Mixed-precision explicit Runge-Kutta methods

MATTEO CROCI

Collaborator: Giacomo Rosilho De Souza (Euler Institute, USI Lugano)

21st IMA Leslie Fox Prize Event, University of Strathclyde, 26 June 2023

Main reference: M. Croci and G. Rosilho de Souza. Mixed-precision explicit stabilized Runge–Kutta methods for single-and multi-scale differential equations. *Journal of Computational Physics*, 464:111349, 2022.

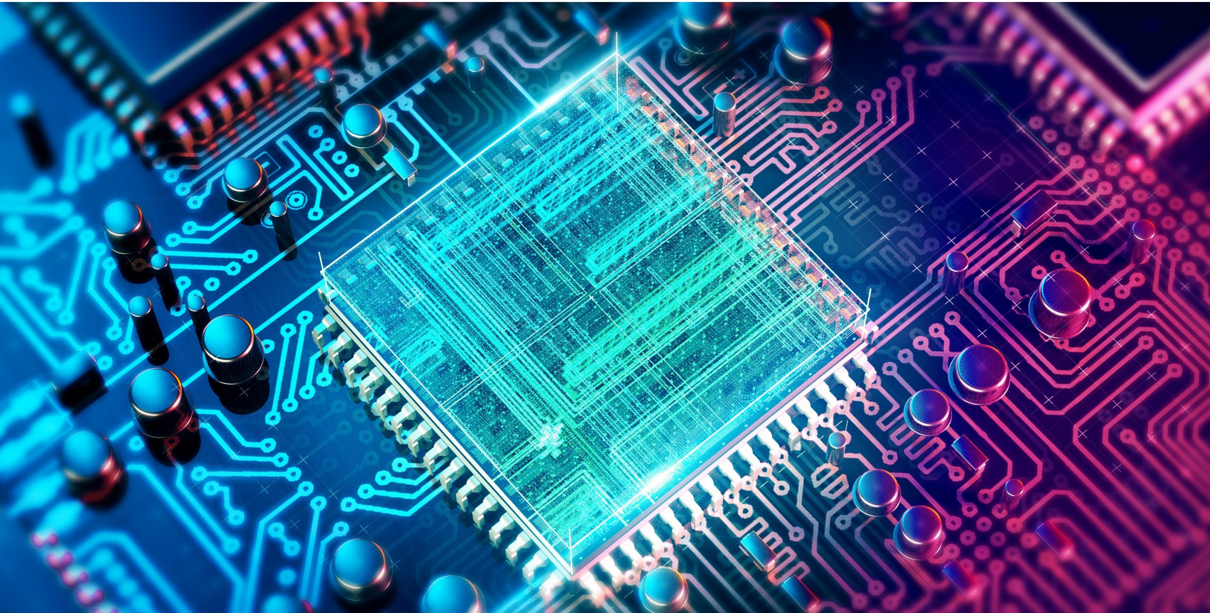
The University of Texas at Austin
Oden Institute for Computational Engineering and Sciences

Overview

1. Introduction

Mixed-precision algorithms

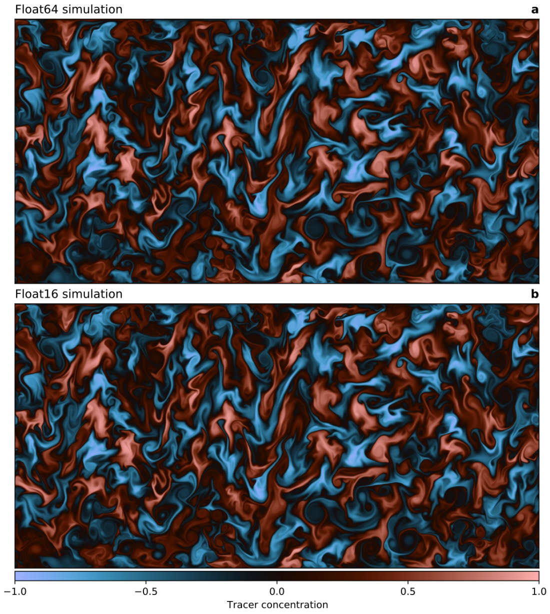
Mixed-precision algorithms combine low- and high-precision computations in order to benefit from the performance gains of reduced precision while retaining good accuracy.



Example application: Weather and climate forecasting [Klower et al. 2022]

Shallow-water eqs for 2D oceanic flow:

$$\begin{cases} \dot{\boldsymbol{v}} + \boldsymbol{v} \cdot \nabla \boldsymbol{v} + \hat{\boldsymbol{z}} \times \boldsymbol{v} = -\nabla \eta + \Delta^2 \boldsymbol{v} - \boldsymbol{v} + \boldsymbol{F}, \\ \dot{\eta} + \nabla \cdot (\boldsymbol{v} h) = 0, \\ \dot{q} + \boldsymbol{v} \cdot \nabla q = -\tau (q - q_0). \end{cases}$$



Common floating-point formats

Format	unit roundoff u	range
bfloat16 (half)	2^{-8} ($\approx$ 2.5 digits)	$\approx 10^{\pm 38}$
fp16 (half)	2^{-11} ($\approx$ 3.5 digits)	$\approx 10^{\pm 4.5}$
fp32 (single)	2^{-24} ($\approx$ 7 digits)	$\approx 10^{\pm 38}$
fp64 (double)	2^{-53} ($\approx$ 15 digits)	$\approx 10^{\pm 308}$

Better performance: 16-bits computations are 4x faster than double precision on CPUs and 16x faster on GPUs. Energy-/memory-efficiency gains are also comparable.

All major chip manufacturers (e.g., AMD, ARM, NVIDIA, Intel, ...) have commercialized chips (CPUs, GPUs, TPUs, FPGAs, ...) supporting half-precision computations.

Note: today we employ double/bfloat16 via software emulation (no timings available).

Today's focus: Mixed-precision explicit Runge–Kutta methods

Our work: design mixed-precision explicit Runge–Kutta schemes for solving:

$$\boldsymbol{y}'(t) = \boldsymbol{f}(\boldsymbol{y}(t)), \quad \boldsymbol{y}(0) = \boldsymbol{y}_0,$$

Objective

Evaluate $\boldsymbol{f}$ in low-precision as much as possible without affecting accuracy or stability.

2. Mixed-precision RK methods for linear problems

Linear problems

We start by considering linear problems in the form:

$$\boldsymbol{y}'(t) = A\boldsymbol{y}(t), \quad \boldsymbol{y}(0) = \boldsymbol{y}_0.$$

Linear problems

We start by considering linear problems in the form:

$$\boldsymbol{y}'(t) = A\boldsymbol{y}(t), \quad \boldsymbol{y}(0) = \boldsymbol{y}_0.$$

Required for the analysis: A rounding error bound for matrix-vector products,

Theorem (Lemma 6.6 in [Higham 2002])

Let $A \in \mathbb{R}^{m\times m}$. Barring underflow/overflow, there exists $c > 0$ depending on m s.t.

$$\widehat{A\boldsymbol{x}} = (A + \Delta A)\boldsymbol{x}, \quad \textit{with} \quad \|\Delta A\|_2 \leq cu\|A\|_2 = O(u).$$

Linear problems - local error

Consider the exact solution at $t = \Delta t$ and its corresponding s-stage, p-th order RK approximation:

$$\begin{aligned} \boldsymbol{y}(\Delta t) &= \exp(\Delta t A)\boldsymbol{y}_0 = \sum_{j=0}^{\infty} \frac{(\Delta t A)^j}{j!}\boldsymbol{y}_0, \\ \boldsymbol{y}_1 &= R_s(A)\boldsymbol{y}_0 = \sum_{j=0}^{p} \frac{(\Delta t A)^j}{j!}\boldsymbol{y}_0 + O(\Delta t^{p+1}). \end{aligned}$$

Giving a local error $\tau = \Delta t^{-1}\|\boldsymbol{y}(\Delta t) - \boldsymbol{y}_1\|_2 = O(\Delta t^p)$.

Linear problems - local error

Consider the exact solution at $t = \Delta t$ and its corresponding s-stage, p-th order RK approximation:

$$\boldsymbol{y}(\Delta t) = \exp(\Delta t A)\boldsymbol{y}_0 = \sum_{j=0}^{\infty} \frac{(\Delta t A)^j}{j!}\boldsymbol{y}_0,$$

$$\boldsymbol{y}_1 = R_s(A)\boldsymbol{y}_0 = \sum_{j=0}^{p} \frac{(\Delta t A)^j}{j!}\boldsymbol{y}_0 + O(\Delta t^{p+1}).$$

Giving a local error $\tau = \Delta t^{-1}\|\boldsymbol{y}(\Delta t) - \boldsymbol{y}_1\|_2 = O(\Delta t^p)$.

Evaluating the scheme in finite precision yields:

$$\hat{\boldsymbol{y}}_1 = \widehat{R_s(A)\boldsymbol{y}_0} = \varepsilon + \boldsymbol{y}_0 + \sum_{j=1}^{p} \frac{\Delta t^j}{j!}\left(\prod_{k=1}^{j}(A + \Delta A_k)\right)\boldsymbol{y}_0 + O(\Delta t^{p+1}).$$

Local error and order preservation

$$\tau = \Delta t^{-1} \|\hat{\boldsymbol{y}}_1 - \boldsymbol{y}(\Delta t)\|_2 = \Delta t^{-1} \left\| \varepsilon + \sum_{j=1}^{p} \frac{\Delta t^j}{j!} \left(\prod_{k=1}^{j} (A + \Delta A_k) - A^j \right) \boldsymbol{y}_0 \right\|_2 + O(\Delta t^p).$$

Local error and order preservation

$$\tau = \Delta t^{-1} \|\hat{\boldsymbol{y}}_1 - \boldsymbol{y}(\Delta t)\|_2 = \Delta t^{-1} \left\| \varepsilon + \sum_{j=1}^{p} \frac{\Delta t^j}{j!} \left(\prod_{k=1}^{j} (A + \Delta A_k) - A^j \right) \boldsymbol{y}_0 \right\|_2 + O(\Delta t^p).$$

Assumption

Operations performed in high-precision are exact.

Local error and order preservation

$$\tau = \Delta t^{-1}\|\hat{\boldsymbol{y}}_1 - \boldsymbol{y}(\Delta t)\|_2 = \Delta t^{-1}\left\|\varepsilon + \sum_{j=1}^{p}\frac{\Delta t^j}{j!}\left(\prod_{k=1}^{j}(A+\Delta A_k) - A^j\right)\boldsymbol{y}_0\right\|_2 + O(\Delta t^p).$$

Assumption

Operations performed in high-precision are exact.

Let us consider the following scenarios (take u to be the low-precision unit roundoff):

1. We have $\varepsilon = O(u)$ and we get $\tau = O(u\Delta t^{-1} + \Delta t^p)$. Rapid error growth!

Local error and order preservation

$$\tau = \Delta t^{-1}\|\hat{\boldsymbol{y}}_1 - \boldsymbol{y}(\Delta t)\|_2 = \Delta t^{-1}\left\|\varepsilon + \sum_{j=1}^{p}\frac{\Delta t^j}{j!}\left(\prod_{k=1}^{j}(A+\Delta A_k) - A^j\right)\boldsymbol{y}_0\right\|_2 + O(\Delta t^p).$$

Assumption

Operations performed in high-precision are exact.

Let us consider the following scenarios (take u to be the low-precision unit roundoff):

1. We have $\varepsilon = O(u)$ and we get $\tau = O(u\Delta t^{-1} + \Delta t^p)$. Rapid error growth!
2. High-precision vector operations: $\varepsilon = 0$ so $\tau = O(u + \Delta t^p)$. $O(u)$ limiting accuracy and loss of convergence.

Local error and order preservation

$$\tau = \Delta t^{-1} \|\hat{\boldsymbol{y}}_1 - \boldsymbol{y}(\Delta t)\|_2 = \Delta t^{-1} \left\| \varepsilon + \sum_{j=1}^{p} \frac{\Delta t^j}{j!} \left(\prod_{k=1}^{j} (A + \Delta A_k) - A^j \right) \boldsymbol{y}_0 \right\|_2 + O(\Delta t^p).$$

Assumption

Operations performed in high-precision are exact.

Let us consider the following scenarios (take u to be the low-precision unit roundoff):

1. We have $\varepsilon = O(u)$ and we get $\tau = O(u\Delta t^{-1} + \Delta t^p)$. Rapid error growth!
2. High-precision vector operations: $\varepsilon = 0$ so $\tau = O(u + \Delta t^p)$. $O(u)$ limiting accuracy and loss of convergence.
3. First $q \geq 1$ matvecs in high precision. Now $\varepsilon = 0$ and $\Delta A_k = 0$ for $k = 1, \ldots, q$, so $\tau = O(u\Delta t^q + \Delta t^p)$. Recover q-th order convergence!

Local error and order preservation

$$\tau = \Delta t^{-1}\|\hat{\boldsymbol{y}}_1 - \boldsymbol{y}(\Delta t)\|_2 = \Delta t^{-1}\left\|\varepsilon + \sum_{j=1}^{p}\frac{\Delta t^j}{j!}\left(\prod_{k=1}^{j}(A+\Delta A_k) - A^j\right)\boldsymbol{y}_0\right\|_2 + O(\Delta t^p).$$

Assumption

Operations performed in high-precision are exact.

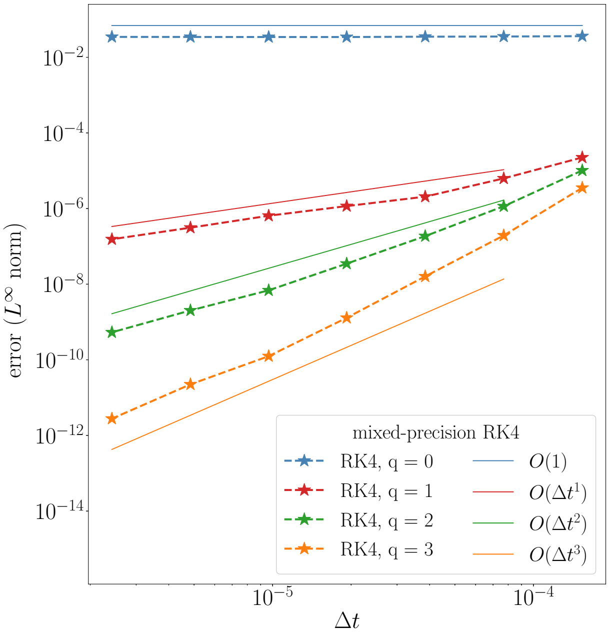
Let us consider the following scenarios (take u to be the low-precision unit roundoff):

1. We have $\varepsilon = O(u)$ and we get $\tau = O(u\Delta t^{-1} + \Delta t^p)$. Rapid error growth!
2. High-precision vector operations: $\varepsilon = 0$ so $\tau = O(u + \Delta t^p)$. $O(u)$ limiting accuracy and loss of convergence.
3. First $q \geq 1$ matvecs in high precision. Now $\varepsilon = 0$ and $\Delta A_k = 0$ for $k = 1, \ldots, q$, so $\tau = O(u\Delta t^q + \Delta t^p)$. Recover q-th order convergence!

Definition: a mixed-precision RK method is ***q*-order-preserving** if it converges with order $q \in \{1, \ldots, p\}$ under the above assumption.

Result: Can construct q-order-preserving schemes for any q for linear problems.

Numerical results - 3D heat equation



3. Mixed-precision RK methods for nonlinear problems

3.1 Overview

Nonlinear problems

We want to design order-preserving mixed-precision explicit RK schemes for

$$\boldsymbol{y}'(t) = \boldsymbol{f}(\boldsymbol{y}(t)), \quad \boldsymbol{y}(0) = \boldsymbol{y}_0.$$

Nonlinear problems

We want to design order-preserving mixed-precision explicit RK schemes for

$$\boldsymbol{y}'(t) = \boldsymbol{f}(\boldsymbol{y}(t)), \quad \boldsymbol{y}(0) = \boldsymbol{y}_0.$$

We start as in the linear case by comparing with the exact solution (see [Butcher 2003]):

$$\boldsymbol{y}(\Delta t) = \boldsymbol{y}_0 + \Delta t \boldsymbol{f}(\boldsymbol{y}_0) + \frac{1}{2}\Delta t^2 \boldsymbol{f}'(\boldsymbol{y}_0)\boldsymbol{f}(\boldsymbol{y}_0) + O(\Delta t^3),$$

Nonlinear problems

We want to design order-preserving mixed-precision explicit RK schemes for

$$\boldsymbol{y}'(t) = \boldsymbol{f}(\boldsymbol{y}(t)), \quad \boldsymbol{y}(0) = \boldsymbol{y}_0.$$

We start as in the linear case by comparing with the exact solution (see [Butcher 2003]):

$$\boldsymbol{y}(\Delta t) = \boldsymbol{y}_0 + \Delta t \boldsymbol{f}(\boldsymbol{y}_0) + \frac{1}{2}\Delta t^2 \boldsymbol{f}'(\boldsymbol{y}_0)\boldsymbol{f}(\boldsymbol{y}_0) + O(\Delta t^3),$$

Our order-preserving schemes must satisfy (e.g., for $q \in \{1, 2\}$),

$$\begin{aligned} \hat{\boldsymbol{y}}_{n+1} &= \hat{\boldsymbol{y}}_n + \Delta t \boldsymbol{f}(\hat{\boldsymbol{y}}_n) + O((1+u)\Delta t^2), && q = 1, \\ \hat{\boldsymbol{y}}_{n+1} &= \hat{\boldsymbol{y}}_n + \Delta t \boldsymbol{f}(\hat{\boldsymbol{y}}_n) + \frac{1}{2}\Delta t^2 \boldsymbol{f}'(\hat{\boldsymbol{y}}_n)\boldsymbol{f}(\hat{\boldsymbol{y}}_n) + O((1+u)\Delta t^3), && q = 2. \end{aligned}$$

Nonlinear problems

We want to design order-preserving mixed-precision explicit RK schemes for

$$\boldsymbol{y}'(t) = \boldsymbol{f}(\boldsymbol{y}(t)), \quad \boldsymbol{y}(0) = \boldsymbol{y}_0.$$

We start as in the linear case by comparing with the exact solution (see [Butcher 2003]):

$$\boldsymbol{y}(\Delta t) = \boldsymbol{y}_0 + \Delta t \boldsymbol{f}(\boldsymbol{y}_0) + \frac{1}{2}\Delta t^2 \boldsymbol{f}'(\boldsymbol{y}_0)\boldsymbol{f}(\boldsymbol{y}_0) + O(\Delta t^3),$$

Our order-preserving schemes must satisfy (e.g., for $q \in \{1, 2\}$),

$$\begin{aligned} \hat{\boldsymbol{y}}_{n+1} &= \hat{\boldsymbol{y}}_n + \Delta t \boldsymbol{f}(\hat{\boldsymbol{y}}_n) + O((1+u)\Delta t^2), && q = 1, \\ \hat{\boldsymbol{y}}_{n+1} &= \hat{\boldsymbol{y}}_n + \Delta t \boldsymbol{f}(\hat{\boldsymbol{y}}_n) + \frac{1}{2}\Delta t^2 \boldsymbol{f}'(\hat{\boldsymbol{y}}_n)\boldsymbol{f}(\hat{\boldsymbol{y}}_n) + O((1+u)\Delta t^3), && q = 2. \end{aligned}$$

Tentative idea: Use high-precision to ensure that local errors are of the right order.

Overview

Preserving order conditions is challenging:

- **Lack of smoothness.** Rounding errors introduce non-smooth noise which affects order conditions and Taylor expansions.
- **Efficiency requirements.** The performance gain of reduced-precision computations must not be outweighed by the cost of matching order conditions.

Overview

Preserving order conditions is challenging:

- **Lack of smoothness.** Rounding errors introduce non-smooth noise which affects order conditions and Taylor expansions.
- **Efficiency requirements.** The performance gain of reduced-precision computations must not be outweighed by the cost of matching order conditions.

Our work:

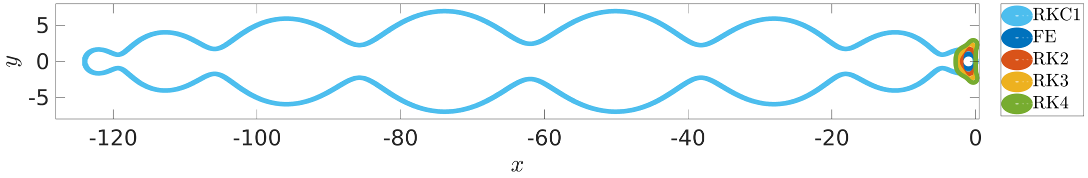
- For nonlinear problems, we can construct efficient q-order-preserving mixed-precision versions of any explicit RK method for $q = 1, 2$.
- In our paper, we applied our technique where it can be most useful: explicit stablised RK schemes for which $s \gg p$.

3.2 Mixed-precision Runge-Kutta-Chebyshev methods

Runge–Kutta–Chebyshev methods[1]

Function evaluations (i.e., # stages) are traditionally used in RK methods to maximise accuracy. The idea of explicit stabilised RK methods is to maximise stability instead.

Runge-Kutta-Chebyshev (RKC) methods are designed for parabolic problems and take $R_s(x)$ to be a Chebyshev polynomial $\rightsquigarrow$ low order ($p \leq 4$), but $O(s^2)$ stability region.



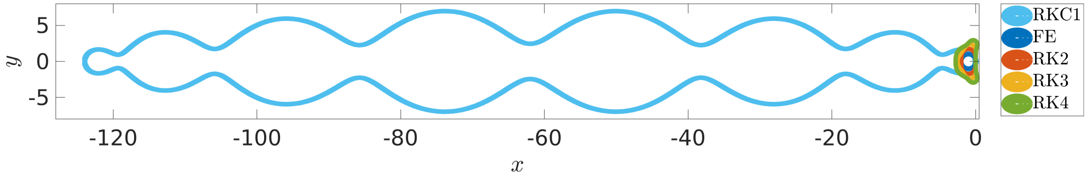
Absolute stability region of RKC1 with $s = 8$ vs those of other explicit methods.

[1]Refs: [van der Houwen and Sommeijer 1980], many papers by Abdulle and collaborators.

Runge–Kutta–Chebyshev methods[1]

Function evaluations (i.e., # stages) are traditionally used in RK methods to maximise accuracy. The idea of explicit stabilised RK methods is to maximise stability instead.

Runge-Kutta-Chebyshev (RKC) methods are designed for parabolic problems and take $R_s(x)$ to be a Chebyshev polynomial $\rightsquigarrow$ low order ($p \leq 4$), but $O(s^2)$ stability region.



Absolute stability region of RKC1 with $s = 8$ vs those of other explicit methods.

Opportunity

Since $s \gg p$, can do most stages in reduced precision!

[1]Refs: [van der Houwen and Sommeijer 1980], many papers by Abdulle and collaborators.

Mixed-precision RKC methods

One step of an s-stage RKC scheme in **exact arithmetic** is given by:

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

Note: $\|\boldsymbol{d}_j\|_2 = O(\Delta t)$ as $\Delta t \to 0$ for all j.

For a q-order preserving method we need to make sure all rounding errors are $O(\Delta t^{q+1})$.

Mixed-precision RKC methods

One step of a tentative **mixed-precision** scheme is given by:

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

Note: $\|\hat{\boldsymbol{d}}_j\|_2 = O(\Delta t)$ as $\Delta t \to 0$ for all j.

For a q-order preserving method we need to make sure all rounding errors are $O(\Delta t^{q+1})$.

The red term leads to an $O(u\Delta t)$ error! $\Rightarrow$ Must rewrite.

Mixed-precision RKC methods

We can rewrite:

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

Note: $\|\hat{\boldsymbol{d}}_j\|_2 = O(\Delta t)$ as $\Delta t \to 0$ for all j.

For a q-order preserving method we need to make sure all rounding errors are $O(\Delta t^{q+1})$.

The above is now a q-order preserving method as long as

$$\hat{\Delta} \boldsymbol{f}_j = \Big(\boldsymbol{f}(\hat{\boldsymbol{y}}_n + \hat{\boldsymbol{d}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}_n) \Big) + O(\Delta t^q) = \Delta \boldsymbol{f}_j + O(\Delta t^q), \quad \forall j.$$

Note: if $\hat{\Delta} \boldsymbol{f}_j = \Delta \boldsymbol{f}_j$ we recover the exact RKC scheme.

Computing the $\hat{\Delta}\boldsymbol{f}_j$ terms - RKC1

For RKC1 we want

$$\hat{\Delta}\boldsymbol{f}_j = \Delta\boldsymbol{f}_j + O(\Delta t) = \Big(\boldsymbol{f}(\hat{\boldsymbol{y}}_n + \hat{\boldsymbol{d}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}_n)\Big) + O(\Delta t), \quad \forall j.$$

Computing the $\hat{\Delta}\boldsymbol{f}_j$ terms - RKC1

For RKC1 we want

$$\hat{\Delta}\boldsymbol{f}_j = \Delta\boldsymbol{f}_j + O(\Delta t) = \Big(\boldsymbol{f}(\hat{\boldsymbol{y}}_n + \hat{\boldsymbol{d}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}_n)\Big) + O(\Delta t), \quad \forall j.$$

It is sufficient that $\hat{\Delta}\boldsymbol{f}_j = \boldsymbol{f}'(\hat{\boldsymbol{y}}_n)\hat{\boldsymbol{d}}_j + O(\Delta t)$ since $\Delta\boldsymbol{f}_j = \boldsymbol{f}'(\hat{\boldsymbol{y}}_n)\hat{\boldsymbol{d}}_j + O(\Delta t)$.

Computing the $\hat{\Delta} \boldsymbol{f}_j$ terms - RKC1

For RKC1 we want

$$\hat{\Delta} \boldsymbol{f}_j = \Delta \boldsymbol{f}_j + O(\Delta t) = \Big(\boldsymbol{f}(\hat{\boldsymbol{y}}_n + \hat{\boldsymbol{d}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}_n)\Big) + O(\Delta t), \quad \forall j.$$

It is sufficient that $\hat{\Delta} \boldsymbol{f}_j = \boldsymbol{f}'(\hat{\boldsymbol{y}}_n)\hat{\boldsymbol{d}}_j + O(\Delta t)$ since $\Delta \boldsymbol{f}_j = \boldsymbol{f}'(\hat{\boldsymbol{y}}_n)\hat{\boldsymbol{d}}_j + O(\Delta t)$.

- Since $\|\hat{\boldsymbol{d}}_j\|_2 = O(\Delta t)$, it is sufficient to approximate/evaluate the action of $\boldsymbol{f}'(\hat{\boldsymbol{y}}_n)$ in low precision. We need strategies that are robust to rounding errors. See next.
- We never need more than one high-precision evaluation of $\boldsymbol{f}$ every s stages.

Available strategies for efficient Jacobian evaluations/approximations

Available strategies for efficient Jacobian evaluations/approximations

- **Cheap analytical expression**.
- **Symbolic differentiation**.
- **Automatic differentiation**.

Available strategies for efficient Jacobian evaluations/approximations

- **Cheap analytical expression**.
- **Symbolic differentiation**.
- **Automatic differentiation**.
- **Noise-aware finite differences**:

$$\hat{\Delta}\boldsymbol{f}_j = \delta^{-1}\left(\hat{\boldsymbol{f}}(\hat{\boldsymbol{y}}_n + \delta\hat{\boldsymbol{d}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}_n)\right) = \boldsymbol{f}'(\hat{\boldsymbol{y}})\hat{\boldsymbol{d}}_j + O(\delta^{-1}u + \delta\|\hat{\boldsymbol{d}}_j\|_2^2).$$

Taking $\delta = 1$ is what is typically done, but leads to an $O(u)$ error!

Available strategies for efficient Jacobian evaluations/approximations

- **Cheap analytical expression**.
- **Symbolic differentiation**.
- **Automatic differentiation**.
- **Noise-aware finite differences**:

$$\hat{\Delta}\boldsymbol{f}_j = \delta^{-1}\left(\hat{\boldsymbol{f}}(\hat{\boldsymbol{y}}_n + \delta\hat{\boldsymbol{d}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}_n)\right) = \boldsymbol{f}'(\hat{\boldsymbol{y}})\hat{\boldsymbol{d}}_j + O(\delta^{-1}u + \delta\|\hat{\boldsymbol{d}}_j\|_2^2).$$

Taking $\delta = 1$ is what is typically done, but leads to an $O(u)$ error!

However, since $\|\hat{\boldsymbol{d}}_j\|_2 = O(\Delta t)$, we can take $\delta = O(\sqrt{u}\Delta t^{-1})$ to obtain

$$\delta^{-1}\left(\hat{\boldsymbol{f}}(\hat{\boldsymbol{y}}_n + \delta\hat{\boldsymbol{d}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}_n)\right) = \boldsymbol{f}'(\hat{\boldsymbol{y}}_n)\hat{\boldsymbol{d}}_j + O(\sqrt{u}\Delta t).$$

Computing the $\hat{\Delta}\boldsymbol{f}_j$ terms - RKC2

For RKC2 we want

$$\hat{\Delta}\boldsymbol{f}_j = \Delta\boldsymbol{f}_j + O(\Delta t^2) = \Big(\boldsymbol{f}(\hat{\boldsymbol{y}}_n + \hat{\boldsymbol{d}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}_n)\Big) + O(\Delta t^2), \quad \forall j.$$

Computing the $\hat{\Delta}\boldsymbol{f}_j$ terms - RKC2

For RKC2 we want

$$\hat{\Delta}\boldsymbol{f}_j = \Delta\boldsymbol{f}_j + O(\Delta t^2) = \Big(\boldsymbol{f}(\hat{\boldsymbol{y}}_n + \hat{\boldsymbol{d}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}_n)\Big) + O(\Delta t^2), \quad \forall j.$$

Let $\hat{\boldsymbol{v}}_j = \hat{\boldsymbol{d}}_j - c_j\Delta t\,\boldsymbol{f}(\boldsymbol{y}_n) = O(\Delta t^2)$. We compute a suitable $\hat{\Delta}\boldsymbol{f}_j$ as

$$\hat{\Delta}\boldsymbol{f}_j = \hat{\Delta}_1\boldsymbol{f}_j + \hat{\Delta}_2\boldsymbol{f}_j = \hat{\boldsymbol{f}}'(\hat{\boldsymbol{y}}_n)\hat{\boldsymbol{v}}_j + c_j\Delta t\,\boldsymbol{f}'(\hat{\boldsymbol{y}}_n)\boldsymbol{f}(\hat{\boldsymbol{y}}_n).$$

We prove that $\hat{\Delta}\boldsymbol{f}_j = \Delta\boldsymbol{f}_j + O(\Delta t^2)$. Again various evaluation strategies available and we never need more than one high-precision evaluation of $\boldsymbol{f}$ and $\boldsymbol{f}'$ every s stages.

Computing the $\hat{\Delta}\boldsymbol{f}_j$ terms - RKC2

For RKC2 we want

$$\hat{\Delta}\boldsymbol{f}_j = \Delta\boldsymbol{f}_j + O(\Delta t^2) = \Big(\boldsymbol{f}(\hat{\boldsymbol{y}}_n + \hat{\boldsymbol{d}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}_n)\Big) + O(\Delta t^2), \quad \forall j.$$

Let $\hat{\boldsymbol{v}}_j = \hat{\boldsymbol{d}}_j - c_j\Delta t\,\boldsymbol{f}(\boldsymbol{y}_n) = O(\Delta t^2)$. We compute a suitable $\hat{\Delta}\boldsymbol{f}_j$ as

$$\hat{\Delta}\boldsymbol{f}_j = \hat{\Delta}_1\boldsymbol{f}_j + \hat{\Delta}_2\boldsymbol{f}_j = \hat{\boldsymbol{f}}'(\hat{\boldsymbol{y}}_n)\hat{\boldsymbol{v}}_j + c_j\Delta t\,\boldsymbol{f}'(\hat{\boldsymbol{y}}_n)\boldsymbol{f}(\hat{\boldsymbol{y}}_n).$$

We prove that $\hat{\Delta}\boldsymbol{f}_j = \Delta\boldsymbol{f}_j + O(\Delta t^2)$. Again various evaluation strategies available and we never need more than one high-precision evaluation of $\boldsymbol{f}$ and $\boldsymbol{f}'$ every s stages.

Warning: This method is indeed 2nd-order accurate, yet it is unstable for s, Δt large!

Computing the $\hat{\Delta}\boldsymbol{f}_j$ terms - RKC2

For RKC2 we want

$$\hat{\Delta}\boldsymbol{f}_j = \Delta\boldsymbol{f}_j + O(\Delta t^2) = \Big(\boldsymbol{f}(\hat{\boldsymbol{y}}_n + \hat{\boldsymbol{d}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}_n)\Big) + O(\Delta t^2), \quad \forall j.$$

Let $\hat{\boldsymbol{v}}_j = \hat{\boldsymbol{d}}_j - c_j\Delta t\boldsymbol{f}(\boldsymbol{y}_n) = O(\Delta t^2)$. We compute a suitable $\hat{\Delta}\boldsymbol{f}_j$ as

$$\hat{\Delta}\boldsymbol{f}_j = \hat{\Delta}_1\boldsymbol{f}_j + \hat{\Delta}_2\boldsymbol{f}_j = \hat{\boldsymbol{f}}'(\hat{\boldsymbol{y}}_n)\hat{\boldsymbol{v}}_j + c_j\Delta t\boldsymbol{f}'(\hat{\boldsymbol{y}}_n)\boldsymbol{f}(\hat{\boldsymbol{y}}_n).$$

We prove that $\hat{\Delta}\boldsymbol{f}_j = \Delta\boldsymbol{f}_j + O(\Delta t^2)$. Again various evaluation strategies available and we never need more than one high-precision evaluation of $\boldsymbol{f}$ and $\boldsymbol{f}'$ every s stages.

Warning: This method is indeed 2nd-order accurate, yet it is unstable for s, Δt large!

Solution: set $\hat{\Delta}\boldsymbol{f}_j = \begin{cases} \hat{\Delta}_1\boldsymbol{f}_j + \hat{\Delta}_2\boldsymbol{f}_j, & \text{if } \|\hat{\boldsymbol{v}}_j\|_2 \leq \|\hat{\boldsymbol{d}}_j\|_2, \\ \tilde{\Delta}\boldsymbol{f}_j, & \text{if } \|\hat{\boldsymbol{v}}_j\|_2 > \|\hat{\boldsymbol{d}}_j\|_2, \end{cases}$

where $\tilde{\Delta}\boldsymbol{f}_j$ is the same 1st-order approximation we used for RKC1.

This modified RKC2 scheme is now both **stable and 2nd-order accurate**!

Convergence, nonlinear stability, and worst-case error behaviour

Theorem (C. and RdS 2022)

Our order-p mixed-precision RKC schemes are p-order preserving if $\boldsymbol{f}$ is of class C^2. Furthermore, let $e_n = \|\hat{\boldsymbol{y}}_n - \boldsymbol{y}_n\|_2$, then there exist constants $C_1, C_2 > 0$ such that, for all n and for all Δt for which the exact method is stable,

$$e_{n+1} \leq e_n + u\Delta t \min(C_1 \Delta t^p, C_2).$$

Convergence, nonlinear stability, and worst-case error behaviour

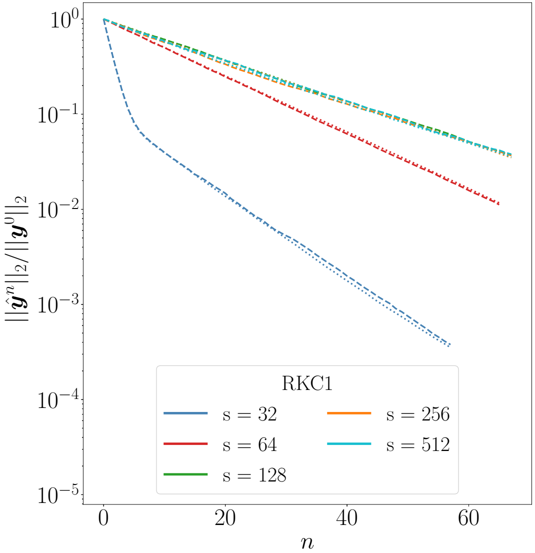
Theorem (C. and RdS 2022)

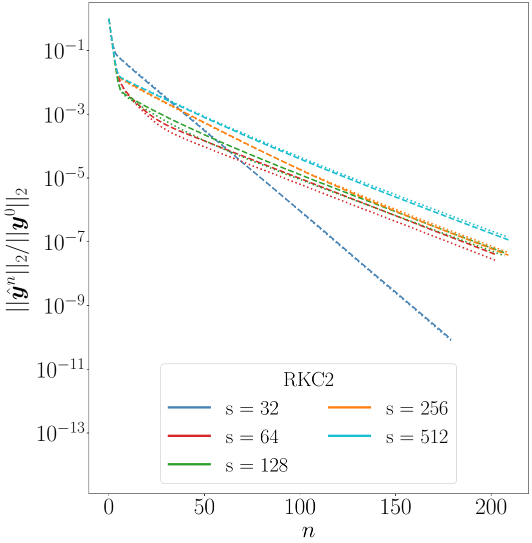
Our order-p mixed-precision RKC schemes are p-order preserving if $\boldsymbol{f}$ is of class C^2. Furthermore, let $e_n = \|\hat{\boldsymbol{y}}_n - \boldsymbol{y}_n\|_2$, then there exist constants $C_1, C_2 > 0$ such that, for all n and for all Δt for which the exact method is stable,

$$e_{n+1} \leq e_n + u\Delta t \min(C_1 \Delta t^p, C_2).$$

- **New theory.** First stability result for mixed-precision RK methods and first convergence result for explicit mixed-precision RK methods. RKC theory updated.
- **No classical stability result** The theory allows the error to grow. We can prove $e_{n+1} \leq e_n$ under stringent conditions. **Methods are stable in practice**.
- **A challenging theory.** Rounding errors are non-smooth and destroy spectral relations. This forbids any analysis based on eigenvalues or smoothness.

Numerical results - stability (2D heat eqn)



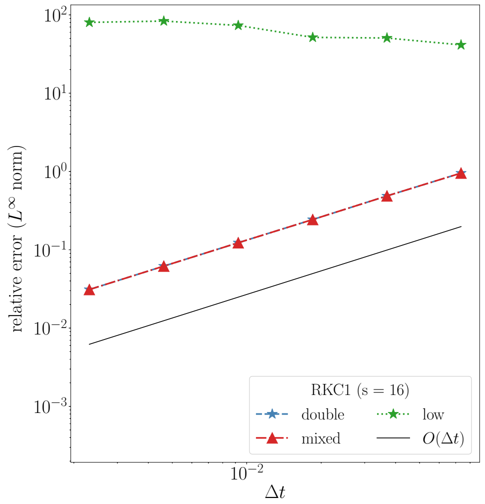


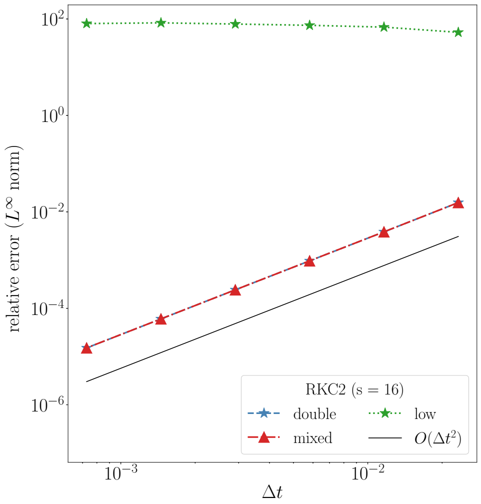
Diffusion coefficient $= 50$, $\Delta x = 4/s$, $\Delta t = s^2\|A\|_2^{-1} = 4s/50$.

Numerical results - time convergence

1D Brussellator model for chemical autocatalytic reactions (with Dirichlet BCs):

$$\begin{cases} \dot{u} = \alpha\Delta u + u^2 v - (b+1)u + a \\ \dot{v} = \alpha\Delta v - u^2 v + b u \end{cases}$$

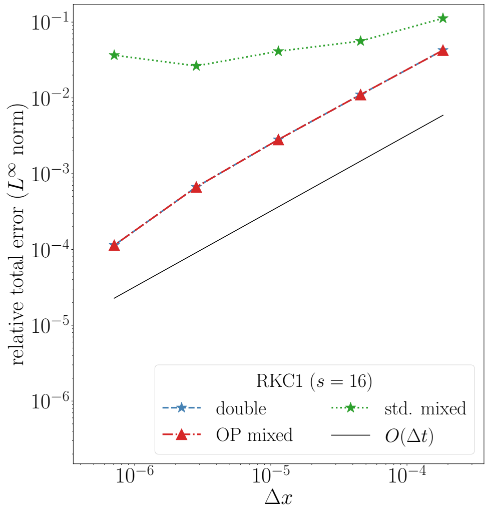


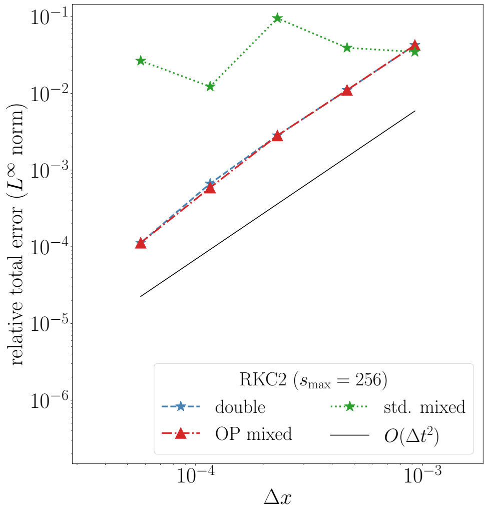


Numerical results - space-time convergence

Nonlinear diffusion model, 1D 4-Laplace diffusion operator (with Dirichlet BCs):

$$\dot{u} = \nabla \cdot (\|\nabla u\|_2^2 \nabla u) + f$$





4. Conclusions

Outlook

To sum up

- Mixed-precision algorithms require a careful implementation, but can bring significant memory, cost, and energy savings.
- We can make RKC methods as accurate as their high precision equivalent and almost as cheap as their fully low-precision counterpart.
- Our work extends to multirate RKC, and to any RK method for $q = 1, 2$.
- For order-preserving mixed-precision implicit RK methods, see, e.g., [Grant 2022].

Future research directions

- Hyperbolic PDE solvers, more reduced-/mixed-precision climate simulation, multilevel Monte Carlo methods.
- Expected speedups of our RKC schemes are 75% on CPUs and 94% on GPUs. It would be nice to verify this on hardware that supports half-precision computations.

Thank you for listening! If you want to know more...

Papers, slides, and more info at: `https://croci.github.io`

References

[1] M. Croci and G. Rosilho de Souza. Mixed-precision explicit stabilized Runge–Kutta methods for single-and multi-scale differential equations. *Journal of Computational Physics*, 464:111349, 2022.

[2] N. J. Higham and T. Mary. Mixed precision algorithms in numerical linear algebra. *Acta Numerica*, 31: 347–414, 2022.

[3] M. Klöwer, S. Hatfield, M. Croci, P. D. Düben, and T. N. Palmer. Fluid simulations accelerated with 16 bits: Approaching 4x speedup on A64FX by squeezing ShallowWaters.jl into Float16. *Journal of Advances in Modeling Earth Systems*, 2021.

[4] A. Abdelfattah, H. Anzt, E. G. Boman, E. Carson, T. Cojean, J. Dongarra, A. Fox, M. Gates, N. J. Higham, X. S. Li, et al. A survey of numerical linear algebra methods utilizing mixed-precision arithmetic. *The International Journal of High Performance Computing Applications*, 35(4):344–369, 2021.

[5] Z. J. Grant. Perturbed Runge–Kutta methods for mixed precision applications. *Journal of Scientific Computing*, 92(1):6, 2022.

[6] J. C. Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2003.

[7] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2002.

[8] A. Abdulle and A. A. Medovikov. Second order Chebyshev methods based on orthogonal polynomials. *Numerische Mathematik*, 90:1–18, 2001.

[9] J. G. Verwer, W. H. Hundsdorfer, and B. P. Sommeijer. Convergence properties of the Runge-Kutta-Chebyshev method. *Numerische Mathematik*, 57:157–178, 1990.

[10] P. J. van Der Houwen and B. P. Sommeijer. On the internal stability of explicit, m-stage Runge-Kutta methods for large m-values. *ZAMM - Journal of Applied Mathematics and Mechanics*, 60(10):479–485, 1980.

APPENDIX

Expected computational savings

Due to the limited availability of CPUs supporting half-precision computations, we rely on in-software emulation $\Rightarrow$ CPU timings not available.

Nevertheless, the mixed-precision schemes should be cheaper by roughly a factor

$$\varrho = \frac{sr - ((s-q) + qr)}{sr}, \quad \text{where} \quad r = \frac{\text{Cost of RHS evals in high}}{\text{Cost of RHS evals in low}}.$$

A scheme in double/half yields up to $r = 4$ on CPU and up to $r = 16$ on GPUs.

- For RK4 this leads to 56% $(q = 1)$ and 40% $(q = 2)$ savings on CPUs.
- Stabilised methods have lots of stages and low order: can essentially take $s \to \infty$, giving $\varrho \to 1 - 1/r$. E.g. this leads to a 75% speedup on CPUs (94% on GPUs).

Note: We have ignored additional savings related to memory/caching effects.

Stabilising RKC2

Recap: we computed $\hat{\Delta}\boldsymbol{f}_j = \hat{\Delta}_1\boldsymbol{f}_j + \hat{\Delta}_2\boldsymbol{f}_j$, where $\hat{\Delta}_1\boldsymbol{f}_j = \boldsymbol{f}'(\dots)\hat{\boldsymbol{v}}_j = O(\Delta t^2)$.

The culprit is the $\hat{\boldsymbol{v}}_j$ term: for small Δt this is small and ensures 2nd order convergence, but for large Δt it becomes large and leads to instability!

Stabilising RKC2

Recap: we computed $\hat{\Delta}\boldsymbol{f}_j = \hat{\Delta}_1\boldsymbol{f}_j + \hat{\Delta}_2\boldsymbol{f}_j$, where $\hat{\Delta}_1\boldsymbol{f}_j = \boldsymbol{f}'(\dots)\hat{\boldsymbol{v}}_j = O(\Delta t^2)$.

The culprit is the $\hat{\boldsymbol{v}}_j$ term: for small Δt this is small and ensures 2nd order convergence, but for large Δt it becomes large and leads to instability!

To fix this, consider the 1-order preserving evaluation of $\hat{\Delta}\boldsymbol{f}_j$ (same as for RKC1):

$$\tilde{\Delta}\boldsymbol{f}_j = \boldsymbol{f}_j'(\hat{\boldsymbol{y}}_j)\hat{\boldsymbol{d}}_j + O(\Delta t).$$

This leads to a stable scheme for large Δt, but is only first-order accurate for small Δt.

Stabilising RKC2

Recap: we computed $\hat{\Delta} \boldsymbol{f}_j = \hat{\Delta}_1 \boldsymbol{f}_j + \hat{\Delta}_2 \boldsymbol{f}_j$, where $\hat{\Delta}_1 \boldsymbol{f}_j = \boldsymbol{f}'(\dots)\hat{\boldsymbol{v}}_j = O(\Delta t^2)$.

The culprit is the $\hat{\boldsymbol{v}}_j$ term: for small Δt this is small and ensures 2nd order convergence, but for large Δt it becomes large and leads to instability!

To fix this, consider the 1-order preserving evaluation of $\hat{\Delta} \boldsymbol{f}_j$ (same as for RKC1):

$$\tilde{\Delta} \boldsymbol{f}_j = \boldsymbol{f}_j'(\hat{\boldsymbol{y}}_j)\hat{\boldsymbol{d}}_j + O(\Delta t).$$

This leads to a stable scheme for large Δt, but is only first-order accurate for small Δt.

Solution: set $\hat{\Delta} \boldsymbol{f}_j = \begin{cases} \hat{\Delta}_1 \boldsymbol{f}_j + \hat{\Delta}_2 \boldsymbol{f}_j, & \text{if } \|\hat{\boldsymbol{v}}_j\|_2 \le \|\hat{\boldsymbol{d}}_j\|_2, \\ \tilde{\Delta} \boldsymbol{f}_j, & \text{if } \|\hat{\boldsymbol{v}}_j\|_2 > \|\hat{\boldsymbol{d}}_j\|_2. \end{cases}$

This leads to a 2nd-order *and* stable method.

Convergence and nonlinear stability

Theorem (C. and RdS 2022)

1) Our order-p mixed-precision RKC schemes are p-order preserving if $\boldsymbol{f}$ is of class C^2.

Convergence and nonlinear stability

Theorem (C. and RdS 2022)

1) Our order-p mixed-precision RKC schemes are p-order preserving if $\boldsymbol{f}$ is of class C^2.

2) Furthermore, if there exist $c_1, c_2 > 0$ independent from Δt such that, for all j, n,

$$\begin{aligned} &\text{(i)} \quad \|\hat{\Delta}\boldsymbol{f}_j - \Delta\boldsymbol{f}_j\|_2 \leq c_1\|\hat{\boldsymbol{y}}_n\|_2, \\ &\text{(ii)} \quad \|\hat{\boldsymbol{d}}_j\|_2 \leq c_2\|\hat{\boldsymbol{y}}_n\|_2, \end{aligned}$$

then there also exist constants $C_1, C_2 > 0$ such that, for all n and for all Δt for which the exact method is stable, the following non-asymptotic bound holds:

$$\|\hat{\boldsymbol{y}}_{n+1} - \boldsymbol{y}_{n+1}\|_2 \leq \|\hat{\boldsymbol{y}}_n - \boldsymbol{y}_n\|_2 + \Delta t \min(C_1\Delta t^p, C_2)$$

Note: (i) and (ii) control the amplification of rounding errors in the non-asymptotic regime. Both conditions are automatically satisfied if either $\Delta t \to 0$ or if $\boldsymbol{f}$ is linear.

Internal error propagation and linear stability

Theorem (C. and RdS 2022)

Let $\boldsymbol{f}(\boldsymbol{y}) = A\boldsymbol{y}$ with A being a symmetric npd matrix. Then, Conditions (i)-(ii) *in the previous theorem are automatically satisfied. Furthermore, our order-p schemes satisfy*

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

where $\boldsymbol{r}_s^p$ contains the rounding errors introduced at time step n, and is bounded by

$$\|\boldsymbol{r}_s^p\|_2 \leq \Psi_p(\Delta t, A)\left((1 + C\Delta t u)^{s-1} - 1\right)\|\hat{\boldsymbol{y}}^n\|_2,$$

where $0 \leq \Psi_p(\Delta t, A) \leq 2$, $\Psi_p(\Delta t, A) = O(\Delta t^p)$, and $C > 0$.

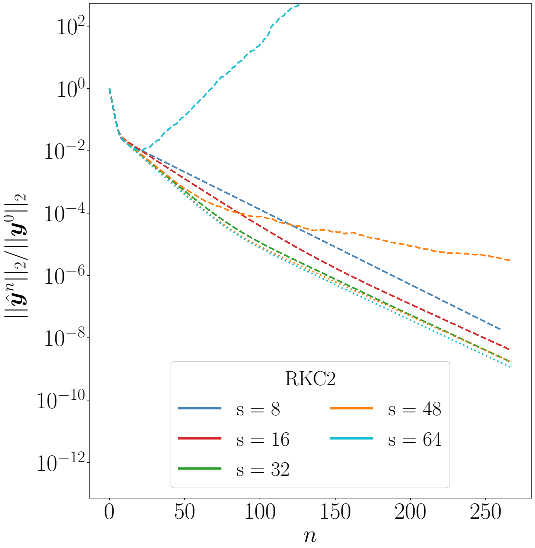
Linear stability result - some comments

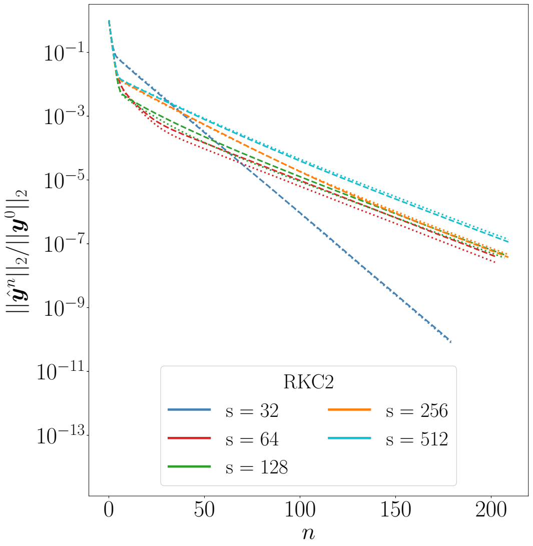
- **Updated RKC theory.** The bounds account for rounding errors propagating from previous stages, an overlooked phenomenon in RKC theory [Verwer et al. 1990].
- **No classical stability proof.** Our theory allows the error to grow for large Δt since

  $$\|\hat{\boldsymbol{y}}_{n+1} - \boldsymbol{y}_{n+1}\|_2 \leq \|R_s^p(\Delta t A)\|_2 \|\hat{\boldsymbol{y}}_n - \boldsymbol{y}_n\|_2 + \|\boldsymbol{r}_s^p(\hat{\boldsymbol{y}}_n)\|_2 \leq (1+\alpha)\|\hat{\boldsymbol{y}}_n - \boldsymbol{y}_n\|_2,$$

  where $\alpha > 0$. We can prove no error growth under stringent conditions on $\kappa(A)$.
- **Rounding errors pose new challenges:** they are non-smooth and destroy any spectral relation between iterates. This forbids any analysis based on eigenvalues or smoothness and results in a pessimistic worst-case bound.
- **Methods are stable in practice**, independently from $\kappa(A)$. The worst-case behaviour is not observed.

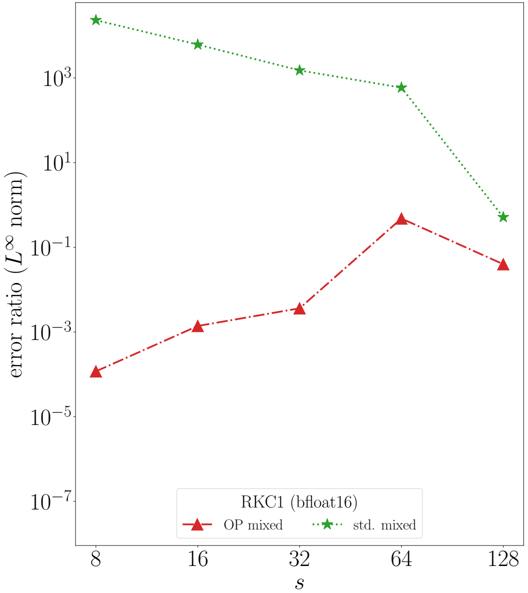
Numerical results - RKC2 stability (2D nonlinear heat eqn, half precision)

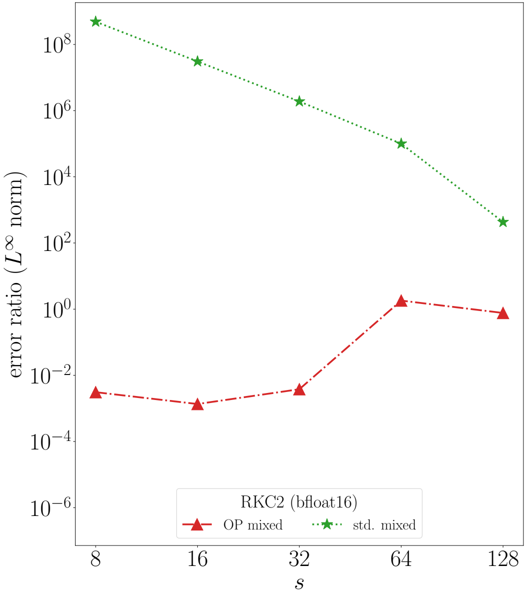




Behaviour of RKC2 numerical solution without (left) and with (right) stabilization.

Numerical results - error vs number of stages (4-Laplace diffusion)





error ratio $=$ rounding error $/$ time-discretization error.