EXPLICIT STABILIZED IMPLEMENTATION OF SINGLY DIAGONALLY IMPLICIT RUNGE-KUTTA METHOD *

IBRAHIM ALMUSLIMANI †, GILLES VILMART ‡, AND KONSTANTINOS C. ZYGALAKIS §

Abstract. Implicit methods are a natural approach for the integration of stiff differential equations, to avoid time-step restrictions faced by standard explicit integrators. Explicit stabilised integrators are an alternative to implicit methods, which can be particularly efficient in high-dimensional applications with diffusive terms. Towards the best of both worlds, we introduce a new explicit stabilised implementation of a class of diagonally implicit Runge-Kutta methods. This allows us to implement high-order singly-diagonally-implicit Runge-Kutta methods for advection-diffusion-reaction PDEs with a provable computational cost analogous to that of standard explicit stabilised methods. The main ingredient is to recast the implicit Runge–Kutta update as the steady state of a modified auxiliary system, which is then computed using a partitioned Runge–Kutta–Chebyshev method inspired by optimisation techniques.

Key words. diagonally implicit Runge-Kutta methods, explicit stabilised methods, partitioned Runge-Kutta methods, stiff problems, advection-diffusion-reaction problems

MSC codes. 65L20, 65M12, 65M20

1. Introduction.

In this paper, we consider systems of ordinary differential equations (ODEs) representing space discretisations of partial differential equations (PDEs) of the form

$$\dot{y} = F(y), \tag{1.1a}$$

$$F(y) := F_D(y) + F_A(y) + F_R(y). \tag{1.1b}$$

where $F_D(y), F_A(y), F_R(y) \in \mathbb{R}^d$ represent diffusion, advection and reaction terms respectively. Typically (1.1) arises from the spatial discretisation of an advection-diffusion-reaction problem in N spatial dimensions, and assuming a spatial grid of size Δx, the dimension of the ODE grows like $d = \mathcal{O}((\Delta x)^{-N})$.

The numerical time integration of (1.1) is very challenging since this is a *stiff* ODE. Following [13], we call a differential equation *stiff* when a standard explicit numerical integrator such as a standard explicit Runge-Kutta method faces a severe time-step restriction. In particular, in the case of (1.1) the eigenvalues of the Jacobian of F_D are typically distributed along the negative real axis in an interval that grows as $[-\mathcal{O}(\Delta x^{-2}), 0]$ (for a symmetric diffusion operator), which implies that the time step Δt that can be used by the explicit Euler method suffers from the CFL $\Delta t \leq C\Delta x^2$. Furthermore, the eigenvalues of the Jacobian of F_A are typically distributed along the imaginary axis in an interval that grows as $[-i\mathcal{O}(\Delta x^{-1}), i\mathcal{O}(\Delta x^{-1})]$, while the eigenvalues of the Jacobian of the reaction term F_R even though are usually not related to Δx can have a ratio $\max_j |\mathcal{R}\lambda_j| / \min_j |\mathcal{R}\lambda_j|$ that can be very large or vary over several orders of magnitude when modeling multiscale reaction terms.

*Submitted to the editors July 8, 2026.

Funding:

†Ecole Polytechnique Fédérale de Lausanne (EPFL), Swiss Plasma Center (SPC), CH-1015 Lausanne, Switzerland. Ibrahim.Almuslimani@epfl.ch

‡Section de Mathématiques, Université de Genève, CP 64, 1211 Genève 4, Switzerland. Gilles.Vilmart@unige.ch

§School of Mathematics and the Maxwell Institute for Mathematical Sciences, University of Edinburgh, Edinburgh, EH9 3FD, UK. K.Zygalakis@ed.ac.uk

There are two different approaches to dealing with the issue of stiffness within the class of Runge-Kutta methods. The first relates to the design of implicit methods, with a prime example being the implicit Euler method. These methods have the advantage of achieving stability unconditionally with respect to the stepsize, which comes with the cost of solving a system of nonlinear equations. When the dimension of the corresponding differential equation is low, the nonlinear system can be solved efficiently using Newton's method [17]. As the dimension of the differential equation increases, diagonally implicit Runge-Kutta methods are attractive as they lead to a triangular system which is easier to solve using lower-dimensional Jacobian matrices using an appropriate partitioning of the coefficient matrix as proposed in [20]. Nevertheless, when one deals with differential equations like (1.1), dealing with the corresponding nonlinear equations (even in the triangular form) can be cumbersome as it requires specialised linear algebra techniques, where the use of preconditioners raises issues with both memory access and storage on parallel supercomputers.

An alternative approach to using implicit methods for solving *stiff* differential equations is to use explicit stabilised methods [2]. In their basic form, these are first-order explicit methods that can take larger time steps than the standard Euler method (for an easily computable numerical cost). This property together with the simplicity of implementation has made explicit stabilised methods the method of choice for diffusion-dominated problems. In particular, the second-order Runge Kutta Chebyshev (RKC) method was very successful for large dimensional parabolic problems, while methods with nearly optimal extended stability domain sizes were proposed as the second-order ROCK2 method [4] and the fourth-order ROCK4 method [1]. In the spirit of implicit-explicit methods (IMEX) [10, 9], the versatility of explicit stabilised methods allowed many extensions to partitioned Runge-Kutta methods for addressing problems with additional advection and possibly stiff reaction terms. Examples include Implicit Runge Kutta Chebyshev (IRKC) [18] to deal with stiff reaction terms and Partitioned Runge Kutta Chebyshev (PRKC) [22] to include advection terms. In addition, the Partitioned Implicit Orthogonal Runge Kutta Chebyshev method (PIROCK) [5] combined these two approaches to propose a second-order method for solving advection-diffusion reaction equations. In particular, PIROCK has recently emerged as an efficient alternative to classical splitting approaches for solar atmosphere simulations in astrophysics [21] and for large-scale gyrokinetic simulations in plasma physics [7, 6]. Another recent use of explicit stabilised methods is for solving optimisation problems. More precisely, in [12] by exploiting the good stability properties of explicit stabilised methods a new method called the *Runge–Kutta Chebyshev* descent (RKCD) was proposed and showed to perform similarly to the state of the art for strongly convex optimisation problems.

A major limitation of methods such as IRKC, PRKC, and PIROCK that use an explicit stabilised method to deal with the F_D term in (1.1), is that they are at most second-order accurate in Δt. Indeed, while the construction of the PIROCK method which is based on a splitting into three terms needed to solve 12 order conditions, similar construction of higher order methods would be cumbersome due to the dramatic increase of order conditions for partitioned Runge-Kutta methods. On the other hand, implicit methods can achieve much higher order of accuracy in Δt, but as discussed above the implementation as well as the storage requirements of the Newton method and variants in high dimensions become an issue. In this paper, inspired by the success of RKCD as an optimisation method, we revisit diagonally implicit methods and their implementation. In particular, instead of solving the nonlinear system with variants of Newton's method, we introduce a new variant of the RKCD method

that only outputs first-order information. This method is simple to implement due to its explicit nature which circumvents the need for specialised linear algebra tools. Furthermore, it leads to rigorous estimates of the computational complexity of the proposed method. In particular, we can show that

1. In the pure linear diffusion case ($F_A = F_R = 0$ in (1.1)), we prove that one time step of an SDIRK method[1] (of arbitrary order) by an iterative method requires $\mathcal{O}(\sqrt{\Delta t}/\Delta x)$ evaluations of F_D avoiding the standard CFL condition of standard explicit Runge Kutta methods. This essentially implies that we are able to implement a higher order diagonally implicit Runge-Kutta method at the cost of an explicit stabilised method.
2. For a class of linear advection-diffusion PDE ($F_R = 0$), we prove that in spite of relatively large Peclet numbers the proposed partitioning allows to implement one step of an SDIRK method with a similar complexity to the one of the pure diffusion case discussed at the previous point.
3. For various nonlinear diffusion-advection-reaction problems, numerical experiments illustrate the robustness of the approach where we show that the explicit stabilised iterations converge in the case of an order 4 SDIRK method with error control, a state-of-the-art Runge-Kutta method proposed in [13, Table IV.6.5, p. 100] and considered in this paper to illustrate the new approach.

The paper is organized as follows. In Section 2 we revisit the ideas of numerical stability of Runge-Kutta methods, while also discussing the basics of explicit stabilised methods. In Section 3 we revisit implicit Runge-Kutta methods and discuss the main idea behind solving the nonlinear system using an explicit stabilised algorithm. In addition, we discuss a specific application of the new explicit implementation of implicit methods in the context of advection-diffusion-reaction equations and present an algorithm for doing so. A detailed convergence analysis in the case of pure linear diffusion and linear diffusion-advection is presented in Section 4. Finally, in Section 5 we present a number of numerical computations that illustrate the theoretical convergence analysis in the pure linear diffusion case, and the linear diffusion-advection case. In addition, applying the new explicit stabilised implementation exSDIRK4 of an order 4 SDIRK method to several nonlinear advection-diffusion-reaction problems illustrates the wide applicability and versatility of our approach.

2. Preliminaries. In this paper, we are interested in the numerical integration of systems of ODEs of the form

$$\dot{y}(t) = g(y(t)), \qquad y(0) = y_0, \quad t > 0 \tag{2.1}$$

where $y(t) \in \mathbb{R}^d$ and the vector field $g : \mathbb{R}^d \mapsto \mathbb{R}^d$ is assumed smooth. In this section, we recall the main tools for stability and accuracy analysis of Runge-Kutta methods.

2.1. Stability function and stability domain. Let $\lambda \in \mathbb{C}$ and consider the Dahlquist test equation

$$\dot{y}(t) = \lambda y(t), \quad y(0) = y_0 \in \mathbb{R}. \tag{2.2}$$

In the case where $\lambda \in \mathbb{C}^- = \{z \in \mathbb{C};\, \mathrm{Re}(z) \leq 0\}$, equation (2.2) has a bounded solution for $t \to +\infty$. The simplest Runge-Kutta method for solving (2.1) is the explicit Euler method which when applied to (2.2) yields the recurrence

$$y_{n+1} = (1 + \lambda \Delta t) y_n \tag{2.3}$$

[1]Here Δt is the time step used by the SDIRK method.

More generally, a Runge-Kutta method with step size Δt applied to (2.2) produces the following recurrence

$$y_n = R(\Delta t\lambda) y_{n-1} = R(\Delta t\lambda)^n y_0, \tag{2.4}$$

where $R(z)$ is called the stability function and the stability domain is defined by

$$\mathcal{S} = \{z \in \mathbb{C}\,;\, |R(z)| \leq 1\}.$$

Note that $z \in \mathcal{S}$ ensures that y_n in (2.4) remains bounded as $n \to \infty$ similarly to the true solution. In the case of the explicit Euler we have that $\mathcal{S}_{\mathrm{E}} = \{|z+1| \leq 1\}$ which corresponds to a disc of radius one, centred at -1 on the complex plane. For stiff problems where $|\lambda| \gg 1$ this implies a severe time-step restriction namely $\Delta t < 2/|\lambda|$ when $\lambda < 0$. In particular, for spatial discretization of the heat equation eigenmodes are typically of size $\mathcal{O}(\Delta x^{-2})$ which would in turn imply the CFL condition $\Delta t < c\Delta x^2$.

To avoid such severe time-step restrictions one solution is to use implicit methods. The simplest one of those is the implicit Euler method which when applied to (2.2) yields the recurrence

$$y_{n+1} = \frac{1}{1-\lambda\Delta t} y_n.$$

In this case we have $\mathcal{S}_{\mathrm{IE}} = \{|z-1| \geq 1\}$ which contains $\mathbb{C}^-$. This property is known as A-stability and can only hold in the case of implicit methods. A-stability means that there is no restriction on the time-step size for eigenvalues with negative real parts contrary to the explicit Euler method and other explicit methods. Nevertheless, in the case of severely stiff dissipative problems, a desirable additional property to A-stability is $R(\infty) = 0$ which is known as L-stability [13].

2.2. Higher order Implicit Runge-Kutta methods. The implicit Euler is an example of an L-stable integrator meaning that is particularly suitable for integrating stiff equations. However, from the point of view of accuracy, it is a first-order method. To address this issue families of L-stable Runge-Kutta methods have been proposed [13] with an arbitrarily higher order of accuracy. Important examples are the families of the Radau and Lobatto Runge-Kutta methods are that achieve orders $2m-1$ and $2m-2$ respectively for any number m of internal stages [15]. In particular, Radau with $m = 1$ reduces to the implicit Euler method, while in the case where $m = 2$ we obtain the following implicit Runge-Kutta method for integrating (2.1),

$$Y_1 = y_n + \frac{5\Delta t}{12} g\left(Y_1\right) - \frac{\Delta t}{12} g(Y_2), \tag{2.5a}$$

$$Y_2 = y_n + \frac{3\Delta t}{4} g\left(Y_1\right) + \frac{\Delta t}{4} g(Y_2), \tag{2.5b}$$

$$y_{n+1} = Y_2. \tag{2.5c}$$

For a general m-stage implicit Runge-Kutta method with coefficients $a_{ij}, b_j, i, j = 1, \ldots, m$, given the numerical solution y_n at time t_n one needs to solve the following nonlinear system in dimension $m \cdot d$ with unknowns the internal stages $Y_i \in \mathbb{R}^d, i = 1, \ldots, m$,

$$Y_i = y_n + \Delta t \sum_{j=1}^{m} a_{ij} g(Y_j), \quad i = 1, \ldots, m, \tag{2.6}$$

to obtain the solution y_{n+1} at time t_{n+1} by

$$y_{n+1} = y_n + \Delta t \sum_{i=1}^{m} b_i g(Y_i). \tag{2.7}$$

It is common to represent Runge-Kutta methods using a Butcher tableau

$$\begin{array}{c|c} c & \mathcal{A} \\ \hline & b \end{array} \tag{2.8}$$

where $\mathcal{A}$ is the matrix of coefficients $\{a_{ij}\}_{i,j=1}^{m}$, b is the weights vector $\{b_i\}_{i=1}^{m}$, and the vector c is such that $c_i = \sum_{j=1}^{m} a_{ij}$. In compact matrix and tensor notations, (2.6) can be written as

$$\boldsymbol{Y} = \mathbb{1}_m \otimes y_n + \Delta t(\mathcal{A} \otimes I_d) g[\boldsymbol{Y}] \tag{2.9}$$

where $\mathcal{A} \in \mathbb{R}^{m\times m}$ is the Runge-Kutta matrix of coefficients a_{ij} and

$$\mathbb{1}_m = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^m, \quad \boldsymbol{Y} = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_m \end{pmatrix} \in \mathbb{R}^{m\cdot d}, \quad g[\boldsymbol{Y}] = \begin{pmatrix} g(Y_1) \\ g(Y_2) \\ \vdots \\ g(Y_m) \end{pmatrix} \in \mathbb{R}^{m\cdot d} \tag{2.10}$$

In order to implement a general m-stage implicit Runge-Kutta method, for example, an m-stage Radau method, one employs Newton's method or variants of it. These methods make use of the Jacobian of the system which is of size $md \times md$. Diagonally implicit Runge-Kutta methods (DIRK) are an attractive alternative to reduce this computational cost. This is because in this case, the matrix $\mathcal{A}$ in (2.9) is lower triangular, which implies that the Jacobians used by the Newton method or its variants to solve (2.9) are of dimension $d \times d$. Furthermore, when all the diagonal coefficients are equal, the method is referred to as singly diagonally implicit (SDIRK), which may offer the potential to further reduce computational costs [13]. A particular diagonal implicit method of interest in this paper is SDIRK4 as presented in [13, Table IV.6.5, p. 100] and whose Butcher tableau is given by

$$\begin{array}{c|ccccc}
 & \frac{1}{4} & & & & \\
 & \frac{1}{2} & \frac{1}{4} & & & \\
 & \frac{17}{50} & \frac{-1}{25} & \frac{1}{4} & & \\
 & \frac{371}{1360} & \frac{-137}{2720} & \frac{15}{544} & \frac{1}{4} & \\
 & \frac{25}{24} & \frac{-49}{48} & \frac{125}{16} & \frac{-85}{12} & \frac{1}{4} \\
\hline
y_1 & \frac{25}{24} & \frac{-49}{48} & \frac{125}{16} & \frac{-85}{12} & \frac{1}{4} \\
\hat{y}_1 & \frac{59}{48} & \frac{-17}{96} & \frac{225}{32} & \frac{-85}{12} & 0 \\
err & -\frac{3}{16} & \frac{-27}{32} & \frac{25}{32} & 0 & \frac{1}{4}
\end{array} . \tag{2.11}$$

Remark 2.1. Note that for SDIRK4 we have $y_{n+1} = Y_m$, where $m = 5$. We call such methods stiffly accurate [13]. The coefficient $\gamma = 1/4$ is chosen in [13, Sect. IV.6, p. 99] to achieve both L-stability and a favorable error constant with an embedded

method for error control and a continuous output of order 3. In [13, Sect. IV.6, p. 99], a numerical better choice with $\gamma = 4/15$ and numerical values of the L-stable Runge-Kutta coefficients is also proposed but we choose in the paper the SDIRK4 method (2.11) for simplicity due to its nice rational coefficients.

Remark 2.2. The SDIRK4 method in (2.11) also includes an embedded method $\hat{y}_1$ for a variable time-step strategy [13] based on the error estimator

$$err = J^{-1}(y_1 - \hat{y}_1) \tag{2.12}$$

where each new step y_1 with time step Δt is accepted if err is small enough compared to the tolerance tol or rejected otherwise. The next time step is then computed as $\Delta t_{new} = \xi \Delta t (tol/err)^{1/5}$ where $\xi = 0.8$ is a safety factor and tol is the tolerance prescribed by the user. The term J^{-1} in (2.12) denotes the inverse of the Jacobian matrix $J = I_d - \Delta t \gamma \frac{\partial g}{\partial y}$ computed during the Newton method or its variant at iterates of the internal stages Y_i, and hence J^{-1} is already computationally available with minor overhead in practice. This is a standard methodology for stiff problems proposed by Shampine, as presented in [13, Sect. IV.8], to circumvent the lack of L-stability of the embedded method and hence stabilize the error estimator which would otherwise be too pessimistic and fail for stiff problems.

2.3. Explicit stabilised Runge-Kutta methods. Even with the dimensionality reduction of diagonally implicit Runge-Kutta methods in terms of the Jacobians used for solving the nonlinear equations, implicit methods in high dimensions become prohibitively expensive. On the other hand, explicit schemes avoid the solution of such high-dimensional systems, but have bounded stability domains. For this reason, standard explicit methods face severe step size restrictions when integrating stiff problems. Explicit stabilised Runge-Kutta methods represent a good compromise, they are fully explicit schemes with extended stability domains over the negative real axis.

Explicit stabilised Runge-Kutta methods are constructed to have (nearly) maximal stability domain sizes along the negative real axis constrained by the fact that they remain explicit. In particular, for an s-stage method the stability polynomial would be of degree s. The goal is then to find, for a given integer p (the order of the method), a polynomial of degree s (the number of internal stages of the method) satisfying

$$R_s(z) = 1 + z + \frac{z^2}{2} + \cdots + \frac{z^p}{p!} + \alpha_{p+1} z^{p+1} + \cdots + \alpha_s z^s,$$

such that $|R_s(z)| \leq 1$ for $z \in [-L_s, 0]$ with $L_s > 0$ is as large as possible. For $p = 1$, the optimal value of L_s is $2s^2$ [2, 13]. The corresponding polynomial is the shifted Chebyshev polynomial $R_s(z) = T_s(1 + z/s^2)$ where $T_s(\cos\theta) = \cos(s\theta)$ is the first kind Chebyshev polynomial of degree s. This polynomial oscillates between -1 and 1 which introduces singularities in the stability domain and prevents convergence for some eigenvalues. For this reason, we modify the stability function in the following way

$$R_s(z) = \frac{T_s(\omega_0 + \omega_1 z)}{T_s(\omega_0)}, \tag{2.13}$$

where $\omega_0 = 1 + \eta/s^2$, $\omega_1 = T_s(\omega_0)/T_s'(\omega_0)$, and η is called the damping parameter. The polynomial (2.13) now oscillates between $-\alpha_s(\eta)$ and $\alpha_s(\eta)$ for every $z \in [-L_{s,\eta}, -\delta_\eta]$,

where

$$\alpha_s(\eta) = \frac{1}{T_s(\omega_0)} < 1, \quad L_{s,\eta} = \frac{1+\omega_0}{\omega_1}, \tag{2.14}$$

where δ_η is a positive real close to zero ($R_s(0) = 1$).

Using the well-known three-term recurrence relation of the first kind Chebyshev polynomials $T_{j+1}(z) = 2zT_j(z) - T_{j-1}(z)$, for $j \geq 2$ (with $T_0(z) = 1$ and $T_1(z) = z$), we can derive an efficient and stable implementation (in terms of memory and rounding errors) of the so-called Chebyshev method or the first order Runge-Kutta-Chebyshev (RKC) method with stability function (2.13), as analyzed by [19] (also for second order explicit stabilised methods). When applied to non-linear problems where the Jacobian of g has negative real eigenvalues, this method allows for taking much larger time steps than standard explicit methods would allow. Algorithm 2.1 summarises one step of the first order s-stage RKC method with step-size Δt for the solution of (2.1). Note that there are three parameters needed for implementing this Algorithm,

Algorithm 2.1 First order Runge-Kutta Chebyshev for the time integration of (2.1)

Input: Initial state y_n, damping η, upper bound L for the eigenvalues of the Jacobian of g at y_n, time step Δt

$$\begin{aligned} K_0 &= y_n, \quad K_1 = K_0 + \mu_1 \Delta t g(K_0), \\ K_j &= \mu_j \Delta t g(K_{j-1}) + \nu_j K_{j-1} + (1-\nu_j) K_{j-2}, \quad j = 2, \ldots, s, \\ y_{n+1} &= K_s, \end{aligned} \tag{2.15}$$

where

$$\mu_j = \frac{2\omega_1 T_{j-1}(\omega_0)}{T_j(\omega_0)}, \quad \nu_j = \frac{2\omega_0 T_{j-1}(\omega_0)}{T_j(\omega_0)}$$

and s should be chosen large enough such that $L_{s,\eta} > L\Delta t$

Output: Solution y_{n+1} after a time step of size Δt

the damping η, the number of stages s as well as the step-size Δt. For a fixed value of the damping parameter η and a given time step Δt one needs to choose the stage parameter s in such a way that $L_{s,\eta} \geq L\Delta t$. Typically for ODE integration one takes η small (e.g $\eta = 0.05$) which gives a nearly optimal stability domain size $L_{s,\eta} \geq (2 - \frac{4}{3}\eta)s^2$, and a stage parameter

$$s = \left\lceil \sqrt{\frac{L\Delta t}{2 - \frac{4}{3}\eta}} \right\rceil$$

where L is an upper bound for the largest eigenvalue of the Jacobian of g evaluated at y_n. For larger values of damping parameter η, the width $L_{s,\eta}$ of the stability domain reduces further but still grows quadratically with s and the number of stages s is then chosen large enough accordingly, see for instance [3, Remark 6.1] for details.

2.4. Application to optimization. In the recent paper [12] the authors utilised (2.15) to solve efficiently stiff optimization problems by an algorithm called Runge-Kutta Chebyshev descent (RKCD). Here, by stiff optimization problems, we mean the case where the Hessian of the objective function has large eigenvalues distributed in the positive half-plane close to the real axis. In particular, consider the problem

$$\min_{x \in \mathbb{R}^d} f(x), \tag{2.16}$$

where the dimension d is large and $f : \mathbb{R}^d \to \mathbb{R}$, is ℓ-strongly convex differentiable function, with L-Lipschitz derivative, and such that the real parts of the eigenvalues of its Hessian matrix $\mathcal{H}f(x)$ are strictly positive for all $x \in \mathbb{R}^d$. The condition number of the Hessian is bounded by $\kappa = \frac{L}{\ell}$.

A starting point for solving the minimization problem (2.16) is to consider discretizations of the gradient flow

$$\dot{x}(t) = -\nabla f(x(t)), \qquad x(0) = x_0 \in \mathbb{R}^d, \tag{2.17}$$

where x_0 is an initialization. Applying an explicit Euler method with step size[2] h to (2.17) is equivalent to the gradient descent method

$$x_{k+1} = x_k - h\nabla f(x_k), \qquad k = 0, 1, 2, ... \tag{2.18}$$

where x_k[3] is the numerical approximation of $x(t_k)$. In the strongly convex setting described above when $(L \gg l)$, this method faces a severe restriction on h to be stable. Furthermore, it is known that the optimal h in terms of convergence speed towards the minimizer is given by $h = \frac{2}{L+\ell}$ [16]. This implies that the number of iterations needed to find the minimizer of f scales linearly with the condition number κ. Note that this behaviour is suboptimal since the most efficient optimization methods that make use only of the gradient of f need $\mathcal{O}(\sqrt{\kappa})$ iterations to find the minimum [16].

To improve the convergence speed of the explicit Euler method one can apply a more sophisticated discretization of the gradient flow (2.17). For example, one can use the first order RKC (2.15) to integrate (2.17). In particular, if one has estimates for the smallest and largest eigenvalues of the Hessian matrix $\nabla^2 f$, then by choosing the parameters η, h, and s in (2.15) appropriately one can obtain a very efficient optimization method called RKCD [12] for strongly convex problems, see also Algorithm 2.2. In the case of RKCD, a calculation of x_{k+1} for x_k requires s gradient evaluation. Now the following proposition from [12] characterizes the speed of convergence of iterates x_k towards the minimizer $x_\star$ when f is quadratic in (2.16).

PROPOSITION 2.3. *[12]. Let $f(x) = \frac{1}{2}x^T Cx + b^T x + c$ and consider the minimization problem* (2.16)*. Now let x_k be the iterate of Algorithm 2.2 with parameters chosen according to* (2.19) *and* (2.20)*. Then the following convergence estimates hold,*

$$\|x_k - x_*\| \leq \alpha_s^k(\eta)\|x_0 - x_*\|, \tag{2.21}$$

and

$$f(x_{k+1}) - f(x_*) \leq \alpha_s^2(\eta)(f(x_k) - f(x_*)), \tag{2.22}$$

where x_* is the unique minimizer of f, and $\alpha_s(\eta)$ satisfies (2.14)*.*

A straightforward calculation shows that the learning rate h in (2.19) satisfies $h = \frac{\eta}{\ell} + \mathcal{O}(\eta^2 + s^{-2})$, for $\eta \to 0, s \to +\infty$ which makes it mildly dependent on s. Precisely, the following estimates holds.

LEMMA 2.4. *The learning rate h defined in* (2.19) *satisfies the bound*

$$h \leq \frac{\eta e^\eta}{\ell}. \tag{2.23}$$

[2]We will use Δt for the actual time step of numerical integrators for differential equations and h for optimization steps.

[3]When we describe an optimization algorithm to approximate a minimizer x_* we will use the notation x_k for the k-th iterate of the algorithm.

Algorithm 2.2 [12] Runge-Kutta Chebyshev descent method (RKCD)

Input: The gradient force $g = -\nabla f$ of a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$. Damping η. Lower and upper bounds ℓ and L for eigenvalues of the Jacobian of g. Initialisation $x_0 \in \mathbb{R}^d$.

Body:

- Compute the time step h (learning rate) and the number of stages s using

$$s = \left\lceil \sqrt{(\kappa - 1)\eta/2} \right\rceil, \quad h = \frac{\omega_0 - 1}{\omega_1 \ell} \tag{2.19}$$

where $\kappa = L/\ell$ and

$$\omega_0 = 1 + \frac{\eta}{s^2}, \quad \omega_1 = \frac{T_s(\omega_0)}{T_s'(\omega_0)}, \tag{2.20}$$

with T_s is the Chebyshev polynomial of the first kind with degree s.

- Until convergence, for $k \in \{0, 1, 2, 3, \ldots\}$, **repeat**:
  - Set $x_k^0 = x_k$ and $x_k^1 = x_k^0 + h\mu_1 g(x_k)$ with $\mu_1 = \omega_1/\omega_0$
  - For $j \in \{2, \cdots, s\}$, **repeat**:

$$x_k^j = \mu_j h g(x_k^{j-1}) + \nu_j x_k^{j-1} - (\nu_j - 1) x_k^{j-2},$$

$$\mu_j = \frac{2\omega_1 T_{j-1}(\omega_0)}{T_j(\omega_0)}, \quad \nu_j = \frac{2\omega_0 T_{j-1}(\omega_0)}{T_j(\omega_0)}.$$

  - Set $x_{k+1} = x_k^s$.

Output: Estimate $\widehat{x} = x_{n+1}$ of a minimiser of (2.16).

Proof. We recall the following formula for all $s \geq 1, k \geq 0$,

$$T_s^{(k)}(1) = \prod_{j=0}^{k-1} \frac{s^2 - j^2}{2j + 1} \leq s^{2k}. \tag{2.24}$$

Using $1/\omega_1 = T_s'(\omega_0)/T_s(\omega_0) \leq T_s'(\omega_0)$, and performing a Taylor expansion of $T_s'(\omega_0)$ for η close to 0 yields

$$\ell h = \frac{\eta}{s^2 \omega_1} \leq \frac{\eta T_s'(\omega_0)}{s^2} = \sum_{k=0}^{s-1} \frac{T_s^{(k+1)}(1)}{k! s^{2k+2}} \eta^{k+1} \leq \sum_{k=0}^{s-1} \frac{\eta^{k+1}}{k!} \leq \eta e^{\eta}$$

where we used (2.24) in the last inequality, which concludes the proof. ∎

Using this proposition and the properties of $\alpha_s(\eta)$ it is possible to show that the effective rate of convergence of function iterates towards the minimizer x_* scales like $1 - c(\eta)/\sqrt{\kappa}$ when $\kappa \gg 1$ [12], exhibiting thus accelerated behavior from an optimization point of view. This, in turn, implies that to calculate $x_\star$ with target accuracy ϵ one needs to do $\mathcal{O}(\sqrt{\kappa} \log \epsilon^{-1})$ gradient evaluations.

3. A new explicit implementation of SDIRK methods. As discussed in Section 2.2, implicit methods are usually implemented using Newton's method or its variants which can be memory demanding and delicate for ill-conditioned problems in particular for high-dimensional problems arising from discretizing PDEs. Here, we

propose an alternative way of implementing implicit Runge-Kutta methods (2.6)-(2.7) based on explicit stabilised methods. To do this, one needs to find the solution $\boldsymbol{Y}_{n+1}$ to the system of non-linear equations (2.9) given the current state y_n. Our starting point for achieving this is to introduce the auxiliary system of ODEs,

$$\dot{\boldsymbol{Y}}(t) = -\boldsymbol{Y}(t) + \mathbb{1}_m \otimes y_n + \Delta t(\mathcal{A} \otimes I_d) g[\boldsymbol{Y}(t)] \tag{3.1}$$

that satisfies under appropriate assumptions

$$\lim_{t \to +\infty} \boldsymbol{Y}(t) = \boldsymbol{Y}_{n+1}.$$

We have thus reformulated the solution of an m-stage implicit Runge-Kutta method at time t_{n+1} as the steady-state solution of the ODE (3.1). In the simplest case of the implicit Euler method and assuming a gradient structure $g(y) = -\nabla V(y)$ in (2.1), it is easy to see that (3.1) also has a gradient structure (2.17) with $f(y) = \Delta t V(y) + \frac{1}{2}\|y - y_n\|_2^2$. In this simple situation, following the ideas in [12] one can apply RKCD directly to compute $\boldsymbol{Y}_{n+1}$. However, this gradient structure assumption is not satisfied in general for a diagonally implicit Runge-Kutta method, even when (1.1) has a gradient structure. In particular, (3.1) in this case becomes

$$\dot{\boldsymbol{Y}}(t) = -\boldsymbol{Y}(t) + \mathbb{1}_m \otimes y_n + \Delta t(\mathcal{A} \otimes I_d)(F_D[\boldsymbol{Y}(t)] + F_A[\boldsymbol{Y}(t)] + F_R[\boldsymbol{Y}(t)]) \tag{3.2}$$

As we will see, without this gradient assumption, applying Algorithm 2.2 to (3.2) would lead to arbitrarily slow convergence to $\boldsymbol{Y}_{n+1}$ as the stiffness grows. To tackle this issue, and inspired by the partitioning $\mathcal{A} = \gamma I_m + (\mathcal{A} - \gamma I_m)$ of the Runge-Kutta coefficients into diagonal part and strictly lower diagonal part proposed in [20], we consider the following partition of (3.2),

(3.3a)

$$\begin{aligned}
\dot{\boldsymbol{Y}}(t) \;&= \mathbb{1}_m \otimes y_n - \boldsymbol{Y}(t) \;+\; \Delta t\, \gamma\, I_m \otimes I_d\, F_D[\boldsymbol{Y}(t)] && =:\; G_D(\boldsymbol{Y}(t)) \\
&\quad +\; \Delta t\, (\mathcal{A} - \gamma I_m) \otimes I_d\, F_D[\boldsymbol{Y}(t)] \;+\; \Delta t\, \mathcal{A} \otimes I_d\, F_A[\boldsymbol{Y}(t)] && =:\; G_A(\boldsymbol{Y}(t)) \qquad \text{(3.3b)} \\
&\quad +\; \Delta t\, \mathcal{A} \otimes I_d\, F_R[\boldsymbol{Y}(t)] && =:\; G_R(\boldsymbol{Y}(t)), \qquad \text{(3.3c)}
\end{aligned}$$

where we split the coefficient matrix $\mathcal{A}$ of the singly diagonally implicit Runge-Kutta method into its diagonal part γI_m and its strictly lower triangular part $\mathcal{A} - \gamma I_m$. The new Algorithm 3.1 is then defined as follows.

Remark 3.1. Algorithm 3.1 calculates $\boldsymbol{Y}_{n+1}$ by applying Algorithm 2.2 to the following new partitioned auxiliary problem:

$$\dot{\boldsymbol{Y}} = G_D(\boldsymbol{Y}) - G_D(\boldsymbol{Y}_n) + J^{-1}\big(G_D(\boldsymbol{Y}_n) + G_A(\boldsymbol{Y}_n) + G_R(\boldsymbol{Y}_n)\big), \tag{3.5}$$

$$J = I_{md} - h\, G_R'(\boldsymbol{Y}_n). \tag{3.6}$$

Note that this partitioning preserves the steady state $\boldsymbol{Y}_{n+1}$ of (3.2) since

$$G_D(\boldsymbol{Y}_{n+1}) + G_A(\boldsymbol{Y}_{n+1}) + G_R(\boldsymbol{Y}_{n+1}) = 0.$$

Remark 3.2. The partitioning (3.3) and the corresponding treatment of the G_R term in (3.5) assume that the reaction term F_R is stiff. Note that the Jacobian inversion in this setting is cheap as it has a block diagonal structure thanks to the

Algorithm 3.1 New Runge-Kutta Chebyshev implementation of diagonally implicit Runge-Kutta methods

Input: The vector fields $\mathbb{R}^d \to \mathbb{R}^d$, F_D (diffusion terms), F_A (advection terms), F_R (reaction terms). The initial state $y_n \in \mathbb{R}^d$, an approximation $\boldsymbol{x}_0 \in \mathbb{R}^{md}$ of $\boldsymbol{Y}_n$. Upper bounds λ_{max} for the spectral radius of ∇F_D. Diagonally Runge-Kutta coefficient matrix A with value $\gamma > 0$ on the diagonal, time-step size Δt. Damping η.

Body:

- Compute the time step h (learning rate) and the number of stages s according to

$$s = \left\lceil \sqrt{(\kappa - 1)\eta/2} \right\rceil, \quad h = \frac{\omega_0 - 1}{\omega_1 \ell}$$

where $\ell = 1$, $L = 1 + \gamma \Delta t \lambda_{max}, \kappa = L/\ell$ and

$$\omega_0 = 1 + \frac{\eta}{s^2}, \quad \omega_1 = \frac{T_s(\omega_0)}{T_s'(\omega_0)},$$

with T_s is the Chebyshev polynomial of the first kind with degree s.

- Until convergence, for $k \in \{0, 1, 2, 3, \ldots\}$, **repeat**:
  - Define

$$G(\boldsymbol{x}) := G_D(\boldsymbol{x}) - G_D(\boldsymbol{x}_k) + J^{-1}\big(G_D(\boldsymbol{x}_k) + G_A(\boldsymbol{x}_k) + G_R(\boldsymbol{x}_k)\big),$$
$$J = I_{md} - h\, G_R'(\boldsymbol{x}_k),$$

  - Set $\boldsymbol{x}_k^0 = \boldsymbol{x}_k$ and $\boldsymbol{x}_k^1 = \boldsymbol{x}_k^0 + h\mu_1 G(\boldsymbol{x}_k)$ with $\mu_1 = \omega_1/\omega_0$
  - For $j \in \{2, \cdots, s\}$, **repeat**:

$$\boldsymbol{x}_k^j = \mu_j h G(\boldsymbol{x}_k^{j-1}) + \nu_j \boldsymbol{x}_k^{j-1} - (\nu_j - 1)\boldsymbol{x}_k^{j-2},$$

$$\mu_j = \frac{2\omega_1 T_{j-1}(\omega_0)}{T_j(\omega_0)}, \quad \nu_j = \frac{2\omega_0 T_{j-1}(\omega_0)}{T_j(\omega_0)}. \tag{3.4}$$

  - Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k^s$.

Output: Approximation $\boldsymbol{x}_{k+1}$ of the Runge-Kutta solution $\boldsymbol{Y}_{n+1} \in \mathbb{R}^{md}$ after one step.

local nature of the reaction terms. In practice, for a non-stiff reaction, one takes G_R to be zero to avoid the Jacobian computations and then take

$$G_A(\boldsymbol{Y}(t)) := \Delta t\, (\mathcal{A} - \gamma I_m) \otimes I_d\, F_D[\boldsymbol{Y}(t)] \;+\; \Delta t\, \mathcal{A} \otimes I_d\, \left(F_A[\boldsymbol{Y}(t)] + F_R[\boldsymbol{Y}(t)]\right).$$

This corresponds to treating together the advection and non-stiff reaction terms as first proposed for PRKC [22] and also used in PIROCK [5].

In Algorithm 3.1, we set $\ell = 1$ for simplicity, as the spectral gap for the Jacobian of G_D. Note that if $\lambda_{\min} \geq 0$ denotes the smallest eigenvalue of the Jacobian of $-F_D$ one could use the more precise value $\ell = 1 + \Delta t \lambda_{\min}$. Concerning the choice of the damping parameter η, while the analysis applies for any value $\eta > 0$, it was shown numerically in [12] that the value $\eta = 1.17$ is an efficient choice for RKCD. For Algorithm 3.1, we choose $\eta = 4$ which reveals efficient in the considered numerical test problems (see Section 5). Concerning the stopping criteria for the main iteration of Algorithm 3.1,

we use for simplicity the condition $\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\| < tol_{it}$, with $tol_{it} = 10^{-12}$. For the initialisation, we use for simplicity $\boldsymbol{x}_0 = \mathbb{1}_m \otimes y_n$ where y_n is the current state.

4. Convergence and stability analysis. In this section, we prove the convergence of Algorithm 3.1 in two special cases, pure linear diffusion and linear advection-diffusion. Additionally, we provide the rationale behind the partition in equation (3.3) for a general advection-diffusion-reaction equation.

4.1. The case of pure linear diffusion. We consider

$$\partial_t u = a\Delta u, \quad (t,x) \in [0,+\infty) \times (0,1)^N \tag{4.1}$$

with periodic boundary conditions and a smooth initial condition $u(x,0) = 1 + e^{-50\|x-c\|^2}$ where c is the center of the spatial domain. We now discretise in space using a finite difference method with a uniform mesh size Δx in each spatial dimension. We thus have

$$F_D(y) = -Dy, \quad F_A(y) = 0 \quad F_R(y) = 0 \tag{4.2}$$

in (1.1) with $D \in \mathbb{R}^{d\times d}$ is the discrete Laplacian with periodic boundary conditions. In this case, we have that the bound for the spectral radius of $\nabla F_D(y) = -D$ is given by $\lambda_{\text{heat},max} = 4Na/\Delta x^2$, for $N = 1, 2, 3$.

Implicit Euler. We consider the solution of (1.1) by implicit Euler when the different terms are given by (4.2). In this case, we have that $m = 1$ and $\mathcal{A} = \gamma = 1$ in (3.2) implying that

$$G_D(y) = \mathbb{1}_m \otimes y_n - y - \Delta t Dy, \quad G_A(y) = 0 \quad G_R(y) = 0 \tag{4.3}$$

in (3.3). We now use Algorithm 3.1 to calculate the solution of implicit Euler at time t_{n+1} given the solution y_n at time t_n. Note that in this case, because $G_A(y) = G_R(y) = 0$ and $m = 1$ (3.2) has a gradient structure and Algorithm 3.1 coincides with Algorithm 2.2. One would thus expect from Proposition 2.3 that given y_n, the cost to approximate y_{n+1} scales proportionally to the $\sqrt{\kappa}$ where κ is the condition number of G_D given by $\kappa = 1 + \gamma\Delta t\lambda_{\text{heat},max} = 1 + 4Na/\Delta x^2$. To illustrate this, we now vary κ by changing the size of the mesh refinement $\Delta x = 1/4, 1/8, 1/16, 1/32, 1/64$ while keeping $\Delta t = 5 \times 10^{-2}$. The results of this experiment are presented in Figure 1 for $N = 1$ and $N = 3$ spatial dimensions, and show that the cost in terms of gradient evaluations indeed scales like $\mathcal{O}(\sqrt{\kappa})$ independently of the value of N. Here we have used a prescribed tolerance $\varepsilon = 10^{-10}$ in the criteria for stopping the iterations.

SDIRK4. We now consider the solution of (1.1) by SDIRK4 when the different terms are given by (4.2). In this case $m = 5$ and the Runge-Kutta matrix $\mathcal{A}$ is given by (2.11), hence $\gamma = 1/4$ and

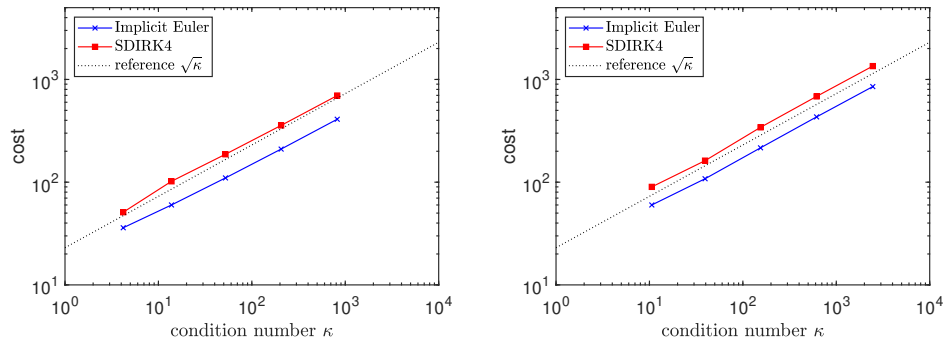
$$G_D(\mathbf{Y}) = \mathbb{1}_m \otimes y_n - \mathbf{Y} - \frac{1}{4}\Delta t(I_m \otimes D)\mathbf{Y}, \tag{4.4a}$$

$$G_A(\mathbf{Y}) = \Delta t\left[\left(\mathcal{A} - \frac{1}{4}I_m\right) \otimes D\right]\mathbf{Y} \tag{4.4b}$$

$$G_R(\mathbf{Y}) = 0. \tag{4.4c}$$

Unlike the implicit Euler method, we see that in the case of SDIRK4 Algorithm 3.1 doesn't coincide with Algorithm 2.2, hence Proposition (2.3) does not directly apply. Nevertheless, as we observe[4] in Figure 1 given y_n, the cost to approximate

[4] we have used the same parameters $\varepsilon, \Delta x, \Delta t$ as for the implicit Euler



(a) 1D case, DOF: $4, 8, 16, 32, 64$ (b) 3D case, DOF: $4^3, 8^3, 16^3, 32^3, 64^3$

Fig. 1: Comparison of the cost (number of matrix-vector products) as a function of the condition number κ for Algorithm 3.1.

y_{n+1} with Algorithm 3.1 scales again proportionally to $\mathcal{O}(\sqrt{\kappa})$ independently of the value of N, where κ is the condition number of G_D. We will prove in Theorem 4.2 that this is indeed a property of Algorithm 3.1 for the implementation of an m-stage SDIRK method in the pure linear diffusion case, equation (4.2).

4.1.1. Why the partitioning?. We now discuss in detail why the partitioning (3.5) is necessary for the construction of Algorithm 3.1 and why applying Algorithm 2.2 without the gradient assumption to (3.2) would fail in general, even in the simplest possible case of pure linear diffusion where F_D, F_R, F_A are given by (4.2). More precisely, then (3.2) becomes

$$\dot{\boldsymbol{Y}}(t) = \mathbb{1}_m \otimes y_n + \mathcal{V}\boldsymbol{Y}(t), \quad \mathcal{V} \coloneqq -I_{md} - \Delta t(\mathcal{A} \otimes D). \tag{4.5}$$

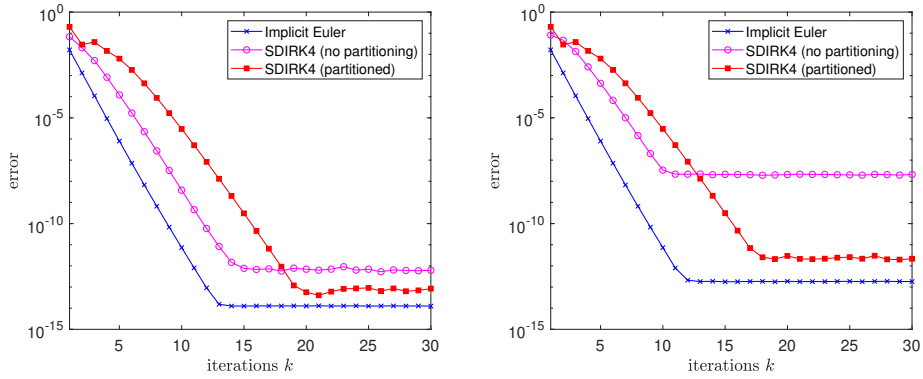
Note that because $\mathcal{A}$ is lower triangular with all entries in the diagonal equal to γ when $m \geq 2$ neither $\mathcal{A}$ nor $\mathcal{V}$ can be put in diagonal form. In addition, the eigenvalues of $\mathcal{V}$ all have multiplicity m and are given by

$$\lambda_j = -1 - \Delta t \gamma \lambda_{\text{heat},j}$$

where $\lambda_{\text{heat},j}$ are the eigenvalues of D, and the corresponding eigenspace associated to λ_j has dimension 1.

We start by considering the case of Implicit Euler ($\gamma = 1$, $m = 1$) for $\Delta t = 5 \times 10^{-2}, N = 1$. Since now $G_A = G_R = 0$ as discussed above Algorithm 3.1 coincides with Algorithm 2.2 and thus Proposition 2.3 applies. This is indeed the case up to round-off errors as we can see in Figure 2 where we plot the error between the iterations x_k for Algorithm 3.1 and the exact solution $\boldsymbol{Y}_{n+1}$ calculated using standard Newton iteration.

If we now consider the case of SDIRK4 ($\gamma = 1/4$ and $m = 5$), $G_A \neq 0$ and then (3.2) even though linear in $\boldsymbol{Y}$ is not of gradient form. As we can see in Figure 2, and we also prove in Theorem 4.2, the partitioning (3.5) is essential for ensuring the convergence of iterates, since the iterates calculated with the non-partitioned Algorithm 2.2 (corresponding to Algorithm 3.1 with the definition of $G(\boldsymbol{x})$ replaced



(a) $\Delta x = 10^{-2}, \kappa = 0.2 \cdot 10^4$. (b) $\Delta x = 10^{-3}, \kappa = 0.2 \cdot 10^6$.

Fig. 2: Behaviour of the error for Algorithm 3.1 and the non-partitioned method

by $G(\boldsymbol{x}) = G_D(\boldsymbol{x}) + G_A(\boldsymbol{x})$ and with $G_R(\boldsymbol{x}) = 0$, or equivalently applying Algorithm 2.2 to equation (3.2) with $F_A = F_R = 0$) suffer from high-round off errors that deteriorate as stiffness increases (see Appendix A for details). In contrast, iterates calculated with Algorithm 3.1 do not suffer from these issues.

We explain this difference in depth in Section A, but in a nutshell, the issue is caused by the fact that the matrix $\mathcal{V}$ in (4.5) is non-symmetric when $m \geq 2$ which creates issues for the non-partitioned method. In the case of the partitioning (3.5), we obtain

$$G_D(\boldsymbol{Y}) = \mathbb{1}_m \otimes y_n + \mathcal{V}_1 \boldsymbol{Y}, \qquad G_A(\boldsymbol{Y}) = \mathcal{V}_2 \boldsymbol{Y}, \qquad G_R(\boldsymbol{Y}) = 0,$$

where the Jacobian $\mathcal{V}$ is decomposed in two parts $\mathcal{V} = \mathcal{V}_1 + \mathcal{V}_2$ as

$$\begin{aligned} \mathcal{V}_1 &= -I_m \otimes \hat{\mathcal{V}}_1, \quad \hat{\mathcal{V}}_1 = (I_d + \gamma \Delta t D), \\ \mathcal{V}_2 &= -(\mathcal{A} - \gamma I_m) \otimes \hat{\mathcal{V}}_2, \quad \hat{\mathcal{V}}_2 = -\Delta t D. \end{aligned}$$

By construction, $\mathcal{V}_1$ is symmetric and has the same eigenvalues as $\mathcal{V}$, while $\mathcal{V}_2$ is non-symmetric and nilpotent.

The iterates $\boldsymbol{x}_k$ for Algorithm 3.1 satisfy the following recurrence relation

$$\begin{aligned} \boldsymbol{x}_{k+1} &= R_s(h\mathcal{V}_1)\boldsymbol{x}_k + hB_s(h\mathcal{V}_1)(\mathbb{1}_m \otimes y_n + \mathcal{V}_2 \boldsymbol{x}_k) \\ &= \boldsymbol{x}_k + hB_s(h\mathcal{V}_1)(\mathbb{1}_m \otimes y_n + \mathcal{V}\boldsymbol{x}_k) \end{aligned} \tag{4.6}$$

where $B_s(z) = (R_s(z) - 1)/z$, and we used $R_s(z) = 1 + B_s(z)z$. It is important to note that the steady state of (4.6) coincides with the steady state of (4.5) as desired. Furthermore, as we can see in (4.6), only the symmetric part of the matrix $\hat{\mathcal{V}}$ is taken as an argument of the stability function R_s, while the non-diagonalizable part $\hat{\mathcal{V}}_2$ is stabilised by B_s. This is, in fact, key for establishing the convergence properties of Algorithm 3.1 as detailed in the next section.

Remark 4.1. Another benefit of the splitting $\mathcal{V} = \mathcal{V}_1 + \mathcal{V}_2$ is that it allows for a parallel implementation of the stabilization terms $R_s(h\mathcal{V}_1)$ and $B_s(h\mathcal{V}_1)$ over m

processors due to their block diagonal structures. Note that a similar splitting was proposed in [20] with the motivation of parallelizing the iterates of a quasi-Newton implementation of SDIRK methods.

4.1.2. Theoretical analysis of Algorithm 3.1.

THEOREM 4.2. *Consider Algorithm 3.1 for the implementation of an m-stage SDIRK method applied to ODE* (1.1) *where F_D, F_A, F_R are given by* (4.2). *Then, for all $\varepsilon > 0$ there exists C_ε (independent of the dimension d) such that the iterates $\boldsymbol{x}_k$ of Algorithm 3.1 converge to the SDIRK solution $\boldsymbol{Y}_{n+1}$ of* (2.9) *with the following convergence estimate,*

$$\|\boldsymbol{x}_k - \boldsymbol{Y}_{n+1}\|_2 \le C_\varepsilon\big(\alpha_s(\eta) + \varepsilon\big)^k \|\boldsymbol{x}_0 - \boldsymbol{Y}_{n+1}\|_2. \tag{4.7}$$

Proof. Using (4.6), we obtain,

$$\boldsymbol{x}_{k+1} - \boldsymbol{Y}_{n+1} = M(\boldsymbol{x}_k - \boldsymbol{Y}_{n+1}), \qquad M = R_s(h\mathcal{V}_1) + B_s(h\mathcal{V}_1)h\mathcal{V}_2, \tag{4.8}$$

where $B_s(z) = (R_s(z) - 1)/z$. In order to prove (4.7) we need to have an estimate of $\|M\|$. The main difficulty in doing so is the term $\mathcal{V}_2$, as this is not diagonalizable. To address this issue, we construct an appropriate change of coordinates which makes this term negligible. Precisely, we consider the Jordan decomposition of the Runge-Kutta matrix $\mathcal{A}$ of coefficients, $\mathcal{A} = P\mathcal{J}P^{-1}$ where $P \in \mathbb{R}^{m\times m}$ is an invertible matrix, and $\mathcal{J} \in \mathbb{R}^{m\times m}$ is a Jordan block matrix of the form

$$\mathcal{J} = \begin{pmatrix} \gamma & 1 & 0 & \cdots & 0 \\ 0 & \gamma & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & & \\ 0 & 0 & & \ddots & 1 \\ 0 & 0 & & \cdots & \gamma \end{pmatrix} = \gamma I_m + F. \tag{4.9}$$

Here, we assume for simplicity of the presentation that there is a single Jordan block $\mathcal{J}$, but the proof generalises straightforwardly to the case of multiple Jordan blocks.

Now let $Q_\varepsilon = \mathrm{diag}(1, \varepsilon, \varepsilon^2, \ldots, \varepsilon^{m-1})$. Using $Q_\varepsilon^{-1} F Q_\varepsilon = \varepsilon F$, we deduce $Q_\varepsilon^{-1}P^{-1}\mathcal{A}PQ_\varepsilon = \gamma \mathbb{1}_m + \varepsilon F$ and

$$(Q_\varepsilon^{-1}P^{-1} \otimes I_d)M(PQ_\varepsilon \otimes I_d) = R_s(h\mathcal{V}_1) - \varepsilon B_s(h\mathcal{V}_1)[F \otimes h\hat{\mathcal{V}}_2],$$

where we have used that $(A \otimes B)(C \otimes D) = AC \otimes BD$ and that the matrix $PQ_\varepsilon \otimes I_d$ and its inverse both commute with $R(h\mathcal{V}_1) = I_m \otimes R_s(h\hat{\mathcal{V}}_1)$ and $B_s(\mathcal{V}_1)$. This yields

$$\begin{aligned} \|\boldsymbol{x}_k - \boldsymbol{Y}_{n+1}\|_2 &= \|M^k(\boldsymbol{x}_0 - \boldsymbol{Y}_{n+1})\|_2 \\ &\le \kappa_2(PQ_\varepsilon)(\|R(h\hat{\mathcal{V}}_1)\|_2 + \varepsilon\|F\|_2\|B_s(h\hat{\mathcal{V}}_1)h\hat{\mathcal{V}}_2\|_2)^k\|\boldsymbol{x}_0 - \boldsymbol{Y}_{n+1}\|_2 \end{aligned} \tag{4.10}$$

where $\kappa_2(PQ_\varepsilon) = \|PQ_\varepsilon\|_2\|(PQ_\varepsilon)^{-1}\|_2$ is the condition number of the matrix $PQ_\varepsilon \in \mathbb{R}^{m\times m}$, which is independent of d. We now have (see [12, Lemma A.1])

$$\|R_s(h\hat{\mathcal{V}}_1)\|_2 \le \alpha_s(\eta) < 1, \qquad \|B_s(h\hat{\mathcal{V}}_1)\|_2 \le 1, \tag{4.11}$$

In addition, we have $\|F\|_2 = 1$, $\hat{\mathcal{V}}_2 = \gamma^{-1}(I_d - \hat{\mathcal{V}}_1)$, and $h \le \eta e^\eta/\ell$ using (2.23), which yields $\|B(h\hat{\mathcal{V}}_1)h\hat{\mathcal{V}}_1\|_2 = \|R_s(h\hat{\mathcal{V}}_1) - I_d\|_2 \le 2$, and

$$\|B_s(h\hat{\mathcal{V}}_1)h\hat{\mathcal{V}}_2\|_2 \le \|h\hat{\mathcal{V}}_2\|_2 \le \gamma^{-1}(2 + \eta e^\eta/\ell) \tag{4.12}$$

and hence

$$\begin{aligned}\|\boldsymbol{x}_k - \boldsymbol{Y}_{n+1}\|_2 &\leq \kappa_2(PQ_\varepsilon)(\alpha_s(\eta) + \varepsilon\gamma^{-1}(2+\eta e^\eta/\ell))^k \|\boldsymbol{x}_0 - \boldsymbol{Y}_{n+1}\|_2 \\ &\leq C_\varepsilon(\alpha_s(\eta)+\varepsilon)^k \|\boldsymbol{x}_0 - \boldsymbol{Y}_{n+1}\|_2\end{aligned}$$

where $C_\varepsilon = \kappa_2(PQ_{\varepsilon'})$, with $\varepsilon' = \varepsilon\gamma(2+\eta e^\eta/\ell)^{-1}$ is a constant independent of the dimension d and this concludes the proof. ∎

In the proof of Theorem 4.2, observe that one can improve the estimate (4.12) for the quantity $B(h\hat{\mathcal{V}}_1)h\hat{\mathcal{V}}_2$, but this is not useful here because ε can be chosen arbitrarily small.

Remark 4.3. In the case of the implicit Euler method as we discussed before Algorithm 3.1 is equivalent to Algorithm 2.2. This is reflected in the statement of Theorem 4.2 since ε=0, $C_\varepsilon = 1$ in (4.7) which agrees with (2.21) in Proposition 2.3.

4.2. The case of linear advection-diffusion. We now consider the following linear advection-diffusion equation

$$\partial_t u = a\Delta u + b\cdot\nabla u, \quad (t,x) \in [0,+\infty)\times(0,1)^N \tag{4.13}$$

with periodic boundary conditions and a smooth initial condition $u(x,0)$, where $a>0$ and $b \in \mathbb{R}_+^N$ are fixed. We now discretise in space using a finite difference method with a uniform mesh size Δx in each spatial dimension. We thus have

$$F_D(y) = -Dy, \quad F_A(y) = Ey \quad F_R(y) = 0. \tag{4.14}$$

where D is the discrete Laplacian with periodic boundary conditions and E is the discrete centred advection operator. Following the presentation in [22] the eigenvalues $\lambda_{\text{heat-adv},j} = \lambda_{\text{heat},j} + i\mu_{\text{adv},j}$ of $F = F_D + F_A$ are located on the following ellipse in the complex plane

$$\left(\frac{2p}{\alpha}+1\right)^2 + \left(\frac{q}{\beta}\right)^2 = 1, \tag{4.15}$$

with half-height $\beta = \|b\|_1 \Delta x^{-1}$ and width $\alpha = 4aN\Delta x^{-2}$, and where $p = \lambda_{\text{heat},j} \in [-\alpha, 0]$ located on the negative real axis are the eigenvalues of F_D and $iq = i\mu_{\text{adv},j} \in [-i\beta, i\beta]$ on the imaginary axis are the eigenvalues of F_A.

Before we present a theorem for an m-stage SDIRK method it is useful to introduce the following function

$$\mathcal{R}_s(P,Q) = R_s(P) + B_s(P)Q. \tag{4.16}$$

that will play an important role in establishing the stability and convergence properties in the general case. For simplicity, we start by considering the implicit Euler case for which the iteration error for Algorithm 3.1 satisfies

$$x_{k+1} - y_{n+1} = \mathcal{R}_s(-hI_d - h\Delta t D, hE)(x_k - y_{n+1}) \tag{4.17}$$

and consider the related stability domain

$$\mathcal{S}_\gamma = \{(p,q)\in\mathbb{R}^2 \ ; \ R_{\text{stab}}(\gamma p, q) < 1\}. \tag{4.18}$$

with $\gamma = 1$ for the implicit Euler method and where

$$R_{\text{stab}}(p,q) \coloneqq |R_s(-h+hp)|^2 + |B_s(-h+hp)|^2 |hq|^2$$

We first prove the stability property of Algorithm 3.1 in the simplest case of the implicit Euler method and based on the stability domain (4.18). We then show that it also implies the convergence property of Algorithm 3.1 for an m-stage SDIRK method.

PROPOSITION 4.4. *The error $\|x_{k+1} - y_{n+1}\|_2$ for the implicit Euler method given by* (4.17) *converges to zero as $k \to +\infty$ if for all $p_j = -\Delta t \lambda_{heat,j}$, $q_j = \Delta t \mu_{adv,j}, j = 1, \ldots, d$, we have $(p_j, q_j) \in \mathcal{S}_1$. In this case, we have*

$$\|x_{k+1} - y_{n+1}\|_2 \leq \rho \, \|x_k - y_{n+1}\|_2 , \qquad \rho := \max_{j=1,\cdots,d} R_{stab}(p_j, q_j)^{1/2} < 1.$$

Proof. If $p_j, q_j \in \mathcal{S}_1$ for all j we have that $R_{\rm stab}(p_j, q_j) < 1$. Now, since we assume periodic boundary conditions, the matrices D, E commute which implies that $\|\mathcal{R}_s(-hI_d - h\Delta t D, hE)\|_2 = \rho < 1$, which yields the convergence for the error $\|x_{n+1} - y_{n+1}\|_2$ in (4.17). ∎

In proving Proposition 4.4, we have explicitly taken into account that because of the periodic boundary conditions, the matrices D and E are commuting. This will, in general, not hold, for example in the case of Neumann or Dirichlet boundary conditions. It is still possible to establish a stability criterion, as stated in the next proposition.

PROPOSITION 4.5. *Consider* (1.1) *with F_D, F_A, F_R given in* (4.14) *and where D is a given positive semi-definite matrix and E is skew symmetric matrix with bound $\|E\| \leq \|b\|_1 \Delta x^{-1}$. Then the error $\|x_{k+1} - y_{n+1}\|_2$ for Algorithm 3.1 applied with the implicit Euler method given by* (4.17) *converges to zero as $k \to +\infty$ if*

$$\hat{\rho} := \alpha_s(\eta) + \frac{\eta e^{\eta}}{\ell} \|b\|_1 \frac{\Delta t}{\Delta x} < 1. \tag{4.19}$$

In this case we have

$$\|x_{k+1} - y_{n+1}\|_2 \leq \hat{\rho} \, \|x_k - y_{n+1}\|_2 .$$

Proof. Using (4.11) and $\|E\| \leq \|b\|_1 \Delta x^{-1}$, we obtain from (4.17), $\|\mathcal{R}_s(-hI_d - h\Delta t D, hE)\|_2 \leq \alpha_s(\eta) + h\Delta t \|E\| \leq \hat{\rho} < 1$ where we use (2.23),(4.19) and the convergence holds. ∎

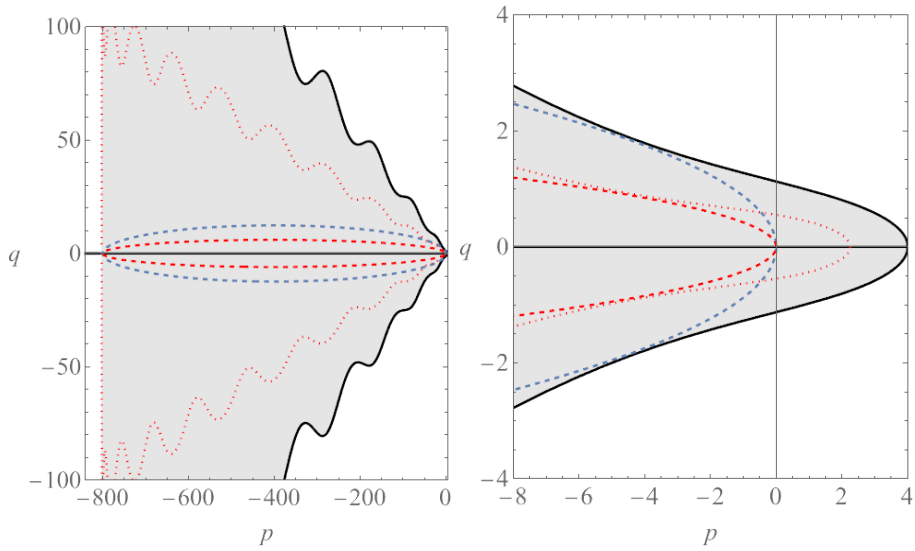
We now state the general result, that the estimate for implicit Euler generalises for an arbitrary m-diagonal SDIRK method.

THEOREM 4.6. *Consider Algorithm 3.1 for the implementation of an m-stage SDIRK Runge-Kutta method applied to ODE* (1.1) *where F_D, F_A, F_R are given by* (4.14)*. Assume that D and E satisfy either the assumptions of Proposition 4.5 or the stronger assumptions of Proposition 4.4, and assume that at least one of the following bounds hold*

$$\hat{\rho} := \alpha_s(\eta) + \frac{\eta e^{\eta}}{\ell} \|b\|_1 \frac{\Delta t}{\Delta x} < 1,$$
$$\rho := \max_{j=1,\ldots,d} \left(\mathcal{R}_s(-h - h\gamma \Delta t \lambda_{heat,j}, h\Delta t \mu_{adv,j})^{1/2} \right) < 1.$$

Then, for all $\varepsilon > 0$ there exists C_ε (independent of the dimension d) such that the iterates $\boldsymbol{x}_k$ of Algorithm 3.1 converge to the SDIRK solution $\boldsymbol{Y}_{n+1}$ of (2.9) *with the following convergence estimate,*

$$\|\boldsymbol{x}_k - \boldsymbol{Y}_{n+1}\|_2 \leq C_\varepsilon \left(\varepsilon + \min(\rho, \hat{\rho})\right)^k \|\boldsymbol{x}_0 - \boldsymbol{Y}_{n+1}\|_2. \tag{4.20}$$



(a) Stability domain (b) Zoomed version

Fig. 3: Stability regions $\mathcal{S}_\gamma$ in (4.18) with $\gamma = 1/4$ of the diffusion-advection coupling for $s = 20$ and damping parameters $\eta = 4$. We also include the largest ellipse of the form (4.15) tangent to the y-axis that can be included in the stability region $\mathcal{S}_\gamma$ in (4.18) (blue dashed lines) where the iteration error decays, and in the stability region $\widehat{\mathcal{S}}_\gamma$ in (4.22) where the iteration error is halved at each iteration.

Proof. Following the lines of the proof of Theorem 4.2, we obtain

$$\boldsymbol{x}_{k+1} - \boldsymbol{Y}_{n+1} = (M + \tilde{M})(\boldsymbol{x}_k - \boldsymbol{Y}_{n+1}).$$

Here $M = R_s(h\mathcal{V}_1) + B_s(h\mathcal{V}_1)(h\mathcal{V}_2)$ is exactly the same matrix that appears in Theorem 4.2 while the new term $\tilde{M} = B_s(h\mathcal{V}_1)(\mathcal{A} \otimes h\Delta t E)$ arises due to the presence of advection. Now using the same change of coordinates as in Theorem 4.2 we have that

$$\begin{aligned}(Q_\varepsilon^{-1}P^{-1} \otimes I_d)(M + \tilde{M})(PQ_\varepsilon \otimes I_d) = I_m \otimes R(h\hat{\mathcal{V}}_1) + \varepsilon F \otimes B(h\hat{\mathcal{V}}_1)h\hat{\mathcal{V}}_2 \\ + (\gamma I_m + \varepsilon F) \otimes B(h\hat{\mathcal{V}}_1)h\Delta t E,\end{aligned}$$

which yields

$$\begin{aligned}\|\boldsymbol{x}_k - \boldsymbol{Y}_{n+1}\|_2 \le C_\varepsilon \Big(&\|\mathcal{R}_s(-hI_d - h\Delta t\gamma D, hE)\|_2 \\ &+ \varepsilon\|F\|_2(\|B_s(h\hat{\mathcal{V}}_1)h\hat{\mathcal{V}}_2\|_2 + h\Delta t\|B_s(h\hat{\mathcal{V}}_1)\|_2\|E\|_2)\Big)^k \|\boldsymbol{x}_0 - \boldsymbol{Y}_{n+1}\|_2.\end{aligned}$$

Now proceeding in a similar way as the end of the proof of Theorem 4.2, we obtain the desired estimate (4.20). ∎

Note that in the absence of advection ($b = 0$), the estimate of Theorem 4.6 reduces to the one of Theorem 4.2 with $\hat{\rho} \le \rho = \alpha_s(\eta)$.

We now plot in Figure 3 the stability domain $\mathcal{S}_\gamma$ given by (4.18) with $\gamma = 1/4$ for $s = 20$ and $\eta = 4$. This domain covers a large portion of the negative real axis

as expected from the properties of the Chebyshev polynomials. On the other hand, the width on the q-axis near the origin is much narrower. The blue ellipse in Figure 3 indicates the largest ellipse of the form (4.15) that contains the origin with principal axes parallel to the p-q axes and that can be included in the stability domain $\mathcal{S}_\gamma$. If we denote by d_s and a_s respectively the width and the half-height of this ellipse we observe that d_s grows quadratically with the number of stages s while a_s grows linearly with s as illustrated in Figure 4. More precisely, we find numerically for this range of parameters that d_s and a_s grow like

$$d_s = 2s^2, \qquad a_s = 0.62s. \tag{4.21}$$

For comparison, we also plot in in Figure 3 in red dotted lines the largest ellipse of the form (4.15) that is included in the modified stability domain

$$\widehat{\mathcal{S}}_\gamma = \{(p,q) \in \mathbb{R}^2 \ ; \ R_{\rm stab}(p,q) < 1/2\} \tag{4.22}$$

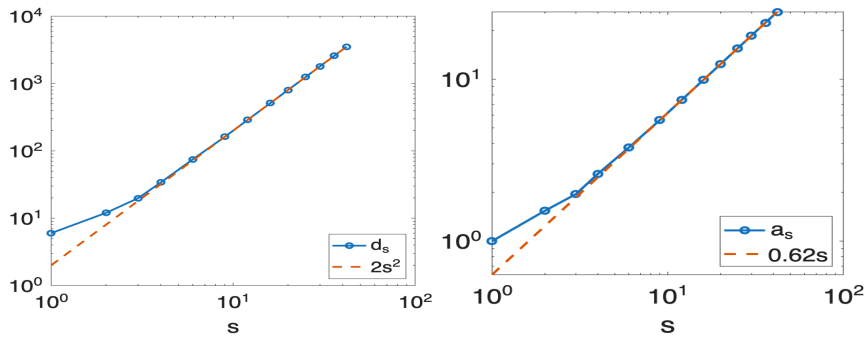
where the iteration error is at least halved at each iteration using $\hat{\rho} < 1/2$ in (4.20). This is different to blue ellipse corresponding to the stability domain $\mathcal{S}$ in (4.18) where for $\hat{\rho} < 1$ only the decay of the iteration error is guaranteed. This illustrates the favorable stabilization of the explicit stabilised iteration taking advantage of the diffusion term even for advection terms with relatively large Peclet numbers. Additionally, note that in the case where $\rho < 1$ in Theorem 4.6 is an appropriate bound for an SDIRK method, the behaviour of the stability function of the implicit Euler studied in Figures 3-4 dictates the choice of parameters, with the only difference is that the stability domain size is scaled with respect to the diffusion (horizontal p-axis in Figure 3) by a factor γ^{-1}

Remark 4.7. As emphasized in [5, Remark 3.4] for the stability analysis of the PIROCK method for advection-diffusion problems, the linear growth in (4.21) of the ellipse half-height is a feature of the proposed explicit stabilised implementation of SDIRK methods as it allows for relatively large Peclet numbers. Indeed, considering the advection-diffusion problem (4.13) with spatial discretization (4.14). Substituting $h\Delta t\alpha = 2s^2$ into $h\Delta t\beta \leq \max(0.62s, 1)$ with $h \simeq 1.40$ yields $\Delta t\beta \leq \max(\frac{0.62^2}{2\cdot 1.40}\alpha/\beta, 1)$. Using $\beta = \|b\|_1 \Delta x^{-1}$, $\alpha = 4aN\Delta x^{-2}$ we deduce the the stability sufficient condition

$$\Delta t \leq \max(0.55aN/\|b\|_1, \Delta x/\|b\|_1),$$

which is a stability constraint on the stepsize Δt independent of the spatial grid size small enough ($\Delta x \leq 0.55aN$). This is a feature of the proposed partitioned explicit stabilised implementation, shared with the PIROCK method satisfying an analogous stability sufficient condition (see [5, Remark 3.4]), in contrast to the other stabilised explicit integrators RKC, PRKC and ROCK2 where only small Peclet numbers can be considered.

5. Numerical tests. In this section, we present a number of numerical experiments on the Brusselator problem for the SDIRK4 method. More precisely, we study the case of a highly stiff reaction as well as a large advection setting. We denote with exSDIRK4 the proposed explicit stabilised implementation of SDIRK4, with MATLAB prototype codes made publicly available at [8], and we compare its performance with the order 2 PIROCK method. Note that established explicit methods of order 4 such as ROCK4 [1], are not suitable for the two problems studied here due to the stiff nature of the advection and reaction terms. In all our numerical experiments, the time step Δt is chosen adaptively as described in the remark below.



(a) Major axis d_s (b) Minor radius a_s

Fig. 4: Stability of the diffusion-advection coupling for $\gamma = 1/4, s = 20$ and damping parameters $\eta = 4$. Growth with respect to the stage parameter s of the width d_s and the half-height a_s in (4.21) of the largest ellipse (4.15) tangent to the y-axis that can be included in the stability region $\mathcal{S}_\gamma$ in (4.18) with $\gamma = 1/4$.

Remark 5.1. Compared to the standard embedded error estimator for stiff problems recalled in Remark 2.2, see also [13, Chap. IV.6], the only difference for the explicit stabilised implementation is that the inverse Jacobian term J^{-1} involved in (2.12) is no longer available. This is indeed the main feature of the proposed approach to avoid such a possibly large dimension Jacobian and its inverse, which can raise linear algebra issues. Alternatively to (2.12), we implement the Shampine trick using the following error estimator of the exSDIRK4 method,

$$err = err_s$$

where we define $\overline{err} = J^{-1}(y_1 - \hat{y}_1)$ where $J = I_d - h\,\Delta t\gamma F_R'(y_1)$ is the Jacobian involved for the reaction term, and the err_j are computed by induction on $j \in \{1, \cdots, s\}$ using $err_0 = err_{-1} = 0$ as

$$err_j = \mu_j \left(h\Delta t\gamma(F_D(y_1 + err_{j-1}) - F_D(y_1)) + \overline{err}\right) + \nu_j err_{j-1} - (\nu_j - 1)err_{j-2},$$

where $\mu_1 = \omega_1/\omega_0, \nu_1 = 1$ and μ_j, ν_j and the stage number s are defined in Algorithm 3.1 (see (3.4)). We emphasize that the above error estimator benefits from L-stability with respect to diffusion and reaction terms in the spirit of the the standard estimator (2.12). Indeed, firstly in the absence of diffusion and reaction ($F_D = F_A = 0$) it reduces to $err = \overline{err} = J^{-1}(y_1 - \hat{y}_1)$ which exactly coincides with (2.12), and secondly in the case of a linear diffusion $F_D = Dy$, it yields $err_s = B_s(h\Delta t\gamma D)J^{-1}(y_1 - \hat{y}_1)$ analogously to (4.6) and where we recall the bound $|B_s(z)| \leq C/|z|$.

Throughout the numerical experiments we make the following implementation assumption: the evaluations of $F_D[\boldsymbol{Y}]$, $F_A[\boldsymbol{Y}]$, and $F_R[\boldsymbol{Y}]$ for the m internal stages, are computed in *parallel*. This is natural given the block-diagonal structure of the stabilisation operators $R_s(h\mathcal{V}_1)$ and $B_s(h\mathcal{V}_1)$, which decouple across stages, and reflects the remark following Theorem 4.2. For a vector $v \in R^d$ that represents a discretized function $f : \mathbb{R}^d \to \mathbb{R}^d$, we define the discrete L^2-norm by

$$\|v\|_{L^2} = \sqrt{\frac{\sum_{i=1}^d v_i^2}{d}}.$$

Remark 5.2. As we mentioned before, for simplicity, we use $\mathbb{1} \otimes y_n$ as initial guess in Algorithm 3.1 and, to coincide with the true solution of SDIRK4, we choose a fixed $tol_{it} = 10^{-12}$ as optimisation tolerance. However, the cost of exSDIRK4 discussed in the numerical tests below can be further reduced by choosing a better initial guess using interpolation or dense output techniques, and by choosing tol_{it} smaller than the relative tolerance by only one or two orders of magnitude.

5.1. 1D Brusselator problem with highly stiff reaction. Consider the following Brusselator problem

$$\begin{aligned} \partial_t u &= \alpha \Delta u + A + u^2 v - (B+1)u, \\ \partial_t v &= \alpha \Delta v - u^2 v + Bu, \\ u(0,t) &= u(1,t), \quad v(0,t) = v(1,t), \end{aligned} \tag{5.1}$$

where $x \in (0,1)$, $t \in (0,T)$, $\alpha = 0.2$, $A = 1$, $B = 3 \times 10^7$, and $T = 1$. We consider the following initial condition

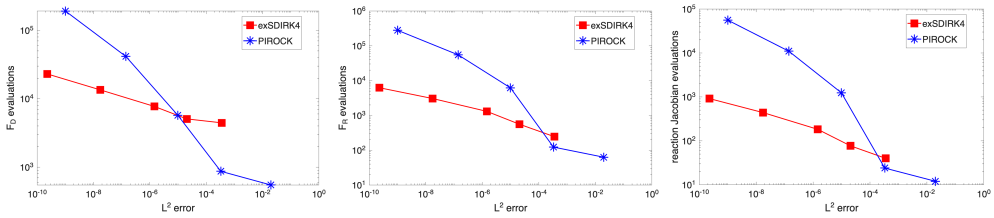
$$u(x,0) = 1 + \sin(2\pi x) \quad \text{and} \quad v(x,0) = 3,$$

and periodic boundary conditions. We use second order finite differences for the space discretisation with mesh size $\Delta x = 1/200$, and the starting time step $\Delta t_0 = 10^{-6}$.

Table 1 and Figure 5 report the cost comparison between exSDIRK4 and PIROCK for the 1D Brusselator problem (5.1). For loose tolerances ($tol = 10^{-1}$ and 10^{-3}), PIROCK requires fewer function evaluations overall; however, exSDIRK4 already delivers a substantially lower error for the same requested tolerance, reflecting its higher order of accuracy. As the tolerance tightens, the balance shifts decisively in favour of exSDIRK4. At $tol = 10^{-5}$, both methods have a comparable number of F_D evaluations, but exSDIRK4 requires more than 4 times fewer F_R evaluations and 6 times fewer F_R' evaluations/inversions. This advantage grows rapidly: at $tol = 10^{-7}$, PIROCK needs approximately 3 times more F_D, 18 times more F_R, and 24 times more F_R' evaluations/inversions than exSDIRK4. At the tightest tolerance $tol = 10^{-9}$, PIROCK accumulates 55,888 accepted steps with 2,928 rejections, while exSDIRK4 completes the integration in only 468 accepted steps with 5 rejections—a reduction of

Method	tol	F_D evals	F_R evals	F_R' evals	acc(rej)	error
exSDIRK4	10^{-1}	4447	245	40	20(0)	3.5×10^{-4}
PIROCK	10^{-1}	548	63	12	12(0)	2×10^{-2}
exSDIRK4	10^{-3}	5072	559	77	42(3)	2.0×10^{-5}
PIROCK	10^{-3}	870	123	24	24(0)	3.3×10^{-4}
exSDIRK4	10^{-5}	7718	1307	183	102(3)	1.4×10^{-6}
PIROCK	10^{-5}	5719	6205	1239	1234(6)	9.9×10^{-6}
exSDIRK4	10^{-7}	13463	3053	439	231(4)	1.7×10^{-8}
PIROCK	10^{-7}	41608	54967	10992	10987(6)	1.4×10^{-7}
exSDIRK4	10^{-9}	23056	6248	915	468(5)	2.2×10^{-10}
PIROCK	10^{-9}	191700	279447	55888	55888(2928)	1.0×10^{-9}

Table 1: Comparison between our explicit stabilised implementation of SDIRK4 and PIROCK for different prescribed tolerances.



(a) F_D evaluations. (b) F_R evaluations. (c) F'_R evaluations

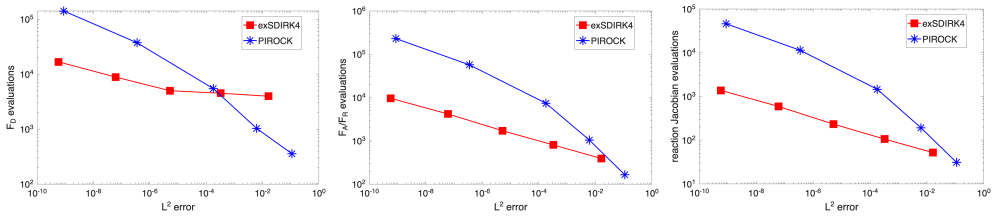
Fig. 5: Cost comparison between exSDIRK4 and PIROCK for Problem (5.1)

more than two orders of magnitude in the number of steps. These trends are clearly visible in Figure 5, which displays the number of F_D, F_R, and F'_R evaluations as a function of the tolerance. The work-precision curves show that the crossover point beyond which exSDIRK4 is uniformly cheaper occurs around $tol \approx 10^{-4}$, and the gap widens as the tolerance decreases, which is consistent with the high-order convergence of exSDIRK4.

5.2. 2D Brusselator with large advection. We now consider the following Brusselator model in 2D:

$$\begin{aligned} \partial_t u &= \nu \Delta u + \mu U \cdot \nabla u + A + u^2 v - (B+1)u, \\ \partial_t v &= \nu \Delta v + \mu V \cdot \nabla v - u^2 v + Bu, \end{aligned} \tag{5.2}$$

where $x \in (0,1)^2$, $t \in (0,T)$, $\nu = 0.1$, $\mu = 2$ $A = 1.3$, $B = 10^7$, $U = (-0.5, 1)^T$, $V = (0.4, 0.7)^T$ and $T = 0.5$. Following [5], we choose $u(x,0) = 22x_2(1-x_2)^{3/2}$, $v(x,0) = 27x_1(1-x_1)^{3/2}$, and periodic boundary conditions. For the space discretisation, we use again second order finite differences with mesh size equals to 10^{-2} in both dimensions and the starting time step $\Delta t_0 = 10^{-6}$.



(a) F_D evaluations. (b) F_A and F_R evaluations. (c) F'_R evaluations.

Fig. 6: Cost comparison between exSDIRK4 and PIROCK for Problem (5.2)

Figure 6 reports analogous results for the 2D Brusselator problem with large advection. The qualitative behaviour is very similar to the 1D case: exSDIRK4 is outperformed by PIROCK for loose tolerances in terms of raw function evaluation counts, but becomes significantly more efficient as the tolerance tightens. The three panels—F_D evaluations, F_A/F_R evaluations, and F'_R evaluations—all exhibit the same crossover pattern observed in Figure 5, confirming that the efficiency advantage of

exSDIRK4 is not an artefact of the one-dimensional setting but carries over to higher-dimensional problems with combined diffusion, advection, and stiff reaction terms.

6. Conclusion. We introduced an explicit stabilised implementation of singly diagonally implicit Runge-Kutta (SDIRK) methods for advection-diffusion-reaction PDEs, with the fourth-order method SDIRK4 as the primary example. The central idea is to reformulate the implicit Runge-Kutta stage equations (2.9) as the steady state of an auxiliary ODE (3.1), and to compute that steady state by a partitioned Runge-Kutta-Chebyshev iteration (Algorithm 3.1) inspired by the RKCD optimisation method of [12]. The partition of the Runge-Kutta coefficient matrix $\mathcal{A} = \gamma I_m + (\mathcal{A} - \gamma I_m)$ is essential: it isolates a symmetric, block-diagonal stiff part G_D that governs the stability of the iteration through the stability polynomial R_s, avoiding the non-diagonalisable amplification matrix that would otherwise prevent convergence, as analysed in detail in Section A. The off-diagonal diffusion and advection contributions enter through the damping polynomial B_s, whose extended stability along the imaginary axis accommodates large advection velocities under a mild CFL-like condition on the iteration step size h — significantly more permissive than the classical CFL restriction on the physical time step Δt. The stiff reaction term is handled implicitly at each inner iteration via a local inversion of $J = I_{md} - hG'_R(\boldsymbol{x}_k)$, which remains computationally inexpensive because the reaction Jacobian carries no spatial coupling between grid points and is therefore block-diagonal. The result is a method that requires no large-scale Newton solves or global linear algebra, and yet achieves the high-order accuracy and strong stability of a fourth-order L-stable integrator. The only implicit component is this local reaction Jacobian inversion, in sharp contrast to the dense, high-dimensional Jacobian systems that a standard Newton-based implementation of SDIRK4 would require.

The convergence and stability properties of the new method are rigorously established in two settings. In the pure linear diffusion case (Theorem 4.2), we prove that Algorithm 3.1 converges with decay factor $\alpha_s(\eta) + \varepsilon$ for any $\varepsilon > 0$, independently of the spatial dimension d, and that implementing one step of an m-stage SDIRK method requires only $\mathcal{O}(\sqrt{\Delta t}/\Delta x)$ evaluations of F_D — the same asymptotic cost as a standard first-order explicit stabilised method. In the linear advection-diffusion case (Theorem 4.6 and Propositions 4.4–4.5), we show that convergence is maintained provided the advection-to-diffusion ratio satisfies a mild CFL-like condition, a condition that is significantly more permissive than the one imposed by a classical explicit scheme on the physical time step Δt.

Assuming parallel implementation of the function evaluations across the method stages, the numerical experiments on the 1D and 2D Brusselator problems with highly stiff reaction confirms the theoretical predictions and demonstrate a clear advantage over the second-order method PIROCK [5] whenever tight tolerances are required. The method occupies a natural middle ground: more implicit than a pure explicit stabilised method such as PIROCK, but far cheaper than a fully Newton-based implicit solver, and this balance is precisely what drives its efficiency at high accuracy demands.

The present framework relies on the lower-triangular, singly diagonal structure of SDIRK methods to decouple the implicit system stage by stage and to identify the symmetric stabilising part G_D. A natural and challenging next step would be to extend the approach to fully implicit Runge-Kutta methods such as Radau IIA or Gauss-Legendre collocation schemes. A major difficulty for their implementation is their full matrix structure of the Runge-Kutta coefficients $\mathcal{A}$, and studied for instance

in [14, 11] where splitting approaches based on triangular matrices and low rank linear algebra techniques are proposed. In such fully implicit case, the Runge-Kutta matrix $\mathcal{A}$ is a full, non-triangular matrix with non-real eigenvalues, so the partitioning strategy proposed in this paper and the associated convergence analysis do not apply straightforwardly.

Acknowledgments. The authors would like to thank Ernst Hairer and Francesca Mazzia for helpful discussions on the implementation of SDIRK and RADAU methods. G.V. was partially supported by the Swiss National Science Foundation, projects No. 200020_214819, No. 200020_192129 and No. 10009199. I.A. was partially supported by the Swiss National Science Foundation, projects No. P500PT_210968 and No. P5R5PT_222213.

REFERENCES

[1] A. Abdulle. Fourth order Chebyshev methods with recurrence relation. *SIAM J. Sci. Comput.*, 23(6):2041–2054, 2002.

[2] A. Abdulle. *Explicit Stabilized Runge–Kutta Methods*, pages 460–468. Encyclopedia of Applied and Computational Mathematics, Springer Berlin Heidelberg, 2015.

[3] A. Abdulle, I. Almuslimani, and G. Vilmart. Optimal explicit stabilized integrator of weak order 1 for stiff and ergodic stochastic differential equations. *SIAM/ASA J. Uncertain. Quantif.*, 6(2):937–964, 2018.

[4] A. Abdulle and A. Medovikov. Second order chebyshev methods based on orthogonal polynomials. *Numer. Math.*, 90(1):1–18, 2001.

[5] A. Abdulle and G. Vilmart. PIROCK: a swiss-knife partitioned implicit-explicit orthogonal Runge-Kutta Chebyshev integrator for stiff diffusion-advection-reaction problems with or without noise. *J. Comput. Phys.*, 242:869–888, 2013.

[6] I. Almuslimani, S. Ogier-Collin, P. Ulbl, and S. Brunner. Accelerating neutral-coupled gyrokinetic simulations in GENE-X via adaptive PIROCK scheme. *In preparation.*

[7] I. Almuslimani, P. Ulbl, and S. Brunner. Implementation and verification of adaptive and partitioned time-stepping schemes in the gyrokinetic code GENE-X. *In preparation.*

[8] I. Almuslimani, G. Vilmart, and K. C. Zygalakis. Explicit stabilised implementation of singly diagonally implicit Runge-Kutta methods. *Yareta [Matlab source code]*, 2026. https://doi.org/10.26037/yareta:k4s5clvpmrbr7dym3tgmirznsy.

[9] U. M. Ascher, S. J. Ruuth, and R. J. Spiteri. Implicit-explicit Runge–Kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics*, 25(2-3):151–167, 1997.

[10] U. M. Ascher, S. J. Ruuth, and B. T. R. Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM J. Numer. Anal.*, 32(3):797–823, 1995.

[11] L. Brugnano, F. Iavernaro, and C. Magherini. Efficient implementation of Radau collocation methods. *Appl. Numer. Math.*, 87:100–113, 2015.

[12] A. Eftekhari, B. Vandereycken, G. Vilmart, and K. C. Zygalakis. Explicit stabilised gradient descent for faster strongly convex optimisation. *BIT Numerical Mathematics*, 61(1):119–139, 2021.

[13] E. Hairer and G. Wanner. *Solving ordinary differential equations II. Stiff and differential-algebraic problems.* Springer-Verlag, Berlin and Heidelberg, 1996.

[14] F. Iavernaro and F. Mazzia. Solving ordinary differential equations by generalized Adams methods: properties and implementation techniques. volume 28, pages 107–126. 1998. Eighth Conference on the Numerical Treatment of Differential Equations (Alexisbad, 1997).

[15] A. Iserles. *A first course in the numerical analysis of differential equations.* Cambridge texts in applied mathematics. Cambridge University Press, Cambridge ; New York, 1996.

[16] Y. Nesterov. *Lectures on Convex Optimization*, volume None of *Springer Optimization and Its Applications.* Springer, 2 edition, December 2018.

[17] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer-Verlang New York, 2006.

[18] L. F. Shampine, B. P. Sommeijer, and J. G. Verwer. Irkc: an imex solver for stiff diffusion-reaction pdes. *J. Comput. Appl. Math.*, 196(2):485–497, Nov. 2006.

[19] P. J. van der Houwen and B. P. Sommeijer. On the internal stability of explicit, m-stage Runge-Kutta methods for large m-values. *Z. Angew. Math. Mech.*, 60(10):479–485, 1980.

[20] P. J. van der Houwen, B. P. Sommeijer, and W. Couzy. Embedded diagonally implicit Runge-Kutta algorithms on parallel computers. *Math. Comp.*, 58(197):135–159, 1992.

[21] Q. M. Wargnier, G. Vilmart, J. Martínez-Sykora, V. H. Hansteen, and B. De Pontieu. Time-adaptive PIROCK method with error control for multi-fluid and single-fluid MHD systems. *A&A*, 695:A262, 2025.
[22] C. J. Zbinden. Partitioned Runge-Kutta-Chebyshev methods for diffusion-advection-reaction problems. *SIAM J. Sci. Comput.*, 33(4):1707–1725, 2011.

Appendix A. Naive implementation. We now explain in more detail the issues observed in Figure 2 with respect to the naive stabilised explicit implementation of SDIRK4 applied without partitioning (corresponding to Algorithm 3.1 with the definition of $G(\boldsymbol{x})$ replaced by $G(\boldsymbol{x}) = G_D(\boldsymbol{x}) + G_A(\boldsymbol{x})$ with $G_R(\boldsymbol{x}) = 0$, or equivalently applying Algorithm 2.2 to equation (3.2) with $F_A = F_R = 0$). Considering for simplicity only one mode of the discrete linear heat equation, corresponding to the Dahlquist test equation $\dot{y} = \lambda y$ with $\lambda = -\Delta t \gamma \lambda_{\mathrm{heat},j} < 0$ corresponds to a possibly large parameter $|\lambda| \gg 1$. We then write one-step recursion induced by Algorithm 2.1 applied to a SDIRK method with $m > 1$ internal stages, namely

$$\mathbf{Y}_{n+1} = R_s(-hI_m + \lambda h\mathcal{A})\mathbf{Y}_n$$

where R_s given by (2.13) is the stability function of RKC, and h is the time step used by the RKC iteration. Consider the Jordan form of the matrix $\mathcal{A} = P\mathcal{J}P^{-1}$ where $\mathcal{J}$ is a Jordan block defined in (4.9). Using the following formula for any analytic function f applied to the Jordan matrix J,

$$f(\mathcal{J}) = \begin{pmatrix} f(\gamma) & f'(\gamma) & \cdots & \frac{f^{(m-1)}(\gamma)}{(m-1)!} \\ & f(\gamma) & \ddots & \vdots \\ & & \ddots & f'(\gamma) \\ 0 & & & f(\gamma) \end{pmatrix},$$

we deduce taking $f(\mathcal{J}) = R(-hI_m + \gamma z\mathcal{J})$ with $z = h\lambda$ and $\hat{z} = -h + \gamma z$,

$$P^{-1}R_s(-hI_m + \lambda h\mathcal{A})P = \begin{pmatrix} R_s(\hat{z}) & R_s'(\hat{z})\gamma z & \cdots & \frac{R_s^{(m-1)}(\hat{z})(\gamma z)^{m-1}}{(m-1)!} \\ & R_s(\hat{z}) & \ddots & \vdots \\ & & \ddots & R_s'(\hat{z})\gamma z \\ 0 & & & R_s(\hat{z}) \end{pmatrix}$$

and it turns out that the off-diagonal terms $\frac{R_s^{(j)}(\hat{z})(\gamma z)^j}{j!}, 1 \le j \le m-1$ are arbitrarily large for large $|z|$ for stiff problems where $|\lambda| \gg 1$. This is due to the z^j factors, growing like $\mathcal{O}(s^{2j})$, and can be shown that the derivatives $R_s^{(j)}$ of the stability function oscillate with an amplitude of size $\mathcal{O}(s^{-j})$ which is not sufficient to damp the z^j terms. We obtain

$$\sup_{s\ge 1, \hat{z}\in[-L_{s,\eta},0]} \|R_s(-hI_m + z\mathcal{A})\| = +\infty$$

and this makes the convergence factor of the naive implementation without partitioning non-uniformly bounded as the stiffness increases when refining the spatial mesh ($\Delta x \to 0$), as illustrate in Figure 2. Note that in the proof of the main Theorem 4.2, we used the same Jordan decomposition of the Runge-Kutta matrix $\mathcal{A} = P\mathcal{J}P^{-1}$, with the main difference in the design of the proposed method with partitioning and its stability analysis that the stability function R_s of the explicit stabilised Chebyshev method is always applied to a diagonalizable matrix.