BIT

Efficient exponential Runge–Kutta methods of high order: construction and implementation

Vu Thai Luan[1]

Abstract

Exponential Runge–Kutta methods have shown to be competitive for the time integration of stiff semilinear parabolic PDEs. The current construction of stiffly accurate exponential Runge–Kutta methods, however, relies on a convergence result that requires weakening many of the order conditions, resulting in schemes whose stages must be implemented in a sequential way. In this work, after showing a stronger convergence result, we are able to derive two new families of fourth- and fifth-order exponential Runge–Kutta methods, which, in contrast to the existing methods, have multiple stages that are independent of one another and share the same format, thereby allowing them to be implemented in parallel or simultaneously, and making the methods to behave like using with much less stages. Moreover, all of their stages involve only one linear combination of the product of φ-functions (using the same argument) with vectors. Overall, these features make these new methods to be much more efficient to implement when compared to the existing methods of the same orders. Numerical experiments on a one-dimensional semilinear parabolic problem, a nonlinear Schrödinger equation, and a two-dimensional Gray–Scott model are given to confirm the accuracy and efficiency of the two newly constructed methods.

Dedicated to Professor Alexander Ostermann on the occasion of his 60th birthday.

Communicated by Mechthild Thalhammer.

✉ Vu Thai Luan
luan@math.msstate.edu

[1] Department of Mathematics and Statistics, Mississippi State University, 410 Allen Hall, 175 President's Circle, Mississippi, MS 39762, USA

✌ Springer

1 Introduction

In this paper, we are concerned with the construction and implementation of new efficient exponential Runge–Kutta integrators for solving stiff parabolic PDEs. These PDEs, upon their spatial discretizations, can be cast in the form of semilinear problems

$$u'(t) = Au(t) + g(t, u(t)) = F(t, u(t)), \qquad u(t_0) = u_0, \tag{1.1}$$

where the linear part Au usually causes stiffness. The nonlinearity $g(t, u)$ is assumed to satisfy a local Lipschitz condition in a strip along the exact solution.

Exponential Runge–Kutta methods are a popular class of exponential integrators [9], which have shown a great promise as an alternative to standard time integration solvers for stiff systems and applications in recent years, see e.g. [8,10–20,22]. The main idea behind these methods is to solve the linear portion of (1.1) exactly and integrate the remaining nonlinear portion explicitly based on a representation of the exact solution using the variation-of-constants formula.

A s-stage explicit exponential Runge–Kutta (expRK) method [8] applied to (1.1) can be reformulated (see [15,17]) as

$$U_{ni} = u_n + c_i h\varphi_1(c_i hA)F(t_n, u_n) + h \sum_{j=2}^{i-1} a_{ij}(hA)D_{nj}, \ 2 \le i \le s, \tag{1.2a}$$

$$u_{n+1} = u_n + h\varphi_1(hA)F(t_n, u_n) + h \sum_{i=2}^{s} b_i(hA)D_{ni}, \tag{1.2b}$$

where

$$D_{ni} = g(t_n + c_i h, U_{ni}) - g(t_n, u_n), \qquad 2 \le i \le s. \tag{1.3}$$

Here, U_{ni} denote the internal stages that approximate $u(t_n + c_i h)$ using the time step size $h = t_{n+1} - t_n > 0$ and nodes c_i. By construction, the coefficients $a_{ij}(z)$ and $b_i(z)$ are usually linear combinations of the entire functions

$$\varphi_k(z) = \int_0^1 e^{(1-\theta)z} \frac{\theta^{k-1}}{(k-1)!} \, d\theta, \quad k \ge 1 \tag{1.4}$$

and their scaled versions $\varphi_k(c_i z)$.

A common approach that has been used to determine the unknown matrix functions $a_{ij}(hA)$ and $b_i(hA)$ is to expand them as $a_{ij}(hA) = \sum_{k\ge 0} \alpha_{ij}^{(k)}(hA)^k$, $b_i(hA) = \sum_{k\ge 0} \beta_i^{(k)}(hA)^k$ (e.g. using classical Taylor series expansions) to obtain order conditions. Clearly, the boundedness of the remainder terms of these expansions (and thus the error terms) are dependent of $\|A\|$. Due to stability reasons, such resulting methods might not be suitable for integrating stiff PDEs, which A typically has a large norm or is even unbounded operator. These methods are thus usually referred as classical (non-stiffly accurate) expRK methods. Unlike this approach, in a seminal

contribution [8], Hochbruck and Ostermann derived a new error expansion with the remainder terms that are bounded independently of the stiffness (i.e. not involving the powers of A), leading to stiff order conditions, which give rise to the construction of stiffly accurate expRK methods of orders up to four. Following this, in [14] Luan and Ostermann developed a systematic theory of deriving stiff order conditions for expRK methods of arbitrary order, thereby allowing the construction of a fifth-order method in [15].

In view of the existing stiffly accurate expRK methods in the literature, we observe that they were derived based on a convergence result that requires weakening many of the stiff order conditions (in order to minimize the number of required stages s and matrix functions used in each internal stages U_{ni}). As a result, their structures contain internal stages U_{ni} that are dependent of the preceding stages, implying that such methods must be implemented by computing each of these stages sequentially. Also, the very last stages usually involve several different linear combinations of $\varphi_k(c_i h A)$-functions (using different nodes c_i in their arguments) acting on different sets of vectors. This would introduce additional computational effort for these stages. For more details, we refer to Sects. 2 and 5.

Motivated by the observations above, in this work we show a stronger convergence result for expRK methods up to order five which requires weakening only one order condition (thereby could improve the stability and accuracy) and offers more degree of freedoms in solving order conditions. Using this result and inspired by our recent algorithm, `phipm_simul_iom`, proposed in [19] (which allows one to simultaneously compute multiple linear combinations of φ- functions acting on a same set of vectors), we construct new methods of orders 4 and 5 which involve only one linear combination of φ- functions for each stage and have multiple internal stages U_{ni} that are independent of one another, thereby allowing them to be computed in parallel. Furthermore, one can derive these independent stages in a way that they share the same form of linear combination of $\varphi_k(c_i h A)$- functions acting on the same set of vectors, allowing them to be implemented simultaneously (by one evaluation). While these independent states can be computed in parallel (as mentioned above) by any algorithm which approximates the action of (the linear combination of) φ- functions, we note that the possibility to compute them simultaneously is a new feature that can be used with our algorithm `phipm_simul_iom` (other algorithms, e.g., that do not require the construction of Krylov subspaces, might not support computing these stages simultaneously). Overall, this makes the new methods to behave like methods using much less number of stages (even when compared to the existing methods of the same orders), meaning that they require much less number of evaluations for linear combinations of φ- functions, and are thus more efficient.

The paper is organized as follows. In Sect. 2, we describe our motivation, propose new ideas, and review the existing expRK methods in the literature with respect to these ideas. Following this, in Sect. 3 we prove a stronger convergence result (Theorem 3.1) for expRK methods, which requires relaxing only one order condition. This allows us to construct more efficient methods in Sect. 4. In particular, we are able to derive two new families of fourth- and fifth- order stiffly accurate expRK methods called `expRK4s6` (4th-order 6-stage but requires 4 independent stage evaluation only) and `expRK5s10` (5th-order 10-stage but requires 5 independent stage evaluation only), respectively. In

Sect. 5, we present details implementation of these two new methods, as well as the existing stiffly accurate expRK schemes of the same orders (for comparison). In the last section, numerical examples including one and two-dimensional stiff PDEs are presented to demonstrate the accuracy and efficiency of the two newly constructed expRK integrators.

2 Motivation and existing methods

In this section, we start our motivation by taking a closer look at an efficient way for implementing expRK methods (1.2). Then, we propose some ideas to derive more efficient methods with respect to this efficient implementation along with reviewing the current methods.

2.1 An efficient way of implementation

Clearly, each stage (U_{ni} or u_{n+1}) of (1.2) requires computing matrix functions of the form $\varphi_k(c_i h A)v_k$ ($0 < c_i \leq 1$), where v_k is some vector (could be $F(t_n, u_n)$, D_{ni} or a linear combination of these). Thanks to recent developments [1,4,6,21], one can efficiently compute a linear combination of φ-functions acting on a set of input vectors $V_0, \ldots, V_q$

$$\varphi_0(M)V_0 + \varphi_1(M)V_1 + \varphi_2(M)V_2 + \cdots + \varphi_q(M)V_q, \tag{2.1}$$

where M is some square matrix. This is crucial when implementing exponential integrators. Very recently, in [19], we were able to improve the implementations presented in [6,21], resulting in the routine `phipm_simul_iom`. The underlying method in this algorithm is the use of an adaptive time-stepping technique combined with Krylov subspace methods, which allows us to simultaneously compute multiple linear combinations of type (2.1) using different scaling factors $\rho_1, \cdots, \rho_r$ of M, i.e.,

$$\varphi_0(\rho_1 M)V_0 + \varphi_1(\rho_1 M)V_1 + \varphi_2(\rho_1 M)V_2 + \cdots + \varphi_N(\rho_1 M)V_q,$$
$$\vdots \tag{2.2}$$
$$\varphi_0(\rho_r M)V_0 + \varphi_1(\rho_r M)V_1 + \varphi_2(\rho_r M)V_2 + \cdots + \varphi_N(\rho_r M)V_q.$$

Now taking $M = hA$ and considering ρ_k ($1 \leq k \leq r$) as nodes c_i used in expRK methods immediately suggests that one can compute the following $(s-1)$ linear linear combinations

$$\varphi_1(c_i h A)V_1 + \varphi_2(c_i h A)V_2 + \cdots + \varphi_N(c_i h A)V_q, \quad 2 \leq i \leq s \tag{2.3}$$

simultaneously by using only one evaluation (i.e., one call to `phipm_simul_iom`). Note that this requires the use of a *same* set of vectors $[V_1, \ldots, V_q]$ for all the linear combinations in (2.3).

Motivated by this, we see that if a s-stage expRK scheme (1.2) is constructed in such a way that each internal stage U_{ni} has the form

$$U_{ni} = u_n + \varphi_1(c_i h A)V_{1i} + \varphi_2(c_i h A)V_{2i} + \cdots + \varphi_N(c_i h A)V_{qi}, \qquad (2.4)$$

which includes only one linear combination of φ- functions using *exactly* node c_i as an argument in all φ_k functions, then the scheme will contain a total of s such linear combinations ($s - 1$ for U_{ni} and 1 for u_{n+1} as (1.2b) can be always written in the form of (2.4) with $c_i = 1$), thereby requiring s evaluations only. Furthermore, since the sets of vectors $[V_{1i}, V_{2i}, \cdots, V_{qi}]$ in (2.4) are usually different for each U_{ni}, (2.3) also suggests that the efficiency will be significantly increased if one could build such stages (or a group of) U_{ni} of the form (2.4) that share the same format (i.e., having the same set of acting vectors $V_{1i} \equiv V_1, \ldots, V_{qi} \equiv V_q$) or that are independent of one another. As this allows to compute such stages simultaneously by one evaluation or to implement them in parallel similarly to our construction of parallel exponential Rosenbrock methods [18]), it certainly reduces the total number of required evaluations and thus speedups the computing time.

With respect to these observations, we now review the existing expRK schemes in the literature. Since our focus is on stiff problems, we will discuss only on stiffly accurate expRK methods, meaning that they satisfy the stiff order conditions (see Sect. 3 below).

2.2 Existing schemes and remarks

In [8], expRK methods of orders up to four have been derived. For later reference, we name the second-order, the third-order, and the fourth-order methods in that work as `expRK2s2`, `expRK3s3`, and `expRK4s5`, respectively. In [15], we have constructed an expRK method of order five called `expRK5s8`. To discuss all of these schemes in terms of the implementation, we rewrite their internal stages U_{ni} and u_{n+1} as linear combinations of φ- functions like (2.4) and display them as follows (Note that since the first-order method, the exponential Euler scheme $u_{n+1} = u_n + \varphi_1(hA)hF(t_n, u_n)$, has no internal stage, we do not consider it here).
`expRK2s2`:

$$\begin{aligned}
U_{n2} &= u_n + \varphi_1(c_2 h A)c_2 h F(t_n, u_n), \\
u_{n+1} &= u_n + \varphi_1(h A)h F(t_n, u_n) + \varphi_2(h A)\frac{1}{c_2}h D_{n2}.
\end{aligned} \qquad (2.5)$$

`expRK3s3` (a representative with $c_2 \neq \frac{2}{3}$):

$$\begin{aligned}
U_{n2} &= u_n + \varphi_1(c_2 h A)c_2 h F(t_n, u_n), \\
U_{n3} &= u_n + \varphi_1\left(\tfrac{2}{3}h A\right)\tfrac{2}{3}h F(t_n, u_n) + \varphi_2\left(\tfrac{2}{3}h A\right)\tfrac{4}{9c_2}h D_{n2}, \\
u_{n+1} &= u_n + \varphi_1(h A)h F(t_n, u_n) + \varphi_2(h A)\tfrac{3}{2}h D_{n2}.
\end{aligned} \qquad (2.6)$$

`expRK4s5` (the only existing fourth-order stiffly accurate expRK method [8]):

$$U_{n2} = u_n + \varphi_1\left(\tfrac{1}{2}hA\right)\tfrac{1}{2}hF(t_n, u_n),$$

$$U_{n3} = u_n + \varphi_1\left(\tfrac{1}{2}hA\right)\tfrac{1}{2}hF(t_n, u_n) + \varphi_2\left(\tfrac{1}{2}hA\right)hD_{n2},$$

$$U_{n4} = u_n + \varphi_1(hA)hF(t_n, u_n) + \varphi_2(hA)h(D_{n2} + D_{n3}),$$

$$U_{n5} = u_n + \left[\varphi_1\left(\tfrac{1}{2}hA\right)\tfrac{1}{2}hF(t_n, u_n) + \varphi_2\left(\tfrac{1}{2}hA\right)\tfrac{1}{4}h(2D_{n2} + 2D_{n3} - D_{n4})\right.$$

$$\left. + \varphi_3\left(\tfrac{1}{2}hA\right)\tfrac{1}{2}h(-D_{n2} - D_{n3} + D_{n4})\right] + \left[\varphi_2(hA)\tfrac{1}{4}h(D_{n2} + D_{n3} - D_{n4})\right.$$

$$\left. + \varphi_3(hA)h(-D_{n2} - D_{n3} + D_{n4})\right],$$

$$u_{n+1} = u_n + \varphi_1(hA)hF(t_n, u_n) + \varphi_2(hA)h(-D_{n4} + 4D_{n5}) + \varphi_3(hA)h(4D_{n4} - 8D_{n5}).$$

$$(2.7)$$

`expRK5s8` (the only existing fifth-order stiffly accurate expRK method [15]):

$$U_{n2} = u_n + \varphi_1\left(\tfrac{1}{2}hA\right)\tfrac{1}{2}hF(t_n, u_n),$$

$$U_{n3} = u_n + \varphi_1\left(\tfrac{1}{2}hA\right)\tfrac{1}{2}hF(t_n, u_n) + \varphi_2\left(\tfrac{1}{2}hA\right)\tfrac{1}{2}hD_{n2},$$

$$U_{n4} = u_n + \varphi_1\left(\tfrac{1}{4}hA\right)\tfrac{1}{4}hF(t_n, u_n) + \varphi_2\left(\tfrac{1}{4}hA\right)\tfrac{1}{8}hD_{n3},$$

$$U_{n5} = u_n + \varphi_1\left(\tfrac{1}{2}hA\right)\tfrac{1}{2}hF(t_n, u_n) + \varphi_2\left(\tfrac{1}{2}hA\right)\tfrac{1}{2}h(-D_{n3} + 4D_{n4})$$

$$+ \varphi_3\left(\tfrac{1}{2}hA\right)h(2D_{n3} - 4D_{n4})$$

$$U_{n6} = u_n + \varphi_1\left(\tfrac{1}{5}hA\right)\tfrac{1}{5}hF(t_n, u_n) + \varphi_2\left(\tfrac{1}{5}hA\right)\tfrac{1}{25}h(8D_{n4} - 2D_{n5})$$

$$+ \varphi_3\left(\tfrac{1}{5}hA\right)\tfrac{1}{125}h(-32D_{n4} + 16D_{n5}),$$

$$U_{n7} = u_n + \left[\varphi_1\left(\tfrac{2}{3}hA\right)\tfrac{2}{3}hF(t_n, u_n) + \varphi_2\left(\tfrac{2}{3}hA\right)h\left(\tfrac{-16}{27}D_{n5} + \tfrac{100}{27}D_{n6}\right)\right.$$

$$\left. + \varphi_3\left(\tfrac{2}{3}hA\right)h\left(\tfrac{320}{81}D_{n5} - \tfrac{800}{81}D_{n6}\right)\right]$$

$$+ \left[\varphi_2\left(\tfrac{1}{5}hA\right)h\left(\tfrac{-20}{81}D_{n4} + \tfrac{5}{243}D_{n5} + \tfrac{125}{486}D_{n6}\right)\right.$$

$$\left. + \varphi_3\left(\tfrac{1}{5}hA\right)h\left(\tfrac{16}{81}D_{n4} - \tfrac{4}{243}D_{n5} - \tfrac{50}{243}D_{n6}\right)\right],$$

$$U_{n8} = u_n + \left[\varphi_1(hA)hF(t_n, u_n) + \varphi_2(hA)h\left(\tfrac{-16}{3}D_{n5} + \tfrac{250}{21}D_{n6} + \tfrac{27}{14}D_{n7}\right)\right.$$

$$+ \varphi_3(hA)h\left(\tfrac{208}{3}D_{n5} - \tfrac{250}{3}D_{n6} - 27D_{n7}\right)$$

$$\left. + \varphi_4(hA)h\left(-240D_{n5} + \tfrac{1500}{7}D_{n6} + \tfrac{810}{7}D_{n7}\right)\right]$$

$$+ \left[\varphi_2\left(\tfrac{1}{5}hA\right)h\left(\tfrac{-4}{7}D_{n5} + \tfrac{25}{49}D_{n6} + \tfrac{27}{98}D_{n7}\right)\right.$$

$$+ \varphi_3\left(\tfrac{1}{5}hA\right)h\left(\tfrac{8}{5}D_{n5} - \tfrac{10}{7}D_{n6} - \tfrac{27}{35}D_{n7}\right)$$

$$\left. + \varphi_4\left(\tfrac{1}{5}hA\right)h\left(\tfrac{-48}{35}D_{n5} + \tfrac{60}{49}D_{n6} + \tfrac{162}{245}D_{n7}\right)\right]$$

$$+ \left[\varphi_2\left(\tfrac{2}{3}hA\right)h\left(\tfrac{-288}{35}D_{n5} + \tfrac{360}{49}D_{n6} + \tfrac{972}{245}D_{n7}\right)\right.$$

$$+ \varphi_3\left(\tfrac{2}{3}hA\right)h\left(\tfrac{384}{5}D_{n5} - \tfrac{480}{7}D_{n6} - \tfrac{1296}{35}D_{n7}\right)$$

$$\left. + \varphi_4\left(\tfrac{2}{3}hA\right)h\left(\tfrac{-1536}{7}D_{n5} + \tfrac{9600}{49}D_{n6} + \tfrac{5184}{49}D_{n7}\right)\right],$$

$$u_{n+1} = u_n + \varphi_1(hA)hF(t_n, u_n) + \varphi_2(hA)h\left(\tfrac{125}{14}D_{n6} - \tfrac{27}{14}D_{n7} + \tfrac{1}{2}D_{n8}\right)$$

$$+\varphi_3(hA)h\left(\frac{-625}{14}D_{n6}+\frac{162}{7}D_{n7}-\frac{13}{2}D_{n8}\right)$$
$$+\varphi_4(hA)h\left(\frac{1125}{14}D_{n6}-\frac{405}{7}D_{n7}+\frac{45}{2}D_{n8}\right).$$

Remark 2.1 In view of the structures of these schemes, one can see that only the second- and third-oder schemes (expRK2s2, expRK3s3) have all U_{ni} in the form (2.4). While expRK2s2 requires one internal stage U_{n2}, expRK3s3 needs two internal stages with U_{n3} depends on U_{n2}, making these stages cannot be computed simultaneously. As for expRK4s5, to the best of our knowledge, this 5-stage scheme is the only existing fourth-order stiffly accurate expRK method. As seen, among its internal stages the three internal stages U_{n2}, U_{n3}, and U_{n4} are of the form (2.4) but again their corresponding sets of vectors $[V_{ki}]$ are not the same ($[V_{k2}]=[\frac{1}{2}hF(t_n,u_n)]$, $[V_{k3}]=[\frac{1}{2}hF(t_n,u_n),hD_{n2}]$, $[V_{k4}]=[hF(t_n,u_n),h(D_{n2}+D_{n3})]$), and because of (1.3), they are not independent of one another (U_{n4} and U_{n3} depend on their preceding stages). Therefore one needs 3 sequential evaluations for computing these three stages. Also, we note that the last internal stage U_{n5} depends on all of its preceding stages and involves two different linear combinations of φ_k- functions with different scaling factors $c_5=\frac{1}{2}$ and $c_4=1$, namely, $\sum_k\varphi_k(\frac{1}{2}hA)V_k$ and $\sum_k\varphi_k(hA)W_k$ (grouped in two different brackets [ ]), which has to be implemented by 2 separate evaluations. The final stage u_{n+1} depends on U_{n4} and U_{n5}. As a result, this scheme must be implemented in a sequential way, which requires totally 6 evaluations for 6 different linear combinations. Similarly, to the best of our knowledge, expRK5s8 is also the only existing fifth-order stiffly accurate expRK methods. From the construction of this scheme [15], one needs 8 stages. Among them, the first five internal stages are of the form (2.4). We note, however, that the last two internal stages U_{n7} and U_{n8} involves 2 and 3 different linear combinations (grouped in different brackets [ ]) of φ_k- functions (with different scaling factors) acting on different sets of vectors. And each stage (U_{ni} or u_{n+1}) depends on all the preceding stages (except for the first stage U_{n2}). Thus, this scheme must be also implemented in a sequential way (it also does not have any group of internal stages that can be computed simultaneously). Clearly, it requires totally 11 evaluations (11 different linear combinations of φ functions).

Remark 2.2 The resulting structures of the expRK schemes discussed in Remark 2.1 can be explained by taking a closer look at their constructions presented in [8,15]. Namely, these methods have been analyzed and derived by using a weakened convergence result, i.e., weakening of many order conditions in order to minimize the number of required stages s and the number of matrix functions in each internal stage U_{ni}. Specifically, for fourth-order methods (e.g., expRK4s5) 4 out of 9 order conditions have to be relaxed and for fifth-order methods (e.g., expRK5s8) 9 out of 16 order conditions have to be relaxed. As a trade off, each stage of these methods depends on the preceding stages (thus the resulting schemes must be implemented by computing each stage sequentially) and the very last stages usually involve different linear combinations of φ_k-functions (with several different nodes c_i as scaling factors) acting on not the same set of vectors, which then require additional sequential evaluations. For more details, see Sect. 4 below.

3 Stiff order conditions and convergence analysis

Inspired by the motivation and remarks in Sect. 2, we next present a stronger convergence result which later allows a construction of new efficient methods of high order. For this, we first recall the stiff order conditions for expRK methods up to order 5 (see [15,17]).

3.1 Stiff order conditions for methods up to order 5

Let $\tilde{e}_{n+1} = \hat{u}_{n+1} - u(t_{n+1})$ denote the local error of (1.2), i.e., the difference between the numerical solution $\hat{u}_{n+1}$ obtained by (1.2) after one step starting from the 'initial condition' $u(t_n)$ and the corresponding exact solution $u(t_{n+1})$ of (1.1) at t_{n+1}.

To simplify the notation in this section we set $f(t) = g(t, u(t))$ as done in [8], and additionally denote $G_{k,n} = D^k g(t_n, u(t_n))$ be the k-th partial Fréchet derivative (with respect to u) evaluated at $u(t_n)$. Our results in [17] (Sect. 4.2) or [14] (Sect. 5.1) showed that

$$
\tilde{e}_{n+1} = h^2 \psi_2(hA) f'(t_n) + h^3 \psi_3(hA) f''(t_n) + h^4 \psi_4(hA) f'''(t_n) + h^5 \psi_5(hA) f^{(4)}(t_n) \tag{3.1}
$$
$$
+ \mathbf{R}_n + \mathcal{O}(h^6)
$$

with the remaining terms

$$
\begin{aligned}
\mathbf{R}_n = {}& h^3 \sum_{i=2}^{s} b_i G_{1,n} \psi_{2,i} \, f'(t_n) + h^4 \sum_{i=2}^{s} b_i G_{1,n} \psi_{3,i} \, f''(t_n) \\
& + h^4 \sum_{i=2}^{s} b_i G_{1,n} \sum_{j=2}^{i-1} a_{ij} G_{1,n} \psi_{2,j} \, f'(t_n) \\
& + h^4 \sum_{i=2}^{s} b_i c_i G_{2,n} \big(u'(t_n), \psi_{2,i} \, f'(t_n)\big) + h^5 \sum_{i=2}^{s} b_i G_{1,n} \psi_{4,i} \, f'''(t_n) \\
& + h^5 \sum_{i=2}^{s} b_i G_{1,n} \sum_{j=2}^{i-1} a_{ij} G_{1,n} \psi_{3,j} \, f''(t_n) \\
& + h^5 \sum_{i=2}^{s} b_i G_{1,n} \sum_{j=2}^{i-1} a_{ij} G_{1,n} \sum_{k=2}^{j-1} a_{jk} G_{1,n} \psi_{2,k} \, f'(t_n) \\
& + h^5 \sum_{i=2}^{s} b_i G_{1,n} \sum_{j=2}^{i-1} a_{ij} c_j G_{2,n} \big(u'(t_n), \psi_{2,j} \, f'(t_n)\big) \\
& + h^5 \sum_{i=2}^{s} b_i c_i G_{2,n} \big(u'(t_n), \psi_{3,i} \, f''(t_n)\big)
\end{aligned}
$$

$$+ h^5 \sum_{i=2}^{s} b_i c_i G_{2,n}\left(u'(t_n), \sum_{j=2}^{i-1} a_{ij} G_{1,n} \psi_{2,j}\, f'(t_n)\right)$$

$$+ h^5 \sum_{i=2}^{s} \frac{b_i}{2!} G_{2,n}\left(\psi_{2,i}\, f'(t_n), \psi_{2,i}\, f'(t_n)\right)$$

$$+ h^5 \sum_{i=2}^{s} b_i \frac{c_i^2}{2!} G_{2,n}\left(u''(t_n), \psi_{2,i}\, f'(t_n)\right)$$

$$+ h^5 \sum_{i=2}^{s} b_i \frac{c_i^2}{2!} G_{3,n}\left(u'(t_n), u'(t_n), \psi_{2,i}\, f'(t_n)\right). \tag{3.2}$$

Here, (and from now on) we use the abbreviations $a_{ij} = a_{ij}(hA), b_i = b_i(hA), \varphi_{j,i} = \varphi_j(c_i hA)$ and

$$\psi_j(hA) = \sum_{i=2}^{s} b_i \frac{c_i^{j-1}}{(j-1)!} - \varphi_j(hA), \quad j \geq 2 \tag{3.3a}$$

$$\psi_{j,i} = \psi_{j,i}(hA) = \sum_{k=2}^{i-1} a_{ik} \frac{c_k^{j-1}}{(j-1)!} - c_i^j \varphi_{j,i}. \tag{3.3b}$$

Requiring a local error truncation $\tilde{e}_{n+1} = \mathcal{O}(h^6)$ results in the stiff order conditions for methods of order up to 5, which are displayed in Table 1 below.

3.2 A stronger convergence result

The convergence analysis of exponential Runge–Kutta methods is usually performed in the framework of analytic semigroups on a Banach space X with the following assumptions (see e.g. [8,15]):

Assumption 1 *The linear operator A is the infinitesimal generator of an analytic semigroup* e^{tA} *on X. This implies that*

$$\|e^{tA}\|_{X \leftarrow X} \leq C, \quad t \geq 0 \tag{3.4}$$

and consequently $\varphi_k(hA)$, the coefficients $a_{ij}(hA)$ and $b_i(hA)$ of the method are bounded operators. Furthermore, the following stability bound (see [8, Lemma 1])

$$\left\| hA \sum_{j=1}^{n} e^{jhA} \right\|_{X \leftarrow X} \leq C \tag{3.5}$$

holds uniformly for all $n \geq 1$ and $h > 0$ with $0 < nh \leq T - t_0$.

Table 1 Stiff order conditions for explicit exponential Runge–Kutta methods up to order 5. The variables Z, J, K, L denote arbitrary square matrices, and B an arbitrary bilinear mapping of appropriate dimensions. The functions ψ_l and $\psi_{k,l}$ are defined in (3.3)

No.	Stiff order condition	Order
1	$\psi_2(Z) = 0 \iff \sum_{i=2}^{s} b_i(Z) c_i = \varphi_2(Z)$	2
2	$\psi_3(Z) = 0 \iff \sum_{i=2}^{s} b_i(Z) \frac{c_i^2}{2!} = \varphi_3(Z)$	3
3	$\sum_{i=2}^{s} b_i(Z) J \psi_{2,i}(Z) = 0$	3
4	$\psi_4(Z) = 0 \iff \sum_{i=2}^{s} b_i(Z) \frac{c_i^3}{3!} = \varphi_4(Z)$	4
5	$\sum_{i=2}^{s} b_i(Z) J \psi_{3,i}(Z) = 0$	4
6	$\sum_{i=2}^{s} b_i(Z) J \sum_{j=2}^{i-1} a_{ij}(Z) J \psi_{2,j}(Z) = 0$	4
7	$\sum_{i=2}^{s} b_i(Z) c_i K \psi_{2,i}(Z) = 0$	4
8	$\psi_5(Z) = 0 \iff \sum_{i=2}^{s} b_i(Z) \frac{c_i^4}{4!} = \varphi_5(Z)$	5
9	$\sum_{i=2}^{s} b_i(Z) J \psi_{4,i}(Z) = 0$	5
10	$\sum_{i=2}^{s} b_i(Z) J \sum_{j=2}^{i-1} a_{ij}(Z) J \psi_{3,j}(Z) = 0$	5
11	$\sum_{i=2}^{s} b_i(Z) J \sum_{j=2}^{i-1} a_{ij}(Z) J \sum_{k=2}^{j-1} a_{jk}(Z) J \psi_{2,k}(Z) = 0$	5
12	$\sum_{i=2}^{s} b_i(Z) J \sum_{j=2}^{i-1} a_{ij}(Z) c_j K \psi_{2,j}(Z) = 0$	5
13	$\sum_{i=2}^{s} b_i(Z) c_i K \psi_{3,i}(Z) = 0$	5
14	$\sum_{i=2}^{s} b_i(Z) c_i K \sum_{j=2}^{i-1} a_{ij}(Z) J \psi_{2,j}(Z) = 0$	5
15	$\sum_{i=2}^{s} b_i(Z) B\big(\psi_{2,i}(Z), \psi_{2,i}(Z)\big) = 0$	5
16	$\sum_{i=2}^{s} b_i(Z) c_i^2 L \psi_{2,i}(Z) = 0$	5

Assumption 2 *(for high-order methods). The solution $u : [t_0, T] \to X$ of* (1.1) *is sufficiently smooth with derivatives in X and $g : [t_0, T] \to X$ is sufficiently often Fréchet differentiable in a strip along the exact solution. All occurring derivatives are assumed to be uniformly bounded.*

Let $e_{n+1} = u_{n+1} - u(t_{n+1})$ denote the global error at time t_{n+1}. In [15], we have shown that e_n satisfies the recursion

$$e_n = h \sum_{j=0}^{n-1} e^{(n-j)hA} \mathcal{K}_j(e_j) e_j + \sum_{j=0}^{n-1} e^{jhA} \tilde{e}_{n-j}, \tag{3.6}$$

where $\mathcal{K}_j(e_j)$ are bounded operators on X.

Motivated by Remark 2.2, we now give a stronger convergence result (compared to those results given in [8,15]) in the sense that it requires relaxing only one order condition.

Theorem 3.1 (Convergence) *Let the initial value problem* (1.1) *satisfy Assumptions 1–2. Consider for its numerical solution an explicit exponential Runge–Kutta method* (1.2) *that fulfills the order conditions of Table* 1 *up to order p ($2 \le p \le 5$) in a strong form with the exception that only one condition $\psi_p(Z) = 0$ holds in a weakened*

form, i.e., $\psi_p(0) = 0$. Then, the method is convergent of order p. In particular, the numerical solution u_n satisfies the error bound

$$\|u_n - u(t_n)\| \leq Ch^p \tag{3.7}$$

uniformly on compact time intervals $t_0 \leq t_n = t_0 + nh \leq T$ with a constant C that depends on $T - t_0$, but is independent of n and h.

Proof The proof can be carried out in a very similar way as done in [15, Theorem 4.2]. In view of (3.1) and (3.2) and employing the assumptions of Theorem 3.1 on the order conditions, we have $\mathbf{R}_n = 0$ and thus

$$\tilde{e}_{n+1} = h^p \big(\psi_p(hA) - \psi_p(0) \big) G_{p-1,n} + h^{p+1} \mathbf{S}_n, \tag{3.8}$$

where $G_{p-1,n}$ is defined in Sect. 3.1 and $\mathbf{S}_n$ involves the terms multiplying h^{p+1} and higher order in (3.1) (clearly, $\|\mathbf{S}_n\| \leq C$). Inserting (3.8) (with index $n - j - 1$ in place of n) into (3.6) and using the fact that there exists a bounded operator $\tilde{\psi}_p(hA)$ such that $\psi_p(hA) - \psi_p(0) = \tilde{\psi}_p(hA)hA$ yields

$$e_n = h \sum_{j=0}^{n-1} e^{(n-j)hA} \mathcal{K}_j(e_j)e_j + h^p \sum_{j=0}^{n-1} hA e^{jhA} \tilde{\psi}_p(hA) \mathbf{G}_{n-j-1,p-1} + h^{p+1} \mathbf{S}_{n-j-1}. \tag{3.9}$$

Using (3.4), (3.5) and an application of a discrete Gronwall lemma shows (3.7). □

With the result of Theorem 3.1 in hand, we are now ready to derive more efficient methods. In particular, we will solve the system of stiff order conditions of Table 1 in the context of Theorem 3.1. It turns out that for methods of high order this will require an increase in the number of stages s. However, we will have more degree of freedoms for constructing our desired methods as seen in Sect. 4 below. In addition, by relaxing only one order condition, we expect methods resulted from Theorem 3.1 to have better stability (and thus may be more accurate) when integrating stiff systems (see Sect. 6).

4 Derivation of new efficient exponential Runge–Kutta methods

In this section, we will derive methods which have the following features: (i) containing multiple internal stages U_{ni} that are independent of each other (henceforth called *parallel stages*) and share the same format (thereby allowing them to be implemented in parallel); (ii) involving less number of evaluations of the form (2.4) when compared to the existing methods of the same orders (thus behaving like methods that use fewer number of stages s).

We first start with methods of order $p \leq 3$. When solving order conditions for these methods (requiring at least $s = 2$ and $s = 3$ for second- and third-order methods, respectively), one can easily show that it is not possible to fulfill the desired feature

(ii), particularly when comparing with `expRK2s2` (order 2, 2-stage) and `expRK3s3` (order 3, 3-stage) mentioned in Sect. 2. We omit the details. Therefore, we will focus on the derivation of new methods of higher orders, namely, orders 4 and 5.

4.1 A family of fourth-order methods with parallel stages

Deriving methods of order 4 requires solving the set of 7 stiff order conditions 1–7 in Table 1. First, we discuss on the required number of stages s. It is shown in [8, Sect.5.3] that $s = 5$ is the minimal number of stages required to construct a family of fourth-order methods which satisfies conditions 1–3 in the strong sense and conditions 4–7 in the weakened form (relaxing $b_i(Z)$ as $b_i(0)$). In other words, with $s = 5$ it is not possible to fulfill the order conditions in the context of Theorem 3.1, which requires only condition 4 holds in a weakened form $\psi_4(0) = 0$ or equivalently $\sum_{i=2}^{s} b_i(0) \frac{c_i^3}{3!} = \varphi_4(0) = 1/24$. Therefore, we consider $s = 6$. In this case, conditions 1, 2, and the weakened condition 4 are

$$b_2 c_2 + b_3 c_3 + b_4 c_4 + b_5 c_5 + b_6 c_6 = \varphi_2, \tag{4.1a}$$

$$b_2 c_2^2 + b_3 c_3^2 + b_4 c_4^2 + b_5 c_5^2 + b_6 c_6^2 = 2\varphi_3, \tag{4.1b}$$

$$b_2(0) c_2^3 + b_3(0) c_3^3 + b_4(0) c_4^3 + b_5(0) c_5^3 + b_6(0) c_6^3 = 6\varphi_4(0) = 1/4, \tag{4.1c}$$

and conditions 3, 5, 7 and 6 are

$$b_2 J \psi_{2,2} + b_3 J \psi_{2,3} + b_4 J \psi_{2,4} + b_5 J \psi_{2,5} + b_6 J \psi_{2,6} = 0, \tag{4.2a}$$

$$b_2 J \psi_{3,2} + b_3 J \psi_{3,3} + b_4 J \psi_{3,4} + b_5 J \psi_{3,5} + b_6 J \psi_{3,6} = 0, \tag{4.2b}$$

$$b_2 c_2 K \psi_{2,2} + b_3 c_3 K \psi_{2,3} + b_4 c_4 K \psi_{2,4} + b_5 c_5 K \psi_{2,5} + b_6 c_6 K \psi_{2,6} = 0, \tag{4.2c}$$

$$b_3 J a_{32} J \psi_{2,2} + b_4 J (a_{42} J \psi_{2,2} + a_{43} J \psi_{2,3}) + b_5 J (a_{52} J \psi_{2,2} + a_{53} J \psi_{2,3} + a_{54} J \psi_{2,4})$$
$$+ b_6 J (a_{62} J \psi_{2,2} + a_{63} J \psi_{2,3} + a_{64} J \psi_{2,4} + a_{65} J \psi_{2,5}) = 0. \tag{4.2d}$$

We now solve these order conditions. We note from (3.3b) that

$$\psi_{2,i} = \sum_{j=2}^{i-1} a_{ij} c_j - c_i^2 \varphi_{2,i}, \quad \psi_{3,i} = \sum_{j=2}^{i-1} a_{ij} \frac{c_j^2}{2!} - c_i^3 \varphi_{3,i} \tag{4.3}$$

and thus $\psi_{2,2} = -c_2^2 \varphi_{2,2} \neq 0$, $\psi_{3,2} = -c_2^3 \varphi_{3,2} \neq 0$ (since $c_2 \neq 0$). Using (4.3), one can infer that either $\psi_{2,3}$ or $\psi_{3,3}$ must be nonzero as well (if both are zero then $a_{32} = \frac{c_3^2}{c_2} \varphi_{2,3} = \frac{2c_3^3}{c_2^2} \varphi_{3,3}$, which is impossible since $c_3 > 0$ and $\{\varphi_2, \varphi_3\}$ are linearly independent). This strongly suggests that $b_2 = b_3 = 0$ in order to later fulfill (4.2) in the strong sense with arbitrary square matrices J and K. Next, we further observe that if $b_4 \neq 0$ one may need both $\psi_{2,4} = \psi_{3,4} = 0$ (which solves $a_{42} \neq 0$, $a_{43} \neq 0$). However, this makes the second term in (4.2d) to be nonzero which is then very difficult to satisfy (4.2d) in the strong form. Putting together, it requires that $b_2 = b_3 = b_4 = 0$.

Using this sufficient condition we can easily solve (4.1) to get

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

for any choice of distinct nodes $c_5, c_6 > 0$, satisfying the condition

$$c_5 = \frac{4c_6 - 3}{6c_6 - 4}. \tag{4.4}$$

Since $b_5, b_6 \neq 0$, we must enforce $\psi_{2,5} = \psi_{3,5} = 0$ and $\psi_{2,6} = \psi_{3,6} = 0$ to satisfy conditions (4.2a)–(4.2c). Using (4.3), this leads to the following 2 systems of two linear equations

$$a_{52}c_2 + a_{53}c_3 + a_{54}c_4 = c_5^2\varphi_{2,5}, \tag{4.5a}$$

$$a_{52}c_2^2 + a_{53}c_3^2 + a_{54}c_4^2 = 2c_5^3\varphi_{3,5}, \tag{4.5b}$$

and

$$a_{62}c_2 + a_{63}c_3 + a_{64}c_4 + a_{65}c_5 = c_6^2\varphi_{2,6}, \tag{4.6a}$$

$$a_{62}c_2^2 + a_{63}c_3^2 + a_{64}c_4^2 + a_{65}c_5^2 = 2c_6^3\varphi_{3,6}. \tag{4.6b}$$

To satisfy conditions (4.2d), we further enforce $a_{52} = a_{62} = 0$ (since $\psi_{2,2} \neq 0$), which immediately solves (4.5) for coefficients (with $c_3 \neq c_4$)

$$a_{53} = \frac{-c_4 c_5^2\varphi_{2,5} + 2c_5^3\varphi_{3,5}}{c_3(c_3 - c_4)} \neq 0, \quad a_{54} = \frac{-c_3 c_5^2\varphi_{2,5} + 2c_5^3\varphi_{3,5}}{c_4(c_4 - c_3)} \neq 0, \tag{4.7}$$

and thus we also need $\psi_{2,3} = \psi_{2,4} = 0$ (since $\psi_{2,5} = 0$), which gives

$$a_{32} = \frac{c_3^2}{c_2}\varphi_{2,3}, \tag{4.8a}$$

$$a_{42}c_2 + a_{43}c_3 = c_4^2\varphi_{2,4}. \tag{4.8b}$$

After fulfilling all the required order conditions in (4.1)–(4.2), we see from (4.6) and (4.8b) that either a_{42} or a_{43} and one of the coefficients among a_{63}, a_{64}, a_{65} can be taken as free parameters. We now use them to construct parallel stages. Guided by (4.7) and (4.8a), we choose $a_{43} = 0$ to make U_{n4} is independent of U_{n3} so that both these stages only depend on U_{n2}, and choose $a_{65} = 0$ to make U_{n6} is independent of U_{n5} so that both these stages only depend on the two preceding stages U_{n3}, U_{n4} (since $a_{52} = a_{62} = 0$). From this we determine the remaining coefficients

$$a_{42} = \frac{c_4^2}{c_2}\varphi_{2,4}, \quad a_{63} = \frac{-c_4 c_6^2\varphi_{2,6} + 2c_6^3\varphi_{3,6}}{c_3(c_3 - c_4)}, \quad a_{64} = \frac{-c_3 c_6^2\varphi_{2,6} + 2c_6^3\varphi_{3,6}}{c_4(c_4 - c_3)}.$$

$$\tag{4.9}$$

Putting altogether and rearranging terms in U_{ni}, u_{n+1} as linear combinations of φ functions, we obtain the following family of 4th-order 6-stage methods (with the pairs of parallel stages $\{U_{n3}, U_{n4}\}$ and $\{U_{n5}, U_{n6}\}$), which will be called `expRK4s6`:

$$U_{n2} = u_n + \varphi_1(c_2 hA)c_2 hF(t_n, u_n), \tag{4.10a}$$

$$U_{n,k} = u_n + \varphi_1(c_k hA)c_k hF(t_n, u_n) + \varphi_2(c_k hA)\frac{c_k^2}{c_2} hD_{n2}, \quad k = 3, 4 \tag{4.10b}$$

$$U_{n,j} = u_n + \varphi_1(c_j hA)c_j hF(t_n, u_n) + \varphi_2(c_j hA)\frac{c_j^2}{c_3 - c_4} h\left(\frac{-c_4}{c_3}D_{n3} + \frac{c_3}{c_4}D_{n4}\right)$$
$$\quad + \varphi_3(c_j hA)\frac{2c_j^3}{c_3 - c_4} h\left(\frac{1}{c_3}D_{n3} - \frac{1}{c_4}D_{n4}\right), \quad j = 5, 6 \tag{4.10c}$$

$$u_{n+1} = u_n + \varphi_1(hA)hF(t_n, u_n) + \varphi_2(hA)\frac{1}{c_5 - c_6} h\left(\frac{-c_6}{c_5}D_{n5} + \frac{c_5}{c_6}D_{n6}\right)$$
$$\quad + \varphi_3(hA)\frac{2}{c_5 - c_6} h\left(\frac{1}{c_5}D_{n5} - \frac{1}{c_6}D_{n6}\right). \tag{4.10d}$$

For the numerical experiments given in Sect. 6, we choose $c_2 = c_3 = \frac{1}{2}$, $c_4 = \frac{1}{3}$, $c_6 = \frac{1}{3}$ which gives $c_5 = \frac{5}{6}$ due to (4.4).

Remark 4.1 (*A comparison with* `expRK4s5`). As seen, `expRK4s6` is resulted from weakening only condition 4 of Table 1 instead of weakening four conditions 4–7 as in the derivation of `expRK4s5`. While the 5-stage method `expRK4s5` requires 6 sequential evaluations in each step (as mentioned in Section 2), the new fourth-order 6-stage method `expRK4s6` requires only 4 sequential evaluations, making it to behave like a 4-stage method. This is due to the fact `expRK4s6` has the pairs of parallel stages $\{U_{n3}, U_{n4}\}$ and $\{U_{n5}, U_{n6}\}$ and all U_{ni} within these pairs have the same format, i.e., same (one) linear combination of $\varphi_k(c_i hA)v_k$, allowing them to be computed in parallel or simultaneously (see Sect. 5).

4.2 A family of fifth-order methods with parallel stages

Constructing fifth-order exponential Runge-Kutta methods needs much more effort as one has to solve 16 order conditions in Table 1. As mentioned in Section 2, the only existing method of order 5 in the literature is `expRK5s8` (see [15]) which requires $s = 8$ stages. Like `expRK4s5`, this method does not have any parallel stages and must be implemented in a sequential way. It also does not satisfy the assumption on the order conditions stated in Theorem 3.1. Indeed, it was constructed by fulfilling conditions 1–7 in the strong form and weakening conditions 8–16 (9 out of 16 order conditions) with $b_i(0)$ in place of $b_i(Z)$. This resulted in the last two internal stages U_{n7} and U_{n8} that involve several different linear combinations of $\varphi_k(c_i hA)v_k$ (with different scalings c_6, c_7, c_8 of hA), for which additional computational efforts are required to compute those stages (as shown in Section 2).

Therefore, to derive a method based on Theorem 3.1 which later allows us to derive parallel stages schemes with U_{ni} involving only one linear combination of $\varphi_k(c_i hA)v_k$, we have to increase $s \geq 9$. To make it easier for readers to follow, we consider $s = 10$ first and later employ the similar procedure to show that it is not possible to fulfill

condition 11 of Table 1 in the strong form (and thus not satisfying Theorem 3.1) with $s = 9$.

(a) The case $s = 10$: Similarly to the derivation presented in Sect. 4.1, using (4.3), it strongly suggests $b_2 = b_3 = b_4 = b_5 = b_6 = b_7 = 0$ in order to solve conditions 3, 5, 9, 7, 16, 13, and 15 in their strong form. Using this, these conditions now read as

$$b_8 J \psi_{2,8} + b_9 J \psi_{2,9} + b_{10} J \psi_{2,10} = 0, \tag{4.11a}$$

$$b_8 J \psi_{3,8} + b_9 J \psi_{3,9} + b_{10} J \psi_{3,10} = 0, \tag{4.11b}$$

$$b_8 J \psi_{4,8} + b_9 J \psi_{4,9} + b_{10} J \psi_{4,10} = 0, \tag{4.11c}$$

$$b_8 c_8 K \psi_{2,8} + b_9 c_9 K \psi_{2,9} + b_{10} c_{10} K \psi_{2,10} = 0, \tag{4.11d}$$

$$b_8 c_8^2 L \psi_{2,8} + b_9 c_9^2 L \psi_{2,9} + b_{10} c_{10}^2 L \psi_{2,10} = 0, \tag{4.11e}$$

$$b_8 c_8 K \psi_{3,8} + b_9 c_9 K \psi_{3,9} + b_{10} c_{10} K \psi_{3,10} = 0, \tag{4.11f}$$

$$b_8 B(\psi_{2,8}, \psi_{2,8}) + b_9 B(\psi_{2,9}, \psi_{2,9}) + b_{10} B(\psi_{2,10}, \psi_{2,10}) = 0, \tag{4.11g}$$

respectively. And conditions 1, 2, 4, and 8 (weakened form) become

$$b_8 c_8 + b_9 c_9 + b_{10} c_{10} = \varphi_2, \tag{4.12a}$$

$$b_8 c_8^2 + b_9 c_9^2 + b_{10} c_{10}^2 = 2\varphi_3, \tag{4.12b}$$

$$b_8 c_8^3 + b_9 c_9^3 + b_{10} c_{10}^3 = 6\varphi_4, \tag{4.12c}$$

$$b_8(0) c_8^4 + b_9(0) c_9^4 + b_{10}(0) c_{10}^4 = 24\varphi_5(0) = 1/5. \tag{4.12d}$$

Solving (4.12) gives

$$b_8 = \frac{c_9 c_{10} \varphi_2 - 2(c_9 + c_{10})\varphi_3 + 6\varphi_4}{c_8 (c_8 - c_9)(c_8 - c_{10})}, \tag{4.13a}$$

$$b_9 = \frac{c_8 c_{10} \varphi_2 - 2(c_8 + c_{10})\varphi_3 + 6\varphi_4}{c_9 (c_9 - c_8)(c_9 - c_{10})}, \tag{4.13b}$$

$$b_{10} = \frac{c_8 c_9 \varphi_2 - 2(c_8 + c_9)\varphi_3 + 6\varphi_4}{c_{10} (c_{10} - c_8)(c_{10} - c_9)} \tag{4.13c}$$

where c_8, c_9, and c_{10} are distinct and positive nodes satisfying the algebraic equation

$$\frac{c_8 + c_9 + c_{10}}{4} - \frac{c_8 c_9 + c_8 c_{10} + c_9 c_{10}}{3} + \frac{c_8 c_9 c_{10}}{2} = \frac{1}{5}. \tag{4.14}$$

Clearly, $b_8, b_9, b_{10} \neq 0$ so one has to enforce

$$\psi_{2,j} = \psi_{3,j} = \psi_{4,j} = 0 \ (j = 8, 9, 10) \tag{4.15}$$

to satisfy (4.11) in the strong sense with arbitrary square matrices J, K, L and B. Next, we consider conditions 6 and 10 taken into account that $b_i = 0 \ (i = 2, \cdots, 7)$

and (4.15), which can be now simplified as

$$\sum_{j=2}^{7}(b_8 J a_{8j} + b_9 J a_{9j} + b_{10} J a_{10j}) J \psi_{m,j} = 0 \quad (m = 2, 3), \tag{4.16}$$

respectively. In order to satisfy the strong form of (4.16) one needs

$$a_{8j} = a_{9j} = a_{10j} = 0 \ (j = 2, 3, 4) \tag{4.17}$$

(this is again due to (4.3)) and

$$\psi_{2,j} = \psi_{3,j} = 0 \ (j = 5, 6, 7). \tag{4.18}$$

With (4.17), we note that U_{n8}, U_{n9}, U_{n10} are independent of the internal stages U_{n2}, U_{n3}, U_{n4}. Taking into all the requirements above, one can easily see that conditions 12 and 14 are now automatically fulfilled. Therefore, the only remaining condition to satisfy is condition 11.

Before working with condition 11, we first solve (4.15) using (4.17). For this, we observe that several coefficients a_{ij} can be considered as free parameters. To have U_{n8}, U_{n9}, U_{n10} are independent of each other, we choose

$$a_{98} = a_{10,8} = a_{10,9} = 0. \tag{4.19}$$

The resulting systems of linear equations from (4.15) is then solved with the unique solution

$$a_{ij} = \frac{c_i^2 c_k c_l \varphi_{2,i} - 2c_i^3 (c_k + c_l)\varphi_{3,i} + 6c_i^4 \varphi_{4,i}}{c_j (c_j - c_k)(c_j - c_l)},$$
$$i = 8, 9, 10; \ j, k, l \in \{5, 6, 7\}, \ j \neq k \neq l \tag{4.20}$$

(i.e., $c_5, c_6, c_7 > 0$ are distinct nodes).
We now use $b_i = 0 \ (i = 2, \cdots, 7)$, (4.15), (4.17), (4.18), and (4.19) to simplify condition 11 as

$$\sum_{i=8}^{10} b_i J \sum_{j=5}^{7} a_{ij} J \left(a_{j2} J \psi_{2,2} + a_{j3} J \psi_{2,3} + a_{j4} J \psi_{2,4}\right) = 0. \tag{4.21}$$

Since $b_8, b_9, b_{10} \neq 0$, coefficients a_{ij} in (4.20) ($i \in \{8, 9, 10\}, j \in \{5, 6, 7\}$) are also nonzero, and that $\psi_{2,2} \neq 0$, we must enforce

$$a_{j2} = 0 \ (j = 5, 6, 7), \text{ i.e., } a_{52} = a_{62} = a_{72} = 0 \tag{4.22}$$

and require that

$$\psi_{2,3} = \psi_{2,4} = 0 \tag{4.23}$$

in order to satisfy (4.21) in the strong sense. Note, because of (4.22), one could not require $a_{53} = 0$ or $a_{54} = 0$ ($j = 5$) in (4.21) or both as this does not agree with the requirement $\psi_{2,5} = \psi_{3,5} = 0$ in (4.18) (in other words, the linear system of equations displayed in (4.5) represented for this requirement has no solution). This justifies the requirement (4.23).

Finally, we solve (4.23) and (4.18) for the remaining coefficients a_{ij}. When solving (4.23) (see (4.8)), we choose $a_{43} = 0$ to have U_{n4} is independent of U_{n3}. This gives

$$a_{32} = \frac{c_3^2}{c_2}\varphi_{2,3}, \quad a_{42} = \frac{c_4^2}{c_2}\varphi_{2,4}. \tag{4.24}$$

When solving (4.18) (using (4.22)), we choose $a_{65} = a_{75} = a_{76} = 0$ to have U_{n5}, U_{n6}, U_{n7} are independent of each other. This results in the following 6 coefficients:

$$a_{ij} = \frac{-c_i^2 c_k \varphi_{2,i} + 2c_i^3 \varphi_{3,i}}{c_j(c_j - c_k)}, \quad i = 5, 6, 7; \ j, k \in \{3, 4\}, \ j \neq k \tag{4.25}$$

(i.e., $c_3, c_4 > 0$ are distinct nodes).

Inserting all the obtained coefficients a_{ij} and b_i into U_{ni}, u_{n+1} and rewriting these stages as linear combinations of φ functions, we obtain the following family of 5th-order 10-stage methods (with the groups of parallel stages $\{U_{n3}, U_{n4}\}$, $\{U_{n5}, U_{n6}, U_{n7}\}$, and $\{U_{n8}, U_{n9}, U_{n10}\}$) which will be called expRK5s10:

$$U_{n2} = u_n + \varphi_1(c_2 hA)c_2 hF(t_n, u_n),$$

$$U_{n\ell} = u_n + \varphi_1(c_\ell hA)c_\ell hF(t_n, u_n) + \varphi_2(c_\ell hA)\frac{c_\ell^2}{c_2} hD_{n2}, \quad \ell = 3, 4$$

$$U_{nm} = u_n + \varphi_1(c_m hA)c_m hF(t_n, u_n) + \varphi_2(c_m hA)c_m^2 h\left(\frac{c_4}{c_3(c_4 - c_3)}D_{n3} + \frac{c_3}{c_4(c_3 - c_4)}D_{n4}\right)$$

$$\quad + \varphi_3(c_m hA)c_m^3 h\left(\frac{2}{c_3(c_3 - c_4)}D_{n3} - \frac{2}{c_4(c_3 - c_4)}D_{n4}\right), \quad m = 5, 6, 7$$

$$U_{nq} = u_n + \varphi_1(c_q hA)c_q hF(t_n, u_n) + \varphi_2(c_q hA)c_q^2 h\left(\alpha_5 D_{n5} + \alpha_6 D_{n6} + \alpha_7 D_{n7}\right)$$

$$\quad + \varphi_3(c_q hA)c_q^3 h\left(\beta_5 D_{n5} - \beta_6 D_{n6} - \beta_7 D_{n7}\right)$$

$$\quad + \varphi_4(c_q hA)c_q^4 h\left(\gamma_5 D_{n5} + \gamma_6 D_{n6} + \gamma_7 D_{n7}\right), \quad q = 8, 9, 10$$

$$u_{n+1} = u_n + \varphi_1(hA)hF(t_n, u_n) + \varphi_2(hA)h\left(\alpha_8 D_{n8} + \alpha_9 D_{n9} + \alpha_{10} D_{n10}\right)$$

$$\quad - \varphi_3(hA)h\left(\beta_8 D_{n8} + \beta_9 D_{n9} + \beta_{10} D_{n10}\right) + \varphi_4(hA)h\left(\gamma_8 D_{n8} + \gamma_9 D_{n9} + \gamma_{10} D_{n10}\right),$$

where

$$\alpha_i = \frac{c_k c_l}{c_i(c_i - c_k)(c_i - c_l)}, \quad \beta_i = \frac{2(c_k + c_l)}{c_i(c_i - c_k)(c_i - c_l)}, \quad \gamma_i = \frac{6}{c_i(c_i - c_k)(c_i - c_l)} \tag{4.26}$$

with $i \in \{5, 6, 7\}$ for $k, l \in \{5, 6, 7\}$, and $i \in \{8, 9, 10\}$ for $k, l \in \{8, 9, 10\}$ (note that i, k, l are distinct indices and that c_i, c_k, c_l are distinct (positive) nodes).

For our numerical experiments, we choose $c_2 = c_3 = c_5 = \frac{1}{2}$, $c_4 = c_6 = \frac{1}{3}$, $c_7 = \frac{1}{4}$, $c_8 = \frac{3}{10}$, $c_9 = \frac{3}{4}$, and $c_{10} = 1$ (satisfying (4.14)).

Remark 4.2 (*A comparison with* `expRK5s8`). Although the new fifth-order method `expRK5s10` has 10 stages (compared to 8 stages of `expRK5s8` displayed in Section 2), its special structure offers much more efficient for implementation. In particular, all U_{ni} in this scheme involve only one linear combination of $\varphi_k(c_i h A) v_k$ which can be computed by one evaluation for each; and more importantly, due to the same format of multiple stages within each of the three groups $\{U_{n3}, U_{n4}\}$, $\{U_{n5}, U_{n6}, U_{n7}\}$, and $\{U_{n8}, U_{n9}, U_{n10}\}$ (same linear combination with different inputs c_i), they can be computed simultaneously or implemented in parallel (see Sect. 5). This makes `expRK5s10` to behave like a 5-stage method only, thereby requiring only 5 sequential evaluations in each step. Moreover, while `expRK5s8` requires weakening 9 out of 16 order conditions of Table 1, `expRK5s10` requires only one condition (number 8) held in the weakened form. Note that by following the similar way of deriving `expRK5s10`, we can derive a scheme that satisfies all the stiff order conditions in Table 1 in the strong sense with $s = 11$. Such a scheme, however, still behaves like a 5-stage method. Therefore, we do not discuss further on this case.

(b) The case $s = 9$ (which does not work): Clearly, in this case we have less degree of freedoms than the case $s = 10$ when solving the order conditions in Table 1. Nonetheless, one can still proceed in a similar way as done for $s = 10$. Again, it strongly suggests $b_2 = b_3 = b_4 = b_5 = b_6 = 0$ (which solves for $b_7, b_8, b_9 \neq 0$ from conditions 1, 2, 4) and

$$\psi_{2,j} = \psi_{3,j} = \psi_{4,j} = 0 \; (j = 7, 8, 9) \tag{4.27}$$

in order to satisfy conditions 1, 2, 3, 4, 5, 7, 9, 13, 15, 16 in the strong form. With this, conditions 6 and 10 now become

$$\sum_{j=2}^{6} (b_7 J a_{7j} + b_8 J a_{8j} + b_9 J a_{9j}) J \psi_{m,j} = 0 \; (m = 2, 3). \tag{4.28}$$

Again, due to the fact that $\psi_{2,2}, \psi_{3,2} \neq 0$ and either $\psi_{2,3}$ or $\psi_{3,3}$ must be nonzero, one needs to enforce $a_{7j} = a_{8j} = a_{9j} = 0 \; (j = 2, 3)$ in (4.28). Using this to solve (4.27) for $j = 7$ ($\psi_{2,7} = \psi_{3,7} = \psi_{4,7} = 0$) gives a unique solution (with $c_4, c_5, c_6 > 0$ and are distinct) for $a_{74}, a_{75}, a_{76} \neq 0$, which then determines U_{n7}. Next, one can solve (4.27) for $j = 8, 9$ to obtain U_{n8}, U_{n9} that are independent of U_{n7}, as well as are independent of each other, by requiring the three free parameters $a_{87} = a_{97} = a_{98} = 0$. As a result, one gets $a_{7j}, a_{8j}, a_{9j} \neq 0 \; (j = 5, 6)$. This immediately suggests $\psi_{2,j} = \psi_{3,j} = 0 \; (j = 4, 5, 6)$ to completely fulfill (4.28) with arbitrary square matrix J. With all of these in place, conditions 12 and 14 are automatically fulfilled, and condition 11 is now reduced to

$$\sum_{i=7}^{9} b_i J \sum_{j=4}^{6} a_{ij} J (a_{j2} J \psi_{2,2} + a_{j3} J \psi_{2,3}) = 0. \tag{4.29}$$

Clearly, since $b_7, b_8, b_9 \neq 0$, $a_{7j}, a_{8j}, a_{9j} \neq 0$ ($j = 4, 5, 6$), and $\psi_{2,2} \neq 0$, (4.29) can be satisfied in the strong sense only if we have one of the following conditions: $a_{j2} = a_{j3} = 0$ or $a_{j2} = \psi_{2,3} = 0$, ($j = 4, 5, 6$). Unfortunately, either of these requirements is in contradiction with $\psi_{2,j} = \psi_{3,j} = 0$ ($j = 4, 5, 6$) which is needed for conditions 6 and 10 mentioned above. For example, solving $\psi_{2,4} = \psi_{3,4} = 0$ results in $a_{42}, a_{43} \neq 0$.

5 Details implementation of fourth- and fifth-order schemes

In this section, we present details implementation of the old and new fourth- and fifth-order expRK schemes (`expRK4s5`, `expRK5s8`, `expRK4s6`, `expRK5s10`) mentioned above.

As mentioned in Sect. 2.1, we will use the MATLAB routine `phipm_simul_iom` (described in details in [19]) to implement expRK methods. In particular, given the following inputs: an array of scaling factors $\mathtt{t} = [\rho_1, \ldots, \rho_r]$ with $0 < \rho_1 < \rho_2 < \cdots < \rho_r \leq 1$ ($\mathtt{t}$ could be a positive scalar), an n-by-n matrix M, and a set of column vectors $\mathtt{V} = [V_0, \ldots, v_q]$ (each v_i is an n-by-1 vector), a tolerance $\mathtt{tol}$, an initial value m for the dimension of the Krylov subspace, and an incomplete orthogonalization length of $\mathtt{iom}$, a call to this function

$$\mathtt{phipm\_simul\_iom(t, M, V, tol, m, iom)} \tag{5.1}$$

simultaneously computes the following r linear combinations

$$L_{\rho_i, \mathrm{V}} = \varphi_0(\rho_i M)v_0 + \varphi_1(\rho_i M)\rho_i v_1 + \varphi_2(\rho_i M)\rho_i^2 v_2 + \cdots + \varphi_q(\rho_i M)\rho_i^q v_q,$$
$$1 \leq i \leq r. \tag{5.2}$$

Note that, by setting $V_j = \rho_i^j v_j$ ($j = 0, \cdots, q$), (5.2) becomes (2.2). In other words, all the linear combinations in (2.2) (if V_j are given instead of v_j) can be then computed at the same time with one call (5.1) by using scaled vectors $v_j = V_j/\rho_i^j$ for the input V.

In the following, we set

$$M = hA, \quad \rho_i = c_i, \quad v = hF(t_n, u_n), \quad d_i = hD_{ni}. \tag{5.3}$$

to simplify notations in presenting details of implementation of the fourth- and fifth-order methods mentioned above. When calling (5.1), we use $\mathtt{tol} = 10^{-12}$, $\mathtt{m} = 1$ (default value), and $\mathtt{imo} = 2$ (as in [19]).

Implementation of `expRK4s5` ($c_2 = c_3 = c_5 = \frac{1}{2}, c_4 = 1$): As discussed in Remark 2.1, `expRK4s5` requires a sequential implementation of the following 6 different linear combinations of the form (5.2), corresponding to 6 calls to `phipm_simul_iom`:

  (i) Evaluate $L_{c_2, \mathrm{V}}$ with $\mathtt{t} = c_2$, $\mathtt{V} = [0, v]$ to get $U_{n2} = u_n + L_{c_2, \mathrm{V}}$.
  (ii) Evaluate $L_{c_3, \mathrm{V}}$ with $\mathtt{t} = c_3$, $\mathtt{V} = [0, v, d_2/c_3^2]$ to get $U_{n3} = u_n + L_{c_3, \mathrm{V}}$.

(iii) Evaluate $L_{c_4,V}$ with $t = c_4$, $V = [0, v, d_2 + d_3]$ to get $U_{n4} = u_n + L_{c_4,V}$.

(iv) Evaluate L_{c_5,V_1} with $t = c_5$, $V_1 = [0, v, 2d_2 + 2d_3 - d_4, (-d_2 - d_3 + d_4)/c_5^2]$ and

(v) Evaluate L_{c_4,V_2} with $t = c_4$, $V_2 = [0, 0, (d_2 + d_3 - d_4)/4, (-d_2 - d_3 + d_4)]$ to get $U_{n5} = u_n + L_{c_5,V_1} + L_{c_4,V_2}$.

(vi) Evaluate $L_{1,V}$ with $t = 1$, $V = [0, v, -d_4 + 5d_5, 4d_4 - 8d_5]$ to get $u_{n+1} = u_n + L_{1,V}$.

Since $d_i = h D_{ni}$ which depends on U_{ni}, these are the 6 (sequential) evaluations.

Implementation of $\texttt{expRK4s6}$ ($c_2 = c_3 = \frac{1}{2}, c_4 = c_6 = \frac{1}{3}, c_5 = \frac{5}{6}$): As discussed in Remark 4.1, $\texttt{expRK4s6}$ can be implemented like a 4-stage method by evaluating the following 4 sequential evaluations, corresponding to 4 calls to $\texttt{phipm\_simul\_iom}$:

(i) Evaluate $L_{c_2,V}$ with $t = c_2$, $V = [0, v]$ to get $U_{n2} = u_n + L_{c_2,V}$.

(ii) Evaluate $L_{c_4,V}$ and $L_{c_3,V}$ simultaneously using $t = [c_4, c_3]$, $V = [0, v, d_2/c_2]$ to get both $U_{n3} = u_n + L_{c_3,V}$ and $U_{n4} = u_n + L_{c_4,V}$.

(iii) Evaluate $L_{c_5,V}$ and $L_{c_6,V}$ simultaneously with $t = [c_6, c_5]$, $V = [0, v, \frac{-c_4}{(c_3-c_4)c_3}d_3 + \frac{c_3}{(c_3-c_4)c_4}d_4, \frac{1}{(c_3-c_4)c_3}d_3 - \frac{1}{(c_3-c_4)c_4}d_4]$ to get both $U_{n5} = u_n + L_{c_5,V}$ and $U_{n6} = u_n + L_{c_6,V}$.

(iv) Evaluate $L_{1,V}$ with $t = 1$, $V = [0, v, \frac{1}{c_5-c_6}(\frac{-c_6}{c_5}d_5 + \frac{c_5}{c_6}d_6), \frac{2}{c_5-c_6}(\frac{1}{c_5}d_5 - \frac{1}{c_6}d_6)]$ to get $u_{n+1} = u_n + L_{1,V}$.

Implementation of $\texttt{expRK5s8}$ ($c_2 = c_3 = c_5 = \frac{1}{2}, c_4 = \frac{1}{4}, c_6 = \frac{1}{5}, c_7 = \frac{2}{3}, c_8 = 1$): As discussed in Remark 2.1, $\texttt{expRK5s8}$ requires a sequential implementation of 11 different linear combinations of the form (5.2), corresponding to the following 11 calls to $\texttt{phipm\_simul\_iom}$:

(i) Evaluate $L_{c_2,V}$ with $t = c_2$, $V = [0, v]$ to get $U_{n2} = u_n + L_{c_2,V}$.

(ii) Evaluate $L_{c_3,V}$ with $t = c_3$, $V = [0, v, d_2/c_3^2]$ to get $U_{n3} = u_n + L_{c_3,V}$.

(iii) Evaluate $L_{c_4,V}$ with $t = c_4$, $V = [0, v, d_3/c_4^2]$ to get $U_{n4} = u_n + L_{c_4,V}$.

(iv) Evaluate $L_{c_5,V}$ with $t = c_5$, $V = [0, v, (-d_3 + 4d_4)/c_5^2, (2d_3 - 4d_4)/c_5^3]$ to get $U_{n5} = u_n + L_{c_5,V}$.

(v) Evaluate $L_{c_6,V}$ with $t = c_6$, $V = [0, v, (8d_4 - 2d_5)/c_6^2, (-32d_4 + 16d_5)/c_6^3]$ to get $U_{n6} = u_n + L_{c_6,V}$.

(vi) Evaluate L_{c_7,V_1} with $t = c_7$, $V_1 = [0, v, (\frac{-16}{27}d_5 + \frac{100}{27}d_6)/c_7^2, (\frac{320}{81}d_5 - \frac{800}{81}d_{n6})/c_7^3]$ and

(vii) Evaluate L_{c_6,V_2} with $t = c_6$, $V_2 = [0, 0, (\frac{-20}{81}d_4 + \frac{5}{243}d_5 + \frac{125}{486}d_6)/c_6^2, (\frac{16}{81}d_4 - \frac{4}{243}d_5 - \frac{50}{243}d_6)/c_6^3]$ to get $U_{n7} = u_n + L_{c_7,V_1} + L_{c_6,V_2}$.

(viii) Evaluate L_{c_8,V_1} with $t = c_8$, $V_1 = [0, v, (\frac{-16}{3}d_5 + \frac{250}{21}d_6 + \frac{27}{14}d_7)/c_8^2, (\frac{208}{3}d_5 - \frac{250}{3}d_6 - 27d_7)/c_8^3, (-240d_5 + \frac{1500}{7}d_6 + \frac{810}{7}d_7)/c_8^4]$ and

(ix) Evaluate L_{c_6,V_2} with $t = c_6$, $V_2 = [0, 0, (\frac{-4}{7}d_5 + \frac{25}{49}d_6 + \frac{27}{98}d_7)/c_6^2, (\frac{8}{5}d_5 - \frac{10}{7}d_6 - \frac{27}{35}d_7)/c_6^3, (\frac{-48}{35}d_5 + \frac{60}{49}d_6 + \frac{162}{245}d_7)/c_6^4]$ and

(x) Evaluate L_{c_7,V_3} with $t = c_7$, $V_3 = [0, 0, (\frac{-288}{35}d_5 + \frac{360}{49}d_6 + \frac{972}{245}d_7)/c_7^2, (\frac{384}{5}d_5 - \frac{480}{7}d_6 - \frac{1296}{35}d_7)/c_7^3, (\frac{-1536}{7}d_5 + \frac{9600}{49}d_6 + \frac{5184}{49}d_7)/c_7^4]$ to get $U_{n8} = u_n + L_{c_8,V_1} + L_{c_6,V_2} + L_{c_7,V_3}$.

(xi) Evaluate $L_{1,V}$ with $t = 1$, $V = [0, v, \frac{125}{14}d_6 - \frac{27}{14}d_7 + \frac{1}{2}d_8, \frac{-625}{14}d_6 + \frac{162}{7}d_7 - \frac{13}{2}d_8, \frac{1125}{14}d_6 - \frac{405}{7}d_7 + \frac{45}{2}d_8]$ to get $u_{n+1} = u_n + L_{1,V}$.

Implementation of `expRK5s10` ($c_2 = c_3 = c_5 = \frac{1}{2}$, $c_4 = c_6 = \frac{1}{3}$, $c_7 = \frac{1}{4}$, $c_8 = \frac{3}{10}$, $c_9 = \frac{3}{4}$, and $c_{10} = 1$): As discussed in Remark 4.2, `expRK5s10` can be implemented like a 5-stage method by evaluating the following 5 sequential evaluations, corresponding to 5 calls to `phipm_simul_iom`:

(i) Evaluate $L_{c_2,V}$ with $\mathtt{t} = c_2$, $\mathtt{V} = [0, v]$ to get $U_{n2} = u_n + L_{c_2,V}$.

(ii) Evaluate $L_{c_4,V}$ and $L_{c_3,V}$ simultaneously using $\mathtt{t} = [c_4, c_3]$, $\mathtt{V} = [0, v, d_2/c_2]$ to get both $U_{n3} = u_n + L_{c_3,V}$ and $U_{n4} = u_n + L_{c_4,V}$.

(iii) Evaluate $L_{c_5,V}$, $L_{c_6,V}$, and $L_{c_7,V}$ simultaneously using $\mathtt{t} = [c_7, c_6, c_5]$, $\mathtt{V} = [0, v, \frac{c_4}{c_3(c_4-c_3)}d_3 + \frac{c_3}{c_4(c_3-c_4)}d_4, \frac{2}{c_3(c_3-c_4)}d_3 - \frac{2}{c_4(c_3-c_4)}d_4]$ to get $U_{n5} = u_n + L_{c_5,V}$, $U_{n6} = u_n + L_{c_6,V}$, $U_{n7} = u_n + L_{c_7,V}$.

(iv) Evaluate $L_{c_8,V}$, $L_{c_9,V}$, and $L_{c_{10},V}$ simultaneously using $\mathtt{t} = [c_9, c_{10}, c_8]$, $\mathtt{V} = [0, v, \alpha_5 d_5 + \alpha_6 d_6 + \alpha_7 d_7, \beta_5 d_5 - \beta_6 d_6 - \beta_7 d_7, \gamma_5 d_5 + \gamma_6 d_6 + \gamma_7 d_7]$ to get $U_{n8} = u_n + L_{c_8,V}$, $U_{n9} = u_n + L_{c_9,V}$, $U_{n10} = u_n + L_{c_{10},V}$.

(v) Evaluate $L_{1,V}$ with $\mathtt{t} = 1$, $\mathtt{V} = [0, v, \alpha_8 d_8 + \alpha_9 d_9 + \alpha_{10} d_{10}, \beta_8 d_8 + \beta_9 d_9 + \beta_{10} d_{10}, \gamma_8 d_8 + \gamma_9 d_9 + \gamma_{10} d_{10}]$ to get $u_{n+1} = u_n + L_{1,V}$ (coefficients α_i, β_i, γ_i are given in (4.26)).

6 Numerical experiments

In this section, we demonstrate the efficiency of our newly derived fourth- and fifth-order expRK time integration methods (`expRK4s6`, `expRK5s10`). Specifically, we will compare their performance against the existing methods of the same orders (`expRK4s5`, `expRK5s8`) on several examples of stiff PDEs. All the numerical simulations are performed in MATLAB on a single workstation, using an iMac 3.6 GHz Intel Core i7, 32 GB 2400 MHz DDR4.

Example 6.1 (A one-dimensional semilinear parabolic problem [8]): We first verify the order of convergence for the new derived fourth- and fifth-order expRK schemes (`expRK4s6`, `expRK5s10`) by considering the following PDE for $u(x, t)$, $x \in [0, 1]$, $t \in [0, 1]$, and subject to homogeneous Dirichlet boundary conditions,

$$\frac{\partial u(x, t)}{\partial t} - \frac{\partial^2 u(x, t)}{\partial x^2} = \frac{1}{1 + u^2(x, t)} + \Phi(x, t), \tag{6.1}$$

whose exact solution is known to be $u(x, t) = x(1 - x)e^t$ for a suitable choice of the source function $\Phi(x, t)$.

Spatial discretization: For this example, we use standard second order finite differences with 200 grid points. This leads to a very stiff system of the form (1.1) (with $\|A\|_\infty \approx 1.6 \times 10^5$).

The resulting system is then integrated on the time interval [0, 1] using constant step sizes, corresponding to the number of time steps $N = 4, 8, 16, 32, 64$. The time integration errors at the final time $t = 1$ are measured in the maximum norm.

In Fig. 1, we plot orders for all the employed integrators in the left diagram and the total CPU time versus the global errors in the right diagram. The left diagram clearly
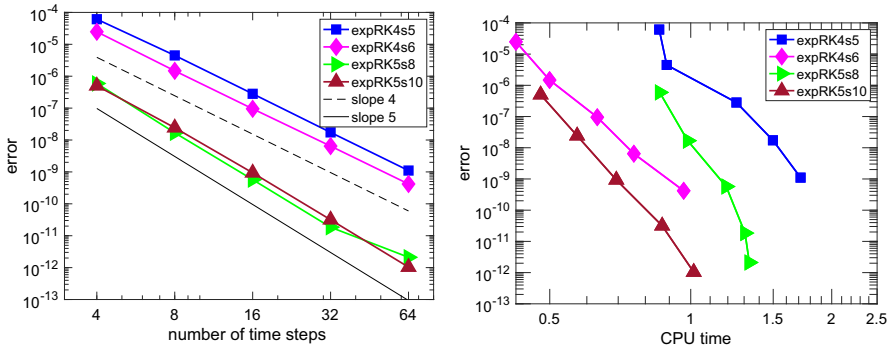
Fig. 1 Order plots (left) and total CPU times (right) of `expRK4s5`, `expRK4s6`, `expRK5s8`, and `expRK5s10` when applied to (6.1). The global errors at time $t = 1$ are plotted as functions of the number of time steps (left) and the total CPU time in second (right). For comparison, straight lines with slopes 4 and 5 are added

shows a perfect agreement with our convergence result in Theorem 3.1, meaning that the two new integrators `expRK4s6` and `expRK5s10` fully achieve orders 4 and 5, respectively. When compared to the old integrators of the same orders `expRK4s5` and `expRK5s8`, we note that, given the same number of time steps, `expRK4s6` is slightly more accurate but is much faster than `expRK4s5` (see the right diagram). In a similar manner, `expRK5s10` gives almost identical global errors but is also much faster than `expRK5s8`. Finally, we observe that, for this example, for a global error that is larger than 10^{-6}, the new fourth-order method `expRK4s6` is the fastest one, and for more stringent errors, `expRK5s10` is the fastest integrator.

Example 6.2 (A nonlinear Schrödinger equation [2,5]): We consider the following one-dimensional nonlinear Schrödinger (NLS) equation with periodic boundary conditions

$$
\begin{aligned}
\mathrm{i}\frac{\partial \Psi(x,t)}{\partial t} &= -\frac{\partial^2 \Psi(x,t)}{\partial x^2} + \left(V(x) + \lambda|\Psi(x,t)|^2\right)\Psi(x,t), \\
\Psi(-\pi,t) &= \Psi(\pi,t), \quad t \geq 0 \\
\Psi(0,t) &= \Psi_0(x), \quad x \in [-\pi,\pi]
\end{aligned}
\tag{6.2}
$$

where the potential function $V(x) = \dfrac{1}{1+\sin^2(x)}$, the initial condition $\Psi_0(x) = \mathrm{e}^{\sin(2x)}$, and the constant $\lambda = 1$ (see [2]).

Spatial discretization: For this example, we use a discrete Fourier transform $\mathscr{F}$ with $ND = 128$ modes, leading to a mildly stiff system of the form (1.1) with

$$
\begin{aligned}
A &= \mathrm{diag}(-\mathrm{i}k^2), \ k = -\frac{ND}{2}+1, \ldots, \frac{ND}{2} = -63, \ldots, 64 \\
g(t,u) &= -\mathrm{i}\mathscr{F}\left((V(x) + \lambda|\mathscr{F}^{-1}(u)|^2)\mathscr{F}^{-1}(u)\right).
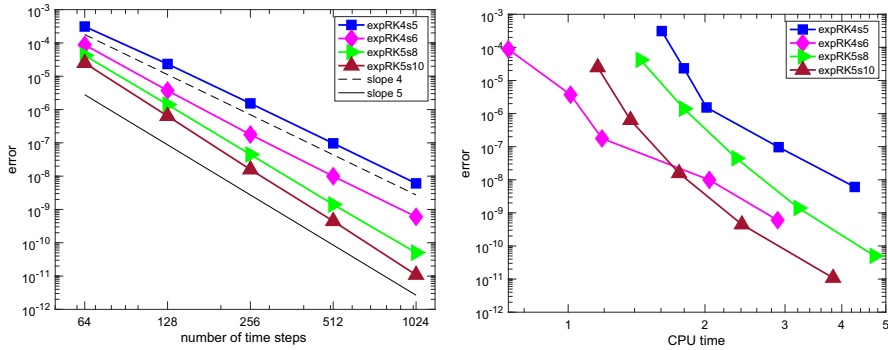\end{aligned}
\tag{6.3}
$$

Fig. 2 Order plots (left) and total CPU times (right) of `expRK4s5`, `expRK4s6` , `expRK5s8`, and `expRK5s10` when applied to Example 6.2. The errors at time $t = 3$ are plotted as functions of the number of time steps (left) and the total CPU time in second (right). For comparison, straight lines with slopes 4 and 5 are added

Next, we integrate this system on the time interval $[0, 3]$ with constant step sizes, corresponding to the number of time steps $N = 64, 128, 256, 512, 1024$. Since the exact solution $\Psi(x, t)$ of (6.2) is unknown, a reliable reference solution is computed by the stiff solver `ode15s` with $ATOL = RTOL = 10^{-14}$. Again, the time integration errors are measured in a discrete maximum norm at the final time $t = 3$.

As seen from the two double-logarithmic diagrams in Fig. 2, we plot the accuracy of the four employed integrators (`expRK4s5`, `expRK4s6` , `expRK5s8`, and `expRK5s10`) as functions of the number of time steps (left) and the total CPU time (right). The left digram clearly indicates that the two new integrators `expRK4s6` and `expRK5s10` achieve their corresponding expected orders 4 and 5. While `expRK5s10` is a little more accurate than `expRK5s8`, `expRK4s6` is much more accurate than `expRK4s5` for a given same number of time steps, meaning that it can take much larger time steps while achieving the same accuracy. Moreover, the right precision digram displays the efficiency plot indicating that both `expRK4s6` and `expRK5s10` are much faster than their counterparts `expRK4s5` and `expRK5s8`, respectively. More specifically, a similar story is observed: for lower accuracy requirements, say error $\sim 10^{-7}$, the new fourth-order method `expRK4s6` is the most efficient, whereas for error $\sim 10^{-8}$ or tighter the new fifth-order method `expRK5s10` is the most efficient.

Example 6.3 (A 2D Gray–Scott model [3,7]): Consider the following two-dimensional reaction-diffusion equation–the Gray–Scott equation model, for $u = u(x, y, t)$, $v = v(x, y, t)$ on the square $\Omega = [0, L]^2$, (here, we choose $L = 1.5$) subject to periodic boundary conditions

$$
\begin{aligned}
\frac{\partial u}{\partial t} &= d_u \Delta u - uv^2 + \alpha(1 - u), \\
\frac{\partial v}{\partial t} &= d_v \Delta v + uv^2 - (\alpha + \beta)v,
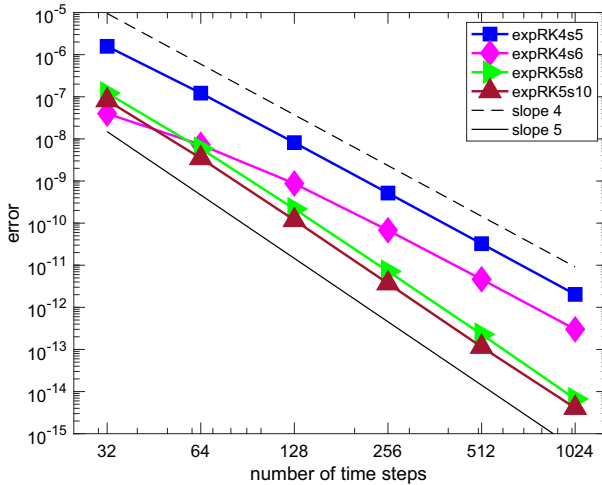\end{aligned}
\tag{6.4}
$$

Fig. 3 Order plots of `expRK4s5`, `expRK4s6`, `expRK5s8`, and `expRK5s10` when applied to Example 6.3. The errors at time $t = 2$ are plotted as functions of the number of time steps. For comparison, straight lines with slopes 4 and 5 are added

where Δ is the Laplacian operator, the diffusion coefficients $d_u = 0.02$, $d_v = 0.01$, and the bifurcation parameters $\alpha = 0.065$, $\beta = 0.035$. The initial conditions are Gaussian pulses

$$u(x, y, 0) = 1 - e^{-150\left((x-L)^2+(y-L)^2\right)}, \quad v(x, y, 0) = e^{-150\left((x-L)^2+2(y-L)^2\right)}.$$

Spatial discretization: For this example, we use standard second order finite differences using 150 grid points in each direction with mesh width $\Delta x = \Delta y = L/150$. This gives a stiff system of the form (1.1).

The system is then solved on the time interval $[0, 2]$ using constant step sizes. In the absence of an analytical solution of (6.4), a high-accuracy reference solution is computed using the `expRK4s6` method with a sufficient small time step. Errors are measured in a discrete maximum norm at the final time $t = 2$.

In Fig. 3, using the same number of time steps $N = 32, 64, 128, 256, 512, 1024$, we again display the order plots of the taken integrators. One can see that `expRK4s6` is much more accurate than `expRK4s5` and `expRK5s10` is slightly more accurate than `expRK5s8`.

In Fig. 4, we display the efficiency plot for which the time step sizes were chosen for each integrator to obtain about same error thresholds 10^{-i}, $i = 5, \ldots, 11$ (The corresponding number of time steps for each integrator are displayed in Table 2. As seen, given about the same level of accuracy, the new methods use smaller steps than the old ones of the same order, meaning that they can take larger step sizes). Again, `expRK4s6` is much faster than `expRK4s5` and it is interesting that this new fourth-order method turns out to be the most efficient (although for error thresholds tighter than 10^{-11} the new fifth-order method `expRK5s10` seems to become the most efficient).
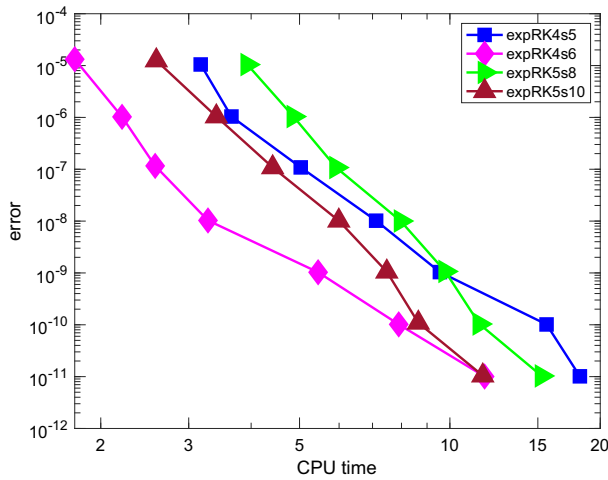
Fig. 4 Total CPU times of `expRK4s5`, `expRK4s6`, `expRK5s8`, and `expRK5s10` when applied to Example 6.3. The time step sizes were chosen in such a way that each integrator achieves about the same error thresholds 10^{-i}, $i = 5, \ldots, 11$. The errors at time $t = 2$ are plotted as functions of the total CPU time (in second)

Table 2 The number of time steps taken to achieve about the same error thresholds 10^{-i}, $i = 5, \ldots, 11$

Method	Error threshold versus number of time steps						
	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}	10^{-11}
`expRK4s5`	18	36	66	121	215	385	685
`expRK4s6`	10	19	28	46	122	230	420
`expRK5s8`	7	18	33	57	92	149	238
`expRK5s10`	8	17	30	51	82	130	208

The numerical results presented on the three examples above clearly confirm the advantage of constructing parallel stages expRK methods based on Theorem 3.1, leading to more efficient and accurate methods `expRK4s6` and `expRK5s10`.

References

1. 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**, 488–511 (2011)
2. Berland, H., Skaflestad, B.: Solving the nonlinear Schrödinger equation using exponential integrators. Technical report (2005)

3. Berland, H., Skaflestad, B., Wright, W.M.: Expint–a matlab package for exponential integrators. ACM Trans. Math. Softw. **33**(1), 4-es (2007)
4. Caliari, M., Kandolf, P., Ostermann, A., Rainer, S.: The Leja method revisited: backward error analysis for the matrix exponential. SIAM J. Sci. Comp. **38**(3), A1639–A1661 (2016)
5. Cazenave, T.: An introduction to nonlinear Schrödinger equations, vol. 22. Universidade Federal do Rio de Janeiro, Centro de Ciências Matemáticas e da (1989)
6. Gaudreault, S., Pudykiewicz, J.: An efficient exponential time integration method for the numerical solution of the shallow water equations on the sphere. J. Comput. Phys. **322**, 827–848 (2016)
7. Gray, P., Scott, S.: Autocatalytic reactions in the isothermal, continuous stirred tank reactor: oscillations and instabilities in the system $A + 2B \rightarrow 3B$; $B \rightarrow C$. Chem. Eng. Sci. **39**(6), 1087–1097 (1984)
8. Hochbruck, M., Ostermann, A.: Explicit exponential Runge-Kutta methods for semilinear parabolic problems. SIAM J. Numer. Anal. **43**, 1069–1090 (2005)
9. Hochbruck, M., Ostermann, A.: Exponential integrators. Acta Numer. **19**, 209–286 (2010)
10. Ju, L., Wang, Z.: Exponential time differencing Gauge method for incompressible viscous flows. Commun. Comput. Phys. **22**(2), 517–541 (2017)
11. Luan, V.T.: High-order exponential integrators. Ph.D. thesis, University of Innsbruck (2014)
12. Luan, V.T.: Fourth-order two-stage explicit exponential integrators for time-dependent PDEs. Appl. Numer. Math. **112**, 91–103 (2017)
13. Luan, V.T., Michels, D.: Efficient exponential time integration for simulating nonlinear coupled oscillators, J. Compt. Appl. Math. (Revised) (2020)
14. Luan, V.T., Ostermann, A.: Exponential B-series: the stiff case. SIAM J. Numer. Anal. **51**, 3431–3445 (2013)
15. Luan, V.T., Ostermann, A.: Explicit exponential Runge-Kutta methods of high order for parabolic problems. J. Comput. Appl. Math. **256**, 168–179 (2014)
16. Luan, V.T., Ostermann, A.: Exponential Rosenbrock methods of order five-construction, analysis and numerical comparisons. J. Comput. Appl. Math. **255**, 417–431 (2014)
17. Luan, V.T., Ostermann, A.: Stiff order conditions for exponential Runge–Kutta methods of order five. In: H.B. et al. (ed.) Modeling, Simulation and Optimization of Complex Processes-HPSC 2012, pp. 133–143. Springer, Berlin (2014)
18. Luan, V.T., Ostermann, A.: Parallel exponential Rosenbrock methods. Comput. Math. Appl. **71**, 1137–1150 (2016)
19. 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)
20. Michels, D.L., Luan, V.T., Tokman, M.: A stiffly accurate integrator for elastodynamic problems. ACM Trans. Graph. **36**(4), 116 (2017)
21. Niesen, J., Wright, W.M.: Algorithm 919: A Krylov subspace algorithm for evaluating the φ-functions appearing in exponential integrators. ACM Trans. Math. Soft. **38**(3), 22 (2012)
22. Pieper, K., Sockwell, K.C., Gunzburger, M.: Exponential time differencing for mimetic multilayer ocean models. J. Comput. Phys. **398**, 817–837 (2019)