A new class of exponential propagation iterative methods of Runge-Kutta type (EPIRK).

M. Tokman\textsuperscript{a,*}

\textsuperscript{a}School of Natural Sciences, University of California, 5200 N. Lake Road, Merced, CA 95343

Abstract

We propose a new class of the exponential propagation iterative methods of Runge-Kutta-type (EPIRK). The EPIRK schemes are exponential integrators that can be competitive with explicit and implicit methods for integration of large stiff systems of ODEs. Introducing the new, more general then previously proposed, ansatz for EPIRK schemes allows for more flexibility in deriving computationally efficient high-order integrators. Recent extension of the theory of B-series to exponential integrators [1] is used to derive classical order conditions for schemes up to order five. An algorithm to systematically solve these conditions is presented and several new fifth order schemes are constructed. Several numerical examples are used to verify the order of the methods is verified and illustrate the performance of the new schemes.

Keywords: exponential integrators, Krylov projections, exponential propagation iterative (EPI) methods, stiff systems, Newton-Krylov implicit methods, order conditions, B-series, large scale computing

2000 MSC: 65F10, 65F60, 65L04, 65L05, 65M22

1. Introduction

The first exponential integrators were introduced as early as the 1960’s [2, 3, 4] as an alternative to explicit and implicit schemes for solving initial value problems for autonomous systems of first-order ODEs

\[ y' = f(y), \quad y(x_0) = y_0, \quad y \in \mathbb{R}^N. \] (1)

An exponential integrator can be derived from the following reformulation of the problem as an integral equation. If the first-order Taylor expansion of \( f(y) \) around
$y_0$ exists we can re-write Eq. (1) as

$$y' = f(y_0) + f'(y_0)(y - y_0) + r(y),$$

where the nonlinear remainder of the first-order Taylor expansion denoted as $r(y) = f(y) - f(y_0) - f'(y_0)(y - y_0)$ and the Jacobian matrix $f'(y_0) \in \mathbb{R}^{N \times N}$ is assumed to exist. Using the integrating factor $e^{-f'(y_0)x}$ we can express the solution at time $x_0 + h$ in an integral form:

$$y(x_0 + h) = y_0 + h \varphi_1(hf'(y_0))f(y_0) + \int_{x_0}^{x_0+h} e^{f'(y_0)(x_0+h-x)}r(y(x))dx,$$

where $\varphi_1(z) = (e^z - 1)/z = \int_0^1 e^{z(1-\theta)}d\theta$ is an analytic function and its matrix-valued form $\varphi_1(hf'(y_0))$ is defined via the Taylor series expansion. After setting $A_0 = f'(y_0)$ and changing the integration variable to $\theta = (x - x_0)/h$ in Eq. (3) we obtain

$$y(x_0 + h) = y_0 + h \varphi_1(hA_0)f(y_0) + h \int_0^1 e^{hA_0(1-\theta)}r(y(x_0 + \theta h))d\theta.$$

This equation can be used as a starting point in derivation of an exponential integrator, when $y_0 = y(x_0)$ is interpreted as an approximation to the solution at the previous integration step and formula (4) is used to approximate $y(x_0 + h)$. To construct an exponential integrator an approximation scheme for the nonlinear integral in (4) and an algorithm to compute exponential-like functions of the matrix $A_0$ (e.g. $\varphi_1(hA_0)f(y_0)$) must be developed.

Initially exponential integrators were used only for problems of small to medium size $N$ since the schemes required evaluation of exponential and exponential-type functions of an $N \times N$ matrix. This operation is prohibitively expensive for large matrices if standard techniques such as Taylor or Padé approximations are employed [5]. Thus, exponential integrators were too computationally expensive compared to implicit and explicit schemes for large scale problems. However, a proposal to use a Krylov projection algorithm for this task has significantly reduced computational cost. This approach first appeared in a paper by Nauts and Wyatt [6] where they used Krylov projection to compute exponentials of symmetric matrices that represented discrete Hamiltonian operators. The idea was later used by Park and Light [7] to exponentially propagate the Schrödinger equation. Van der Vorst extended this method and proposed to apply Krylov projection to approximate general functions of matrices [8]. A resurgence of interest in exponential methods followed these ideas and a number of such schemes have been proposed over the past two decades [4, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18].
Many proposed exponential integrators targeted problems for which the nonlinear function $f(y)$ can be decomposed into a linear part and a nonlinear remainder $f(y) = Ay + g(y)$ (in this case a modified version of equation (4) is derived and used to construct an integrator using $A_0 = A$) \[4, 19, 10, 11, 12, 20, 18\]. Such an approach is appropriate where only the matrix $A$ is responsible for the stiffness of the system and the nonlinear remainder $g(y)$ is small. For many problems, however, particularly those addressed by high performance computing, such decomposition is not possible. These systems require a more dynamic integrator for which $A_0$ is a Jacobian matrix changing at every time step. Thus, several exponential-type functions of $A_0$ have to be evaluated at every integration step. While most exponential integrators can be generalized to solve this type of problems, the majority of them will be inefficient since they will require too many of these expensive calculations to be performed every time step. Particular care must be taken in constructing an efficient exponential scheme that minimizes the number of required evaluations of matrix exponential functions and the computational cost of each of these calculations. Exponential propagation iterative Runge-Kutta-type (EPIRK) methods employ the strategy of reusing scaled exponential function-vector products at each stage of the method to minimize the cost of their evaluation \[16, 21\]. Other strategies were used to construct EXP4 integrator \[15\] and Exponential Rosenbrock (EXPROS) schemes \[17\]. The convergence and performance analysis of exponential integrators yield these methods as competitors to standard implicit and explicit integrators in solving large scale stiff systems \[15, 22, 23\]; a recent review \[24\] outlines progress in the field.

At present only a few high-order exponential integrators have been proposed and thoroughly studied; for example, to our knowledge only one single-step fifth order scheme has been constructed, but not tested, in the literature \[25\]. The limited availability of such methods is in part due to the complexity of deriving and solving their order conditions. Recently, however, Butcher \[1\] extended his theory of the order conditions to exponential methods. This advance significantly simplifies the derivation of the order conditions for exponential integrators and opens new directions in constructing more sophisticated schemes.

In this paper we propose a new class of EPIRK methods. Using a more general ansatz then previously proposed, we discover a range of new exponential schemes which have favorable properties such as combination of a low number of stages and high-order. We use Butcher’s theory to derive classical order conditions for EPIRK methods up to order five. We then develop an algorithm to systematically solve the order conditions and construct new EPIRK methods. In particular, the flexibility of the new ansatz allows us to find fifth order EPIRK schemes with only three stages. Note that all previously proposed three-stage methods were of fourth order. We
verify the order of the newly found schemes and illustrate their performance using numerical examples.

The B-series approach yields classical order conditions since the coefficients of the global error depend on elementary differentials including the stiff operator $f'(y)$. The derivation of the stiff order EPIRK methods (i.e. methods with the global error bounded by $Ch^p$, where $h$ is the time step size and $C$ is a constant independent of the stiffness) will be the subject of future publications. However, the numerical examples included here indicate that even classical order EPIRK methods can perform well on stiff problems and be competitive with other stiff order integrators.

The paper is organized as follows. Section 2 describes the EPIRK schemes, their advantages and relationship to other exponential integrators. A brief overview of the Butcher’s extension of the B-series theory to exponential integrators is presented in Section 3. The order conditions for the fifth order exponential methods are derived and the algorithm for their solution is included in Section 4. In Section 5 the numerical examples are used to verify the order and illustrate performance of the new methods. Conclusions and future work are discussed in Section 6.

2. Construction of EPIRK methods

Building an exponential integrator using (4), consists of accomplishing two tasks:

(I) developing an approximation to the integral $h \int_0^1 e^{hA_0(1-\theta)}r(y(x_0 + \theta h))d\theta$, and

(II) building an algorithm to evaluate products of functions of matrices and vectors arising from the second term of the right-hand-side of (4) and possibly from the approximation chosen for the integral in (I).

A polynomial approximation to the nonlinear remainder function $r(y)$ in (4) will result in an exponential scheme which computes the solution as a linear combination of the products of type $\varphi_k(\gamma A)v_k$ with $v \in \mathbb{R}^N$, constant $\gamma$ and functions $\varphi_k(z)$ defined as

$$\varphi_k(z) = \int_0^1 e^{z(1-\theta)} \frac{\theta^{k-1}}{(k-1)!} d\theta, \quad k = 1, 2, \ldots \quad (5)$$

Note that $\varphi_1(z) = (e^z - 1)/z$. Obviously either Runge-Kutta or multistep approaches can be used in the derivation as well as any other construct that yields an approximation to the integral in (4). Once a certain ansatz for the approximation to the solution is assumed, the order conditions for the coefficients can be derived and solved to obtain exponential integrators of the desired accuracy.

After constructing the quadrature to approximate the integral in (4) one needs to address task (II) and to choose an algorithm to estimate the products $\varphi_k(\gamma A)v$, 

where $\gamma$ is a constant, $A$ is a matrix, and $v$ is a vector. For small systems a number of techniques such as Taylor or Padé expansions can be used [5]. If the system size $N$ is large, Krylov projection algorithm becomes the method of choice [26]. In this approach an Arnoldi iteration [27] is employed to approximate the product of a function of a matrix $\psi(A)$ and a vector $v$ by projection of the matrix and the vector onto the Krylov subspace $K_m(A, v) = \text{span}\{v, Av, ..., A^{m-1}v\}$. In the course of the iteration, the Krylov basis size $m$ required to meet prescribed tolerances is determined dynamically using appropriate residuals [28, 15]. Note that the sufficient size of the Krylov basis $m$ depends on the eigenvalues of the matrix $A$, the magnitude of the vector $v$ and the type of function $\psi(z)$. Based on analytical error bounds for matrices with specific spectrum [29] and numerical experiments with a variety of matrices [16, 23], it is expected that as $k$ increases the number of Krylov vectors $m$, required to approximate the product $\varphi_k(A)v$ with a prescribed tolerance, decreases.

Approximation of the products $\varphi_k(\gamma_{kl}A_0)v_{lk}$ constitutes the main computational cost of an exponential scheme [21]. Considering efficiency of an exponential-Krylov integrator from the perspective of tasks (I) and (II), it is clear that the computational cost of an exponential scheme for a large system (1) depends on two main features of the chosen method: (i) the total number of products $\varphi_k(\gamma_{kl}A_0)v_{lk}$ that have to be computed and (ii) the number of Krylov vectors that each of these products will require to achieve a prescribed accuracy. Thus, if we want to construct an exponential integrator of a certain order it is prudent to derive a scheme which minimizes both of these parameters, i.e. requires the minimum possible number of Krylov projections and chooses appropriate functions $\psi(z)$ and small vectors $v$ so that these projections are fast. These considerations are taken into account in construction of EPIRK methods as follows.

The first EPI method of Runge-Kutta-type was proposed in [16]. The nonlinear remainder function can be approximated by an interpolating polynomial

$$r(y(x)) = r(y(x_0 + \theta h)) \approx r(y_0) + \sum_{k=1}^{s} \binom{s\theta}{k} \Delta^k r(y_0), \quad (6)$$

where $0 \leq \theta \leq 1$, $\binom{s\theta}{k}$ is the binomial coefficient and $\Delta^k$ is the $k$-the forward difference constructed on the nodes $(x_0 + jh/s)$, $j = 0, ..., (s-1)$. If such an approximation is made, the nonlinear integral in (4) can be viewed as a linear combination of the products of the functions

$$\psi_j(z) = \int_0^1 e^{z(1-\theta)} \binom{s\theta}{j-1} d\theta \quad (7)$$
evaluated at $hA_0$ and the forward differences vectors $\Delta^k r$. Since the values $y(x_0 + jh/s)$ are not known, their approximations have to be computed to evaluate $\Delta^k r$. This is accomplished in EPIRK methods by applying the Runge-Kutta stage structure to approximate solution at intermediate nodes. Denoting the intermediate stages $Y_i$ we get the following general structure of an explicit EPIRK method

$$Y_i = y_0 + a_{i1}\psi_1(g_{i1}hA_0)hf(y_0) + \sum_{j=2}^{i-1} a_{ij}\psi_j(g_{ij}hA_0)h\Delta^{(j-1)}r(y_0), \quad i = 1, \ldots, (s - 1)$$

$$y_1 = y_0 + b_1\psi_1(g_{s1}hA_0)hf(y_0) + \sum_{j=2}^{s} b_j\psi_j(g_{sj}hA_0)h\Delta^{(j-1)}r(y_0), \quad (8)$$

where $s$ is the number of stages of the method. The forward differences $\Delta^{(j-1)}r(y_0)$ are computed on the nodes $y_0, Y_1, Y_2, \ldots, Y_{s-1}$ (recall that for any $y$, the remainder function can be evaluated as $r(y) = f(y) - f(y_0) - A_0(y - y_0)$ and $r(y_0) = 0$). The coefficients $a_{ij}, g_{ij}$ and $b_i$ are determined from the order conditions. Since $\binom{s-1}{k}$ is a polynomial in $\theta$, functions $\psi_j(z)$ can be expressed as linear combinations of the functions $\varphi_k(z)$ defined in (5). Note that functions $\psi_k(z)$ do not necessarily have to be of the form (7) as was assumed in the initial derivation in [16]. By relaxing this requirement we have more flexibility in constructing a scheme.

Thus we define a more general class of EPIRK methods with the following ansatz:

$$Y_i = y_0 + a_{i1}\psi_1(g_{i1}hA_0)hf(y_0) + \sum_{j=2}^{i-1} a_{ij}\psi_j(g_{ij}hA_0)h\Delta^{(j-1)}r(y_0), \quad i = 1, \ldots, (s - 1)$$

$$y_1 = y_0 + b_1\psi_1(g_{s1}hA_0)hf(y_0) + \sum_{j=2}^{s} b_j\psi_j(g_{sj}hA_0)h\Delta^{(j-1)}r(y_0), \quad (9)$$

where $\psi_{ij}(z)$ functions are defined as

$$\psi_{ij}(z) = \sum_{k=1}^{s} p_{ijk}\varphi_k(z). \quad (10)$$

The coefficients $a_{ij}, g_{ij}, b_j$ and $p_{ijk}$ are chosen based on the order conditions. The flexibility offered by this ansatz allows development of many efficient schemes as demonstrated in subsequent Sections. In fact, an interesting direction of research is development of tailor-made schemes where the coefficients are chosen so that the method is particularly efficient for a given problem. For example, it might be beneficial to choose the $g_{ij}$ coefficients and the $\psi_k(z)$ functions based on a specific problem.
and the properties of its Jacobian $J$ in such a way that the number of Krylov vectors required to estimate the product $\psi_k(\gamma J)b$ is minimized. Construction of such schemes will be addressed in future publications.

The EPIRK methods are structured to maximize computational efficiency for large scale problems. Several features of the schemes ensure that the computational cost of the Krylov projections, used to approximate the exponential matrix function-vector products, is reduced.

First, to minimize the total number of the Krylov projections needed each step, the scheme of type (9) reuses the term $\psi_{ij}(g_{ij}hA_0)\Delta^{(j-1)}r(y_0)$ ($j$ fixed) with different coefficients $g_{ij}$ at each stage. The Arnoldi algorithm is scale invariant and it takes only one iteration to approximate both $\psi(\gamma A)v$ and $\psi(\gamma'A)v$. Thus, all terms $\psi_{ij}(g_{ij}hA_0)\Delta^{(j-1)}r(Y_0)$, where $j$ is fixed and $i = 1, ..., s$, can be approximated using only one Arnoldi iteration. Once a Krylov basis is constructed, each of these terms will require a separate calculation of $\psi_{ij}(g_{ij}hH_m)$ for an $m \times m$ matrix $H_m$ which is a byproduct of the Krylov projection. But since $H_m$ is expected to be small all the computational cost of this operation is negligible compared to the Arnoldi iteration to construct the basis of $K_m$.

Second, each new Krylov projection has to approximate a higher order $\varphi_j(z)$ function. The convergence of the Krylov projection algorithm to approximate $\psi(A)b$ depends on eigenvalues of $A$ and the function $\psi(x)$. In [29] Hochbruck and Lubich derived a bound for the error of the Krylov approximation of $\psi(A)b$. The error bound allowed them to demonstrate that for negative semi-definite Hermitian matrices, the convergence of the Krylov iteration to approximate $\psi(x) = e^x$ is faster than for $\psi(x) = 1/(1-x)$. A similar result can be obtained for the functions $\psi(x) = \varphi_j(x)$. Intuitively, we can understand the result by considering the Taylor series of $\psi(A)b$ and realizing that the series for $e^x$ is converging faster [16]. Numerical studies [16, 15, 30, 23] supported these results and additionally demonstrated that as the order $j$ increases, functions $\varphi_j(z)$ require fewer Krylov vectors to approximate the product $\varphi_j(\gamma A)b$ to a given tolerance. It is difficult to show this theoretically for an arbitrary matrix $A$ since convergence of the Krylov iteration is strongly dependent on the spectrum of $A$. However, numerical experiments provide strong indication that this is indeed the case (e.g. see Section 5). Thus the use of higher order $\varphi_j(z)$ functions is computationally advantageous for an EPIRK scheme.

Finally, the third feature of the method is the choice of the vectors $v$ in the products $\psi(\gamma A)v$. For the schemes (9) the vectors $v$ are defined via forward differences of the function $r(y)$. The nonlinear remainder function $r(y)$ is expected to be small if $y(t)$ is close to $y_0$ (i.e. $t$ is close to $t_0$, $t \in [t_0, t_0 + h]$) and the forward differences built on a set of nodes $y$ close to $y_0$ will be even smaller in magnitude. If the stage
values $Y_i$ represent such a set of nodes, the magnitudes of vectors $\Delta_j r(y_0)$ will be small, thus reducing the magnitude of products $\psi(\gamma A)v$ and subsequently requiring fewer Krylov vectors to obtain their accurate approximations. This point can be particularly clearly seen if the stage values $Y_i$ can be interpreted as approximations to $y(t_0 + g_{i1} h)$ for small constants $g_{i1}$. For EPIRK schemes for which $\psi_{i1}(z) = \varphi_1(z)$ and an additional restriction on the coefficients $a_{i1} = g_{i1}$ is imposed, the stage values $Y_i$ are approximations of at least order two to $y(t_0 + g_{i1} h)$ (the first and second terms in (9) represent solution via exponential Euler method). Thus for such EPIRK schemes, the $Y_i$'s are precisely the approximations to the solution on a small interval $[t_0, t_0 + h]$. Similar argument could be made if we were to formulate implicit EPIRK methods (i.e. extend the summation indices in (9) to change from $j = 2$ to $s$). For non-implicit EPIRK methods without the restrictions on functions $\psi_{i1}$ and the coefficients $a_{i1}, g_{i1}$, this rationale becomes more heuristic but still appears to hold in practice. An explanation of the relationship between EPIRK and exponential Rosenbrock methods presented below, further clarifies and supports this argument.

The three structural features described above distinguish the EPIRK methods from other exponential integrators. For example, the EXP4 scheme proposed in [15] constructs an approximation to (4) that requires three evaluations of products of type $\varphi_1(\gamma A_0)v_i$ for three different vectors $v_i$. Each of the subsequent vectors $v_i$ is smaller in norm for a reason similar to the third feature of EPIRK4, but the exponential function $\varphi_1(z)$ is the same for each of these products, no higher order $\varphi_j(z)$ are used in the EXP4 formulation.

A more closely related to EPIRK schemes methods are the exponential Rosenbrock integrators [3, 17]. These schemes are direct extensions of exponential integrators proposed in the literature in several forms [3, 31, 32] for the linearized problem $f(y) = Ly + N(y)$ with Jacobian replacing the static matrix $L$. The original formulation of the exponential Rosenbrock (EXPROS) methods is

$$U_i = e^{c_i h A_0} y_0 + h \sum_{j=2}^{i-1} a_{ij}(hA_0) g(U_j), \quad i = 1, \ldots, s \tag{11}$$

$$y_1 = e^{h A_0} y_0 + h \sum_{i=2}^{s} b_i(hA_0) g(U_j), \tag{12}$$

where $g(y) = f(y) - A_0 y = r(y) + f(y_0) - A_0 y_0$ and functions $a_{ij}(z)$ and $b_i(z)$ are linear combinations of $\varphi_k(z)$. Formulation (12) requires computation of the exponential terms $\exp(hJ_n)v$ in addition to $\varphi$-functions-vector products which is usually more computationally expensive. However, in [22] the EXPROS methods were reformulated using the EPIRK framework to avoid this additional computation.
and obtain more efficient integrators of type

\[ U_i = y_0 + c_i h \psi_1 (c_i h A_0) f(y_0) + h \sum_{j=2}^{i-1} a_{ij} (h A_0) r(U_j), \quad i = 1, \ldots, s \quad (13) \]

\[ y_1 = y_0 + h \psi_1 (h A_0) f(y_0) + h \sum_{i=2}^s b_i (h A_0) r(U_j). \quad (14) \]

Any EXPROS method can be re-written in the EPIRK form. The reverse, however, is not true. The main difference between EPIRK and EXPROS methods is the use of coefficients \( g_{ij} \) in EPIRK. In EXPROS all \( g_{ij} \) are set to 1 except \( g_{i1} \) and additional restrictions \( g_{i1} = a_{i1} = c_i \) are imposed. By allowing \( g_{ij} \) to vary we can derive additional methods, for example, three stage fifth order methods as described in Section 4. Another distinction is that the EPIRK methods formulation employs the forward difference of \( r(y) \) rather than the function itself. This feature can improve efficiency for higher order methods by reducing the magnitude of \( v \) vectors in products \( \psi(\gamma A)v \) and thus decreasing the number of Krylov vectors needed to approximate such products. It is instructive to rewrite a three-stage fifth-order EPIRK method (28) with \( \psi_1(z) = \varphi_1(z) \) in form close to (12) to obtain:

\[
\begin{align*}
U_1 &= (a_{11} - g_{11}) y_0 + (1 - g_{11} + a_{11}) e^{g_{11} A_0 h} y_0 + h a_{11} \varphi_1 (g_{11} h A_0) g(y_0), \\
U_2 &= (a_{21} - g_{21}) y_0 + (1 - g_{21} + a_{21}) e^{g_{21} A_0 h} y_0 + \\
&\quad + h (a_{21} \varphi_1 (g_{21} h A_0) - a_{21} \varphi_2 (g_{22} h A_0)) g(y_0) + h a_{22} \varphi_2 (g_{22} h A_0) g(U_1), \\
y_1 &= (b_{11} - g_{31}) y_0 + (1 - g_{31} + b_{11}) e^{g_{31} A_0 h} y_0 + \\
&\quad + h (b_{11} \varphi_1 (g_{31} h A_0) - b_{11} \varphi_2 (g_{32} h A_0) + b_{2} \varphi_3 (g_{33} h A_0)) g(y_0) + \\
&\quad + h (b_{2} \varphi_2 (g_{32} h A_0) - 2b_{2} \varphi_3 (g_{33} h A_0)) g(U_1) + h b_{3} \varphi_3 (g_{33} h A_0) g(U_2). \quad (15)
\end{align*}
\]

As can be seen in these formulas, the terms \( e^{g_{i1} A_0 h} y_0 \) in (12) are replaced by a weighted sums of \( y_0 \) and \( e^{g_{i1} A_0 h} y_0 \) if \( a_{i1} \neq g_{i1} \). Also if \( \psi_1(z) \neq \varphi_1(z) \) the weights of \( g(y_0) \) in the third terms of the formulas would also be changed. Note that while for such methods the stage values \( U_i \) will not necessarily be exponential Euler approximations of \( y(t_0 + g_{i1} h) \), depending on the values of the coefficients \( a_{ij}, g_{ij}, p_{ij} \) \( U_i \)’s might still be relatively close to \( y_0 \). It is also clear that the stiff order conditions derived for exponential Rosenbrock methods will not directly apply to EPIRK. Derivation of stiff order conditions for EPIRK is the subject of current investigation and will be reported elsewhere.

It is important to note that the construction similar to the EPIRK method can be also applied to problems where the right-hand-side of (1) can be split as \( f(y) = Ly + \)
$N(y)$, where $L$ is the stiff part of the operator constant in time. The order conditions can be derived and specific methods of the EPIRK type can be constructed. The general class of such methods will contain Exponential-Time-Differencing (ETD) [10] and exponential Runge-Kutta schemes [4, 20]. Developing schemes specifically for such problems is outside the scope of this paper. However, if we compare solving these systems with EPIRK methods versus schemes like ETD, we observe that the former evaluates products like $\psi_{ij}(\gamma A_0)v$, while the latter requires computation of $\psi_{ij}(\gamma L)v$. If there exists a very efficient algorithm to approximate $\psi_{ij}(\gamma L)v$ separately, these matrices can be computed once, stored and reused every time step for the whole time integration interval. However, for very large stiff matrices $L$, it is unlikely that such algorithm and the associated storage requirements will be feasible in terms of the computational complexity and memory demands. In such cases, the products $\psi_{ij}(\gamma L)v$ will still have to be evaluated at every time step using algorithms such as the Krylov projection. Thus the computational complexity of EPIRK methods and ETD or exponential Runge-Kutta methods will be comparable and the methods which require fewest number of the fastest converging Krylov projections will be more efficient. Consider, for example, a forth-order Runge-Kutta ETD scheme ETD4RK (formulas (23)-(25) in [10]):

\[
U_1 = e^{Lh/2}y_0 + \varphi_1(Lh/2)(h/2)f(y_0),
\]
\[
U_2 = e^{Lh/2}y_0 + \varphi_1(Lh/2)(h/2)f(U_1),
\]
\[
U_3 = e^{Lh/2}U_1 + \varphi_1(Lh/2)(h/2)(2f(U_2) - f(y_0)),
\]
\[
y_1 = e^{Lh/2}y_0 + [e^{Lh}(4 - 3hL + h^2L^2)]f(y_0)...
\]
\[
+ [2 + hL + e^{Lh}(-2 + Lh)]2(f(U_1) + f(U_2))...
\]
\[
+ [-4 - 3Lh - (Lh)^2 + e^{Lh}(4 - Lh)](hL)^{-3}hf(U_3),
\]

If written in EPIRK form (use $e^z = \varphi_1(z)z + 1$ and collect terms to form products $\psi_{ij}(\gamma L)v$), this method will be a three-stage scheme of order four, requiring four Krylov projections each time step. Thus it will be more efficient to solve a large stiff problem with the EXP4, EXPROS4 or EPIRK4 methods compared to ETD4RK. The Gray-Scott equation included in Section 5 is one of such numerical examples.

Additional comparisons of the structure and performance of exponential integrators can be also found in [21, 23].
3. Overview of the order conditions derivation for exponential integrators using B-series

In this section we outline Butcher’s extension of order conditions theory to exponential integrators [1]. The purpose of this overview is to provide a brief but intuitive and straightforward explanation that can enable readers, not familiar with the theory, to use the formalism to construct new integrators. A thorough treatment of elementary differentials and rooted trees as means to derive order conditions for Runge-Kutta methods can be found in [33, 34, 35]. Here we only outline the key points.

At the base of the theory is the observation that if \( y' = f(y) \), then operators involving \( y \) and its derivatives, such as \( f(y(t_0 + h)) \) or the nested expression \( f(y_0 + hf(y_0)) \), can be written as an expansion in terms of elementary differentials evaluated at \( y_0 \) [34]. As an example, consider operator \( y(x_0 + h) \) which represents the solution to the ODE \( y' = f(y) \) at \( x_0 + h \). Using Taylor series, the ODE itself and denoting \( f^{(n)}(y_0) = f^{(n)} \) this operator can be written as

\[
y(x_0 + h) = y(x_0) + y'(x_0)h + \frac{1}{2!}h^2 y''(x_0) + \frac{1}{3!}h^3 y'''(x_0) + \frac{1}{4!}h^4 y^{(4)}(x_0) + \cdots
\]

\[
= y_0 + hf + \frac{1}{2!}h^2 f' f + \frac{1}{3!}h^3 \left( f''(f, f) + f' f' f \right)
\]

\[
+ \frac{1}{4!}h^4 \left( f^{(3)}(f, f, f) + 3f''(f, f') f + f' f''(f, f) + f' f' f' f \right) + \cdots
\]

Note that higher order derivatives of \( f(y) \) are multilinear operators, e.g. \( f'' \) is a bilinear form which acts on the linear operator \( f \) in the expression \( f''(f, f) \). The products of \( f \) and its derivatives called elementary differentials are used as a basis for the expansion in (16) (below we will denote these elementary differentials as \( F(t)(y(x)) \)). To simplify understanding of their structure and to automate construction of higher order differentials, the elementary differentials can be associated with rooted tree graphs. Table 1 shows elementary differentials up to order five along with the corresponding tree representation. The set of all rooted trees is usually denoted as \( T \). Each rooted tree \( t \in T \) has an associated elementary differential.

Two useful functions on \( T \), describing properties of each tree, are the order of a tree \( r(t) \) and the symmetry of a tree \( \sigma(t) \) \( t \in T \). \( r(t) \) is defined as the number of vertices of a particular tree, and \( \sigma(t) \) is the order of its symmetry group. A detailed description of \( \sigma(t) \) and the recursive and direct formulas for computing this function can be found in [34]. Here we just describe a simple algorithm to compute \( \sigma(t) \). First, we start with a pictorial representation of a tree and group vertices by the number of edges that separates them from the root. Second, within each group we compute
the number of vertex permutations that do not change the pictorial representation of the tree. Finally, the value of $\sigma(t)$ is computed by multiplying all of the obtained numbers of permutations for each group. Table 1 lists values of $r(t)$ and $\sigma(t)$ for all trees up to order five.

Table 1: Trees representation of the elementary differentials up to order five and B-series coefficients of $y(t_0 + h)$ corresponding to these elementary differentials.

<table>
<thead>
<tr>
<th>Trees, $r(t)$</th>
<th>Order, $r(t)$</th>
<th>Symmetry, $\sigma(t)$</th>
<th>B-series coefficients of $y(t_0 + h)$, $1/\gamma(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{5}$</td>
</tr>
<tr>
<td>$t_2$</td>
<td>2</td>
<td>1</td>
<td>$\frac{1}{10}$</td>
</tr>
<tr>
<td>$t_3$</td>
<td>3</td>
<td>2</td>
<td>$\frac{1}{15}$</td>
</tr>
<tr>
<td>$t_4$</td>
<td>3</td>
<td>1</td>
<td>$\frac{1}{1} F(t_1)$</td>
</tr>
<tr>
<td>$t_5$</td>
<td>4</td>
<td>6</td>
<td>$\frac{1}{5} F(t_2)$</td>
</tr>
<tr>
<td>$t_6$</td>
<td>4</td>
<td>1</td>
<td>$\frac{1}{2} F(t_3)$</td>
</tr>
<tr>
<td>$t_7$</td>
<td>4</td>
<td>2</td>
<td>$\frac{1}{4} F(t_4)$</td>
</tr>
<tr>
<td>$t_8$</td>
<td>4</td>
<td>1</td>
<td>$\frac{1}{8} F(t_5)$</td>
</tr>
<tr>
<td>$t_9$</td>
<td>5</td>
<td>24</td>
<td>$\frac{1}{5} F(t_6)$</td>
</tr>
<tr>
<td>$t_{10}$</td>
<td>5</td>
<td>1</td>
<td>$\frac{1}{10} F(t_7)$</td>
</tr>
<tr>
<td>$t_{11}$</td>
<td>5</td>
<td>2</td>
<td>$\frac{1}{15} F(t_8)$</td>
</tr>
<tr>
<td>$t_{12}$</td>
<td>5</td>
<td>1</td>
<td>$\frac{1}{2} F(t_9)$</td>
</tr>
<tr>
<td>$t_{13}$</td>
<td>5</td>
<td>2</td>
<td>$\frac{1}{4} F(t_{10})$</td>
</tr>
<tr>
<td>$t_{14}$</td>
<td>5</td>
<td>1</td>
<td>$\frac{1}{8} F(t_{11})$</td>
</tr>
<tr>
<td>$t_{15}$</td>
<td>5</td>
<td>2</td>
<td>$\frac{1}{16} F(t_{12})$</td>
</tr>
<tr>
<td>$t_{16}$</td>
<td>5</td>
<td>1</td>
<td>$\frac{1}{32} F(t_{13})$</td>
</tr>
<tr>
<td>$t_{17}$</td>
<td>5</td>
<td>2</td>
<td>$\frac{1}{64} F(t_{14})$</td>
</tr>
<tr>
<td>$t_{18}$</td>
<td>5</td>
<td>1</td>
<td>$\frac{1}{128} F(t_{15})$</td>
</tr>
</tbody>
</table>

The expansions in terms of elementary differentials can be also viewed algebraically by mapping the set of trees onto a set of real numbers. If $T^# = \{\emptyset, T\}$ is the set of trees together with an empty tree $\emptyset$, and $a : T^# \to \mathbb{R}$ defines a set of real coefficients $a(t)$ then the following expansion in terms of elementary differentials $F(t)(y(x))$

$$B(a, y(x)) = a(\emptyset) y(x) + \sum_{t \in T} a(t) \frac{h^r(t)}{\sigma(t)} F(t)(y(x))$$

(17)

is called a B-series [34, 35]. As an illustration consider the B-series representing the solution $y(x_0 + h)$ in (16). Table 1 provides coefficients $a(t)$ of the B-series for all the terms up to order five.

The set of all mappings $T^# \to \mathbb{R}$, i.e. B-series, is usually denoted as $G$. It is useful to distinguish mappings for which the empty tree $\emptyset$ is mapped onto 0 as, for
instance, for the B-series of operator \( y'(x_0) = f(y(x_0)) \) whose coefficients are all zero except for \( a(t_1) \), i.e. \( a = \{0, 1, 0, 0, ..., 0, ...\} \). The set of such mappings is denoted \( G_0 \). Another important set of B-series is \( G_1 \), i.e. the set for which \( a(\emptyset) = 1 \), an example of such operator is \( y(x_0 + h) \). Operators \( f(y_0) \) and \( y(x_0 + h) \) are frequently encountered in the theory of order conditions for Runge-Kutta methods and their the B-series are denoted by \( D = \{0, 1, 0, 0, ..., 0, ...\} \in G_0 \) and

\[
E = B(a, y(x_0 + h)) = \left\{1, \left[\frac{1}{\gamma(t_i)}\right]_i^\infty\right\}, \quad E \in G_1, \tag{18}
\]

where the function \( \gamma(t) \) is called the density of a tree. \( \gamma(t) \) can be calculated either by using a recursion formula, or by assigning each vertex a number representing the total number of vertices to which the chosen vertex serves as a parent, including the chosen vertex itself, and computing the product of all such numbers for each vertex of the tree \( t \) (see [34] for details). The values for \( \gamma(t_i) \) for all trees up to order five are given in Table 1.

One of the main benefits of representing operators in the B-series form is that it allows to simplify and automate computation of sums, differences and, most importantly, compositions of two operators in terms of elementary differentials. Obviously, if \( B(a, y(x)) \) and \( B(b, y(x)) \) are two B-series with coefficients \( a_i = a(t_i) \) and \( b_i = b(t_i) \) \( (t \in T) \) and \( \alpha, \beta \) are real numbers, a B-series representing the linear combination \( B(c, y(x)) = \alpha B(a, y(x)) + \beta B(b, y(x)) \) has coefficients \( c_i = \alpha a_i + \beta b_i \).

A more complex operation that can be performed with B-series is their composition. An example where such computation is needed is the calculation of the elementary differentials expansion of \( y_2 = hf(y_1) = hf(y_0 + hf(y_0)) \). It is clear that the B-series coefficients of \( y_1 = y_0 + hf(y_0) \) are

\[
a = \{a(\emptyset) = 1, a(t_1) = 1, a(t_i) = 0 \text{ for } i > 1\} \tag{19}
\]

and the coefficients of \( hf(y_0) \) are

\[
b = \{b(\emptyset) = 0, b(t_1) = 1, b(t_i) = 0 \text{ for } i > 1\}. \tag{20}
\]

We would like to be able to compute B-series coefficients of \( y_2 = hf(y_0 + hf(y_0)) \) from the coefficients \( a \) and \( b \). A composition operation constructed in [33] accomplishes exactly that.

Suppose that the following two operators are represented by the corresponding B-series \( y_0 = B(a, y_0(x)) \) and \( y_1 = B(b, y_1(x)) \). It can be shown that the operator \( y_2(x) = y_1(y_0(x)) \) can be also represented by a B-series \( B(c, y_2(x)) = B(b, B(a, y_0)) \) whose coefficients \( c \) are computed from the coefficients \( a \) and \( b \). The multiplicative
operation \((a \circ b) : G_1 \times G \rightarrow G\) is defined to represent such composition of the B-series. Note that the order in the composition operation is important and \(a \circ b\) represents \(B(b, B(a, y))\), i.e. the \(a\) coefficients of the B-series of \(y\) are composed with the B-series \(b\). The general formula for the coefficients \(c = (a \circ b)\) is quite complex and we refer the reader to the sources that provide exhaustive treatment [34, 35]. Here we simply list the composed coefficients \(c_i \ (i = 0, 1, \ldots, 17)\) for trees up to order five (Table 2) since we will use them in the subsequent sections.

Finally, we note that in order to compute the B-series representation of the exponential integrators of type (9) we need to know how to calculate the B-series coefficients for function \(r(y) = f(y) - f(y_0) - f'(y_0)(y - y_0)\) and the products of type \(\psi(hf'(y_0))hr(y)\). This is accomplished using a key observation of Butcher [1] that if an operator \(y_1\) with the corresponding B-series coefficients \(a(t_i)\) is multiplied by \(hf'(y_0)\), computing the B-series coefficients of the resulting B-series of \(hf'(y_0)y_1\) is simple since coefficient of tree in the expansion of \(y_1\) gets shifted to a coefficient of a tree with one more vertex which is placed at the root. Figure 1 illustrates how the B-series coefficients \(a(t_2), a(t_3), a(t_6)\) become coefficients \(b(t_4), b(t_7), b(t_{15})\) correspondingly after the expansion of \(y_1\) is multiplied by \(hf'(y_0)\) on the left. A tree obtained by adding an extra vertex to a tree \(t \in T^\#\) as a root is usually denoted as \([t]\). Multiplication by \(hf'(y_0)\) can be represented by a function \(J : G_0 \rightarrow G_0\) such that for any \(a \in G_0\) the coefficients of the resulting \(b = J(a)\) are

\[
b(t_i) = \begin{cases} a(t_j), & \text{if } t_i = [t_j] \\ 0, & \text{otherwise} \end{cases} \tag{21}
\]

Figure 1: Shift of B-series coefficients under the operation of multiplying elementary differentials by \(f'(y_0)\)

\(a\) \(b(t_4) = (J(a))(t_4) = a(t_2)\) \(b\) \(b(t_6) = (J(a))(t_6) = a(t_3)\) \(c\) \(b(t_{15}) = (J(a))(t_{15}) = a(t_6)\)

Using the definition of the function \(J\) we can, for example, find the B-series coefficients of an operator \(hr(y) = h(f(y) - f(y_0) - f'(y_0)(y - y_0))\) by performing
operations of addition, composition and application of \( J \) on the B-series of operators comprising \( hr(y) \). If \( a \) is the B-series representing operator \( y \) and \( 1 \) is the B-series representing \( y_0 \) (the coefficients of the latter are \( 1 = \{1, 0, \ldots, 0, \ldots\} \)) then B-series \( R \in G_0 \) of \( hr(y) \) can be computed as

\[
R(a) = a \circ D - D - J(a - 1).
\]

(22)

Furthermore, since we know how to compute \( b = J(a) \), the B-series coefficients for any term of type \((hf'(y_0))^n y \) (where \( y \) is the operator corresponding to B-series \( a \)) can be calculated by sequential application of \( J \). Thus, we can compute the B-series of any function of \( hf'(y_0) \) which can be expanded as a polynomial or has a Taylor series, e.g. the B-series \( \Psi(a) \) of a product \( \psi(hf'(y_0))y \) where

\[
\psi(z) = c_0 + c_1 z + c_2 z^2 + \ldots + c_n z^n + \ldots
\]

(23)

and \( a \) is the B-series of \( y \) can be evaluated as

\[
\Psi(a) = c_0 a + c_1 J(a) + c_2 J(J(a)) + \ldots + c_n J^n(a) + \ldots
\]

(24)

In the next section we use the theoretical tools outlined above to derive the order conditions for EPIRK methods up to order five.

4. Derivation of EPIRK methods of order up to five

The general form of EPIRK methods (9) allows many options in terms of selecting functions \( \psi_{ij}(z) \). Since we are particularly interested in constructing schemes which optimize efficiency we will focus on the schemes which reuse the products of type \( \psi_{ij}(g_{ij}hA_0)v \) every stage without having to recompute the Krylov projection (i.e. changing vector \( v \)). As a further simplification we also use the same functions \( \psi_{ij}(z) \) for a fixed \( j \). Thus, we consider methods of form (9) with

\[
\psi_{ij}(z) = \psi_j(z) = \sum_{k=1}^{j} p_{jk} \varphi_k(z).
\]

(25)

To derive the order conditions for the EPIRK methods (Eqs. 9,25) we re-write them as sums and compositions of the B-series as demonstrated in (27). Since these computations are quite cumbersome we have implemented a Mathematica script which computes sums, differences and compositions of given B-series as well as the action of a provided function \( \psi(J) \) of the Jacobian operator \( J \) on a B-series. Using
Table 2: Coefficients of the composite B-series $B(c, y) = B(b, B(a, y))$ for trees up to order five.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$a(t_i)$</th>
<th>$b(t_i)$</th>
<th>$c_i = (a \circ b)(t_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>$b_0$</td>
<td>$b_0$</td>
</tr>
<tr>
<td>1</td>
<td>$a_1$</td>
<td>$b_1$</td>
<td>$a_1 b_0 + b_1$</td>
</tr>
<tr>
<td>2</td>
<td>$a_2$</td>
<td>$b_2$</td>
<td>$a_2 b_0 + b_1 a_1 + b_2$</td>
</tr>
<tr>
<td>3</td>
<td>$a_3$</td>
<td>$b_3$</td>
<td>$a_3 b_0 + a_1^2 b_1 + 2 a_1 b_2 + b_3$</td>
</tr>
<tr>
<td>4</td>
<td>$a_4$</td>
<td>$b_4$</td>
<td>$a_4 b_0 + a_2 b_1 + a_1 b_2 + b_4$</td>
</tr>
<tr>
<td>5</td>
<td>$a_5$</td>
<td>$b_5$</td>
<td>$a_5 b_0 + a_3^2 b_1 + 3 a_1^2 b_2 + 3 a_1 b_3 + b_5$</td>
</tr>
<tr>
<td>6</td>
<td>$a_6$</td>
<td>$b_6$</td>
<td>$a_6 b_0 + a_1 a_2 b_1 + (a_1^2 + a_2) b_2 + a_1 (b_3 + b_4) + b_6$</td>
</tr>
<tr>
<td>7</td>
<td>$a_7$</td>
<td>$b_7$</td>
<td>$a_7 b_0 + a_3 b_1 + a_1^2 b_2 + 2 a_1 b_4 + b_7$</td>
</tr>
<tr>
<td>8</td>
<td>$a_8$</td>
<td>$b_8$</td>
<td>$a_8 b_0 + a_2 b_1 + a_1 b_2 + b_8$</td>
</tr>
<tr>
<td>9</td>
<td>$a_9$</td>
<td>$b_9$</td>
<td>$a_9 b_0 + a_1^4 b_1 + 4 a_1^2 b_2 + 6 a_1 b_3 + 4 a_1 b_5 + b_9$</td>
</tr>
<tr>
<td>10</td>
<td>$a_{10}$</td>
<td>$b_{10}$</td>
<td>$a_{10} b_0 + a_1^2 a_2 b_1 + (2 a_1 a_2 + a_1^2) b_2 + (2 a_1^2 + a_2) b_3 + a_1^2 b_4 + a_1 b_5 + 2 a_1 b_6 + b_{10}$</td>
</tr>
<tr>
<td>11</td>
<td>$a_{11}$</td>
<td>$b_{11}$</td>
<td>$a_{11} b_0 + a_3 b_1 + (a_1^3 + a_3) b_2 + a_1^3 b_3 + 2 a_1^2 b_4 + 2 a_1 b_6 + a_1 b_7 + b_{11}$</td>
</tr>
<tr>
<td>12</td>
<td>$a_{12}$</td>
<td>$b_{12}$</td>
<td>$a_{12} b_0 + a_4 b_1 + (a_1 a_2 + a_4) b_2 + a_2 b_3 + a_1^2 b_4 + a_1 b_6 + a_1 b_8 + b_{12}$</td>
</tr>
<tr>
<td>13</td>
<td>$a_{13}$</td>
<td>$b_{13}$</td>
<td>$a_{13} b_0 + a_2^2 b_1 + 2 a_1 a_2 b_2 + a_1^2 b_3 + 2 a_2 b_4 + 2 a_1 b_6 + b_{13}$</td>
</tr>
<tr>
<td>14</td>
<td>$a_{14}$</td>
<td>$b_{14}$</td>
<td>$a_{14} b_0 + a_5 b_1 + a_1^4 b_2 + 3 a_1^2 b_3 + 3 a_1 b_7 + b_{14}$</td>
</tr>
<tr>
<td>15</td>
<td>$a_{15}$</td>
<td>$b_{15}$</td>
<td>$a_{15} b_0 + a_6 b_1 + a_1 a_2 b_2 + (a_1^2 + a_2) b_4 + a_1 b_7 + a_1 b_8 + b_{15}$</td>
</tr>
<tr>
<td>16</td>
<td>$a_{16}$</td>
<td>$b_{16}$</td>
<td>$a_{16} b_0 + a_3 b_1 + a_3 b_2 + a_1^2 b_4 + 2 a_1 b_8 + b_{16}$</td>
</tr>
<tr>
<td>17</td>
<td>$a_{17}$</td>
<td>$b_{17}$</td>
<td>$a_{17} b_0 + a_8 b_1 + a_4 b_2 + a_2 b_4 + a_1 b_8 + b_{17}$</td>
</tr>
</tbody>
</table>

the script we find B-series for operators such as $Y_i$ or $y_1$ by sequentially performing these operations, e.g. the B-series $\Upsilon_1$ and $\Upsilon_2$ for $Y_1$ and $Y_2$ can be computed as

$$\Upsilon_1 = \mathbf{1} + a_{11} \Psi_1 (g_{11} D),$$
$$\Upsilon_2 = \mathbf{1} + a_{21} \Psi_1 (g_{21} D) + a_{22} \Psi_2 (g_{22} R(\Upsilon_1)).$$

(26) (27)

**Derivation of three-stage fifth-order methods:** As discussed above, an efficient integrator minimizes the number of Krylov projections that have to be computed each time step. Thus, we would like to derive EPIRK methods of the highest possible order but with fewest possible stages. We first focus on deriving three-stage methods of order five. The general form of such methods is

$$Y_1 = y_0 + a_{11} \psi_1 (g_{11} h A_0) h f(y_0)$$
$$Y_2 = y_0 + a_{21} \psi_1 (g_{21} h A_0) h f(y_0) + a_{22} \psi_2 (g_{22} h A_0) h r(Y_1)$$
$$y_1 = y_0 + b_1 \psi_1 (g_{31} h A_0) h f(y_0) + b_2 \psi_2 (g_{32} h A_0) h r(Y_1) + b_3 \psi_3 (g_{33} h A_0) h \Delta^2 r(y_0),$$

(28)
with \( \Delta^2 r(y_0) = r(y_0) - 2r(Y_1) + r(Y_2) = -2r(Y_1) + r(Y_2) \) (recall \( r(y_0) = 0 \)) and

\[
\psi_j(z) = \sum_{k=1}^{j} p_{jk} \varphi_k(z) = \sum_{n=0}^{\infty} \left( \sum_{k=1}^{j} \frac{p_{jk}}{(n+k)!} \right) z^n
\]

(29)

Using the Taylor expansion coefficients of functions \( \psi_j(z) \) in (29) to define the corresponding B-series valued functions \( \Psi_j(a) \) as in (24), and comparing the B-series of \( y_1 \) in (28) with the B-series of the exact solution (Table 1), we arrive at the seventeen order conditions corresponding to trees up to order five. The order conditions are listed in Table 3.

By analyzing the order conditions of Table 3 we develop the following algorithm to reduce the system of seventeen nonlinear algebraic equations to a polynomial in \( a_{11} \) with coefficients parameterized by \( p_{ij} \):

Step 1) Note, the order conditions are independent of \( g_{22} \) so this coefficient can be chosen arbitrarily;

Step 2) Conditions (C1, C2, C8, C17) imply \( g_{31} = 1 \) and \( b_1 = 1/p_{11} \) and \( g_{31} = 1 \) (Note: all other order conditions are independent of the coefficient \( b_1 \));

Step 3) Conditions (C3, C5) form a linear system with respect to \( b_2 \) and \( b_3 \) and are solved for these coefficients;

Step 4) Substitution of expressions obtained in Step 3 for \( b_2 \) and \( b_3 \) into (C9) yields a simple formula for \( a_{21} = 3(5a_{11}p_{11} - 4)/(5p_{11}(4a_{11}p_{11} - 3)) \);

Step 5) Conditions (C7, C14) form a linear system with respect to \( g_{32}, g_{33} \) which is solved to obtain expressions for these coefficients in terms of \( a_{11} \) and \( p_{ij} \)’s;

Step 6) Linear system (C6), (C10) yields expressions for \( g_{11} \) and \( g_{21} \) which automatically satisfy condition (C15);

Step 7) Condition (C11) is solved for \( a_{22} \).

After recursive substitution of expressions for \( a_{21}, a_{22}, b_2, b_3, g_{11}, g_{12}, g_{13}, g_{21}, g_{31}, g_{32}, g_{33} \) obtained with this algorithm, all order conditions hold true except (C1, C2, C4, C8, C16). These remaining conditions, however, are written as functions of \( a_{11} \) and \( p_{ij} \) (i, j = 1, 2, 3 and \( i \leq j \)). In fact, (C1) is just a formula for \( b_1 \) and all four conditions (C2, C4, C8, C16) can be reduced to the same expression - a twelfth-order polynomial in \( a_{11}p_{11} \) with coefficients parametrized by the rest of \( p_{ij} \)’s.

In order to construct a specific three-stage fifth order method we can choose values for \( p_{ij} \)’s, compute roots of the polynomial for \( a_{11} \) and proceed to evaluate the remaining coefficients of the method by using expressions obtained in Steps (1-6). Thus, we have derived a whole family of the three-stage fifth order EPIRK methods.
Table 3: Order conditions for EPIRK methods (9) up to order five.

<table>
<thead>
<tr>
<th>Tag Cᵢ</th>
<th>Order condition corresponding to a tree tᵢ</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>$b_1 p_{11} - 1 = 0$</td>
</tr>
<tr>
<td>C2</td>
<td>$b_1 g_{31} p_{11} - 1 = 0$</td>
</tr>
</tbody>
</table>
| C3     | $6a_{11} b_2 p_{11} p_{21} + 3a_{11} b_2 p_{11} p_{22} - 12a_{11} b_3 p_{11} p_{31} + 6a_{21} b_3 p_{11} p_{31} \ldots$  
|        | $- 6a_{11} b_2 p_{11} p_{32} + 3a_{21} b_3 p_{11} p_{32} - 2a_{11} b_3 p_{11} p_{33} + a_{21} b_3 p_{11} p_{33} - 2 = 0$ |
| C4     | $b_1 g_{31} p_{11} - 1 = 0$               |
| C5     | $12a_{11} b_2 g_{11} p_{11} p_{21} + 6a_{11} b_2 p_{11} p_{21} - 24a_{11} b_3 p_{11} p_{31} + 12a_{11} b_3 p_{11} p_{31} \ldots$  
|        | $- 12a_{11} b_3 p_{11} p_{32} + 6a_{21} b_3 p_{11} p_{32} - 4a_{11} b_3 p_{11} p_{33} + 2a_{21} b_3 p_{11} p_{33} - 3 = 0$ |
| C6     | $12a_{11} b_2 g_{11} p_{11} p_{21} + 6a_{11} b_2 g_{11} p_{11} p_{22} - 24a_{11} b_3 g_{11} p_{11} p_{31} + 12a_{11} b_3 g_{11} p_{11} p_{31} \ldots$  
|        | $- 12a_{11} b_3 g_{11} p_{11} p_{32} + 6a_{21} b_3 g_{11} p_{11} p_{32} - 4a_{11} b_3 g_{11} p_{11} p_{33} + 2a_{21} b_3 g_{11} p_{11} p_{33} - 3 = 0$ |
| C7     | $12a_{11} b_2 g_{32} p_{11} p_{21} + 4a_{11} b_2 g_{32} p_{11} p_{22} - 24a_{11} b_3 g_{33} p_{11} p_{31} + 12a_{11} b_3 g_{33} p_{11} p_{31} \ldots$  
|        | $- 8a_{11} b_3 g_{33} p_{11} p_{32} + 4a_{21} b_3 g_{33} p_{11} p_{32} - 2a_{11} b_3 g_{33} p_{11} p_{33} + a_{21} b_3 g_{33} p_{11} p_{33} - 2 = 0$ |
| C8     | $b_1 g_{31} p_{11} - 1 = 0$               |
| C9     | $30a_{11} b_2 p_{11} p_{21} + 15a_{11} b_2 p_{11} p_{22} - 60a_{11} b_3 p_{11} p_{31} + 30a_{21} b_3 p_{11} p_{31} \ldots$  
|        | $- 30a_{11} b_3 p_{11} p_{32} + 15a_{21} b_3 p_{11} p_{32} - 10a_{11} b_3 p_{11} p_{33} + 5a_{21} b_3 p_{11} p_{33} - 6 = 0$ |
| C10    | $30a_{11} b_2 g_{11} p_{11} p_{21} + 15a_{11} b_2 g_{11} p_{11} p_{22} - 60a_{11} b_3 g_{11} p_{11} p_{31} + 30a_{21} b_3 g_{11} p_{11} p_{31} \ldots$  
|        | $- 30a_{11} b_3 g_{11} p_{11} p_{32} + 15a_{21} b_3 g_{11} p_{11} p_{32} - 10a_{11} b_3 g_{11} p_{11} p_{33} + 5a_{21} b_3 g_{11} p_{11} p_{33} - 6 = 0$ |
| C11    | $60a_{11} b_2 a_{21} a_{22} b_3 p_{11} p_{21} p_{31} + 30a_{11} a_{21} a_{22} b_3 p_{11} p_{22} p_{31} + 30a_{11} a_{21} a_{22} b_3 p_{11} p_{22} p_{32}$  
|        | $+ 15a_{11} a_{21} a_{22} b_3 p_{11} p_{22} p_{32} + 10a_{11} a_{21} a_{22} b_3 p_{11} p_{22} p_{33} + 5a_{11} a_{21} a_{22} b_3 p_{11} p_{22} p_{33} - 4 = 0$ |
| C12    | $30a_{11} b_2 g_{11} p_{11} p_{21} + 15a_{11} b_2 g_{11} p_{11} p_{22} - 60a_{11} b_3 g_{11} p_{11} p_{31} + 30a_{21} b_3 g_{11} p_{11} p_{31} \ldots$  
|        | $- 30a_{11} b_3 g_{11} p_{11} p_{32} + 15a_{21} b_3 g_{11} p_{11} p_{32} - 10a_{11} b_3 g_{11} p_{11} p_{33} + 5a_{21} b_3 g_{11} p_{11} p_{33} - 6 = 0$ |
| C13    | $30a_{11} b_2 g_{11} p_{11} p_{21} + 15a_{11} b_2 g_{11} p_{11} p_{22} - 60a_{11} b_3 g_{11} p_{11} p_{31} + 30a_{21} b_3 g_{11} p_{11} p_{31} \ldots$  
|        | $- 30a_{11} b_3 g_{11} p_{11} p_{32} + 15a_{21} b_3 g_{11} p_{11} p_{32} - 10a_{11} b_3 g_{11} p_{11} p_{33} + 5a_{21} b_3 g_{11} p_{11} p_{33} - 6 = 0$ |
| C14    | $60a_{11} b_2 g_{32} p_{11} p_{21} + 20a_{11} b_2 g_{32} p_{11} p_{22} - 120a_{11} b_3 g_{33} p_{11} p_{31} + 60a_{21} b_3 g_{33} p_{11} p_{31} \ldots$  
|        | $- 40a_{11} b_3 g_{33} p_{11} p_{32} + 20a_{21} b_3 g_{33} p_{11} p_{32} - 10a_{11} b_3 g_{33} p_{11} p_{33} + 5a_{21} b_3 g_{33} p_{11} p_{33} - 6 = 0$ |
| C15    | $60a_{11} b_2 g_{11} g_{32} p_{11} p_{21} + 20a_{11} b_2 g_{11} g_{32} p_{11} p_{22} - 120a_{11} b_3 g_{11} g_{33} p_{11} p_{31}$  
|        | $+ 60a_{21} b_3 g_{11} g_{33} p_{11} p_{32} + 20a_{21} b_3 g_{11} g_{33} p_{11} p_{32} - 10a_{11} b_3 g_{11} g_{33} p_{11} p_{33} + 5a_{21} b_3 g_{11} g_{33} p_{11} p_{33} - 6 = 0$ |
| C16    | $20a_{11} b_2 g_{32} p_{11} p_{21} + 5a_{11} b_2 g_{32} p_{11} p_{22} - 40a_{11} b_3 g_{33} p_{11} p_{31} + 20a_{21} b_3 g_{33} p_{11} p_{31} \ldots$  
|        | $- 10a_{11} b_3 g_{33} p_{11} p_{32} + 5a_{21} b_3 g_{33} p_{11} p_{32} - 2a_{11} b_3 g_{33} p_{11} p_{33} + a_{21} b_3 g_{33} p_{11} p_{33} - 2 = 0$ |
| C17    | $b_1 g_{31} p_{11} - 1 = 0$               |
Further research will be required to compare these schemes and determine additional constraints on the method coefficients which guarantee favorable properties such as fulfillment of stiff order conditions, small error constant or other features. Here we construct specific schemes which satisfy one more important constraint, namely the requirement that all coefficients \( g_{ij} \) are positive. Since these parameters enter the method as coefficients of the Jacobian matrix within functions \( \psi_j(g_{ij}hA_0) \), positivity of \( g_{ij} \)'s is important to be able to accurately estimate these functions for stable matrices \( A_0 \).

Not all values for \( p_{ij} \)'s, however, guarantee the existence of three-stage fifth order EPIRK methods with all positive \( g_{ij} \)'s. For example, if we choose \( p_{11} = 1, p_{21} = 0, p_{22} = 4, p_{31} = 0, p_{32} = -2, p_{33} = 16 \) (these values correspond to choosing \( \psi_j(z) \) as in (7) with \( s = 4 \)) the twelfth order polynomial for \( a_{11} \) has only two real roots that can be numerically computed and are approximately equal to \(-0.26 \) and \(0.77 \). The two corresponding fifth order methods both have negative \( g_{ij} \) coefficients. For \( a_{11} \approx -0.26 \) the negative coefficient is \( g_{11} = a_{11} \), and for \( a_{11} \approx 0.77 \) the negative coefficient is \( g_{21} \approx -1.15 \). Similarly if \( \psi_j(z) \) are chosen as in (7) with \( s = 3 \), the corresponding values for \( p_{ij} \) produce four real roots for the polynomial in \( a_{11} \) but none of these roots result in a method with all positive \( g_{ij} \) coefficients.

To construct a three-stage fifth order method with positive \( p_{ij} \) we consider the expressions obtained for these coefficients with respect to \( a_{11} \) and \( p_{ij} \) (note \( g_{31} = 1 \) and \( g_{22} \) is arbitrary according to the order conditions):

\[
\begin{align*}
g_{11} &= a_{11} p_{11}, \
g_{21} &= \frac{3(-4 + 5a_{11} p_{11})}{5(-3 + 4a_{11} p_{11})}, \\
g_{32} &= \frac{(432 - 1512 a_{11} p_{11} + 5805 a_{11}^2 p_{11}^2 - 23625 a_{11}^3 p_{11}^3 + 48600 a_{11}^4 p_{11}^4 - 45600 a_{11}^5 p_{11}^5 + 16000 a_{11}^6 p_{11}^6) \times (2p_{21} + p_{22})}{(432 - 1080 a_{11} p_{11} + 20925 a_{11}^2 p_{11}^2 - 108000 a_{11}^3 p_{11}^3 + 216000 a_{11}^4 p_{11}^4 - 192000 a_{11}^5 p_{11}^5 + 64000 a_{11}^6 p_{11}^6) \times (3p_{21} + 2p_{22})}, \\
g_{33} &= \frac{4(-3 + 5a_{11} p_{11}) (6p_{31} + 3p_{32} + p_{33})}{5(-3 + 4a_{11} p_{11}) (12p_{31} + 4p_{32} + p_{33})}. 
\end{align*}
\]

From these formulas we conclude that if \( p_{21}, p_{22}, p_{31}, p_{32}, p_{33} \) are positive then to ensure the positivity of \( g_{ij} \)'s we need to choose a positive \( x = a_{11} p_{11} \) so that the following functions are also positive:

\[
\begin{align*}
w_1(x) &= \frac{3(-4 + 5x)}{5(-3 + 4x)}, \\
w_2(x) &= \frac{(432 - 1512 x + 5805 x^2 - 23625 x^3 + 48600 x^4 - 45600 x^5 + 16000 x^6)}{(432 - 1080 x + 20925 x^2 - 108000 x^3 + 216000 x^4 - 192000 x^5 + 64000 x^6)}, \\
w_3(x) &= \frac{4(-3 + 5x)}{5(-3 + 4x)}. 
\end{align*}
\]
Figure 2: Graphs of functions $w_0(x) = x = a_{11}p_{11}$ and $w_i(x)$ ($i = 1, 2, 3$) from (31) showing the regions of positivity of $g_{ij}$.

As can be seen from Figure 2, where graphs of functions $w_i(x)$ are shown, to ensure positivity of $w_i(x)$ we need to choose $x = a_{11}p_{11}$ from the intervals $0 < x < 3/5$ and $x > 4/5$. For example, a choice of $p_{11} = 1$, $p_{21} = 2/3$, $p_{22} = 2/3$, $p_{31} = 2/3$, $p_{32} = 1/2$, $p_{33} = 1$ produces a fifth-order three-stage EPIRK5s2 method with coefficients listed in Table 4.

Four-stage fifth-order EPIRK methods: The four-stage methods have more degrees of freedom with respect to the method coefficients. For example, we are able to derive fourth- and fifth-order embedded methods by assigning values to a number of coefficients first and solving for the rest using the order conditions. If we follow the choice of $\psi_j(z)$ functions given by (7) with $s = 4$ and additionally prescribe the following values for the coefficients $a_{11} = 1/4$, $a_{21} = 2/4$, $a_{31} = 3/4$, as was done in [16], the order conditions yield two different methods EPIRK5s4A and EPIRK5s4B. The coefficients of these methods are given in Table 4 (Note: methods (A) and (B) only differ by coefficients $g_{4i}$ ($i = 1, \ldots, 4$) and are listed together; a coefficient has an Arbitrary tag if the order conditions are independent of it).

5. Numerical experiments

In order to numerically verify the order of the EPIRK methods derived in the previous section we choose the following four test problems. To strengthen numerical evidence we also compare the new schemes to previously derived and tested exponential integrators, the fourth-order EPIRK4 [16, 23], the fourth order EXP4 method [15] and the fourth-order exponential Rosenbrock EXPROS4 [22]. The first test problem is a nonlinear oscillator system [1]:

$$
\begin{bmatrix}
y_1' \\
y_2'
\end{bmatrix} = \begin{bmatrix}
y_2 \\
-y_1^2y_2 - y_1
\end{bmatrix}, \quad \begin{bmatrix}
y_1(0) \\
y_2(0)
\end{bmatrix} = \begin{bmatrix}1 \\
1
\end{bmatrix}.
$$

(32)
Table 4: Coefficients of EPIRK methods.

**EPIRK5s3**

\[
p = \begin{bmatrix}
1 \\
4s_1^2 + 2s_2^2 \\
s_2^2 \\
1
\end{bmatrix}
\]

\[
\begin{bmatrix}
a_{11} \\
a_{21} & a_{22} \\
b_1 & b_2 & b_3
\end{bmatrix} = \begin{bmatrix}
0.41657015580651858694 & 1.32931146991722972036 \\
0.86246743701274574979 & 0.30931492086655796815
\end{bmatrix}
\]

\[
\begin{bmatrix}
g_{11} \\
g_{21} & g_{22} \\
g_{31} & g_{32} & g_{33}
\end{bmatrix} = \begin{bmatrix}
0.41657015580651858694 & 0.5 \\
0.86246743701274574979 & 0.730416157608327661916 \\
1.0 & 0.32507696706078277327 & 0.3315
\end{bmatrix}
\]

**EPIRK5s4A(B)**

\[
p = \begin{bmatrix}
1 \\
0 \\
4 \\
0 \\
-2 \\
16
\end{bmatrix}
\]

\[
\begin{bmatrix}
a_{11} \\
a_{21} & a_{22} \\
a_{31} & a_{32} & a_{33} \\
b_1 & b_2 & b_3 & b_4
\end{bmatrix} = \begin{bmatrix}
1/4 \\
2/4 & -1/108 \\
3/4 & 1/2 & 1/4 \\
1 & 52/45 & 68/75 & 52/45
\end{bmatrix}
\]

\[
\begin{bmatrix}
g_{11} \\
g_{21} & g_{22} \\
g_{31} & g_{32} & g_{33} \\
g_{41} & g_{42} & g_{43} & g_{44}
\end{bmatrix} = \begin{bmatrix}
1/4 \\
1/2 & 1/2(Arbitrary) \\
3/4 & 1/2(Arbitrary) & 1/2(Arbitrary) \\
1 & (195 \pm \sqrt{3315})/260 & (255 \pm \sqrt{3315})/204 & (60 \pm \sqrt{3315})/104
\end{bmatrix}
\]

**EPIRK4**

\[
p = \begin{bmatrix}
1 \\
0 \\
1 \\
0 \\
-1 \\
6
\end{bmatrix}
\]

\[
\begin{bmatrix}
a_{11} \\
a_{21} & a_{22} \\
b_1 & b_2 & b_3
\end{bmatrix} = \begin{bmatrix}
27(s^2+18) & 0 & 0 \\
12(54-3s^2+2s^3) & 0 & 0 \\
18(s^2+18) & 0 & 0
\end{bmatrix}, \quad s = \pm \sqrt{30}
\]

\[
\begin{bmatrix}
g_{11} \\
g_{21} & g_{22} \\
g_{31} & g_{32} & g_{33}
\end{bmatrix} = \begin{bmatrix}
1/3 \\
2/3 & 2/3 \\
1 & 1 & 1
\end{bmatrix}
\]
Since the Jacobian matrix is only 2 by 2 in this case, we can use Padé approximation to compute the products of \( \psi_j(\gamma A_0) v \).

The rest of the test problems are implemented with the Krylov projections-based evaluation of the products \( \psi_j(\gamma A_0) v \). The tolerance for the Krylov iterations is set to \( 10^{-13} \) to avoid the accuracy of the evaluation impact the overall error. The second and third test problems are the Burgers equation

\[
    u_t = -u u_x + \nu u_{xx}, \quad x \in [0, 1], \quad t \in [0, 1]
\]

with two different values of \( \nu \) and the initial and Dirichlet boundary conditions prescribed using \( u = (\sin(2\pi x))^2(1-x)^{3/2} \). The spatial operators are discretized using centered second-order finite differences with \( N = 500 \) nodes and the discretization for the advective term is conservative. We perform computations with two values of the diffusion constant \( \nu = 0.0003 \) and 0.03. The second value produces a more stiff problem. We want to see how the methods accommodate this change since none of the new integrators have been constructed to satisfy the stiff order conditions except EXPROS4.

Our last test problem is the semilinear parabolic equation from [20]

\[
    \frac{\partial U}{\partial t}(x, t) - \frac{\partial^2 U}{\partial^2 x}(x, t) = \int_0^1 U(s, t) ds + \Phi(x, t), \quad x \in [0, 1], \quad t \in [0, 1],
\]

subject to the homogeneous Dirichlet boundary conditions and with the source function \( \Phi(x, t) \) chosen so that the exact solution of the problem is \( U(x, t) = x(1 - x)e^t \). The problem is discretized in space on \( N = 200 \) nodes, using second-order finite differences and the trapezoidal rule for the approximation of the integral. This problem was used in [20] to demonstrate that the exponential integrators, that do not satisfy the stiff order conditions derived in that paper, experience order reduction phenomenon. Since none of the EPIRK methods were derived using stiff order conditions, it is interesting to test whether the order-reduction phenomenon is observed.

Figure 3 shows the log − log plots of the error vs. the step size for all five integrators (EXP4, EPIRK4, EXPROS4, EPIRK5s3, EPIRK5s4). As we can see all of the methods display the full predicted order for the two less stiff problems, the nonlinear oscillator and the Burgers equation with \( \nu = 0.0003 \). For the semilinear parabolic problem (34) all of the methods experience order reduction except EXPROS4 and EPIRK4. The result for EXPROS4 is expected since the problem was designed specifically to illustrate the stiff order of this method [20], but is unexpected for EPIRK4 since this scheme was designed only with classical order. For the Burgers equation with \( \nu = 0.03 \), all of the methods, however, suffer from order reduction.
The degree of the order reduction varies for different integrators with EPIRK5s3 method experiencing the smallest order reduction (approximate orders are 3.7 for EXP4, 3.7 for EPIRK4, 3.5 for EXPROS4, 4.8 for EPIRK5s3, and EPIRK5s4’s order reduces to 3.8). Surely, these results cannot lead to a definitive conclusion that EPIRK methods should or should not experience order reduction for stiff problems. However, they do indicate that stiffly accurate EPIRK methods can be constructed and call for further research into the derivation of the stiff order conditions for these schemes.

![Graphs of error vs time step size for different integrators](image)

Figure 3: The plot of the error on the log-log scale of the five exponential integrators versus the time step size. For convenience the lines with slopes four (dash-dotted) and five (dashed) are shown.
A detailed study of the performance of the EPIRK methods requires carefully fine-tuning implementations for both serial and parallel computers, development of adaptive strategies to optimize the efficiency of the Krylov projections and automatic time-stepping, investigating the sensitivity of the time integrators to different spatial discretizations, and addressing other practical issues. Each of these important questions requires detailed investigations that are beyond the scope of this paper. Here, however, we present two examples that illustrate the performance potential of the methods. The results presented below verify the reduction of the total number of the Krylov vectors for EPIRK methods and demonstrate the resulting computational savings.

The first example is the two-dimensional Gray-Scott system [36]:

\[
\begin{align*}
  u_t &= d_u \nabla^2 u - uv^2 + a(1-u), \quad x, y \in [0, 1], \quad t \in [0, 0.1], \\
  v_t &= d_v \nabla^2 v + uv^2 - (a + b)v,
\end{align*}
\]

with \( d_u = 0.2, \ d_v = 0.1, \ a = 0.04, \) and \( b = 0.06. \) Periodic boundary conditions are used and the initial conditions are prescribed with

\[
\begin{align*}
  u(x, y, 0) &= 1 - e^{-150(x-\frac{1}{2})^2+(y-\frac{1}{2})^2}, \\
  v(x, y, 0) &= e^{-150(x-\frac{1}{2})^2+2(y-\frac{1}{2})^2}.
\end{align*}
\]

The spatial discretization is done using second-order finite differences.

The second example is the nonlinear reaction-diffusion (NRD) equation, which is used in models of biological phenomena [37]:

\[
\begin{align*}
  u_t &= uu_{xx} + u(u-1), \quad x \in [-23, 50], t \in [0, 50], \\
  u(x, 0) &= \begin{cases} 
    e^{-\mu x}, & x > 0 \\
    1, & x < 0
  \end{cases}
\end{align*}
\]

with homogeneous Neumann boundary conditions. The problem is discretized using second-order finite differences in space.

The discretized in space equations were solved using several exponential integrators - EPIRK4, EXP4, EROS4 and EPIRK5s3 - and a fourth-order implicit backward-differentiation formula based integrator BDF4 (e.g. see [38]). The four stages EPIRK5s4 method has not been included since it involves four stages and will not be computationally more efficient then the three stages methods. Both, computations of the \( \varphi(\gamma A)v \) products for exponential method and approximations of \( (I - \gamma A)^{-1}v \) for each of the Newton iterations within the BDF4 method have been carried out until the norms of the residuals (e.g. see [15, 39, 38] for residual
calculations algorithms) were reduced to a specified tolerance (in the examples the
tolerance is set to $10^{-12}$). Clearly this is not the most efficient method to perform
such calculations and as a result the Krylov projection can be overcomputed. How-
ever, such choice ensures fair comparison. More efficient adaptive techniques exist
for Newton-Krylov implicit methods and similar adaptive algorithms are currently
being developed for the exponential methods. Our current studies include the develop-
ment, thorough comparison and evaluation of such adaptive techniques, the results
of which will be reported in a separate publication.

Figure (4) and Tables (5 - 6) summarize the results of the numerical experiments.
Fig. (4) shows the precision diagrams for Gray-Scott problem with $N = 150^2$ and
for NRD problem with $N = 1500$. As we can see for both of these examples the
EPIRK5s3 is the best performing method since it obtains the most accurate solution
while requiring the least amount of computational time. The NRD problem is ac-
tually less stiff than the Gray-Scott system and therefore the computational savings
resulting from the reduction in the number of Krylov vectors are larger for Gray-
Scott. Changing the sizes of the problems ($N = 300^2$ for Gray-Scott and $N = 500$
for NRD) resulted in very similar graphs and tables; the data is not presented here
simply because it repeats the results without adding substance to the narration. It
is important to note that for stiff problems both Newton-Krylov and exponential-
Krylov methods exhibit the bend in the precision diagrams curves, i.e. the larger
step sizes (e.g. $h = 0.01$ for Gray-Scott problem) actually yield a solution which is
less accurate but requires more computational time than some lower values of $h$. In
fact, as can be seen from the precision diagrams in Fig. (4) there exists an "opti-
mal" value of $h$ (i.e. the leftmost tip of the precision curves). An effective adaptive
algorithm should be able to find this "optimal" value and evolve the system with
this time-step if the corresponding approximation to the solution meets prescribed
accuracy.

The data presented in tables (5 - 6) clarifies the reason for the relative cost of
each of the integrators. For each of the integrators and each time step value, the
tables include the average number of Krylov vectors required for each of the pro-
jections. Note that unlike the fixed number of Krylov projections required by the
exponential methods, the number of Newton iterations within the BDF4 method
varies. However, as can be seen from the tables, even when BDF4 method requires
only two projections compared to three projections for exponential methods (see Ta-
ble 5), the BDF4 projections require significantly more Krylov vectors and therefore
are more computationally expensive resulting in the overall slower performance of
the integrator.
6. Conclusions and future work

In this paper we have introduced a new class of exponential methods, the EPIRK schemes. We have shown that by assuming a more general ansatz for an exponential integrator than previously proposed, we can derive schemes with favorable properties such as high order methods with low computational cost. The B-series theory was used to derive order conditions for methods up to order five. We developed algorithms to reduce the order conditions to a simpler form and solve them to construct specific fifth-order schemes with three and four stages. The order of these new methods has been confirmed using numerical experiments and their performance illustrated using comparisons with other integrators.

There are many interesting questions arising with respect to the new class of integrators. It is important to investigate whether particular schemes which satisfy stiff, in addition to classical, order conditions can be derived. While all exponential integrators are trivially A-stable, additional studies are needed to understand nonlinear stability properties of these integrators. It is also desirable to find particular schemes with the smallest error constants. Finally, the issues of implementation of such schemes for very large scale applications and their comparison with current state-of-the-art integrators must be addressed. An important issue for practical use of the new methods is development of adaptive schemes that couple Krylov space projections and the overall time-stepping algorithm. Two adaptivity strategies have been proposed previously [15, 30]. We are currently investigating whether these
Table 5: Average number of Krylov vectors per projection and the total CPU time for Gray-Scott equation (35) with $N = 150^2$.

<table>
<thead>
<tr>
<th></th>
<th>Average # of Krylov vectors</th>
<th># Projs. per step</th>
<th>Total vectors</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Proj. 1</td>
<td>Proj. 2</td>
<td>Proj. 3</td>
<td>Total</td>
</tr>
<tr>
<td>$h = 0.01$:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exp4</td>
<td>60.7</td>
<td>50.8</td>
<td>52.9</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK4</td>
<td>59.9</td>
<td>46.0</td>
<td>41.6</td>
<td>3</td>
</tr>
<tr>
<td>ERow4</td>
<td>58.9</td>
<td>50.7</td>
<td>39.2</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK5s3</td>
<td>59.4</td>
<td>40.4</td>
<td>22.5</td>
<td>3</td>
</tr>
<tr>
<td>BDF4</td>
<td>126.1</td>
<td>81.0</td>
<td>n/a</td>
<td>2.00</td>
</tr>
<tr>
<td>$h = 0.005$:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exp4</td>
<td>36.9</td>
<td>29.2</td>
<td>30.8</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK4</td>
<td>36.7</td>
<td>26.1</td>
<td>19.1</td>
<td>3</td>
</tr>
<tr>
<td>ERow4</td>
<td>36.3</td>
<td>29.2</td>
<td>18.9</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK5s3</td>
<td>36.9</td>
<td>22.9</td>
<td>12.8</td>
<td>3</td>
</tr>
<tr>
<td>BDF4</td>
<td>74.1</td>
<td>42.2</td>
<td>n/a</td>
<td>2.00</td>
</tr>
<tr>
<td>$h = 0.0025$:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exp4</td>
<td>23.2</td>
<td>16.2</td>
<td>17.4</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK4</td>
<td>23.1</td>
<td>14.5</td>
<td>9.9</td>
<td>3</td>
</tr>
<tr>
<td>ERow4</td>
<td>22.9</td>
<td>16.2</td>
<td>8.2</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK5s3</td>
<td>23.4</td>
<td>12.6</td>
<td>7.3</td>
<td>3</td>
</tr>
<tr>
<td>BDF4</td>
<td>39.0</td>
<td>21.4</td>
<td>n/a</td>
<td>2.00</td>
</tr>
<tr>
<td>$h = 0.00125$:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exp4</td>
<td>14.8</td>
<td>8.9</td>
<td>9.6</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK4</td>
<td>14.9</td>
<td>7.6</td>
<td>5.5</td>
<td>3</td>
</tr>
<tr>
<td>ERow4</td>
<td>14.8</td>
<td>8.9</td>
<td>3.9</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK5s3</td>
<td>15.0</td>
<td>6.8</td>
<td>3.9</td>
<td>3</td>
</tr>
<tr>
<td>BDF4</td>
<td>20.2</td>
<td>11.1</td>
<td>n/a</td>
<td>2.00</td>
</tr>
<tr>
<td>$h = 0.000625$:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exp4</td>
<td>9.8</td>
<td>4.4</td>
<td>5.0</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK4</td>
<td>9.9</td>
<td>3.9</td>
<td>3.3</td>
<td>3</td>
</tr>
<tr>
<td>ERow4</td>
<td>9.8</td>
<td>4.4</td>
<td>2.5</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK5s3</td>
<td>9.9</td>
<td>3.7</td>
<td>2.1</td>
<td>3</td>
</tr>
<tr>
<td>BDF4</td>
<td>11.9</td>
<td>5.9</td>
<td>n/a</td>
<td>2.00</td>
</tr>
</tbody>
</table>

Approaches are effective for the EPIRK methods and developing new more efficient adaptive algorithms. Additionally, it is important to understand how performance of the EPIRK methods depends on the spatial discretization methods. Studies comparing finite-difference discretizations and discontinuous Galerkin methods are underway. We plan to implement the exponential schemes for serial and parallel computers and test their performance on large scale scientific computing problems.
Table 6: Average number of Krylov vectors per projection and the total CPU time for integrating nonlinear reaction-diffusion (NRD) equation with $N = 1500$.

<table>
<thead>
<tr>
<th></th>
<th>Average # of Krylov vectors per step</th>
<th># Projs. per step</th>
<th>Total vectors</th>
<th>CPU time total</th>
<th>% Exp4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Proj. 1</td>
<td>Proj. 2</td>
<td>Proj. 3</td>
<td>Proj. 4</td>
<td></td>
</tr>
<tr>
<td>$h = 5e-6$:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exp4</td>
<td>46.9</td>
<td>43.6</td>
<td>44.7</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK4</td>
<td>48.2</td>
<td>41.4</td>
<td>39.2</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>ERow4</td>
<td>46.8</td>
<td>43.7</td>
<td>36.0</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK5s3</td>
<td>48.8</td>
<td>37.1</td>
<td>22.2</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>BDF4</td>
<td>98.6</td>
<td>88.3</td>
<td>64.0</td>
<td>23.8</td>
<td>3.97</td>
</tr>
<tr>
<td>$h = 2.5e-6$:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exp4</td>
<td>29.3</td>
<td>26.2</td>
<td>27.1</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK4</td>
<td>29.5</td>
<td>24.2</td>
<td>22.3</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>ERow4</td>
<td>29.3</td>
<td>26.3</td>
<td>20.5</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK5s3</td>
<td>29.3</td>
<td>20.9</td>
<td>12.1</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>BDF4</td>
<td>59.1</td>
<td>51.3</td>
<td>29.5</td>
<td>1.0</td>
<td>3.13</td>
</tr>
<tr>
<td>$h = 1.25e-6$:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exp4</td>
<td>18.6</td>
<td>15.7</td>
<td>16.5</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK4</td>
<td>18.6</td>
<td>14.0</td>
<td>12.4</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>ERow4</td>
<td>18.4</td>
<td>15.6</td>
<td>11.6</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK5s3</td>
<td>18.1</td>
<td>12.6</td>
<td>7.8</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>BDF4</td>
<td>31.8</td>
<td>26.4</td>
<td>10.8</td>
<td>1.0</td>
<td>3.00</td>
</tr>
<tr>
<td>$h = 6.25e-7$:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exp4</td>
<td>11.9</td>
<td>9.3</td>
<td>9.8</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK4</td>
<td>12.0</td>
<td>8.2</td>
<td>5.8</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>ERow4</td>
<td>11.8</td>
<td>9.2</td>
<td>6.1</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK5s3</td>
<td>12.0</td>
<td>7.6</td>
<td>4.7</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>BDF4</td>
<td>17.5</td>
<td>13.8</td>
<td>3.3</td>
<td>1.0</td>
<td>2.9</td>
</tr>
<tr>
<td>$h = 3.125e-7$:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exp4</td>
<td>8.1</td>
<td>5.7</td>
<td>6.9</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK4</td>
<td>8.1</td>
<td>4.9</td>
<td>3.8</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>ERow4</td>
<td>8.0</td>
<td>5.7</td>
<td>3.0</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>EpiRK5s3</td>
<td>8.3</td>
<td>4.0</td>
<td>3.0</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>BDF4</td>
<td>10.6</td>
<td>7.5</td>
<td>1.2</td>
<td>n/a</td>
<td>2.01</td>
</tr>
</tbody>
</table>

7. Acknowledgements

This work was supported in part by a grant from the U.S. Department of Energy, Office of Science, Offices of Advanced Scientific Computing Research, and Biological & Environmental Research through the U.C. Merced Center for Computational Biology #DE-FG02-04ER25625. We would like to thank John Loffeld for assistance with performing the calculations. In addition, we would like to extend deepest gratitude
to John Butcher for support, inspiration and helpful comments on the manuscript.

References


