J Sci Comput
https://doi.org/10.1007/s10915-018-0761-3

EPIRK-W and EPIRK-K Time Discretization Methods

Mahesh Narayanamurthi[1] · **Paul Tranquilli**[1] · **Adrian Sandu**[1] · **Mayya Tokman**[2]

Received: 20 February 2017 / Revised: 17 December 2017 / Accepted: 2 June 2018

Abstract Exponential integrators are special time discretization methods where the traditional linear system solves used by implicit schemes are replaced with computing the action of matrix exponential-like functions on a vector. A very general formulation of exponential integrators is offered by the Exponential Propagation Iterative methods of Runge–Kutta type (EPIRK) family of schemes. The use of Jacobian approximations is an important strategy to drastically reduce the overall computational costs of implicit schemes while maintaining the quality of their solutions. This paper extends the EPIRK class to allow the use of inexact Jacobians as arguments of the matrix exponential-like functions. Specifically, we develop two new families of methods: EPIRK-W integrators that can accommodate any approximation of the Jacobian, and EPIRK-K integrators that rely on a specific Krylov-subspace projection of the exact Jacobian. Classical order conditions theories are constructed for these families. Practical EPIRK-W methods of order three and EPIRK-K methods of order four are developed. Numerical experiments indicate that the methods proposed herein are computationally favorable when compared to a representative state-of-the-art exponential integrator, and a Rosenbrock–Krylov integrator.

Keywords Time integration · Exponential integrator · Krylov · B-series · Butcher trees

✉ Mahesh Narayanamurthi
maheshnm@vt.edu

Paul Tranquilli
ptranq@vt.edu

Adrian Sandu
sandu@cs.vt.edu

Mayya Tokman
mtokman@ucmerced.edu

[1] Computational Science Laboratory, Department of Computer Science, Virginia Tech, Blacksburg, VA 24060, USA

[2] School of Natural Sciences, University of California, Merced, CA 95343, USA

Published online: 14 June 2018

Mathematics Subject Classification 65L05 · 65L04 · 65F60 · 65M22

1 Introduction

The following initial value problem for a system of ordinary differential equations (ODEs)

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y}_0, \quad t_0 \le t \le t_F, \quad \mathbf{y}(t) \in \mathbb{R}^N \tag{1}$$

arises in many applications including numerical solution of partial differential equations (PDEs) using a method of lines. Different time discretization methods can then be applied to solve (1) and approximate numerical solutions $\mathbf{y}_n \approx \mathbf{y}(t_n)$ at discrete times t_n.

Runge–Kutta methods [9] are the prototypical one-step discretizations that use internal stage approximations to propagate the numerical solution forward in time from t_n to $t_{n+1} = t_n + h$. Explicit Runge–Kutta methods [9, Section II.2] perform inexpensive calculations at each step, but suffer from stability restrictions on the time step size which makes them inappropriate for solving stiff systems. Implicit Runge–Kutta methods [9, Section II.7] require solution of non-linear systems of equations at each step that can be done, for instance, by using a Newton-like approach. This eases the numerical stability restrictions and allows to use large time step sizes for stiff problems, however it also increases the computational cost per step.

Rosenbrock methods [10, Section IV.7] arise from the linearization of diagonally implicit Runge–Kutta methods, and solve only linear systems of equations at each time step. In Rosenbrock methods, the Jacobian of the ODE function (1)

$$\mathbf{J}(t, \mathbf{y}) = \frac{\partial \mathbf{f}(t, \mathbf{y})}{\partial \mathbf{y}} \tag{2}$$

appears explicitly in the formulation. Consequentially, the order conditions theory of these methods relies on using an exact Jacobian during computations. For the solution of large linear systems any popular implementation of the Rosenbrock method will use Krylov projection-based iterative methods like `GMRES` [23,35]. Early truncation of the iterative method is equivalent to the use of an approximation of the Jacobian, and the overall scheme may suffer from order reduction [30]. Rosenbrock-W (ROW) methods [10,34] mitigate this by admitting arbitrary approximations of the Jacobian ($\mathbf{A} \approx \mathbf{J}$). This is possible at the cost of a vastly increased number of order conditions, and by using many more stages to allow for enough degrees of freedom to satisfy them. Rosenbrock-K (ROK) methods [32] are built in the ROW framework by making use of a specific Krylov-based approximation of the Jacobian. This approximation leads to a number of trees that arise in ROW order conditions theory becoming isomorphic to one another, leading to a dramatic reduction in the total number of order conditions.

Exponential integrators [1,3,4,7,12–16,18,19,24,27,28,33] are a class of timestepping methods that replace the linear system solves in Rosenbrock style methods with operations involving the action of matrix exponential-like functions on a vector. For many problems these products of a matrix exponential-like function with a vector can be evaluated more efficiently than solving the corresponding linear systems.

Both the W-versions [13] and K-versions [31] of exponential methods have been established for particular formulations of exponential methods. This paper extends the theory of W- and K-type methods to the general class of Exponential Propagation Iterative methods of Runge–Kutta type (EPIRK) [27,28].

The remainder of the paper is laid out as follows. In Sect. 2, we describe the general form of the EPIRK method and some simplifying assumptions that we make in this paper. In Sect. 3, we derive the order conditions for an EPIRK-W method and construct methods with two different sets of coefficients. In Sect. 4, we extend the K-theory, introduced in [32] for Rosenbrock–Krylov methods, to EPIRK methods and construct an EPIRK-K method. Section 5 addresses strategies to evaluate products involving exponential-like functions of matrices and vectors that are present in the formulation of methods discussed in this paper. Section 6 presents the numerical results, and conclusions are drawn in Sect. 7.

2 Exponential Propagation Iterative Methods of Runge–Kutta Type

A very general formulation of exponential integrators is offered by the EPIRK family of methods, which was first introduced in [27]. As is typical for exponential methods, the formulation of EPIRK is derived from the following integral form of the problem (1) on the time interval $[t_n, t_n + h]$:

$$\mathbf{y}(t_n + h) = \mathbf{y}_n + h\,\boldsymbol{\varphi}_1\big(h\,\mathbf{J}_n\big)\,\mathbf{f}(\mathbf{y}_n) + h\int_0^1 e^{h\,\mathbf{J}_n\,(1-\theta)}\,\mathbf{r}\big(\mathbf{y}(t_n + \theta h)\big)\,d\theta. \tag{3}$$

Equation (3) is derived by splitting the right hand-side function of (1) into a linear term and non-linear remainder using first-order Taylor expansion

$$\mathbf{y}' = \mathbf{f}(\mathbf{y}_n) + \mathbf{J}_n\,(\mathbf{y} - \mathbf{y}_n) + \mathbf{r}(\mathbf{y}), \quad \mathbf{r}(\mathbf{y}) := \mathbf{f}(\mathbf{y}) - \mathbf{f}(\mathbf{y}_n) - \mathbf{J}_n\,(\mathbf{y} - \mathbf{y}_n), \tag{4}$$

and using the integrating factor method to transform this ODE into an integral equation [28]. In Eq. (3) the Jacobian matrix (2) evaluated at time t_n is $\mathbf{J}_n = \mathbf{J}(t_n, \mathbf{y}_n)$, and $\mathbf{r}(\mathbf{y})$ is the non-linear remainder term of the Taylor expansion of the right-hand side function. The function $\varphi_1(z) = (e^z - 1)/z$ is the first one in the sequence of analytic functions defined by

$$\varphi_k(z) = \int_0^1 e^{z\,(1-\theta)}\,\frac{\theta^{k-1}}{(k-1)!}\,d\theta = \sum_{i=0}^{\infty}\frac{z^i}{(i+k)!}, \qquad k = 1, 2, \ldots, \tag{5}$$

and satisfying the recurrence relation

$$\varphi_0(z) = e^z; \quad \varphi_{k+1}(z) = \frac{\varphi_k(z) - 1/k!}{z}, \quad k = 1, 2, \ldots; \quad \varphi_k(0) = \frac{1}{k!}. \tag{6}$$

As explained in [28], the construction of exponential integrators requires a numerical approximation of the integral term in Eq. (3), and to compute exponential-like matrix function times vector products $\boldsymbol{\varphi}_k(\cdot)\,v$ that include the second term in the right hand side of (3) as well as additional terms needed by the integral approximation.

Construction of an EPIRK-type scheme begins by considering a general ansatz [28] that describes an s-stage method:

$$\begin{aligned}
\mathbf{Y}_i &= \mathbf{y}_n + a_{i,1}\,\boldsymbol{\psi}_{i,1}(g_{i,1}\,h\mathbf{A}_{i,1})\,h\mathbf{f}(\mathbf{y}_n) + \sum_{j=2}^{i} a_{i,j}\,\boldsymbol{\psi}_{i,j}(g_{i,j}h\,\mathbf{A}_{i,j})\,h\Delta^{(j-1)}\mathbf{r}(\mathbf{y}_n),\\
&\quad i = 1, \ldots, s-1,\\
\mathbf{y}_{n+1} &= \mathbf{y}_n + b_1\,\boldsymbol{\psi}_{s,1}(g_{s,1}\,h\mathbf{A}_{s,1})\,h\mathbf{f}(\mathbf{y}_n) + \sum_{j=2}^{s} b_j\,\boldsymbol{\psi}_{s,j}(g_{s,j}h\,\mathbf{A}_{s,j})\,h\Delta^{(j-1)}\mathbf{r}(\mathbf{y}_n),
\end{aligned} \tag{7}$$

where $\mathbf{Y}_i$ are the internal stage vectors; $\mathbf{y}_n$, also denoted as $\mathbf{Y}_0$, is the input stage vector; and $\mathbf{y}_{n+1}$ is the final output stage of a single step. The j-th forward difference $\Delta^{(j)}\mathbf{r}(\mathbf{y}_n)$ is calculated using the residual values at the stage vectors:

$$\Delta^{(1)}\mathbf{r}(\mathbf{Y}_i) = \mathbf{r}(\mathbf{Y}_{i+1}) - \mathbf{r}(\mathbf{Y}_i), \quad \Delta^{(j)}\mathbf{r}(\mathbf{Y}_i) = \Delta^{(j-1)}\mathbf{r}(\mathbf{Y}_{i+1}) - \Delta^{(j-1)}\mathbf{r}(\mathbf{Y}_i).$$

Particular choices of matrices $\mathbf{A}_{ij}$, functions $\boldsymbol{\psi}_{ij}$ and $\mathbf{r}(y)$ define classes of EPIRK methods as well as allow derivation of specific schemes. In this paper we will focus on general, so called unpartitioned, EPIRK methods which can be constructed from (7) by choosing all matrices as a Jacobian matrix evaluated at t_n, i.e. $\mathbf{A}_{ij} = \mathbf{J}_n$, function $\mathbf{r}(y)$ to be the remainder of the first-order Taylor expansion of $\mathbf{f}(y)$ as in (4) and functions $\boldsymbol{\psi}_{ij}$ to be linear combinations of $\boldsymbol{\varphi}_k$'s:

$$\boldsymbol{\psi}_{i,j}(z) = \sum_{k=1}^{s} p_{i,j,k}\, \boldsymbol{\varphi}_k(z). \tag{8}$$

Here we will also use the simplifying assumption from [28, eq. 25]

$$\boldsymbol{\psi}_{i,j}(z) = \boldsymbol{\psi}_j(z) = \sum_{k=1}^{j} p_{j,k}\, \boldsymbol{\varphi}_k(z). \tag{9}$$

These choices lead to the class of general unpartitioned EPIRK schemes described by

$$\begin{aligned}
\mathbf{Y}_i &= \mathbf{y}_n + a_{i,1}\, \boldsymbol{\psi}_1(g_{i,1}\, h\mathbf{J}_n)\, h\mathbf{f}(\mathbf{y}_n) + \sum_{j=2}^{i} a_{i,j}\, \boldsymbol{\psi}_j(g_{i,j} h\, \mathbf{J}_n)\, h\Delta^{(j-1)}\mathbf{r}(\mathbf{y}_n), \\
& \quad i = 1, \ldots, s-1, \\
\mathbf{y}_{n+1} &= \mathbf{y}_n + b_1\, \boldsymbol{\psi}_1(g_{s,1}\, h\mathbf{J}_n)\, h\mathbf{f}(\mathbf{y}_n) + \sum_{j=2}^{s} b_j\, \boldsymbol{\psi}_j(g_{s,j} h\, \mathbf{J}_n)\, h\Delta^{(j-1)}\mathbf{r}(\mathbf{y}_n).
\end{aligned} \tag{10}$$

Coefficients $p_{j,k}$, $a_{i,j}$, $g_{i,j}$, and b_j have to be determined from order conditions to build schemes of specific order.

Remark 1 Computationally efficient strategies to evaluate exponential-like matrix function times vector products vary depending on the size of the ODE system. Approximating products $\boldsymbol{\psi}(z)v$ constitutes the major portion of overall computational cost of the method. Taylor expansion or Pade based approximations can be used to evaluate the products $\boldsymbol{\psi}(z)v$ for small systems. For large scale problems, particularly those where matrices cannot be stored explicitly and a matrix-free formulation of an integrator is a necessity, Krylov projection-based methods become the de facto choice [28].

EPIRK methods of a given order can be built by solving either classical [28] or stiff [21] order conditions. In this paper we will focus on the classically accurate EPIRK methods. A classical EPIRK scheme is constructed by matching required number of Taylor expansion coefficients of the exact and approximate solutions to derive order conditions. The order conditions are then solved to compute the coefficients of the method. To ensure that the error has the desired order, the exact Jacobian matrix $\mathbf{J}_n$ has to be used in the Taylor expansion of the numerical solution. If the Jacobian or its action on a vector is not computed with full accuracy, the numerical method will suffer from order reduction.

In this paper we construct EPIRK methods that retain full accuracy even with an approximated Jacobian. The theory of using inexact Jacobians in the method formulation, first

proposed in [26] for implicit schemes, is extended to EPIRK methods. More specifically, EPIRK-W methods that admit arbitrary approximations of the Jacobian while maintaining full order of accuracy are derived in Sect. 3. The drawback of W-methods, as mentioned in [10, section IV.7], is the very fast growth of the number of order conditions with increasing order of the method. In such cases, significantly larger number of stages are typically needed to build higher-order methods.

In [31,32] we developed K-methods, versions of W-methods that use a specific Krylov-subspace approximation of the Jacobian, and dramatically reduce the number of necessary order conditions. Here we construct K-methods for EPIRK schemes in Sect. 4. The K-method theory enables the construction of high order methods with significantly fewer stages than W-methods. For example, we show that three stage EPIRK W-methods only exist up to order three, whereas we derive here two three-stage fourth order K-methods. Furthermore, as shown in the the numerical results in Sect. 6, for some problems K-methods have better computational performance than traditional exponential integrators.

3 EPIRK-W Methods

An EPIRK-W method is formulated like a traditional EPIRK method (10) with the only difference being the use of an inexact Jacobian ($\mathbf{A}_n$) in place of the exact Jacobian ($\mathbf{J}_n$). Using the simplification (9) the method reads:

$$\begin{aligned}
\mathbf{Y}_i &= \mathbf{y}_n + a_{i,1}\,\boldsymbol{\psi}_1(g_{i,1}\,h\,\mathbf{A}_n)\,h\mathbf{f}(\mathbf{y}_n) + \sum_{j=2}^{i} a_{i,j}\,\boldsymbol{\psi}_j(g_{i,j}\,h\,\mathbf{A}_n)\,h\Delta^{(j-1)}\mathbf{r}(\mathbf{y}_n), \\
& \quad i = 1,\ldots,s-1, \\
\mathbf{y}_{n+1} &= \mathbf{y}_n + b_1\,\boldsymbol{\psi}_1(g_{s,1}\,h\,\mathbf{A}_n)\,h\mathbf{f}(\mathbf{y}_n) + \sum_{j=2}^{s} b_j\,\boldsymbol{\psi}_j(g_{s,j}\,h\,\mathbf{A}_n)\,h\Delta^{(j-1)}\mathbf{r}(\mathbf{y}_n). \qquad (11)
\end{aligned}$$

This requires the order conditions theory to be modified to accommodate the approximation.

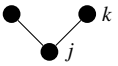
3.1 Order Conditions Theory for EPIRK-W Methods

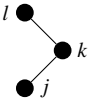
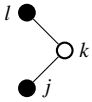
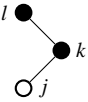
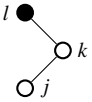
The classical order conditions for EPIRK-W methods result from matching the Taylor series expansion coefficients of the numerical solution $\mathbf{y}_{n+1}$ up to a certain order with those from the Taylor series expansion of the exact solution $\mathbf{y}(t_n + h)$. The construction of order conditions is conveniently expressed in terms of Butcher trees [9]. The trees corresponding to the elementary differentials of the numerical solution are the TW-trees. TW-trees are rooted Butcher trees with two different colored nodes—fat (empty) and meagre (full)—such that the end vertices are meagre and the fat vertices are singly branched. The TW-trees up to order four are shown in Tables 1 and 2 following [31].

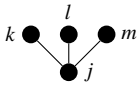
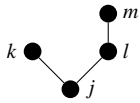
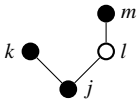
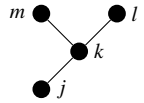
In TW-trees that correspond to elementary differentials of the Taylor expansion of numerical solution, the meagre nodes represents the appearance of f and its derivatives. The fat nodes represent the appearance of the inexact Jacobian $\mathbf{A}_n$. It is useful to note that trees that contain both meagre and fat nodes do not appear in the trees corresponding to the Taylor expansion of the exact solution, as such an expansion does not contain the inexact Jacobian.

In order to construct the order conditions for the W-method we rely on B-series theory. A B-series [9, Section II.12] is an operator that maps a set of of trees onto the set of real coefficients corresponding to the coefficients of the elementary differentials in a Taylor expansion. If

Table 1 TW-trees up to order four (part one of two) [31]

τ				
W-tree name	τ_1^W	τ_2^W	τ_3^W	τ_4^W
K-tree name	τ_1^K	τ_2^K	–	τ_3^K
$F(\tau)$	f^J	$f_K^J f^K$	$\mathbf{A}_{JK} f^K$	$f_{KL}^J f^K f^L$
$\mathsf{a}(\tau)$	x_1	x_2	x_3	x_4
$B^{\#}\,(h\mathbf{f}(B(\mathsf{a}, y)))$	1	x_1	0	x_1^2
$B^{\#}\,(h\mathbf{A}B(\mathsf{a}, y))$	0	0	x_1	0
$B^{\#}\left(\boldsymbol{\varphi}_j(h\,\mathbf{A})B(\mathsf{a}, y)\right)$	$c_0 x_1$	$c_0 x_2$	$c_0 x_3 + c_1 x_1$	$c_0 x_4$

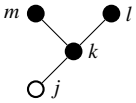
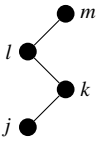
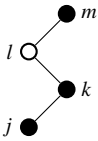
τ				
W-tree name	τ_5^W	τ_6^W	τ_7^W	τ_8^W
K-tree name	τ_4^K	–	–	–
$F(\tau)$	$f_K^J f_L^K f^L$	$f_K^J \mathbf{A}_{KL} f^L$	$\mathbf{A}_{JK} f_L^K f^L$	$\mathbf{A}_{JK}\mathbf{A}_{KL} f^L$
$\mathsf{a}(\tau)$	x_5	x_6	x_7	x_8
$B^{\#}\,(h\mathbf{f}(B(\mathsf{a}, y)))$	x_2	x_3	0	0
$B^{\#}\,(h\mathbf{A}B(\mathsf{a}, y))$	0	0	x_2	x_3
$B^{\#}\left(\boldsymbol{\varphi}_j(h\,\mathbf{A})B(\mathsf{a}, y)\right)$	$c_0 x_5$	$c_0 x_6$	$c_0 x_7 + c_1 x_2$	$c_0 x_8 + c_1 x_3 + c_2 x_1$

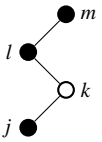
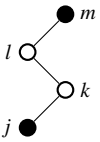
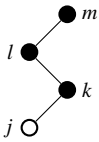
τ				
W-tree name	τ_9^W	τ_{10}^W	τ_{11}^W	τ_{12}^W
K-tree name	τ_5^K	τ_6^K	–	τ_7^K
$F(\tau)$	$f_{KLM}^J f^K f^L f^M$	$f_{KL}^J f_M^L f^M f^K$	$f_{KL}^J \mathbf{A}_{LM} f^M f^K$	$f_K^J f_{LM}^K f^M f^L$
$\mathsf{a}(\tau)$	x_9	x_{10}	x_{11}	x_{12}
$B^{\#}\,(h\mathbf{f}(B(\mathsf{a}, y)))$	x_1^3	$x_1 x_2$	$x_1 x_3$	x_4
$B^{\#}\,(h\mathbf{A}B(\mathsf{a}, y))$	0	0	0	0
$B^{\#}\left(\boldsymbol{\varphi}_j(h\,\mathbf{A})B(\mathsf{a}, y)\right)$	$c_0 x_9$	$c_0 x_{10}$	$c_0 x_{11}$	$c_0 x_{12}$

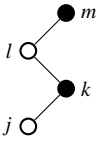
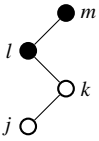
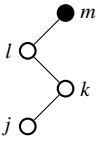
$\mathsf{a} : TW \cup \emptyset \to \mathbb{R}$ is a mapping from the set of TW trees and the empty tree to real numbers the corresponding B-series is given by:

$$B(\mathsf{a}, y) = \mathsf{a}(\emptyset)\, y + \sum_{\tau \in TW} \mathsf{a}(\tau)\, \frac{h^{|\tau|}}{\sigma(\tau)}\, F(\tau)(y).$$

Table 2 TW-trees up to order four (part two of two) [31]

τ			
W-tree name	τ_{13}^{W}	τ_{14}^{W}	τ_{15}^{W}
K-tree name	τ_{8}^{K}	τ_{9}^{K}	–
$F(\tau)$	$\mathbf{A}_{JK} f_{LM}^{K} f^{L} f^{M}$	$f_{K}^{J} f_{L}^{K} f_{M}^{L} f^{M}$	$f_{K}^{J} f_{L}^{K} \mathbf{A}_{LM} f^{M}$
$\mathsf{a}(\tau)$	x_{13}	x_{14}	x_{15}
$B^{\#}\left(h\mathbf{f}(B(\mathsf{a}, y))\right)$	0	x_5	x_6
$B^{\#}\left(h\mathbf{A}B(\mathsf{a}, y)\right)$	x_4	0	0
$B^{\#}\left(\boldsymbol{\varphi}_j(h\,\mathbf{A})B(\mathsf{a}, y)\right)$	$c_0 x_{13} + c_1 x_4$	$c_0 x_{14}$	$c_0 x_{15}$

τ			
W-tree name	τ_{16}^{W}	τ_{17}^{W}	τ_{18}^{W}
K-tree name	–	–	–
$F(\tau)$	$f_{K}^{J} \mathbf{A}_{KL} f_{M}^{L} f^{M}$	$f_{K}^{J} \mathbf{A}_{KL} \mathbf{A}_{LM} f^{M}$	$\mathbf{A}_{JK} f_{L}^{K} f_{M}^{L} f^{M}$
$\mathsf{a}(\tau)$	x_{16}	x_{17}	x_{18}
$B^{\#}\left(h\mathbf{f}(B(\mathsf{a}, y))\right)$	x_7	x_8	0
$B^{\#}\left(h\mathbf{A}B(\mathsf{a}, y)\right)$	0	0	x_5
$B^{\#}\left(\boldsymbol{\varphi}_j(h\,\mathbf{A})B(\mathsf{a}, y)\right)$	$c_0 x_{16}$	$c_0 x_{17}$	$c_0 x_{18} + c_1 x_5$

τ			
W-tree name	τ_{19}^{W}	τ_{20}^{W}	$\tau_{2,1}^{W}$
K-tree name	–	–	–
$F(\tau)$	$\mathbf{A}_{JK} f_{L}^{K} \mathbf{A}_{LM} f^{M}$	$\mathbf{A}_{JK} \mathbf{A}_{KL} f_{M}^{L} f^{M}$	$\mathbf{A}_{JK} \mathbf{A}_{KL} \mathbf{A}_{LM} f^{M}$
$\mathsf{a}(\tau)$	x_{19}	x_{20}	x_{21}
$B^{\#}\left(h\mathbf{f}(B(\mathsf{a}, y))\right)$	0	0	0
$B^{\#}\left(h\mathbf{A}B(\mathsf{a}, y)\right)$	x_6	x_7	x_8
$B^{\#}\left(\boldsymbol{\varphi}_j(h\,\mathbf{A})B(\mathsf{a}, y)\right)$	$c_0 x_{19} + c_1 x_6$	$c_0 x_{20} + c_1 x_7 + c_2 x_2$	$c_0 x_{21} + c_1 x_8 + c_2 x_3 + c_3 x_1$

Here τ is a TW-tree and $|\tau|$ is the order of the tree (the number of nodes of the TW-tree), and $\sigma(\tau)$ is the order of the symmetry group of the tree [5,28]. A simple algorithm for evaluating $\sigma(\tau)$ is given in [28]. Lastly, $F(\tau)(y)$ is the elementary differential corresponding to the tree τ evaluated at y.

We also make use of the operator $B^{\#}(g)$, first defined in [31], which takes a B-series g and returns its coefficients. Therefore:

$$B^{\#}\big(B(\mathsf{a}, y)\big) = \mathsf{a}.$$

TK-trees, a subset of TW-trees, arise when the Jacobian is approximated in the Krylov-subspace and are central to the discussion of K-methods, in the next section. Tables 1 and 2 list both TW- and TK-trees up to order four and the corresponding tree names. The subsequent rows of the two tables show the following:

- The elementary differentials corresponding to each TW-trees up to order four.
- A set of coefficients x_i for an arbitrary B-series $\mathsf{a}(\tau)$.
- The coefficients of the B-series resulting from composing the function f with the B-series $\mathsf{a}(\tau)$. The rules for composing B-series are discussed in [6,28].
- The coefficients of the B-series resulting from left-multiplying the B-series $\mathsf{a}(\tau)$ by an inexact Jacobian. The multiplication rule for B-series is explained in [6].
- The coefficients of the B-series resulting from left-multiplying the B-series $\mathsf{a}(\tau)$ by $\boldsymbol{\varphi}_j(h\,\mathbf{A}_n)$. The rules for multiplying a B-series by $\boldsymbol{\psi}(h\,\mathbf{J}_n)$, where $\mathbf{J}_n$ is the exact Jacobian at $\mathbf{y}_n$, are given in [28]. The rules for multiplying a B-series by $\boldsymbol{\psi}(h\,\mathbf{A}_n)$, for an arbitrary matrix $\mathbf{A}_n$, are given in [31].

In addition, we note that the B and $B^{\#}$ operators are linear. Consequently, the B-series of the sum of two arbitrary functions is the sum of the B-series of each and likewise, the operator $B^{\#}$ returns the sum of the coefficients of each lined up against corresponding elementary differentials.

Using B-series operations, as described above, one can systematically construct the B-series coefficients of the EPIRK-W numerical solution. This is achieved in Algorithm 1. The algorithm starts by initializing the set of B-series coefficients to those of the solution at the current time step, $\mathbf{y}_n$. Next, it operates on the sequence of coefficients by applying the B-series operations that correspond to the mathematical calculations performed during each stage of the EPIRK-W method. All B-series operations are done on coefficient sets truncated to the desired order (order four herein). We repeat this process for each internal stage, and finally for the last stage, to obtain the coefficients of the B-series corresponding to the numerical solution of one EPIRK-W step.

We now recall the following definition:

Definition 1 *(Density of a tree* [5,9]*)* The density $\gamma(\tau)$ of a tree τ is the product over all vertices of the orders of the sub-trees rooted at those vertices.

And we have the following theorem:

Theorem 1 (B-series of the exact solution) *The B-series expansion of the exact solution at the next step* $\mathbf{y}(t_n + h)$*, performed over TW-trees, has the coefficients:*

$$\mathsf{a}(\tau) = \begin{cases} 0 & \tau \in TW \setminus T, \\ \gamma(\tau) & \tau \in T. \end{cases}$$

where T is the set of trees corresponding to the exact solution.

Algorithm 1 Compute the B-series coefficient of numerical solution

Input: $\mathsf{y_n}$ ▷ B-series coefficient of the current numerical solution.
for $i = 1 : s - 1$ **do** ▷ Do for each internal stage of the method
  $\mathsf{u} = B^{\#}(h\,\mathbf{f}(B(\mathsf{y_n}, y)))$ ▷ Composition of f with B-series of the current solution.
  $\mathsf{u} = B^{\#}(\boldsymbol{\psi}_{i,1}(h\,g_{i,1}\,\mathbf{A}_n)\,\cdot\,B(\mathsf{u}, y))$ ▷ Multiplication by $\boldsymbol{\psi}$ function
  $\mathsf{u} = a_{i,1} * \mathsf{u}$ ▷ Scaling by a constant
  for $j = 2 : i$ **do**
    $\mathsf{v} = B^{\#}(h\,\Delta^{(j-1)}\mathbf{r}(\mathsf{y_n}))$ ▷ Recursive forward-difference
    $\mathsf{v} = B^{\#}(\boldsymbol{\psi}_{i,j}(h\,g_{i,j}\,\mathbf{A}_n)\,\cdot\,B(\mathsf{v}, y))$
    $\mathsf{u} = \mathsf{u} + a_{i,j} * \mathsf{v}$
  end for
  $\mathbf{Y_i} = \mathsf{y_n} + \mathsf{u}$ ▷ Addition of two B-series
end for
$\mathsf{u} = B^{\#}(h\,\mathbf{f}(B(\mathsf{y_n}, y)))$
$\mathsf{u} = B^{\#}(\boldsymbol{\psi}_{s,1}(h\,g_{s,1}\,\mathbf{A}_n)\,\cdot\,B(\mathsf{u}, y))$
$\mathsf{u} = b_1 * \mathsf{u}$
for $j = 2 : s$ **do**
  $\mathsf{v} = B^{\#}(h\,\Delta^{(j-1)}\mathbf{r}(\mathsf{y_n}))$
  $\mathsf{v} = B^{\#}(\boldsymbol{\psi}_{s,j}(h\,g_{s,j}\,\mathbf{A}_n)\,\cdot\,B(\mathsf{v}, y))$
  $\mathsf{u} = \mathsf{u} + b_j * \mathsf{v}$
end for
$\mathsf{y_{n+1}} = \mathsf{y_n} + \mathsf{u}$
Output: $\mathsf{y_{n+1}}$ ▷ B-series coefficient of the next step numerical solution.

Proof First part of the proof follows from the observation that the elementary differentials in the B-series expansion of the exact solution cannot have the approximate Jacobian appearing anywhere in their expressions. Second part of the proof follows from [9, Theorem 2.6, 2.11] and [5, Subsection 302, 380]. □

EPIRK-W order conditions are obtained by imposing that the B-series coefficients of the numerical solution match the B-series coefficients of the exact solution up to the desired order of accuracy.

3.2 Construction of Practical EPIRK-W Integrators

We now focus on constructing EPIRK-W methods (11) with three stages. Such a method using an approximate Jacobian reads:

$$\begin{aligned}
\mathbf{Y}_1 &= \mathbf{y}_n + a_{1,1}\,\boldsymbol{\psi}_{1,1}(g_{1,1}\,h\,\mathbf{A}_n)\,h\mathbf{f}(\mathbf{y}_n),\\
\mathbf{Y}_2 &= \mathbf{y}_n + a_{2,1}\,\boldsymbol{\psi}_{2,1}(g_{2,1}\,h\,\mathbf{A}_n)\,h\mathbf{f}(\mathbf{y}_n) + a_{2,2}\,\boldsymbol{\psi}_{2,2}(g_{2,2}\,h\,\mathbf{A}_n)\,h\Delta^{(1)}\mathbf{r}(\mathbf{y}_n),\\
\mathbf{y}_{n+1} &= \mathbf{y}_n + b_1\,\boldsymbol{\psi}_{3,1}(g_{3,1}\,h\,\mathbf{A}_n)\,h\mathbf{f}(\mathbf{y}_n) + b_2\,\boldsymbol{\psi}_{3,2}(g_{3,2}\,h\,\mathbf{A}_n)\,h\Delta^{(1)}\mathbf{r}(\mathbf{y}_n)\\
&\quad + b_3\,\boldsymbol{\psi}_{3,3}(g_{3,3}\,h\,\mathbf{A}_n)\,h\Delta^{(2)}\mathbf{r}(\mathbf{y}_n).
\end{aligned} \tag{12}$$

The order conditions corresponding to trees $\tau_1^W \ldots \tau_8^W$ of the three stage EPIRK-W method (12) are given in Table 3. The difference between B-series coefficients of the exact solution and of the numerical solution corresponding to τ_{14}^W turns out to be equal to $-1/24$ and cannot be zeroed out. τ_{14}^W is a tree that has four meagre nodes in it and appears in the B-series expansions of both the exact solution and the numerical solution. The conclusion is that three stage EPIRK-W methods with the simplified choice of $\boldsymbol{\psi}_{i,j}(z)$ given in Eq. (9) cannot achieve order four. Consequently, we will limit our solution procedure to constructing third order EPIRK-W methods by satisfying the above eight order conditions. We use

Table 3 Order conditions for the three stage EPIRK-W method

Tree	Order	Order condition: $B^{\#}(\mathbf{y}_{n+1}) - B^{\#}\big(\mathbf{y}(t_n + h)\big) = 0$
τ_1^W	1	$b_1\, p_{1,1} - 1 = 0$
τ_2^W	2	$\frac{1}{6}(6a_{1,1}\, b_2\, p_{1,1}\, p_{2,1} + 3a_{1,1}\, b_2\, p_{1,1}\, p_{2,2} - 12a_{1,1}\, b_3\, p_{1,1}\, p_{3,1} + 6a_{2,1}\, b_3\, p_{1,1}\, p_{3,1} - 6a_{1,1}\, b_3\, p_{1,1}\, p_{3,2} + 3a_{2,1}\, b_3\, p_{1,1}\, p_{3,2} - 2a_{1,1}\, b_3\, p_{1,1}\, p_{3,3} + a_{2,1}\, b_3\, p_{1,1}\, p_{3,3} - 3) = 0$
τ_3^W	2	$\frac{1}{6}\, p_{1,1}\,(3b_1\, g_{3,1} - 6a_{1,1}\, b_2\, p_{2,1} - 3a_{1,1}\, b_2\, p_{2,2} + 12a_{1,1}\, b_3\, p_{3,1} - 6a_{2,1}\, b_3\, p_{3,1} + 6a_{1,1}\, b_3\, p_{3,2} - 3a_{2,1}\, b_3\, p_{3,2} + 2a_{1,1}\, b_3\, p_{3,3} - a_{2,1}\, b_3\, p_{3,3}) = 0$
τ_4^W	3	$\frac{1}{6}\Big(6a_{1,1}^2\, b_2\, p_{2,1}\, p_{1,1}^2 + 3a_{1,1}^2\, b_2\, p_{2,2}\, p_{1,1}^2 - 12a_{1,1}^2\, b_3\, p_{3,1}\, p_{1,1}^2 + 6a_{2,1}^2\, b_3\, p_{3,1}\, p_{1,1}^2 - 6a_{1,1}^2\, b_3\, p_{3,2}\, p_{1,1}^2 + 3a_{2,1}^2\, b_3\, p_{3,2}\, p_{1,1}^2 - 2a_{1,1}^2\, b_3\, p_{3,3}\, p_{1,1}^2 + a_{2,1}^2\, b_3\, p_{3,3}\, p_{1,1}^2 - 2\Big) = 0$
τ_5^W	3	$\frac{1}{12}(12a_{1,1}\, a_{2,2}\, b_3\, p_{1,1}\, p_{2,1}\, p_{3,1} + 6a_{1,1}\, a_{2,2}\, b_3\, p_{1,1}\, p_{2,2}\, p_{3,1} + 6a_{1,1}\, a_{2,2}\, b_3\, p_{1,1}\, p_{2,1}\, p_{3,2} + 3a_{1,1}\, a_{2,2}\, b_3\, p_{1,1}\, p_{2,2}\, p_{3,2} + 2a_{1,1}\, a_{2,2}\, b_3\, p_{1,1}\, p_{2,1}\, p_{3,3} + a_{1,1}\, a_{2,2}\, b_3\, p_{1,1}\, p_{2,2}\, p_{3,3} - 2) = 0$
τ_6^W	3	$\frac{1}{12}\, p_{1,1}\,(6a_{1,1}\, b_2\, g_{1,1}\, p_{2,1} - 12a_{1,1}\, a_{2,2}\, b_3\, p_{3,1}\, p_{2,1} - 6a_{1,1}\, a_{2,2}\, b_3\, p_{3,2}\, p_{2,1} - 2a_{1,1}\, a_{2,2}\, b_3\, p_{3,3}\, p_{2,1} + 3a_{1,1}\, b_2\, g_{1,1}\, p_{2,2} - 12a_{1,1}\, b_3\, g_{1,1}\, p_{3,1} + 6a_{2,1}\, b_3\, g_{2,1}\, p_{3,1} - 6a_{1,1}\, a_{2,2}\, b_3\, p_{2,2}\, p_{3,1} - 6a_{1,1}\, b_3\, g_{1,1}\, p_{3,2} + 3a_{2,1}\, b_3\, g_{2,1}\, p_{3,2} - 3a_{1,1}\, a_{2,2}\, b_3\, p_{2,2}\, p_{3,2} - 2a_{1,1}\, b_3\, g_{1,1}\, p_{3,3} + a_{2,1}\, b_3\, g_{2,1}\, p_{3,3} - a_{1,1}\, a_{2,2}\, b_3\, p_{2,2}\, p_{3,3}) = 0$
τ_7^W	3	$\frac{1}{24}\, p_{1,1}\,(12a_{1,1}\, b_2\, g_{3,2}\, p_{2,1} - 24a_{1,1}\, a_{2,2}\, b_3\, p_{3,1}\, p_{2,1} - 12a_{1,1}\, a_{2,2}\, b_3\, p_{3,2}\, p_{2,1} - 4a_{1,1}\, a_{2,2}\, b_3\, p_{3,3}\, p_{2,1} + 4a_{1,1}\, b_2\, g_{3,2}\, p_{2,2} - 24a_{1,1}\, b_3\, g_{3,3}\, p_{3,1} + 12a_{2,1}\, b_3\, g_{3,3}\, p_{3,1} - 12a_{1,1}\, a_{2,2}\, b_3\, p_{2,2}\, p_{3,1} - 8a_{1,1}\, b_3\, g_{3,3}\, p_{3,2} + 4a_{2,1}\, b_3\, g_{3,3}\, p_{3,2} - 6a_{1,1}\, a_{2,2}\, b_3\, p_{2,2}\, p_{3,2} - 2a_{1,1}\, b_3\, g_{3,3}\, p_{3,3} + a_{2,1}\, b_3\, g_{3,3}\, p_{3,3} - 2a_{1,1}\, a_{2,2}\, b_3\, p_{2,2}\, p_{3,3}) = 0$
τ_8^W	3	$\frac{1}{24}\, p_{1,1}\,\Big(4b_1\, g_{3,1}^2 - 12a_{1,1}\, b_2\, g_{1,1}\, p_{2,1} - 12a_{1,1}\, b_2\, g_{3,2}\, p_{2,1} - 6a_{1,1}\, b_2\, g_{1,1}\, p_{2,2} - 4a_{1,1}\, b_2\, g_{3,2}\, p_{2,2} + 24a_{1,1}\, b_3\, g_{1,1}\, p_{3,1} - 12a_{2,1}\, b_3\, g_{2,1}\, p_{3,1} + 24a_{1,1}\, b_3\, g_{3,3}\, p_{3,1} - 12a_{2,1}\, b_3\, g_{3,3}\, p_{3,1} + 24a_{1,1}\, a_{2,2}\, b_3\, p_{2,1}\, p_{3,1} + 12a_{1,1}\, a_{2,2}\, b_3\, p_{2,2}\, p_{3,1} + 12a_{1,1}\, b_3\, g_{1,1}\, p_{3,2} - 6a_{2,1}\, b_3\, g_{2,1}\, p_{3,2} + 8a_{1,1}\, b_3\, g_{3,3}\, p_{3,2} - 4a_{2,1}\, b_3\, g_{3,3}\, p_{3,2} + 12a_{1,1}\, a_{2,2}\, b_3\, p_{2,1}\, p_{3,2} + 6a_{1,1}\, a_{2,2}\, b_3\, p_{2,2}\, p_{3,2} + 4a_{1,1}\, b_3\, g_{1,1}\, p_{3,3} - 2a_{2,1}\, b_3\, g_{2,1}\, p_{3,3} + 2a_{1,1}\, b_3\, g_{3,3}\, p_{3,3} - a_{2,1}\, b_3\, g_{3,3}\, p_{3,3} + 4a_{1,1}\, a_{2,2}\, b_3\, p_{2,1}\, p_{3,3} + 2a_{1,1}\, a_{2,2}\, b_3\, p_{2,2}\, p_{3,3}\Big) = 0$

`Mathematica`® to solve the order conditions using two different approaches, as discussed next.

First approach to solving the order conditions In the first approach we seek to make the terms of each order condition as similar to another as possible. We start the solution process by noticing that the first order condition trivially reduces to the substitution $b_1 \to 1/p_{1,1}$. The substitution $p_{3,2} \to -2p_{3,1}$ will zero out the following four terms: $-12a_{1,1}\, b_3\, p_{1,1}\, p_{3,1} + 6a_{2,1}\, b_3\, p_{1,1}\, p_{3,1} - 6a_{1,1}\, b_3\, p_{1,1}\, p_{3,2} + 3a_{2,1}\, b_3\, p_{1,1}\, p_{3,2}$, in order conditions τ_2^W and τ_3^W and hence reducing the complexity of the two conditions. It is immediately observed that $g_{3,1} \to 1$ as all the other terms in order conditions τ_2^W and τ_3^W are the same. Additionally, we make a choice that $g_{3,3} \to 0$. After making the substitutions, we solve order condition τ_2^W or τ_3^W to get,

Table 4 Coefficients for EPIRKW3A

$$a = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad \begin{bmatrix} b \\ \widehat{b} \end{bmatrix} = \begin{bmatrix} \frac{3}{4} & \frac{1}{2} & 1 \\ \frac{3}{4} & \frac{3}{4} & \frac{6}{5} \end{bmatrix}, \quad g = \begin{bmatrix} \frac{2}{3} & 0 & 0 \\ 0 & 0 & 0 \\ 1 & \frac{3}{5} & 0 \end{bmatrix}, \quad p = \begin{bmatrix} \frac{4}{3} & 0 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & \frac{3}{4} \end{bmatrix}.$$

$$p_{3,3} \rightarrow \frac{3(2a_{1,1}\, b_2\, p_{1,1}\, p_{2,1} + a_{1,1}\, b_2\, p_{1,1}\, p_{2,2} - 1)}{b_3\, p_{1,1}\, (2a_{1,1} - a_{2,1})}.$$

This substitution results in a number of terms in multiple order conditions having the expression $(2a_{1,1} - a_{2,1})$ in the denominator. So we arbitrarily choose $a_{2,1} \rightarrow 2a_{1,1} - 1$. The order conditions are far simpler now with several terms having the factor $(-1 + 2a_{1,1})$ and its powers in their expression. We choose $a_{1,1} \rightarrow 1/2$. Following this substitution, we can solve from order condition τ_4^W that $p_{1,1} \rightarrow 4/3$. After plugging in the value for $p_{1,1}$, order conditions τ_5^W and τ_6^W suggest that $g_{1,1} \rightarrow 2/3$. Now order conditions τ_5^W, τ_6^W, τ_7^W and τ_8^W have the following coefficients as part of their expressions: $a_{2,2}$, $p_{2,1}$, $p_{2,2}$, $g_{3,2}$ and b_2. We arbitrarily set $p_{2,2} \rightarrow 2p_{2,1}$ and solve for the remaining coefficients to obtain

$$b_2 \rightarrow \frac{3a_{2,2}\, p_{2,1} + 1}{8a_{2,2}\, p_{2,1}^2} \quad \text{and} \quad g_{3,2} \rightarrow \frac{12\, a_{2,2}\, p_{2,1}}{5(3\, a_{2,2}\, p_{2,1} + 1)}.$$

We now select arbitrary values for some of the remaining coefficients. We choose $p_{2,1} \rightarrow 1$, $a_{2,2} \rightarrow 1$, $b_3 \rightarrow 1$, $a_{1,2} \rightarrow 0$, $a_{1,3} \rightarrow 0$, $a_{2,3} \rightarrow 0$, $p_{3,1} \rightarrow 0$, $p_{2,3} \rightarrow 0$, $p_{1,2} \rightarrow 0$, $p_{1,3} \rightarrow 0$, $g_{1,2} \rightarrow 0$, $g_{1,3} \rightarrow 0$, $g_{2,1} \rightarrow 0$, $g_{2,2} \rightarrow 0$, and $g_{2,3} \rightarrow 0$.

In order to solve for a second order embedded method we rewrite the final stage of the method with new coefficients $\widehat{b}_i$. Plugging in the substitutions that have been arrived at while solving the order conditions for the third order method results in a set of new conditions in only the $\widehat{b}_i$. It is again straightforward to observe that $\widehat{b}_1 \rightarrow 1/p_{1,1}$. We next solve the conditions for *TW*-trees up to order = 2. Solving order conditions τ_2^W and τ_3^W we get $\widehat{b}_3 \rightarrow -3 + 8\,\widehat{b}_2$.

The resulting coefficients for a third-order EPIRK-*W* method (EPIRKW3A) with an embedded second-order method are given in Table 4.

The choices that we have made for the coefficients a's, b's, g's and p's result in the sum of coefficients on trees τ_5^W, τ_6^W, τ_7^W and τ_8^W to sum to zero in the embedded method no matter what value is chosen for $\widehat{b}_2$. And when we choose $\mathbf{A}_n = \mathbf{J}_n$, the exact Jacobian, trees τ_5^W, τ_6^W, τ_7^W and τ_8^W are the same tree and it turns out that we incidentally solve the embedded method up to third order as well. To get a second-order embedded method, we need to make a, g and p be independent of b and solve separately for $\widehat{b}$.

Second approach to solving the order conditions In this approach we impose a horizontal structure on the g coefficients akin to the horizontally adaptive method described in [21]. In order to impose a horizontal structure on g we make the following substitutions: $g_{1,2} \rightarrow g_{1,1}$, $g_{1,3} \rightarrow g_{1,1}$, $g_{2,2} \rightarrow g_{2,1}$, $g_{2,3} \rightarrow g_{2,1}$, $g_{3,2} \rightarrow g_{3,1}$, and $g_{3,3} \rightarrow g_{3,1}$. We also note that the first order condition reduces to $b_1 \rightarrow 1/p_{1,1}$ again. Order-conditions τ_2^W and τ_3^W imply $g_{3,1} \rightarrow 1$. We solve order-conditions τ_2^W and τ_3^W and get two different solutions for a subset of the variables. We plugin one of the two solutions and solve order conditions τ_4^W, τ_5^W and τ_6^W. We continue this process of solving a subset of order conditions, each time plugging the result back into the remaining order conditions. This process led to the following family of order three method coefficients:

Table 5 Coefficients for EPIRKW3B

$$a = \begin{bmatrix} 0.22824182961171620396 & 0 & 0 \\ 0.45648365922343240794 & 0.33161664063356950085 & 0 \end{bmatrix},$$

$$b = \begin{bmatrix} 1 \\ 2.0931591383832578214 \\ 1.2623969257900804404 \end{bmatrix}^T,$$

$$\widehat{b} = \begin{bmatrix} 1 \\ 2.0931591383832578214 \\ 1 \end{bmatrix}^T,$$

$$g = \begin{bmatrix} 0 & 0 & 0 \\ 0.34706341174296320958 & 0.34706341174296320958 & 0.34706341174296320958 \\ 1 & 1 & 1 \end{bmatrix},$$

$$p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2.0931604100438501004 & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

Table 6 Coefficients for EPIRKW3C

$$a = \begin{bmatrix} \frac{282}{311} & 0 & 0 \\ \frac{294}{311} & \frac{-7}{94} & 0 \end{bmatrix}, \quad \begin{bmatrix} b \\ \widehat{b} \end{bmatrix} = \begin{bmatrix} 1 & \frac{-3421}{987} & \frac{-622}{105} \\ 1 & \frac{13}{9} & 1 \end{bmatrix}, \quad g = \begin{bmatrix} \frac{1}{5} & 0 & 0 \\ \frac{1}{8} & \frac{1}{8} & 0 \\ 1 & 1 & 1 \end{bmatrix}, \quad p = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}.$$

$$\begin{aligned}
&g_{1,2} \to g_{1,1}\,, g_{1,3} \to g_{1,1}\,, g_{2,2} \to g_{2,1}\,, g_{2,3} \to g_{2,1}\,, g_{3,2} \to g_{3,1}\,, g_{3,3} \to g_{3,1}\,,\\
&b_1 \to \frac{1}{p_{1,1}}, b_2 \to -\frac{a_{2,2}\,(3a_{2,1}\,p_{1,1}\,-4)}{2a_{2,1}^2\,p_{1,1}^2}, b_3 \to -\frac{3a_{2,1}\,p_{1,1}\,-4}{a_{2,1}^2\,p_{1,1}^2\,(6p_{3,1}\,+3p_{3,2}\,+p_{3,3})},\\
&p_{2,1} \to 0, a_{1,1} \to \frac{a_{2,1}}{2}, g_{3,1} \to 1, g_{2,1} \to \frac{2(3a_{2,1}\,g_{1,1}\,p_{1,1}\,-a_{2,1}\,p_{1,1}\,-2g_{1,1})}{3a_{2,1}\,p_{1,1}\,-4},\\
&p_{2,2} \to -\frac{2\left(9a_{1,1}\,a_{2,1}\,a_{2,2}\,p_{1,1}\,p_{2,1}\,-6a_{1,1}\,a_{2,1}\,p_{1,1}\,-12a_{1,1}\,a_{2,2}\,p_{2,1}\,+8a_{1,1}\,+6a_{2,1}^2\,p_{1,1}\,-4a_{2,1}\right)}{3a_{1,1}\,a_{2,2}\,(3a_{2,1}\,p_{1,1}\,-4)}.
\end{aligned} \tag{13}$$

After making the above substitutions in the embedded method, we obtain the following solutions for the embedded coefficients:

$$\widehat{b_1} \to \frac{1}{p_{1,1}}, \quad \widehat{b_2} \to -\frac{a_{2,2}\,(3a_{2,1}\,p_{1,1}\,-4)}{2a_{2,1}^2\,p_{1,1}^2},$$

with $\widehat{b_3}$ a free parameter that can be chosen to obtain a second order embedded method.

A solution to the above family leads to the third order EPIRK-W method (EPIRKW3B) in Table 5, with a second order embedded method.

Additionally, we derived EPIRKW3C, a third order W-method with a second order embedded method, to have full a, b, g and p coefficient matrices such that all $\boldsymbol{\psi}$ function products appearing in the three stage formulation (12) need to be computed and contribute to the final one-step solution. Table 6 lists the coefficients for EPIRKW3C.

4 EPIRK-K Methods

W-methods have the advantage that any approximation of the Jacobian that ensures stability can be used, therefore they have the potential for attaining excellent computational efficiency in comparison to methods that require exact Jacobians. The drawback of the W-methods, however, is the very large number of order conditions that need to be solved for constructing the method [10, Section IV.7].

In order to reduce the number of order conditions Krylov-based methods (K-methods) were developed for Rosenbrock integrators in [32] and for exponential integrators in [31]. K-methods are built in the framework of W-methods and use a very specific approximation to the Jacobian constructed in the Krylov space. This approximation allows TW-trees with linear sub-trees having fat root to be re-colored (as meagre), leading to the new class of TK-trees. The recoloring results in substantially fewer trees, and therefore fewer order conditions for the K-methods [32, Lemma 3.2, 3.3].

In this section we derive K-methods in the framework of EPIRK integrators.

4.1 Krylov-Subspace Approximation of Jacobian

K-methods build the Krylov-subspace M-dimensional Krylov-subspace $\mathcal{K}_M$, $M \ll N$, based on the exact Jacobian $\mathbf{J}_n$ and the ODE function value $\mathbf{f}_n$ at the current time step t_n:

$$\mathcal{K}_M = \text{span}\{\mathbf{f}_n, \mathbf{J}_n\mathbf{f}_n, \mathbf{J}_n^2\mathbf{f}_n, \ldots, \mathbf{J}_n^{M-1}\mathbf{f}_n\}. \tag{14}$$

The modified Arnoldi iteration [35] computes an orthonormal basis $\mathbf{V}$ and upper-Hessenberg matrix $\mathbf{H}$ such that:

$$\begin{aligned} \mathcal{K}_M = \text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_M\}, \quad \mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_M] \in \mathbb{R}^{N\times M}, \\ \mathbf{V}^T\mathbf{V} = \mathbf{I}_M, \quad \mathbf{H} = \mathbf{V}^T\mathbf{J}_n\mathbf{V} \in \mathbb{R}^{M\times M}. \end{aligned} \tag{15}$$

Using $\mathbf{H}$ and $\mathbf{V}$ the approximation of the Jacobian matrix $\mathbf{J}_n$ in the Krylov sub-space is defined as

$$\mathbf{A}_n = \mathbf{V}\,\mathbf{H}\,\mathbf{V}^T = \mathbf{V}\mathbf{V}^T\mathbf{J}_n\mathbf{V}\mathbf{V}^T \in \mathbb{R}^{N\times N}. \tag{16}$$

This approximation of the Jacobian is used by K-methods. The important properties of this approximation are given by the following Lemmas from [31].

Lemma 1 (Powers of $\mathbf{A}_n$ [31]) *Powers of* $\mathbf{A}_n$*:* $\mathbf{A}_n^k = \mathbf{V}\mathbf{H}^k\mathbf{V}^T$ *for any* $k \geq 1$.

Lemma 2 (Evaluation of $\boldsymbol{\varphi}$ functions (5) on the approximate Jacobian (16) [31]) *We have that:*

$$\boldsymbol{\varphi}_k(h\,\gamma\,\mathbf{A}_n) = \frac{1}{k!}\left(\mathbf{I}_N - \mathbf{V}\mathbf{V}^T\right) + \mathbf{V}\,\boldsymbol{\varphi}_k(h\,\gamma\,\mathbf{H})\,\mathbf{V}^T, \quad k = 1, 2, \ldots.$$

In order to derive the Krylov formulation of the EPIRK methods (12) we need to extend Lemma 2 to the evaluation of $\boldsymbol{\psi}$ functions (9). We have the following result.

Lemma 3 (Evaluation of $\boldsymbol{\psi}$ functions (9) on the approximate Jacobian (16))

$$\boldsymbol{\psi}_j(h\,\gamma\,\mathbf{A}_n) = \widetilde{p}_j\left(\mathbf{I}_N - \mathbf{V}\mathbf{V}^T\right) + \mathbf{V}\,\boldsymbol{\psi}_j\,(h\gamma\mathbf{H})\;\mathbf{V}^T, \;\; j = 1, 2, \ldots \;\; \textit{where} \;\; \widetilde{p}_j = \sum_{k=1}^{j}\frac{p_{j,k}}{k!}.$$

Proof From the definition (9) we have:

$$\begin{aligned}\boldsymbol{\psi}_j(h\gamma\mathbf{A}_n) &= \sum_{k=1}^{j} p_{j,k}\,\boldsymbol{\varphi}_k(h\,\gamma\,\mathbf{A}_n)\\ &= \sum_{k=1}^{j} p_{j,k}\left[\frac{1}{k!}(\mathbf{I}_N - \mathbf{V}\mathbf{V}^T) + \mathbf{V}\,\boldsymbol{\varphi}_k(h\,\gamma\,\mathbf{H})\,\mathbf{V}^T\right]\\ &= (\mathbf{I}_N - \mathbf{V}\mathbf{V}^T)\left[\sum_{k=1}^{j}\frac{p_{j,k}}{k!}\right] + \mathbf{V}\,\boldsymbol{\psi}_j(h\gamma\mathbf{H})\,\mathbf{V}^T\\ &= \widetilde{p}_j(\mathbf{I}_N - \mathbf{V}\mathbf{V}^T) + \mathbf{V}\,\boldsymbol{\psi}_j(h\gamma\mathbf{H})\,\mathbf{V}^T.\end{aligned}$$

□

4.2 Formulation of EPIRK-*K* Methods

We now derive the Krylov-subspace formulation of the EPIRK methods, which will be called EPIRK-*K* methods. For this we begin with the EPIRK-*W* formulation (11) and use the Jacobian approximation (16). The first step in the method derivation is to split all vectors appearing in the method formulation (11) into components within the Krylov-subspace and components orthogonal to it, as follows:

1. Splitting the internal stage vectors leads to:
$$\mathbf{Y}_i = \mathbf{V}\boldsymbol{\lambda}_i + \mathbf{Y}_i^{\perp} \quad \text{where} \quad \mathbf{V}^T\mathbf{Y}_i = \boldsymbol{\lambda}_i, \quad \left(\mathbf{I}_N - \mathbf{V}\mathbf{V}^T\right)\mathbf{Y}_i = \mathbf{Y}_i^{\perp}, \tag{17}$$
where $\mathbf{Y}_0 \equiv \mathbf{y}_n$.
2. Splitting the right-hand side function evaluated at the internal stage vectors gives:
$$\mathbf{f}_i := \mathbf{f}(\mathbf{Y}_i) = \mathbf{V}\boldsymbol{\eta}_i + \mathbf{f}_i^{\perp} \quad \text{where} \quad \mathbf{V}^T\mathbf{f}_i = \boldsymbol{\eta}_i, \quad \left(\mathbf{I}_N - \mathbf{V}\mathbf{V}^T\right)\mathbf{f}_i = \mathbf{f}_i^{\perp}, \tag{18}$$
where $\mathbf{f}_0 \equiv \mathbf{f}(\mathbf{y}_n)$.
3. Splitting the non-linear Taylor remainder terms of the right-hand side functions yields:
$$\begin{aligned}&\mathbf{r}(\mathbf{Y}_i) = \mathbf{f}(\mathbf{Y}_i) - \mathbf{f}(\mathbf{y}_n) - \mathbf{A}_n\,(\mathbf{Y}_i - \mathbf{y}_n) = \mathbf{f}_i - \mathbf{f}_0 - \mathbf{V}\,\mathbf{H}\,\mathbf{V}^T\,(\mathbf{Y}_i - \mathbf{y}_n),\\ &\text{where} \quad \mathbf{V}^T\,\mathbf{r}(\mathbf{Y}_i) = \boldsymbol{\eta}_i - \boldsymbol{\eta}_0 - \mathbf{H}\,(\boldsymbol{\lambda}_i - \boldsymbol{\lambda}_0),\\ &\left(\mathbf{I}_N - \mathbf{V}\mathbf{V}^T\right)\mathbf{r}(\mathbf{Y}_i) = \mathbf{f}_i^{\perp} - \mathbf{f}_0^{\perp}.\end{aligned} \tag{19}$$
4. Splitting the forward differences of the non-linear remainder terms leads to:
$$\begin{aligned}&\widetilde{\mathbf{r}}_{(j-1)} := \Delta^{(j-1)}\mathbf{r}(\mathbf{y}_n) = \mathbf{V}\,\mathbf{d}_{(j-1)} + \widetilde{\mathbf{r}}^{\perp}_{(j-1)},\\ &\text{where} \quad \mathbf{V}^T\,\widetilde{\mathbf{r}}_{(j-1)} = \mathbf{d}_{(j-1)}, \quad \left(\mathbf{I}_N - \mathbf{V}\mathbf{V}^T\right)\widetilde{\mathbf{r}}_{(j-1)} = \widetilde{\mathbf{r}}^{\perp}_{(j-1)}.\end{aligned} \tag{20}$$

In the above equations, $\mathbf{V}\boldsymbol{\lambda}_i$, $\mathbf{V}\boldsymbol{\eta}_i$ and $\mathbf{V}\mathbf{d}_{(j-1)}$ are components of $\mathbf{Y}_i$, $\mathbf{f}_i$ and $\Delta^{(j-1)}\mathbf{r}(\mathbf{y}_n)$ in the Krylov-subspace, whereas $\mathbf{Y}_i^{\perp}$, $\mathbf{f}_i^{\perp}$ and $\widetilde{\mathbf{r}}^{\perp}_{(j-1)}$ lie in the orthogonal subspace.

Using the above equations and Lemma 3, the intermediate stage equations of the method (11)

$$\mathbf{Y}_i = \mathbf{y}_n + a_{i,1}\,\boldsymbol{\psi}_1\left(g_{i,1}\,h\,\mathbf{V}\,\mathbf{H}\,\mathbf{V}^T\right)\,h\mathbf{f}(\mathbf{y}_n) + \sum_{j=2}^{i} a_{i,j}\,\boldsymbol{\psi}_j\left(g_{i,j}\,h\,\mathbf{V}\,\mathbf{H}\,\mathbf{V}^T\right)\,h\Delta^{(j-1)}\mathbf{r}(\mathbf{y}_n), \tag{21}$$

become:

$$\mathbf{Y}_i = \mathbf{V}\boldsymbol{\lambda}_i + \mathbf{Y}_i^{\perp} = \mathbf{y}_n + h\,a_{i,1}\Big(\widetilde{p}_1\,(\mathbf{f}_0 - \mathbf{V}\boldsymbol{\eta}_0) + \mathbf{V}\,\boldsymbol{\psi}_1(h\,g_{i,1}\,\mathbf{H})\boldsymbol{\eta}_0\Big) + \sum_{j=2}^{i} h\,a_{i,j}\left(\widetilde{p}_j\,\widetilde{\mathbf{r}}_{(j-1)}^{\perp} + \mathbf{V}\,\boldsymbol{\psi}_j(h\,g_{i,j}\,\mathbf{H})\,\mathbf{d}_{(j-1)}\right), \tag{22}$$

where $\widetilde{\mathbf{r}}_{(j-1)}^{\perp}$ and $\mathbf{d}_{(j-1)}$ can be proved to be, as is done in "Appendix B",

$$\mathbf{d}_{(j-1)} = \sum_{k=0}^{j-1}\left((-1)^k\binom{j-1}{k}\boldsymbol{\eta}_{j-1-k} - \mathbf{H}\left((-1)^k\binom{j-1}{k}\boldsymbol{\lambda}_{j-1-k}\right)\right), \tag{23a}$$

$$\widetilde{\mathbf{r}}_{(j-1)}^{\perp} = \sum_{k=0}^{j-1}\left((-1)^k\binom{j-1}{k}(\mathbf{f}_{j-1-k} - \mathbf{V}\boldsymbol{\eta}_{j-1-k})\right). \tag{23b}$$

The reduced stage vector for the K-method is obtained by multiplying the full stage vector in Eq. (22) by $\mathbf{V}^T$ from the left,

$$\boldsymbol{\lambda}_i = \boldsymbol{\lambda}_0 + h\,a_{i,1}\boldsymbol{\psi}_1(h\,g_{i,1}\,\mathbf{H})\,\boldsymbol{\eta}_0 + \sum_{j=2}^{i} h\,a_{i,j}\,\boldsymbol{\psi}_j(h\,g_{i,j}\,\mathbf{H})\,\mathbf{d}_{(j-1)}, \tag{24}$$

and the component of the full stage vector, when multiplied by $(I - \mathbf{V}\mathbf{V}^T)$, in the orthogonal subspace is,

$$\mathbf{Y}_i^{\perp} = (\mathbf{y}_n - \mathbf{V}\boldsymbol{\lambda}_0) + h\,a_{i,1}\,\widetilde{p}_1\,(\mathbf{f}_0 - \mathbf{V}\boldsymbol{\eta}_0) + \sum_{j=2}^{i} h\,a_{i,j}\,\widetilde{p}_j\,\widetilde{\mathbf{r}}_{(j-1)}^{\perp}. \tag{25}$$

The full-stage vector can be recovered by first projecting the reduced stage vector $\boldsymbol{\lambda}_i$ back into full space and adding the piece that is orthogonal to it, $\mathbf{Y}_i^{\perp}$, as done in Eq. (22).

Similarly, the computation of the next solution in the method (11)

$$\mathbf{y}_{n+1} = \mathbf{y}_n + b_1\,\boldsymbol{\psi}_{s,1}(g_{s,1}\,h\,\mathbf{A}_n)\,h\mathbf{f}(\mathbf{y}_n) + \sum_{j=2}^{s} b_j\,\boldsymbol{\psi}_{s,j}(g_{s,j}\,h\,\mathbf{A}_n)\,h\Delta^{(j-1)}\mathbf{r}(\mathbf{y}_n), \tag{26}$$

becomes:

$$\mathbf{y}_{n+1} = \mathbf{V}\boldsymbol{\lambda}_s + \mathbf{y}_{n+1}^{\perp}, \tag{27}$$

where

$$\boldsymbol{\lambda}_s = \boldsymbol{\lambda}_0 + h\,b_1\,\boldsymbol{\psi}_1(h\,g_{s,1}\,\mathbf{H})\boldsymbol{\eta}_0 + \sum_{j=2}^{s} h\,b_j\,\boldsymbol{\psi}_j(h\,g_{s,j}\,\mathbf{H})\,\mathbf{d}_{(j-1)}, \tag{28}$$

and

$$\mathbf{y}_{n+1}^{\perp} = (\mathbf{y}_n - \mathbf{V}\boldsymbol{\lambda}_0) + h\,b_1\,\widetilde{p}_1\,(\mathbf{f}_0 - \mathbf{V}\boldsymbol{\eta}_0) + \sum_{j=2}^{s} h\,b_j\,\widetilde{p}_j\,\widetilde{\mathbf{r}}_{(j-1)}^{\perp}. \tag{29}$$

One step of the resulting EPIRK-K method (for an autonomous system) is summarized in Algorithm 2.

Algorithm 2 EPIRK-K

$\mathbf{f}_0 := \mathbf{f}(\mathbf{y}_n)$ ▷ Repeat for every timestep in the timespan

$\mathbf{J}_n := \mathbf{J}(\mathbf{y}_n)$

$[\mathbf{H}, \mathbf{V}] = \text{Arnoldi}(\mathbf{J}_n, \mathbf{f}_0)$

$\boldsymbol{\lambda}_0 = \mathbf{V}^T \mathbf{y}_n$

$\boldsymbol{\eta}_0 = \mathbf{V}^T \mathbf{f}_0$

for $i = 1 : s - 1$ **do** ▷ Do for each stage of the method

$$\mathbf{d}_{(i-1)} = \begin{cases} 0 & , \text{if } i = 1 \\ \sum_{k=0}^{i-1} \left((-1)^k \binom{i-1}{k} \boldsymbol{\eta}_{i-1-k} - \mathbf{H} \left((-1)^k \binom{i-1}{k} \boldsymbol{\lambda}_{i-1-k} \right) \right) & , \text{if } i \geq 2 \end{cases}$$

$$\widetilde{\mathbf{r}}^{\perp}_{(i-1)} = \begin{cases} 0 & , \text{if } i = 1 \\ \sum_{k=0}^{i-1} \left((-1)^k \binom{i-1}{k} (\mathbf{f}_{i-1-k} - \mathbf{V}\, \boldsymbol{\eta}_{i-1-k}) \right) & , \text{if } i \geq 2 \end{cases}$$

$$\boldsymbol{\lambda}_i = \boldsymbol{\lambda}_0 + h\, a_{i,1}\, \boldsymbol{\psi}_1(h\, g_{i,1}\, \mathbf{H})\, \boldsymbol{\eta}_0 + \sum_{j=2}^{i} h\, a_{i,j}\, \boldsymbol{\psi}_j(h\, g_{i,j}\, \mathbf{H})\, \mathbf{d}_{(j-1)}$$

$$\mathbf{Y}_i = \mathbf{V}\, \boldsymbol{\lambda}_i + (\mathbf{y}_n - \mathbf{V}\, \boldsymbol{\lambda}_0) + h\, a_{i,1}\, \widetilde{p_1}\, (\mathbf{f}_0 - \mathbf{V}\, \boldsymbol{\eta}_0) + \sum_{j=2}^{i} h\, a_{i,j}\, \widetilde{p}_j\, \widetilde{\mathbf{r}}^{\perp}_{(j-1)}$$

$\mathbf{f}_i = \mathbf{f}(\mathbf{Y}_i)$

$\boldsymbol{\eta}_i = \mathbf{V}^T \mathbf{f}_i$

end for

$$\mathbf{d}_{(s-1)} = \sum_{k=0}^{s-1} \left((-1)^k \binom{s-1}{k} \boldsymbol{\eta}_{s-1-k} - \mathbf{H} \left((-1)^k \binom{s-1}{k} \boldsymbol{\lambda}_{s-1-k} \right) \right)$$

$$\widetilde{\mathbf{r}}^{\perp}_{(s-1)} = \sum_{k=0}^{s-1} \left((-1)^k \binom{s-1}{k} (\mathbf{f}_{s-1-k} - \mathbf{V}\, \boldsymbol{\eta}_{s-1-k}) \right)$$

$$\boldsymbol{\lambda}_s = \boldsymbol{\lambda}_0 + h\, b_1\, \boldsymbol{\psi}_1(h\, g_{s,1}\, \mathbf{H})\, \boldsymbol{\eta}_0 + \sum_{j=2}^{s} h\, b_j\, \boldsymbol{\psi}_j(h\, g_{s,j}\, \mathbf{H})\, \mathbf{d}_{(j-1)}$$

$$\mathbf{y}_{n+1} = \mathbf{V}\, \boldsymbol{\lambda}_s + (\mathbf{y}_n - \mathbf{V}\, \boldsymbol{\lambda}_0) + h\, b_1\, \widetilde{p_1}\, (\mathbf{f}_0 - \mathbf{V}\, \boldsymbol{\eta}_0) + \sum_{j=2}^{s} h\, b_j\, \widetilde{p}_j\, \widetilde{\mathbf{r}}^{\perp}_{(j-1)}$$

4.3 Order Conditions Theory for EPIRK-K Methods

K-methods construct a single Krylov-subspace per timestep, and use it to approximate the Jacobian. All stage vectors are also computed in this reduced space, before being projected back into the full space between successive stages. The order condition theory accounts for this computational procedure [32, Theorem 3.6].

Before we discuss the order conditions for the K-methods, we define the trees that arise in the expansion of their solutions. Recalling that a linear tree is one where each node has only one child, consider the following definition:

Definition 2 *(TK-Trees* [32]*)*

$$TK = \left\{ \text{TW-trees: no linear sub-tree has a fat (empty) root} \right\}.$$

$$TK(k) = \left\{ \text{TW-trees: no linear sub-tree of order } \leqslant \text{ k has a fat (empty) root} \right\}.$$

Theorem 2 *Trees corresponding to series expansion of the numerical solution of K-method with Krylov-subspace dimension M are TK(M).*

Proof Using Lemmas 1, 2, 3, and [31, Lemmas 3, 4] we arrive at the desired result. □

The order conditions are derived from matching the coefficients of the B-series expansion of the numerical solution to those of the B-series expansion of the exact solution. If the expansion is made in the elementary differentials corresponding to the *TK* trees, then for a fourth order method there is a single additional tree (τ_8^K) in the numerical solution having both meagre and fat nodes in comparison to the *T*-trees up to order four for a classical EPIRK method. In contrast with the *W*-method that has twenty-one order conditions, the *K*-method has just nine for a fourth order method, which is almost the same as a classical EPIRK having eight.

As mentioned earlier, *TK*-trees come from re-coloring all the linear sub-trees with a fat root as meagre in the *TW*-trees. This significantly reduces the number of order conditions for the *K*-method since groups of *TW*-trees become isomorphic to one another after recoloring. *TK*-trees up to order four are given in [31]. This also indicates that the coefficients in front of these *TK*-trees in the B-series expansion can be obtained by summing together the coefficients of *TW*-trees, which become isomorphic to one another, from the corresponding expansion.

The order four conditions for EPIRK-*K* methods with three stages (12) are discussed next and are summarized in Table 7.

Corollary 1 *Any EPIRK-K-method of order p gives rise to a classical EPIRK method of order p (with the same coefficients).*

Proof The proof follows directly from comparing the set of *T*-trees to *TK*-trees. □

Corollary 2 *Any classical EPIRK method of order $p \geq 3$ gives rise to an equivalent K-method of order at least three (with the same coefficients).*

Proof The proof follows directly from comparing the set of *T*-trees to *TK*-trees. □

4.4 Construction of Practical EPIRK-*K* Integrators

Note that order condition τ_8^K is the additional order condition that was mentioned earlier whose *TW*-tree could not be re-colored according to [32, Lemmas 3.2, 3.3]. This is the only tree that is not present in *T*-trees up to order 4, and therefore we impose that the associated coefficient is equal to zero. To solve the order conditions we use `Mathematica`® and the solution procedure described for the first variant of EPIRK-*W* method where we make terms of each order condition similar to one another by making suitable substitutions. We arrive at the EPIRK-*K* method (EPIRKK4A) of order four in Table 8, with an embedded method of order three.

We also derived EPIRKK4B, a fourth-order *K*-method with a third order embedded method, whose coefficients are given in Table 9. EPIRKK4B has full a, b, g and p matrices so that every $\boldsymbol{\psi}$ function product appearing in the three stage formulation will contribute to the computation of the final one-step solution. Furthermore, the g coefficients were chosen so as to minimize the number of matrix exponential operations needed for the three stage formulation.

The theory of *K*-methods gives a lower bound on the Krylov-subspace size that guarantees the order of convergence [32, Theorem 3.6]. This bound depends only on the order of convergence of the method and not on the dimension of the ODE system. Here we have constructed two fourth order EPIRK-*K* methods, whose coefficients are listed in Tables 8 and 9, and they

Table 7 Order conditions for the three stage EPIRK-K method

Tree #	Order	Order condition: $B^{\#}(\mathbf{y}_{n+1}) - B^{\#}\big(\mathbf{y}(t_n + h)\big) = 0$
τ_1^K	1	$b_1\, p_{1,1} - 1 = 0$
τ_2^K	2	$\frac{1}{2}(b_1\, g_{3,1}\, p_{1,1} - 1) = 0$
τ_3^K	3	$\frac{1}{6}\Big(6a_{1,1}^2\, b_2\, p_{1,1}^2\, p_{2,1} + 3a_{1,1}^2\, b_2\, p_{1,1}^2\, p_{2,2} - 12a_{1,1}^2\, b_3\, p_{1,1}^2\, p_{3,1} - 6a_{1,1}^2\, b_3\, p_{1,1}^2\, p_{3,2} - 2a_{1,1}^2\, b_3\, p_{1,1}^2\, p_{3,3} + 6a_{2,1}^2\, b_3\, p_{1,1}^2\, p_{3,1} + 3a_{2,1}^2\, b_3\, p_{1,1}^2\, p_{3,2} + a_{2,1}^2\, b_3\, p_{1,1}^2\, p_{3,3} - 2\Big) = 0$
τ_4^K	3	$\frac{1}{6}\Big(b_1\, g_{3,1}^2\, p_{1,1} - 1\Big) = 0$
τ_5^K	4	$\frac{1}{12}\Big(12a_{1,1}^3\, b_2\, p_{1,1}^3\, p_{2,1} + 6a_{1,1}^3\, b_2\, p_{1,1}^3\, p_{2,2} - 24a_{1,1}^3\, b_3\, p_{1,1}^3\, p_{3,1} - 12a_{1,1}^3\, b_3\, p_{1,1}^3\, p_{3,2} - 4a_{1,1}^3\, b_3\, p_{1,1}^3\, p_{3,3} + 12a_{2,1}^3\, b_3\, p_{1,1}^3\, p_{3,1} + 6a_{2,1}^3\, b_3\, p_{1,1}^3\, p_{3,2} + 2a_{2,1}^3\, b_3\, p_{1,1}^3\, p_{3,3} - 3\Big) = 0$
τ_6^K	4	$\frac{1}{24}\Big(12a_{1,1}^2\, b_2\, g_{1,1}\, p_{1,1}^2\, p_{2,1} + 6a_{1,1}^2\, b_2\, g_{1,1}\, p_{1,1}^2\, p_{2,2} - 24a_{1,1}^2\, b_3\, g_{1,1}\, p_{1,1}^2\, p_{3,1} - 12a_{1,1}^2\, b_3\, g_{1,1}\, p_{1,1}^2\, p_{3,2} - 4a_{1,1}^2\, b_3\, g_{1,1}\, p_{1,1}^2\, p_{3,3} + 12a_{2,1}^2\, b_3\, g_{2,1}\, p_{1,1}^2\, p_{3,1} + 6a_{2,1}^2\, b_3\, g_{2,1}\, p_{1,1}^2\, p_{3,2} + 2a_{2,1}^2\, b_3\, g_{2,1}\, p_{1,1}^2\, p_{3,3} - 3\Big) = 0$
τ_7^K	4	$\frac{1}{12}\Big(12a_{1,1}^2\, a_{2,2}\, b_3\, p_{1,1}^2\, p_{2,1}\, p_{3,1} + 6a_{1,1}^2\, a_{2,2}\, b_3\, p_{1,1}^2\, p_{2,1}\, p_{3,2} + 2a_{1,1}^2\, a_{2,2}\, b_3\, p_{1,1}^2\, p_{2,1}\, p_{3,3} + 6a_{1,1}^2\, a_{2,2}\, b_3\, p_{1,1}^2\, p_{2,2}\, p_{3,1} + 3a_{1,1}^2\, a_{2,2}\, b_3\, p_{1,1}^2\, p_{2,2}\, p_{3,2} + a_{1,1}^2\, a_{2,2}\, b_3\, p_{1,1}^2\, p_{2,2}\, p_{3,3} - 1\Big) = 0$
τ_8^K	4	$\frac{1}{24}p_{1,1}^2\Big(-24a_{1,1}^2\, a_{2,2}\, b_3\, p_{2,1}\, p_{3,1} - 12a_{1,1}^2\, a_{2,2}\, b_3\, p_{2,1}\, p_{3,2} - 4a_{1,1}^2\, a_{2,2}\, b_3\, p_{2,1}\, p_{3,3} - 12a_{1,1}^2\, a_{2,2}\, b_3\, p_{2,2}\, p_{3,1} - 6a_{1,1}^2\, a_{2,2}\, b_3\, p_{2,2}\, p_{3,2} - 2a_{1,1}^2\, a_{2,2}\, b_3\, p_{2,2}\, p_{3,3} + 12a_{1,1}^2\, b_2\, g_{3,2}\, p_{2,1} + 4a_{1,1}^2\, b_2\, g_{3,2}\, p_{2,2} - 24a_{1,1}^2\, b_3\, g_{3,3}\, p_{3,1} - 8a_{1,1}^2\, b_3\, g_{3,3}\, p_{3,2} - 2a_{1,1}^2\, b_3\, g_{3,3}\, p_{3,3} + 12a_{2,1}^2\, b_3\, g_{3,3}\, p_{3,1} + 4a_{2,1}^2\, b_3\, g_{3,3}\, p_{3,2} + a_{2,1}^2\, b_3\, g_{3,3}\, p_{3,3}\Big) = 0$
τ_9^K	4	$\frac{1}{24}\Big(b_1\, g_{3,1}^3\, p_{1,1} - 1\Big) = 0$

Table 8 Coefficients for EPIRKK4A

$$a = \begin{bmatrix} \frac{692665874901013}{799821658665135} & 0 & 0 \\ \frac{692665874901013}{799821658665135} & \frac{3}{4} & 0 \end{bmatrix}, \qquad \begin{bmatrix} b \\ \widehat{b} \end{bmatrix} = \begin{bmatrix} \frac{799821658665135}{692665874901013} & \frac{352}{729} & \frac{64}{729} \\ \frac{799821658665135}{692665874901013} & \frac{32}{81} & 0 \end{bmatrix},$$

$$g = \begin{bmatrix} \frac{3}{4} & 0 & 0 \\ \frac{3}{4} & 0 & 0 \\ 1 & \frac{9}{16} & \frac{9}{16} \end{bmatrix}, \qquad p = \begin{bmatrix} \frac{692665874901013}{799821658665135} & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}.$$

require a Krylov-subspace of dimension four [32, Theorem 3.6]. All expensive operations such as computing products of $\boldsymbol{\psi}$ function of matrices with vectors are performed in this reduced space. Significant computational advantages are obtained if the Krylov-subspace captures all the stiff eigenmodes of the system. If not all stiff eigenmodes are captured, stability requirements will force the integrator to take smaller timesteps, which will increase

Table 9 Coefficients for EPIRKK4B

$$a = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}, \qquad \begin{bmatrix} b \\ \widehat{b} \end{bmatrix} = \begin{bmatrix} \frac{4}{3} & \frac{112}{243} & 1 \\ \frac{4}{3} & \frac{80}{243} & -1 \end{bmatrix},$$

$$g = \begin{bmatrix} \frac{3}{4} & 0 & 0 \\ \frac{3}{4} & \frac{3}{4} & 0 \\ 1 & \frac{3}{4} & \frac{3}{4} \end{bmatrix}, \qquad p = \begin{bmatrix} \frac{3}{4} & 0 & 0 \\ 1 & 1 & 0 \\ 1 & \frac{-962}{243} & \frac{524}{81} \end{bmatrix}.$$

the overall cost of integration. In such cases we typically observe that adding more vectors to the Krylov-subspace can improve the performance of the K-type integrator.

5 Implementation Strategies for Exponential Integrators

Computing products of $\boldsymbol{\varphi}$ (or $\boldsymbol{\psi}$) functions of matrices times vectors is at the core of exponential integrators. We briefly review the strategies we use to evaluate these products on a per integrator basis before we segue into a discussion on the construction of Krylov-subspaces that are an integral part of this process. Table 10 lists the methods under study and their implementation framework.

5.1 Evaluation of $\boldsymbol{\varphi_k}$ and $\boldsymbol{\psi}$ Products

We start our discussion with the classical exponential integrators listed in Table 10. For EPIRKK4- CLASSICAL and EPIRK5 [28, Table 4, Equation 28] the evaluation of products of $\boldsymbol{\psi}$ functions with vectors proceeds by first approximating the individual $\boldsymbol{\varphi}$ function products in the Krylov-subspace as illustrated in [27, sec 3.1], i.e. $\boldsymbol{\varphi}_k(h\,\gamma\,\mathbf{A}_n)\mathbf{b} \approx s\|\mathbf{b}\|\mathbf{V}\boldsymbol{\varphi}_k(h\,\gamma\,\mathbf{H})\mathbf{e}_1$. The $\boldsymbol{\varphi}$ function products associated with the approximation, $\boldsymbol{\varphi}_k(h\,\gamma\,\mathbf{H})\mathbf{e}_1$, are computed by constructing an augmented-matrix and exponentiating it as described in [25, Theorem 1]. Finally, taking a linear combination of columns of the resultant matrix gives the $\boldsymbol{\psi}$ function product.

In the augmented-matrix approach, an Arnoldi (or Lanczos – for a symmetric matrix) iteration is needed for each distinct vector in the formulation given in Eq. (10). Each of the classical integrators listed earlier has three stages that work with three different vectors, $\mathbf{b}$, requiring three Arnoldi (or Lanczos) iterations per timestep. EPIRK5P1BVAR [29], another classical method that we include in our numerical experiments, also performs three Arnoldi (or Lanczos) projections per timestep. It combines the Arnoldi (or Lanczos) projection, and the evaluation of $\boldsymbol{\varphi}$ function products by sub-stepping, into a single computational process as described in [20].

Recall that W-methods admit arbitrary Jacobian approximations while maintaining full accuracy. In order to demonstrate this, we have four alternate implementations that use different approximations to the Jacobian: the Jacobian itself; the identity matrix; the zero matrix, which reduces the EPIRKW scheme (12) to an explicit Runge–Kutta scheme; and lastly, the diagonal of the Jacobian. We will refer to these methods as EPIRKW3 in the paper and highlight the approximation in context. Figure 1 in the following section shows that these alternative implementations retain full order when we use an approximation to $\mathbf{J}_n$. It is to be noted, however, that in the case of W-methods, $\mathbf{A}_n = \mathbf{J}_n$ might be needed to assure stability. Some discussion in this regard when $\mathbf{A}_n = \mathbf{J}_n$ or when $\|\mathbf{A}_n - \mathbf{J}_n\|$ is small, is done in [10, Sec IV.7, IV.11] and [26].

The different implementations of the W-method vary in the way the $\boldsymbol{\psi}$ function products are computed. We restrict our discussion to the following cases:

1. $\mathbf{A}_n = \text{diag}(\mathbf{J}_n)$. We compute the individual $\boldsymbol{\varphi}_k$ functions with successively higher subscripts using the recursive definition given in (6), where $\boldsymbol{\varphi}_0$ is computed as point-wise exponential of the entries along the diagonal. Next, we take a linear combination of the $\boldsymbol{\varphi}$ functions to evaluate the $\boldsymbol{\psi}$ function as defined in Eq. (9). Finally, the product of $\boldsymbol{\psi}$ function of matrix times vector is evaluated as a matrix-vector product. A similar procedure is adapted for other approximations where $\mathbf{A}_n$ is either zero or identity.
2. $\mathbf{A}_n = \mathbf{J}_n$. The evaluation of products of $\boldsymbol{\psi}$ function of Jacobian with vectors is similar to classical integrators. An Arnoldi (or Lanczos) iteration is needed for each distinct vector in the three stage formulation given in Eq. (12). Implementations with non-trivial approximations to the Jacobian may use a similar strategy to compute the $\boldsymbol{\psi}$ function products.

We have implemented EPIRKK4, a fourth-order K-type integrator, and EPIRK5- K [28, Table 4], a classical fifth-order method built in the K-framework. Both use the reduced space formulation for stage values as shown in Algorithm 2. The reduced space is constructed using a single Arnoldi (or Lanczos) projection per timestep. K-methods require only as many basis vectors in the reduced space as the order of method to guarantee full accuracy [32]. This is in direct contrast to classical methods, where the basis size of Arnoldi (or Lanczos) projection is varied to keep the residual of $\boldsymbol{\varphi_1}$ function evaluated on the upper-Hessenberg matrix, $\mathbf{H_m}$, below a chosen tolerance, resulting in larger subspaces.

EPIRKK4, being a fourth order K-method, theoretically, requires only four vectors in the Krylov-subspace for full accuracy. Stability requirements may, however, impose a need for larger subspaces. Since the stage values of EPIRKK4 are computed in this reduced space, the size of the matrices used in the computation of $\boldsymbol{\psi}$ function products is usually smaller than the size of the ODE system under consideration. As a result, in our implementation of EPIRKK4, quantities $\boldsymbol{\varphi}_k(h\,\gamma\,\mathbf{H})$ are directly computed using the recursive definition given in (6); they are stored independently, and linearly combined to compute the $\boldsymbol{\psi}$ function. The product of a $\boldsymbol{\psi}$ function of a matrix with a vector is computed as a matrix-vector product. We repeat the procedure with five basis vectors in the Krylov-subspace for EPIRK5- K.

5.2 Building Krylov-Subspaces

When solving large systems of ODEs, $\boldsymbol{\varphi}$ (and $\boldsymbol{\psi}$) function products are approximated in the Krylov-subspace. We use Lanczos iteration to compute the subspace when the Jacobian (or its approximation) is symmetric, and we use Arnoldi when the Jacobian is not symmetric. The computational complexity of Lanczos iteration scales linearly with the size of subspace, whereas it scales quadratically with the subspace size for Arnoldi. Order condition theory for classical exponential methods relies on using the exact Jacobian when computing $\boldsymbol{\varphi}$ (or $\boldsymbol{\psi}$) function products. As a consequence, larger subspaces are usually needed to approximate the action of the Jacobian accurately. W- and K-methods, on the other hand, account for Jacobian approximations in their order condition theory and therefore work satisfactorily with comparably smaller subspaces.

For stiff problems, classical methods incur a huge cost per timestep in building large subspaces to capture enough stiff modes, albeit they take larger timesteps. In classical methods, if the cost of building the subspace is disproportionately high such that even taking larger timesteps does not offset it, then W- and K-methods turn out to be more efficient, despite

the fact that they shrink the stepsize. We observe this to be the case, particularly, when we use Arnoldi iteration to compute the subspace.

We now delve into the details of construction of Krylov-subspaces for each method that we evaluate in our variable time-stepping experiments later in the paper.

- **EPIRKK4** We test two different implementations of EPIRKK4. The first one constructs the Krylov-subspace adaptively, by limiting the residual of $\boldsymbol{\varphi_1}$ evaluated on the (scaled) upper-Hessenberg matrix, $\mathbf{H_m}$, to be below 1e−12. The residual computations are performed only at certain fixed indices, where the cost of computing a residual equals the total cost of all previous residual computations [13, Section 6.4]. The maximum dimension of the subspace is set to 100; and the minimum, which has to be at-least four for fourth order accuracy, is indicated on plot legends in the subsequent section as a lower bound on the subspace dimension. Residual checks only occur once the minimum bound has been exceeded.
  The second implementation of EPIRKK4 constructs Krylov-subspaces with a fixed number of basis vectors, and does not incur the cost of computing the residuals. In each experiment, we choose a range of subspace sizes that are a subset of the indices at which the adaptive variant computes the residuals. We only plot those that performed the best.
- **EPIRKK4-CLASSICAL**: In our implementation, the Krylov-subspace is built adaptively using the desired solution tolerance as the tolerance during residual checks, with maximum dimension of the subspace left unrestricted.
- **EPIRKW3**: The implementation uses adaptive Krylov to construct subspaces by limiting the residual to be below the desired solution tolerance, and the maximum dimension of the subspace is set to 100.
- **EPIRK5P1BVAR**: No limits are imposed on the subspace size as this would intervene with the algorithm that builds the Krylov-subspace and does sub-stepping in a single computational process to compute the φ function product [20]. Implementation uses the desired solution tolerance to determine when the Krylov algorithm has converged.

We also include a Rosenbrock–Krylov method, ROK4A [32], in our numerical experiments. The implementation of ROK4A that we use has Arnoldi iteration baked into it. Consequentially, we do not use it in numerical experiments that involve symmetric problems, where we employ Lanczos iteration to build Krylov-subspaces in other integrators, as such a comparison will be unfair to ROK4A. Unlike exponential methods that use the Krylov-subspace to compute $\boldsymbol{\varphi}$ (and $\boldsymbol{\psi}$) function products, ROK4A uses it to solve linear systems corresponding to reduced stage vectors. Furthermore, our ROK4A implementation builds Krylov-subspaces of fixed size and does not incur cost of computing residuals.

6 Numerical Results

The integrators discussed in Sect. 5 were evaluated by running fixed-step convergence experiments on the Lorenz-96 system [17] and variable time-stepping experiments on Allen–Cahn [2] and BSVD system [11]. We used the `MATLODE` framework [8], a `MATLAB`® -based ODE solver suite developed by the co-authors, in performing the variable time-stepping experiments.

6.1 Lorenz-96 System

Lorenz-96 model [17] is described by the system of ODEs:

Table 10 Comparative study of various exponential integrators

Integrator	Implementation framework	Coefficients	Derived order	Fixed step convergence order
‡ EPIRKW3B [$\mathbf{A}_n = \mathbf{0}$]	W-type	Table 5	3	2.977000
‡ EPIRKW3B [$\mathbf{A}_n = \text{diag}(\mathbf{J}_n)$]	W-type	Table 5	3	2.967430
‡ EPIRKW3B [$\mathbf{A}_n = \mathbf{I}$]	W-type	Table 5	3	2.987911
EPIRKW3B [$\mathbf{A}_n = \mathbf{J}_n$]	W-type	Table 5	3	2.994241
EPIRKW3C [$\mathbf{A}_n = \mathbf{J}_n$]	W-type	Table 6	3	3.032945
EPIRKK4B	K-type	Table 9	4	4.014407
EPIRKK4A	K-type	Table 8	4	4.018722
EPIRKK4A- CLASSICAL	Classical	Table 8	4	4.009777
‡ EPIRK5- K	K-type	[28, Table 4]	5*	3.051511
‡ EPIRK5	Classical	[28, Table 4]	5	5.016092
EPIRK5P1BVAR	Classical	[29, Table 3]	5	–
ROK4A	K-type	[32, Table 1]	4	–

The convergence orders shown are obtained from fixed step size experiments with the Lorenz-96 model (30). Integrators implemented only for studying fixed step convergence behavior that are excluded from variable timestepping experiments are prefixed with ‡ in column 1. * indicates the order of the original classical method EPIRK5

$$\frac{dy_j}{dt} = -y_{j-1}(y_{j-2} - y_{j+1}) - y_j + F, \quad j = 1 \ldots N, \quad y_0 = y_N. \tag{30}$$

Here N denotes the number of states and F is the forcing term. For our experiments we let $N = 40$, and $F = 8$. The initial condition for the model was obtained by integrating $\bar{y}_{-1} = \text{linspace}\,(-2, 2, N)$ over $[0, 0.3]$ time unit using `MATLAB`'s `ODE45` integrator and time span for the experiment is $[0, 0.3]$ time units.

Fixed-step convergence experiment was performed on the Lorenz-96 system using a subset of integrators listed in Table 10. The reference solution was computed using `ODE45` with absolute and relative tolerances set to 1e−12. The convergence plots for this experiment are shown in Fig. 1. The results for each implementation and coefficient combination are summarized in Table 10. These results support the following conclusions:

1. The numerical experiment verifies the order of methods derived in the paper. EPIRKK4 (A and B) and EPIRKW3 (B and C) show their respective theoretical order of convergence.
2. The theory of W-methods is validated by testing different approximations to the Jacobian. As expected, EPIRKW3B consistently shows third order convergence irrespective of the approximation used for the Jacobian.
3. The results demonstrate that every K-method can be implemented as a classical method of the same order. Here we test EPIRKK4A- CLASSICAL, the implementation of the method EPIRKK4A (see Sect. 4.4) in the classical framework as discussed in Sect. 5. It has the same order of convergence as EPIRKK4A since the coefficients derived for EPIRKK4A method satisfy all the order conditions of a fourth order classical EPIRK method. This is true for every K-method as shown in Corollary 1.
4. In general, classical methods implemented as a K-method will reduce to a lower order. EPIRK5 and EPIRK5- K were implemented to test this hypothesis, where EPIRK5- K is a K-type implementation of the fifth order classical method EPIRK5 derived in [28]. From Table 10, we see that the K-type implementation reduces to a lower order. The coeffi-

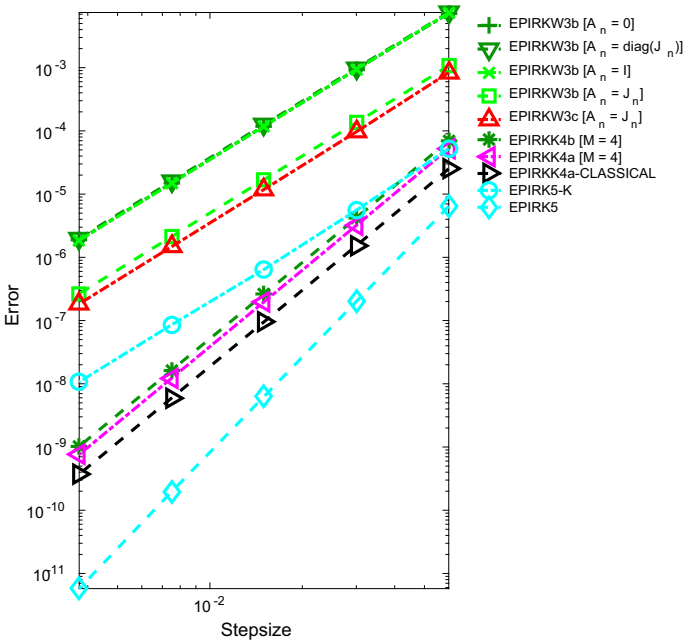


Fig. 1 Convergence plot for different methods applied to Lorenz-96 model (30)

cients of fifth order classical method EPIRK5 puts a non-zero in front of the elementary differential corresponding to τ_8^K in the K-type implementation.
Solving the order conditions for a K-type method is more restrictive than for a classical method due to the additional order conditions that arise from approximating the Jacobian in the Krylov-subspace. As a consequence, not all classical methods lead to K-type integrators of the same order. This is in line with Corollary 2.

To evaluate the relative computational efficiency of the integrators—EPIRKK4 (A and B), EPIRKW3 (B and C), and EPIRKK4A- CLASSICAL—we perform variable time-stepping experiments with the Allen–Cahn system [2], and the BSVD system [11]. We include a classical exponential scheme, EPIRK5P1BVAR [29], a Rosenbrock–Krylov method, ROK4A, [32], and MATLAB's built-in one-step integrator for stiff systems, ODE23S, as benchmarks for comparison. Experiments are run for different relative tolerances [1e−1, 3.9e−3, . . . , 1e−8]; the error controller [9, Section II.4], which is the same across all methods, with the exception of ODE23S, adjusts the step size to ensure that the solution meets tolerance requirements. ODE23S has its own error controller built inside MATLAB®. Reference solutions are computed using ODE15S with absolute and relative tolerances set equal to 1e−12.

6.2 The Allen–Cahn Problem

We consider the Allen–Cahn equation [2]:

$$\frac{\partial u}{\partial t} = \alpha\,\nabla^2 u + \gamma\,(u - u^3), \quad (x, y) \in [0, 1] \times [0, 1]\ \text{(space units)}, \quad t \in [0, 0.3]\ \text{(time units)}. \tag{31}$$

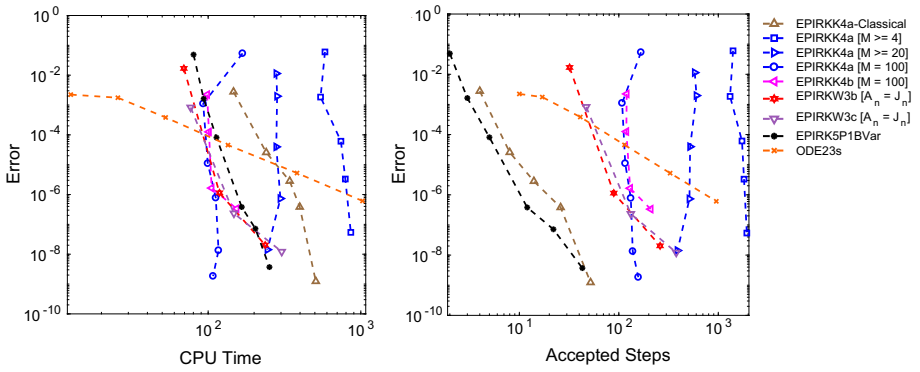


Fig. 2 Comparison of different integrators when applied to Allen–Cahn system (31) on a 300 × 300 grid. (using Lanczos to compute Krylov subspace). **a** Work precision diagram. **b** Convergence diagram

discretized using second-order finite difference on a 300×300 grid. The diffusion coefficient, α, equals 1 and the coefficient on reaction term, γ, equals 10. The model is subject to homogeneous Neumann boundary conditions and the initial condition is $u(t = 0) = 0.4 + 0.1\,(x + y) + 0.1\,\sin(10x)\,\sin(20y)$.

The Jacobian of Allen–Cahn equation (31) is symmetric. Therefore, the methods use Lanczos iteration to compute the Krylov-subspace. ROK4A is excluded from this study as the implementation we use has Arnoldi baked in. Figure 2 shows both the work-precision and convergence diagram for the integrators under study (excluding ROK4A) on Allen–Cahn equations.

6.2.1 Observations

Explicit methods ODE45, and MATLODE's DOPRI5 do not converge to a solution for the given settings.

The timesteps of EPIRKK4 methods are bounded by stability considerations indicated by the nearly vertical lines in the convergence diagram 2. Nonetheless, EPIRKK4 [$M = 100$] (both A and B) perform better than classical integrators—EPIRKK4A- CLASSICAL and EPIRK5P1BVAR—because they are cheaper per timestep by virtue of building comparably smaller subspaces, and not incurring the cost of computing residuals. Also, the computational cost of constructing large subspaces by classical methods (see Table 11) is not offset by the large step sizes that they take. Furthermore, in additional experiments, not reported here, artificially limiting the subspace size of the classical methods leads to a poor performing method, where the global error is $\mathcal{O}(1)$ for several tolerance settings.

Adaptive variants EPIRKK4A [$M \geq 4$] and EPIRKK4A [$M \geq 20$] are computationally more expensive than EPIRKK4 [$M = 100$] (both A and B) with fixed basis size. They incur additional costs in computing the residuals and end up taking many more steps than all of the other integrators (see convergence diagram 2). One reason could be that the estimate of the residual inside adaptive Krylov algorithm, which is used to terminate the Krylov iterations, turns out to be an underestimate. This leads to an early termination of the Krylov iterations resulting in smaller subspaces for the adaptive methods. Furthermore, K-methods only do a single projection unlike three projections per timestep in other methods that we consider.

Table 11 Root mean square number of Krylov vectors per projection for each integrator applied to Allen–Cahn equation on a 300 × 300 grid (31)

Integrator	Tolerance					
	1e−1	3.98e-3	1.58e-4	6.31e-6	2.51e-7	1e−8
EPIRKK4a-classical	2	669	564	493	402	309
EPIRKK4a [$M \geq 4$]	23	23	23	24	25	25
EPIRKK4a [$M \geq 20$]	28	33	33	40	42	52
EPIRKK4a [$M = 100$]	100	100	100	100	100	100
EPIRKK4b [$M = 100$]	–	100	100	100	100	–
EPIRKW3b [$A_n = J_n$]	93	83	93	78	60	39
EPIRKW3c [$A_n = J_n$]	2	58	69	67	49	29
EPIRK5P1BVar	3099	2811	2543	1754	1523	1294

The smaller subspaces coupled with the single projection inside K-methods restricts the stepsize of the adaptive K-methods (EPIRKK4A [$M \geq 4$] and EPIRKK4A [$M \geq 20$]). It is not immediately clear what measure of residual to use for adaptive K-methods to make them perform better. Comparing EPIRKK4A [$M \geq 4$] and EPIRKK4A [$M \geq 20$] suggests that setting a higher value for the lower bound of the subspace size is one solution.

EPIRKW3 (both B and C) performs just as well as EPIRKK4 methods with fixed basis size for low-medium accuracy solutions. For tighter solution tolerances, EPIRKW3 shrinks both the stepsize and subspace size, and takes longer than EPIRKK4 [$M = 100$] methods to compute the solution.

The new methods (both W and K) are also significantly faster than MATLAB's built-in onestep method to tackle stiff problems (ODE23S), when computing medium-high accuracy solutions for this setup.

Remark 2 Earlier, we observed that EPIRKK4 with [$M = 100$] is stability bounded with the integrator taking about the same number of steps for different tolerance settings. However, we would like to point out that the order of EPIRKK4 method is partially recovered by increasing the subspace dimension—see Fig. 3, where we increased the Krylov-subspace size to 180 vectors and we see EPIRKK4A beginning to take different sized steps for tight tolerances.

Remark 3 The inexpensive Krylov-subspace iterations, while using Lanczos for symmetric problems, results in competitive CPU times for classical methods when compared against W- and K-methods (see Fig. 2). If the Jacobian of the problem is not symmetric or if Arnoldi iterations are used to compute the Krylov-subspace, and the problem is large, then the construction of large Krylov-subspaces, as the classical methods do, will dominate the total cost of time-stepping. In additional numerical experiments not reported here, we observed this to be the case when we used Arnoldi iterations to compute the Krylov-subspace for the Allen–Cahn problem (31). Both the classical methods—EPIRKK4- CLASSICAL and EPIRK5P1BVAR—were half-order of magnitude slower than EPIRKK4 and EPIRKW3 methods, while taking fewer steps.

6.3 BSVD: Bistable Equation with Space-Dependent Diffusion

BSVD [11] is a reaction–diffusion PDE:

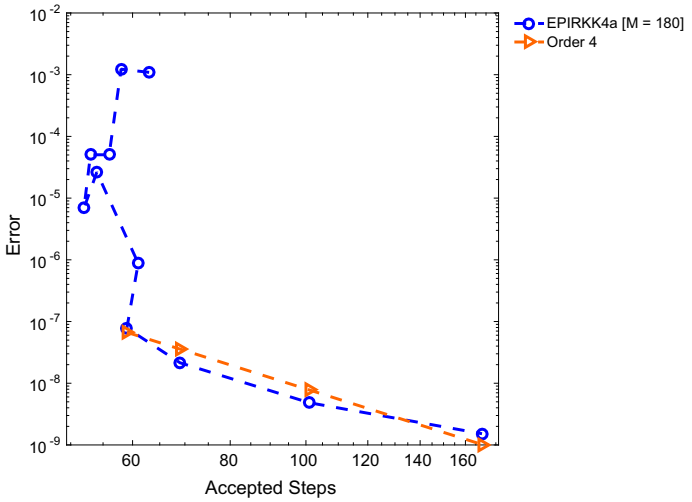


Fig. 3 Convergence diagram of EPIRKK4A on the Allen–Cahn problem (31) when increasing the Krylov-subspace size to 180 vectors

$$\frac{\partial u}{\partial t} = \nabla \cdot (D(x, y)\nabla u) + 10\,(1 - u^2)(u + 0.6), \quad (x, y) \in [0, 1] \times [0, 1] \text{ (space units)},$$
$$t \in [0, 7] \text{ (time units)}, \tag{32}$$

where space-dependent diffusion coeffcient is defined as:

$$D(x, y) = \frac{1}{10} \sum_{i=1}^{3} e^{-100((x-0.5)^2+(y-y_i)^2)},$$

and the initial condition is given by:

$$u(x, y, 0) = 2e^{-10((x-0.5)^2+(y+0.1)^2)} - 1.$$

y_i for $i = 1, 2, 3$ are $y_1 = 0.6$, $y_2 = 0.75$ and $y_3 = 0.9$. We discretize the PDE using second-order finite difference on a 150×300 grid, while imposing homogeneous Neumann boundary conditions

$$D(x, y)\hat{\mathbf{n}}(x, y).\nabla u(x, y, t) = 0,$$

using ghost nodes at the boundary. $\hat{\mathbf{n}}(x, y)$ is the normal vector pointing outwards at the boundary. The problem is implemented in the Computational Science Lab's ODE Test Problems [22] currently under active development.

The Jacobian of BSVD equation (32) is nonsymmetric and the methods that we test use Arnoldi to compute the Krylov-subspace. Figure 4 shows both the work-precision and convergence diagram for this study.

6.3.1 Observations

ODE45 needed $\mathcal{O}(10^5)$ steps to compute the solution and filled up the system memory on a 64GB machine. We had to exclude it from our study as the operating system was thrashing. DOPRI5, available through `MATLODE`, didn't converge to a solution.

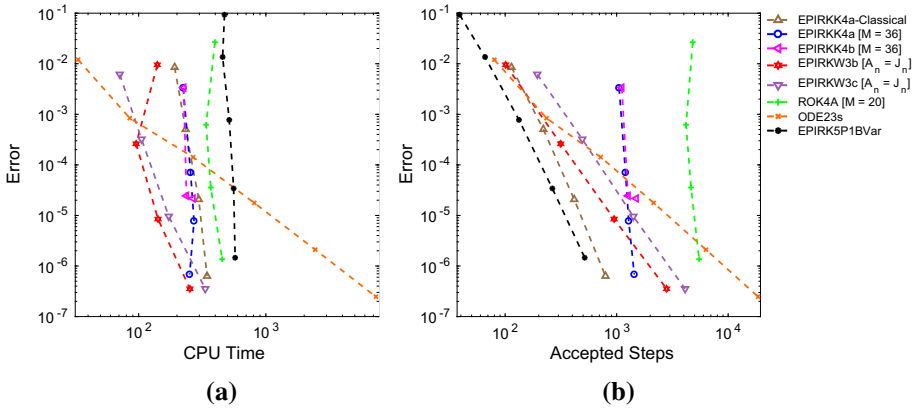


Fig. 4 Comparison of different integrators when applied to BSVD system (32) on a 150 × 300 grid. (using Arnoldi to compute Krylov subspace). **a** Work precision diagram. **b** Convergence diagram

Table 12 Root mean square number of Krylov vectors per projection for each integrator applied to BSVD equation on a 150 × 300 grid (32)

Integrator	Tolerance					
	1e−1	3.98e−3	1.58e−4	6.31e−6	2.51e−7	1e−8
EPIRKK4-classical	248	161	77	62	51	39
EPIRKK4a [M >= 20]	–	27	27	27	29	30
EPIRKK4a [M = 36]	–	36	36	36	36	36
EPIRKK4b [M = 36]	36	36	36	36	36	–
EPIRKW3b [$A_n = J_n$]	88	40	60	28	16	10
EPIRKW3c [$A_n = J_n$]	43	38	29	22	13	8
ROK4A [M = 20]	–	–	20	20	20	20
ROK4A [M = 36]	–	36	36	36	36	36
EPIRK5P1BVar	–	1177	946	656	472	351

Timesteps of K-methods—both EPIRKK4 and ROK4A—are bounded by stability considerations, which is evident from the nearly vertical lines on the convergence diagram 4. These methods also reject a number of timesteps, and this may have to do with the single projection that the K-methods do internally, which likely affects the stepsize; whereas, all other methods (excluding ODE23S) perform three Arnoldi projections per timestep. EPIRKK4 is more efficient than ROK4A [32], a Rosenbrock–Krylov method built using similar principles as EPIRKK4. Note that ROK4A [$M = 20$] performs better than ROK4A [$M = 36$]; whereas, EPIRKK4 [$M = 36$] was the best performing among different subspace sizes that we tested on, including EPIRKK4 [$M = 20$].

EPIRKW3 is the best performing amongst all methods under study. It admits Jacobian approximations making it computationally cheap per timestep; in comparison with classical methods—EPIRKK4A- CLASSICAL and EPIRK5P1BVAR—EPIRKW3 needs fewer vectors in the Krylov-subspace (see Table 12), besides being capped at a maximum of 100 vectors in our experiments. It takes larger timesteps than EPIRKK4 methods, and is not bounded by stability considerations unlike K-methods.

The cost of building large subspaces by the classical exponential method, EPIRK5P1BVAR, is not offset by the large steps that it takes. As a result, EPIRK5P1BVAR performs poorly in comparison to all other exponential integrators. Furthermore, we can observe that building large subspaces using Arnoldi iterations considerably affects the timing of EPIRK5P1BVAR, where the method takes at least twice as much time as EPIRKW3 and EPIRKK4 methods to compute the solution. Compare this to the Allen–Cahn problem, where the integrators used Lanczos iteration to compute the Krylov-subspace and the performance of EPIRK5P1BVAR was about the same as all other methods (see Fig. 2).

Clearly, all the methods perform significantly better than the stiff one-step integrator built inside MATLAB—ODE23S.

7 Conclusions

Exponential Propagation Iterative Methods of Runge–Kutta type (EPIRK) rely on the computation of exponential-like matrix functions of the Jacobian times vector products. This paper develops two new classes of EPIRK methods that allow the use of inexact Jacobians as arguments of the matrix functions. We derive a general order conditions theory for EPIRK-W methods, which admit arbitrary approximations of the Jacobian. We also derive a similar theory for EPIRK-K methods, which uses a specific Krylov-subspace based approximation of the Jacobian. A computational procedure to derive order conditions for these methods with an arbitrary number of stages is provided, and the order conditions of a three stage formulation is solved to obtain coefficients for three third order EPIRK-W-methods, named EPIRKW3 (A, B and C), and two fourth order EPIRK-K-method, named EPIRKK4 (A and B). Furthermore, several alternative implementations of W-methods and K-methods are discussed, and their properties are studied.

Numerical experiments are conducted with three different test problems to study the performance of the new methods. The results confirm empirically that EPIRKW3 method retains third order of convergence with different Jacobian approximations. The EPIRKW3 method is computationally more efficient than EPIRKK4, for stiff problems, and performs better than classical methods in a number of different scenarios considered in the paper. In particular EPIRKK4 outperforms EPIRKK4- CLASSICAL, a method with the same set of coefficients, but implemented in the classical framework for exponential integrators; EPIRKK4 also outperforms EPIRK5P1BVAR, a classical exponential integrator, and ROK4A a Rosenbrock–Krylov method built using the same principles as EPIRKK4. More numerical experiments are needed to assess how these results generalize to different applications.

It was also observed that increasing the basis size can make EPIRKK4 more stable, but there is a cost to pay for it and that it is important to balance gains in stability against the increased computational cost. Adaptive approach to basis size selection for the K-methods is relevant in this regard and will be considered again in a future work.

Acknowledgements This work has been supported in part by NSF through Awards NSF DMS-1419003, NSF DMS-1419105, NSF CCF-1613905, by AFOSR through Award AFOSR FA9550-12-1-0293-DEF, and by the Computational Science Laboratory at Virginia Tech.

A Derivation of *K*-Methods

The EPIRKK method is derived from EPIRKW method, where a specific Krylov-subspace approximation of the Jacobian is used instead of an arbitrary approximate Jacobian as admitted by the W-method. We start the derivation by first stating the general form of the EPIRKW method:

$$\begin{aligned}
\mathbf{Y}_i &= \mathbf{y}_n + a_{i,1}\,\boldsymbol{\psi}_{i,1}(g_{i,1}\,h\,\mathbf{A}_n)\,h\mathbf{f}(\mathbf{y}_n) + \sum_{j=2}^{i} a_{i,j}\,\boldsymbol{\psi}_{i,j}(g_{i,j}\,h\,\mathbf{A}_n)\,h\Delta^{(j-1)}\mathbf{r}(\mathbf{y}_n),\\
&\quad i = 1,\ldots,s-1,\\
\mathbf{y}_{n+1} &= \mathbf{y}_n + b_1\,\boldsymbol{\psi}_{s,1}(g_{s,1}\,h\,\mathbf{A}_n)\,h\mathbf{f}(\mathbf{y}_n) + \sum_{j=2}^{s} b_j\,\boldsymbol{\psi}_{s,j}(g_{s,j}\,h\,\mathbf{A}_n)\,h\Delta^{(j-1)}\mathbf{r}(\mathbf{y}_n).
\end{aligned}$$

In the above equation, the $\boldsymbol{\psi}$ function is as defined in Eq. (8) and the following simplifying assumption is made about it:

$$\boldsymbol{\psi}_{i,j}(z) = \boldsymbol{\psi}_j(z) = \sum_{k=1}^{j} p_{j,k}\,\boldsymbol{\varphi}_k(z),$$

where $\boldsymbol{\varphi}$ function is as defined in Eqs. (5) and (6). Additionally, the remainder function ($\mathbf{r}(\mathbf{y})$) and the forward difference operator ($\Delta^{(j)}\mathbf{r}(\mathbf{Y}_i)$) are defined accordingly below:

$$\begin{aligned}
\mathbf{r}(\mathbf{y}) &= \mathbf{f}(\mathbf{y}) - \mathbf{f}(\mathbf{y}_n) - \mathbf{A}_n\,(\mathbf{y}-\mathbf{y}_n),\\
\Delta^{(j)}\mathbf{r}(\mathbf{Y}_i) &= \Delta^{(j-1)}\mathbf{r}(\mathbf{Y}_{i+1}) - \Delta^{(j-1)}\mathbf{r}(\mathbf{Y}_i),\\
\Delta^{(1)}\mathbf{r}(\mathbf{Y}_i) &= \mathbf{r}(\mathbf{Y}_{i+1}) - \mathbf{r}(\mathbf{Y}_i).
\end{aligned} \tag{33}$$

The K-method uses a specific Krylov-subspace based approximation of the Jacobian ($\mathbf{J}_n$). An M-dimensional Krylov-subspace is built as,

$$\mathcal{K}_M = \text{span}\{\mathbf{f}_n, \mathbf{J}_n\mathbf{f}_n, \mathbf{J}_n^2\mathbf{f}_n, \ldots, \mathbf{J}_n^{M-1}\mathbf{f}_n\}, \tag{34}$$

whose basis is the orthonormal matrix $\mathbf{V}$ and $\mathbf{H}$ is the upper-Hessenberg matrix obtained from Arnoldi iteration defined as

$$\mathbf{H} = \mathbf{V}^T\mathbf{J}_n\,\mathbf{V}. \tag{35}$$

The corresponding Krylov-subspace based approximation of the Jacobian is built as

$$\mathbf{A}_n = \mathbf{V}\,\mathbf{H}\,\mathbf{V}^T = \mathbf{V}\mathbf{V}^T\mathbf{J}_n\,\mathbf{V}\mathbf{V}^T. \tag{36}$$

The use of Krylov-subspace based approximation of the Jacobian reduces the $\boldsymbol{\varphi}$ and $\boldsymbol{\psi}$ function in accordance with Lemmas 2 and 3 to the following

$$\boldsymbol{\varphi}_k(h\,\gamma\,\mathbf{A}_n) = \frac{1}{k!}\left(\mathbf{I} - \mathbf{V}\mathbf{V}^T\right) + \mathbf{V}\boldsymbol{\varphi}_k(h\,\gamma\,\mathbf{H})\mathbf{V}^T, \tag{37}$$

$$\boldsymbol{\psi}_j(h\gamma\mathbf{A}_n) = \widetilde{p}_j\left(\mathbf{I} - \mathbf{V}\mathbf{V}^T\right) + \mathbf{V}\,\boldsymbol{\psi}_j(h\gamma\mathbf{H})\mathbf{V}^T, \tag{38}$$

where $\widetilde{p}_j$ is defined as

$$\widetilde{p}_j = \sum_{k=1}^{j}\frac{p_{j,k}}{k!}. \tag{39}$$

In order to derive the reduced stage formulation of the EPIRKK method we need to resolve the vectors in the formulation into components in the Krylov-subspace and orthogonal to it. Repeating the splittings from the main text:

– Splitting the internal stage vectors noting that $\mathbf{Y}_0 \equiv \mathbf{y}_n$:

$$\mathbf{Y}_i = \mathbf{V}\boldsymbol{\lambda}_i + \mathbf{Y}_i^{\perp} \quad \text{where} \quad \mathbf{V}^T\mathbf{Y}_i = \boldsymbol{\lambda}_i, \quad \left(\mathbf{I} - \mathbf{V}\mathbf{V}^T\right)\mathbf{Y}_i = \mathbf{Y}_i^{\perp}. \tag{40}$$

– Splitting the right-hand side function evaluated at internal stage vectors while noting that $\mathbf{f}_0 \equiv \mathbf{f}(\mathbf{y}_n)$:

$$\mathbf{f}_i := \mathbf{f}(\mathbf{Y}_i) = \mathbf{V}\boldsymbol{\eta}_i + \mathbf{f}_i^{\perp} \quad \text{where} \quad \mathbf{V}^T\mathbf{f}_i = \boldsymbol{\eta}_i, \quad \left(\mathbf{I} - \mathbf{V}\mathbf{V}^T\right)\mathbf{f}_i = \mathbf{f}_i^{\perp}. \tag{41}$$

– Splitting the non-linear Taylor remainder of the right-hand side functions:

$$\begin{aligned}
&\mathbf{r}(\mathbf{Y}_i) = \mathbf{f}(\mathbf{Y}_i) - \mathbf{f}(\mathbf{y}_n) - \mathbf{A}_n\,(\mathbf{Y}_i - \mathbf{y}_n) = \mathbf{f}_i - \mathbf{f}_0 - \mathbf{V}\,\mathbf{H}\,\mathbf{V}^T\,(\mathbf{Y}_i - \mathbf{y}_n),\\
&\quad \text{where} \quad \mathbf{V}^T\,\mathbf{r}(\mathbf{Y}_i) = \boldsymbol{\eta}_i - \boldsymbol{\eta}_0 - \mathbf{H}\,(\boldsymbol{\lambda}_i - \boldsymbol{\lambda}_0),\\
&\quad \left(\mathbf{I} - \mathbf{V}\mathbf{V}^T\right)\,\mathbf{r}(\mathbf{Y}_i) = \mathbf{f}_i^{\perp} - \mathbf{f}_0^{\perp}.
\end{aligned} \tag{42}$$

– Splitting the forward differences of the non-linear remainder terms:

$$\begin{aligned}
&\widetilde{\mathbf{r}}_{(j-1)} := \Delta^{(j-1)}\mathbf{r}(\mathbf{y}_n) = \mathbf{V}\,\mathbf{d}_{(j-1)} + \widetilde{\mathbf{r}}_{(j-1)}^{\perp},\\
&\quad \text{where} \quad \mathbf{V}^T\,\widetilde{\mathbf{r}}_{(j-1)} = \mathbf{d}_{(j-1)},\\
&\quad \left(\mathbf{I} - \mathbf{V}\mathbf{V}^T\right)\,\widetilde{\mathbf{r}}_{(j-1)} = \widetilde{\mathbf{r}}_{(j-1)}^{\perp}.
\end{aligned} \tag{43}$$

Using these each internal stage of the EPIRKK method can be expressed as below:

$$\begin{aligned}
\mathbf{V}\boldsymbol{\lambda}_i + \mathbf{Y}_i^{\perp} &= \mathbf{y}_n + h\,a_{i,1}\left(\widetilde{p}_1\left(\mathbf{I} - \mathbf{V}\mathbf{V}^T\right) + \mathbf{V}\,\boldsymbol{\psi}_1(h\,g_{i,1}\,\mathbf{H})\mathbf{V}^T\right)\left(\mathbf{V}\boldsymbol{\eta}_0 + \mathbf{f}_0^{\perp}\right)\\
&\quad + \sum_{j=2}^{i} h\,a_{i,j}\left(\widetilde{p}_j\left(\mathbf{I} - \mathbf{V}\mathbf{V}^T\right) + \mathbf{V}\,\boldsymbol{\psi}_j(h\,g_{i,j}\,\mathbf{H})\mathbf{V}^T\right)\left(\mathbf{V}\mathbf{d}_{(j-1)} + \widetilde{\mathbf{r}}_{(j-1)}^{\perp}\right)\\
&= \mathbf{y}_n + h\,a_{i,1}\left(\widetilde{p}_1\,\mathbf{f}_0^{\perp} + \mathbf{V}\,\boldsymbol{\psi}_1(h\,g_{i,1}\,\mathbf{H})\boldsymbol{\eta}_0\right)\\
&\quad + \sum_{j=2}^{i} h\,a_{i,j}\left(\widetilde{p}_j\,\widetilde{\mathbf{r}}_{(j-1)}^{\perp} + \mathbf{V}\,\boldsymbol{\psi}_j(h\,g_{i,j}\,\mathbf{H})\,\mathbf{d}_{(j-1)}\right).
\end{aligned} \tag{44}$$

The reduced stage formulation of the EPIRKK method is obtained by multiplying the above equation by $\mathbf{V}^T$ from the left

$$\begin{aligned}
\boldsymbol{\lambda}_i &= \mathbf{V}^T\mathbf{y}_n + h\,a_{i,1}\boldsymbol{\psi}_1(h\,g_{i,1}\,\mathbf{H})\boldsymbol{\eta}_0 + \sum_{j=2}^{i} h\,a_{i,j}\,\boldsymbol{\psi}_j(h\,g_{i,j}\,\mathbf{H})\,\mathbf{d}_{(j-1)}\\
&= \boldsymbol{\lambda}_0 + h\,a_{i,1}\boldsymbol{\psi}_1(h\,g_{i,1}\,\mathbf{H})\boldsymbol{\eta}_0 + \sum_{j=2}^{i} h\,a_{i,j}\,\boldsymbol{\psi}_j(h\,g_{i,j}\,\mathbf{H})\,\mathbf{d}_{(j-1)}.
\end{aligned} \tag{45}$$

And the full stage vector can be recovered by first computing the reduced stage vector and adding the orthogonal piece $\mathbf{Y}_i^{\perp}$ obtained when multiplying Eq. (44) by $\left(\mathbf{I}-\mathbf{V}\mathbf{V}^T\right)$

$$\begin{aligned}\mathbf{Y}_i^{\perp} &= \left(\mathbf{I}-\mathbf{V}\mathbf{V}^T\right)\mathbf{y}_n + h\,a_{i,1}\,\widetilde{p}_1\,\mathbf{f}_0^{\perp} + \sum_{j=2}^{i} h\,a_{i,j}\,\widetilde{p}_j\,\widetilde{\mathbf{r}}_{(j-1)}^{\perp} \\ &= (\mathbf{y}_n - \mathbf{V}\boldsymbol{\lambda}_0) + h\,a_{i,1}\,\widetilde{p}_1\,\mathbf{f}_0^{\perp} + \sum_{j=2}^{i} h\,a_{i,j}\,\widetilde{p}_j\,\widetilde{\mathbf{r}}_{(j-1)}^{\perp}. \end{aligned} \tag{46}$$

The final stage can also be written in the above form with multipliers b_i in place of a_{ij}. Notice that the expensive computations are performed in the reduced space, i.e. the $\boldsymbol{\psi}$ function is computed in the reduced space instead of the full space offering potential computational savings. In the above equations, the quantities $\mathbf{d}_{(j-1)}$ and $\widetilde{\mathbf{r}}_{(j-1)}^{\perp}$ can be shown to be

$$\mathbf{d}_{(j-1)} = \sum_{k=0}^{j-1}\left((-1)^k\binom{j-1}{k}\boldsymbol{\eta}_{j-1-k} - \mathbf{H}\left((-1)^k\binom{j-1}{k}\boldsymbol{\lambda}_{j-1-k}\right)\right), \tag{47}$$

$$\widetilde{\mathbf{r}}_{(j-1)}^{\perp} = \sum_{k=0}^{j-1}\left((-1)^k\binom{j-1}{k}\mathbf{f}_{j-1-k}^{\perp}\right), \tag{48}$$

as is done in the following "Appendix".

B Proofs

In order to prove Eqs. (23a) and (23b), we start with the definition of the remainder function and forward difference.

$$\mathbf{r}(\mathbf{y}) = \mathbf{f}(\mathbf{y}) - \mathbf{f}(\mathbf{y}_n) - \mathbf{A}_n(\mathbf{y}-\mathbf{y}_n), \tag{49}$$

$$\Delta^{(j)}\mathbf{r}(\mathbf{Y}_i) = \Delta^{(j-1)}\mathbf{r}(\mathbf{Y}_{i+1}) - \Delta^{(j-1)}\mathbf{r}(\mathbf{Y}_i), \tag{50a}$$

$$\Delta^{(1)}\mathbf{r}(\mathbf{Y}_i) = \mathbf{r}(\mathbf{Y}_{i+1}) - \mathbf{r}(\mathbf{Y}_i). \tag{50b}$$

Lemma 4 $\Delta^{(j)}\mathbf{r}(\mathbf{Y}_i) = \sum_{k=0}^{j}(-1)^k\binom{j}{k}\mathbf{r}(\mathbf{Y}_{i+j-k})$.

Proof In order to prove the lemma, we resort to mathematical induction. Base case $j=1$,

$$\begin{aligned}\Delta^{(1)}\mathbf{r}(\mathbf{Y}_i) &= \sum_{k=0}^{1}(-1)^k\binom{1}{k}\mathbf{r}(\mathbf{Y}_{i+1-k}) \\ &= \mathbf{r}(\mathbf{Y}_{i+1}) - \mathbf{r}(\mathbf{Y}_i).\end{aligned}$$

The base case is true by definition. We now assume that the proposition holds true for all j up to $k-1$. We have,

$$\Delta^{(k-1)}\mathbf{r}(\mathbf{Y}_i) = \sum_{l=0}^{k-1}(-1)^l\binom{k-1}{l}\mathbf{r}(\mathbf{Y}_{i+k-1-l}), \tag{51}$$

$$\Delta^{(k-1)}\mathbf{r}(\mathbf{Y}_{i+1}) = \sum_{l=0}^{k-1}(-1)^l\binom{k-1}{l}\mathbf{r}(\mathbf{Y}_{i+k-l}). \tag{52}$$

Then for $j = k$,

$$\begin{aligned}
\Delta^{(k)}\mathbf{r}(\mathbf{Y}_i) &= \Delta^{(k-1)}\mathbf{r}(\mathbf{Y}_{i+1}) - \Delta^{(k-1)}\mathbf{r}(\mathbf{Y}_i)\\
&= \sum_{l=0}^{k-1}(-1)^l\binom{k-1}{l}\mathbf{r}(\mathbf{Y}_{i+k-l}) - \sum_{l=0}^{k-1}(-1)^l\binom{k-1}{l}\mathbf{r}(\mathbf{Y}_{i+k-1-l})\\
&= \mathbf{r}(\mathbf{Y}_{i+k}) + \sum_{l=1}^{k-1}(-1)^l\binom{k-1}{l}\mathbf{r}(\mathbf{Y}_{i+k-l})\\
&\quad - \sum_{l=0}^{k-2}(-1)^l\binom{k-1}{l}\mathbf{r}(\mathbf{Y}_{i+k-1-l}) + (-1)^k\mathbf{r}(\mathbf{Y}_i).
\end{aligned}$$

We perform a change of variable for the second summation, $m = l+1 \implies l = (m-1)$,

$$\begin{aligned}
\Delta^{(k)}\mathbf{r}(\mathbf{Y}_i) &= \mathbf{r}(\mathbf{Y}_{i+k}) + \sum_{l=1}^{k-1}(-1)^l\binom{k-1}{l}\mathbf{r}(\mathbf{Y}_{i+k-l})\\
&\quad - \sum_{m=1}^{k-1}(-1)^{m-1}\binom{k-1}{m-1}\mathbf{r}(\mathbf{Y}_{i+k-m}) + (-1)^k\mathbf{r}(\mathbf{Y}_i)\\
&= \mathbf{r}(\mathbf{Y}_{i+k}) + \sum_{l=1}^{k-1}(-1)^l\binom{k-1}{l}\mathbf{r}(\mathbf{Y}_{i+k-l})\\
&\quad + \sum_{m=1}^{k-1}(-1)^m\binom{k-1}{m-1}\mathbf{r}(\mathbf{Y}_{i+k-m}) + (-1)^k\mathbf{r}(\mathbf{Y}_i).
\end{aligned}$$

The summations run between the same start and end indices, and can be collapsed.

$$\Delta^{(k)}\mathbf{r}(\mathbf{Y}_i) = \mathbf{r}(\mathbf{Y}_{i+k}) + \sum_{l=1}^{k-1}(-1)^l\left(\binom{k-1}{l} + \binom{k-1}{l-1}\right)\mathbf{r}(\mathbf{Y}_{i+k-l}) + (-1)^k\mathbf{r}(\mathbf{Y}_i).$$

We use the identity,

$$\binom{k-1}{l} + \binom{k-1}{l-1} = \binom{k}{l},$$

and arrive at the desired result,

$$\begin{aligned}
\Delta^{(k)}\mathbf{r}(\mathbf{Y}_i) &= \mathbf{r}(\mathbf{Y}_{i+k}) + \sum_{l=1}^{k-1}(-1)^l\binom{k}{l}\mathbf{r}(\mathbf{Y}_{i+k-l}) + (-1)^k\mathbf{r}(\mathbf{Y}_i)\\
&= (-1)^0\binom{k}{0}\mathbf{r}(\mathbf{Y}_{i+k}) + \sum_{l=1}^{k-1}(-1)^l\binom{k}{l}\mathbf{r}(\mathbf{Y}_{i+k-l}) + (-1)^k\binom{k}{k}\mathbf{r}(\mathbf{Y}_i)\\
&= \sum_{l=0}^{k}(-1)^l\binom{k}{l}\mathbf{r}(\mathbf{Y}_{i+k-l}).
\end{aligned} \tag{53}$$

□

Lemma 5 *Given*

$$\Delta^{(j-1)}\mathbf{r}(\mathbf{y}_n) = \mathbf{V}\mathbf{d}_{(j-1)} + \widetilde{\mathbf{r}}^{\perp}_{(j-1)}, \tag{54}$$

we need to prove

$$\mathbf{d}_{(j-1)} = \sum_{k=0}^{j-1}\left((-1)^k\binom{j-1}{k}\boldsymbol{\eta}_{j-1-k} - \mathbf{H}\left((-1)^k\binom{j-1}{k}\boldsymbol{\lambda}_{j-1-k}\right)\right), \tag{55}$$

$$\widetilde{\mathbf{r}}^{\perp}_{(j-1)} = \sum_{k=0}^{j-1}\left((-1)^k\binom{j-1}{k}\mathbf{f}^{\perp}_{j-1-k}\right). \tag{56}$$

Proof We start with Lemma 4 where we have proven that

$$\Delta^{(j)}\mathbf{r}(\mathbf{Y}_i) = \sum_{k=0}^{j}(-1)^k\binom{j}{k}\mathbf{r}(\mathbf{Y}_{i+j-k}).$$

We plugin the value $i = 0$ which corresponds to $\Delta^{(j)}\mathbf{r}(\mathbf{Y}_0) \equiv \Delta^{(j)}\mathbf{r}(\mathbf{y}_n)$ and we get,

$$\Delta^{(j)}\mathbf{r}(\mathbf{y}_n) = \sum_{k=0}^{j}(-1)^k\binom{j}{k}\mathbf{r}(\mathbf{Y}_{j-k}).$$

WLOG replacing j by $j-1$ yields

$$\Delta^{(j-1)}\mathbf{r}(\mathbf{y}_n) = \sum_{k=0}^{j-1}(-1)^k\binom{j-1}{k}\mathbf{r}(\mathbf{Y}_{j-1-k}). \tag{57}$$

Since the left-hand side of Eq. (57) can be written as

$$\Delta^{(j-1)}\mathbf{r}(\mathbf{y}_n) = \mathbf{V}\mathbf{d}_{(j-1)} + \widetilde{\mathbf{r}}^{\perp}_{(j-1)},$$

we have the following result

$$\mathbf{V}\mathbf{d}_{(j-1)} + \widetilde{\mathbf{r}}^{\perp}_{(j-1)} = \sum_{k=0}^{j-1}(-1)^k\binom{j-1}{k}\mathbf{r}(\mathbf{Y}_{j-1-k}). \tag{58}$$

The remainder function $\mathbf{r}(\mathbf{Y}_i)$ is defined as

$$\mathbf{r}(\mathbf{Y}_i) = \mathbf{f}(\mathbf{Y}_i) - \mathbf{f}(\mathbf{y}_n) - \mathbf{A}_n * (\mathbf{Y}_i - \mathbf{y}_n).$$

Plugging in the definition of remainder function in (58) and observing that $\mathbf{r}(\mathbf{y}_n) = 0$, we get

$$\begin{aligned}\mathbf{V}\mathbf{d}_{(j-1)} + \widetilde{\mathbf{r}}^{\perp}_{(j-1)} &= \sum_{k=0}^{j-2}(-1)^k\binom{j-1}{k}\left(\mathbf{f}(\mathbf{Y}_{j-1-k}) - \mathbf{f}(\mathbf{y}_n) - \mathbf{A}_n * \left(\mathbf{Y}_{j-1-k} - \mathbf{y}_n\right)\right)\\ &= \sum_{k=0}^{j-2}(-1)^k\binom{j-1}{k}\left(\mathbf{f}(\mathbf{Y}_{j-1-k}) - \mathbf{A}_n * \mathbf{Y}_{j-1-k}\right)\\ &\quad - \sum_{k=0}^{j-2}(-1)^k\binom{j-1}{k}\left(\mathbf{f}(\mathbf{y}_n) - \mathbf{A}_n * \mathbf{y}_n\right).\end{aligned} \tag{59}$$

Consider the identity involving alternating sum and difference of binomial coefficients,

$$\sum_{i=0}^{k}(-1)^i\binom{k}{i}=0. \tag{60}$$

Applying (60) to (59) we get,

$$\begin{aligned}
\mathbf{Vd}_{(j-1)}+\widetilde{\mathbf{r}}^{\perp}_{(j-1)} &= \sum_{k=0}^{j-2}(-1)^k\binom{j-1}{k}\Big(\mathbf{f}(\mathbf{Y}_{j-1-k})-\mathbf{f}(\mathbf{y}_n)-\mathbf{A}_n*\Big(\mathbf{Y}_{j-1-k}-\mathbf{y}_n\Big)\Big)\\
&= \sum_{k=0}^{j-2}(-1)^k\binom{j-1}{k}\Big(\mathbf{f}(\mathbf{Y}_{j-1-k})-\mathbf{A}_n*\mathbf{Y}_{j-1-k}\Big)\\
&\quad -(-(-1)^{(j-1)})\Big(\mathbf{f}(\mathbf{y}_n)-\mathbf{A}_n*\mathbf{y}_n\Big)\\
&= \sum_{k=0}^{j-2}(-1)^k\binom{j-1}{k}\Big(\mathbf{f}(\mathbf{Y}_{j-1-k})-\mathbf{A}_n*\mathbf{Y}_{j-1-k}\Big)\\
&\quad +(-1)^{(j-1)}\binom{j-1}{j-1}\Big(\mathbf{f}(\mathbf{y}_n)-\mathbf{A}_n*\mathbf{y}_n\Big)\\
&= \sum_{k=0}^{j-1}(-1)^k\binom{j-1}{k}\Big(\mathbf{f}(\mathbf{Y}_{j-1-k})-\mathbf{A}_n*\mathbf{Y}_{j-1-k}\Big).
\end{aligned} \tag{61}$$

Additionally, since we are replacing the Jacobian by the approximation in the Krylov-subspace, i.e. $\mathbf{A}_n=\mathbf{V}\,\mathbf{H}\,\mathbf{V}^T$ we have

$$\mathbf{Vd}_{(j-1)}+\widetilde{\mathbf{r}}^{\perp}_{(j-1)}=\sum_{k=0}^{j-1}(-1)^k\binom{j-1}{k}\Big(\mathbf{f}(\mathbf{Y}_{j-1-k})-\mathbf{VHV}^T*\mathbf{Y}_{j-1-k}\Big). \tag{62}$$

We get the expression for $\mathbf{d}_{(j-1)}$ by multiplying Eq. (62) from the left by V^T and that for $\widetilde{\mathbf{r}}^{\perp}_{(j-1)}$ by multiplying by $(\mathbf{I}-\mathbf{VV}^T)$. □

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**(2), 488–511 (2011)
2. Allen, S.M., Cahn, J.W.: A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening. Acta Metall. **27**(6), 1085–1095 (1979). https://doi.org/10.1016/0001-6160(79)90196-2
3. Berland, H., Owren, B., Skaflestad, B.: B-series and order conditions for exponential integrators. SIAM J. Numer. Anal. **43**(4), 1715–1727 (2005)
4. Berland, H., Skaflestad, B., Wright, W.M.: Expint—a matlab package for exponential integrators. ACM Trans. Math. Softw. (TOMS) **33**(1), 4 (2007)
5. Butcher, J.: Numerical Methods for Ordinary Differential Equations, 2nd edn. Wiley, New York (2008). https://doi.org/10.1002/9780470753767
6. Butcher, J.: Trees, B-series and exponential integrators. IMA J. Numer. Anal. **30**, 131–140 (2010)
7. Caliari, M., Ostermann, A.: Implementation of exponential rosenbrock-type integrators. Appl. Numer. Math. **59**(3–4), 568–581 (2009)
8. D'Augustine, A.F.: MATLODE: A MATLAB ODE solver and sensitivity analysis toolbox. Masters Thesis, Virginia Tech (2018)

9. Hairer, E., Norsett, S., Wanner, G.: Solving Ordinary Differential Equations I: Nonstiff Problems. Springer Series in Computational Mathematics, vol. 8. Springer, Berlin (1993)
10. Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems. Springer Series in Computational Mathematics, vol. 14, 2nd edn. Springer, Berlin (1996)
11. Heineken, W., Warnecke, G.: Partitioning methods for reaction–diffusion problems. Appl. Numer. Math. **56**(7), 981–1000 (2006)
12. Hochbruck, M., Lubich, C.: On Krylov subspace approximations to the matrix exponential operator. SIAM J. Numer. Anal. **34**(5), 1911–1925 (1997)
13. Hochbruck, M., Lubich, C., Selhofer, H.: Exponential integrators for large systems of differential equations. SIAM J. Sci. Comput. **19**(5), 1552–1574 (1998)
14. Hochbruck, M., Ostermann, A.: Explicit exponential Runge–Kutta methods for semilinear parabolic problems. SIAM J. Numer. Anal. **43**(3), 1069–1090 (2005)
15. Hochbruck, M., Ostermann, A.: Exponential integrators. Acta Numer. **19**, 209–286 (2012)
16. Loffeld, J., Tokman, M.: Comparative performance of exponential, implicit, and explicit integrators for stiff systems of odes. J. Comput. Appl. Math. **241**, 45–67 (2013)
17. Lorenz, E.N.: Predictability—a problem partly solved. In: Palmer, T., Hagedorn, R. (eds.) Predictability of Weather and Climate, pp. 40–58. Cambridge University Press (CUP), Cambridge (1996). https://doi.org/10.1017/cbo9780511617652.004
18. Lu, Y.Y.: Computing a matrix function for exponential integrators. J. Comput. Appl. Math. **161**(1), 203–216 (2003)
19. Minchev, B.V., Wright, W.: A review of exponential integrators for first order semi-linear problems. Technical report (2005)
20. Niesen, J., Wright, W.M.: Algorithm 919: a krylov subspace algorithm for evaluating the φ-functions appearing in exponential integrators. ACM Trans. Math. Softw. **38**(3), 22:1–22:19 (2012). https://doi.org/10.1145/2168773.2168781
21. Rainwater, G., Tokman, M.: A new approach to constructing efficient stiffly accurate Epirk methods. J. Comput. Phys. **323**, 283–309 (2016). https://doi.org/10.1016/j.jcp.2016.07.026
22. Roberts, S., Popov, A., Sandu, A.: ODE Test Problems. https://sibiu.cs.vt.edu/ode-test-problems/index.html. Accessed 16 Dec 2017
23. Saad, Y.: Iterative Methods for Sparse Linear Systems. SIAM, Philadelphia (2003)
24. Schulze, J.C., Schmid, P.J., Sesterhenn, J.L.: Exponential time integration using Krylov subspaces. Int. J. Numer. Meth. Fluids **60**(6), 561–609 (2008)
25. Sidje, R.B.: Expokit: a software package for computing matrix exponentials. ACM Trans. Math. Softw. **24**(1), 130–156 (1998). https://doi.org/10.1145/285861.285868
26. Steihaug, T., Wolfbrandt, A.: An attempt to avoid exact Jacobian and nonlinear equations in the numerical solution of stiff differential equations. Math. Comput. **33**(146), 521–521 (1979). https://doi.org/10.1090/s0025-5718-1979-0521273-8
27. Tokman, M.: Efficient integration of large stiff systems of ODEs with exponential propagation iterative (EPI) methods. J. Comput. Phys. **213**(2), 748–776 (2006)
28. Tokman, M.: A new class of exponential propagation iterative methods of Runge–Kutta type (EPIRK). J. Comput. Phys. **230**, 8762–8778 (2011)
29. Tokman, M., Loffeld, J., Tranquilli, P.: New adaptive exponential propagation iterative methods of Runge–Kutta type. SIAM J. Sci. Comput. **34**(5), A2650–A2669 (2012)
30. Tranquilli, P., Glandon, S.R., Sarshar, A., Sandu, A.: Analytical Jacobian-vector products for the matrix-free time integration of partial differential equations. J. Comput. Appl. Math. (2016). https://doi.org/10.1016/j.cam.2016.05.002
31. Tranquilli, P., Sandu, A.: Exponential–Krylov methods for ordinary differential equations. J. Comput. Phys. **278**, 31–46 (2014). https://doi.org/10.1016/j.jcp.2014.08.013
32. Tranquilli, P., Sandu, A.: Rosenbrock–Krylov methods for large systems of differential equations. SIAM J. Sci. Comput. **36**(3), A1313–A1338 (2014). https://doi.org/10.1137/130923336
33. Trefethen, L.N.: Evaluating matrix functions for exponential integrators via Carathéodory–Fejér approximation and contour integrals. Electron. Trans. Numer. Anal. **29**, 1–18 (2007)
34. Verwer, J.G., Spee, E.J., Blom, J.G., Hundsdorfer, W.: A second-order rosenbrock method applied to photochemical dispersion problems. SIAM J. Sci. Comput. **20**(4), 1456–1480 (1999)
35. van der Vorst, H.A.: Iterative Krylov Methods for Large Linear Systems. Cambridge University Press, Cambridge (2003). https://doi.org/10.1017/cbo9780511615115