Solving parabolic PDEs in half precision

UNIVERSITY OF OXFORD

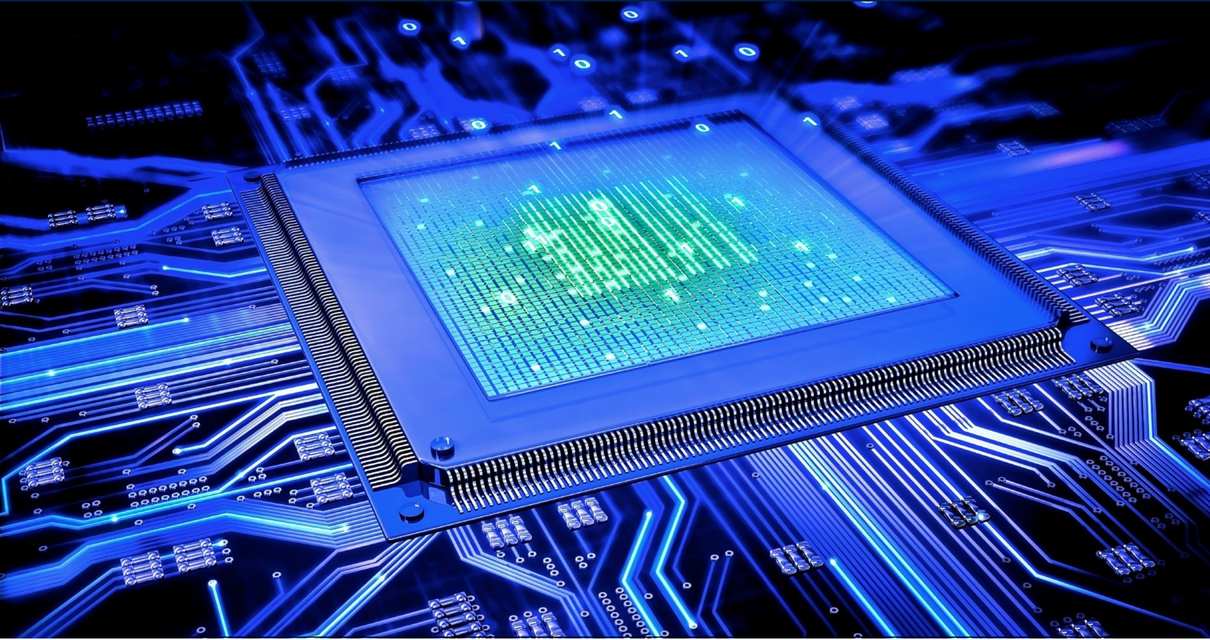
Mathematical Institute

Matteo Croci & Mike Giles
Mathematical Institute
University of Oxford

SIAM Conference on Computational Science and Engineering 2021

Oxford Mathematics

Objective: developing low-precision PDE solvers

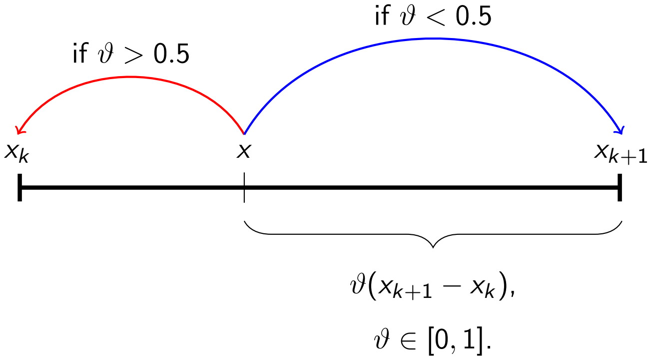


Overview

Background

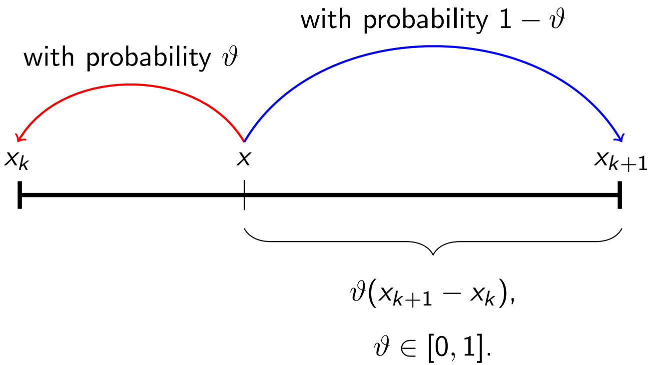
A 3-step guide to solving the heat equation in low precision

Round to nearest



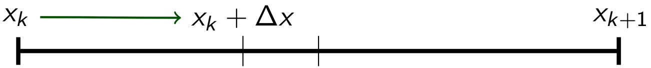
$$\text{fl}(x) = x(1 + \delta), \quad \text{with} \quad |\delta| \leq u.$$

Stochastic rounding

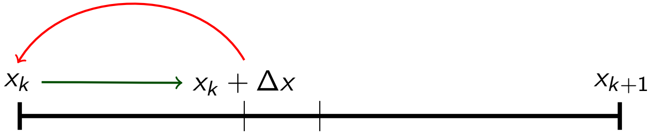


$$\text{sr}(x) = x(1+\delta(\omega)), \quad \text{with} \quad |\delta(\omega)| \leq 2u, \quad \text{and} \quad \mathbb{E}[sr(x)] = x.$$

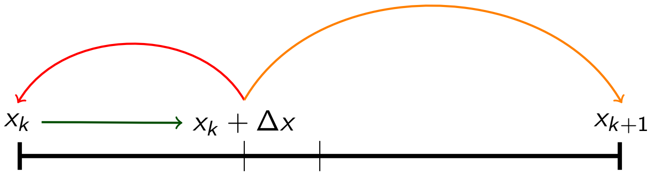
RtN might cause stagnation



RtN might cause stagnation



SR is resilient to stagnation



Heat equation with nonzero forcing

We consider the heat equation with non-zero forcing:

$$\begin{cases} \dot{u}(t,\boldsymbol{x}) = \nabla^2 u(t,\boldsymbol{x}) + f(t,\boldsymbol{x}), & \boldsymbol{x} \in D = [0,1]^d, \quad t > 0, \\ u(0,\boldsymbol{x}) = u_0(x), & \boldsymbol{x} \in D, \\ u(t,\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \partial D, \quad\quad\quad\quad t > 0. \end{cases}$$

We use finite differences in space and a Runge-Kutta method in time with discretisation parameters: Δt, h, $\lambda = \Delta t/h^2$.

Heat equation with nonzero forcing

We consider the heat equation with non-zero forcing:

$$\begin{cases} \dot{u}(t,\boldsymbol{x}) = \nabla^2 u(t,\boldsymbol{x}) + f(t,\boldsymbol{x}), & \boldsymbol{x} \in D = [0,1]^d, \quad t > 0, \\ u(0,\boldsymbol{x}) = u_0(x), & \boldsymbol{x} \in D, \\ u(t,\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \partial D, \quad\quad\quad\quad t > 0. \end{cases}$$

We use finite differences in space and a Runge-Kutta method in time with discretisation parameters: Δt, h, $\lambda = \Delta t/h^2$.

Let A be the (spd) stiffness matrix. The numerical scheme is:

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

for some matrix S dependent on ΔtA. For instance,

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

We work in **bfloat16 half precision**, $u = 2^{-8} \approx 4 \times 10^{-3}$.
Everything extends to FEM and linear parabolic equations.

Background

A 3-step guide to solving the heat equation in low precision

1) Local rounding errors and the delta form

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

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

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

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

1) Local rounding errors and the delta form

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. $S_{\mathrm{FE}} = (I - \Delta tA)$, $\tilde{S}_{FE} = 1$, and $S_{\mathrm{BE}} = \tilde{S}_{\mathrm{BE}} = (I + \Delta tA)^{-1}$.

- Errors in the computation of $S\boldsymbol{U}^{n}$ are of order u (machine precision).
- Errors in the computation of $\Delta\boldsymbol{U}^{n}$ are of order $\Delta t^{p}u$, $p \geq 0$.

We prove that:

- 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.

2) Exploit exact subtraction

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

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

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.

UNIVERSITY OF OXFORD

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]

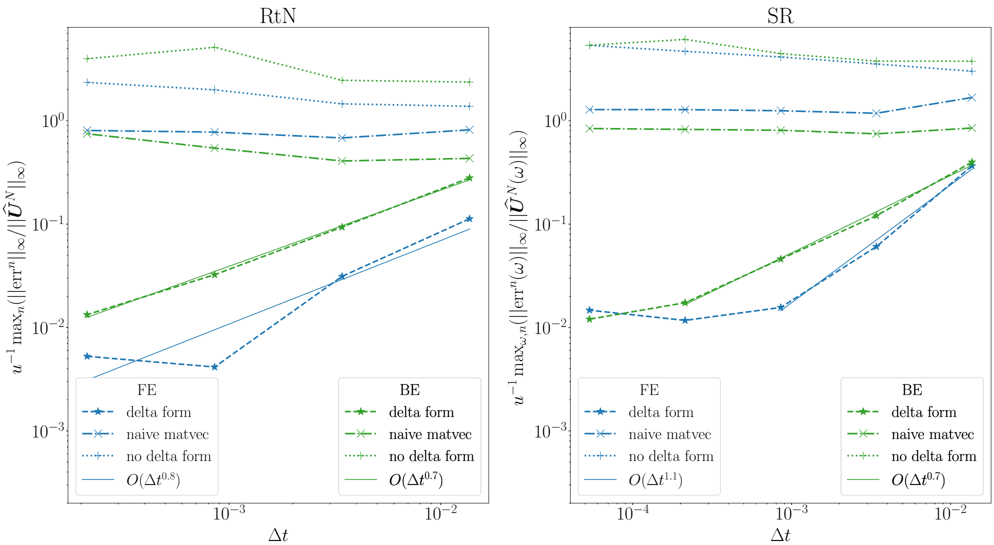
If $a, b \in \mathbb{R}$ are exactly represented in floating point arithmetic, and

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

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

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

Worst-case local rounding errors in 2D



Note: from now on we use the delta form with "smart" matvecs.

UNIVERSITY OF OXFORD

Why is RtN in low precision bad for parabolic equations?

a) Stagnation:

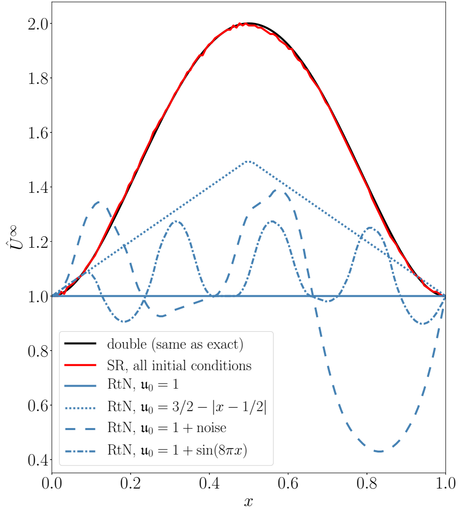
- RtN always stagnates for sufficiently small Δt (recall $\Delta \boldsymbol{U}^n = O(u\Delta t^p)$).
- The RtN solution is initial condition, discretization and precision dependent.

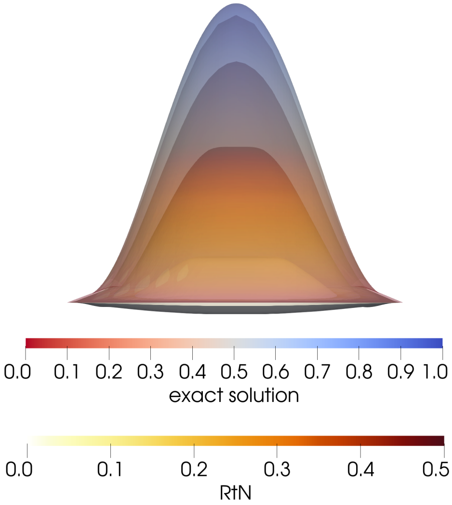
b) Global error:

- RtN rounding errors are strongly correlated and cannot be modelled as zero-mean independent random variables.
- RtN global errors grow like $O(u\Delta t^{-1})$ until stagnation.

SR fixes all these issues!

3a) Stagnation (left 1D, right 2D)





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

3b) 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.

We can distinguish two cases:

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

SR: the ε_i^n are zero-mean spatially independent and temporally mean-independent.

3b) 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.

We can distinguish two cases:

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

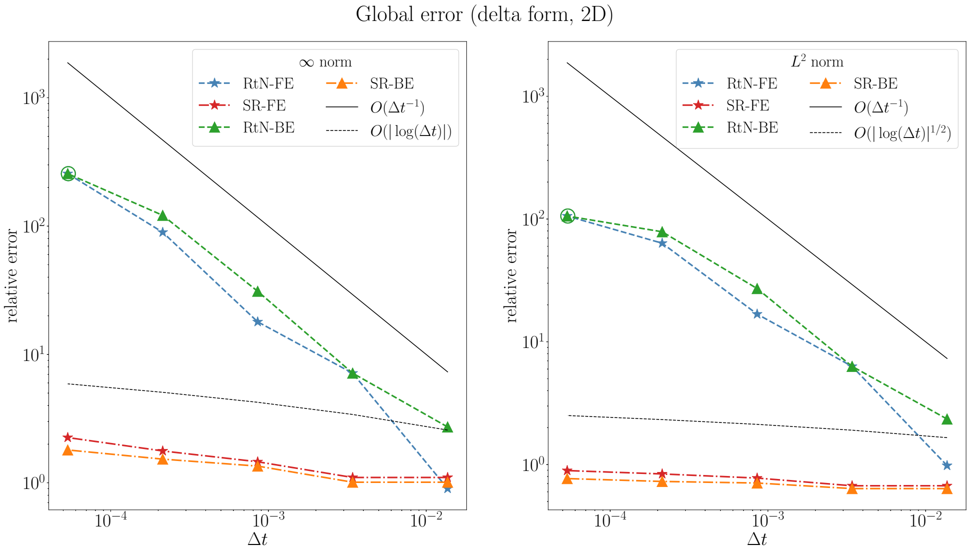
SR: the ε_i^n are zero-mean spatially independent and temporally mean-independent.

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

Asymptotic global rounding error blow-up rates; $\ell(\Delta t) = |\log(\lambda^{-1}\Delta t)|$.
Note that the RtN rates are well-known [Henrici 1962-1963, Jézéquel 1995].

Spatial independence of SR errors means more error cancellation in higher dimensions!

3b) Global rounding errors (here at steady-state)



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

Outlook

- Working in low precision can bring large speed, memory and energy consumption improvements. New hardware supports low-precision.
- SR might be an effective way of obtaining accurate results in much lower precision when solving time-dependent parabolic PDEs.
- Custom-built C++ low-precision emulator (`bitbucket.org/croci/libchopping/`) inspired by [Higham and Pranesh 2019] and Milan Kloewer's Julia routines (`github.com/milankl?tab=repositories`).

Current/future research directions

- Hyperbolic PDEs, stabilised explicit RK methods, nested multilevel Monte Carlo.
- Weather forecasting and brain simulation applications.

Thank you for listening!

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

[1] 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, 2020. URL `http://arxiv.org/abs/2010.16225`.

[2] M. P. Connolly, N. J. Higham, and T. Mary. Stochastic Rounding and its Probabilistic Backward Error Analysis, 2020. URL `https://hal.archives-ouvertes.fr/hal-02556997/document`.

[3] N. J. Higham and T. Mary. A new approach to probabilistic rounding error analysis. *SIAM Journal of Scientific Computing*, 41(5):2815–2835, 2019. doi: 10.1137/18M1226312.

[4] N. J. Higham and S. Pranesh. Simulating low precision floating-point arithmetic. *SIAM Journal on Scientific Computing*, 41(5):C585–C602, 2019. doi: 10.1137/19M1251308.

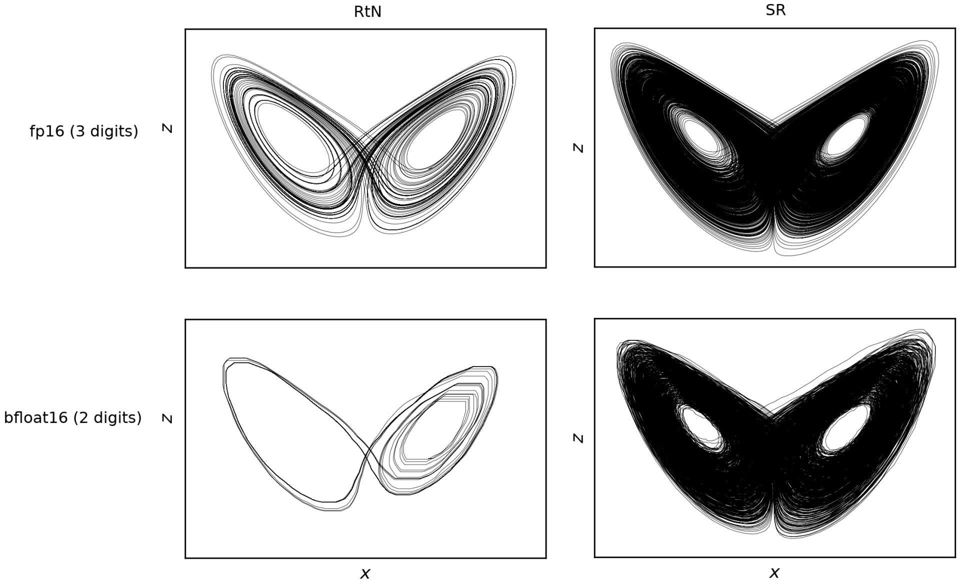
[5] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2002.

[6] F. Jézéquel. Round-off error propagation in the solution of the heat equation by finite differences. *Journal of Universal Computer Science*, 1(7):465–479, 1995.

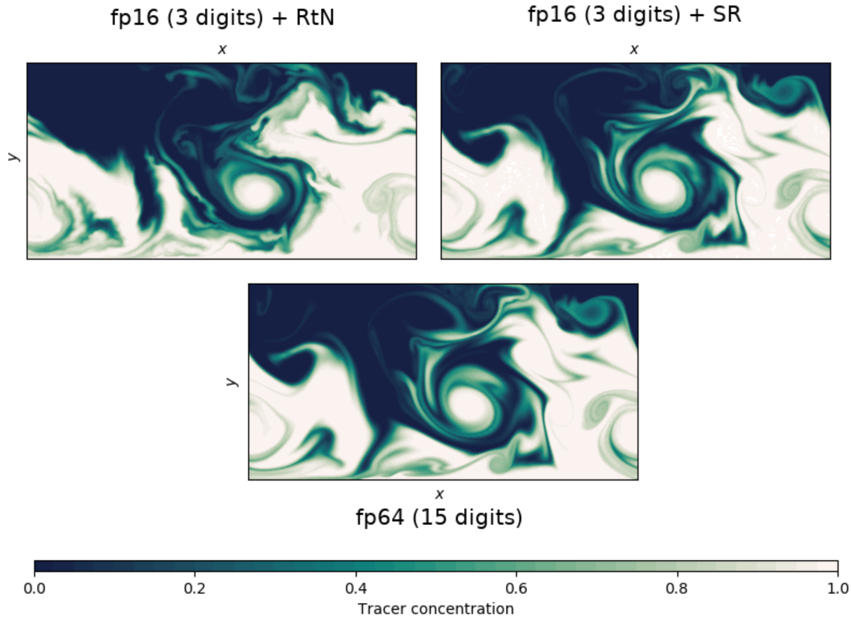
[7] M. Arató. Round-off error propagation in the integration of ordinary differential equations by one step methods. *Acta Scientiarium Mathematicarum*, 45:23–31, 1983. doi: 10.13140/2.1.3920.9608.

[8] P. Henrici. *Discrete Variable Methods in Ordinary Differential Equations*. John Wiley & Sons, Inc., 1962.

Promising results (by Milan Kloewer in Oxford Physics)



Prominsing results (by Milan Kloewer in Oxford Physics)



Stagnation $\mathrm{fl}(x+\epsilon) = x$ occurs whenever $\frac{u}{2}|x| \geq |\epsilon|$. For the PDE:

$$\frac{u}{2}|\mathfrak{u}(t_n, \boldsymbol{x_i})| \approx \frac{u}{2}|\hat{\boldsymbol{U}}_{\boldsymbol{i}}^n| \geq |\Delta \hat{\boldsymbol{U}}_{\boldsymbol{i}}^n| = |\hat{\boldsymbol{U}}_{\boldsymbol{i}}^{n+1} - \hat{\boldsymbol{U}}_{\boldsymbol{i}}^n| \approx \Delta t |\dot{\mathfrak{u}}(t_n, \boldsymbol{x_i})|,$$

This shows that $\hat{\boldsymbol{U}}_{\boldsymbol{i}}^n$ will not be updated whenever

$$|\mathfrak{u}(t_n, \boldsymbol{x_i})| \gtrapprox 2(\Delta t/u)|\dot{\mathfrak{u}}(t_n, \boldsymbol{x_i})|.$$

More formally,

Lemma [C. and Giles 2020]

Assume that the delta form is used and that $p > 0$. If there exists $\epsilon > 0$ such that $|\hat{\boldsymbol{U}}_{\boldsymbol{i}}^{\bar{n}}| \geq \epsilon$ for some $\boldsymbol{i}$, $\bar{n}$, then there exists $\tau(\epsilon) > 0$ such that if $\Delta t < \tau$, we have $\hat{\boldsymbol{U}}_{\boldsymbol{i}}^n = \hat{\boldsymbol{U}}_{\boldsymbol{i}}^{\bar{n}}$ for all $n \geq \bar{n}$.