Solving differential equations in reduced- and mixed-precision

Matteo Croci

Oden Institute Seminar, UT Austin, 21 April 2022

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

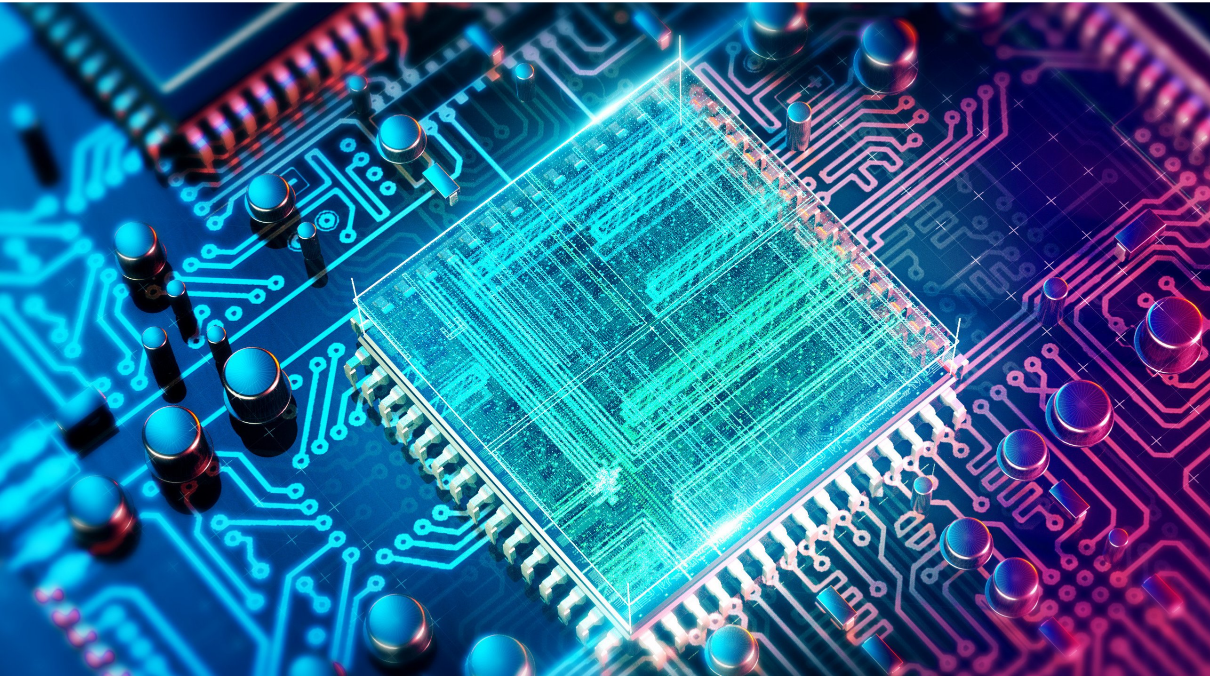
Overview

1. Introduction and background

Main references:

- A. Abdelfattah, H. Anzt, E. G. Boman, E. Carson, 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
- N. J. Higham and T. Mary. Mixed precision algorithms in numerical linear algebra, 2021. URL `http://eprints.maths.manchester.ac.uk/2841/1/paper_eprint.pdf`
- M. Croci, M. Fasi, N. J. Higham, T. Mary, and M. Mikaitis. Stochastic rounding: implementation, error analysis and applications. *Royal Society Open Science*, 9:211631, 2022
- 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

Objective: developing reduced- and mixed-precision DE solvers



Reduced- and mixed-precision algorithms

Reduced-precision algorithms

Reduced-precision algorithms obtain an as accurate solution as possible given the precision while avoiding catastrophic rounding error accumulation.

[1]Review articles: [Abdelfattah et al. 2021], [Higham and Mary 2021], [C. et al. 2021].

Reduced- and mixed-precision algorithms

Reduced-precision algorithms

Reduced-precision algorithms obtain an as accurate solution as possible given the precision while avoiding catastrophic rounding error accumulation.

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.

[1] Review articles: [Abdelfattah et al. 2021], [Higham and Mary 2021], [C. et al. 2021].

Reduced- and mixed-precision algorithms

Reduced-precision algorithms

Reduced-precision algorithms obtain an as accurate solution as possible given the precision while avoiding catastrophic rounding error accumulation.

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.

- This is now a very active field of investigation[1] with many new developments led mainly by the numerical linear algebra community.
- Lots of new reduced-/mixed-precision algorithms: matrix factorizations, direct linear solvers, Krylov subspace methods, preconditioning, multigrid, nonlinear solvers.
- There is still much to discover on the topic.

[1] Review articles: [Abdelfattah et al. 2021], [Higham and Mary 2021], [C. et al. 2021].

Floating point formats

Format	unit roundoff u	$x_{\min}$	$x_{\max}$
bfloat16 (half)	$2^{-8} \approx 3.91 \times 10^{-3}$	1.18×10^{-38}	3.39×10^{38}
fp16 (half)	$2^{-11} \approx 4.88 \times 10^{-4}$	6.10×10^{-5}	6.55×10^{4}
fp32 (single)	$2^{-24} \approx 5.96 \times 10^{-8}$	1.18×10^{-38}	3.40×10^{38}
fp64 (double)	$2^{-53} \approx 1.11 \times 10^{-16}$	2.22×10^{-308}	1.80×10^{308}

Recent trend in scientific computing: u is getting larger!

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

Example: towards climate simulations in half precision [Klöwer et al. 2021]

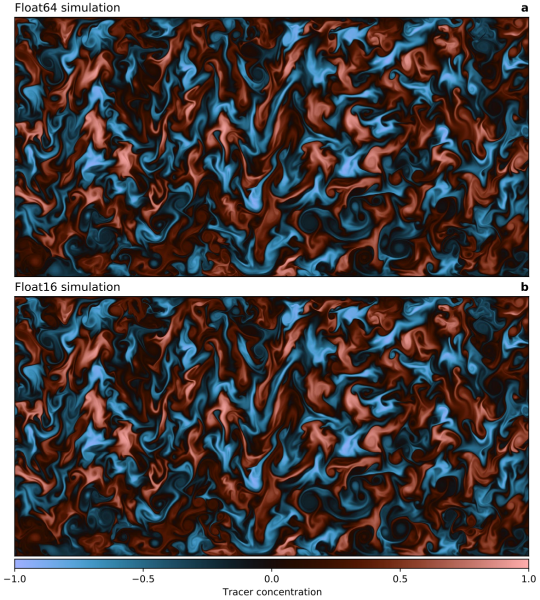
Shallow-water eqs for 2D oceanic flow:

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

Numerical scheme: explicit 4th-order timestepping on a staggered grid.

Techniques used for fp16 simulations:

- Scaling and squeezing.
- Kahan compensated summation.
- Performed using A64FX chips on Fugaku (1st in TOP500).



Note: all other results in this talk use *precision emulation* in software.

2. Solving parabolic PDEs in half precision

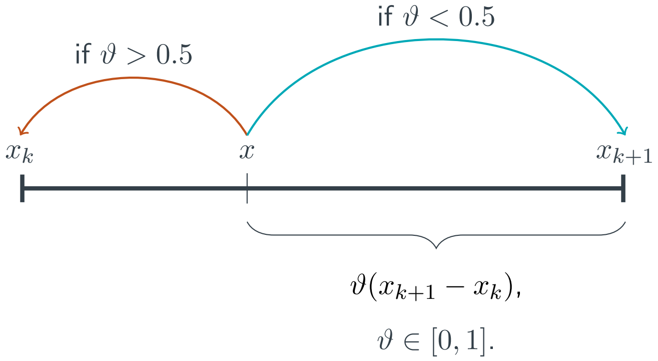
Joint with: M. B. Giles (University of Oxford)

Algorithm type: reduced-precision (half), using **stochastic rounding**.

Main references:

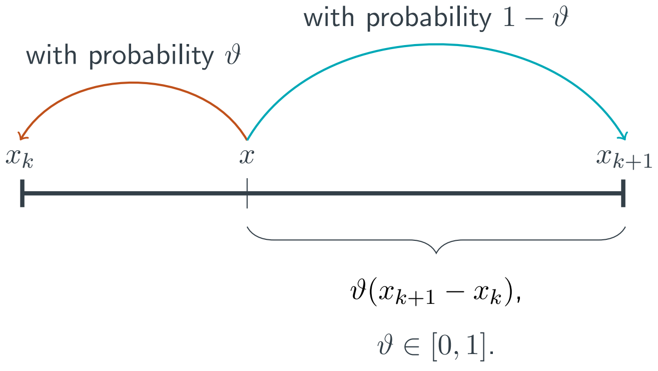
- M. Croci and M. B. Giles. Effects of round-to-nearest and stochastic rounding in the numerical solution of the heat equation in low precision. *IMA Journal of Numerical Analysis (to appear)*, 2022. URL `https://arxiv.org/pdf/2010.16225`
- M. Croci, M. Fasi, N. J. Higham, T. Mary, and M. Mikaitis. Stochastic rounding: implementation, error analysis and applications. *Royal Society Open Science*, 9:211631, 2022

Round to nearest



$$\mathrm{fl}(x) = x(1+\delta), \quad \text{with} \quad |\delta| \le u.$$

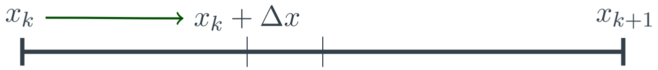
Stochastic rounding (review article [C. et al. 2022])


with probability $1-\vartheta$
with probability ϑ
x_k
x
x_{k+1}
$\vartheta(x_{k+1}-x_k)$,
$\vartheta \in [0,1]$.

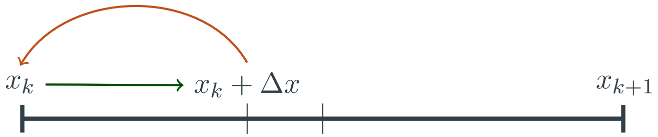
$$\mathrm{sr}(x) = x(1+\delta(\omega)), \quad |\delta| \le 2u, \quad \text{and} \quad \mathbb{E}[sr(x)] = x, \quad \mathbb{E}[\delta_i|\delta_1,\dots,\delta_{i-1}] = \mathbb{E}[\delta_i] = 0.$$

Limited (yet growing) hardware support. Many new applications in Sci. Comp. and ML.

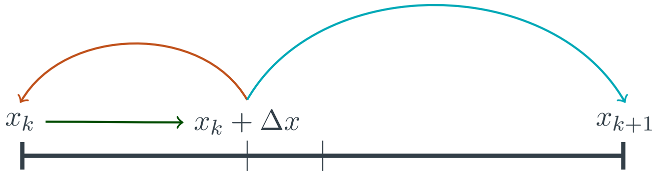
RtN might cause stagnation



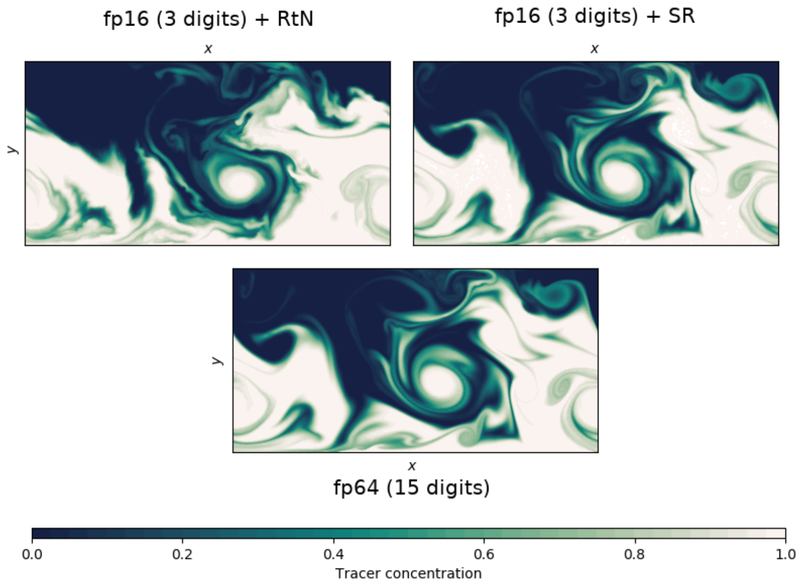
RtN might cause stagnation



SR is resilient to stagnation



Interesting results by Milan Klöwer (University of Oxford)



Note: not just due to stagnation, SR decorrelates errors and causes error cancellation!

Heat equation as a test problem

$$\begin{cases} \dot{\mathfrak{u}}(t,\boldsymbol{x}) = \nabla^2 \mathfrak{u}(t,\boldsymbol{x}) + f(t,\boldsymbol{x}), & \boldsymbol{x} \in D = [0,1]^d, \quad t > 0, \\ \mathfrak{u}(0,\boldsymbol{x}) = \mathfrak{u}_0(x), & \boldsymbol{x} \in D, \\ \mathfrak{u}(t,\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \partial D, \qquad\qquad t > 0. \end{cases}$$

Heat equation as a test problem

$$\begin{cases} \dot{\mathfrak{u}}(t,\boldsymbol{x}) = \nabla^2 \mathfrak{u}(t,\boldsymbol{x}) + f(t,\boldsymbol{x}), & \boldsymbol{x} \in D = [0,1]^d, \quad t > 0, \\ \mathfrak{u}(0,\boldsymbol{x}) = \mathfrak{u}_0(x), & \boldsymbol{x} \in D, \\ \mathfrak{u}(t,\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \partial D, \qquad\qquad t > 0. \end{cases}$$

We use **finite differences** in space: let $\boldsymbol{U}(t) \in \mathbb{R}^K$ with $\boldsymbol{U}_i(t) \approx \mathfrak{u}(t,\boldsymbol{x}_i)$, then

$$\dot{\boldsymbol{U}}(t) = -A\boldsymbol{U}(t) + \boldsymbol{F}(t),$$

where A is the (spd) **stiffness matrix** or discrete laplacian.

Heat equation as a test problem

$$\begin{cases} \dot{\mathfrak{u}}(t,\boldsymbol{x}) = \nabla^2 \mathfrak{u}(t,\boldsymbol{x}) + f(t,\boldsymbol{x}), & \boldsymbol{x} \in D = [0,1]^d, \quad t > 0, \\ \mathfrak{u}(0,\boldsymbol{x}) = \mathfrak{u}_0(x), & \boldsymbol{x} \in D, \\ \mathfrak{u}(t,\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \partial D, \quad\quad\quad t > 0. \end{cases}$$

We use **finite differences** in space: let $\boldsymbol{U}(t) \in \mathbb{R}^K$ with $\boldsymbol{U}_i(t) \approx \mathfrak{u}(t, \boldsymbol{x}_i)$, then

$$\dot{\boldsymbol{U}}(t) = -A\boldsymbol{U}(t) + \boldsymbol{F}(t),$$

where A is the (spd) **stiffness matrix** or discrete laplacian.
Discretising in time with a **Runge-Kutta** method yields the numerical scheme:

$$\boldsymbol{U}^{n+1} = S\boldsymbol{U}^n + \Delta t \boldsymbol{F}^n$$

for some matrix S dependent on $\Delta t A$, hence on $\lambda = \Delta t / h^2$. For instance,

$\boldsymbol{U}^{n+1} = (I - \Delta t A)\boldsymbol{U}^n + \Delta t \boldsymbol{F}^n_{\text{FE}}$, (FE), $\boldsymbol{U}^{n+1} = (I + \Delta t A)^{-1}\boldsymbol{U}^n + \Delta t \boldsymbol{F}^n_{\text{BE}}$, (BE).

In this part of the talk we work in **bfloat16 half precision**, $u = 2^{-8} \approx 4 \times 10^{-3}$.

Use the delta form [Hairer and Wanner 1996]

How to best implement the Runge-Kutta scheme? Use the **delta form**!

$$\textbf{Standard form:}\quad \boldsymbol{U}^{n+1} = S\boldsymbol{U}^n + \Delta t \boldsymbol{F}^n.$$

$$\textbf{Delta form:}\quad \boldsymbol{U}^{n+1} = \boldsymbol{U}^n + \Delta t \left(-\tilde{S}A\boldsymbol{U}^n + \tilde{\boldsymbol{F}}^n\right) = \boldsymbol{U}^n + \Delta \boldsymbol{U}^n.$$

e.g. $S_{\mathsf{FE}} = (I - \Delta t A)$, $\tilde{S}_{FE} = 1$, and $S_{\mathsf{BE}} = \tilde{S}_{\mathsf{BE}} = (I + \Delta t A)^{-1}$.

Use the delta form [Hairer and Wanner 1996]

How to best implement the Runge-Kutta scheme? Use the **delta form**!

$$\textbf{Standard form:}\quad \boldsymbol{U}^{n+1} = S\boldsymbol{U}^n + \Delta t \boldsymbol{F}^n.$$

$$\textbf{Delta form:}\quad \boldsymbol{U}^{n+1} = \boldsymbol{U}^n + \Delta t\left(-\tilde{S}A\boldsymbol{U}^n + \tilde{\boldsymbol{F}}^n\right) = \boldsymbol{U}^n + \Delta\boldsymbol{U}^n.$$

e.g. $S_{\text{FE}} = (I - \Delta t A)$, $\tilde{S}_{FE} = 1$, and $S_{\text{BE}} = \tilde{S}_{\text{BE}} = (I + \Delta t A)^{-1}$.

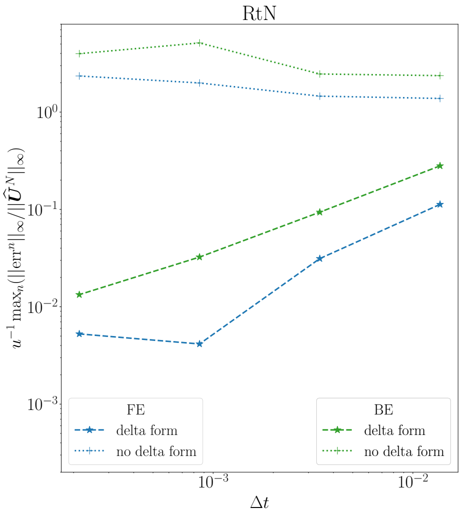
We prove that:

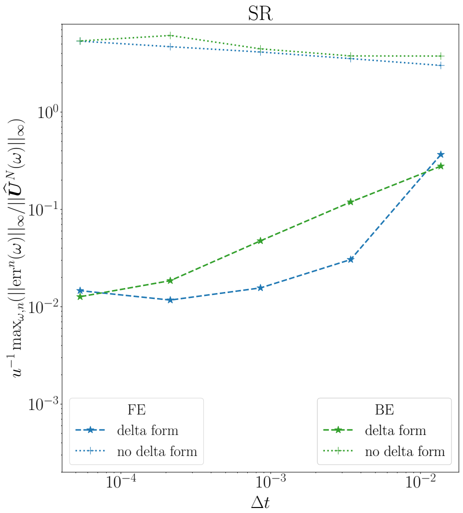
- Errors in the computation of $S\boldsymbol{U}^n$ are $O(u)$ with large constant.
- Errors in the computation of $\Delta\boldsymbol{U}^n$ are $O(\Delta t^p u)$, $p > 0$.

Therefore:

- The delta form produces much smaller rounding errors at each time step.
- Most of the rounding errors in the delta form are introduced into the final addition.

Worst-case local rounding errors in 2D





Note: from now on we use the delta form.

RtN vs SR

Why is RtN in low precision bad for the heat equation?

a) Stagnation:

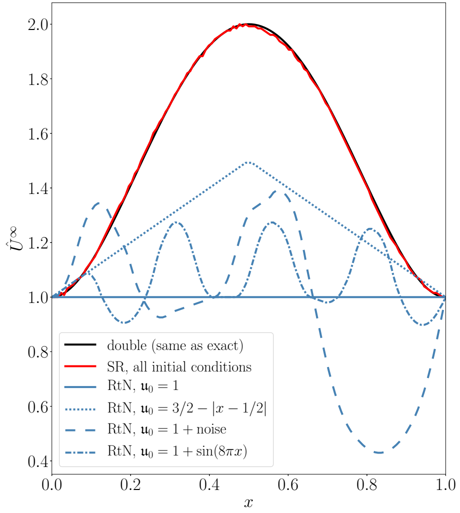
- RtN always stagnates for sufficiently small Δt (recall $\Delta \boldsymbol{U}^n = O(u\Delta t^p)$).
- The RtN solution is initial condition, discretization and precision dependent.

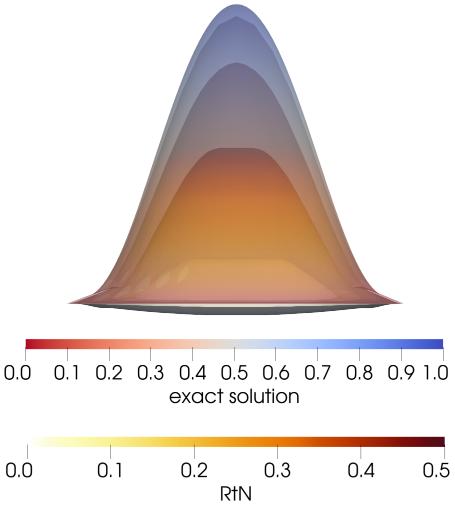
b) Global error:

- RtN rounding errors are strongly correlated and grow rapidly until stagnation.

SR fixes all these issues!

a) Stagnation (left 1D, right 2D)





RtN computations are discretization and initial condition dependent. SR works!

b) Global rounding errors [C. and Giles 2020]

Let $\varepsilon^n \in \mathbb{R}^K$ be the vector containing all rounding errors introduced at time step n. Define the global rounding error $\boldsymbol{E}^n = \hat{\boldsymbol{U}}^n - \boldsymbol{U}^n$. It can be shown that

$$\boldsymbol{E}^{n+1} = S\boldsymbol{E}^n + \varepsilon^n.$$

Traditional results for ODEs [Henrici 1962-1963, Arató 1983]: ε^n is $O(\Delta t^2)$.

We can distinguish two cases:

RtN: we can only assume the worst-case scenario, $|\varepsilon_i^n| = O(u)$ for all n, i.

SR: the $\varepsilon_{\boldsymbol{i}}^n$ are zero-mean, independent in space and mean-independent in time.

b) Global rounding errors [C. and Giles 2020]

Let $\varepsilon^n \in \mathbb{R}^K$ be the vector containing all rounding errors introduced at time step n. Define the global rounding error $\boldsymbol{E}^n = \hat{\boldsymbol{U}}^n - \boldsymbol{U}^n$. It can be shown that

$$\boldsymbol{E}^{n+1} = S\boldsymbol{E}^n + \varepsilon^n.$$

Traditional results for ODEs [Henrici 1962-1963, Arató 1983]: ε^n is $O(\Delta t^2)$.

We can distinguish two cases:

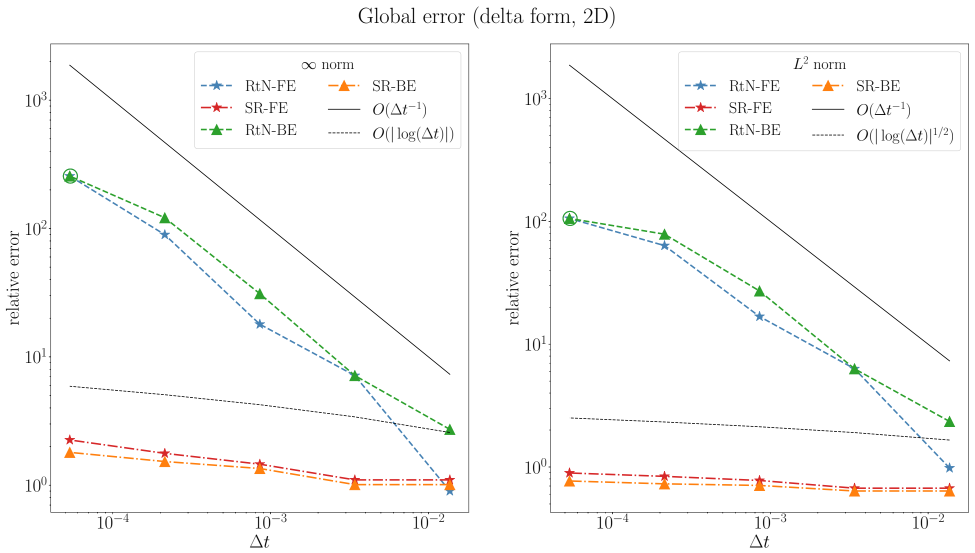
RtN: we can only assume the worst-case scenario, $|\varepsilon_i^n| = O(u)$ for all n, i.

SR: the $\varepsilon_{\boldsymbol{i}}^n$ are zero-mean, independent in space and mean-independent in time.

Mode	Norm	1D	2D	3D
RtN	L^2, ∞	$O(u\Delta t^{-1})$	$O(u\Delta t^{-1})$	$O(u\Delta t^{-1})$
SR	$\mathbb{E}[\|\|\cdot\|\|_{\infty}^2]^{1/2}$	$O(u\Delta t^{-1/4}\ell(\Delta t)^{1/2})$	$O(u\ell(\Delta t))$	$O(u\ell(\Delta t)^{1/2})$
SR	$\mathbb{E}[\|\|\cdot\|\|_{L^2}^2]^{1/2}$	$O(u\Delta t^{-1/4})$	$O(u\ell(\Delta t)^{1/2})$	$O(u)$

Asymptotic global rounding error blow-up rates; $\ell(\Delta t) = |\log(\lambda^{-1}\Delta t)|$.

b) Global rounding errors (here at steady-state)



Note: relative error $=$ error $\times\ (u||\boldsymbol{U}^N||)^{-1}$

3. Mixed-precision explicit stabilised Runge-Kutta methods

Joint with: G. Rosilho De Souza (EPFL, USI Lugano).

Algorithm type: mixed-precision (double/bfloat16) using **round-to-nearest**.

Main reference:

- M. Croci and G. R. de Souza. Mixed-precision explicit stabilized Runge-Kutta methods for single- and multi-scale differential equations, 2022. URL `https://arxiv.org/pdf/2109.12153`

Framework and objective

We consider mixed-precision explicit RK schemes for the solution of ODEs in the form

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

where $\boldsymbol{f}(t, \boldsymbol{y})$ is of sufficiently smooth, and from now on set $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{y}(t))$ for simplicity. In our experiments: the ODE is a result of a method-of-lines discretisation of a PDE.

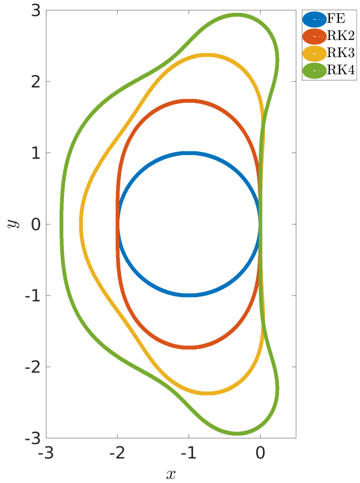
Objective

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

Recall: in this part of the talk we only use RtN.

Absolute stability

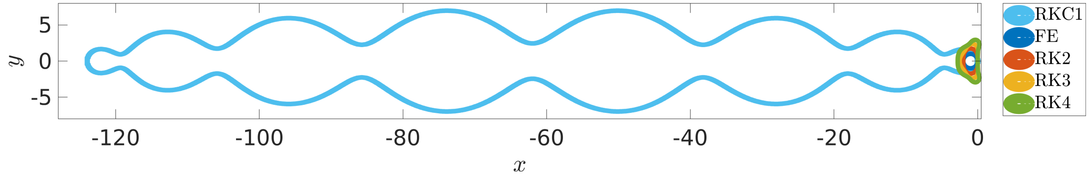
Dahlquist's test problem: $y' = \lambda y$, $y(0) = 1$.
s-stage RK method $y^n = R_s(z)^n$, where $z = \Delta t \lambda = x + iy$. Stable if $|R_s(z)| < 1$.



Explicit stabilized Runge-Kutta methods[2]

Idea: pick the poly $R_s(z)$ so as to maximise the stability region. For parabolic problems: use orthogonal polys, e.g. Runge-Kutta-Chebyshev (RKC). $\rightsquigarrow O(s^2)$ region.

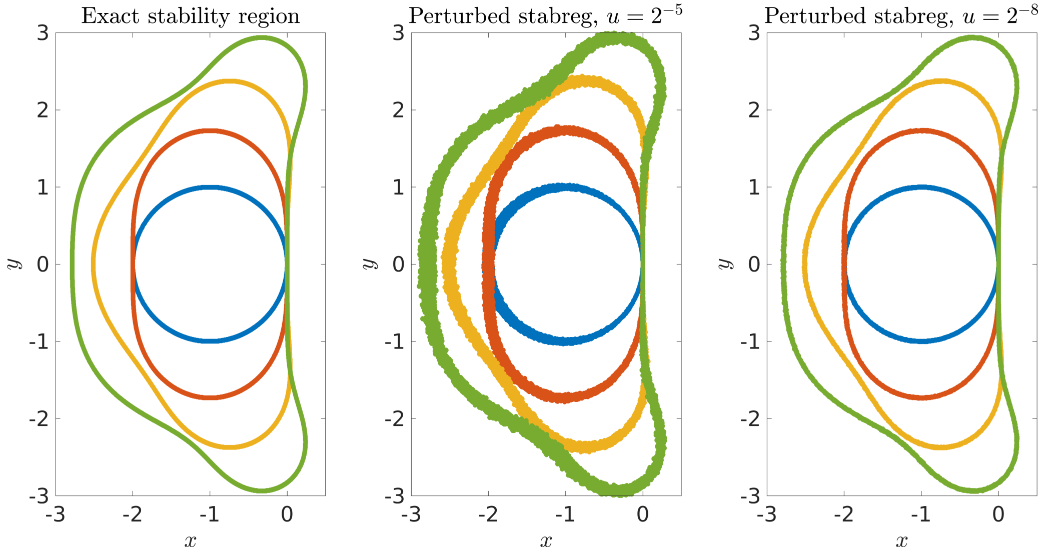
ESRK methods are of low-order ($p \leq 4$), but they use a lot of stages to maximise stability (i.e. not for accuracy purposes) $\rightarrow$ can do most of these in low precision!



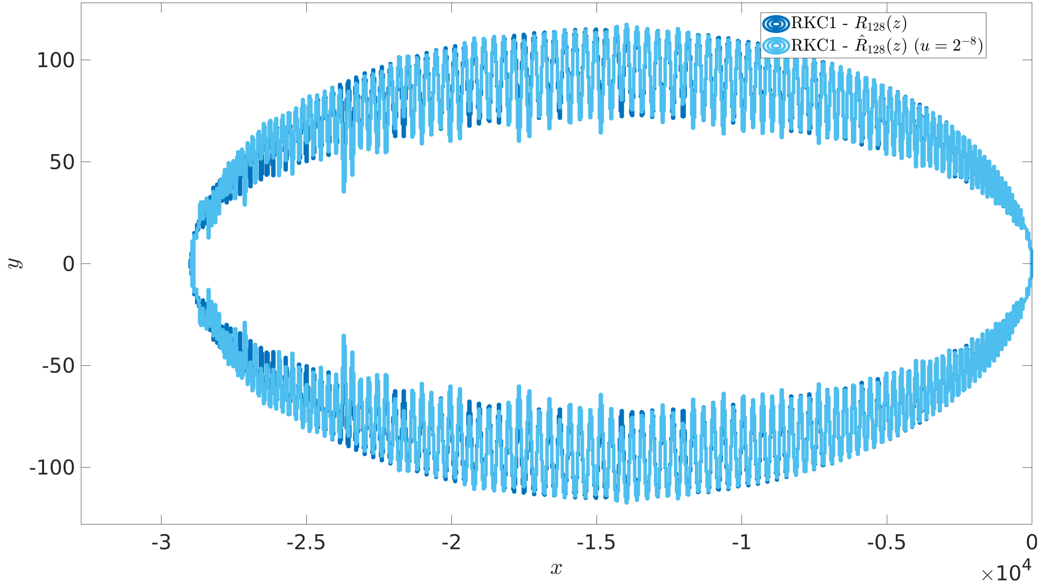
Absolute stability region of RKC1 with $s = 8$ vs those of other explicit methods.

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

Linear stability for RK methods (in practice)



Linear stability for RKC (in practice, $s = 128$, $u = 2^{-8}$)



Linear problems, i.e. $\boldsymbol{f}(\boldsymbol{y}) = A\boldsymbol{y}$

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

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

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

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

Linear problems, i.e. $\boldsymbol{f}(\boldsymbol{y}) = A\boldsymbol{y}$

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

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

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

Evaluating the scheme in finite precision yields:

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

Linear problems

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

Linear problems

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

Let us consider the following scenarios:

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

Linear problems

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

Let us consider the following scenarios:

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

Linear problems

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

Let us consider the following scenarios:

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

Order-preserving mixed-precision RK methods

From now on we set u to be the unit roundoff of the low-precision format.

Order-preserving mixed-precision RK methods

From now on we set u to be the unit roundoff of the low-precision format.

Assumption

Operations performed in high-precision are exact.

Definition (Order-preserving mixed-precision RK method)

A p-th order mixed-precision RK method is q-order-preserving ($q \in \{1, \ldots, p\}$), if it converges with order q under the above assumption.

Order-preserving mixed-precision RK methods

From now on we set u to be the unit roundoff of the low-precision format.

Assumption

Operations performed in high-precision are exact.

Definition (Order-preserving mixed-precision RK method)

A p-th order mixed-precision RK method is q-order-preserving ($q \in \{1, \dots, p\}$), if it converges with order q under the above assumption.

Existing methods: Standard mixed-precision RK schemes perform all function evaluations in low-precision and they are therefore not order-preserving.

Our objective: Use q function evaluations in high precision to obtain a q-order-preserving mixed-precision RK method.

Result: We can construct q-order preserving RK methods for any q for linear problems, and for $q = 1, 2$ for nonlinear problems. Today we focus on RKC methods.

Mixed-precision RKC methods

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

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

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:

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

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:

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

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

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

- Many options available, both generic and problem-dependent (see our paper), to approximate/evaluate $\boldsymbol{f}'(\hat{\boldsymbol{y}}^n)$ (can use low precision).
- In practice we never need more than one low-precision Jacobian evaluation and one high-precision evaluation of $\boldsymbol{f}$ every s stages.

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

For RKC2 we want

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

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

$$\hat{\Delta}\boldsymbol{f}_j = \hat{\Delta}_1\boldsymbol{f}_j + \hat{\Delta}_2\boldsymbol{f}_j = \left[\hat{\boldsymbol{f}}'(\hat{\boldsymbol{y}}^n + c_j\Delta t\boldsymbol{f}(\hat{\boldsymbol{y}}^n))\hat{\boldsymbol{v}}_j\right] + \left[c_j\Delta t\boldsymbol{f}'(\hat{\boldsymbol{y}}^n)\boldsymbol{f}(\hat{\boldsymbol{y}}^n)\right].$$

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

$$\hat{\Delta}\boldsymbol{f}_j = \hat{\Delta}_1\boldsymbol{f}_j + \hat{\Delta}_2\boldsymbol{f}_j = \left[\hat{\boldsymbol{f}}'(\hat{\boldsymbol{y}}^n + c_j\Delta t\boldsymbol{f}(\hat{\boldsymbol{y}}^n))\hat{\boldsymbol{v}}_j\right] + \left[c_j\Delta t\boldsymbol{f}'(\hat{\boldsymbol{y}}^n)\boldsymbol{f}(\hat{\boldsymbol{y}}^n)\right].$$

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

$$\hat{\Delta}\boldsymbol{f}_j = \hat{\Delta}_1\boldsymbol{f}_j + \hat{\Delta}_2\boldsymbol{f}_j = \left[\hat{\boldsymbol{f}}'(\hat{\boldsymbol{y}}^n + c_j\Delta t\boldsymbol{f}(\hat{\boldsymbol{y}}^n))\hat{\boldsymbol{v}}_j\right] + \left[c_j\Delta t\boldsymbol{f}'(\hat{\boldsymbol{y}}^n)\boldsymbol{f}(\hat{\boldsymbol{y}}^n)\right].$$

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

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 and linear stability

Theorem

Our order-p mixed-precision RKC schemes are p-order preserving for any $\boldsymbol{f}$ of class C^2.

Convergence and linear stability

Theorem

Our order-p mixed-precision RKC schemes are p-order preserving for any $\boldsymbol{f}$ of class C^2.

Theorem

Let $\boldsymbol{f}(\boldsymbol{y}) = A\boldsymbol{y}$ with A being a symmetric npd matrix. Our order-p schemes satisfy

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

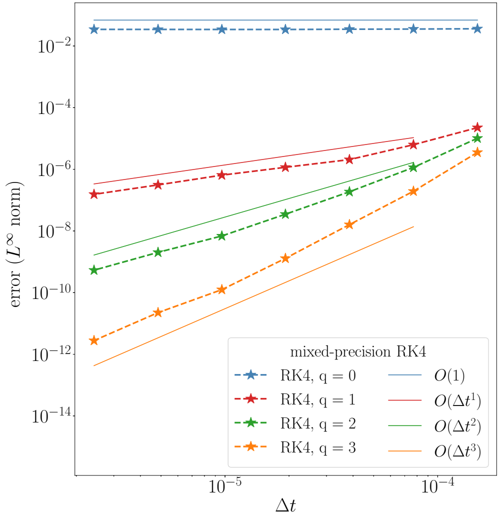
where $\boldsymbol{r}_s^p$ contains the rounding errors introduced at time step n, and is bounded by

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

where $\Psi_p(\Delta t, A) = O(\Delta t^p)$ and $C_p(s) > 0$.

Note: for more details, see our paper [C. and Rosilho de Souza 2022].

Numerical results - convergence (3D heat eqn)

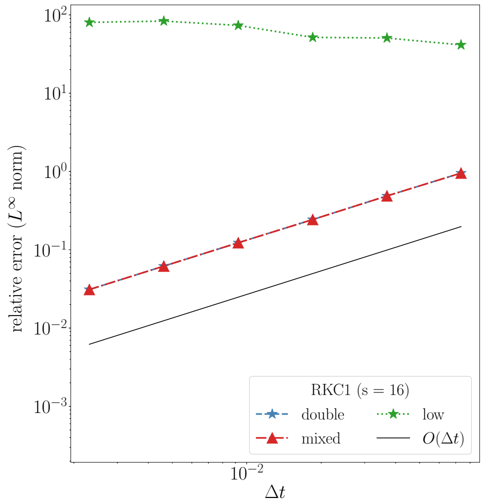


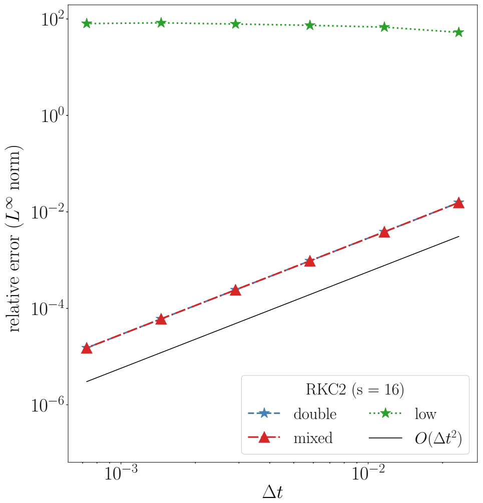
The transition from order p to order q happens roughly when $\Delta t = O(||A||^{-1} u^{\frac{1}{p-q}})$

Numerical results - convergence

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

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

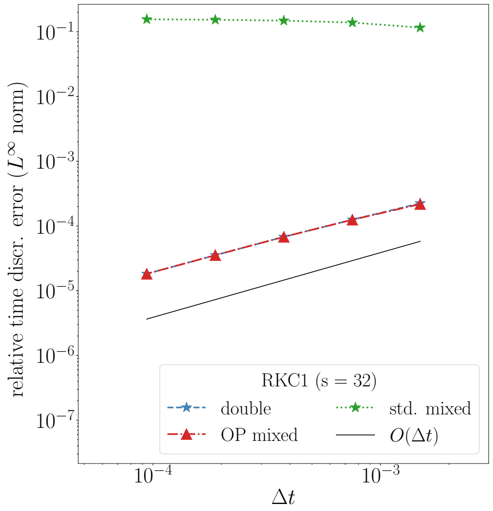


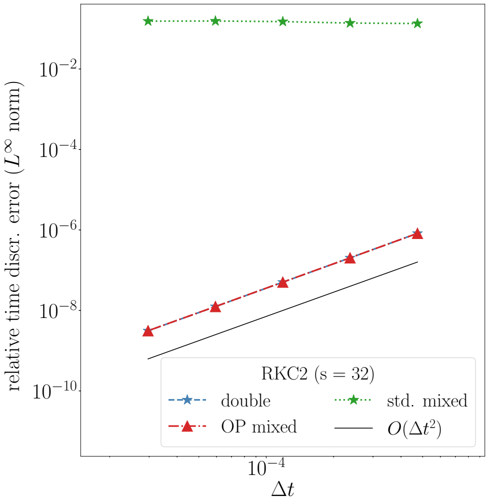


Numerical results - convergence

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

- Reduced-/mixed-precision algorithms require a careful implementation, but can bring significant memory, cost, and energy savings.
- SR makes computations very resilient to stagnation and error accumulation. If used correctly it can make reduced-precision computations very robust.
- We can make ESRK methods as accurate as their high precision equivalent and as cheap as their fully low-precision counterpart.
- Our work extends to multirate ESRK and to RK methods in general for $q = 1, 2$.

Future research directions

- Hyperbolic PDE solvers, multifidelity Monte Carlo methods, operator inference, numerical optimization...
- It would be nice to employ these techniques for engineering applications using hardware that actually supports reduced-precision computations.

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

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

References

[1] A. Abdelfattah, H. Anzt, E. G. Boman, E. Carson, 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.

[2] N. J. Higham and T. Mary. Mixed precision algorithms in numerical linear algebra, 2021. URL `http://eprints.maths.manchester.ac.uk/2841/1/paper_eprint.pdf`.

[3] M. Croci, M. Fasi, N. J. Higham, T. Mary, and M. Mikaitis. Stochastic rounding: implementation, error analysis and applications. *Royal Society Open Science*, 9:211631, 2022.

[4] 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.

[5] M. Croci and M. B. Giles. Effects of round-to-nearest and stochastic rounding in the numerical solution of the heat equation in low precision. *IMA Journal of Numerical Analysis (to appear)*, 2022. URL `https://arxiv.org/pdf/2010.16225`.

[6] M. Croci and G. R. de Souza. Mixed-precision explicit stabilized Runge-Kutta methods for single- and multi-scale differential equations, 2022. URL `https://arxiv.org/pdf/2109.12153`.

[7] M. P. Connolly, N. J. Higham, and T. Mary. Stochastic rounding and its probabilistic backward error analysis. *SIAM Journal on Scientific Computing*, 43(1):566–585, 2021.

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

APPENDIX

Exploit exact subtraction

How to best implement the matrix-vector product $-A\boldsymbol{U}^n$?

$$\frac{\boldsymbol{U}_{i+1}^n - 2\boldsymbol{U}_i^n + \boldsymbol{U}_{i-1}^n}{h^2}, \qquad \frac{(\boldsymbol{U}_{i+1}^n - \boldsymbol{U}_i^n) - (\boldsymbol{U}_i^n - \boldsymbol{U}_{i-1}^n)}{h^2}.$$

Exploit exact subtraction

How to best implement the matrix-vector product $-A\boldsymbol{U}^n$?

$$\frac{\boldsymbol{U}_{i+1}^n - 2\boldsymbol{U}_i^n + \boldsymbol{U}_{i-1}^n}{h^2}, \qquad \frac{(\boldsymbol{U}_{i+1}^n - \boldsymbol{U}_i^n) - (\boldsymbol{U}_i^n - \boldsymbol{U}_{i-1}^n)}{h^2}.$$

Leads to $O(h^{-2})$ error! **Leads to near-exact matvecs.**

A similar trick works for FEM as well. Only requires small modification of CSR matvecs.

Exploit exact subtraction

How to best implement the matrix-vector product $-A\boldsymbol{U}^n$?

$$\frac{\boldsymbol{U}^n_{i+1} - 2\boldsymbol{U}^n_i + \boldsymbol{U}^n_{i-1}}{h^2}, \qquad \frac{(\boldsymbol{U}^n_{i+1} - \boldsymbol{U}^n_i) - (\boldsymbol{U}^n_i - \boldsymbol{U}^n_{i-1})}{h^2}.$$

Leads to $O(h^{-2})$ error! **Leads to near-exact matvecs.**

A similar trick works for FEM as well. Only requires small modification of CSR matvecs.

Parts of a Theorem [C. and Giles 2020]

If $a, b \in \mathbb{R}$ are exactly represented in floating point arithmetic, and

$$|a-b| \leq \min(|a|, |b|)$$

then $(a-b)$ is computed exactly.

See also Section 2.5 in "Accuracy and Stability of Numerical Algorithms" by Nick Higham.

Computational savings

The mixed-precision scheme is cheaper by roughly a factor

$$\varrho = \frac{(s-q)(r-1)}{sr}, \quad \text{where} \quad r = \frac{\text{Cost of RHS evals in high}}{\text{Cost of RHS evals in low}}.$$

A scheme in double/half yields $r = 4^{\ell}$ for $O(N^{\ell})$-cost RHS evaluations.

- For RK4 and $\ell = 1$ this leads to 56% $(q = 1)$ and 40% $(q = 2)$ savings.
- 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 75% savings if $\ell = 1$.

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

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 $\hat{\boldsymbol{v}}_j = \hat{\boldsymbol{d}}_j - c_j\Delta t\boldsymbol{f}(\boldsymbol{y}^n)$, which is $O(\Delta t^2)$. We split $\Delta\boldsymbol{f}_j = \Delta_1\boldsymbol{f}_j + \Delta_2\boldsymbol{f}_j$, where

$$\begin{aligned}\Delta_1\boldsymbol{f}_j &= \boldsymbol{f}(\hat{\boldsymbol{y}}^n + c_j\Delta t\boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \hat{\boldsymbol{v}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}^n + c_j\Delta t\boldsymbol{f}(\hat{\boldsymbol{y}}^n)),\\ \Delta_2\boldsymbol{f}_j &= \boldsymbol{f}(\hat{\boldsymbol{y}}^n + c_j\Delta t\boldsymbol{f}(\hat{\boldsymbol{y}}^n)) - \boldsymbol{f}(\hat{\boldsymbol{y}}^n).\end{aligned}$$

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 $\hat{\boldsymbol{v}}_j = \hat{\boldsymbol{d}}_j - c_j\Delta t\boldsymbol{f}(\boldsymbol{y}^n)$, which is $O(\Delta t^2)$. We split $\Delta\boldsymbol{f}_j = \Delta_1\boldsymbol{f}_j + \Delta_2\boldsymbol{f}_j$, where

$$\begin{aligned}\Delta_1\boldsymbol{f}_j &= \boldsymbol{f}(\hat{\boldsymbol{y}}^n + c_j\Delta t\boldsymbol{f}(\hat{\boldsymbol{y}}^n) + \hat{\boldsymbol{v}}_j) - \boldsymbol{f}(\hat{\boldsymbol{y}}^n + c_j\Delta t\boldsymbol{f}(\hat{\boldsymbol{y}}^n)),\\ \Delta_2\boldsymbol{f}_j &= \boldsymbol{f}(\hat{\boldsymbol{y}}^n + c_j\Delta t\boldsymbol{f}(\hat{\boldsymbol{y}}^n)) - \boldsymbol{f}(\hat{\boldsymbol{y}}^n).\end{aligned}$$

It is sufficient to approximate both terms with $O(\Delta t^2)$ error. Again we use derivatives:

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

Note: various evaluation strategies available. We never need more than one high-precision evaluation of $\boldsymbol{f}$ and $\boldsymbol{f}'$ every s stages.

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 huge 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 huge 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 huge 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.

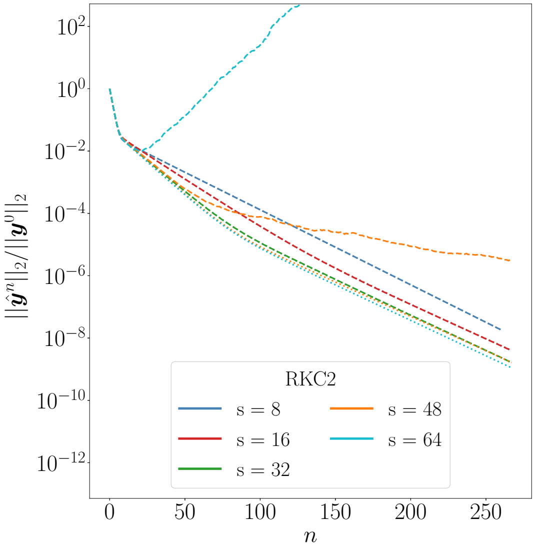
Solution: set $\hat{\Delta}\boldsymbol{f}_j = \begin{cases} \hat{\Delta}_1\boldsymbol{f}_j + \hat{\Delta}_2\boldsymbol{f}_j, & \text{if } \|\hat{\boldsymbol{v}}_j\|_2 \leq \|\hat{\boldsymbol{d}}_j\|_2, \\ \tilde{\Delta}\boldsymbol{f}_j, & \text{if } \|\hat{\boldsymbol{v}}_j\|_2 > \|\hat{\boldsymbol{d}}_j\|_2.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

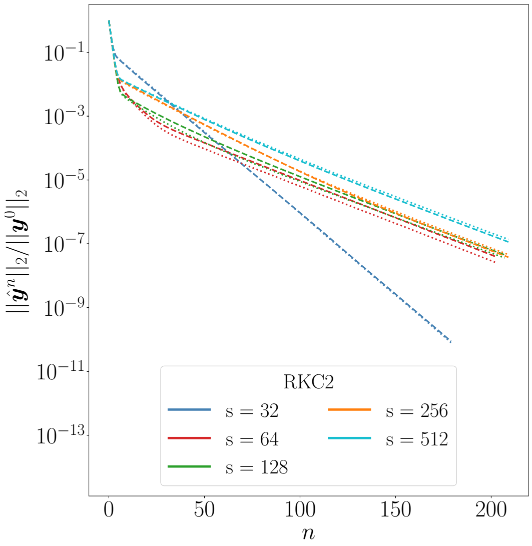
This leads to a 2nd-order *and* stable method.

Linear stability result - some comments

- The bounds account for rounding errors propagating from previous stages, an overlooked phenomenon in standard RKC theory [Verwer et al. 1990].
- No stability in the classical sense: our theory allows the error to grow for large Δt. We can prove no error growth under stringent conditions on $\kappa(A)$.
- The reason is that rounding errors destroy smoothness and act on all frequencies destroying any spectral relation between the iterates. This forbids any analysis based on eigenvalues or smoothness. The result is a very pessimistic bound.
- Our methods appear to be extremely stable in practice independently from $\kappa(A)$. They appear to be as accurate as the corresponding fully high-precision scheme.

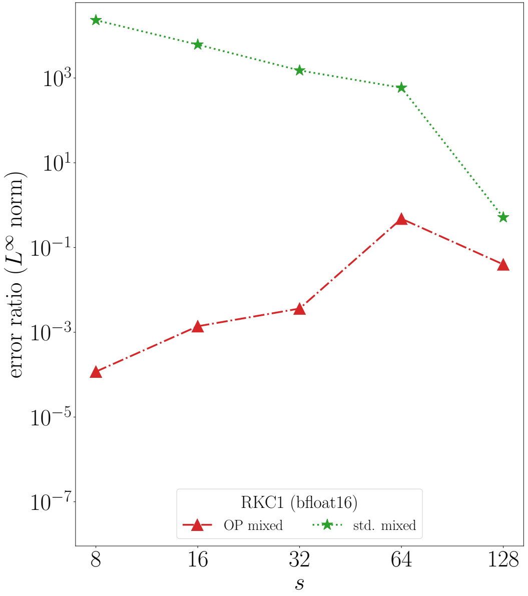
Numerical results - RKC2 stability (2D nonlinear heat eqn, half precision)

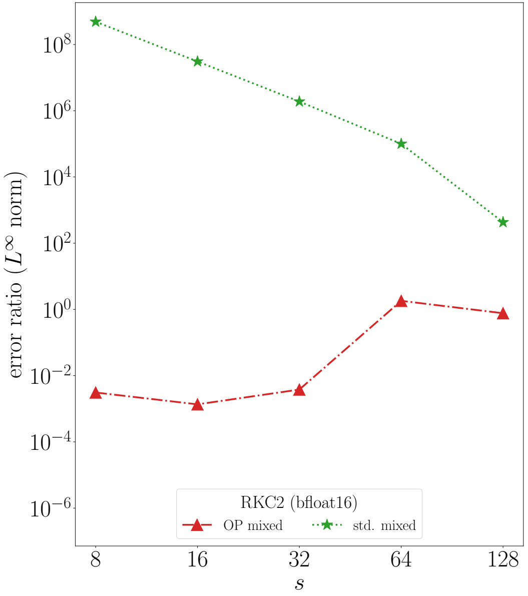




Behaviour of RKC2 numerical solution without (left) and with (right) stabilization.

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





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