{g}(t_n, \boldsymbol{u}_n) + \tau\sum_{i=2}^{\nu} b_i(\tau K)\boldsymbol{d}_{ni} \\
&= \boldsymbol{u}_n + \tau\varphi_1(\tau K)\boldsymbol{f}(t_n, \boldsymbol{u}_n) + \tau\sum_{i=2}^{\nu} b_i(\tau K)\boldsymbol{d}_{ni},
\end{aligned} \tag{8a}$$

where

$$\boldsymbol{d}_{ni} = \boldsymbol{g}(t_n + c_i\tau, \boldsymbol{u}_{ni}) - \boldsymbol{g}(t_n, \boldsymbol{u}_n). \tag{8b}$$

Notice that for $\nu = 1$ this integrator reduces to the exponential Euler method (6), while for $\nu = 2$, $c_2 = 1$, and $b_2 = \varphi_2$ we retrieve the ETD2RK method (7). The generic scheme (8) can be written in a compact way using the reduced tableau [26]

$$\begin{array}{c|cccc}
c_2 & & & & \\
c_3 & a_{32} & & & \\
\vdots & \vdots & \ddots & & \\
c_\nu & a_{\nu 2} & \dots & a_{\nu\nu-1} & \\
\hline
 & b_2 & \dots & b_{\nu-1} & b_\nu
\end{array}$$

Here and throughout the paper, by "reduced tableau" we mean that for each stage and for the final approximation $\boldsymbol{u}_{n+1}$ we write only the coefficients corresponding to the perturbation of the underlying exponential Euler scheme. The matrix functions a_{ij} and b_i are linear combinations of φ-functions. Thus, each stage $\boldsymbol{u}_{ni}$ and the approximation $\boldsymbol{u}_{n+1}$ turn out to be *general* linear combinations of actions of φ-functions, not necessarily in form (4). In fact, it could be more convenient to compute actions of different φ-functions on the same vector (3), and use them to assemble the stages and the final approximation (see the discussion in Sect. 4.4.1). We start by describing this procedure.

3.1 Actions of φ-functions on the same vector

Let us consider for the moment the simple second-order exponential Runge–Kutta method (7a), which requires the computation of $\varphi_1(\tau K)\tau \boldsymbol{f}(t_n, \boldsymbol{u}_n)$ and $\varphi_2(\tau K)\tau(\boldsymbol{g}(t_{n+1}, \boldsymbol{u}_{n2}) - \boldsymbol{g}(t_n, \boldsymbol{u}_n))$. We start with the computation of $\varphi_1(K)\boldsymbol{v}$, where, for clarity of exposition, we omit the time step size τ and use a generic vector $\boldsymbol{v}$. As mentioned in Sect. 2, the idea is to fully exploit the possibility to apply the Tucker operator to compute actions of suitable matrix exponentials. Hence, we directly approximate the integral representation

$$\varphi_1(K)\boldsymbol{v} = \int_0^1 \exp((1-\theta)K)\boldsymbol{v}\, d\theta \tag{9}$$

by a quadrature formula. To avoid an impractical number of quadrature points, we introduce a scaling strategy. Therefore, the quadrature rule is applied to the computation of $\varphi_1(K/2^s)\boldsymbol{v}$, that is

$$\varphi_1(K/2^s)\boldsymbol{v} = \int_0^1 \exp((1-\theta)K/2^s)\boldsymbol{v}\, d\theta \approx \sum_{i=1}^{q} w_i \exp((1-\theta_i)K/2^s)\boldsymbol{v},$$

where θ_i and w_i are q quadrature nodes and weights, respectively. Notice that we choose to scale the matrix K by a power of two to employ the favorable scaling and squaring algorithm [17] for φ-functions. The choices of the quadrature formula, of the number q of quadrature nodes, and of the nonnegative integer scaling s will be discussed in detail in Sect. 3.3. Then, the evaluation of the integrand above at each quadrature point $\theta_i \in [0, 1]$ can be performed by the Tucker operator

$$\boldsymbol{V} \times_1 \exp((1-\theta_i)A_1/2^s) \times_2 \cdots \times_d \exp((1-\theta_i)A_d/2^s), \tag{10}$$

see formula (5). Finally, to recover $\varphi_1(K)\boldsymbol{v}$ from its scaled version, we use the following squaring formula (see again Reference [17])

$$\begin{cases} \varphi_1(K/2^{j-1})\boldsymbol{v} = \dfrac{1}{2}\left(\exp(K/2^j)\varphi_1(K/2^j)\boldsymbol{v} + \varphi_1(K/2^j)\boldsymbol{v}\right), \\ \exp(A_\mu/2^{j-1}) = \exp(A_\mu/2^j)\exp(A_\mu/2^j), \end{cases}$$

which has to be repeated for $j = s, s-1, \ldots, 1$. To perform the squaring, *no* full matrix $\exp(K/2^j)$ has to be evaluated in practice. In fact, to compute its action on $\varphi_1(K/2^j)\boldsymbol{v}$, which is available as a tensor, it is enough to compute the Tucker operator with the small sized matrices $\exp(A_\mu/2^j)$. Notice that the idea of approximating the integral definition of the φ-functions by a quadrature formula and computing the action of the matrix exponential by a Tucker operator has also recently been presented in Reference [27]. Nevertheless, we decided to report the above description for the sake of clarity and to introduce later additional features critical for exponential integrators, such as the effective usage of scaled quantities [see formula (13)] and the extension of the technique to linear combinations of actions of φ-functions (see the next section).

Let us proceed by considering the approximation of the action of $\varphi_2(K)$, that is

$$\varphi_2(K)\boldsymbol{v} = \int_0^1 \theta \exp((1-\theta)K)\boldsymbol{v} d\theta. \tag{11}$$

Comparing integrals (9) and (11) it appears clear that, if we define a common scaling strategy, we can compute the two approximations at once by selecting the same quadrature nodes and weights, but different integrand functions

$$\exp((1-\theta)K/2^s)\boldsymbol{v} \quad \text{and} \quad \theta \exp((1-\theta)K/2^s)\boldsymbol{v}.$$

Therefore, the two quadrature formulas can be implemented with common evaluations of the matrices $\exp((1-\theta_i)A_\mu/2^s)$ for each quadrature point θ_i and each μ. Their action on $\boldsymbol{v}$ is computed with a single Tucker operator (10), followed by the multiplication by the scalar θ_i needed for the approximation of $\varphi_2(K/2^s)\boldsymbol{v}$. After assembling the quadrature, the steps of the squaring are

$$\begin{cases} \varphi_2(K/2^{j-1})\boldsymbol{v} = \dfrac{1}{4}\left(\exp(K/2^j)\varphi_2(K/2^j)\boldsymbol{v} + \varphi_1(K/2^j)\boldsymbol{v} + \varphi_2(K/2^j)\boldsymbol{v}\right), \\ \varphi_1(K/2^{j-1})\boldsymbol{v} = \dfrac{1}{2}\left(\exp(K/2^j)\varphi_1(K/2^j)\boldsymbol{v} + \varphi_1(K/2^j)\boldsymbol{v}\right), \\ \exp(A_\mu/2^{j-1}) = \exp(A_\mu/2^j)\exp(A_\mu/2^j), \end{cases}$$

to be repeated for $j = s, s-1, \ldots, 1$.

The generalization to the computation of the action of the first p φ-functions on the *same* vector $\boldsymbol{v}$

$$\{\varphi_1(K)\boldsymbol{v}, \varphi_2(K)\boldsymbol{v}, \ldots, \varphi_p(K)\boldsymbol{v}\}$$

is straightforward. First, we compute their approximations at the same scaled matrix by the common quadrature rule, i.e.,

$$\varphi_\ell(K/2^s)\boldsymbol{v} \approx \sum_{i=1}^{q} w_i \frac{\theta_i^{\ell-1}}{(\ell-1)!} \exp((1-\theta_i)K/2^s)\boldsymbol{v}, \quad \ell = 1, \ldots, p. \tag{12a}$$

Then, we perform the squaring procedure

$$\begin{cases} \varphi_\ell(K/2^{j-1})\boldsymbol{v} \\ \quad = \dfrac{1}{2^\ell}\Big(\exp(K/2^j)\varphi_\ell(K/2^j)\boldsymbol{v} + \displaystyle\sum_{k=1}^{\ell} \frac{\varphi_k(K/2^j)\boldsymbol{v}}{(\ell-k)!}\Big), \quad \ell = p, p-1, \ldots, 1, \\ \exp(A_\mu/2^{j-1}) = \exp(A_\mu/2^j)\exp(A_\mu/2^j), \end{cases} \tag{12b}$$

for $j = s, s-1, \ldots, 1$. The action of $\exp(K)$ on $\boldsymbol{v}$ can be computed using an additional single Tucker operator of type (5). We stress that the relevant computations in formulas (12) are performed by means of the μ-mode approach, without forming the large sized matrix K. Notice also that, from formulas (12), we obtain at no additional cost also $\varphi_\ell(K/2^{j-1})\boldsymbol{v}$, $j = 2, \ldots, \hat{s}$, where $\hat{s} \le s+1$ is the number of desired scales. This feature could be useful for the efficient implementation of exponential integrators that require, for instance, the quantities

$$\exp(c_j K)\boldsymbol{v},\ \varphi_1(c_j K)\boldsymbol{v},\ \varphi_2(c_j K)\boldsymbol{v}, \ldots, \varphi_p(c_j K)\boldsymbol{v}, \quad j = 1, \ldots, \hat{s}, \tag{13}$$

with $c_j = c/2^{j-1}$ and $c \in \mathbb{C}$, as shown in the numerical examples of Sects. 4.3 and 4.4.1.

Remark 1 Notice that the quadrature rule in formula (12a) is equivalent to

$$\sum_{i=1}^{q} w_i \frac{\theta_i^{\ell-1}}{(\ell-1)!} \exp((1-\theta_i)(K-\sigma I)/2^s)\mathrm{e}^{(1-\theta_i)\sigma/2^s}\boldsymbol{v}, \quad \ell = 1, \ldots, p,$$

where $\sigma \in \mathbb{C}$ is a shift parameter. Given the Kronecker sum structure of K, it is possible to choose σ as the sum of d shifts σ_μ, selected in such a way that $A_\mu - \sigma_\mu I$ has a smaller norm than A_μ (and thus its exponential can be possibly computed in a more efficient way [8, 28]). A common and effective choice for σ_μ is the trace of the matrix A_μ divided by n_μ, which corresponds to its average eigenvalue and minimizes the Frobenius norm of $A_\mu - \sigma_\mu I$.

We now summarize the number of Tucker operators of the whole procedure inside PHIKS needed to obtain the quantities in formula (13). We recall that, if we assume $n_1 = \ldots = n_d = n$, the computational cost of a single Tucker operator is $\mathcal{O}(n^{d+1})$. For each quadrature point we need to compute one Tucker operator. Then, for each step of the squaring phase, we have p Tucker operators. Finally, we have one Tucker operator for the computation of $\exp(K/2^{j-1})\boldsymbol{v}$ for each $j = 1, \ldots, \hat{s}$. Therefore, the total number of Tucker operators is

$$T_\#(s, \hat{s}, q, p) = q + sp + \hat{s}. \tag{14}$$

We remark that for $d \ge 3$ the number $T_\#$ gives an adequate indication of the computational cost of the whole procedure, being the Tucker operator the most expensive

operation. On the other hand, for $d < 3$ other tasks such as the computation of the matrix exponential may have a comparable cost (or even higher, in the trivial case $d = 1$).

3.2 Linear combination of actions of φ-functions

Let us consider for the moment the Runge–Kutta scheme (7b), which requires the two linear combinations of actions of φ-functions $\exp(\tau K)\boldsymbol{u}_n + \varphi_1(\tau K)\tau \boldsymbol{g}(t_n, \boldsymbol{u}_n)$ and $\exp(\tau K)\boldsymbol{u}_n + \varphi_1(\tau K)\tau \boldsymbol{g}(t_n, \boldsymbol{u}_n) + \varphi_2(\tau K)\tau(\boldsymbol{g}(t_{n+1}, \boldsymbol{u}_{n2}) - \boldsymbol{g}(t_n, \boldsymbol{u}_n))$.

We then introduce the compact notation

$$\Phi_j(K)\boldsymbol{v}_{i_1,i_2,\ldots,i_p} = \frac{\varphi_1(K/2^j)\boldsymbol{v}_{i_1}}{2^j} + \frac{\varphi_2(K/2^j)\boldsymbol{v}_{i_2}}{2^{2j}} + \cdots + \frac{\varphi_p(K/2^j)\boldsymbol{v}_{i_p}}{2^{pj}}, \quad p > 1,$$

with j nonnegative integer and, for simplicity of exposition, we describe in detail the approximation of $\Phi_0(K)\boldsymbol{v}_{1,2}$. The idea is to apply a quadrature formula to the integral

$$\Phi_0(K)\boldsymbol{v}_{1,2} = \int_0^1 \exp((1-\theta)K)(\boldsymbol{v}_1 + \theta \boldsymbol{v}_2)d\theta$$

in combination with a scaling strategy. To do so, we first approximate by a common quadrature rule the scaled linear combinations

$$\begin{aligned}
\Phi_s(K)\boldsymbol{v}_2 &= \int_0^1 \exp((1-\theta)K/2^s)\frac{\boldsymbol{v}_2}{2^s}d\theta \approx \sum_{i=1}^{q} w_i \exp((1-\theta_i)K/2^s)\frac{\boldsymbol{v}_2}{2^s},\\
\Phi_s(K)\boldsymbol{v}_{1,2} &= \int_0^1 \exp((1-\theta)K/2^s)\left(\frac{\boldsymbol{v}_1}{2^s} + \theta\frac{\boldsymbol{v}_2}{2^{2s}}\right)d\theta\\
&\approx \sum_{i=1}^{q} w_i \exp((1-\theta_i)K/2^s)\left(\frac{\boldsymbol{v}_1}{2^s} + \theta_i\frac{\boldsymbol{v}_2}{2^{2s}}\right).
\end{aligned}$$

Notice that the approximations in the above formulas require the common evaluation of the matrices $\exp((1-\theta_i)A_\mu/2^s)$ for every quadrature node θ_i and every μ. In addition, each of the two approximations needs a single Tucker operator for every θ_i. Then, it is possible to perform the squaring procedure of $\Phi_s(K)\boldsymbol{v}_{1,2}$ by

$$\begin{cases}
\Phi_{j-1}(K)\boldsymbol{v}_{1,2} = \exp(K/2^j)\Phi_j(K)\boldsymbol{v}_{1,2} + \dfrac{\Phi_j(K)\boldsymbol{v}_2}{2^j} + \Phi_j(K)\boldsymbol{v}_{1,2},\\
\Phi_{j-1}(K)\boldsymbol{v}_2 = \exp(K/2^j)\Phi_j(K)\boldsymbol{v}_2 + \Phi_j(K)\boldsymbol{v}_2,\\
\exp(A_\mu/2^{j-1}) = \exp(A_\mu/2^j)\exp(A_\mu/2^j),
\end{cases}$$

with $j = s, s-1, \ldots, 1$. The first two rows can be easily verified by computing blocks $(1,3)$ and $(1,2)$ of the squares of the matrices in equality

$$\exp\left(2^{-j}\begin{bmatrix} K & \boldsymbol{v}_2 & \boldsymbol{v}_1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}\right) = \begin{bmatrix} \exp(K/2^j) & \Phi_j(K)\boldsymbol{v}_2 & \Phi_j(K)\boldsymbol{v}_{1,2} \\ 0 & 1 & 2^{-j} \\ 0 & 0 & 1 \end{bmatrix}.$$

The generalization to a *linear combination* of actions of the first p φ-functions on the vectors $\boldsymbol{v}_\ell$

$$\Phi_0(K)\boldsymbol{v}_{1,2,\ldots,p} = \varphi_1(K)\boldsymbol{v}_1 + \varphi_2(K)\boldsymbol{v}_2 + \cdots + \varphi_p(K)\boldsymbol{v}_p$$

requires first the application of the common quadrature rule to the scaled linear combinations

$$\Phi_s(K)\boldsymbol{v}_{p-\ell+1,p-\ell+2,\ldots,p} \approx \sum_{i=1}^{q} w_i \exp((1-\theta_i)K/2^s)\left(\sum_{k=1}^{\ell} \frac{\theta_i^{\ell-k}}{(\ell-k)!}\frac{\boldsymbol{v}_{p+1-k}}{2^{(\ell-k+1)s}}\right), \tag{15a}$$

with $\ell = 1, \ldots, p$. Then, the squaring procedure

$$\begin{cases} \Phi_{j-1}(K)\boldsymbol{v}_{p-\ell+1,p-\ell+2,\ldots,p} \\ \quad = \exp(K/2^j)\Phi_j(K)\boldsymbol{v}_{p-\ell+1,p-\ell+2,\ldots,p} \\ \qquad + \displaystyle\sum_{k=1}^{\ell} \frac{\Phi_j(K)\boldsymbol{v}_{p-k+1,p-k+2,\ldots,p}}{(\ell-k)!2^{(\ell-k)j}}, \quad \ell = p, p-1, \ldots, 1, \\ \exp(A_\mu/2^{j-1}) = \exp(A_\mu/2^j)\exp(A_\mu/2^j) \end{cases} \tag{15b}$$

has to be repeated for $j = s, s-1, \ldots, 1$. Clearly, the action of $\exp(K)$ on the vector $\boldsymbol{v}_0$ can be added to the linear combination $\Phi_0(K)\boldsymbol{v}_{1,2,\ldots,p}$ at the cost of a single additional Tucker operator of type (5). We stress again that all the computations in (15) are performed in a μ-mode fashion. Notice also that, from the squaring formula, we obtain at no additional cost the quantities $\Phi_{j-1}(K)\boldsymbol{v}_{1,2,\ldots,p}$, $j = 2, \ldots, \hat{s}$, which can be employed in the efficient implementation of exponential integrators that require, for instance, combinations of the form

$$\exp(c_j K)\boldsymbol{v}_0 + c_j\varphi_1(c_j K)\boldsymbol{v}_1 + c_j^2\varphi_2(c_j K)\boldsymbol{v}_2 + \cdots + c_j^p\varphi_p(c_j K)\boldsymbol{v}_p, \quad j = 1, \ldots, \hat{s}, \tag{16}$$

with $c_j = c/2^{j-1}$ and $c \in \mathbb{C}$. An explicit example will be presented in the numerical experiment of Sect. 4.4.2.

We now summarize the number of Tucker operators needed by the whole procedure inside PHIKS to obtain the quantities in formula (16). For each quadrature point we need to compute p Tucker operators. Then, for each step of the squaring phase, we have p Tucker operators. Finally, we have one Tucker operator for the computation of $\exp(K/2^{j-1})\boldsymbol{v}_0$ for each $j = 1, \ldots, \hat{s}$. Therefore, the total number of Tucker

operators is

$$T_{\#}(s, \hat{s}, q, p) = qp + sp + \hat{s}. \tag{17}$$

3.3 Choice of *s*, *q*, and quadrature formula

The choice of the scaling value s and the number of quadrature points q is based on a suitable expansion of the error of the quadrature formula. After this selection, the algorithm is *direct* and no convergence test or exit criterion is needed. We start writing

$$\varphi_\ell(K) = \int_0^1 f_\ell(\theta, K)d\theta = \sum_{i=1}^{q} w_i f_\ell(\theta_i, K) + R_q(f_\ell(\cdot, K)), \tag{18}$$

where $R_q(f_\ell(\cdot, K))$ is the remainder

$$R_q(f_\ell(\cdot, K)) = \frac{1}{2\pi \mathrm{i}} \oint_\Gamma k_q(z) f_\ell(z, K)dz, \tag{19}$$

see Sect. 4.6 of Reference [29]. Here, $\Gamma \subset \mathbb{C}$ is an arbitrary simple closed curve surrounding the interval $[0, 1]$ and k_q is the kernel defined by

$$k_q(z) = \int_0^1 \frac{\pi_q(t)}{\pi_q(z)(z - t)}dt,$$

with $\pi_q(t)$ the monic polynomial of degree q with the quadrature points as roots. Now, we describe a practical procedure to choose the scaling value s and the number of quadrature points q which balances accuracy and efficiency of the overall algorithms. In particular, we determine the parameters so that the remainder is below a certain tolerance, while the number of Tucker operators $T_{\#}$ is kept reasonably low to ensure that the computational cost is not excessive.

Let us consider first the case of actions of φ-functions on the same vector, as in formula (12a). Then, given a tolerance δ and starting from the scaling $s_0 = 0$, we look for the smallest number $q_0 \in [q_{\min}, q_{\max}]$ of quadrature points such that

$$\|R_{q_0}(f_\ell(\cdot, K))\|\|\boldsymbol{v}\| \leq \delta, \quad \ell = 1, \ldots, p.$$

We then repeat the procedure for increasing values of the scaling $s_j \in \{1, 2, \ldots\}$ and look for the corresponding smallest value q_j such that

$$\|R_{q_j}(f_\ell(\cdot, K/2^{s_j}))\|\|\boldsymbol{v}\| \leq \delta \cdot 2^{\ell s_j}, \quad \ell = 1, \ldots, p.$$

Here, the tolerance is amplified by the factor $2^{\ell s_j}$ because we take into account that squaring formula (12b) requires s_j divisions by 2^ℓ. We continue until the number of Tucker operators $T_{\#}(s_{\bar{j}+1}, \hat{s}, q_{\bar{j}+1}, p)$ in formula (14) is larger than $T_{\#}(s_{\bar{j}}, \hat{s}, q_{\bar{j}}, p)$. The obtained values $s = s_{\bar{j}}$ and $q = q_{\bar{j}}$ are then employed in the approximation of actions of φ-functions applied on the vector $\boldsymbol{v}$ through formulas (12).

On the other hand, when considering a linear combination of actions of φ-functions, the quantities to be computed in formula (15a), starting from the scaling $s_0 = 0$, correspond to the integrand functions

$$\sum_{k=1}^{\ell} f_{\ell-k+1}(\theta, K)\boldsymbol{v}_{p+1-k}, \quad \ell = 1, \dots, p.$$

Therefore, for a given tolerance δ, we look for the smallest number $q_0 \in [q_{\min}, q_{\max}]$ of quadrature points needed for all the values such that

$$\sum_{k=1}^{\ell} \|R_{q_0}(f_{\ell-k+1}(\cdot, K))\| \|\boldsymbol{v}_{p+1-k}\| \le \delta, \quad \ell = 1, \dots, p.$$

Then, similarly to the previous case, we repeat the calculation for increasing values of the scaling $s_j \in \{1, 2, \dots\}$ to obtain the corresponding smallest value q_j such that

$$\sum_{k=1}^{\ell} \|R_{q_j}(f_{\ell-k+1}(\cdot, K/2^{s_j}))\| \frac{\|\boldsymbol{v}_{p+1-k}\|}{2^{(\ell-k+1)s_j}} \le \delta, \quad \ell = 1, \dots, p.$$

We continue this procedure until the number of Tucker operators $T_{\#}(s_{\bar{j}+1}, \hat{s}, q_{\bar{j}+1}, p)$ in formula (17) is larger than $T_{\#}(s_{\bar{j}}, \hat{s}, q_{\bar{j}}, p)$. The obtained values $s = s_{\bar{j}}$ and $q = q_{\bar{j}}$ are then employed in the approximation of the linear combination of φ-functions applied to the vectors $\boldsymbol{v}_1, \dots, \boldsymbol{v}_p$ through formulas (15).

The previous estimates clearly require computable bounds for the remainders with different numbers of quadrature points, integrand functions, and scaling parameters. To avoid cumbersome notation, we explain the procedure for $R_q(f_\ell(\cdot, K))$ in formula (19). We choose $\Gamma = \Gamma_r$ to be the ellipse with foci in $\{0, 1\}$ and logarithmic capacity (half sum of its semi-axes) $r > 1/4$, that is

$$\Gamma_r = \left\{ z \in \mathbb{C} : z = z(\zeta) = r\mathrm{e}^{\mathrm{i}\zeta} + \frac{1}{2} + \frac{\mathrm{e}^{-\mathrm{i}\zeta}}{16r}, \quad \text{with } \zeta \in [0, 2\pi) \right\}.$$

Then, we have

$$\begin{aligned} \|R_q(f_\ell(\cdot, K))\| &= \left\| \frac{1}{2\pi\mathrm{i}} \oint_{\Gamma_r} k_q(z) f_\ell(z, K) dz \right\| \\ &= \frac{1}{2\pi} \left\| \int_0^{2\pi} k_q(z(\zeta)) f_\ell(z(\zeta), K) \left(r\mathrm{e}^{\mathrm{i}\zeta} - \frac{\mathrm{e}^{-\mathrm{i}\zeta}}{16r} \right) d\zeta \right\|. \end{aligned}$$

Finally, by using the fact that the numerical range of K (denoted by $\mathcal{W}(K)$), is a $(1+\sqrt{2})$-spectral set [30], we estimate in the 2-norm

$$\|R_q(f_\ell(\cdot,K))\|_2 \leq \frac{1+\sqrt{2}}{2\pi} \sup_{w\in\Omega} \left| \int_0^{2\pi} k_q(z(\zeta)) f_\ell(z(\zeta),w) \left(r\mathrm{e}^{\mathrm{i}\zeta} - \frac{\mathrm{e}^{-\mathrm{i}\zeta}}{16r} \right) d\zeta \right|, \tag{20}$$

being $\Omega \subset \mathbb{C}$ a smooth, bounded, convex domain which embraces $\mathcal{W}(K)$. In our situation, we can easily find such a domain without assembling the matrix K. Indeed, it is possible to show that

$$\mathcal{W}(K) = \mathcal{W}(A_{\otimes 1}) + \mathcal{W}(A_{\otimes 2}) + \cdots + \mathcal{W}(A_{\otimes d}) = \mathcal{W}(A_1) + \mathcal{W}(A_2) + \cdots + \mathcal{W}(A_d),$$

and $\mathcal{W}(A_\mu)$ can be estimated [9] with a rectangle Ξ_μ obtained by computing the norms of the Hermitian and the skew-Hermitian parts of the small sized matrices A_μ. Thus, the rectangle $\Xi = \Xi_1 + \cdots + \Xi_d$ embraces $\mathcal{W}(K)$ and, thanks to the maximum modulus principle, the supremum in estimate (20) is attained at the boundary of Ξ, which we suitably discretize. Moreover, we approximate the integral by the trapezoidal rule.

Concerning the choice of the main quadrature formula (18), we use the Gauss–Lobatto–Legendre one with a number of quadrature points in the interval $[q_{\min}, q_{\max}] = [3, 12]$. These bounds appear to be adequate for the addressed numerical experiments. Besides being very accurate, it employs the endpoints of the integration interval $[0, 1]$. This allows on one side to avoid one Tucker operator of type (10) (since $\theta_q = 1$), and on the other to avail of the quantities $\exp(A_\mu/2^s)$ (corresponding to $\theta_1 = 0$), which are needed for the squaring procedures. This shrewdness, together with the fact that we can avoid computing the Tucker operators for the matrix exponential in formula (16) if $\boldsymbol{v}_0$ is zero, is taken into account in the actual implementation of the algorithm PHIKS. Finally, the evaluation of the kernel k_q in estimate (20) is obtained by the recurrence relation of the underlying orthogonal polynomials (see Reference [31]).

4 Numerical experiments

In this section, we validate our MATLAB[1] implementation of PHIKS and present the effectiveness of the proposed algorithm for the numerical solution of stiff systems of ODEs with exponential Runge–Kutta integrators from stiff order one to four. The implemented algorithm, which works in *any* space dimension d, employs the function TUCKER (contained in the package KronPACK[2]) to compute the underlying Tucker operators by means of μ-mode products. In addition, it uses the internal MATLAB function EXPM for the approximation of the needed matrix exponentials. Such a function is based on the double precision scaling and squaring Padé algorithm [32].

Concerning the two-dimensional example described in Sect. 4.3, we compare the efficiency of our approach with a technique recently introduced for the computa-

[1] The code is available at https://github.com/caliarim/phiks and is fully compatible with GNU Octave.

[2] https://github.com/caliarim/KronPACK.

tion of φ-functions of matrices that have Kronecker sum structure [33]. The method, which was developed for the two-dimensional case only, is direct, does not require an input tolerance, and retrieves the action of a single φ-function of order ℓ by solving ℓ Sylvester equations. This approach has some restrictions on the input matrices (A_1 and $-A_2$ must have disjoint spectra to have a unique solution of the Sylvester equation) and it may suffer of ill-conditioning for φ-functions of high order. The accompanying software[3] of Reference [33] contains the scripts used by the authors to perform their numerical examples. For our purposes, we extracted the parts devoted to the computation of the φ-functions using the MATLAB function SYLVESTER, and collected them in a function (that we named SYLVPHI) to be easily employed in our numerical experiments. In addition, in this two-dimensional example and in all the remaining three-dimensional ones, we compare our approach with recent and popular algorithms for computing linear combinations of actions of φ-functions for large and sparse general matrices, whose code in publicly available. For convenience of the reader, we briefly describe them in the following.

- PHIPM_SIMUL_IOM[4] is a Krylov subspace solver with incomplete orthogonalization [6] which computes linear combinations of actions of φ-functions at different time scales, by expressing everything in terms of the highest order φ-function and using a recurrence relation.
- KIOPS[5] is another adaptive Krylov subspace solver with incomplete orthogonalization [5]. It computes linear combinations of actions of φ-functions at different time scales by using the augmented matrix technique.
- BAMPHI[6] is a hybrid Krylov-polynomial method [10] for computing linear combinations of actions of φ-functions at different time scales, equipped with a backward error analysis of the underlying polynomial approximation. In contrast to the previous methods, it does not require to store a Krylov subspace.

We used all these methods with an incomplete orthogonalization procedure of length two. Moreover, since their MATLAB implementations output some information that can be effectively used for successive calls, such as an estimate of the appropriate Krylov subspace size, in our numerical experience we obtained overall the best results by adopting the following strategy: for each call of the routine at a certain time step we input the information obtained by the same call at the previous time step. In addition, these three methods, together with PHIKS, require an input tolerance, but their error estimates are substantially different. For this reason, we decided to set the tolerance of each method to a value proportional to both the local error of the used time marching scheme and the 2-norm of the current approximation $\boldsymbol{u}_n$. The proportionality constant has been selected for each method and each integrator as large as possible among the powers of two, in such a way that the final error measured with respect to a reference solution is not affected by the approximation error of the matrix functions. We believe that running the experiments with tolerances obtained in this way yields a

[3] https://github.com/jmunoz022/Kronecker_EI.

[4] https://github.com/drreynolds/Phipm_simul_iom.

[5] https://gitlab.com/stephane.gaudreault/kiops/-/tree/master/.

[6] https://github.com/francozivcovich/bamphi.

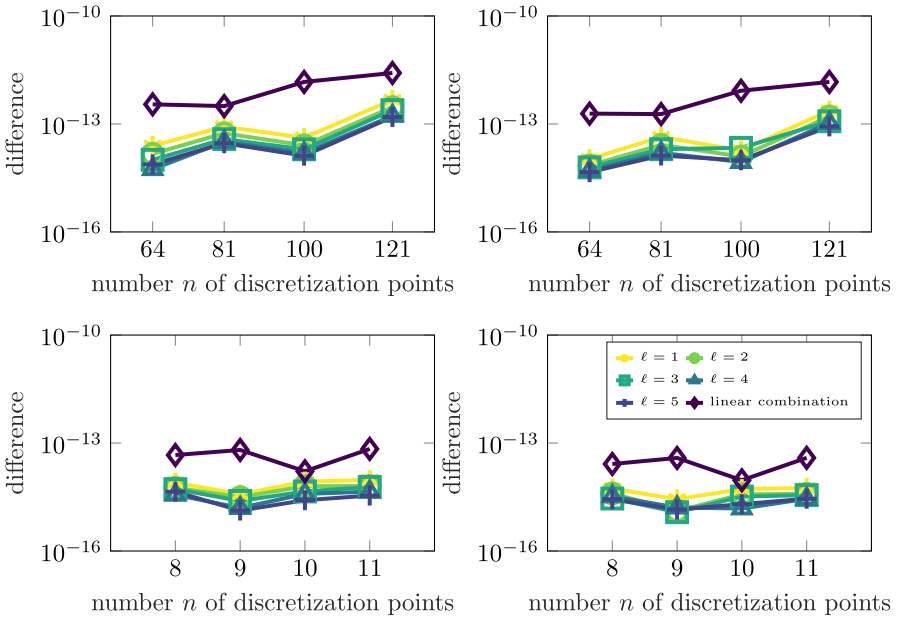

Fig. 1 Relative difference between KIOPS and PHIKS, measured in the infinity norm, for the actions $\varphi_\ell(K/2^j)\boldsymbol{v}$, $\ell = 1, \ldots, p$, and for the linear combinations $\Phi_j(K)\boldsymbol{v}_{1,2,\ldots,p}$ in the code validation. The plots refer to $j = 0$ and $d = 3$ (top left), $j = 1$ and $d = 3$ (top right), $j = 0$ and $d = 6$ (bottom left), $j = 1$ and $d = 6$ (bottom right)

fair comparison among all the methods, ensuring the minimal effort needed to reach the accuracy of the considered time marching schemes. The study of a more sophisticated technique for an effective choice of the tolerances is far beyond the scope of this manuscript.

All the numerical experiments were performed on an Intel® Core™ i7-10750 H CPU with six physical cores and 16GB of RAM, using MathWorks MATLAB® R2022a. The errors were measured in the infinity norm relatively to either the analytical solution, when available, or to a reference solution computed by the fourth-order integrator (28), implemented with the PHIKS routine and a sufficiently large number of time steps.

4.1 Code validation

We extensively tested the PHIKS routine and we present here the results regarding the approximation of actions of φ-functions on the same vector and linear combinations of actions of φ-functions up to order $p = 5$. The test matrices arise from the discretization by standard second order finite differences of the complex operator $(1 + \mathrm{i})/100 \cdot \Delta$ in the spatial domain $[0, 1]^d$, for $d = 3$ and $d = 6$, with homogeneous Dirichlet boundary conditions. The application vectors $\boldsymbol{v}_1 = \ldots = \boldsymbol{v}_p = \boldsymbol{v}$ are the discretization of

$$4096(1 + \mathrm{i}) \prod_{\mu=1}^{d} x_\mu (1 - x_\mu).$$

Table 1 Values of the scaling parameter s, number of quadrature points q, and number of Tucker operators $T_\#$ to compute actions of φ-functions on the same vector (top) and linear combinations of actions of φ-functions (bottom), employed by PHIKS in the code validation

	$d=3$				$d=6$			
φ-functions on the same vector								
n	64	81	100	121	8	9	10	11
s	8	8	9	9	3	3	3	4
q	10	12	11	12	11	11	12	10
$T_\#$	52	54	58	59	28	28	29	32
Linear combination of φ-functions								
n	64	81	100	121	8	9	10	11
s	10	11	11	12	5	6	6	6
q	7	7	8	7	8	7	7	7
$T_\#$	87	92	97	97	67	67	67	67

The number n of discretization points for each spatial direction ranges from 64 to 121 for $d = 3$, and from 8 to 11 for $d = 6$. As a term of comparison we consider the results obtained with KIOPS. Both routines were called with input tolerance set to the double precision unit roundoff value 2^{-53}. We report in Fig. 1 the relative difference in the infinity norm between the approaches and, for PHIKS, we collect in Table 1 the values of the scaling parameter s, the number of quadrature points q, and the number of Tucker operators $T_\#$ [see formulas (14) and (17)]. Overall, we observe an homogeneous behavior of the relative difference between KIOPS and PHIKS for all the values of d, n, and ℓ, and a number of Tucker operators required by the routine PHIKS which increases very slowly with n.

4.2 Evolutionary advection–diffusion–reaction equation

In this section we consider the following evolutionary Advection–Diffusion–Reaction (ADR) equation

$$\begin{cases} \partial_t u(t,x_1,x_2,x_3) = \varepsilon \Delta u(t,x_1,x_2,x_3) + \alpha(\partial_{x_1}+\partial_{x_2}+\partial_{x_3})u(t,x_1,x_2,x_3) \\ \qquad\qquad + g(t,x_1,x_2,x_3,u(t,x_1,x_2,x_3)), \\ u_0(x_1,x_2,x_3) = 64x_1(1-x_1)x_2(1-x_2)x_3(1-x_3), \end{cases} \tag{21a}$$

in the spatial domain $[0,1]^3$, where the nonlinear function g is defined by

$$g(t,x_1,x_2,x_3,u(t,x_1,x_2,x_3)) = \frac{1}{1+u(t,x_1,x_2,x_3)^2} + \Psi(t,x_1,x_2,x_3). \tag{21b}$$

Here, $\Psi(t,x_1,x_2,x_3)$ is chosen so that the analytical solution is $u(t,x_1,x_2,x_3) = \mathrm{e}^t u_0(x_1,x_2,x_3)$. Finally, the equation is coupled with homogeneous Dirichlet boundary conditions. The diffusion and advection parameters are set to $\varepsilon = 0.5$ and $\alpha = 10$, respectively. After the semidiscretization in space by second-order centered finite differences we end up with an ODEs system of type (1), with K a matrix with hep-

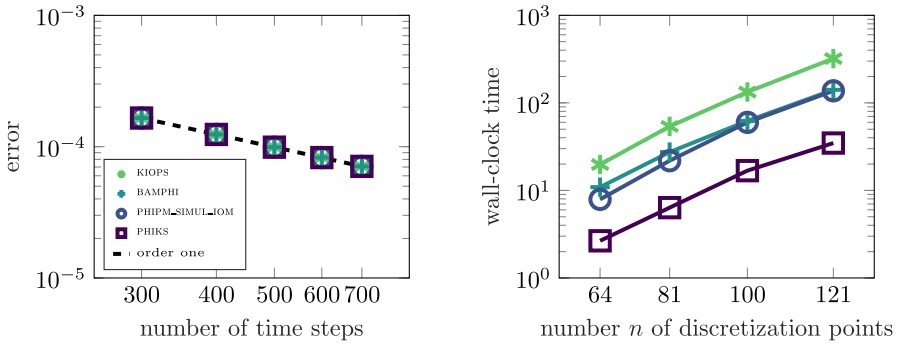

Fig. 2 Rate of convergence of the exponential Euler scheme (6b) for the semidiscretization of ADR equation (21) with $n_1 = n_2 = n_3 = n = 20$ discretization points (left) and wall-clock time in seconds for increasing number n of discretization points and 250 time steps up to final time $T = 0.1$ (right)

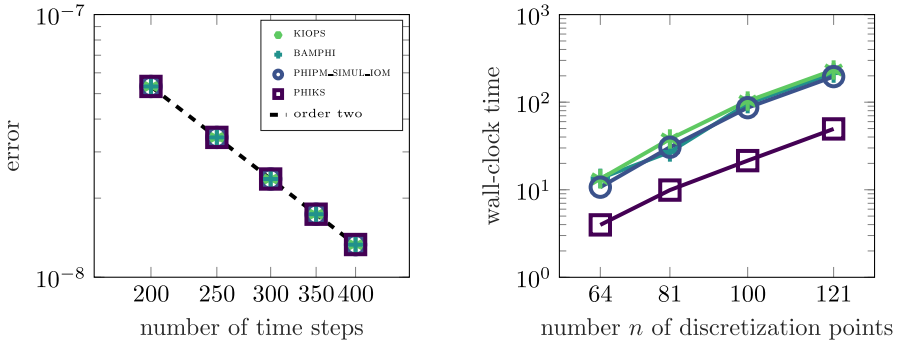

Fig. 3 Rate of convergence of the ETD2RK scheme (7b) for the semidiscretization of ADR equation (21) with $n_1 = n_2 = n_3 = n = 20$ discretization points (left) and wall-clock time in seconds for increasing number n of discretization points and 100 time steps up to final time $T = 0.1$ (right)

tadiagonal structure. This is a three-dimensional variation of a standard stiff example [15] for exponential integrators.

4.2.1 Exponential Euler

We start by implementing the exponential Euler scheme (6b) and test its correct order of convergence for a discretization in space with $n_1 = n_2 = n_3 = n = 20$ internal points and a final simulation time $T = 0.1$ (Fig. 2, left). Then, we test the efficiency of the underlying methods for computing linear combination (6b), with a number of discretization points in each direction ranging from $n = 64$ to $n = 121$. The results are presented in Fig. 2, right. We observe that PHIKS turns out to be at least two times faster than the other considered methods.

Table 2 Wall-clock times in seconds and average number of Tucker operators $T_\#$ per time step for the solution of the semidiscretized PDE (21) by ETD2RK implemented by two calls of PHIKS either for the linear combinations [see formula (7b)] or separately for the functions φ_1 and φ_2 [see formula (7a)]

	Linear combination of φ-functions				φ-functions on the same vector			
n	64	81	100	121	64	81	100	121
Wall-clock	3.97	9.91	21.5	49.5	2.96	6.33	17.8	34.9
$T_\#$	20.0	23.0	23.0	26.0	12.0	12.0	15.0	15.0

4.2.2 Exponential Runge–Kutta scheme of order two

We then move to the implementation of the ETD2RK scheme. This integrator is implemented following the linear combination approach [see formula (7b)] and thus requires two calls of the algorithms (see the beginning of Sect. 3.2) for each time step. Again, since we consider four different routines for the approximation of the actions of matrix functions, we test the correct order of convergence of the schemes. In addition, we measure the performance of the routines as the discretization in space becomes finer and finer. The results, collected in Fig. 3, are similar to the exponential Euler case. We notice that PHIKS turns out to be almost four times faster than the best of the other methods, PHIPM_SIMUL_IOM, in the largest size scenario (total number of degrees of freedom $N = 121^3$).

For comparison, we also implemented the integrator ETD2RK by two calls of the routine PHIKS to compute separately the actions of the functions φ_1 and φ_2 (see the beginning of Sect. 3.1). The results are reported in Table 2 and show that this approach leads to a smaller number of Tucker operators which translates into less wall-clock time. Predicting which approach gives the smallest computational cost for a generic integrator is difficult. As a rule of thumb, we suggest to use the version of the algorithm which requires less calls and, when their number is the same, to prefer the computation of actions of φ-functions on the same vector, since the total number of Tucker operators $T_\#$ is smaller [compare formulas (14) and (17)].

4.3 Allen–Cahn equation

In this section we examine an example similar to the one reported in Reference [33], which describes the Sylvester approach for the computation of the φ-functions. It is the two-dimensional Allen–Cahn phase-field model equation [34] for the concentration u

$$\begin{cases} \partial_t u(t,x_1,x_2) = \Delta u(t,x_1,x_2) + \dfrac{1}{\epsilon^2} u(t,x_1,x_2)(1-u^2(t,x_1,x_2)) \\ \qquad\qquad\qquad = \left(\Delta + \dfrac{1}{\epsilon^2}\right) u(t,x_1,x_2) + g(u(t,x_1,x_2)), \\ u(0,x_1,x_2) = u_0(x_1,x_2), \end{cases} \tag{22a}$$

in the spatial domain $[0, 1]^2$, coupled with homogeneous Neumann boundary conditions. The initial condition is given by

$$u_0(x_1, x_2) = \tanh\left(\frac{\frac{1}{4} + \frac{1}{10}\cos\left(\beta \text{atan2}\left(x_2 - \frac{1}{2}, x_1 - \frac{1}{2}\right)\right) - \sqrt{\left(x_1 - \frac{1}{2}\right)^2 + \left(x_2 - \frac{1}{2}\right)^2}}{\sqrt{2}\alpha}\right). \quad (22b)$$

We set $\epsilon = 0.05$, $\beta = 7$, $\alpha = 0.75$, and we discretize in space with second-order centered finite differences, thus obtaining a system in form (1) with K a matrix with pentadiagonal structure. We simulate until final time $T = 0.025$. Notice that the linear operator $\Delta + \frac{1}{\epsilon^2}$ guarantees a unique solution for the corresponding Sylvester equation, even with homogeneous Neumann boundary conditions. As time marching scheme, we employ the third-order exponential Runge–Kutta integrator with reduced tableau

$$\begin{array}{c|cc} c_2 & & \\ c_3 & \gamma c_2 \varphi_{2,2} + \frac{c_3^2}{c_2}\varphi_{2,3} & \\ \hline & \frac{\gamma}{\gamma c_2 + c_3}\varphi_2 & \frac{1}{\gamma c_2 + c_3}\varphi_2 \end{array} \quad (23)$$

with $c_3 = 2c_2 = 1/2$ and $\gamma = \frac{(3c_3-2)c_3}{(2-3c_2)c_2} = -4/5$, see formula (5.9) in Reference [15]. Here and in the next tableaux $\varphi_{\ell,j}$ denotes $\varphi_\ell(c_j \tau K)$. The implementation involves the usage only of the φ_1 and φ_2 functions, which do not trigger the ill-conditioning of the Sylvester equation observed in Reference [33]. This integrator requires to compute the following actions (scaled by proper coefficients)

$$\begin{aligned} &\varphi_1(\tau K/2^j)\boldsymbol{f}(t_n, \boldsymbol{u}_n), && j = 0, 1, 2, \\ &\varphi_2(\tau K/2^j)\boldsymbol{d}_{n2}, && j = 0, 1, 2, \\ &\varphi_2(\tau K)\boldsymbol{d}_{n3}, \end{aligned}$$

see formula (8). The SYLVPHI routine is then called in total six times: three times to compute the action of the φ_1 function at the different scales of K, twice to compute the action of the φ_2 function for the scales $j = 1$ and $j = 2$ and, finally, once to compute the action of $\varphi_2(\tau K)$ to $(\gamma \boldsymbol{d}_{n2} + \boldsymbol{d}_{n3})/(\gamma c_2 + c_3)$. Therefore, nine Sylvester equations have to be solved. The other four routines have to be called three times, one for each of the above rows. In fact, all of them are natively able to produce the action of single φ-functions simultaneously at different scales of K (see, in particular, Sect. 3.1 for PHIKS). The results are summarized in Fig. 4. Also in this two-dimensional example, with numbers of degrees of freedom up to $N = 651^2$, the PHIKS routine turns out to be always the fastest by a factor of roughly 1.5 with respect to the other techniques.

4.4 Brusselator model

In this section we apply the μ-mode approach to the block diagonal ODEs system which arises from the semidiscretization in space of the three-dimensional Brusselator

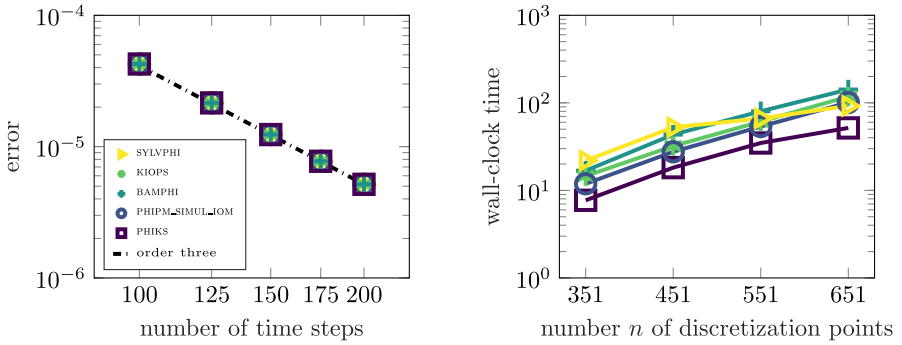

Fig. 4 Rate of convergence of the exponential Runge–Kutta scheme (23) for the semidiscretization of Allen–Cahn equation (22) with $n_1 = n_2 = n = 21$ discretization points (left) and wall-clock time in seconds for increasing number n of discretization points and 20 time steps up to final time $T = 0.025$ (right)

model [5, 35] for the two chemical concentrations u and v

$$\begin{cases} \partial_t u(t,x_1,x_2,x_3) = d_1 \Delta u(t,x_1,x_2,x_3) - (b+1)u(t,x_1,x_2,x_3) \\ \qquad\qquad + a + u^2(t,x_1,x_2,x_3)v(t,x_1,x_2,x_3), \\ \partial_t v(t,x_1,x_2,x_3) = d_2 \Delta v(t,x_1,x_2,x_3) \\ \qquad\qquad + bu(t,x_1,x_2,x_3) - u^2(t,x_1,x_2,x_3)v(t,x_1,x_2,x_3), \\ u(0,x_1,x_2,x_3) = 64^2 x_1^2(1-x_1)^2 x_2^2(1-x_2)^2 x_3^2(1-x_3)^2, \\ v(0,x_1,x_2,x_3) = c, \end{cases} \tag{24}$$

in the spatial domain $[0,1]^3$. The system is completed with homogeneous Neumann boundary conditions. The semidiscretization in space by finite differences yields the system

$$\begin{pmatrix} \boldsymbol{u}'(t) \\ \boldsymbol{v}'(t) \end{pmatrix} = \begin{pmatrix} K_1 & 0 \\ 0 & K_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}(t) \\ \boldsymbol{v}(t) \end{pmatrix} + \begin{pmatrix} a + \boldsymbol{u}^2(t)\boldsymbol{v}(t) \\ b\boldsymbol{u}(t) - \boldsymbol{u}^2(t)\boldsymbol{v}(t) \end{pmatrix} \iff \tag{25}$$
$$\boldsymbol{w}'(t) = \hat{K}\boldsymbol{w}(t) + \boldsymbol{g}(t,\boldsymbol{w}(t)),$$

where K_1 is the discretization matrix of the linear operator $d_1\Delta - (b+1)$ and K_2 is the discretization matrix of the linear operator $d_2\Delta$, both clearly Kronecker sums (1b). System (25) cannot be written in Kronecker form (1). However, the actions of φ-functions can still be efficiently computed by the proposed approach, since the matrix $\hat{K}$ in system (25) is block diagonal. We set $a = c = 1$, $b = 3$, $d_1 = d_2 = 0.02$, and, since we are going to employ two exponential integrators of order four, we also increase the accuracy in space by using finite differences of order four (leading to matrices K_1 and K_2 with a 13-diagonal structure), and we simulate until final time $T = 1$ employing two fourth-order exponential integrators of Runge–Kutta type.

4.4.1 An exponential Runge–Kutta scheme of order four with five stages

The first exponential integrator that we consider has reduced tableau

$$\begin{array}{c|cccc} \frac{1}{2} & & & & \\ \frac{1}{2} & \varphi_{2,3} & & & \\ 1 & \varphi_{2,4} & \varphi_{2,4} & & \\ \frac{1}{2} & a_{52} & a_{52} & \frac{1}{4}\varphi_{2,5} - a_{52} & \\ \hline & 0 & 0 & -\varphi_2 + 4\varphi_3 & 4\varphi_2 - 8\varphi_3 \end{array} \tag{26}$$

with

$$a_{52} = \frac{1}{2}\varphi_{2,5} - \varphi_{3,4} + \frac{1}{4}\varphi_{2,4} - \frac{1}{2}\varphi_{3,5},$$

see tableau (5.19) in Reference [15]. It is possible to effectively use the routine PHIKS to evaluate φ-functions applied to the same vector. In this way, the following quantities have to be computed

$$\begin{aligned} &\varphi_1(\tau\hat{K}/2^j)\boldsymbol{f}(t_n, \boldsymbol{w}_n), && && j = 0, 1, \\ &\varphi_\ell(\tau\hat{K}/2^j)\boldsymbol{d}_{n2}, && \ell = 2, 3, && j = 0, 1, \\ &\varphi_\ell(\tau\hat{K}/2^j)\boldsymbol{d}_{n3}, && \ell = 2, 3, && j = 0, 1, \\ &\varphi_\ell(\tau\hat{K}/2^j)\boldsymbol{d}_{n4}, && \ell = 2, 3, && j = 0, 1, \\ &\varphi_\ell(\tau\hat{K})\boldsymbol{d}_{n5}, && \ell = 2, 3. \end{aligned}$$

In each of the five lines, only two calls of PHIKS are needed to compute the desired actions, due to the block diagonal structure of the matrix $\hat{K}$. Indeed, in the first four lines we can obtain the actions of φ-functions simultaneously at $j = 0$ and $j = 1$, thanks to the squaring algorithm (12b). In addition, in the second, third, and fourth line, *both* φ_2 and φ_3 are produced at different scales. This is not possible for the routines PHIPM_SIMUL_IOM, KIOPS, and BAMPHI. By proceeding with a sequential implementation of the stages in tableau (26), they would require six calls with the action of the matrix $\hat{K}$, as already noticed in Reference [16]. However, we found that it is possible to alternatively assemble the stages by computing the following quantities

$$\begin{aligned} &\varphi_1(\tau\hat{K}/2^j)\boldsymbol{f}(t_n, \boldsymbol{w}_n), && j = 0, 1, \\ &\varphi_2(\tau\hat{K}/2^j)\boldsymbol{d}_{n2}, && j = 0, 1, \\ &\varphi_2(\tau\hat{K}/2^j)\boldsymbol{d}_{n3}, && j = 0, 1, \\ &\frac{\varphi_2(\tau\hat{K}/2^j)}{2^{2j}}\boldsymbol{d}_{n4} + \frac{\varphi_3(\tau\hat{K}/2^j)}{2^{3j}}(4\boldsymbol{d}_{n2} + 4\boldsymbol{d}_{n3} - 4\boldsymbol{d}_{n4}), && j = 0, 1, \\ &\varphi_2(\tau\hat{K})(4\boldsymbol{d}_{n5} - \boldsymbol{d}_{n4}) + \varphi_3(\tau\hat{K})(4\boldsymbol{d}_{n4} - 8\boldsymbol{d}_{n5}), \end{aligned} \tag{27}$$

which require again two calls to PHIKS for each of the five lines, and in total five calls to the other routines. To ensure a fair comparison, we follow approach (27) with

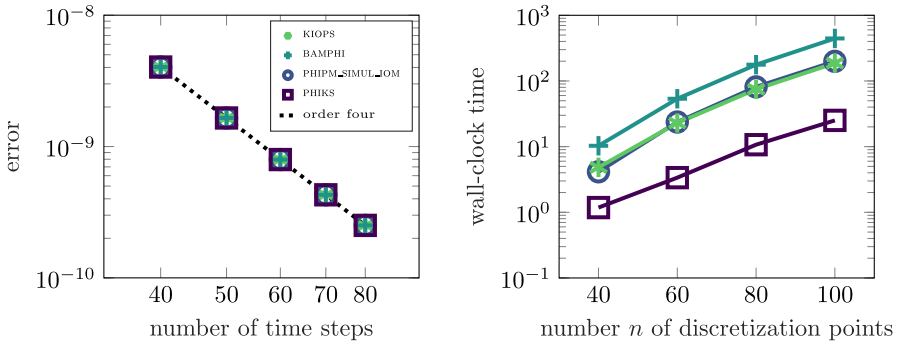

Fig. 5 Rate of convergence of the exponential Runge–Kutta scheme (26) for the semidiscretization of Brusselator model (24) with $n_1 = n_2 = n_3 = n = 11$ discretization points (left) and wall-clock time in seconds for increasing number n of discretization points and 20 time steps up to final time $T = 1$ (right)

all the routines under comparison. As we did in the previous numerical examples, we check the correct order of convergence of the integrator for different numbers of time steps and $n_1 = n_2 = n_3 = n = 20$, and we measure the wall-clock time for different numbers n of discretization points for each dimension, with n ranging from 40 to 100, leading to a maximum number of degrees of freedom $N = 2 \cdot 100^3$. The integration is performed up to the final time $T = 1$. The results are collected in Fig. 5. Again, PHIKS performs better than the other methods, being up to 7.5 times faster. The speed-up is larger than in the previous examples mainly due to the choice of the spatial discretization, that leads to denser matrices, and which affects all the routines but PHIKS (as it is based on a μ-mode approach). The insensitivity of the μ-mode approach to the density of the matrices was already pointed out in Reference [18].

4.4.2 An exponential Runge–Kutta scheme of order four with six stages

Finally, we consider the exponential Runge–Kutta integrator of order four with reduced tableau

$$\begin{array}{c|cccccc} c_2 & & & & & & \\ c_3 & \frac{c_3^2}{c_2}\varphi_{2,3} & & & & & \\ c_4 & \frac{c_4^2}{c_2}\varphi_{2,4} & & & & & \\ c_5 & 0 & a_{53} & a_{54} & & & \\ c_6 & 0 & a_{63} & a_{64} & & & \\ \hline & 0 & 0 & 0 & b_5 & b_6 \end{array} \tag{28}$$

where

$$a_{53} = \frac{c_4 c_5^2}{c_3(c_4 - c_3)}\varphi_{2,5} + \frac{2c_5^3}{c_3(c_3 - c_4)}\varphi_{3,5}, \quad a_{54} = \frac{c_3 c_5^2}{c_4(c_3 - c_4)}\varphi_{2,5} + \frac{2c_5^3}{c_4(c_4 - c_3)}\varphi_{3,5},$$
$$a_{63} = \frac{c_4 c_6^2}{c_3(c_4 - c_3)}\varphi_{2,6} + \frac{2c_6^3}{c_3(c_3 - c_4)}\varphi_{3,6}, \quad a_{64} = \frac{c_3 c_6^2}{c_4(c_3 - c_4)}\varphi_{2,6} + \frac{2c_6^3}{c_4(c_4 - c_3)}\varphi_{3,6},$$

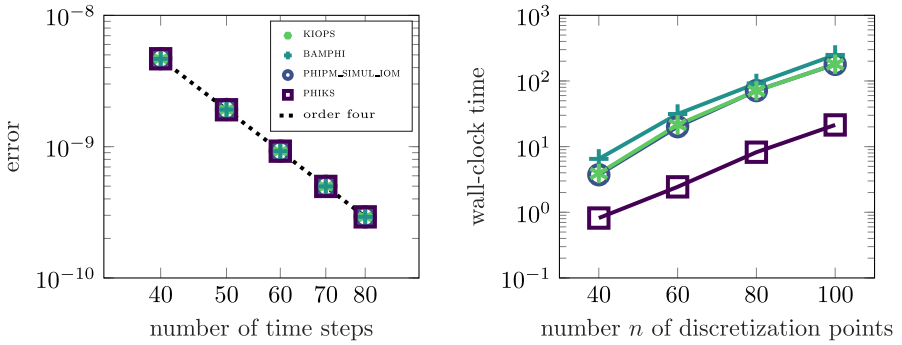

Fig. 6 Rate of convergence of the exponential Runge–Kutta scheme (28) for the semidiscretization of Brusselator model (24) with $n_1 = n_2 = n_3 = n = 11$ discretization points (left) and wall-clock time in seconds for increasing number n of discretization points and 20 time steps up to final time $T = 1$ (right)

$$b_5 = \frac{c_6}{c_5(c_6 - c_5)}\varphi_2 + \frac{2}{c_5(c_5 - c_6)}\varphi_3, \qquad b_6 = \frac{c_5}{c_6(c_5 - c_6)}\varphi_2 + \frac{2}{c_6(c_6 - c_5)}\varphi_3,$$

and $c_3 \neq c_4, c_5 \neq c_6, c_6 = (4c_5 - 3)/(6c_5 - 4)$, see equation (4.10) in Reference [16]. This integrator was designed so that multiple stages can be computed simultaneously. In fact, stages three and four and stages five and six require the same combination of φ-functions at different intermediate times, and therefore they can be efficiently implemented by the routines KIOPS, BAMPHI, and PHIPM_SIMUL_IOM (the one which has been originally employed with this integrator). Here, by selecting $c_4 = 2c_3 = 2/3$ and $c_6 = 2c_5 = 1$ we can do the same with the routine PHIKS, since the evaluation of a linear combination at a half time comes for free by the new squaring algorithm (15b). The results, by setting the remaining free coefficient $c_2 = 1/3$, are reported in Fig. 6. Since the number of calls needed by this integrator is smaller than in the previous case, all the methods turn out to be slightly faster. The speed-up of PHIKS with respect to the other routines ranges from 4.5 to 8.6, depending on the size of the problem.

5 Conclusions

In this manuscript, we proposed an efficient μ-mode approach to compute actions of φ-functions for matrices K which are Kronecker sums of any number of arbitrary matrices A_μ. This structure naturally arises when suitably discretizing in space some evolutionary PDEs of great importance in science and engineering, such as advection–diffusion–reaction, Allen–Cahn, or Brusselator equations, among the others. The corresponding stiff system of ODEs can be effectively solved by exponential integrators, which rely on the efficient approximation of the action of single φ-functions or linear combinations of them. Our new method, that we named PHIKS, approximates the integral definition of φ-functions by the Gauss–Lobatto–Legendre quadrature formula, employs scaling and squaring techniques, and computes the required actions in a μ-mode fashion by means of Tucker operators and exponentials of the small sized matrices A_μ, exploiting the efficiency of modern hardware architectures to perform

level 3 BLAS operations. Moreover, it allows to compute the desired quantities at different time scales, feature of great importance in the context of high order exponential integrators. We tested our approach on different stiff ODEs systems arising from the discretization of important PDEs in two and three space dimensions, using different exponential integrators (from stiff order one to four) and different discretization matrices (finite differences of order two or four). As terms of comparison, we considered another technique for computing actions of φ-functions of Kronecker sums of matrices (based on Sylvester equations, and currently limited to two space dimensions) and more general techniques for computing actions of φ-functions. Our method turned out to be always faster than the others, with speed-ups ranging from 1.5 to 8.6, depending on the example under consideration. The numerical experiments suggest that PHIKS is preferable to existing methods, in particular for problems with denser matrices and for exponential integrators that can be implemented by computing suitable scales of the underlying (linear combinations of) φ-functions. Interesting future developments are the application of the method to space-fractional diffusion equations [36] and second-order in time partial differential equations [37].

Acknowledgements Marco Caliari and Fabio Cassini are members of the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM).

Funding Open access funding provided by Università degli Studi di Verona within the CRUI-CARE Agreement. The authors received partial support from the Program Ricerca di Base 2019 No. RBVR199YFL of the University of Verona entitled "Geometric Evolution of Multi Agent Systems". Fabio Cassini received financial support from the Italian Ministry of University and Research (MUR) with the PRIN Project 2022 No. 2022N9BM3N "Efficient numerical schemes and optimal control methods for time-dependent partial differential equations".

Declarations

Conflict of interest The authors declare no potential Conflict of interest.

References

1. Hochbruck, M., Ostermann, A.: Exponential integrators. Acta Numer. **19**, 209–286 (2010)
2. Berland, H., Skaflestad, B., Wright, W.M.: EXPINT–A MATLAB package for exponential integrators. ACM Trans. Math. Softw. **33**(1), 4 (2007)
3. López-Fernández, M.: A quadrature based method for evaluating exponential-type functions for exponential methods. BIT Numer. Math. **50**, 631–655 (2010)
4. Li, D., Yang, S., Lan, J.: Efficient and accurate computation for the φ-functions arising from exponential integrators. Calcolo **59**(1), 11 (2022)

5. Gaudreault, S., Rainwater, G., Tokman, M.: KIOPS: A fast adaptive Krylov subspace solver for exponential integrators. J. Comput. Phys. **372**, 236–255 (2018)
6. Luan, V.T., Pudykiewicz, J.A., Reynolds, D.R.: Further development of efficient and accurate time integration schemes for meteorological models. J. Comput. Phys. **376**, 817–837 (2019)
7. Niesen, J., Wright, W.M.: Algorithm 919: A Krylov subspace algorithm for evaluating the ϕ-functions appearing in exponential integrators. ACM Trans. Math. Softw. **38**(3), 1–19 (2012)
8. Al-Mohy, A.H., Higham, N.J.: Computing the action of the matrix exponential, with an application to exponential integrators. SIAM J. Sci. Comput. **33**(2), 488–511 (2011)
9. Caliari, M., Cassini, F., Zivcovich, F.: Approximation of the matrix exponential for matrices with a skinny field of values. BIT Numer. Math. **60**(4), 1113–1131 (2020)
10. Caliari, M., Cassini, F., Zivcovich, F.: BAMPHI: Matrix and transpose free action of the combinations of φ-functions from exponential integrators. J. Comput. Appl. Math. **423**, 114973 (2023)
11. Caliari, M., Kandolf, P., Ostermann, A., Rainer, S.: The Leja method revisited: Backward error analysis for the matrix exponential. SIAM J. Sci. Comput. **38**(3), 1639–1661 (2016)
12. Caliari, M., Kandolf, P., Zivcovich, F.: Backward error analysis of polynomial approximations for computing the action of the matrix exponential. BIT Numer. Math. **58**(4), 907–935 (2018)
13. Neudecker, H.: A note on Kronecker matrix products and matrix equation systems. SIAM J. Appl. Math. **17**(3), 603–606 (1969)
14. Benzi, M., Simoncini, V.: Approximation of functions of large matrices with Kronecker structure. Numer. Math. **135**, 1–26 (2017)
15. Hochbruck, M., Ostermann, A.: Explicit exponential Runge–Kutta methods for semilinear parabolic problems. SIAM J. Numer. Anal. **43**(3), 1069–1090 (2005)
16. Luan, V.T.: Efficient exponential Runge–Kutta methods of high order: construction and implementation. BIT Numer. Math. **61**(2), 535–560 (2021)
17. Skaflestad, B., Wright, W.M.: The scaling and modified squaring method for matrix functions related to the exponential. Appl. Numer. Math. **59**(3–4), 783–799 (2009)
18. Caliari, M., Cassini, F., Einkemmer, L., Ostermann, A., Zivcovich, F.: A μ-mode integrator for solving evolution equations in Kronecker form. J. Comput. Phys. **455**, 110989 (2022)
19. Caliari, M., Cassini, F., Zivcovich, F.: A μ-mode BLAS approach for multidimensional tensor structured problems. Numer. Algorithms **92**, 2483–2508 (2023)
20. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009)
21. Dongarra, J.J., Du Croz, J., Hammarling, S., Duff, I.S.: A set of level 3 basic linear algebra subprograms. ACM Trans. Math. Softw. **16**(1), 1–17 (1990)
22. Intel Corporation: Intel Math Kernel Library. https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/onemkl.html (2024)
23. Xianyi, Z., Qian, W., Yunquan, Z.: Model-driven Level 3 BLAS Performance Optimization on Loongson 3A Processor. In: 2012 IEEE 18th International Conference on Parallel and Distributed Systems, pp. 684–691 (2012)
24. NVIDIA Corporation: cuBLAS documentation. https://docs.nvidia.com/cuda/cublas/index.html (2024)
25. Cox, S.M., Matthews, P.C.: Exponential time differencing for stiff systems. J. Comput. Phys. **176**(2), 430–455 (2002)
26. Luan, V.T., Ostermann, A.: Explicit exponential Runge–Kutta methods of high order for parabolic problems. J. Comput. Appl. Math. **256**, 168–179 (2014)
27. Croci, M., Muñoz-Matute, J.: Exploiting Kronecker structure in exponential integrators: Fast approximation of the action of φ-functions of matrices via quadrature. J. Comput. Sci. **67**, 101966 (2023)
28. Caliari, M., Zivcovich, F.: On-the-fly backward error estimate for matrix exponential approximation by Taylor algorithm. J. Comput. Appl. Math. **346**, 532–548 (2019)
29. Davis, P.J., Rabinowitz, P.: Methods of Numerical Integration, 2nd edn. Academic Press, San Diego, CA (1984)
30. Crouzeix, M., Palencia, C.: The numerical range is a $(1 + \sqrt{2})$-spectral set. SIAM J. Matrix Anal. Appl. **38**(2), 649–655 (2017)
31. Gautschi, W., Varga, R.S.: Error bounds for Gaussian quadrature of analytic functions. SIAM J. Numer. Anal. **20**(6), 1170–1186 (1983)
32. Al-Mohy, A.H., Higham, N.J.: A new scaling and squaring algorithm for the matrix exponential. SIAM J. Matrix Anal. Appl. **31**(3), 970–989 (2009)

33. Muñoz-Matute, J., Pardo, D., Calo, V.M.: Exploiting the Kronecker product structure of φ-functions in exponential integrators. Int. J. Numer. Methods Eng. **123**(9), 2142–2161 (2022)
34. Feng, X., Prohl, A.: Numerical analysis of the Allen-Cahn equation and approximation for mean curvature flows. Numer. Math. **94**(1), 33–65 (2003)
35. Uecker, H., Wetzel, D.: Snaking branches of planar BCC fronts in the 3D Brusselator. Phys. D **406**, 132383 (2020)
36. Zhao, M., Wang, H., Cheng, A.: A fast finite difference method for three-dimensional time-dependent space-fractional diffusion equations with fractional derivative boundary conditions. J. Sci. Comput. **74**, 1009–1033 (2018)
37. Phan, D., Ostermann, A.: Exponential integrators for second-order in time partial differential equations. J. Sci. Comput. **93**, 58 (2022)