Mixed-precision explicit stabilized Runge–Kutta methods for single- and multi-scale differential equations

Matteo Croci[a,*], Giacomo Rosilho de Souza[b]

[a]*Mathematical Institute, University of Oxford, Oxford, UK*
[b]*ANMC, Institute of Mathematics, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland*

Abstract

Mixed-precision algorithms combine low- and high-precision computations in order to benefit from the performance gains of reduced-precision without sacrificing accuracy. In this work, we design mixed-precision Runge–Kutta–Chebyshev (RKC) methods, where high-precision is used for accuracy, and low-precision for stability. Generally speaking, RKC methods are low-order explicit schemes with a stability domain growing quadratically with the number of function evaluations. We present three mixed-precision schemes: a first- and a second-order RKC method, and a first-order multirate RKC scheme for multiscale problems.

A naïve mixed-precision implementation of any Runge–Kutta scheme can harm the convergence order of the method and limit its accuracy. In this paper we present a class of mixed-precision RKC schemes that are unaffected by this limiting behaviour. The idea is simple: in RKC schemes most of the computational effort is spent on stability rather than accuracy purposes. Therefore, after reformulating the scheme, we perform only the few function evaluations needed for accuracy in high-precision, while the rest are performed in low-precision. We show that the methods retain the convergence order of the corresponding high-precision scheme and demonstrate numerically the efficiency and stability of our methods. Numerical experiments show that these methods are essentially as cheap as their fully low-precision equivalent, and as accurate as their high-precision counterpart.

Keywords: Explicit stabilized Runge–Kutta methods, mixed-precision computing, rounding errors, reduced precision, floating-point arithmetic, multirate methods
2010 MSC: 65L04, 65L06, 65L20, 65M12, 65M20, 65G50, 65G30, 65M15, 65Y99

1. Introduction

Recent years saw the return of hardware-supported low-precision arithmetic, with a drastic increase in the number of chips (GPUs, CPUs, and chips designed for machine learning) supporting the fp16 and bfloat16 half-precision floating-point formats. As a consequence, the design and analysis of algorithms that perform all or part of the computations in reduced precision has now become an active field of investigation. A popular technique is to carefully combine high and low precision computations so as to perform most of the heavy lifting in low-precision while leaving the precision-sensitive calculations in high precision. The result is a mixed-precision algorithm[1]. Mixed-precision algorithms aim to achieve the best of two worlds: perform computations that are as stable and as accurate as their fully high-precision equivalent, but with the performance benefits (in terms of speed, memory, and energy consumption) of low-precision computations, and have become very popular in the numerical linear algebra [1], machine learning [22, 49, 52], climate model simulation [9, 39, 40, 53], and in the numerical integration literature [17, 26].

In this paper we design mixed-precision Runge–Kutta (RK) methods for stiff differential equations

$$\boldsymbol{y}' = \boldsymbol{f}(\boldsymbol{y}), \qquad \boldsymbol{y}(0) = \boldsymbol{y}^0, \tag{1}$$

*Corresponding author
Email addresses: matteo.croci@maths.ox.ac.uk (Matteo Croci), giacomo.rosilhodesouza@epfl.ch (Giacomo Rosilho de Souza)
[1]Or multi-precision algorithm, if more than two floating point formats are used.

where $\boldsymbol{y}(t) \in \mathbb{R}^n$ and $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ is a twice differentiable function. We also consider multirate problems

$$\boldsymbol{y}' = \boldsymbol{f}_F(\boldsymbol{y}) + \boldsymbol{f}_S(\boldsymbol{y}), \qquad \boldsymbol{y}(0) = \boldsymbol{y}^0, \tag{2}$$

where $\boldsymbol{f}_F$ is a cheap but stiff term associated to fast (F) time-scales and $\boldsymbol{f}_S$ is an expensive but mildly stiff term associated to slower (S) time-scales. We do not assume any scale separation, hence, in addition to all fast terms, $\boldsymbol{f}_F$ may contain part of the slow dynamics too. For instance, $\boldsymbol{f}_F$ can be associated to a discrete Laplacian.

Standard explicit Runge–Kutta schemes are exceedingly inefficient for the solution of stiff problems. Therefore, we must resort to implicit or explicit stabilized methods. Implicit methods are unconditionally stable at the price of solving a possibly nonlinear system at every time step. Those systems are solved by Newton methods in conjunction with linear algebra routines; therefore, their performance strongly depends on nonlinearities, system size and efficiency of direct solvers or precoditioners when the problem size demands iterative solvers. Furthermore, convergence of Newton methods is not guaranteed for large step sizes. Explicit stabilized Runge–Kutta methods (ESRK) are a compromise between standard explicit and implicit methods. They are fully explicit, hence do not require the solution of linear systems, and their stability domain along the negative real axis grows as s^2 for an s-stage method. Thanks to this quadratic relation between work load and stability they do not have any step size restriction, require few function evaluations and compete with implicit methods, especially for large nonlinear problems [2, 4, 24, 48, 61]. In the literature we find a few families of ESRK methods, such as the DUMKA methods based on compositions of Euler steps [42, 43, 48], the Runge–Kutta–Chebyshev (RKC) methods based on recursive formulas for Chebyshev polynomials [54, 57, 60], the orthogonal Runge–Kutta–Chebyshev (ROCK) methods based on optimal orthogonal polynomials [2, 6] and the Runge–Kutta–Legendre (RKL) methods based on Legendre polynomials [51]. More recently, multirate RKC (mRKC) methods [4] for (2) and stochastic versions of RKC, ROCK and mRKC [3, 5, 7, 8] have been introduced.

Currently, many new mixed-precision algorithms are being developed by the numerical linear algebra community, among which algorithms for matrix factorization [11, 15, 46, 63, 64], iterative refinement [12, 18, 19], and Krylov subspace methods [10, 27]. For an overview of recent developments in mixed-precision computing we refer to this excellent community review [1]. The development of preconditioned iterative linear solvers is an active field of investigation due to the complications arising with loss of orthogonality of the Arnoldi/Lanczos vectors [14, 50]. However, some new fascinating results have been obtained for mixed-precision GMRES [27], and flexible GMRES [10]. Mixed-precision multigrid solvers based on iterative refinement have also been developed [47, 55]. We were unable to find any work in the numerical optimization literature specific to mixed-precision nonlinear solvers. However, the work by Tisseur [56] and various results on inexact Newton-methods [23] might be applicable here. Overall, there is still much to discover about the behaviour of all the ingredients required by implicit timestepping schemes in finite precision (iterative linear and nonlinear solvers, their preconditioning, and the interplay between these). These considerations inspired our research into explicit stabilized methods.

In this paper we design and analyze mixed-precision explicit stabilized schemes for (1) and (2) based on the RKC and the mRKC schemes, respectively. The schemes preserve the original order of convergence of the high precision methods, but the number of high precision evaluations of the right-hand side is reduced to the bare minimum. For instance, in an s-stage first-order RKC method (RKC1) only one function evaluation is needed for accuracy, and the remaining $s - 1$ evaluations are only used to increase stability. With an appropriate reformulation of the scheme, we are able to perform only one function evaluation in high precision and the remaining $s - 1$ in a low precision format without impacting accuracy. Our methodology consists in linearizing the numerical scheme and carefully evaluating the Jacobian of the right-hand side in reduced-precision arithmetic. The mixed-precision first- and second-order RKC schemes for (1) that we propose in Section 3 require only one or two, respectively, high precision evaluations of the right-hand side. The first-order mixed-precision mRKC scheme for multirate problems (2) introduced in Section 4 requires only one high precision evaluation of $\boldsymbol{f}_F$ and $\boldsymbol{f}_S$. All the function evaluations needed for stability are exclusively performed in a cheaper low-precision format. In addition to proving that the mixed-precision schemes preserve the right order of convergence, we study the propagation of rounding errors and briefly discuss how the low precision computations can impact stability. Unfortunately, rounding errors destroy any spectral relation between the integration variables and therefore we were not able to provide a rigorous stability analysis in the traditional ODE sense. Nevertheless, we provide an extensive numerical study of the stability and convergence properties of the mixed-precision schemes.

That we are aware of, the only other work on mixed-precision RK methods in the literature is by Grant [26], and by Burnett et al. [17]. However, their focus is on implicit RK methods and consequently their approach is quite different from ours. There, the authors consider mixed-precision implicit RK methods where the implicit systems are solved in reduced precision. This operation impacts the order of convergence of the scheme, which is then recovered by performing additional explicit stages in high precision. In our work we instead preserve the order of convergence by performing a single stage in high precision and the remaining in low precision without altering the overall number of stages needed. The authors of [26] and [17] cast their strategy in the framework of additive RK methods and the order conditions are derived using B-series. However, stability is not addressed in general.

This paper is structured as follows. In Section 2 we recall the most common floating-point formats, we introduce the rounding error model used in the paper, and we recall the first- and second-order RKC methods. In Section 3 we motivate mixed-precision Runge–Kutta methods, and we present the mixed-precision RKC schemes, together with a few strategies for cheap Jacobian evaluations in reduced precision that avoid the insurgence of stagnation. Later in the same section we also analyze the accuracy and stability of the mixed-precision RKC schemes. In Section 4 we introduce and analyze the mixed-precision multirate RKC scheme. In Section 5 we confirm numerically the accuracy and stability properties of the schemes. Finally, in Section 6 we present our conslusions and final remarks.

2. Preliminaries

2.1. Floating-point formats used and rounding error model

A mixed-precision algorithm uses a combination of high and low precision computations so as to maximize stability and efficiency. To set the scene, in this paper we only consider the (common) floating point number formats presented in Table 1. We typically refer to double precision as "high precision" and to any of the other formats in Table 1 as "low precision", albeit our theory and algorithms are not restricted to these choices and are still perfectly valid under other combinations and formats.

Format	u	$x_{\min}$	$x_{\max}$	t	exponent bits
bfloat16	3.91×10^{-3}	1.18×10^{-38}	3.39×10^{38}	8	8
fp16	4.88×10^{-4}	6.10×10^{-5}	6.55×10^{4}	11	5
fp32 (single)	5.96×10^{-8}	1.18×10^{-38}	3.40×10^{38}	24	8
fp64 (double)	1.11×10^{-16}	2.22×10^{-308}	1.80×10^{308}	53	11

Table 1: *Overview of the floating point systems mentioned in the paper. Here $u = 2^{-t}$ is the roundoff unit, $x_{\min}$ in the smallest normalized positive number, $x_{\max}$ is the largest finite number and t is the precision. While bfloat16 and fp32 have the same range and exponent bits, fp16 has a smaller roundoff unit (higher precision) at the cost of a smaller range.*

Assumption 2.1. *The theory presented in this paper ignores effects related to floating-point range (e.g. underflow/overflow) for simplicity. However, we remark that most range issues in our mixed-precision algorithms can easily be avoided by simple rescaling and a careful implementation, cf. Remark 5.1.*

In our work, we adopt the following standard floating point error model for round-to-nearest (cf. Chapter 2 of [35]):

$$\widehat{(a \text{ op } b)} = (a \text{ op } b)(1+\delta), \quad |\delta| < u, \quad \text{op} \in \{+, -, \times, \backslash\}, \tag{3}$$

where u is the roundoff unit (cf. Table 1) and δ is called a roundoff error. Here and in the rest of the paper we use hats to denote quantities that are the result of finite precision computations. By using this model it is possible to derive *a priori* rounding error bounds for a plethora of different algorithms and operations [35]. The main result we use in this paper is the backward error bound for matrix-vector products (cf. Section 3.5 in [35]): for a matrix $A \in \mathbb{R}^{\bar{n}\times n}$ with at most $\bar{m}$ nonzero entries per row, and a vector $\boldsymbol{x} \in \mathbb{R}^n$ we have that computing $\boldsymbol{y} = A\boldsymbol{x}$ in finite precision yields instead the vector $\hat{\boldsymbol{y}} \in \mathbb{R}^{\bar{n}}$ satisfying[2]

$$\hat{\boldsymbol{y}} = \widehat{A\boldsymbol{x}} = (A + \Delta A)\boldsymbol{x}, \quad \text{with} \quad |\Delta A| \le \gamma_{\bar{m}}|A|, \quad (\Delta A \in \mathbb{R}^{\bar{n}\times n}), \quad \gamma_{\bar{m}} = \frac{\bar{m}u}{1-\bar{m}u}.$$

[2]This is the same result as in Section 3.5 of [35], but it accounts for the fact that multiplications by zero are performed exactly.

Here by $|\cdot|$ we denote the entrywise absolute value and for round-to-nearest $\gamma_{\bar{m}}$ can be replaced with $\bar{\gamma}\bar{m} = \bar{m}u(1 + 4u + 2u^2) \approx \bar{m}u$. This last result is a consequence of the backward error bound for inner products in Corollary 3.2 in [41], after accounting for possibly non-representable entries. Equation (3) straight-forwardly implies the normwise bounds $\|\Delta A\boldsymbol{x}\|_p \leq \gamma_{\bar{m}}\|A\|_p\|\boldsymbol{x}\|_p$ for $p = 1, \infty$. We will also need a bound for the spectral norm, which we provide in the following lemma.

Lemma 2.1 (Lemma 6.6. in [35])**.** *Let $A, B \in \mathbb{R}^{\bar{n}\times n}$ have at most $\bar{m}$ nonzero entries per row and column, and satisfy $|B| \leq c|A|$ for some constant $c > 0$. Then $\|B\|_2 \leq c\min(\bar{m}, r^{1/2})\|A\|_2$, where $r = rank(A)$. Setting $c = \bar{\gamma}\bar{m}$, for u sufficiently small we then have $\|\Delta A\|_2 \leq \bar{c}\bar{m}^2 u\|A\|_2$ for $\bar{c} = 1 + 4u + 2u^2 \approx 1$.*

Proof. Let $\|A\|_{\max} = \max_{ij}|A_{ij}|$. This result is essentially Lemma 6.6 in [35] after accounting for the sparsity in A. Owing to Lemma 6.6 in [35] we have that $\|B\|_2 \leq \| |B| \|_2 \leq c\| |A| \|_2$. Since the $\|\cdot\|_2$ norm of a symmetric matrix is its spectral radius which in turn is a lower bound for any vector-induced norm, we have $\| |A| \|_2 = \| |A|^T|A| \|_2^{1/2} \leq \| |A|^T|A| \|_\infty^{1/2} \leq (\|A\|_1\|A\|_\infty)^{1/2} \leq \bar{m}\|A\|_{\max} \leq \bar{m}\|A\|_2$. For the second inequality, we use the more traditional bound $\| |A| \|_2 \leq \|A\|_F \leq r^{1/2}\|A\|_2$. □

2.2. The Runge–Kutta–Chebyshev methods

In the design of classical explicit Runge–Kutta schemes the main goal is to reach the highest possible order p for the given number of stages s. For instance, as long as $s \leq 4$, we can achieve $p = s$. However, this design strategy leaves no room for enhancing stability. In contrast, explicit stabilized Runge–Kutta methods fix the order p and use an increased number of stages $s \geq p$ to improve the stability properties of the scheme, thereby relaxing the stringent stability conditions affecting classical explicit methods.

In this paper we concentrate on first- and second-order Runge–Kutta–Chebyshev (RKC) methods [54, 57, 60], which we denote with RKC1 and RKC2 respectively. We consider problems of the form (1), and we present these schemes in delta form since it leads to smaller rounding errors [30, IV.8]. Let $\boldsymbol{y}^n$ be an approximation of $\boldsymbol{y}(t^n)$, where $t^n = n\Delta t$ and Δt is the step size. One step, of size Δt, of an s-stage RKC scheme is given by the recurrence formula

$$\begin{cases} \boldsymbol{d}_0 = \boldsymbol{0}, \quad \boldsymbol{d}_1 = \mu_1\Delta t\boldsymbol{f}(\boldsymbol{y}^n), \\ \boldsymbol{d}_j = \nu_j\boldsymbol{d}_{j-1} + \kappa_j\boldsymbol{d}_{j-2} + \mu_j\Delta t\boldsymbol{f}(\boldsymbol{y}^n + \boldsymbol{d}_{j-1}) + \gamma_j\Delta t\boldsymbol{f}(\boldsymbol{y}^n), \quad j = 2, \dots, s, \\ \boldsymbol{y}^{n+1} = \boldsymbol{y}^n + \boldsymbol{d}_s. \end{cases} \tag{4}$$

The coefficients $\mu_j, \nu_j, \kappa_j, \gamma_j$ are given by, for $j = 2, \dots, s$,

$$\mu_1 = b_1\omega_1, \qquad \mu_j = 2\omega_1 b_j/b_{j-1}, \qquad \nu_j = 2\omega_0 b_j/b_{j-1}, \qquad \kappa_j = -b_j/b_{j-2}, \qquad \gamma_j = -\mu_j a_{j-1}, \tag{5}$$

and $a_j = 1 - b_j T_j(\omega_0)$ for $j = 0, \dots, s$, where $T_j(x)$ is the Chebyshev polynomial of the first kind of degree j, defined recursively by

$$T_0(x) = 1, \qquad T_1(x) = x, \qquad T_j(x) = 2xT_{j-1}(x) - T_{j-2}(x), \quad j \geq 2.$$

The core coefficients ω_0, ω_1 and b_j for $j = 0, \dots, s$ depend on the order $p = 1, 2$, the number of stages s, and the so-called damping parameter $\varepsilon \geq 0$. For the first-order RKC1 method it holds

$$\omega_0 = 1 + \varepsilon/s^2, \quad \omega_1 = T_s(\omega_0)/T_s'(\omega_0), \quad b_j = 1/T_j(\omega_0), \quad j = 0, \dots, s \tag{6}$$

and for the second-order RKC2 method

$$\omega_0 = 1 + \varepsilon/s^2, \quad \omega_1 = T_s'(\omega_0)/T_s''(\omega_0), \quad b_0 = b_1 = b_2, \quad b_j = T_j''(\omega_0)/T_j'(\omega_0)^2, \quad j = 2, \dots, s. \tag{7}$$

Typical damping parameters are $\varepsilon = 0.05$ for RKC1, and $\varepsilon = 2/13$ for RKC2 [60]. The purpose of the damping parameter is to increase the width of the stability region, making RKC methods more resilient to small perturbations. Note that for the RKC1 scheme $a_j = 0$ and therefore $\gamma_j = 0$. We remark that the RKC1 scheme with $s = 1$ is the explicit Euler method. Let

$$c_0 = 0, \qquad c_1 = \mu_1, \qquad c_j = \nu_j c_{j-1} + \kappa_j c_{j-2} + \mu_j + \gamma_j, \quad j = 2, \dots, s. \tag{8}$$

Verwer et al. in [60] show that for RKC1 and RKC2 respectively we have $\boldsymbol{d}_j = c_j \Delta t \boldsymbol{f}(\boldsymbol{y}^n) + O(\Delta t^2)$ and $\boldsymbol{d}_j = c_j \Delta t \boldsymbol{f}(\boldsymbol{y}^n) + c_j^2 \Delta t^2/2 \boldsymbol{f}'(\boldsymbol{y}^n)\boldsymbol{f}(\boldsymbol{y}^n) + O(\Delta t^3)$. Therefore $\boldsymbol{y}^n + \boldsymbol{d}_j$ is respectively a first- or a second-order approximation of the exact solution at time $t^n + c_j \Delta t$.

When applied to the Dahlquist test equation $y' = \lambda y$ with $\lambda \in \mathbb{C}_-$, the RKC method (4) with coefficients (5) yields

$$y^{n+1} = R_s(z) y^n, \quad \text{with} \quad R_s(z) = a_s + b_s T_s(\omega_0 + \omega_1 z), \quad z = \lambda \Delta t.$$

The polynomial $R_s(z)$ is called the stability polynomial of the method. Using properties of Chebyshev polynomials, as the fact that $|T_s(x)|$ is an even function, increasing for $x \geq 1$, that $T_s(1) = 1$ and $T_s(x) \in [-1, 1]$ for $x \in [-1, 1]$ it is possible to show that $|R_s(z)| \leq 1$ for all z such that $-\omega_0 \leq \omega_0 + \omega_1 z \leq \omega_0$, i.e. for all $z \in [-\ell_s^\varepsilon, 0]$, where $\ell_s^\varepsilon = 2\omega_0/\omega_1$ [60]. We call ℓ_s^ε the real stability boundary of the method. Let

$$\beta^1(s, \varepsilon) = \left(2 - \frac{4}{3}\varepsilon\right) s^2, \qquad \beta^2(s, \varepsilon) = \frac{2}{3}\left(1 - \frac{2}{15}\varepsilon\right)(s^2 - 1). \tag{9}$$

In [60] it is shown that for RKC1 $\ell_s^\varepsilon \geq \beta^1(s, \varepsilon)$ and for RKC2 $\ell_s^\varepsilon \geq \beta^2(s, \varepsilon)$, therefore the stability domain of both methods grows *quadratically*, with respect to the number of function evaluations s, along the negative real axis. Moreover, for $z \in \mathbb{R}_-$, $|z| \leq \beta^p(s, \varepsilon)$, $p = 1, 2$, is a sufficient condition for stability. Note that the real stability boundary ℓ_s^ε of RKC2 grows slower than for RKC1 (the constant in (9) is smaller).

For more general right-hand sides, as in (1), the number of stages s is chosen at each time step so that $\Delta t \rho \leq \beta^p(s, \varepsilon)$, where ρ is the spectral radius of the Jacobian of $\boldsymbol{f}$ evaluated at $\boldsymbol{y}^n$. Note that ρ can cheaply be approximated using nonlinear power methods [44, 58]. We claim that RKC methods do not have any step size restriction since it is sufficient to take s large enough to guarantee stability. We also remark that for RKC methods the number of function evaluations is proportional to $\sqrt{\Delta t \rho}$, and not to $\Delta t \rho$ as for classical explicit RK methods such as, e.g. RK4 or DOPRI45.

Remark 2.1. In this paper we only consider autonomous problems for simplicity and easiness of notation. The mixed-precision schemes we are about to introduce can straight-forwardly be extended to nonautonomous problems after applying this simple modification: for nonautonomous problems $\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y})$, we simply replace $\boldsymbol{f}(\boldsymbol{y}^n + \boldsymbol{d}_j)$, $j = 0, \ldots, s-1$, in (4) with $\boldsymbol{f}(t^n + c_j \Delta t,\ \boldsymbol{y}^n + \boldsymbol{d}_j)$, and c_j as in (8).

3. Order-preserving mixed-precision RKC methods

We start by defining what we mean when we say that a mixed-precision timestepping scheme is order-preserving. First we need our main working assumption:

Assumption 3.1. *Throughout the paper, we assume that computations performed in high precision are exact.*

Here by "high precision" we indicate the highest precision used in the mixed-precision scheme (typically double or single precision). In a mixed-precision RK scheme, computations performed in low precision produce large roundoff errors and a naïve implementation may lead to an order-reduction phenomenon or even stagnation. When this happens, the mixed-precision scheme has a limiting convergence behaviour of order q, where q is smaller than the convergence order p of the original scheme. This motivates the following definition:

Definition 3.1 (Order-preserving mixed-precision scheme)**.** Consider a p-th order timestepping scheme. A mixed-precision implementation of the same scheme is order-preserving up to order $q \in \mathbb{N}$, $1 \leq q \leq p$ (or q-order-preserving), if it converges with order q under Assumption 3.1. If a mixed-precision implementation does not converge as $\Delta t \to 0$ (i.e. the error stagnates or blows up), then it is not order-preserving.

Throughout this section it will be clearer why a mixed-precision scheme implemented naïvely might not be order-preserving and is thus unable to reduce the error below the machine precision of the low precision format. To the extent of our knowledge, the methods we present in this paper are the first explicit mixed-precision order-preserving methods to be presented in the literature. We remark that implicit order-preserving methods are instead available [17, 26], although the approach used for these is considerably different.

3.1. Mixed-precision explicit Runge–Kutta schemes for linear problems

We begin by considering linear problems. To fix the ideas, we start with a generic s-stage order p explicit RK method and we take $\boldsymbol{f}(\boldsymbol{y}) = A\boldsymbol{y}$. We then know that the exact solution to (1) and one step of the numerical scheme in exact arithmetic are respectively given by

$$\boldsymbol{y}(t^{n+1}) = \exp(\Delta t A)\boldsymbol{y}(t^n) = \sum_{j=0}^{\infty} \frac{(\Delta t A)^j}{j!}\boldsymbol{y}(t^n), \qquad \boldsymbol{y}^{n+1} = R(\Delta t A)\boldsymbol{y}^n, \tag{10}$$

where $R(z)$ is the stability function of the method (a polynomial of degree s). The method is then of order p if $|\exp(z) - R(z)| = O(z^{p+1})$, i.e. if the coefficients of the $p+1$ lowest-degree terms of $R(z)$ match the first $p+1$ terms in the exponential series. The second equation in (10) then can be written as

$$\boldsymbol{y}^{n+1} = \sum_{j=0}^{p} \frac{(\Delta t A)^j}{j!}\boldsymbol{y}^n + \sum_{j=p+1}^{s} (1 + a_j)\frac{(\Delta t A)^j}{j!}\boldsymbol{y}^n, \tag{11}$$

where a_j for $j = p+1, \dots, s$ are coefficients which are typical of the method. After setting $\boldsymbol{y}^n = \boldsymbol{y}(t^n)$ and subtracting (11) from the first equation in (10) it is then clear that the RK method has a truncation error $O(\Delta t^{p+1})$ and a convergence rate of $O(\Delta t^p)$. However, the argument ceases to work when computations are affected by rounding errors. In this case we have something that looks like[3]

$$\hat{\boldsymbol{y}}^{n+1} = \boldsymbol{y}^n + \sum_{j=1}^{p} \frac{\Delta t^j}{j!}\left(\prod_{k=1}^{j} \hat{A}_k\right)\boldsymbol{y}^n + \sum_{j=p+1}^{s} (1 + a_j)\frac{(\Delta t)^j}{j!}\left(\prod_{k=1}^{j} \hat{A}_k\right)\boldsymbol{y}^n + \varepsilon_n. \tag{12}$$

Here $\hat{A}_k = A + \Delta A_k$ for $k = 1, \dots, s$, and the $\{\Delta A_k\}_{k=1}^s$ terms satisfy $|\Delta A_k| \le \gamma_{\tilde{m}}|A|$ for all k, and contain the rounding errors in the matrix-vector products with A. The term ε_n instead is of order $O(u)$ and contains the rounding errors from all other computations (vector multiplication by a scalar and additions). It is now immediately clear that this scheme is not of order p anymore. In fact, it is not even convergent as it blows up with a rate that is $O(u\Delta t^{-1} + \Delta t^p)$. This is a classical result (see e.g. [32, 33]), but it is often overlooked when working in double precision as u is extremely small and makes the $u\Delta t^{-1}$ term negligible. If computations are performed using lower precisions (fp16, bfloat16, and possibly fp32), however, this term becomes significant and the method stops converging [21].

A simple mixed-precision approach for RK methods is to perform all expensive matrix-vector products in low precision and all less expensive vector computations (as additions) in high precision. Under Assumption 3.1, the ε_n term in equation (12) then vanishes and rounding errors stop causing the global error to grow like $O(u\Delta t^{-1})$. Nevertheless, things are still not entirely satisfactory: using the same argument as before we obtain a convergence rate of $O(u + \Delta t^p)$ since

$$\Delta t^{-1}\|\hat{\boldsymbol{y}}^{n+1} - \boldsymbol{y}(t^{n+1})\|_2 = \|\Delta A_1 \boldsymbol{y}^n\|_2 + O(u\Delta t + \Delta t^p) = O(u + \Delta t^p).$$

It is therefore clear that standard mixed-precision RK methods are not order-preserving, i.e. they are unable to reduce the approximation error below a threshold proportional to the machine precision of the low-precision format used.

The idea of our new mixed-precision RK methods is to instead compute the first p matrix vector products exactly so that $\Delta A_k = 0$ for all $k = 1, \dots, p$, and the final convergence rate is

$$\Delta t^{-1}\|\hat{\boldsymbol{y}}^{n+1} - \boldsymbol{y}(t^{n+1})\|_2 = \left\|\frac{1 + a_{p+1}}{(p+1)!}\Delta A_{p+1}(\Delta t A)^p \boldsymbol{y}^n\right\|_2 + \left\|\frac{a_{p+1}}{(p+1)!}\Delta t^p A^{p+1}\boldsymbol{y}_0\right\|_2 + O(\Delta t^{p+1}) = O\left(u\Delta t^p + \Delta t^p\right),$$

which is the same as for the method in exact arithmetic, albeit with a slightly perturbed constant. This method is p-order-preserving. More generally, one might afford to only do q matrix-vector products in high precision, yielding a q-order-preserving method with a final convergence rate of $O(u\Delta t^q + \Delta t^p)$. In this scenario, the method will initially converge at a rate p up until $\Delta t \propto u^{1/(p-q)}$, after which the order will decay to q. We note that depending on the problem, accurate enough solutions might be obtainable before this lower-order regime kicks in and choices of $q \ll p$ for high-order methods might be feasible.

[3] The order in which computations are performed matters little for the sake of our argument here.

Remark 3.1. Given an s-stage order-p RK method and a $q \in \{1, \dots, p\}$, it is always possible to construct a q-order-preserving mixed-precision equivalent as

$$\hat{\boldsymbol{y}}^{n+1} = \left(\sum_{j=0}^{q} \frac{(\Delta t A)^j}{j!} \hat{\boldsymbol{y}}^n \right) + \left(\sum_{j=q+1}^{s} (1 + a_j) \frac{(\Delta t)^j}{j!} \left(\prod_{k=q+1}^{j} \hat{A}_k \right) A^q \hat{\boldsymbol{y}}^n \right) = \boldsymbol{u}_q + \boldsymbol{c}_s, \tag{13}$$

where $a_j = 0$ for $j = q+1, \dots, \min(p+1, s)$. Here $\boldsymbol{u}_q$ is the explicit method of order q that matches the first $q+1$ terms in the exponential series and is computed exactly, while $\boldsymbol{c}_s$ is a stabilising $O((\Delta t A)^{q+1})$ correction term that is computed in low precision.

Assuming that matrix-vector products dominate the computations, by using a q-order-preserving mixed-precision scheme we would reduce the cost by a factor

$$\varrho = 1 - \frac{(s-q) + qr}{sr} = \frac{(s-q)(r-1)}{sr}, \tag{14}$$

where r is the ratio between the costs of performing a matrix-vector product in high and in low precision. For instance, if we choose $q = 2$ in the classical RK4 method, and we use a combination of fp64 (double) and fp16, we have $r = 4$ for a sparse matrix and $r = 16$ for a dense matrix, yielding $\varrho = 37.5\%$ and $\varrho \approx 47\%$. Furthermore RK4 will converge with order 4 up until $\Delta t \propto u^{1/(p-q)} = 0.022$, after which it will converge with order 2. If we instead take $q = 1$ and consider a 64-stages RKC2 method we have $\varrho \approx 74\%$ and $\varrho \approx 92\%$ for a sparse and dense matrix respectively, and that the scheme will retain its second order until $\Delta t \propto u^{1/(p-q)} = 2^{-11}$. We remark that these are only rough estimates and that in practice r might be larger if computations are memory-bound.

While the overall idea should hopefully now be clear, there are three complications: 1) In most traditional RK methods we have that s is not much larger than p and, for instance if $s = p$, the whole scheme must be run in high precision to retain the full order. 2) The presence of nonlinearities disrupts the argument we just presented and the mixed-precision scheme must be constructed more carefully. 3) Performing some computations in low precision might disrupt the numerical stability of the method.

In this paper we focus on stabilized RK methods for which $s \gg p$, so point 1 is not really an issue (in fact point 1 is what motivated us to look at ESRK in the first place). For all other explicit methods we simply advocate that using $q < p$ might still bring some computational advantage, especially for large Δt. As far as points 2 and 3 are concerned, in the rest of this section and in Section 4 we explain how to implement the mixed-precision schemes so as to deal with nonlinearities and we derive under which conditions these schemes are still numerically stable. However, there are some limitations:

Remark 3.2. We are currently unable to develop efficient mixed-precision ESRK schemes based on three-term recurrence relations that are more than second-order preserving. Furthermore, the maximum number of stages in our second-order ESRK schemes is restricted by the low precision format used.

3.2. Mixed-precision RKC schemes for nonlinear problems

In order to construct a mixed-precision version of method (4) that is order-preserving up to order $q = p$ we must ensure that all $p+1$ lowest-order terms are computed exactly (i.e. in high precision). The resulting mixed-precision methods therefore vary according to the value of p. In this section we consider problem (1) and in what follows we write $\boldsymbol{f}(\boldsymbol{y})$ and $\hat{\boldsymbol{f}}(\boldsymbol{y})$ to indicate function evaluations in high or low precision, respectively.

First-order-preserving RKC1 scheme.

We consider here the first-order version of (4), hence with coefficients given by (5) and (6) and thus $\gamma_j = 0$. Our 1-order-preserving mixed-precision RKC1 method needs only one high precision evaluation of the right-hand side $\boldsymbol{f}$, to preserve accuracy. The remaining $s-1$ evaluations are for stability and can be performed in low precision.

Let $\hat{\boldsymbol{y}}^n$ be an approximation to $\boldsymbol{y}(t^n)$ computed with the mixed-precision scheme and $s \in \mathbb{N}$ such that $\Delta t \rho \leq \beta^1(s, \varepsilon)$, where ρ is the spectral radius of the Jacobian of $\boldsymbol{f}$ evaluated in $\hat{\boldsymbol{y}}^n$ and $\beta^1(s, \varepsilon)$ is given in (9). One step of the 1-order-

preserving mixed-precision RKC1 scheme is given by:

$$\begin{cases} \hat{\boldsymbol{d}}_0 = \boldsymbol{0}, \quad \hat{\boldsymbol{d}}_1 = \mu_1 \Delta t \boldsymbol{f}(\hat{\boldsymbol{y}}^n), \\ \hat{\boldsymbol{d}}_j = \nu_j \hat{\boldsymbol{d}}_{j-1} + \kappa_j \hat{\boldsymbol{d}}_{j-2} + \mu_j \Delta t (\boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \hat{\Delta}\boldsymbol{f}_{j-1}), \quad j = 2, \dots, s, \\ \hat{\boldsymbol{y}}^{n+1} = \hat{\boldsymbol{y}}^n + \hat{\boldsymbol{d}}_s, \end{cases} \tag{15}$$

where the $\{\hat{\Delta}\boldsymbol{f}_j\}_{j=1}^{s-1}$ are quantities evaluated in low precision satisfying

$$\hat{\Delta}\boldsymbol{f}_j = \Delta \boldsymbol{f}_j + O(\epsilon \Delta t + \Delta t^2), \quad \forall j, \tag{16}$$

where ϵ is a small positive constant and $\Delta \boldsymbol{f}_j = \boldsymbol{f}(\hat{\boldsymbol{y}}^n + \hat{\boldsymbol{d}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}^n)$. Note that if in (15) we replace $\hat{\Delta}\boldsymbol{f}_j$ with $\Delta \boldsymbol{f}_j$ we obtain the original RKC1 scheme (4). However, evaluating this difference in high precision is expensive, and for this reason we instead compute an $O(\epsilon \Delta t + \Delta t^2)$ approximation in low precision. The $O(\Delta t)$ accuracy of the approximation is the central ingredient that is required to obtain a 1-order-preserving method, while the small constant ϵ ensures that $\hat{\Delta}\boldsymbol{f}_j$ is close to $\Delta \boldsymbol{f}_j = O(\Delta t)$, which in turn is an $O(\Delta t^2)$ approximation of $\boldsymbol{f}'(\hat{\boldsymbol{y}}^n)\hat{\boldsymbol{d}}_j$ and is crucial to ensure stability. In fact, in this case $\hat{\Delta}\boldsymbol{f}_j$ mimics the Jacobian of $\boldsymbol{f}$ and damps the high frequencies in the $\boldsymbol{f}(\hat{\boldsymbol{y}}^n)$ term. The challenge here is that a naïve low-precision evaluation of $\boldsymbol{f}$ leads to rounding errors that in general are not $O(\Delta t)$, but only $O(u)$, thus impacting the limiting accuracy of the scheme, and for this reason the $\hat{\Delta}\boldsymbol{f}_j$ terms must be carefully implemented. There are multiple ways of computing $\hat{\Delta}\boldsymbol{f}_j$ so as to satisfy (16), but we refer to Section 3.3 for a discussion on the available options.

Remark 3.3. We only presented the mixed-precision RKC1 scheme (15) as an example of how to construct a 1-order-preserving RKC method. However, the same strategy (use the delta form and store and factor out the first-order terms) can be straight-forwardly employed to construct the 1-order-preserving mixed-precision version of higher-order schemes such as RKC2 or the ROCK methods [2, 6].

Second-order-preserving RKC2 scheme.
In the first-order scheme (15) we only evaluate the nonlinear term $\boldsymbol{f}$ once in high precision since the $\hat{\Delta}\boldsymbol{f}_j$ terms are only needed for stability and therefore can be approximated by low precision evaluations. In a second-order scheme the Jacobian is also needed for accuracy and we must be more careful. So as to derive the 2-order-preserving mixed-precision RKC2 scheme we first reformulate its exact version (cf. (4)) subtracting $c_j \Delta t \boldsymbol{f}(\boldsymbol{y}^n)$ from each stage, where the c_j are defined in (8). This yields

$$\begin{cases} \boldsymbol{v}_0 = \boldsymbol{0}, \quad \boldsymbol{v}_1 = \boldsymbol{0}, \\ \boldsymbol{v}_j = \nu_j \boldsymbol{v}_{j-1} + \kappa_j \boldsymbol{v}_{j-2} + \mu_j \Delta t \left(\boldsymbol{f}(\boldsymbol{y}^n + c_{j-1} \Delta t \boldsymbol{f}(\boldsymbol{y}^n) + \boldsymbol{v}_{j-1}) - \boldsymbol{f}(\boldsymbol{y}^n) \right), \quad j = 2, \dots, s, \\ \boldsymbol{y}^{n+1} = \boldsymbol{y}^n + \Delta t \boldsymbol{f}(\boldsymbol{y}^n) + \boldsymbol{v}_s, \end{cases} \tag{17}$$

where $\boldsymbol{v}_j = \boldsymbol{d}_j - c_j \Delta t \boldsymbol{f}(\boldsymbol{y}^n)$ and the number of stages s is such that $\Delta t \rho \leq \beta^2(s, \varepsilon)$. A 2-order-preserving mixed-precision RKC2 method can be obtained from (17) by linearization and computing Jacobian multiplications with $\boldsymbol{v}_j$ in low precision. As $\boldsymbol{d}_j = c_j \Delta t \boldsymbol{f}(\boldsymbol{y}^n) + O(\Delta t^2)$ [60] then $\boldsymbol{v}_j = O(\Delta t^2)$ does not contain any first-order term and multiplications $\Delta t \boldsymbol{f}'(\boldsymbol{y}^n)\boldsymbol{v}_j$ can be performed in low precision without sacrificing accuracy.

More generally, the 2-order-preserving RKC2 scheme is given by

$$\begin{cases} \hat{\boldsymbol{v}}_0 = \boldsymbol{0}, \quad \hat{\boldsymbol{v}}_1 = \boldsymbol{0}, \\ \hat{\boldsymbol{v}}_j = \nu_j \hat{\boldsymbol{v}}_{j-1} + \kappa_j \hat{\boldsymbol{v}}_{j-2} + \mu_j \Delta t \hat{\Delta}\boldsymbol{f}_{j-1}, \quad j = 2, \dots, s, \\ \hat{\boldsymbol{y}}^{n+1} = \hat{\boldsymbol{y}}^n + \Delta t \boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \hat{\boldsymbol{v}}_s, \end{cases} \tag{18}$$

where now we require

$$\hat{\Delta}\boldsymbol{f}_j = \Delta \boldsymbol{f}_j + O(\Delta t^2), \quad \forall j, \tag{19}$$

and the $\Delta \boldsymbol{f}_j$ are now computed as $\Delta \boldsymbol{f}_j = \boldsymbol{f}(\hat{\boldsymbol{y}}^n + c_j \Delta t \boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \hat{\boldsymbol{v}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}^n)$. If we can afford evaluating $\Delta \boldsymbol{f}_j$ in high-precision then we recover the standard RKC2 method (17). However, if this is not possible, we need to provide a second-order approximation $\hat{\Delta} \boldsymbol{f}_j$ instead. In fact, evaluating the $\Delta t \Delta \boldsymbol{f}_j$ term with at most $O(\Delta t^3)$ error is crucial in order to obtain a 2-order-preserving method. Again, multiple strategies are feasible here and we will describe them in Section 3.3.

We will see in Sections 3.5 and 5.2.1 how the scheme (18) grows unstable if the number of stages is larger than a given threshold dictated by the low precision format. This effect is due to the fact that we iterate over $\hat{\boldsymbol{v}}_j$ instead of the more stable $\hat{\boldsymbol{d}}_j$. While it is possible to obtain a 1-order-preserving RKC2 method without this limitation by iterating over $\hat{\boldsymbol{d}}_j$ (cf. Remark 3.3), we are currently unable to find a 2-order-preserving scheme that is unaffected by this restriction.

3.3. Evaluation of the $\hat{\Delta} \boldsymbol{f}_j$ terms

As part of our order-preserving RKC schemes (15) and (18) we required that the $\hat{\Delta} \boldsymbol{f}_j$ terms could be evaluated in reduced precision at the given accuracy (16) for RKC1 and (19) for RKC2. We now explain how this can be done in practice under different scenarios. For this purpose, suppose that $\boldsymbol{f}$ has the form

$$\boldsymbol{f}(\boldsymbol{y}) = A\boldsymbol{y} + \boldsymbol{g}(\boldsymbol{y}),$$

with $A \in \mathbb{R}^{n \times n}$, and $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^n$ be any function with the same smoothness as $\boldsymbol{f}$. Obviously, one can put $A = 0$ and let $\boldsymbol{g}$ absorb all linear terms. However, factorizing out the linear terms helps in reducing rounding errors, and A here can also be considered as the Jacobian of some other terms within $\boldsymbol{f}$ for which the derivative is more readily available.

Scenario 1: the nonlinear term is much cheaper to evaluate than the linear term. In this case, for the first-order scheme, it is simply possible to implement $\hat{\Delta} \boldsymbol{f}_j$ using mixed-precision as

$$\hat{\Delta} \boldsymbol{f}_j = \hat{A} \hat{\boldsymbol{d}}_j + \boldsymbol{g}(\hat{\boldsymbol{y}}^n + \hat{\boldsymbol{d}}_j) - \boldsymbol{g}(\hat{\boldsymbol{y}}^n) = \Delta \boldsymbol{f}_j + O(u \Delta t), \tag{20}$$

since $\hat{\boldsymbol{d}}_j = O(\Delta t)$ and the low-precision multiplication by $\hat{A}$ yields an $O(u\Delta t)$ error (i.e. $\varepsilon = u$). This strategy requires applying A in high precision only to $\hat{\boldsymbol{y}}^n$ at the first stage of (15). If possible, these high-precision matvecs could even be performed matrix-free to avoid storing A in high precision. In the second-order case we instead compute $\hat{\Delta} \boldsymbol{f}_j$ as

$$\hat{\Delta} \boldsymbol{f}_j = \hat{A} \hat{\boldsymbol{v}}_j + c_j \Delta t A \boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \boldsymbol{g}(\hat{\boldsymbol{y}}^n + c_j \Delta t \boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \hat{\boldsymbol{v}}_j) - \boldsymbol{g}(\hat{\boldsymbol{y}}^n) = \Delta \boldsymbol{f}_j + O(u \Delta t^2), \tag{21}$$

since now $\hat{\boldsymbol{v}}_j = O(\Delta t^2)$ and multiplication by $\hat{A}$ yields an $O(u\Delta t^2)$ error. This strategy only requires evaluating $\boldsymbol{f}(\hat{\boldsymbol{y}}^n)$ and $A\boldsymbol{f}(\hat{\boldsymbol{y}}^n)$ once in high-precision every s stages.

Scenario 2: both linear and nonlinear terms are expensive to evaluate. In this case we need to implement the whole $\hat{\Delta} \boldsymbol{f}_j$ in low-precision while still ensuring the right order of accuracy and stability. The idea is to employ Jacobian approximations. For the first-order scheme we have that

$$\Delta \boldsymbol{f}_j = \boldsymbol{f}'(\hat{\boldsymbol{y}}^n) \hat{\boldsymbol{d}}_j + O(\Delta t^2),$$

and therefore we can implement $\hat{\Delta} \boldsymbol{f}_j$ by approximating the action of the Jacobian $\boldsymbol{f}'(\hat{\boldsymbol{y}}^n)$ against $\hat{\boldsymbol{d}}_j$ in low precision, since $\hat{\boldsymbol{d}}_j = O(\Delta t)$ and this leads to an $O(u\Delta t)$ rounding error (again $\varepsilon = u$). If an analytic expression for the directional derivative of $\boldsymbol{f}$ is available, then the easiest option is to just evaluate the action of the derivative in low precision. When the Jacobian is not known analytically, there are various techniques available to compute the action of a Jacobian against a vector efficiently. However, we do not describe these techniques in detail here, and we only mention two. The first is automatic differentiation [28], through which we can compute the action of $\boldsymbol{f}'$ at up to roughly the same cost of a couple of evaluations of $\boldsymbol{f}$ itself (see Chapter 4 in [28]). The second simply entails computing

$$\hat{\Delta} \boldsymbol{f}_j \coloneqq \hat{A} \hat{\boldsymbol{d}}_j + \delta^{-1} \left(\hat{\boldsymbol{g}} \left(\hat{\boldsymbol{y}}^n + \delta \, \hat{\boldsymbol{d}}_j \right) - \boldsymbol{g} \left(\hat{\boldsymbol{y}}^n \right) \right) = \Delta \boldsymbol{f}_j + O(\sqrt{u} \Delta t + \Delta t^2), \qquad \delta = \frac{\sqrt{u}}{\Delta t}, \tag{22}$$

which only requires low precision evaluations of $\boldsymbol{g}$, as $\boldsymbol{g}(\hat{\boldsymbol{y}}^n)$ is already known from $\boldsymbol{f}(\hat{\boldsymbol{y}}^n)$, which is needed during the first stage of (15). Estimate (22) yields an accurate enough approximation with $\varepsilon = \sqrt{u}$ and is proved in Lemma A.1. The introduction of the coefficient δ is crucial to guarantee a good Jacobian approximation. Indeed, the roundoff

introduced by $\hat{\boldsymbol{g}}$ is $O(u)$ and a multiplication by δ^{-1} ensures that $\hat{\Delta}\boldsymbol{f}_j = \boldsymbol{f}'(\hat{\boldsymbol{y}}^n)\hat{\boldsymbol{d}}_j + O(\sqrt{u}\Delta t)$. See Lemma A.1 for details.

For the second-order scheme, we rewrite $\Delta\boldsymbol{f}_j$ as

$$\Delta\boldsymbol{f}_j = \Big(\boldsymbol{f}(\hat{\boldsymbol{y}}^n + c_j\Delta t\boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \hat{\boldsymbol{v}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}^n + c_j\Delta t\boldsymbol{f}(\hat{\boldsymbol{y}}^n))\Big) + \Big(\boldsymbol{f}(\hat{\boldsymbol{y}}^n + c_j\Delta t\boldsymbol{f}(\hat{\boldsymbol{y}}^n)) - \boldsymbol{f}(\hat{\boldsymbol{y}}^n)\Big) = \Delta_1\boldsymbol{f}_j + \Delta_2\boldsymbol{f}_j.$$

The idea is then to approximate/evaluate the two $\Delta_1\boldsymbol{f}_j$ and $\Delta_2\boldsymbol{f}_j$ terms with $O(\Delta t^2)$ error, i.e. to compute

$$\hat{\Delta}_1\boldsymbol{f}_j = \Delta_1\boldsymbol{f}_j + O(\Delta t^2), \qquad \hat{\Delta}_2\boldsymbol{f}_j = \Delta_2\boldsymbol{f}_j + O(\Delta t^2).$$

Here $\hat{\Delta}_1\boldsymbol{f}_j$ and $\hat{\Delta}_2\boldsymbol{f}_j$ can again be obtained via Jacobian approximation/evaluation:

$$\hat{\Delta}_1\boldsymbol{f}_j = \hat{\boldsymbol{f}}'(\hat{\boldsymbol{y}}^n + c_j\Delta t\boldsymbol{f}(\hat{\boldsymbol{y}}^n))\hat{\boldsymbol{v}}_j = \Delta_1\boldsymbol{f}_j + O(\Delta t^2), \qquad \hat{\Delta}_2\boldsymbol{f}_j = c_j\Delta t\boldsymbol{f}'(\hat{\boldsymbol{y}}^n)\boldsymbol{f}(\hat{\boldsymbol{y}}^n) = \Delta_2\boldsymbol{f}_j + O(\Delta t^2),$$

for which we can use the same techniques mentioned in the first-order case. Note that the hat in the Jacobian in the expression for $\hat{\Delta}_1\boldsymbol{f}_j$ indicates that the Jacobian approximation can be performed in low-precision, while the expression for $\hat{\Delta}_2\boldsymbol{f}_j$ requires the Jacobian to be evaluated in high-precision. This different choice is crucial to ensure that the overall error in the approximation is $O(\Delta t^2)$. We remark that for the $\hat{\Delta}_2\boldsymbol{f}_j$ term a single high-precision evaluation of $\boldsymbol{f}'(\hat{\boldsymbol{y}}^n)\boldsymbol{f}(\hat{\boldsymbol{y}}^n)$ every s stages is sufficient since the only thing that varies with j is c_j.

Other scenarios.

- If the linear term is cheaper to evaluate than the nonlinear term the solution is to simply apply the strategy for Scenario 2 and evaluate the matrix-vector products in high precision.
- In some cases it is possible to implement differences like $\boldsymbol{g}(\boldsymbol{y}+\boldsymbol{b}) - \boldsymbol{g}(\boldsymbol{y})$ in such a way that the rounding errors are automatically of the right order of accuracy. An example scenario is when there is an analytical expression for the difference of the right order, see Example 3.1.

Example 3.1. Take the nonlinear convective term of the Navier-Stokes equations, $\boldsymbol{g}(\boldsymbol{y}) = \boldsymbol{y}\nabla\boldsymbol{y}$. We then have that

$$\boldsymbol{g}(\boldsymbol{y}+\boldsymbol{b}) - \boldsymbol{g}(\boldsymbol{y}) = \boldsymbol{b}\nabla\boldsymbol{y} + \boldsymbol{y}\nabla\boldsymbol{b} + \boldsymbol{b}\nabla\boldsymbol{b}. \tag{23}$$

For the first-order methods we then take $\boldsymbol{y} = \hat{\boldsymbol{y}}_n$, $\boldsymbol{b} = \hat{\boldsymbol{d}}_j = O(\Delta t)$, and (23) evaluated in low precision yields an $O(u\Delta t)$ error. For the second-order methods we instead take $\boldsymbol{y} = \hat{\boldsymbol{y}}_n + c_j\Delta t\boldsymbol{f}(\hat{\boldsymbol{y}}^n)$ and $\boldsymbol{b} = \hat{\boldsymbol{v}}_j = O(\Delta t^2)$ for the $\hat{\Delta}_1\boldsymbol{f}_j$ term, yielding an $O(u\Delta t^2)$ rounding error if (23) is evaluated in low-precision. For the $\hat{\Delta}_2\boldsymbol{f}_j$ term we instead set $\boldsymbol{y} = \hat{\boldsymbol{y}}^n$ and $\boldsymbol{b} = c_j\Delta t\boldsymbol{f}(\hat{\boldsymbol{y}}^n)$ yielding $c_j\Delta t\left(\boldsymbol{f}(\hat{\boldsymbol{y}}^n)\nabla\hat{\boldsymbol{y}}^n + \hat{\boldsymbol{y}}^n\nabla\boldsymbol{f}(\hat{\boldsymbol{y}}^n)\right) + c_j^2\Delta t^2\boldsymbol{f}(\hat{\boldsymbol{y}}^n)\nabla\boldsymbol{f}(\hat{\boldsymbol{y}}^n)$, where each term is constant across the stages (except for the scalings by c_j and c_j^2) and they can be pre-computed once every RKC time step in high-precision (i.e. no error). In practice, the $\boldsymbol{b}\nabla\boldsymbol{b}$ term in (23) is of higher order and can possibly be dropped to save on computations. Note that by dropping higher-order terms we recover the directional derivative of $\boldsymbol{g}$.

3.4. Cost analysis

Before analysing the convergence and stability properties of algorithms (15) and (18) we first derive an expression for the computational savings of using our mixed-precision methods with respect to a method fully implemented in high precision. We assume that vector operations are negligible[4], and we define r to be the ratio between the cost of evaluating $\boldsymbol{f}$ in high precision and the cost of computing $\hat{\Delta}\boldsymbol{f}_j$ with one of the strategies we just presented (the latter not including the cost of the quantities computed once every s-stages). We then have a cost reduction factor ϱ of

$$\varrho = 1 - \frac{(s-q)+qr}{sr} = \frac{(s-q)(r-1)}{sr}, \quad q \in \{1,2\}.$$

$$\varrho \longrightarrow 1 - \frac{1}{r}, \quad \text{as} \quad s \to \infty, \qquad \text{and} \qquad \varrho \longrightarrow 1 - \frac{q}{s}, \quad \text{as} \quad r \to \infty.$$

[4]Note that methods (15) and (18) can be implemented with the same number of vector operations as their high precision equivalents.

Note that this expression for ϱ is essentially the same as in (14). By looking at the limit cases for $s, r \to \infty$, we see that the best cost reduction factor we can hope for when s is large (typical in ESRK methods) is $1 - r^{-1}$, which for r also large becomes very close to 1. The actual value of r grows as the number of bits of the low-precision format chosen decreases and is problem-dependent.

We can present a couple of examples for Scenario 2 under some simplifying assumptions: 1) We only look at flop counts and we ignore savings related to memory efficiency. 2) One flop in a format using twice or four times the number of bits costs twice or four times as much. 3) The cost of evaluating $\hat{\Delta}\boldsymbol{f}_j$ is roughly the same as that of evaluating $\hat{\boldsymbol{f}}$ in the same precision[5] For instance, for sparse A and linear-cost evaluations of the nonlinear term (e.g. this is the case for the heat equation with a nonlinear reaction term acting entrywise on the solution) we obtain $r = 2$ for double-single or single-half combinations, and $r = 4$ for double-half, reducing the overall cost by half or a factor of 4 respectively. For dense A and/or quadratic-cost evaluations of $\boldsymbol{g}$ we instead get up to $r = 4$ (double-single or single-half) and $r = 16$ (double-half) leading to much greater savings.

In memory-bound computations r might actually be larger since most function evaluations are performed in a low-precision format, which might allow for better cache exploitation. We remark that our q-order-preserving schemes require the storage of q additional vectors. However, they might also allow to avoid storing some of the data needed to evaluate $\boldsymbol{f}$ in high-precision since high-precision evaluations of $\boldsymbol{f}$ occur less often. For instance, the matrix A could be implemented matrix-free in high-precision and only explicitly stored in low-precision.

3.5. Convergence and stability analysis

We present here the accuracy and stability analysis for the mixed-precision schemes (15) and (18) introduced in Section 3.2. We start by introducing Lemma 3.1 below which collects some known results that are crucial for the modelling of stage perturbations as truncation or rounding errors. There, $U_j(x)$ is the Chebyshev polynomial of the second kind of degree j, defined recursively by

$$U_0(x) = 1, \qquad U_1(x) = 2x, \qquad U_j(x) = 2xU_{j-1}(x) - U_{j-2}(x), \quad j \geq 2.$$

Lemma 3.1. *Let $p = 1, 2$, $M \in \mathbb{R}^{n\times n}$ be a symmetric nonpositive definite matrix and s such that $\rho(M) \leq \beta^p(s, \varepsilon)$, where $\rho(M)$ is the spectral radius of M and $\beta^p(s, \varepsilon)$ is as in (9). Let μ_j, ν_j, κ_j be as in (5), with ω_0, ω_1, b_j as in (6) if $p = 1$ and as in (7) if $p = 2$. Let $\boldsymbol{r}_j \in \mathbb{R}^n$, $j = 1, \ldots, s$, and*

$$\boldsymbol{k}_0 = \boldsymbol{0}, \qquad \boldsymbol{k}_1 = \mu_1\boldsymbol{r}_1, \qquad \boldsymbol{k}_j = \nu_j\boldsymbol{k}_{j-1} + \kappa_j\boldsymbol{k}_{j-2} + \mu_j M\boldsymbol{k}_{j-1} + \mu_j\boldsymbol{r}_j, \quad j = 2, \ldots, s.$$

Then:

i) Let $I \in \mathbb{R}^{n\times n}$ the identity matrix. It holds

$$\boldsymbol{k}_k = \sum_{j=1}^{k} \frac{b_k}{b_j} U_{k-j}(\omega_0 I + \omega_1 M)\mu_j\boldsymbol{r}_j, \qquad k = 1, \ldots, s.$$

ii) We have the identities

$$\overline{R}_k(z) := \frac{R_k(z) - 1}{z} = \sum_{j=1}^{k} \frac{b_k}{b_j} U_{k-j}(\omega_0 + \omega_1 z)\mu_j, \qquad R_k(z) = 1 + z + z^2 \sum_{j=2}^{k} \frac{b_k}{b_j} U_{k-j}(\omega_0 + \omega_1 z)\mu_j c_{j-1}.$$

Here $R_k(z) = a_k + b_k T_k(\omega_0 + \omega_1 z)$ is the internal stability polynomial of the RKC scheme. Note that $R_k(z) = 1 + c_k z + O(z^2)$ [60] and thus $\overline{R}_k(z)$ is as well a polynomial.

iii) If $\boldsymbol{r} = \boldsymbol{r}_j$ for $j = 1, \ldots, s$ then $\boldsymbol{k}_k = \overline{R}_k(M)\boldsymbol{r}$.

[5] This assumption holds for evaluations in the style of (22). When Jacobian matrix-vector products are instead computed via a forward pass of automatic differentiation, computing $\hat{\Delta}\boldsymbol{f}_j$ costs up to 2.5 times as $\hat{\boldsymbol{f}}$, even though in practice it might be cheaper, cf. Chapter 4 in [28].

iv) $\|\boldsymbol{k}_k\|_2 \leq c_k \max_{j=1,\dots,k} \|\boldsymbol{r}_j\|_2$ *with* c_k *as in* (8)*. Note that* $c_k \leq 1$ *[60].*

Proof. Item i) has been proved in [60] and the first identity in ii) in [4]. For the second identity we apply the equivalent schemes (4) and (17) to $y' = \lambda y$, the result follows by comparing the results (with $z = \Delta t\lambda$). Realtion iii) follows from ii). For iv) we use

$$\|U_{k-j}(\omega_0 I + \omega_1 M)\|_2 \leq \max_{-\beta^p(s,\varepsilon)\leq z\leq 0} |U_{k-j}(\omega_0 + \omega_1 z)| \leq |U_{k-j}(\omega_0)| = U_{k-j}(\omega_0),$$

and the fact that $b_j, \mu_j > 0$, hence

$$\|\boldsymbol{k}_k\|_2 \leq \sum_{j=1}^{k} \frac{b_k}{b_j} \|U_{k-j}(\omega_0 I + \omega_1 M)\|_2 \mu_j \|\boldsymbol{r}_j\|_2 \leq \max_{j=1,\dots,k} \|\boldsymbol{r}_j\|_2 \sum_{j=1}^{k} \frac{b_k}{b_j} U_{k-j}(\omega_0)\mu_j = \max_{j=1,\dots,k} \|\boldsymbol{r}_j\|_2 \overline{R}_k(0),$$

and we conclude using $\overline{R}_k(0) = R_k'(0) = c_k$ [60]. □

Convergence analysis.

We present here the convergence analysis in which we prove that the mixed-precision RKC1 (15) and RKC2 (18) schemes are first- and second-order accurate, respectively.

The order of convergence of first- and second-order explicit stabilized schemes is typically only proven for linear problems [6, 59, 60], as this is sufficient to infer convergence in the nonlinear case as well [29]. By exploiting the internal stability properties of these methods, this approach yields error bounds that are independent from the problem stiffness. Such results are unusual for explicit methods and are akin to those obtained with B-convergence analysis for implicit methods. In contrast, in our analysis we directly consider the nonlinear case to show that the low-precision Jacobian approximations proposed in Section 3.3 do not impact convergence. For this purpose, we first perform a standard Taylor expansion of the mixed-precision schemes in Theorem 3.2, and verify that the order of convergence is preserved. However, with this strategy we cannot see the advantages of $\hat{\Delta}\boldsymbol{f}_j$ approximating the Jacobian. Therefore, in Theorem 3.3 below we adopt a stronger assumption and unveil the good stability properties brought by the $\hat{\Delta}\boldsymbol{f}_j$ terms.

Theorem 3.2. *Assuming* (16)*, the solution* $\hat{\boldsymbol{y}}^{n+1}$ *of the mixed-precision RKC1 scheme* (15) *satisfies*

$$\hat{\boldsymbol{y}}^{n+1} = \hat{\boldsymbol{y}}^n + \Delta t \boldsymbol{f}(\hat{\boldsymbol{y}}^n) + O(\epsilon\Delta t^2 + \Delta t^2). \tag{24}$$

Assuming (19)*, the solution* $\hat{\boldsymbol{y}}^{n+1}$ *of the mixed-precision RKC2 scheme* (18) *satisfies*

$$\hat{\boldsymbol{y}}^{n+1} = \hat{\boldsymbol{y}}^n + \Delta t \boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \frac{1}{2}\Delta t^2 \boldsymbol{f}'(\hat{\boldsymbol{y}}^n)\boldsymbol{f}(\hat{\boldsymbol{y}}^n) + O(\Delta t^3). \tag{25}$$

Proof. To obtain (24) we apply Lemma 3.1 to (15), with $M = 0$ and $\boldsymbol{r}_j = \Delta t(\boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \hat{\Delta}\boldsymbol{f}_{j-1})$ for $j = 1, \dots, s$ (with $\hat{\Delta}\boldsymbol{f}_0 = \boldsymbol{0}$), we obtain

$$\hat{\boldsymbol{y}}^{n+1} = \hat{\boldsymbol{y}}^n + \hat{\boldsymbol{d}}_s = \hat{\boldsymbol{y}}^n + \sum_{j=1}^{s} \frac{b_s}{b_j} U_{s-j}(\omega_0)\mu_j \Delta t(\boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \hat{\Delta}\boldsymbol{f}_{j-1}) = \hat{\boldsymbol{y}}^n + \overline{R}_s(0)\Delta t \boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \Delta t \sum_{j=1}^{s} \frac{b_s}{b_j} U_{s-j}(\omega_0)\mu_j \hat{\Delta}\boldsymbol{f}_{j-1}.$$

The result follows by using $\overline{R}_s(0) = 1$ and applying Lemma 3.1 iv) to the last sum. We now prove (25). From (18) and (19) it can be shown by induction that $\hat{\boldsymbol{v}}_j = O(\Delta t^2)$. Therefore, by using (19) it follows that $\hat{\Delta}\boldsymbol{f}_{j-1} = \boldsymbol{f}'(\hat{\boldsymbol{y}}^n)c_{j-1}\Delta t \boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \boldsymbol{r}_j$ with $\boldsymbol{r}_j = O(\Delta t^2)$. The scheme (18) then gives

$$\hat{\boldsymbol{v}}_s = \Delta t \sum_{j=2}^{s} \frac{b_s}{b_j} U_{s-j}(\omega_0)\mu_j \hat{\Delta}\boldsymbol{f}_{j-1} = \Delta t \sum_{j=2}^{s} \frac{b_s}{b_j} U_{s-j}(\omega_0)\mu_j (\boldsymbol{f}'(\hat{\boldsymbol{y}}^n)c_{j-1}\Delta t \boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \boldsymbol{r}_j).$$

Using $\sum_{j=2}^{s} \frac{b_s}{b_j} U_{s-j}(\omega_0)\mu_j c_{j-1} = \lim_{z\to 0} (R_s(z) - 1 - z)/z^2 = 1/2$ (Lemma 3.1 ii)) we obtain

$$\hat{\boldsymbol{y}}^{n+1} = \hat{\boldsymbol{y}}^n + \Delta t \boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \hat{\boldsymbol{v}}_s = \hat{\boldsymbol{y}}^n + \Delta t \boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \frac{1}{2}\Delta t^2 \boldsymbol{f}'(\hat{\boldsymbol{y}}^n)\boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \Delta t \sum_{j=2}^{s} \frac{b_s}{b_j} U_{s-j}(\omega_0)\mu_j \boldsymbol{r}_j,$$

and again we conclude by applying Lemma 3.1 iv), and the fact that $\boldsymbol{r}_j = O(\Delta t^2)$ to bound the last term on the right-hand side. □

In the proof of Theorem 3.2 we cannot infer anything about the stability of the methods. In order to show stability, we require the following assumption:

Assumption 3.2. *The schemes (15) and (18) satisfy* $\|\hat{\boldsymbol{d}}_j\|_2 \le C_{\boldsymbol{d}}\Delta t\|\boldsymbol{f}(\hat{\boldsymbol{y}}^n)\|_2$, *and* $\|c_j\Delta t\boldsymbol{f}(\hat{\boldsymbol{y}}^n)+\hat{\boldsymbol{v}}_j\|_2 \le C_{\boldsymbol{d}}\Delta t\|\boldsymbol{f}(\hat{\boldsymbol{y}}^n)\|_2$, *for some* $C_{\boldsymbol{d}} > 0$, *and similar assumptions also hold for the schemes (4) and (17). Furthermore,* $\boldsymbol{f}$ *is twice differentiable with* $\|\boldsymbol{f}''(\boldsymbol{y})\|_2 \le C_{f''}$, *where* $C_{f''} > 0$ *is a small constant.*

The first requirement is true for small Δt and is an internal stability assumption otherwise. We will discuss internal stability (i.e. error propagation within one time step) in Section 3.6. For our analysis it is sufficient for the bound $\|\boldsymbol{f}''(\boldsymbol{y})\|_2 \le C_{f''}$ to be satisfied in a neighborhood of the solution. Note that assuming $C_{f''}$ to be small is not restrictive since the norm of $\boldsymbol{f}''$ is not related to stiffness. To see this consider, for instance, the linear case $\boldsymbol{f}(\boldsymbol{y}) = A\boldsymbol{y}$ where $C_{f''} = 0$, or also the examples of Section 5.

Theorem 3.3 below estimates the local errors and their propagation under Assumption 3.2. Note that the constants in the error estimates (26) and (27) do not depend on s nor on the stiffness of $\boldsymbol{f}$.

Theorem 3.3. *Let Assumption 3.2 be satisfied and let* $\boldsymbol{f}'(\boldsymbol{y})$ *be symmetric and nonpositive definite. Assuming (16), the error between the exact RKC1 scheme (4) and the mixed-precision RKC1 scheme (15) satisfies*

$$\|\hat{\boldsymbol{y}}^{n+1} - \boldsymbol{y}^{n+1}\|_2 \le \|\hat{\boldsymbol{y}}^n - \boldsymbol{y}^n\|_2 + C_{f''}\Delta t\|\hat{\boldsymbol{y}}^n - \boldsymbol{y}^n\|_2^2 + C_{f''}C_{\boldsymbol{d}}^2\Delta t^3(\|\boldsymbol{f}(\hat{\boldsymbol{y}}^n)\|_2^2 + \|\boldsymbol{f}(\boldsymbol{y}^n)\|_2^2) + \hat{C}(\epsilon + \Delta t)\Delta t^2. \tag{26}$$

Assuming (19), the error between the exact RKC2 scheme (4) (or (17)) and the mixed-precision RKC2 scheme (18) satisfies

$$\|\hat{\boldsymbol{y}}^{n+1} - \boldsymbol{y}^{n+1}\|_2 \le \|\hat{\boldsymbol{y}}^n - \boldsymbol{y}^n\|_2 + C_{f''}\Delta t(1 + \Delta t\|\boldsymbol{f}'(\hat{\boldsymbol{y}}^n)\|_2)\|\hat{\boldsymbol{y}}^n - \boldsymbol{y}^n\|_2^2 + C_{f''}C_{\boldsymbol{d}}^2\Delta t^3(\|\boldsymbol{f}(\hat{\boldsymbol{y}}^n)\|_2^2 + \|\boldsymbol{f}(\boldsymbol{y}^n)\|_2^2) + \hat{C}\Delta t^3. \tag{27}$$

The constants $C_{f''}, C_{\boldsymbol{d}}$ *are those in Assumption 3.2 and* $\hat{C}$ *depends on the definition of* $\hat{\Delta}\boldsymbol{f}_j$ *in either (16) or (19) respectively for RKC1 or RKC2.*

Proof. We prove first (26). Let $\boldsymbol{E}^n = \hat{\boldsymbol{y}}^n - \boldsymbol{y}^n$ and $\boldsymbol{e}_j = \hat{\boldsymbol{d}}_j - \boldsymbol{d}_j$, subtracting (4) from (15) yields

$$\begin{aligned}
\boldsymbol{e}_0 &= \boldsymbol{0}, \qquad \boldsymbol{e}_1 = \mu_1\Delta t(\boldsymbol{f}(\hat{\boldsymbol{y}}^n) - \boldsymbol{f}(\boldsymbol{y}^n)),\\
\boldsymbol{e}_j &= \nu_j\boldsymbol{e}_{j-1} + \kappa_j\boldsymbol{e}_{j-2} + \mu_j\Delta t(\boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \hat{\Delta}\boldsymbol{f}_{j-1} - \boldsymbol{f}(\boldsymbol{y}^n + \boldsymbol{d}_{j-1})), \quad j = 2,\dots,s,\\
\boldsymbol{E}^{n+1} &= \boldsymbol{E}^n + \boldsymbol{e}_s.
\end{aligned} \tag{28}$$

From (16) we have $\boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \hat{\Delta}\boldsymbol{f}_j = \boldsymbol{f}(\hat{\boldsymbol{y}}^n + \hat{\boldsymbol{d}}_j) + \boldsymbol{r}_j$ with $\|\boldsymbol{r}_j\|_2 \le \hat{C}(\epsilon + \Delta t)\Delta t$ and $\hat{C}, \epsilon$ depending on the definition of $\hat{\Delta}\boldsymbol{f}_j$. Hence, using $\hat{\boldsymbol{y}}^n = \boldsymbol{y}^n + \boldsymbol{E}^n$,

$$\boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \hat{\Delta}\boldsymbol{f}_j - \boldsymbol{f}(\boldsymbol{y}^n + \boldsymbol{d}_j) = \boldsymbol{f}(\boldsymbol{y}^n + \boldsymbol{E}^n + \hat{\boldsymbol{d}}_j) - \boldsymbol{f}(\boldsymbol{y}^n + \boldsymbol{d}_j) + \boldsymbol{r}_j = \boldsymbol{f}'(\boldsymbol{y}^n)(\boldsymbol{E}^n + \boldsymbol{e}_j) + \boldsymbol{r}_j + \boldsymbol{t}_j, \tag{29}$$

where $\boldsymbol{t}_j$ is the residual of the Taylor expansions of $\boldsymbol{f}(\boldsymbol{y}^n + \boldsymbol{E}^n + \hat{\boldsymbol{d}}_j)$ and $\boldsymbol{f}(\boldsymbol{y}^n + \boldsymbol{d}_j)$. Since $\|\boldsymbol{f}''(\boldsymbol{y})\|_2 \le C_{f''}$, then

$$\|\boldsymbol{t}_j\|_2 \le C_{f''}(\|\boldsymbol{E}^n\|_2^2 + \|\hat{\boldsymbol{d}}_j\|_2^2 + \|\boldsymbol{d}_j\|_2^2) \le C_{f''}(\|\boldsymbol{E}^n\|_2^2 + C_{\boldsymbol{d}}^2\Delta t^2(\|\boldsymbol{f}(\hat{\boldsymbol{y}}^n)\|_2^2 + \|\boldsymbol{f}(\boldsymbol{y}^n)\|_2^2)),$$

where we used Assumption 3.2. Similarly, we have $\boldsymbol{f}(\hat{\boldsymbol{y}}^n) - \boldsymbol{f}(\boldsymbol{y}^n) = \boldsymbol{f}'(\boldsymbol{y}^n)\boldsymbol{E}^n + \boldsymbol{t}_0$ with $\|\boldsymbol{t}_0\|_2 \le C_{f''}\|\boldsymbol{E}^n\|_2^2$. Inserting (29) into (28) yields

$$\begin{aligned}
\boldsymbol{e}_0 &= \boldsymbol{0}, \qquad \boldsymbol{e}_1 = \mu_1\Delta t(\boldsymbol{f}'(\boldsymbol{y}^n)\boldsymbol{E}^n + \boldsymbol{t}_0),\\
\boldsymbol{e}_j &= \nu_j\boldsymbol{e}_{j-1} + \kappa_j\boldsymbol{e}_{j-2} + \mu_j\Delta t\boldsymbol{f}'(\boldsymbol{y}^n)\boldsymbol{e}_{j-1} + \mu_j\Delta t(\boldsymbol{f}'(\boldsymbol{y}^n)\boldsymbol{E}^n + \boldsymbol{r}_{j-1} + \boldsymbol{t}_{j-1}), \quad j = 2,\dots,s,
\end{aligned}$$

Let $\boldsymbol{r}_0 = \boldsymbol{0}$, Lemma 3.1 i), iii) and $R_s(z) = 1 + \overline{R}_s(z)z$ imply

$$\begin{aligned}
\boldsymbol{E}^{n+1} = \boldsymbol{E}^n + \boldsymbol{e}_s &= \boldsymbol{E}^n + \sum_{j=1}^{s}\frac{b_s}{b_j}U_{s-j}(\omega_0 I + \omega_1\Delta t\boldsymbol{f}'(\boldsymbol{y}^n))\mu_j\Delta t(\boldsymbol{f}'(\boldsymbol{y}^n)\boldsymbol{E}^n + \boldsymbol{r}_{j-1} + \boldsymbol{t}_{j-1})\\
&= \boldsymbol{E}^n + \overline{R}_s(\Delta t\boldsymbol{f}'(\boldsymbol{y}^n))\Delta t\boldsymbol{f}'(\boldsymbol{y}^n)\boldsymbol{E}^n + \Delta t\sum_{j=1}^{s}\frac{b_s}{b_j}U_{s-j}(\omega_0 I + \omega_1\Delta t\boldsymbol{f}'(\boldsymbol{y}^n))\mu_j(\boldsymbol{r}_{j-1} + \boldsymbol{t}_{j-1})\\
&= R_s(\Delta t\boldsymbol{f}'(\boldsymbol{y}^n))\boldsymbol{E}^n + \Delta t\sum_{j=1}^{s}\frac{b_s}{b_j}U_{s-j}(\omega_0 I + \omega_1\Delta t\boldsymbol{f}'(\boldsymbol{y}^n))\mu_j(\boldsymbol{r}_{j-1} + \boldsymbol{t}_{j-1}).
\end{aligned}$$

We conclude with Lemma 3.1 iv) and $\|R_s(\Delta t \boldsymbol{f}'(\boldsymbol{y}^n))\|_2 \leq 1$.

Let us now prove (27). Owing to (19) for the mixed-precision scheme (18), we have

$$\begin{aligned}\hat{\Delta}\boldsymbol{f}_j &= \boldsymbol{f}(\hat{\boldsymbol{y}}^n + c_j\Delta t \boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \hat{\boldsymbol{v}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \boldsymbol{r}_j = \boldsymbol{f}'(\hat{\boldsymbol{y}}^n)(c_j\Delta t \boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \hat{\boldsymbol{v}}_j) + \boldsymbol{r}_j + \boldsymbol{t}_{1,j} \\ &= \boldsymbol{f}'(\boldsymbol{y}^n)\left(c_j\Delta t\left(\boldsymbol{f}(\boldsymbol{y}^n) + \boldsymbol{f}'(\boldsymbol{y}^n)\boldsymbol{E}^n\right) + \hat{\boldsymbol{v}}_j\right) + \boldsymbol{r}_j + \boldsymbol{t}_{1,j} + \boldsymbol{t}_{2,j},\end{aligned} \tag{30}$$

with $\|\boldsymbol{r}_j\|_2 \leq \hat{C}\Delta t^2$ (again, depending on the definition of $\hat{\Delta}\boldsymbol{f}_j$), and

$$\begin{aligned}\|\boldsymbol{t}_{1,j}\|_2 + \|\boldsymbol{t}_{2,j}\|_2 &\leq \frac{1}{2}C_{f''}C_{\boldsymbol{d}}^2\Delta t^2\|\boldsymbol{f}(\hat{\boldsymbol{y}}^n)\|_2^2 + C_{f''}C_{\boldsymbol{d}}\Delta t\|\boldsymbol{E}^n\|_2\|\boldsymbol{f}(\hat{\boldsymbol{y}}^n)\|_2 + C_{f''}\Delta t\|\boldsymbol{f}'(\hat{\boldsymbol{y}}^n)\|_2\|\boldsymbol{E}^n\|_2^2 \\ &\leq C_{f''}C_{\boldsymbol{d}}^2\Delta t^2\|\boldsymbol{f}(\hat{\boldsymbol{y}}^n)\|_2^2 + C_{f''}\left(\frac{1}{2} + \Delta t\|\boldsymbol{f}'(\hat{\boldsymbol{y}}^n)\|_2\right)\|\boldsymbol{E}^n\|_2^2.\end{aligned}$$

For the exact scheme (17), it holds

$$\boldsymbol{f}(\boldsymbol{y}^n + c_j\Delta t \boldsymbol{f}(\boldsymbol{y}^n) + \boldsymbol{v}_j) - \boldsymbol{f}(\boldsymbol{y}^n) = \boldsymbol{f}'(\boldsymbol{y}^n)(c_j\Delta t \boldsymbol{f}(\boldsymbol{y}^n) + \boldsymbol{v}_j) + \boldsymbol{t}_{3,j}, \tag{31}$$

where $\|\boldsymbol{t}_{3,j}\|_2$ can be bound via the same argument by $\|\boldsymbol{t}_{3,j}\|_2 \leq C_{f''}C_{\boldsymbol{d}}^2\Delta t^2\|\boldsymbol{f}(\boldsymbol{y}^n)\|_2^2$. Inserting (30) and (31) into (18) and (17), respectively, then subtracting (17) from (18) and using Lemma 3.1 i) yields

$$\begin{aligned}\hat{\boldsymbol{v}}_s - \boldsymbol{v}_s &= \sum_{j=2}^{s}\frac{b_s}{b_j}U_{s-j}(\omega_0 I + \omega_1\Delta t\boldsymbol{f}'(\boldsymbol{y}^n))\mu_j(c_{j-1}(\Delta t\boldsymbol{f}'(\boldsymbol{y}^n))^2\boldsymbol{E}^n + \Delta t(\boldsymbol{r}_{j-1} + \boldsymbol{t}_{1,j-1} + \boldsymbol{t}_{2,j-1} - \boldsymbol{t}_{3,j-1})) \\ &= \left(R_s(\Delta t\boldsymbol{f}(\boldsymbol{y}^n)) - I - \Delta t\boldsymbol{f}'(\boldsymbol{y}^n)\right)\boldsymbol{E}^n + \Delta t\sum_{j=2}^{s}\frac{b_s}{b_j}U_{s-j}(\omega_0 I + \omega_1\Delta t\boldsymbol{f}'(\boldsymbol{y}^n))\mu_j(\boldsymbol{r}_{j-1} + \boldsymbol{t}_{1,j-1} + \boldsymbol{t}_{2,j-1} - \boldsymbol{t}_{3,j-1}).\end{aligned}$$

Estimate (27) follows from $\boldsymbol{E}^{n+1} = \boldsymbol{E}^n + \Delta t(\boldsymbol{f}(\hat{\boldsymbol{y}}^n) - \boldsymbol{f}(\boldsymbol{y}^n)) + \hat{\boldsymbol{v}}_s - \boldsymbol{v}_s$ with $\boldsymbol{f}(\hat{\boldsymbol{y}}^n) - \boldsymbol{f}(\boldsymbol{y}^n) = \boldsymbol{f}'(\boldsymbol{y}^n)\boldsymbol{E}^n + \boldsymbol{t}_0$ and $\|\boldsymbol{t}_0\|_2 \leq C_{f''}\|\boldsymbol{E}^n\|_2^2/2$. □

Note that thanks to $\hat{\Delta}\boldsymbol{f}_j$ approximating the Jacobian we could use the stability polynomials of the RKC methods and show that the lower order term $\|\hat{\boldsymbol{y}}^n - \boldsymbol{y}^n\|_2$ is not amplified. Due to the nonlinearity we also have a term $\Delta t\|\hat{\boldsymbol{y}}^n - \boldsymbol{y}^n\|_2^2$ in the recursive relation, but for RKC1 this is of higher order. On the other hand, for RKC2 this term is multiplied by $\Delta t\|\boldsymbol{f}'(\hat{\boldsymbol{y}}^n)\|_2$, which can be large and trigger instabilities. We will indeed see in Sections 3.6 and 5.2.1 that the mixed-precision RKC2 scheme is less stable than RKC1.

3.6. Internal error propagation

We now investigate the propagation of rounding errors within one time step. For this purpose, we assume that the problem is linear and therefore $\boldsymbol{f}(\boldsymbol{y}) = A\boldsymbol{y}$, where $A \in \mathbb{R}^{n\times n}$ is a symmetric nonpositive definite matrix. We stress that the next estimates are very pessimistic as they are worst-case bounds and do not take into account rounding error cancellation effects [36].

Theorem 3.4. *Let $\hat{\boldsymbol{y}}^n$ be the solution computed by the mixed-precision RKC schemes (15) or (18) and s such that $\Delta t\rho \leq \beta^p(s,\varepsilon)$, where ρ is the spectral radius of the nonpositive definite matrix A. Then*

$$\hat{\boldsymbol{y}}^{n+1} = R_s(\Delta t A)\hat{\boldsymbol{y}}^n + \boldsymbol{r}_s(\hat{\boldsymbol{y}}^n), \tag{32}$$

where $\boldsymbol{r}_s(\hat{\boldsymbol{y}}^n)$ represents the rounding errors introduced at time step n. Let $\bar{c}$, $\bar{m}$ as in Lemma 2.1 and define

$$C(s,\varepsilon) = 2\bar{c}\bar{m}^2\omega_1\rho\max_{j=0,\dots,s-2}\|U_j(\omega_0 I + \omega_1\Delta t A)\|_2.$$

For the mixed-precision RKC1 method (15) it holds

$$\|\boldsymbol{r}_s(\hat{\boldsymbol{y}}^n)\|_2 \leq \max_{k=1,\dots,s-1}\|R_k(\Delta t A) - I\|_2\left((1 + C(s,\varepsilon)\Delta t u)^{s-1} - 1\right)\|\hat{\boldsymbol{y}}^n\|_2,$$

and for the mixed-precision RKC2 method (18)

$$\|\boldsymbol{r}_s(\hat{\boldsymbol{y}}^n)\|_2 \leq \max_{k=1,\dots,s-1}\|R_k(\Delta t A) - I - c_k\Delta t A\|_2\left((1 + C(s,\varepsilon)\Delta t u)^{s-1} - 1\right)\|\hat{\boldsymbol{y}}^n\|_2.$$

Proof. We first prove the result for the mixed-precision RKC1 method. Scheme (15) with $\boldsymbol{f}(\boldsymbol{y}) = A\boldsymbol{y}$ and $\hat{\Delta}\boldsymbol{f}_j$ as in (20) reads

$$\begin{aligned}\hat{\boldsymbol{d}}_0 &= \boldsymbol{0}, \quad \hat{\boldsymbol{d}}_1 = \mu_1 \Delta t A \hat{\boldsymbol{y}}^n,\\ \hat{\boldsymbol{d}}_j &= \nu_j \hat{\boldsymbol{d}}_{j-1} + \kappa_j \hat{\boldsymbol{d}}_{j-2} + \mu_j \Delta t \hat{A} \hat{\boldsymbol{d}}_{j-1} + \mu_j \Delta t A \hat{\boldsymbol{y}}^n\\ &= \nu_j \hat{\boldsymbol{d}}_{j-1} + \kappa_j \hat{\boldsymbol{d}}_{j-2} + \mu_j \Delta t A \hat{\boldsymbol{d}}_{j-1} + \mu_j \Delta t (A\hat{\boldsymbol{y}}^n + \Delta A_{j-1} \hat{\boldsymbol{d}}_{j-1}), \quad j = 2, \dots, s.\end{aligned}$$

Lemma 3.1 i) implies

$$\hat{\boldsymbol{d}}_k = \sum_{j=1}^{k} \frac{b_k}{b_j} U_{k-j}(\omega_0 I + \omega_1 \Delta t A) \mu_j \Delta t (A\hat{\boldsymbol{y}}^n + \Delta A_{j-1} \hat{\boldsymbol{d}}_{j-1}) = \overline{R}_k(\Delta t A) \Delta t A \hat{\boldsymbol{y}}^n + \boldsymbol{r}_k(\hat{\boldsymbol{y}}^n) = (R_k(\Delta t A) - I)\hat{\boldsymbol{y}}^n + \boldsymbol{r}_k(\hat{\boldsymbol{y}}^n), \quad (33)$$

with

$$\boldsymbol{r}_k(\hat{\boldsymbol{y}}^n) = \sum_{j=1}^{k} \frac{b_k}{b_j} U_{k-j}(\omega_0 I + \omega_1 \Delta t A) \mu_j \Delta t \Delta A_{j-1} \hat{\boldsymbol{d}}_{j-1} = 2\omega_1 \Delta t \sum_{j=1}^{k-1} \frac{b_k}{b_j} U_{k-j-1}(\omega_0 I + \omega_1 \Delta t A) \Delta A_j \hat{\boldsymbol{d}}_j, \quad (34)$$

where for the second equality we used $\hat{\boldsymbol{d}}_0 = \boldsymbol{0}$ and (5). Equation (33) and $\hat{\boldsymbol{y}}^{n+1} = \hat{\boldsymbol{y}}^n + \hat{\boldsymbol{d}}_s$ yield (32). Lemma 2.1, (34) and $b_k/b_j \le 1$ imply

$$\|\boldsymbol{r}_k(\hat{\boldsymbol{y}}^n)\|_2 \le 2\omega_1 \Delta t \max_{j=0,\dots,k-2} \|U_j(\omega_0 I + \omega_1 \Delta t A)\|_2 \sum_{j=1}^{k-1} \|\Delta A_j \hat{\boldsymbol{d}}_j\|_2 \le C(s,\varepsilon)\Delta t u \sum_{j=1}^{k-1} \|\hat{\boldsymbol{d}}_j\|_2. \quad (35)$$

Using (35) in (33) we obtain

$$\|\hat{\boldsymbol{d}}_k\|_2 \le \|R_k(\Delta t A) - I\|_2 \|\hat{\boldsymbol{y}}^n\|_2 + C(s,\varepsilon)\Delta t u \sum_{j=1}^{k-1} \|\hat{\boldsymbol{d}}_j\|_2,$$

and prove, recursively, that

$$\|\boldsymbol{r}_s(\hat{\boldsymbol{y}}^n)\|_2 \le C(s,\varepsilon)\Delta t u \sum_{j=1}^{s-1} \|\hat{\boldsymbol{d}}_j\|_2 \le \max_{k=1,\dots,s-1} \|R_k(\Delta t A) - I\|_2 \left((1 + C(s,\varepsilon)\Delta t u)^{s-1} - 1\right) \|\hat{\boldsymbol{y}}^n\|_2.$$

For the mixed-precision RKC2 scheme we use $\hat{\Delta}\boldsymbol{f}_j$ as in (21), $R_k(z) - 1 - c_k z = z^2 \sum_{j=2}^{k} \frac{b_k}{b_j} U_{k-j}(\omega_0 + \omega_1 z)\mu_j c_{j-1}$, and we get (32) again, only this time with

$$\boldsymbol{r}_k(\hat{\boldsymbol{y}}^n) = 2\omega_1 \Delta t \sum_{j=2}^{k-1} \frac{b_k}{b_j} U_{k-j-1}(\omega_0 I + \omega_1 \Delta t A) \Delta A_j \hat{\boldsymbol{v}}_j, \qquad \text{and} \qquad \hat{\boldsymbol{v}}_k = (R_k(\Delta t A) - I - c_k \Delta t A)\hat{\boldsymbol{y}}_n + \boldsymbol{r}_k(\hat{\boldsymbol{y}}^n).$$

The same argument as for RKC1 concludes the proof. □

Relation (32) has a crucial difference with respect to the rounding error estimate given in [60, Eq. (3.10)]. In [60] the perturbations at each stage were assumed to be independent, while here we are considering the propagation of previous perturbations. Indeed, in (34) each perturbation $\Delta A_k \hat{\boldsymbol{d}}_k$ depends on $\hat{\boldsymbol{d}}_k$, which in turn depends on $\Delta A_j \hat{\boldsymbol{d}}_j$ for $j = 1 \dots, k-1$ (33). The estimate found in [60] is for a standard RKC method (4) in which all operations are performed with the same precision u, and we can thus compare their result with the one we obtained for our mixed-precision scheme. Let $\psi_{\max} \coloneqq \max_{k=1,\dots,s-1} \|R_k(\Delta t A) - I\|_2$ for RKC1 and $\psi_{\max} \coloneqq \max_{k=1,\dots,s-1} \|R_k(\Delta t A) - I - c_k \Delta t A\|_2$ for RKC2. Then, from (32),

$$\|\hat{\boldsymbol{y}}^{n+1}\|_2 \le (1 + \psi_{\max}((1 + C(s,\varepsilon)\Delta t u)^{s-1} - 1))\|\hat{\boldsymbol{y}}^n\|_2. \quad (36)$$

Using the bound $\max_{j=0,\dots,s-2} \|U_j(\omega_0 I + \omega_1 \Delta t A)\|_2 \le \tilde{C}(s-1)$, with $\tilde{C}$ close to 1, we have $C(s,\varepsilon)\Delta t u \le C\omega_1 \Delta t \rho (s-1) u$ for some constant C. In the asymptotic regime

$$((1 + C(s,\varepsilon)\Delta t u)^{s-1} - 1) \approx C(s,\varepsilon)\Delta t u (s-1) \le C\omega_1 \Delta t \rho (s-1)^2 u \le C\Delta t \rho u,$$

where we used $\omega_1 = O(1/s^2)$. Moreover, $\psi_{\max} = O(\Delta t^p)$ and therefore the stability estimate of the mixed-precision RKC schemes behaves as $1 + C\rho u \Delta t^{p+1} \leq 1 + C s^2 u \Delta t^p$. In contrast, in [60] the authors find that the constant in the stability estimate behaves as $1 + C s^2 u$ and is independent from Δt. This difference stems from the fact that standard RKC schemes are not order preserving, i.e. if they are run entirely in low-precision they do not converge and their error stagnates (or grows like $O(u\Delta t^{-1})$, cf. [21]) as $\Delta t \to 0$.

In a non-asymptotic regime $\psi_{\max} = O(1)$ for RKC1, and $\psi_{\max} = O(\Delta t \rho)$ for RKC2, where $\Delta t \rho = O(s^2)$. Thus the constant in estimate (36) grows at least as $1 + C s^{2p} u$ and suggests that the scheme can become unstable whenever $s^{2p} u$ is large. We investigate the stability of the mixed-precision RKC schemes in practice in Section 5 and verify that the RKC1 scheme remains stable for a very large number of stages (we stopped our experiments at $s = 512$) even when the low precision computations are performed in half-precision. In contrast, the double/half mixed-precision RKC2 scheme grows unstable faster due to the fact that we iterate over $\hat{\boldsymbol{v}}_j$ instead of $\hat{\boldsymbol{d}}_j$ and therefore $\psi_{\max} = O(\Delta t \rho) = O(s^2)$. In practice instability with RKC2 seem to arise when $s^2 u > 1$. If the low-precision computations are instead performed in single precision, for which u is much smaller, the RKC2 scheme remains stable even at large values of s, cf. Section 5.

Unfortunately, we were unable to prove stability of the schemes analytically. The main difficulty stems from the fact that rounding errors affect all frequencies, and destroy any spectral relation between $\hat{\boldsymbol{y}}^n$ and $\hat{\boldsymbol{d}}_j, \hat{\boldsymbol{v}}_j$, therefore forbidding any kind of stability analysis based on: 1) damping effects due to eigenvalues far from the origin, and 2) accuracy for those close to zero. Indeed, the accuracy properties of the stability polynomial would need to be taken into account to achieve better estimates: for z close to zero we have $U_j(\omega_0 + \omega_1 z) = O(j+1)$, causing roundoff errors $\Delta A_j \hat{\boldsymbol{d}}_j$ with low frequencies to be amplified (cf. (34)). If rounding errors preserved spectral relations, these errors would then be compensated by the fact that $R_k(z) - I \approx 0$ (cf. (33)). However, frequencies of $\hat{\boldsymbol{y}}_n$ and $\Delta A_j \hat{\boldsymbol{d}}_j$ are uncorrelated, making such an analysis impossible. Under the assumption that the smallest (in magnitude) eigenvalue of $\Delta t A$ is sufficiently separated from the origin, we can prove that the schemes are stable by using damping properties only. This is possible thanks to the fact that $|U_j(\omega_0 + \omega_1 z)| \leq 2$ for z sufficiently far from 0. However, this assumption requires A to have a small condition number, which is a very restrictive condition. Interestingly, lack of separation between the eigenvalues of $\Delta t A$ and the origin does not seem to affect stability in practical experiments (cf. Section 5).

4. Mixed-precision multirate RKC method

In this section we consider a multirate differential equation of the type (2), where $\boldsymbol{f}_S$ is an expensive, but only mildly stiff term associated to relatively slow (S) time-scales and $\boldsymbol{f}_F$ is a cheap, yet severely stiff term associated to fast (F) time scales. Typical applications are chemical kinetics problems with different reaction rates, electric circuits with active and latent components, and systems stemming from the spatial discretization of diffusion-dominated parabolic PDEs over graded meshes. In this latter case, $\boldsymbol{f}_F$ and $\boldsymbol{f}_S$ typically correspond to the discrete diffusion operator over the fine and coarse degrees-of-freedom respectively (i.e. over refined and coarse mesh portions).

When an explicit stabilized scheme as RKC is applied to (2), the number of stages s is determined by the stiffness of $\boldsymbol{f}_F$, even when $\boldsymbol{f}_F$ has very few severely stiff degrees of freedom. Hence, the number s of expensive $\boldsymbol{f}_S$ evaluations depends on $\boldsymbol{f}_F$ and this relation destroys the efficiency of the RKC scheme. In Section 4.1 below we recall the mRKC scheme from [4], where the evaluation of $\boldsymbol{f}_F, \boldsymbol{f}_S$ is decoupled and the number of $\boldsymbol{f}_S$ evaluations depends solely on the mild stiffness of $\boldsymbol{f}_S$ itself. Hence, the mRKC scheme is barely affected by few severely stiff terms and recovers the original efficiency of RKC methods without sacrificing accuracy.

4.1. The multirate RKC method

The mRKC scheme is based on the modified equation

$$\boldsymbol{y}_\eta' = \boldsymbol{f}_\eta(\boldsymbol{y}_\eta), \qquad \boldsymbol{y}_\eta(0) = \boldsymbol{y}^0, \tag{37}$$

for (2). The modified right-hand side $\boldsymbol{f}_\eta$, called *averaged force*, depends on a free parameter $\eta \geq 0$ and is a good approximation to the exact $\boldsymbol{f} = \boldsymbol{f}_F + \boldsymbol{f}_S$. Yet, for the right choice of η, the stiffness of $\boldsymbol{f}_\eta$ depends on $\boldsymbol{f}_S$ only and integration of (37) with an RKC scheme is cheaper than (2). Evaluation of $\boldsymbol{f}_\eta$ requires the solution of a stiff, yet cheap auxiliary problem, that is also approximated using an RKC scheme.

The averaged force.
Before defining the mRKC scheme we introduce the averaged force $\boldsymbol{f}_\eta$ and briefly discuss its properties. We refer to [4] for further details.

Definition 4.1. For $\eta > 0$, the averaged force $\boldsymbol{f}_\eta : \mathbb{R}^n \to \mathbb{R}^n$ is defined as

$$\boldsymbol{f}_\eta(\boldsymbol{y}) = \frac{1}{\eta}(\boldsymbol{u}(\eta) - \boldsymbol{y}), \tag{38}$$

where *the auxiliary solution* $\boldsymbol{u} : [0, \eta] \to \mathbb{R}^n$ is defined by *the auxiliary problem*

$$\boldsymbol{u}' = \boldsymbol{f}_F(\boldsymbol{u}) + \boldsymbol{f}_S(\boldsymbol{y}), \qquad \boldsymbol{u}(0) = \boldsymbol{y}. \tag{39}$$

For $\eta = 0$, let $\boldsymbol{f}_0 = \boldsymbol{f}$ (note that $\boldsymbol{f}_0 = \lim_{\eta \to 0^+} \boldsymbol{f}_\eta$).

Hence, an auxiliary problem (39) with initial condition $\boldsymbol{y} = \boldsymbol{y}_\eta(t)$ must be solved whenever $\boldsymbol{f}_\eta(\boldsymbol{y}_\eta(t))$ is evaluated in (37). Using (38) and (39) we compute

$$\boldsymbol{f}_\eta(\boldsymbol{y}) = \frac{1}{\eta}\int_0^\eta \boldsymbol{u}'(s)\,\mathrm{d}s = \boldsymbol{f}_S(\boldsymbol{y}) + \frac{1}{\eta}\int_0^\eta \boldsymbol{f}_F(\boldsymbol{u}(s))\,\mathrm{d}s,$$

thus $\boldsymbol{f}_\eta$ evaluates $\boldsymbol{f}_S$ exactly and computes an average of $\boldsymbol{f}_F$ along the auxiliary solution $\boldsymbol{u}$. This average has a damping effect on $\boldsymbol{f}_F$ and reduces its stiffness. In the next lemma, proved in [4], we show in a particular case the effects of the average and the size of η.

Lemma 4.1. *Let* $\boldsymbol{f}_F(\boldsymbol{y}) = A_F\,\boldsymbol{y}$ *with* $A_F \in \mathbb{R}^{n\times n}$. *Then*

$$\boldsymbol{f}_\eta(\boldsymbol{y}) = \varphi(\eta A_F)\boldsymbol{f}(\boldsymbol{y}), \tag{40}$$

where

$$\varphi(z) = \frac{e^z - 1}{z}, \quad \textit{for } z \neq 0, \qquad \textit{and} \qquad \varphi(0) = 1.$$

In (40), the $\varphi(\eta A_F)$ term has a damping effect on $\boldsymbol{f}$ owing to the negative definiteness of the matrix A_F, and the exponential-like behaviour of $\varphi(z)$. In fact, $\varphi(z)$ satisfies $\lim_{z\to-\infty}\varphi(z) = 0$ and $\varphi(z) \in (0, 1)$ for all $z < 0$. The free parameter η in (40) can be used to tune this damping effect. Let ρ_η, and ρ_S be the spectral radii of $\boldsymbol{f}_\eta$, and $\boldsymbol{f}_S$, respectively. It was shown in [4] that $\rho_\eta \leq \rho_S$ already holds for η relatively small, and therefore the stiffness of (37) does not depend on $\boldsymbol{f}_F$ anymore, but solely on $\boldsymbol{f}_S$. In [4], the authors also proved that $\boldsymbol{y}_\eta$ is an $O(\eta)$ approximation of $\boldsymbol{y}$ and that, in some cases, $\boldsymbol{f}_\eta$ inherits the contractivity properties of $\boldsymbol{f}$.

The mRKC scheme.
The multirate RKC scheme is nothing else than an s-stage RKC scheme applied to (37), with s depending solely on ρ_S, the spectral radius of $\boldsymbol{f}_S$. Whenever $\boldsymbol{f}_\eta$ must be evaluated, it is approximated by solving the auxiliary problem (39) with an m-stage RKC method, where m depends on ρ_F, the spectral radius of $\boldsymbol{f}_F$. However, integration of (39) is cheap since $\boldsymbol{f}_S$ is frozen at the initial value. The tuning parameter η is chosen so that the approximation to $\boldsymbol{f}_\eta$ is less stiff than $\boldsymbol{f}_S$, and thus the s-stage RKC scheme remains stable. More precisely, the number of stages s, m are the smallest integers satisfying

$$\Delta t\rho_S \leq \beta s^2, \qquad \eta\rho_F \leq \beta m^2, \qquad \text{with} \qquad \eta = \frac{6\Delta t}{\beta s^2}\frac{m^2}{m^2 - 1} \tag{41}$$

and $\beta = 2 - 4\varepsilon/3$ (see (9)). One step of the mRKC scheme is then given by

$$\begin{cases} \boldsymbol{d}_0 = \boldsymbol{0}, \quad \boldsymbol{d}_1 = \mu_1\Delta t\overline{\boldsymbol{f}}_\eta(\boldsymbol{y}^n), \\ \boldsymbol{d}_j = \nu_j\boldsymbol{d}_{j-1} + \kappa_j\boldsymbol{d}_{j-2} + \mu_j\Delta t\overline{\boldsymbol{f}}_\eta(\boldsymbol{y}^n + \boldsymbol{d}_{j-1}) \quad j = 2, \ldots, s, \\ \boldsymbol{y}^{n+1} = \boldsymbol{y}^n + \boldsymbol{d}_s, \end{cases} \tag{42}$$

where the parameters μ_j, ν_j, κ_j are those of the RKC1 scheme defined in (5) and (6) and $\overline{\boldsymbol{f}}_\eta(\boldsymbol{y})$ is a numerical approximation to $\boldsymbol{f}_\eta(\boldsymbol{y}) = (\boldsymbol{u}(\eta) - \boldsymbol{y})/\eta$, cf. (38). Hence, in the mRKC scheme (42), at each evaluation of $\overline{\boldsymbol{f}}_\eta(\boldsymbol{y}^n + \boldsymbol{d}_j)$ an approximation of $(\boldsymbol{u}(\eta) - \boldsymbol{y})/\eta$ is computed, with $\boldsymbol{u}(\eta)$ as in (39), and $\boldsymbol{y} = \boldsymbol{y}^n + \boldsymbol{d}_j$. This is performed by integrating (39) with one m-stage RKC step of size η in which each stage is divided by η itself:

$$\begin{cases} \boldsymbol{h}_0 = \boldsymbol{0}, \quad \boldsymbol{h}_1 = \alpha_1(\boldsymbol{f}_F(\boldsymbol{y}) + \boldsymbol{f}_S(\boldsymbol{y})), \\ \boldsymbol{h}_j = \beta_j \boldsymbol{h}_{j-1} + \gamma_j \boldsymbol{h}_{j-2} + \alpha_j(\boldsymbol{f}_F(\boldsymbol{y} + \eta \boldsymbol{h}_{j-1}) + \boldsymbol{f}_S(\boldsymbol{y})) \quad j = 2, \dots, m, \\ \overline{\boldsymbol{f}}_\eta(\boldsymbol{y}) = \boldsymbol{h}_m. \end{cases} \tag{43}$$

Here, the parameters $\alpha_j, \beta_j, \gamma_j$ of the m-stage RKC scheme (43) are given by [4]

$$\upsilon_0 = 1 + \varepsilon/m^2, \qquad \upsilon_1 = T_m(\upsilon_0)/T_m'(\upsilon_1), \qquad a_j = 1/T_j(\upsilon_0) \qquad \text{for } j = 0, \dots, m$$

and $\alpha_1 = \upsilon_1/\upsilon_0$,

$$\alpha_j = 2\upsilon_1 a_j/a_{j-1}, \qquad \beta_j = 2\upsilon_0 a_j/a_{j-1}, \qquad \gamma_j = -a_j/a_{j-2} \qquad \text{for } j = 2, \dots, m.$$

To compute m and η in (41), we insert $\eta = 6\Delta t m^2/(\beta s^2(m^2 - 1))$ into $\eta\rho_F \le \beta m^2$, and first compute m, then η. The mRKC method is given by (41)–(43) and its stability and first-order accuracy were proved in [4].

4.2. The mixed-precision multirate RKC method

Roughly speaking, the mRKC scheme (41)–(43) is obtained by applying an RKC1 scheme to (37) and a second RKC1 scheme to (39) whenever the right-hand side needs to be evaluated. In our mixed-precision mRKC scheme we instead apply the mixed-precision RKC1 method (15) to (37) and then another mixed-precision RKC1 scheme to (39). The resulting method then only requires one evaluation of $\boldsymbol{f}_F$ and $\boldsymbol{f}_S$ in high precision (per timestep), with all the subsequent evaluations performed in low precision. We now present our mixed-precision mRKC scheme, and we analyze its accuracy in Section 4.3.

The mixed-precision mRKC scheme.
Let s, m and η be as in (41). One step of the mixed-precision mRKC scheme is given by

$$\begin{cases} \hat{\boldsymbol{d}}_0 = \boldsymbol{0}, \quad \hat{\boldsymbol{d}}_1 = \mu_1 \Delta t \tilde{\boldsymbol{f}}_\eta(\hat{\boldsymbol{y}}^n), \\ \hat{\boldsymbol{d}}_j = \nu_j \hat{\boldsymbol{d}}_{j-1} + \kappa_j \hat{\boldsymbol{d}}_{j-2} + \mu_j \Delta t(\tilde{\boldsymbol{f}}_\eta(\hat{\boldsymbol{y}}^n) + \hat{\Delta}\boldsymbol{f}_{\eta,j-1}), \quad j = 2, \dots, s, \\ \hat{\boldsymbol{y}}^{n+1} = \hat{\boldsymbol{y}}^n + \hat{\boldsymbol{d}}_s, \end{cases} \tag{44}$$

where $\tilde{\boldsymbol{f}}_\eta(\boldsymbol{y})$ is given by

$$\begin{cases} \tilde{\boldsymbol{h}}_0 = \boldsymbol{0}, \quad \tilde{\boldsymbol{h}}_1 = \alpha_1(\boldsymbol{f}_F(\boldsymbol{y}) + \boldsymbol{f}_S(\boldsymbol{y})), \\ \tilde{\boldsymbol{h}}_j = \beta_j \tilde{\boldsymbol{h}}_{j-1} + \gamma_j \tilde{\boldsymbol{h}}_{j-2} + \alpha_j(\boldsymbol{f}_F(\boldsymbol{y}) + \boldsymbol{f}_S(\boldsymbol{y}) + \hat{\Delta}\boldsymbol{f}_{F,j-1}), \quad j = 2, \dots, m, \\ \tilde{\boldsymbol{f}}_\eta(\boldsymbol{y}) = \tilde{\boldsymbol{h}}_m. \end{cases} \tag{45}$$

The $\hat{\Delta}\boldsymbol{f}_{F,j}$ are computed in low precision and must satisfy

$$\hat{\Delta}\boldsymbol{f}_{F,j} = \boldsymbol{f}_F(\boldsymbol{y} + \eta\tilde{\boldsymbol{h}}_j) - \boldsymbol{f}_F(\boldsymbol{y}) + O(\epsilon\eta + \eta^2), \tag{46}$$

as $\eta \to 0$, where $\epsilon \ge 0$ is a small constant. For the evaluation of $\hat{\Delta}\boldsymbol{f}_{F,j}$ we can again employ the techniques described in Section 3.2, with $\boldsymbol{f}$ and Δt replaced by $\boldsymbol{f}_F$ and η, respectively. The low precision $\{\hat{\Delta}\boldsymbol{f}_{\eta,j}\}_{j=1}^{s-1}$ terms in (44) must again satisfy (cf. (16))

$$\hat{\Delta}\boldsymbol{f}_{\eta,j} = \Delta\boldsymbol{f}_{\eta,j} + O(\epsilon\Delta t + \Delta t^2), \quad \forall j, \tag{47}$$

where $\Delta \boldsymbol{f}_{\eta,j} = \overline{\boldsymbol{f}}_\eta(\hat{\boldsymbol{y}}^n + \hat{\boldsymbol{d}}_j) - \overline{\boldsymbol{f}}_\eta(\hat{\boldsymbol{y}}^n)$. Again, we can employ the strategies from Section 3.2. For instance, one can use automatic differentiation or alternatively define

$$\hat{\Delta}\boldsymbol{f}_{\eta,j} = \delta^{-1}\left(\hat{\boldsymbol{f}}_\eta(\hat{\boldsymbol{y}}^n + \delta\,\hat{\boldsymbol{d}}_j) - \tilde{\boldsymbol{f}}_\eta(\hat{\boldsymbol{y}}^n)\right), \qquad \delta = \frac{\sqrt{u}}{\Delta t} \tag{48}$$

with $\hat{\boldsymbol{f}}_\eta(\boldsymbol{y})$ given by

$$\begin{cases} \hat{\boldsymbol{h}}_0 = \boldsymbol{0}, \quad \hat{\boldsymbol{h}}_1 = \alpha_1(\hat{\boldsymbol{f}}_F(\boldsymbol{y}) + \hat{\boldsymbol{f}}_S(\boldsymbol{y})), \\ \hat{\boldsymbol{h}}_j = \beta_j\hat{\boldsymbol{h}}_{j-1} + \gamma_j\hat{\boldsymbol{h}}_{j-2} + \alpha_j(\hat{\boldsymbol{f}}_F(\boldsymbol{y} + \eta\hat{\boldsymbol{h}}_{j-1}) + \hat{\boldsymbol{f}}_S(\boldsymbol{y})) \quad j = 2,\dots,m, \\ \hat{\boldsymbol{f}}_\eta(\boldsymbol{y}) = \hat{\boldsymbol{h}}_m. \end{cases} \tag{49}$$

We prove in Lemma B.2 that if $\Delta t \le \sqrt{u}$ then $\hat{\Delta}\boldsymbol{f}_{\eta,j}$ defined as in (48) and (49) satisfies (47) with $\epsilon = \sqrt{u}$. Condition $\Delta t \le \sqrt{u}$ is very weak since the method is intended to be used when Δt is smaller or proportional to $u \ll \sqrt{u}$. Note that the difference between $\tilde{\boldsymbol{f}}_\eta$ and $\hat{\boldsymbol{f}}_\eta$ is that in (45) the functions $\boldsymbol{f}_F, \boldsymbol{f}_S$ are evaluated once in high precision while in (49) they are always evaluated in low precision. Hence, $\hat{\boldsymbol{f}}_\eta$ is a simple low precision evaluation of $\overline{\boldsymbol{f}}_\eta$ (compare (43) and (49)), while $\tilde{\boldsymbol{f}}_\eta$ has the lowest order term evaluated in high precision (exactly under Assumption 3.1). Again, we remark that our mixed-precision mRKC scheme only needs one evaluation of $\boldsymbol{f}_F$, and $\boldsymbol{f}_S$ in high precision per timestep.

4.3. Convergence analysis

We compute here the Taylor expansion of the mixed-precision mRKC scheme, as we did in Theorem 3.2 for the mixed-precision RKC schemes. For the sake of brevity, we omit the convergence analysis in the sense of Theorem 3.3, and the rounding error propagation analysis of the mixed-precision mRKC scheme. The results are similar as for the mixed-precision RKC1 scheme, only with added technicalities in the proofs due to the use of embedded methods. Numerically, we observe that the mixed-precision mRKC scheme is more stable than the mixed-precision RKC1 scheme thanks to the reduced stiffness of the right-hand side, and the decreased number of stages. Therefore, in this section we only prove that the mixed-precision mRKC scheme (44) to (47) is first-order preserving by performing a Taylor expansion of the numerical solution.

In order to prove the main convergence result, Theorem 4.3, we first need a technical lemma.

Lemma 4.2. *Let $\boldsymbol{y} \in \mathbb{R}^n$, $\tilde{\boldsymbol{f}}_\eta$ as in (45), and $\overline{\boldsymbol{f}}_\eta$ as in (43). Then $\tilde{\boldsymbol{f}}_\eta(\boldsymbol{y}) = \overline{\boldsymbol{f}}_\eta(\boldsymbol{y}) + O(\epsilon\eta + \eta^2)$.*

Proof. From (46) we have $\boldsymbol{f}_F(\boldsymbol{y}) + \hat{\Delta}\boldsymbol{f}_{F,j} = \boldsymbol{f}_F(\boldsymbol{y} + \eta\tilde{\boldsymbol{h}}_j) + \boldsymbol{r}_j$ with $\|\boldsymbol{r}_j\|_2 \le \hat{C}(\epsilon + \eta)\eta$ and $\hat{C}$ depending on the definition of $\hat{\Delta}\boldsymbol{f}_{F,j}$. Hence, subtracting (43) from (45) yields

$$\begin{aligned} &\boldsymbol{e}_0 = \boldsymbol{0}, \qquad \boldsymbol{e}_1 = \boldsymbol{0}, \\ &\boldsymbol{e}_j = \beta_j\boldsymbol{e}_{j-1} + \gamma_j\boldsymbol{e}_{j-2} + \alpha_j(\boldsymbol{f}_F(\boldsymbol{y} + \eta\tilde{\boldsymbol{h}}_{j-1}) - \boldsymbol{f}_F(\boldsymbol{y} + \eta\boldsymbol{h}_{j-1}) + \boldsymbol{r}_{j-1}) \\ &\quad = \beta_j\boldsymbol{e}_{j-1} + \gamma_j\boldsymbol{e}_{j-2} + \alpha_j\boldsymbol{f}_F'(\boldsymbol{y})\eta\boldsymbol{e}_{j-1} + \alpha_j(\boldsymbol{r}_{j-1} + \boldsymbol{t}_{j-1}) \quad j = 2,\dots,m, \end{aligned}$$

with $\|\boldsymbol{t}_j\|_2 \le C(\|\boldsymbol{h}_j\|_2^2 + \|\tilde{\boldsymbol{h}}_j\|_2^2)\eta^2$ and C depending on $\boldsymbol{f}_F''$. Using Lemma 3.1 follows $\tilde{\boldsymbol{f}}_\eta(\boldsymbol{y}) - \overline{\boldsymbol{f}}_\eta(\boldsymbol{y}) = \boldsymbol{e}_m = O(\epsilon\eta + \eta^2)$. □

We are now ready to prove the following theorem, which ensures that our mixed-precision mRKC method is indeed first-order preserving.

Theorem 4.3. *The mixed-precision mRKC scheme (44) to (47) satisfies*

$$\hat{\boldsymbol{y}}^{n+1} = \hat{\boldsymbol{y}}^n + \Delta t(\boldsymbol{f}_F(\hat{\boldsymbol{y}}^n) + \boldsymbol{f}_S(\hat{\boldsymbol{y}}^n)) + O(\epsilon\Delta t^2 + \Delta t^2).$$

Proof. We proceed similarly as in Theorem 3.2. By applying Lemma 3.1 to (44), with $M = 0$ and $\boldsymbol{r}_j = \Delta t(\tilde{\boldsymbol{f}}_\eta(\hat{\boldsymbol{y}}^n) + \hat{\Delta}\boldsymbol{f}_{\eta,j-1})$ for $j = 1,\dots,s$ (with $\hat{\Delta}\boldsymbol{f}_{\eta,0} = \boldsymbol{0}$), we obtain

$$\begin{aligned} \hat{\boldsymbol{y}}^{n+1} &= \hat{\boldsymbol{y}}^n + \sum_{j=1}^{s}\frac{b_s}{b_j}U_{s-j}(\omega_0)\mu_j\Delta t(\tilde{\boldsymbol{f}}_\eta(\hat{\boldsymbol{y}}^n) + \hat{\Delta}\boldsymbol{f}_{\eta,j-1}) = \hat{\boldsymbol{y}}^n + \Delta t\tilde{\boldsymbol{f}}_\eta(\hat{\boldsymbol{y}}^n) + \Delta t\sum_{j=1}^{s}\frac{b_s}{b_j}U_{s-j}(\omega_0)\mu_j\hat{\Delta}\boldsymbol{f}_{\eta,j-1} \\ &= \hat{\boldsymbol{y}}^n + \Delta t(\boldsymbol{f}_F(\hat{\boldsymbol{y}}^n) + \boldsymbol{f}_S(\hat{\boldsymbol{y}}^n)) + O(\epsilon\eta\Delta t + \eta^2\Delta t + \eta\Delta t + \Delta t^2 + \epsilon\Delta t^2 + \Delta t^3), \end{aligned}$$

where we used Lemma 4.2, the relation $\overline{\boldsymbol{f}}_\eta(\boldsymbol{y}) = \boldsymbol{f}_F(\boldsymbol{y}) + \boldsymbol{f}_S(\boldsymbol{y}) + O(\eta)$ [4], and we applied Lemma 3.1 iv) to the last summation. We conclude using the fact that $\eta \leq 8\Delta t$ (usually $\eta \ll \Delta t$). □

5. Numerical experiments

In this section we test the algorithms and theory presented in the paper. We will often compare our order-preserving mixed-precision methods to some more naïve mixed-precision implementations that perform all function evaluations in low precision and only vector sums and multiplications in high precision; therefore these schemes do not converge (not even under Assumption 3.1, cf. Definition 3.1). In this section we will refer to these naïve schemes as not order-preserving or as "standard" mixed-precision schemes.

5.1. Test problems and computational setup

5.1.1. Problem 1: Nonlinear reaction-diffusion equation

Problem 1 is a standard nonlinear reaction-diffusion equation in d-dimensions with Dirichlet boundary conditions:

$$\begin{cases} \dot{\mathrm{u}}(t,\boldsymbol{x}) = \mathcal{D}\Delta\,\mathrm{u} - h(\mathrm{u}) + f_1(\boldsymbol{x}), & \boldsymbol{x} \in D = [0,1]^d, \quad t \in [0,T], \\ \mathrm{u}(0,\boldsymbol{x}) = 1 & \boldsymbol{x} \in D = [0,1]^d, \\ \mathrm{u}(t,\boldsymbol{x}) = 1 & \boldsymbol{x} \in \partial D, \qquad\quad t \in [0,T], \end{cases}$$

where $d \in \{1,2,3\}$, $T = 1$, $\mathcal{D} = 100$, $h(\mathrm{u}) = \mathrm{u}^2$, and $f_1(x)$ is chosen so that the exact solutions in 1D, 2D, and 3D at steady-state are

$$\mathrm{u}_{1D}(\infty, x) = (4\,x(1-x))^2 + 1, \quad \mathrm{u}_{2D}(\infty,\boldsymbol{x}) = (16\,xy(1-x)(1-y))^2 + 1,$$
$$\mathrm{u}_{3D}(\infty,\boldsymbol{x}) = (64\,xyz(1-x)(1-y)(1-z))^2 + 1.$$

We pick D_h, the mesh of D, to be uniform with $d!N^d$ cells, where $N \in \mathbb{N}$, to be given later.

5.1.2. Problem 2: Heat equation on a graded L-shaped domain

Problem 2 is the classic heat equation on an L-shaped 2D domain D_L with a near-singular forcing term:

$$\begin{cases} \dot{\mathrm{u}}(t,\boldsymbol{x}) = \Delta\,\mathrm{u} + f_2(\boldsymbol{x}), & \boldsymbol{x} \in D_L, \quad t \in [0,T], \\ \mathrm{u}(0,\boldsymbol{x}) = 1 & \boldsymbol{x} \in D_L, \\ \mathrm{u}(t,\boldsymbol{x}) = 1 & \boldsymbol{x} \in \partial D_L, \quad t \in [0,T], \end{cases}$$

Here $T = 1$, $f_2(x) = -10\log(\theta(x,y))$ where $\theta(x,y) = 2((x-0.501)^2 + (y-0.501)^2)$, and D_L is the polygon delimited by the points $\{(0,0), (1,0), (1,0.5), (0.5,0.5), (0.5,1), (0,1)\}$. We take D_L^h, the mesh of D_L, to be unstructured and graded near the re-entrant corner $(0.5,0.5)$. More specifically, D_L^h is constructed so that the size of its cells is roughly given by $N^{-3/2} + N^{-1}\,(1 - \exp(-20\log(2)\theta(x,y)))$, where the value of $N \in \mathbb{N}$ will be given later. We will use Problem 2 to test the multirate RKC method with degrees-of-freedom splitting presented in [4]. In this case we split the matrix A into $A = A_F + A_S$, ($\boldsymbol{f}_F = A_F$, $\boldsymbol{f}_S = A_S$, cf. Section 4.1) so that its stiff part A_F is given by the degrees of freedom with coordinates satisfying $\theta(x,y) < 1/50$. The mesh used for this problem and the degrees-of-freedom splitting is shown in Figure 2 (left), shown later.

5.1.3. Problem 3: Brussellator model

Problem 3 is the 1D Brussellator PDE model from Chapter IV.I of the book by Hairer and Wanner [30]:

$$\begin{cases} \dot{\mathrm{u}}(t,x) = \alpha\Delta\,\mathrm{u} + \mathrm{u}^2\,v - (b+1)\,\mathrm{u} + a, & x \in D = [0,1], & t \in [0,T], \\ \dot{v}(t,x) = \alpha\Delta v - \mathrm{u}^2\,v + b\,\mathrm{u}, & x \in D = [0,1], & t \in [0,T], \\ \mathrm{u}(t,0) = \mathrm{u}(t,1) = a, \quad v(t,0) = v(t,1) = b, & \mathrm{u}(0,x) = a + \sin(2\pi x), & v(0,x) = b. \end{cases}$$

Here $T = 10$, $a = 1$, $b = 3$, $\alpha = 1/50$, and we use the same unit interval mesh as for Problem 1.

5.1.4. Computational setup

Unfortunately, half-precision is still not widely supported on laptop CPUs, including our own. For this reason, in our experiments all low-precision computations are emulated in software via our custom-built C++/Python precision emulator, libchopping [6] [20]. Number format emulation is extremely expensive and our software relies on vectorization, OpenMP and MPI so as to improve efficiency. Nevertheless, emulated operations are slower than for native formats and we are thus unable to provide actual CPU timings for our algorithms. Consequently, we can only rely on the theoretical estimates of Section 3.4.

We solve the test problems via the finite element method by using continuous piecewise-linear elements on simplices. We employ the open-source finite element software FEniCS [45] for the assembly of the finite element matrices involved, and Python numpy [31], scipy.sparse [62] and libchopping [20] linear algebra kernels for the computations. We use mass-lumping to avoid solving the mass-matrix linear system at every timestep, and we take the linear part of the discretized PDE, A, to be the stiffness matrix scaled on the left by the inverse lumped mass matrix. So as to better squeeze A into the range of the low-precision format (cf. Remark 5.1), we divide A by its max norm $\|A\|_{\max} = \max_{ij} |A_{ij}|$ before rounding it (we multiply back by $\|A\|_{\max}$ in the high-precision format after each matrix-vector product).

Remark 5.1 (Matrix squeezing). When doing computations in reduced precision one must be careful about underflow/overflow, especially when working with formats with a small range such as fp16 (cf. Table 1). We remark that there exist matrix-squeezing algorithms [38] that first rescale and then round a matrix in such a way that the available range is fully exploited. These algorithms typically work by applying a two-sided diagonal scaling to a matrix A so as to obtain a new matrix $\tilde{A} = D_1 A D_2$ (here D_1, D_2 are diagonal matrices) that better fits into the available range. The advantage of working in mixed-precision is that it is possible to compute matrix-vector products in low precision using $\tilde{A}$ and then rescale the result back, e.g. as $(D_1^{-1})\tilde{A}(D_2^{-1}\boldsymbol{b})$, where $\tilde{A}$ is applied in low precision and the remaining (linear-cost) operations are performed so that the result is stored in high precision. Similar techniques are also available for nonlinear terms, see e.g. [40] for an application of these techniques to weather simulation.

5.2. Numerical results

5.2.1. Stability

We start by looking at the numerical stability of our mixed-precision methods. As previously mentioned, establishing any theoretical stability result is extremely complicated since rounding errors disrupt both the smoothness of the solution and the spectrum of A. For instance, whenever $\Delta t A$ has small nonpositive eigenvalues, these can be perturbed by rounding errors and made positive, thus amplifying the error. At the same time, a solution affected by noise due to rounding errors loses its smoothness, which prevents us from obtaining sharp *a priori* error bounds.

We remark that these theoretical issues arise even when computations are performed in high precision. However, we know that when computations are performed in double precision the situation in practice is much different, and numerical methods for ODEs work as they should. We now show that the same happens in practice for our mixed-precision methods, and that the low-precision computations we use do not impact numerical stability (at least for RKC1, RKC2 is partially affected). For this purpose, we take Problem 1 in 2D with $h(u)$, $f_1 \equiv 0$ (i.e. the standard heat equation with no forcing), homogeneous Dirichlet boundary conditions, and $u(0, \boldsymbol{x}) = (16\, xy(1-x)(1-y))^2$, and we investigate how the ratio $\|\hat{\boldsymbol{y}}^n\|_2 / \|\boldsymbol{y}^0\|_2$ evolves as the mesh size is refined for fixed Δt for different values of s across a larger timespan of $T = 8$. We look at order-preserving mixed-precision RKC implementations, and at a simpler not order-preserving version in which all function evaluations are performed in low precision. For mixed-precision RKC1 we use a double/bfloat16 format combination, we fix $\mathcal{D} = 50$, and we take $s = 2^{5+i}$, $N = 2^{2+i}$, $\Delta t = s^2/\rho$, for $i = 0, \dots, 4$. For mixed-precision RKC2 we instead use double/fp16, we fix $\mathcal{D} = 1$, and we take $s \in \{8, 16, 32, 48, 64\}$, $N = s$, and $\Delta t = \frac{1}{2}\beta^2(s, \frac{2}{13})/\rho$ (cf. (9)). Results are shown in Figure 1. Clearly, the standard mixed-precision algorithm (dotted lines) is stable in practice, as stable as the high-precision implementation results (which we are not showing). Our order-preserving mixed-precision RKC1 method (dashed lines in the left figure) is likewise stable. For RKC2 instead (dashed line in the right figure), we note that instability arises for larger values of s. This is a consequence of iterating over $\hat{\boldsymbol{v}}_j$ in (18). As we will see later, this instability threshold diminishes as u grows, and is around $s = 20$

[6] This code was inspired by Higham and Pranesh's work [37] and by Milan Klöwer's emulators in Julia https://github.com/milankl?tab=repositories.

for bfloat16, and around $s = 50$ for fp16 (at least for this problem). As we said in Remark 3.3, for large values of s one should just resort to either single precision (for which the instability threshold is much higher), or to a stable 1-order-preserving version of RKC2.

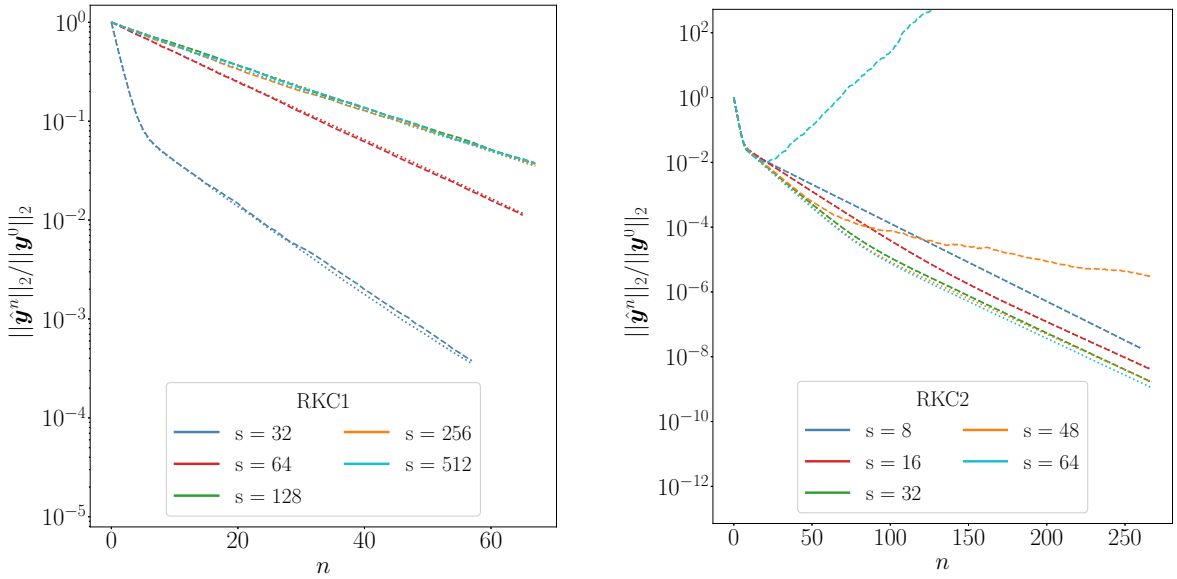

Figure 1: *Behaviour of the* 2*-norm of the numerical solution of the heat equation in mixed precision with RKC vs number of timesteps for different values of s. Dotted lines correspond to results obtained using a non-order-preserving implementation, while dashed lines correspond to our mixed-precision algorithms* (15) *and* (18)*. A decaying trend follows the behaviour of the true solution of the PDE and denotes stability.*

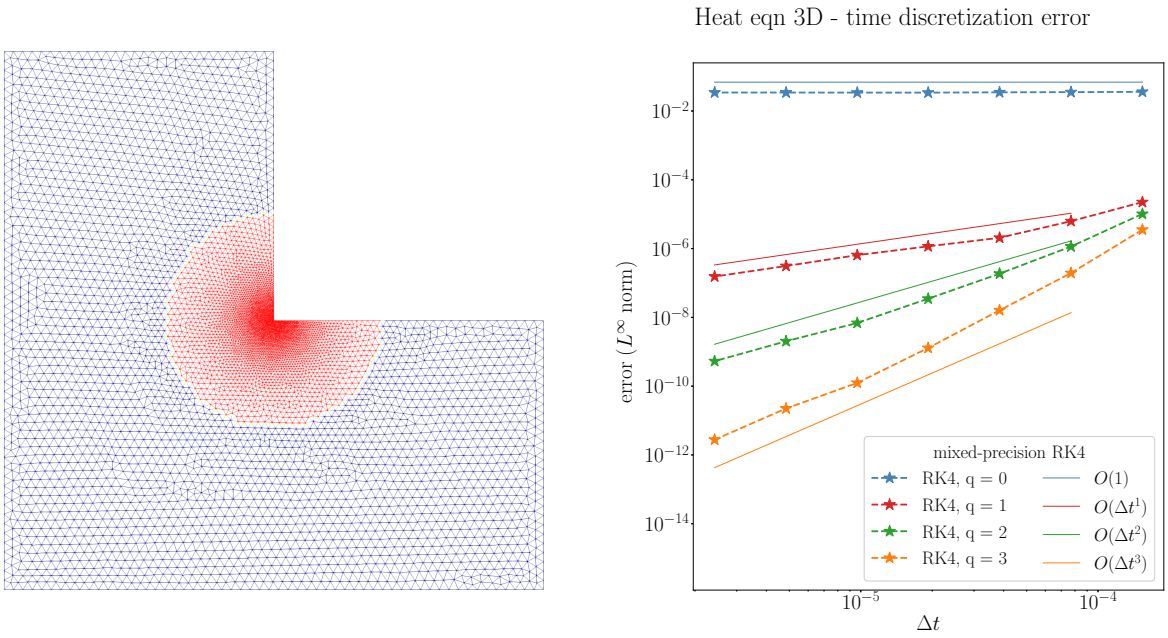

Figure 2: *On the left, the graded mesh used for Problem 2. For this problem, the stiff part of A is given by the entries corresponding to the dofs near the re-entrant corner (colored in red). On the right, the convergence behaviour of the q-order preserving mixed-precision RK4 for the nonlinear heat equation in 3D as q varies.*

5.2.2. Convergence

In order to sanitize our results from spatial discretization errors we compare the numerical solutions $\hat{\mathrm{u}}_h^n \approx \mathrm{u}|_{t=n\Delta t}$ and $\hat{v}_h^n \approx v|_{t=n\Delta t}$ against the much more accurate solutions $\bar{\mathrm{u}}_h^n$ and $\bar{v}_h^n$ obtained by using the same spatial discretization, but in exact arithmetic and with the classic fourth-order method RK4 with a much smaller timestep $\Delta t_{\mathrm{ref}} = \min(2\rho^{-1}, \Delta t/4)$.

We first verify that our methods are indeed order-preserving by estimating what their order of convergence is in practice. For this purpose, we take the maximum L^∞ norm over time, defined as (for Problems 1, 2, and 3 respectively)

$$\max_n \|\hat{\mathrm{u}}_h^n - \bar{\mathrm{u}}_h^n\|_{L^\infty(D)}, \qquad \max_n \|\hat{\mathrm{u}}_h^n - \bar{\mathrm{u}}_h^n\|_{L^\infty(D_L)}, \qquad \max_n \max\left(\|\hat{\mathrm{u}}_h^n - \bar{\mathrm{u}}_h^n\|_{L^\infty(D)},\ \|\hat{v}_h^n - \bar{v}_h^n\|_{L^\infty(D)}\right). \tag{50}$$

We also consider relative errors computed by dividing the quantities in (50) by the roundoff unit u of the low-precision format.

Linear problems. We begin by considering a linear problem and investigating the effect of changing q, the number of high-precision matvecs. For this, we take Problem 1 with $h(\mathrm{u}) \equiv 0$ (i.e. the standard heat equation) in 3D with $N = 2^5$ (i.e. a mesh of 196608 tetrahedra), which we solve using the q-order-preserving RK4 method constructed following (13). We choose $q \in \{0, 1, 2, 3\}$ and we show the results in Figure 2 (right). As we can see, taking q high-precision matrix vector products as in (13) is sufficient to recover q-th order convergence. Furthermore, for $q > 0$ we can distinguish a pre-asymptotic regime in which the mixed-precision method converges with full order (i.e. 4th) before tailing off to the lower rate. The duration of this regime appears to be growing with q. This suggests that even when $q \geq 1$ is much smaller than the full order p it is still be possible to reduce the error significantly. Indeed we observe that even with only $q = 1$ the error is already reduced by up to 4 orders of magnitude with respect to the non-order preserving method.

Remark 5.2. In some scenarios an $O(u)$ limiting accuracy is sufficient. However, the limiting error constant is problem-dependent, and in the worst-case it might be comparable to u^{-1}. In this case we suggest that a simple 1-order-preserving mixed-precision method would be enough to avoid losing all accuracy. We remark that for some problems it is possible to ensure near-$O(u)$ limiting accuracy without ever resorting to higher precision. Techniques such as compensated summation [34, 40] or stochastic rounding [21] can be used for this purpose.

Nonlinear problems. We now solve the nonlinear Problems 1-3 using our methods and estimate their empirical convergence order. We consider the order preserving RKC schemes (15) and (18) with $s = 16$, and the nonlinear terms evaluated according to Scenario 1 (high-precision evalutions of $\boldsymbol{g}$) and Scenario 2 (low-precision evaluations of the Jacobian), and the order-preserving multirate RKC method (44). We investigate the behaviour of the time-discretization error as the timestep is refined by taking bfloat16 and double precision as the low- and high-precision formats respectively. With these methods and formats, we solve Problem 1 in 2D with $N = 2^6$ (Figure 3), Problem 2 with $N = 2^5$ (Figure 4) and Problem 3 with $N = 2^6$ (Figure 5). In Figures 3, 4 and 5 we plot the relative error (the error measures in (50) divided by u) versus Δt. We note how the order-preserving schemes successfully ensure that the full order of the method is preserved even when almost all function evaluations are performed in low precision. On the other hand, we observe that the error of the non order-preserving schemes stagnates at roughly $10u$ (two digits of accuracy) for Problems 1 and 2 and $100u$ (less than one digit of accuracy!) for Problem 3. The order-preserving methods are up to 2-8 orders of magnitude more accurate.

5.2.3. Number of stages vs error

We now investigate the stability of our methods as the number of stages increases. More specifically, we look at how the number of stages s affects the global rounding error in the mixed-precision schemes. We only consider Problem 1 in 2D with $N = 2^4$ and Problem 3 with $N = 2^8$, and we fix $\Delta t\rho = s^2$ for RKC1 and $\Delta t\rho = \frac{1}{2}\beta^2(s, \frac{2}{13})$ for RKC2. We estimate how the global rounding error compares to the time-discretization error of the schemes when run in exact arithmetic. Namely, we look at the ratios:

$$\frac{\max_n \|\hat{\mathrm{u}}_h^n - \mathrm{u}_h^n\|_{L^\infty(D)}}{\max_n \|\mathrm{u}_h^n - \bar{\mathrm{u}}_h^n\|_{L^\infty(D)}}, \qquad \max\left(\frac{\max_n \|\hat{\mathrm{u}}_h^n - \mathrm{u}_h^n\|_{L^\infty(D)}}{\max_n \|\mathrm{u}_h^n - \bar{\mathrm{u}}_h^n\|_{L^\infty(D)}}, \frac{\max_n \|\hat{v}_h^n - v_h^n\|_{L^\infty(D)}}{\max_n \|v_h^n - \bar{v}_h^n\|_{L^\infty(D)}}\right),$$

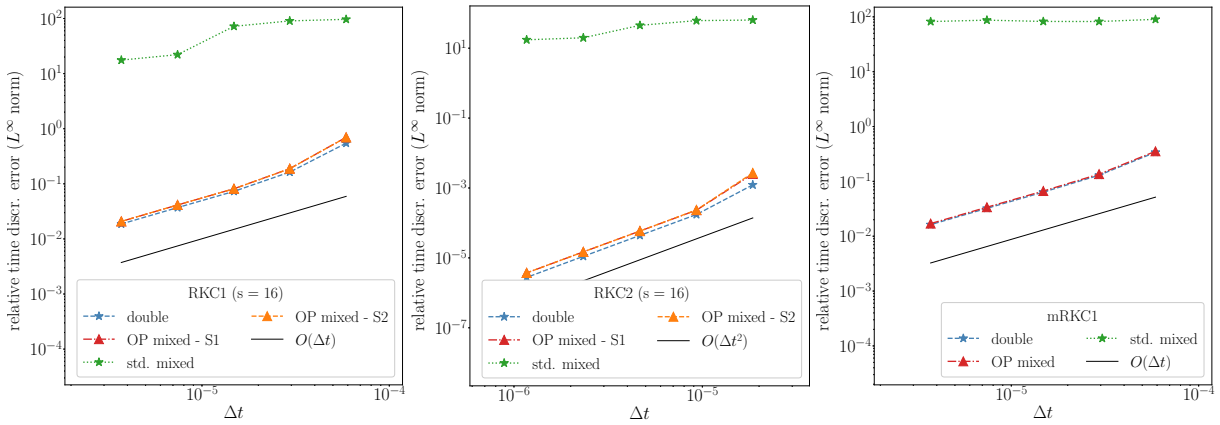

Figure 3: *Mixed-precision RKC: convergence for the nonlinear heat equation in 2D. "OP" stands for order-preserving and S1 and S2 stand for Scenario 1 and 2 respectively, while "std. mixed" indicates a standard non-order preserving mixed-precision implementation.*

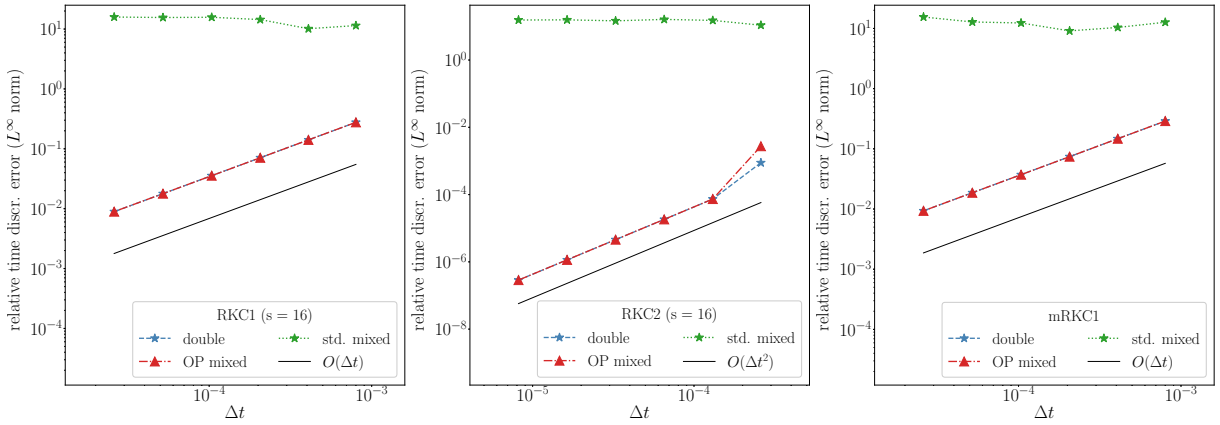

Figure 4: *Mixed-precision RKC: time-discretization error convergence for the heat equation in the L-shaped domain. "OP" stands for order-preserving and "std. mixed" indicates a standard non-order preserving mixed-precision implementation.*

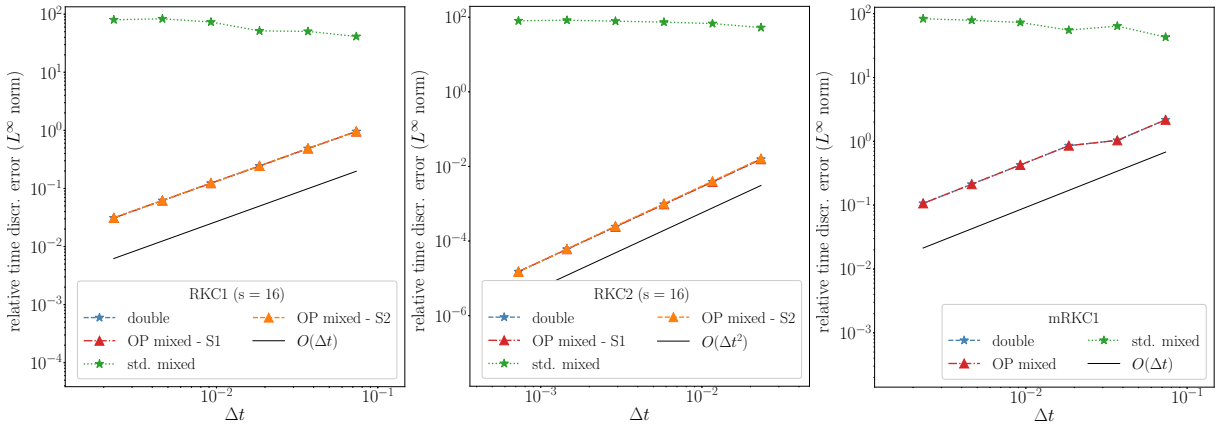

Figure 5: *Mixed-precision RKC: time-discretization error convergence for the Brussellator. "OP" stands for order-preserving and S1 and S2 stand for Scenario 1 and 2 respectively, while "std. mixed" indicates a standard non-order preserving mixed-precision implementation.*

for Problems 1 and 3 respectively. Here u_h^n and v_h^n are obtained by running the same numerical scheme as for $\hat{u}_h^n$ and $\hat{v}_h^n$, only in fully high precision. We consider the mixed-precision (bfloat16/double) RKC1 scheme and the mixed-

precision RKC2 scheme run by taking bfloat16, fp16, and single precision as the low-precision formats, as well as their fully low-precision equivalents. We evaluate the nonlinear terms according to both Scenario 1 (high-precision evalutions of $\boldsymbol{g}$) and Scenario 2 (low-precision evaluations of the Jacobian). The purpose of this test is to assess the magnitude of rounding errors vs discretization errors and to validate the results in Section 3 by estimating in practice the range of values of s for which our mixed-precision schemes are stable.

Results are shown in Figures 6 and 7 (Problem 1), and in Figures 8 and 9 (Problem 3). We observe that while the rounding error of the mixed-precision schemes is of roughly the same order of the time-discretization error (or even smaller), the fully low precision scheme is orders of magnitude larger for small to moderate vaues of s. Nevertheless, the timestep, and consequently the discretisation error, increase with s and eventually the accuracy of the fully low-precision scheme and its mixed-precision counterpart become comparable.

We remark that in these experiments the mixed-precision RKC1 method was stable for all values of s, suggesting that the mixed-precision RKC1 scheme is actually more robust than our theory predicts. On the other hand, as predicted in Section 3, the 2-order-preserving RKC2 scheme becomes unstable for large s. While this is not appreciated when single/double precision is used (in this case s can be much larger than the values used in the figures), this is more of an issue for the half-precision formats (bfloat16, fp16). For the problems considered and the half/double precision combinations with RKC2, instabilities arise for values of s larger than those shown in Figures 6, 7, 8, and 9. This threshold appears to be problem-dependent. The fp16/double combination seems to be the better choice in this case as it is still stable slightly above 50 stages. If an application requires larger values of s we recommend using single precision, an intermediate custom floating-point format, or the 1-order preserving version of RKC2 (obtainable by iterating over the $\boldsymbol{d}_j$ rather than over the $\boldsymbol{v}_j$, cf. Remark 3.3). These results also suggest that for ESRK methods it might be difficult to obtain q-order-preserving mixed-precision methods with $q > 2$ using our strategy, as in this case we expect an even narrower range of "stable" values of s.

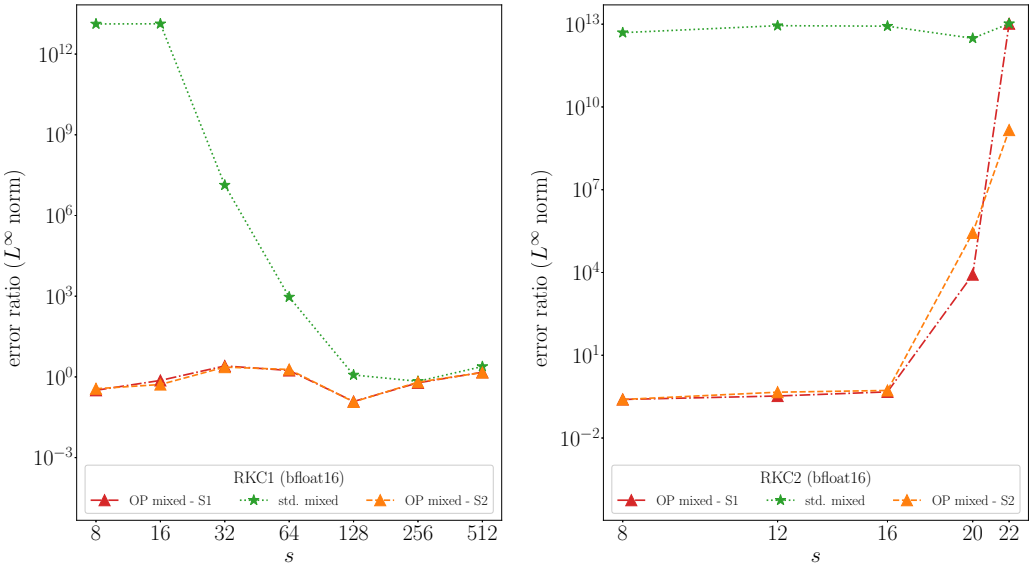

Figure 6: *Mixed-precision RKC: ratio between rounding error and time-discretization error vs number of stages for the nonlinear heat equation in 2D. "OP" stands for order-preserving and S1 and S2 stand for Scenario 1 and 2 respectively, while "std. mixed" indicates a standard non-order preserving mixed-precision implementation.*

5.2.4. Space-time convergence

We conclude the section by testing the convergence in time and space of our mixed-precision methods. We consider the relative error measure

$$u^{-1} \max_n \|\hat{\mathfrak{u}}_h^n - \mathfrak{u}(t^n, \cdot)\|_{L^\infty(D)},$$

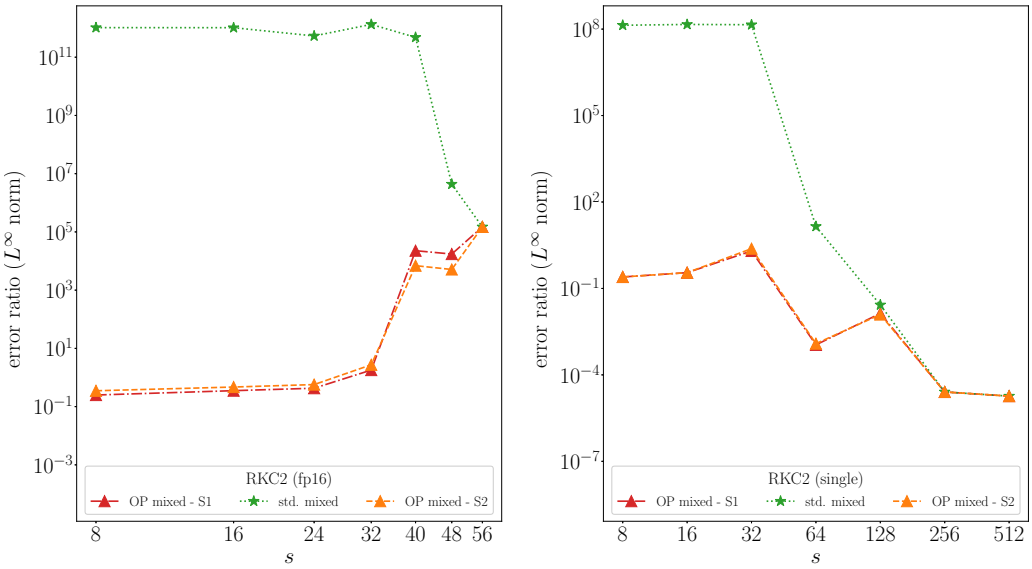

Figure 7: *Mixed-precision RKC: ratio between rounding error and time-discretization error vs number of stages for the nonlinear heat equation in 2D. "OP" stands for order-preserving and S1 and S2 stand for Scenario 1 and 2 respectively, while "std. mixed" indicates a standard non-order preserving mixed-precision implementation.*

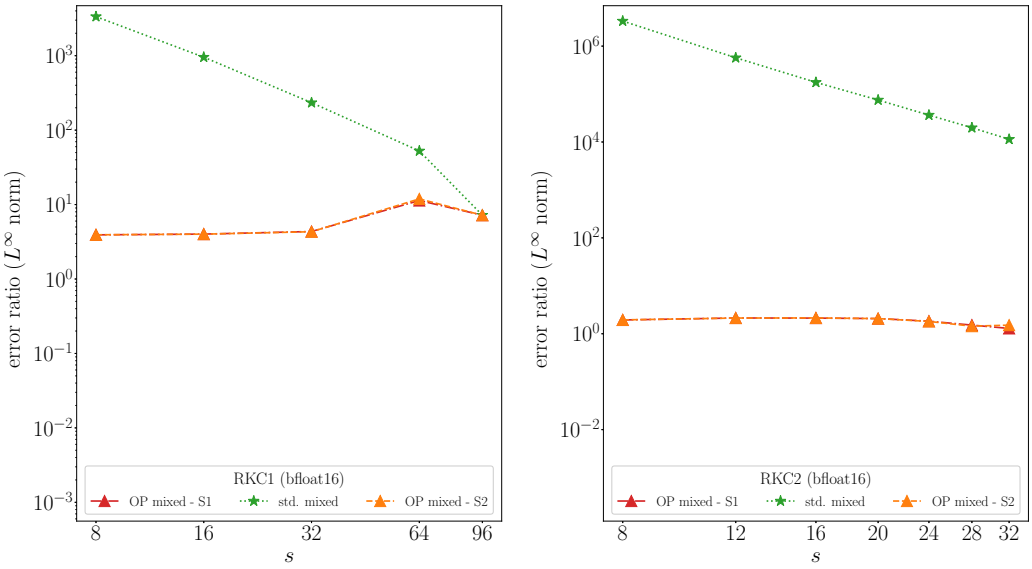

Figure 8: *Mixed-precision RKC: ratio between rounding error and time-discretization error vs number of stages for the Brussellator. "OP" stands for order-preserving and S1 and S2 stand for Scenario 1 and 2 respectively, while "std. mixed" indicates a standard non-order preserving mixed-precision implementation.*

and we only consider Problem 1 as this is the only problem for which an exact solution is available. For RKC1 we fix $s = 16$ and $\Delta t\rho = s^2$, i.e. $\Delta t = O(N^{-2})$, while for RKC2 we vary $N_i = 2^i$, $s_i = \lceil 2\sqrt{N_i} \rceil$, and $\Delta t\rho(N_i) = \frac{1}{2}\beta_2(s_i, \frac{2}{13})$ (cf. (9)) for $i = 2, \ldots, 6$, i.e. $\Delta t = O(N^{-1})$. Results are shown in Figure 10. The convergence behaviour of our order-preserving methods under Strategies 1 and 2 is the same as for the schemes run fully in double precision. On the other hand, the non-order preserving methods stagnate and are unable to reduce the total error below a given limiting threshold.

Remark 5.3. As an example of how advantageous can be using mixed-precision RKC methods, we also solved Prob-

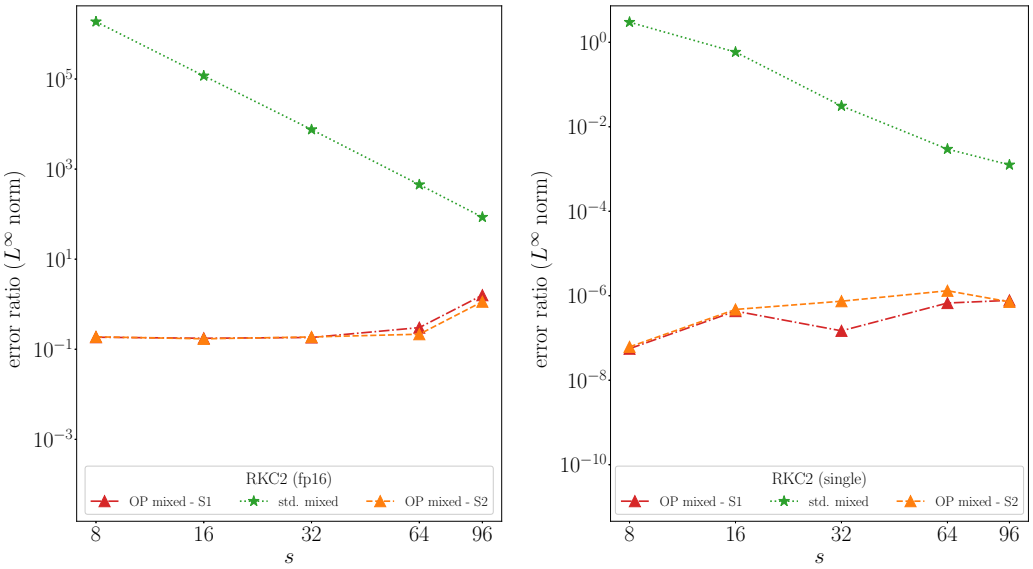

Figure 9: *Mixed-precision RKC: ratio between rounding error and time-discretization error vs number of stages for the Brussellator. "OP" stands for order-preserving and S1 and S2 stand for Scenario 1 and 2 respectively, while "std. mixed" indicates a standard non-order preserving mixed-precision implementation.*

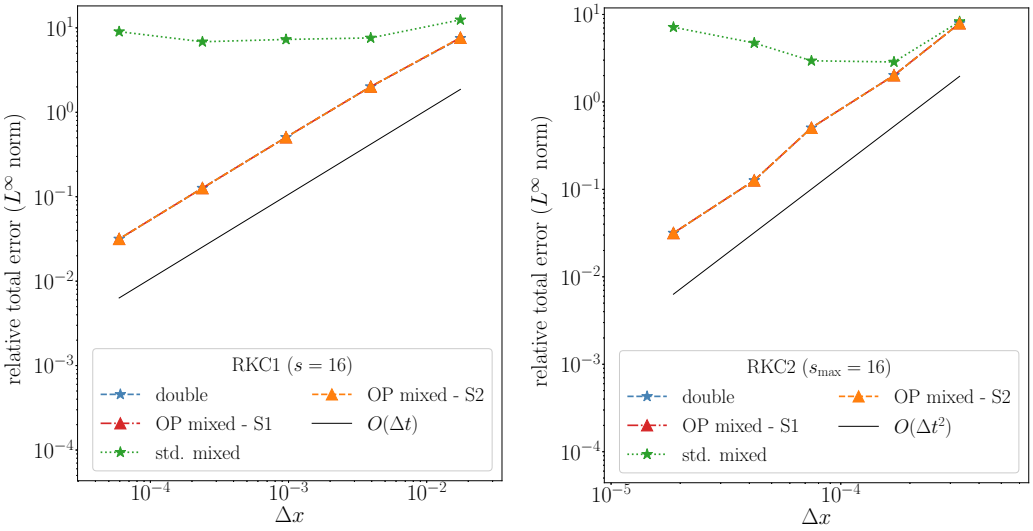

Figure 10: *Mixed-precision RKC: total error vs Δt for the nonlinear heat equation in 2D. "OP" stands for order-preserving and S1 and S2 stand for Scenario 1 and 2 respectively, while "std. mixed" indicates a standard non-order preserving mixed-precision implementation. While we kept $\Delta t = O(N^{-2})$ for RKC1, we instead used $\Delta t = O(N^{-1})$ for RKC2 and increased s accordingly up to $s = 16$. The convergence orders are as predicted by the theory.*

lem 1 in 3D with $N = 2^5$ and $\Delta t\rho = s^2$, $s = 33$, using implicit Euler timestepping using the PETSc software library [13] implementation of Newton's method and the preconditioned conjugate gradient method. As a preconditioner, we used the BoomerAMG algebraic multigrid routines of the Hypre library [25]. Overall, the solution of Problem 3 required on average roughly 6 preconditioned conjugate gradient iterations and 3 Newton iterations per time step. Assuming that the cost of 1 multigrid cycle is twice the cost of one high-precision matvec (see Section 5 in [16]), we obtain that the number of high-precision matvecs required by implicit Euler is roughly 54 per timestep, compared to only one high-precision matvec and 32 half-precision matvecs for a bfloat16/double order-preserving mixed-precision RKC1.

6. Conclusions

We presented some new mixed-precision explicit stabilized schemes for stiff differential equations, considering both multirate and non multirate problems. While the error of naïvely implemented mixed-precision methods stagnates due to rounding errors, the mixed-precision schemes we proposed preserve the full order of convergence of the original high precision methods. This order preservation is achieved by performing only one or two (for first- and second-order methods respectively) high precision evaluations of the right-hand side, while the remaining function evaluations are only needed to preserve stability, and can be performed in low precision. Our order-preserving mixed-precision schemes were constructed by linearizing the original methods and carefully evaluating the Jacobian of the right-hand side in low precision. For this purpose, we proposed different strategies for accurate low-precision Jacobian evaluations.

We showed that the mixed-precision methods preserve the original order of convergence, see Theorems 3.2 and 4.3, and we studied their stability properties in Theorem 3.4. We remark that our worst-case rounding error analysis does not take into account roundoff cancellation effects, which explains why our schemes behave better in practice than in theory. Since rounding errors disrupt all smoothness and spectral properties of the solution we were unable to prove stability in the standard ODE sense. However, extensive numerical experiments show that our RKC1 method remains stable, and that our RKC2 method is also stable when run using single precision as the low precision format. Through our numerical experiments we also confirmed that the mixed-precision schemes preserve the full order of convergence, and that whenever they are stable their error is barely distinguishable from the error of the original high-precision schemes.

Our work naturally extends to other explicit stabilized methods based on orthogonal polynomials, and an extension to strong-stability-preserving RK methods is in preparation. Possible other future extensions to this work include the design of mixed-precision explicit stabilized methods for stiff stochastic differential equations, which are often run on chips supporting low-precision arithmetic. Open questions remain the design of stable order-preserving mixed-precision strategies for high-order stabilized methods, and the development of a stability theory that is able to circumvent the analytical obstacles deriving from rounding errors, namely loss of smoothness and destruction of spectral properties.

Acknowledgements

We would like to thank Giacomo Garegnani for introducing us and making this project possible, and Milan Klöwer for the useful discussion and his help in making our low-precision emulator faster.

A. Technical results for the mixed-precision RKC1 and RKC2 methods

Here we prove Lemma A.1. We indicate with C a generic positive constant that only depends on $\boldsymbol{f}$ and not on $s, u, \Delta t$. The actual value of C might change from line to line.

Lemma A.1. *Let $\hat{\Delta}\boldsymbol{f}_j$ as in (22), then $\hat{\Delta}\boldsymbol{f}_j = \boldsymbol{f}(\hat{\boldsymbol{y}}^n + \hat{\boldsymbol{d}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}^n) + O(\sqrt{u}\Delta t + \Delta t^2)$.*

Proof. We have

$$\hat{\boldsymbol{g}}(\hat{\boldsymbol{y}}^n + \delta\hat{\boldsymbol{d}}_j) - \boldsymbol{g}(\hat{\boldsymbol{y}}^n) = \boldsymbol{g}(\hat{\boldsymbol{y}}^n + \delta\hat{\boldsymbol{d}}_j) - \boldsymbol{g}(\hat{\boldsymbol{y}}^n) + \boldsymbol{r} = \boldsymbol{g}'(\hat{\boldsymbol{y}}^n)\delta\hat{\boldsymbol{d}}_j + \boldsymbol{r} + \boldsymbol{t},$$

where $\boldsymbol{r}$ and $\boldsymbol{t}$ represent rounding and truncation errors, respectively. It holds $\|\boldsymbol{r}\|_2 \le Cu$ and $\|\boldsymbol{t}\|_2 \le C\delta^2\|\hat{\boldsymbol{d}}_j\|_2^2 \le C\delta^2\Delta t^2 \le Cu$, where we used $\|\hat{\boldsymbol{d}}_j\|_2 \le C\Delta t$. Hence

$$\delta^{-1}(\hat{\boldsymbol{g}}(\hat{\boldsymbol{y}}^n + \delta\hat{\boldsymbol{d}}_j) - \boldsymbol{g}(\hat{\boldsymbol{y}}^n)) = \boldsymbol{g}'(\hat{\boldsymbol{y}}^n)\hat{\boldsymbol{d}}_j + \delta^{-1}(\boldsymbol{r} + \boldsymbol{t}) = \boldsymbol{g}'(\hat{\boldsymbol{y}}^n)\hat{\boldsymbol{d}}_j + O(\sqrt{u}\Delta t).$$

Therefore, using $\hat{A}\hat{\boldsymbol{d}}_j = A\hat{\boldsymbol{d}}_j + \Delta A_j\hat{\boldsymbol{d}}_j$, and (22), it holds

$$\hat{\Delta}\boldsymbol{f}_j = A\hat{\boldsymbol{d}}_j + \boldsymbol{g}'(\hat{\boldsymbol{y}}^n)\hat{\boldsymbol{d}}_j + O(\sqrt{u}\Delta t) + \Delta A_j\hat{\boldsymbol{d}}_j = \boldsymbol{f}'(\hat{\boldsymbol{y}}^n)\hat{\boldsymbol{d}}_j + O(\sqrt{u}\Delta t),$$

where we used $\|\Delta A_j\hat{\boldsymbol{d}}_j\|_2 \le \bar{c}\bar{m}^2\|A\|_2 u\|\hat{\boldsymbol{d}}_j\|_2 \le C\bar{c}\bar{m}^2\|A\|_2 u\Delta t$. We conclude by Taylor expanding $\boldsymbol{f}(\hat{\boldsymbol{y}}^n + \hat{\boldsymbol{d}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}^n)$. □

B. Technical results for the mixed-precision mRKC method

Now we prove Lemma B.2. In what follows, we denote with C a generic positive constant depending on $\boldsymbol{f}_F$ and $\boldsymbol{f}_S$, and not on $s, m, \eta, u, \Delta t$. The actual value of C might change from line to line. Before proving Lemma B.2 we first need another auxiliary lemma:

Lemma B.1. *Let $\boldsymbol{y} \in \mathbb{R}^n$, $\hat{\boldsymbol{f}}_\eta$ as in (49), and $\overline{\boldsymbol{f}}_\eta$ as in (43). Then $\hat{\boldsymbol{f}}_\eta(\boldsymbol{y}) = \overline{\boldsymbol{f}}_\eta(\boldsymbol{y}) + O(u + \eta^2)$.*

Proof. In (49) we replace $\hat{\boldsymbol{f}}_F(\boldsymbol{y} + \eta\hat{\boldsymbol{h}}_j) + \hat{\boldsymbol{f}}_S(\boldsymbol{y}) = \boldsymbol{f}_F(\boldsymbol{y} + \eta\hat{\boldsymbol{h}}_j) + \boldsymbol{f}_S(\boldsymbol{y}) + \boldsymbol{r}_j$, with $\|\boldsymbol{r}_j\|_2 \le Cu$. By subtracting (43), we then obtain

$$\boldsymbol{e}_0 = \boldsymbol{0}, \qquad \boldsymbol{e}_1 = \alpha_1 \boldsymbol{r}_0, \qquad \boldsymbol{e}_j = \beta_j \boldsymbol{e}_{j-1} + \gamma_j \boldsymbol{e}_{j-2} + \alpha_j(\boldsymbol{f}_F(\boldsymbol{y} + \eta\hat{\boldsymbol{h}}_j) - \boldsymbol{f}_F(\boldsymbol{y} + \eta\boldsymbol{h}_j)) + \alpha_j \boldsymbol{r}_{j-1}.$$

We then conclude the proof by first using a Taylor expansion of $\boldsymbol{f}_F$, and then invoking Lemma 3.1. □

Lemma B.2. *Assume $\Delta t \le \sqrt{u}$. For $\hat{\Delta}\boldsymbol{f}_{\eta,j}$ as in (48) it holds $\hat{\Delta}\boldsymbol{f}_{\eta,j} = \overline{\boldsymbol{f}}'_\eta(\hat{\boldsymbol{y}}^n)\hat{\boldsymbol{d}}_j + O\left((\sqrt{u} + \epsilon)\Delta t + \Delta t^2\right)$, and thus (47) holds with ϵ replaced by $\sqrt{u} + \epsilon$.*

Proof. Note that the Jacobian of $\overline{\boldsymbol{f}}_\eta$ exists, as it can be obtained by simply differentiating (43). Using (48), the relation $\delta^{-1} = \Delta t/\sqrt{u}$, and Lemmas 4.2 and B.1, we obtain

$$\hat{\Delta}\boldsymbol{f}_{\eta,j} = \delta^{-1}\left(\overline{\boldsymbol{f}}_\eta(\hat{\boldsymbol{y}}^n + \delta\hat{\boldsymbol{d}}_j) - \overline{\boldsymbol{f}}_\eta(\hat{\boldsymbol{y}}^n)\right) + \boldsymbol{r} = \overline{\boldsymbol{f}}'_\eta(\hat{\boldsymbol{y}}^n)\hat{\boldsymbol{d}}_j + \boldsymbol{r} + \boldsymbol{t},$$

where again $\boldsymbol{r}$, $\boldsymbol{t}$ represent rounding and truncation errors, respectively, and satisfy

$$\|\boldsymbol{r}\|_2 \le C\delta^{-1}(u + \epsilon\eta + \eta^2) \le C(\sqrt{u}\Delta t + \epsilon\eta\Delta t/\sqrt{u} + \eta^2\Delta t/\sqrt{u}), \qquad \|\boldsymbol{t}\|_2 \le C\delta^{-1}\|\delta\hat{\boldsymbol{d}}_j\|_2^2 \le C\delta\Delta t^2 \le C\sqrt{u}\Delta t.$$

Using $\eta = O(\Delta t)$ (see [4]), and $\Delta t \le \sqrt{u}$, we obtain $\|\boldsymbol{r}\|_2 + \|\boldsymbol{t}\|_2 = O(\sqrt{u}\Delta t + \epsilon\Delta t + \Delta t^2)$, which concludes the proof. □

References

[1] Abdelfattah, A., Anzt, H., Boman, E.G., Carson, E., Cojean, T., Dongarra, J., Fox, A., Gates, M., Higham, N.J., Li, X.S., et al., 2021. A survey of numerical linear algebra methods utilizing mixed-precision arithmetic. The International Journal of High Performance Computing Applications 35, 344–369.

[2] Abdulle, A., 2002. Fourth order Chebyshev methods with recurrence relation. SIAM Journal on Scientific Computing 23, 2041–2054.

[3] Abdulle, A., Almuslimani, I., Vilmart, G., 2018. Optimal explicit stabilized integrator of weak order one for stiff and ergodic stochastic differential equations. Siam Journal on Uncertainty Quantification 6, 937–964.

[4] Abdulle, A., Grote, M.J., Rosilho de Souza, G., 2020. Explicit stabilized multirate method for stiff differential equations. Technical Report arXiv:2006.00744.

[5] Abdulle, A., Li, T., 2008. S-ROCK methods for stiff Itô SDEs. Communications in Mathematical Sciences 6, 845–868.

[6] Abdulle, A., Medovikov, A.A., 2001. Second order Chebyshev methods based on orthogonal polynomials. Numerische Mathematik 18, 1–18.

[7] Abdulle, A., Rosilho de Souza, G., 2020. Explicit stabilized multirate method for stiff stochastic differential equations. Technical Report arXiv:2010.15193.

[8] Abdulle, A., Vilmart, G., Zygalakis, K.C., 2013. Weak second order explicit stabilized methods for stiff stochastic differential equations. SIAM Journal on Scientific Computing 35, A1792–A1814.

[9] Ackmann, J., Düben, P.D., Palmer, T.N., Smolarkiewicz, P.K., 2021. Mixed-precision for linear solvers in global geophysical flows. arXiv preprint arXiv:2103.16120 .

[10] Agullo, E., Cappello, F., Di, S., Giraud, L., Liang, X., Schenkels, N., 2020. Exploring variable accuracy storage through lossy compression techniques in numerical linear algebra: a first application to flexible GMRES. Ph.D. thesis. Inria Bordeaux Sud-Ouest.

[11] Amestoy, P., Boiteau, O., Buttari, A., Gerest, M., Jézéquel, F., l'Excellent, J.Y., Mary, T., 2021a. Mixed precision low rank approximations and their application to block low rank lu factorization .

[12] Amestoy, P., Buttari, A., Higham, N.J., l'Excellent, J.Y., Mary, T., Vieuble, B., 2021b. Five-precision gmres-based iterative refinement .

[13] Balay, S., Abhyankar, S., Adams, M., Brown, J., Brune, P.R., Buschelman, K., Eijkhout, V., Gropp, W., Kaushik, D., Knepley, M.G., Others, 2017. PETSc users manual revision 3.8. Technical Report. Argonne National Laboratory (ANL).

[14] Björck, Å., Paige, C.C., 1992. Loss and recapture of orthogonality in the modified gram–schmidt algorithm. SIAM journal on matrix analysis and applications 13, 176–190.

[15] Blanchard, P., Higham, N.J., Lopez, F., Mary, T., Pranesh, S., 2020. Mixed precision block fused multiply-add: Error analysis and application to gpu tensor cores. SIAM Journal on Scientific Computing 42, C124–C141.

[16] Briggs, W.L., Henson, V.E., McCormick, S.F., 2000. A multigrid tutorial. SIAM.

[17] Burnett, B., Gottlieb, S., Grant, Z.J., Heryudono, A., 2021. Performance Evaluation of Mixed-Precision Runge-Kutta Methods arXiv:2107.03357.
[18] Carson, E., Higham, N.J., 2017. A new analysis of iterative refinement and its application to accurate solution of ill-conditioned sparse linear systems. SIAM Journal on Scientific Computing 39, A2834–A2856.
[19] Carson, E., Higham, N.J., 2018. Accelerating the solution of linear systems by iterative refinement in three precisions. SIAM Journal on Scientific Computing 40, A817–A847.
[20] Croci, M., 2019. Libchopping - a parallel low-precision emulator in C++ and Python. URL: https://bitbucket.org/croci/libchopping/.
[21] Croci, M., Giles, M.B., 2020. Effects of round-to-nearest and stochastic rounding in the numerical solution of the heat equation in low precision. Technical Report. URL: http://arxiv.org/abs/2010.16225, arXiv:2010.16225.
[22] Das, D., Mellempudi, N., Mudigere, D., Kalamkar, D., Avancha, S., Banerjee, K., Sridharan, S., Vaidyanathan, K., Kaul, B., Georganas, E., et al., 2018. Mixed precision training of convolutional neural networks using integer operations. arXiv preprint arXiv:1802.00930 .
[23] Dembo, R.S., Eisenstat, S.C., Steihaug, T., 1982. Inexact newton methods. SIAM Journal on Numerical analysis 19, 400–408.
[24] Dumont, T., Duarte, M., Descombes, S., Dronne, M.A., Massot, M., Louvet, V., 2013. Simulation of human ischemic stroke in realistic 3D geometry. Communications in Nonlinear Science and Numerical Simulation 18, 1539–1557.
[25] Falgout, R.D., Yang, U.M., 2002. *Hypre*: A library of high performance preconditioners, in: International Conference on Computational Science, Springer. pp. 632–641.
[26] Grant, Z.J., 2020. Perturbed Runge-Kutta methods for mixed precision applications arXiv:2012.13055.
[27] Gratton, S., Simon, E., Titley-Peloquin, D., Toint, P., 2019. Exploiting variable precision in gmres. arXiv preprint arXiv:1907.10550 .
[28] Griewank, A., Walther, A., 2008. Evaluating derivatives: principles and techniques of algorithmic differentiation. SIAM.
[29] Hairer, E., Nörsett, S.P., Wanner, G., 2008. Solving ordinary differential equations I. volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin.
[30] Hairer, E., Wanner, G., 1996. Solving Ordinary Differential Equations II. volume 14. 2 ed., Springer-Verlag Berlin Heidelberg. doi:10.1007/978-3-642-05221-7.
[31] Harris, C.R., Millman, K.J., et al., 2020. Array programming with NumPy. Nature 585, 357–362. doi:10.1038/s41586-020-2649-2.
[32] Henrici, P., 1962. Discrete Variable Methods in Ordinary Differential Equations. 1 ed., John Wiley & Sons, Inc.
[33] Henrici, P., 1963. Error Propagation in Difference Methods. 1 ed., John Wiley & Sons, Inc.
[34] Higham, N.J., 1993. The accuracy of floating point summation. SIAM Journal on Scientific Computing 14, 783–799.
[35] Higham, N.J., 2002. Accuracy and Stability of Numerical Algorithms. SIAM.
[36] Higham, N.J., Mary, T., 2019. A new approach to probabilistic rounding error analysis. SIAM Journal of Scientific Computing 41, 2815–2835.
[37] Higham, N.J., Pranesh, S., 2019. Simulating low precision floating-point arithmetic. Siam Journal on Scientific Computing 41, C585–C602. URL: https://doi.org/10.1137/19M1251308.
[38] Higham, N.J., Pranesh, S., Zounon, M., 2019. Squeezing a matrix into half precision, with an application to solving linear systems. SIAM Journal on Scientific Computing 41, A2536–A2551. doi:10.1137/18M1229511.
[39] Klöwer, M., Düben, P., Palmer, T., 2020. Number formats, error mitigation, and scope for 16-bit arithmetics in weather and climate modeling analyzed with a shallow water model. Journal of Advances in Modeling Earth Systems 12, e2020MS002246.
[40] Klöwer, M., Hatfield, S., Croci, M., Düben, P.D., Palmer, T.N., 2021. Fluid simulations accelerated with 16 bit: Approaching 4x speedup on A64FX by squeezing ShallowWaters.jl into Float16. Technical Report. University of Oxford.
[41] Lange, M., Rump, S.M., 2017. Error estimates for the summation of real numbers with application to floating-point summation. BIT Numerical Mathematics 57, 927–941. doi:10.1007/s10543-017-0658-9.
[42] Lebedev, V.I., 1994. How to solve stiff systems of differential equations by explicit methods, in: Numer. methods Appl.. CRC, Boca Raton, FL, pp. 45–80.
[43] Lebedev, V.I., Medovikov, A.A., 1994. Explicit methods of second order for the solution of stiff systems of ODEs. Russ. Acad. Sci. .
[44] Lindberg, B., 1972. IMPEX: a program package for solution of systems of stiff differential equations. Technical Report. Dept. of Information Processing, Royal Inst. of Tech.. Stockholm.
[45] Logg, A., Mardal, K.A., Wells, G., 2012. Automated Solution of Differential Equations by the Finite Element Method: the FEniCS Book. volume 84. Springer Science & Business Media.
[46] Lopez, F., Mary, T., 2020. Mixed Precision LU Factorization on GPU Tensor Cores: Reducing Data Movement and Memory Footprint. Technical Report. The University of Manchester.
[47] McCormick, S.F., Benzaken, J., Tamstorf, R., 2021. Algebraic error analysis for mixed-precision multigrid solvers. SIAM Journal on Scientific Computing , S392–S419.
[48] Medovikov, A.A., 1998. High order explicit methods for parabolic equations. BIT Numerical Mathematics 38, 372–390.
[49] Mellempudi, N., Srinivasan, S., Das, D., Kaul, B., 2019. Mixed precision training with 8-bit floating point. arXiv preprint arXiv:1905.12334 .
[50] Meurant, G., Strakoš, Z., 2006. The lanczos and conjugate gradient algorithms in finite precision arithmetic. Acta Numerica 15, 471–542.
[51] Meyer, C.D., Balsara, D.S., Aslam, T.D., 2014. A stabilized Runge-Kutta-Legendre method for explicit super-time-stepping of parabolic and mixed equations. Journal of Computational Physics 257, 594–626. doi:10.1016/j.jcp.2013.08.021.
[52] Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., et al., 2017. Mixed precision training. arXiv preprint arXiv:1710.03740 .
[53] Paxton, E.A., Chantry, M., Klöwer, M., Saffin, L., Palmer, T., 2021. Climate modelling in low-precision: Effects of both deterministic & stochastic rounding. arXiv preprint arXiv:2104.15076 .
[54] Sommeijer, B.P., Shampine, L., Verwer, J.G., 1998. RKC: An explicit solver for parabolic PDEs. Journal of Computational and Applied Mathematics 88, 315–326.
[55] Tamstorf, R., Benzaken, J., McCormick, S.F., 2021. Discretization-error-accurate mixed-precision multigrid solvers. SIAM Journal on Scientific Computing , S420–S447.

[56] Tisseur, F., 2001. Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems. SIAM Journal on Matrix Analysis and Applications 22, 1038–1057.
[57] Van der Houwen, P.J., Sommeijer, B.P., 1980. On the internal stability of explicit, m-stage Runge–Kutta methods for large m-values. Zeitschrift für Angewandte Mathematik und Mechanik 60, 479–485.
[58] Verwer, J.G., 1980. An implementation of a class of stabilized explicit methods for the time integration of parabolic equations. ACM Transactions on Mathematical Software (TOMS) 6, 188–205.
[59] Verwer, J.G., 1996. Explicit Runge–Kutta methods for parabolic partial differential equations. Appl. Numer. Math. 22, 359–379.
[60] Verwer, J.G., Hundsdorfer, W.H., Sommeijer, B.P., 1990. Convergence properties of the Runge-Kutta-Chebyshev method. Numerische Mathematik 57, 157–178.
[61] Verwer, J.G., Sommeijer, B.P., 2004. An implicit-explicit RungeKuttaChebyshev scheme for diffusion-reaction equations. SIAM Journal on Scientific Computing 25, 1824–1835.
[62] Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., et al., 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods 17, 261–272. doi:10.1038/s41592-019-0686-2.
[63] Yamazaki, I., Tomov, S., Dongarra, J., 2015. Mixed-precision cholesky qr factorization and its case studies on multicore cpu with multiple gpus. SIAM Journal on Scientific Computing 37, C307–C330.
[64] Yang, L.M., Fox, A., Sanders, G., 2021. Rounding error analysis of mixed precision block householder qr algorithms. SIAM Journal on Scientific Computing 43, A1723–A1753.