UNIVERSITY OF OXFORD
Mathematical Institute

Order-preserving mixed-precision Runge-Kutta methods

M. Croci[†], G. R. de Souza[*]

[†] *Mathematical Institute, University of Oxford*
[*] *École Polytechnique Fédérale de Lausanne*

NA group internal seminar, 1 June 2021

Oxford Mathematics

Overview

Introduction and background

Linear problems

Nonlinear problems

Conclusions

Overview

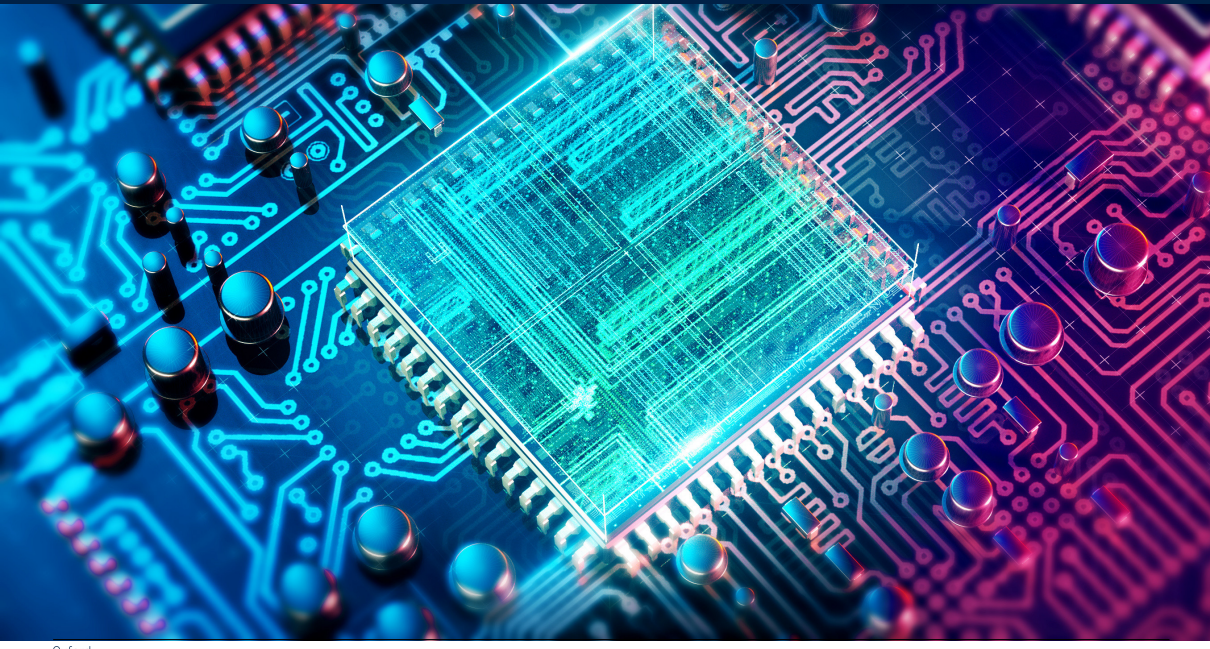
Introduction and background

Linear problems

Nonlinear problems

Conclusions

Objective: developing mixed-precision PDE solvers



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

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

where $\boldsymbol{g}(t, \boldsymbol{y})$ is Lipschitz continuous. In our experiments: MOL discretisation of a PDE.

Objective

Use as many low-precision RHS evaluations as possible without affecting accuracy or stability.

Why do we focus on explicit methods?

- The development of mixed-/reduced-precision linear/nonlinear iterative solvers is a very active field of research. Lots of exciting new work: [Abdelfattah et al. 2020].
- Avoiding nonlinear/linear solves is in general a big advantage. We use stabilised RK methods to minimise timestep restrictions.

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

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

where $\boldsymbol{g}(t, \boldsymbol{y})$ is Lipschitz continuous. In our experiments: MOL discretisation of a PDE.

Objective

Use as many low-precision RHS evaluations as possible without affecting accuracy or stability.

- Most performance gains in our algorithms come from reducing the precision of the linear term.
- Extensions to reduced-precision evaluations of $\boldsymbol{g}$ currently only possible under strong assumptions.
- Help and suggestions are welcome!

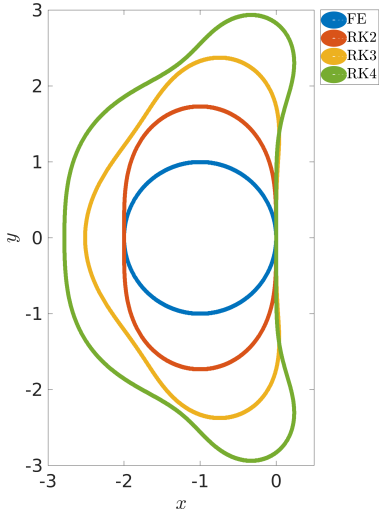
Floating point formats

Our mixed-precision algorithms combine high (double) and low (half) precision formats.

Format	Roundoff unit 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}

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 stabilised RK methods

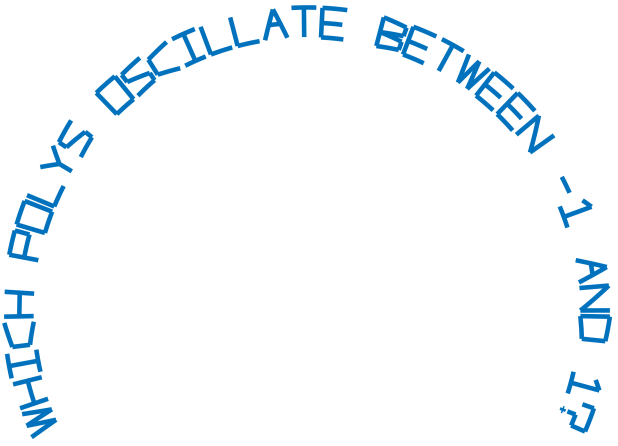
Idea: pick the poly $R_s(z)$ so as to maximise the stability region. Two approaches:

1. Maximise the real stability region (good for parabolic problems).
2. Maximise the region in which the method is TVD (good for hyperbolic problems).

Explicit stabilised RK methods

Idea: pick the poly $R_s(z)$ so as to maximise the stability region. Two approaches:

1. Maximise the real stability region (good for parabolic problems).
2. Maximise the region in which the method is TVD (good for hyperbolic problems).



Idea: pick the poly $R_s(z)$ so as to maximise the stability region. Two approaches:

1. Maximise the real stability region (good for parabolic problems).
2. Maximise the region in which the method is TVD (good for hyperbolic problems).

Idea: pick the poly $R_s(z)$ so as to maximise the stability region. Two approaches:

1. Maximise the real stability region: use orthogonal polys $\rightsquigarrow$ RKC, RKL, RKG, etc..
2. Maximise the region in which the method is TVD $\rightsquigarrow$ SSPRK.

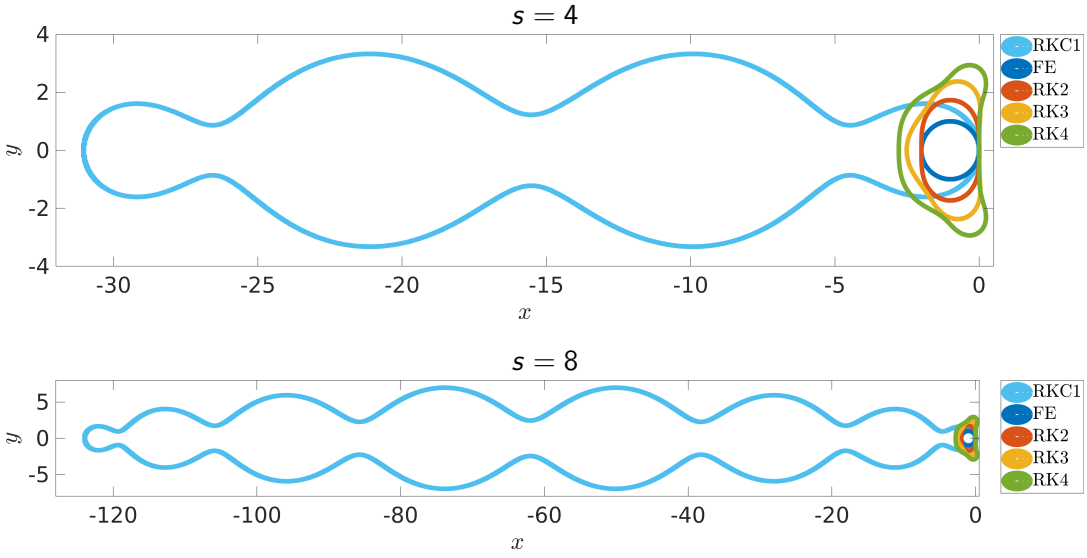
The largest possible region is $O(s^2)$ and is obtained by taking $R_s(z) = T_s\left(1 + \frac{z}{s^2}\right)$. Other orthogonal polynomials are also good choices (e.g. Legendre, Gegenbauer).

These 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!

As a curiosity: the stages of RKC methods are implemented via three-term recurrences:

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

Absolute stability of RKC methods (RKC1)



Overview

Introduction and background

Linear problems

Nonlinear problems

Conclusions

Linear problems, i.e. $\boldsymbol{g}(t, \boldsymbol{y}) = \boldsymbol{g} = \text{const}$

Consider the exact solution at $t = \Delta t$ and its corresponding p-th order RK approximation (take $\boldsymbol{g} = 0$ for simplicity):

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

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

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

Linear problems, i.e. $\boldsymbol{g}(t, \boldsymbol{y}) = \boldsymbol{g} = \text{const}$

Consider the exact solution at $t = \Delta t$ and its corresponding p-th order RK approximation (take $\boldsymbol{g} = 0$ for simplicity):

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

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

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:

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

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

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

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!

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

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.

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

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!

From now on we set u to be the roundoff unit 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 to obtain a q-order-preserving mixed-precision RK method.

Linear stability and convergence

In the linear case, we can easily obtain a q-order-preserving method by performing all vector operations, and only q matvecs in high precision.

In the linear case, we can easily obtain a q-order-preserving method by performing all vector operations, and only q matvecs in high precision.

Let $z = \Delta t||A||_2$, $||\boldsymbol{y}_0||_2 \leq 1$. We then have

$$||\hat{R}_s - R_s||_2 \leq c_m(s-q)u \sum_{j=q+1}^{s} \frac{z^j}{j!} = O(uz^{q+1}).$$

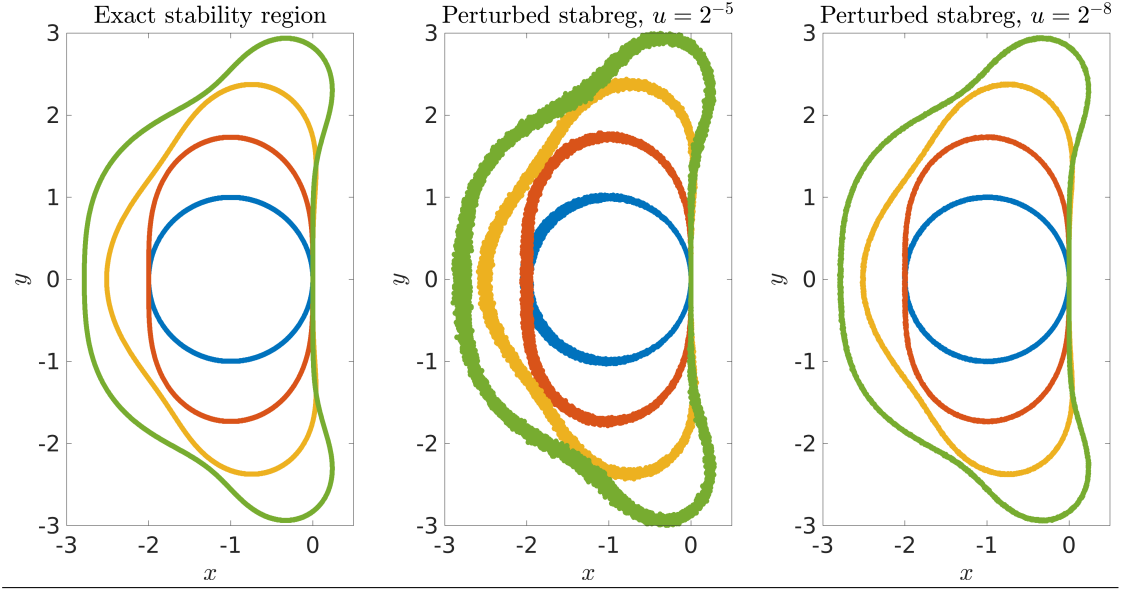
In the linear case, we can easily obtain a q-order-preserving method by performing all vector operations, and only q matvecs in high precision.

Let $z = \Delta t||A||_2$, $||\boldsymbol{y}_0||_2 \leq 1$. We then have

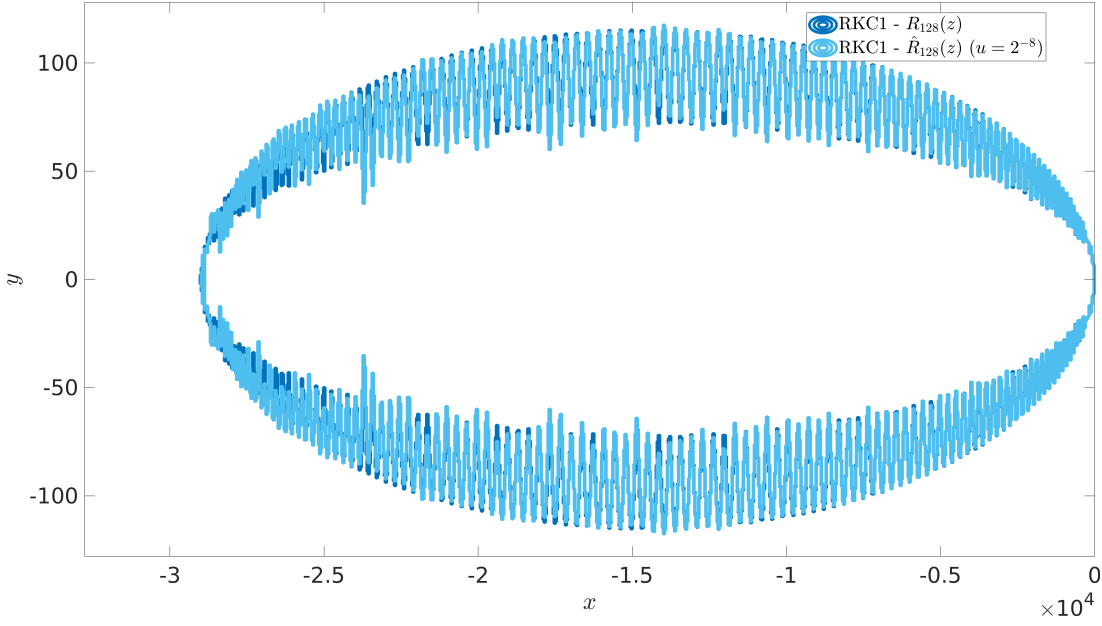
$$||\hat{R}_s - R_s||_2 \leq c_m(s-q)u \sum_{j=q+1}^{s} \frac{z^j}{j!} = O(uz^{q+1}).$$

- This ensures stability as long as the method has a "small" stability region and/or Δt is small enough.
- For stabilised methods we need a better bound since $z = O(s^2)$, but we can still prove stability if A is non-singular. In practice the methods are always stable for singular A and for all s if $q = 1$ and for $s \leq \frac{1}{\sqrt{u}}$ for $q = 2$.
- $\hat{R}_s(z) = R_s(z) + O(uz^{q+1}) \Rightarrow$ order q method. Comparable/smaller error constant.

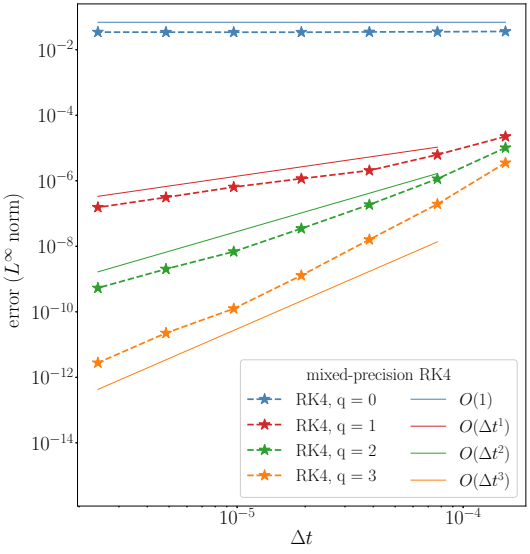
Linear stability for RK methods (in practice)



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



Linear problems - convergence (3D heat eqn)



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

Overview

Introduction and background

Linear problems

Nonlinear problems

Conclusions

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

When $\boldsymbol{g} \neq$ const, we can still construct q-order-preserving mixed-precision RK methods for $q \leq 2$ under one of the following assumptions/restrictions:

(H) - $\boldsymbol{g}$ is cheap to evaluate wrt $A\boldsymbol{y}$.

(H) - $\boldsymbol{g}$ is non-stiff/much less stiff than $A\boldsymbol{y}$ $\rightsquigarrow$ use multirate schemes.

(L) - Lipschitz continuity of $\boldsymbol{g}$ can be made to hold in low precision:

$$||\mathrm{fl}\left(\boldsymbol{g}(t + \Delta t, \boldsymbol{y} + \Delta \boldsymbol{y}) - \boldsymbol{g}(t, \boldsymbol{y})\right)|| = O(\Delta t + ||\Delta \boldsymbol{y}||) + \varepsilon.$$

Want ε to be $O(\Delta t + ||\Delta \boldsymbol{y}||)$. Examples:

- Analyitic representation of differences,
  e.g. $(\boldsymbol{u} + \delta \boldsymbol{u})\nabla(\boldsymbol{u} + \delta \boldsymbol{u}) - \boldsymbol{u}\nabla\boldsymbol{u} = \delta\boldsymbol{u}\nabla\boldsymbol{u} + \boldsymbol{u}\nabla\delta\boldsymbol{u} + \delta\boldsymbol{u}\nabla\delta\boldsymbol{u} = O(\delta\boldsymbol{u})$.
- $\boldsymbol{g}$ acts entrywise on $\boldsymbol{y}$, e.g. reaction terms.

A generic RK method in Butcher form reads:

$$\boldsymbol{y}^{n+1} = \boldsymbol{y}^n + \Delta \boldsymbol{y}^n = \boldsymbol{y}^n + \sum_{i=1}^{s} b_i \boldsymbol{k}_i,$$

$$\boldsymbol{k}_i = \Delta t \boldsymbol{f}\left(t^n + c_i \Delta t, \boldsymbol{y}^n + \sum_{j=1}^{i-1} a_{ij} \boldsymbol{k}_j \right), \quad i = 1, \ldots, s.$$

In our specific case, this becomes:

$$
\begin{aligned}
\boldsymbol{y}^{n+1} &= \boldsymbol{y}^n + \Delta \boldsymbol{y}^n = \boldsymbol{y}^n + \sum_{i=1}^{s} b_i \boldsymbol{k}_i, \\
\boldsymbol{k}_1 &= \Delta t A \boldsymbol{y}^n + \Delta t \boldsymbol{g}(t^n, \boldsymbol{y}^n), \\
\boldsymbol{k}_i &= \Delta t A \boldsymbol{y}^n + \Delta t A \sum_{j=1}^{i-1} a_{ij} \boldsymbol{k}_j + \Delta t \boldsymbol{g}^i, \quad i = 2, \ldots, s. \\
\boldsymbol{g}^i &= \boldsymbol{g}\left(t^n + c_i \Delta t,\ \boldsymbol{y}^n + \sum_{j=1}^{i-1} a_{ij} \boldsymbol{k}_j \right).
\end{aligned}
$$

For a 1-order-preserving method we need to make sure all rounding errors are $O(\Delta t^2)$.

In our specific case, this becomes:

$$
\begin{aligned}
\boldsymbol{y}^{n+1} &= \boldsymbol{y}^n + \Delta \boldsymbol{y}^n = \boldsymbol{y}^n + \sum_{i=1}^{s} b_i \boldsymbol{k}_i, \\
\boldsymbol{k}_1 &= \Delta t A \boldsymbol{y}^n + \Delta t \boldsymbol{g}(t^n, \boldsymbol{y}^n), \\
\boldsymbol{k}_i &= \Delta t A \boldsymbol{y}^n + \Delta t \hat{A} \sum_{j=1}^{i-1} a_{ij} \boldsymbol{k}_j + \Delta t \boldsymbol{g}^i, \quad i = 2, \dots, s. \\
\boldsymbol{g}^i &= \boldsymbol{g}\left(t^n + c_i \Delta t,\ \boldsymbol{y}^n + \sum_{j=1}^{i-1} a_{ij} \boldsymbol{k}_j \right).
\end{aligned}
$$

For a 1-order-preserving method we need to make sure all rounding errors are $O(\Delta t^2)$.

$\Rightarrow$ If $\boldsymbol{g}$ is cheap to evaluate compute all orange terms exactly.

What if $\boldsymbol{g}$ is expensive? Rewrite:

$$\boldsymbol{y}^{n+1} = \boldsymbol{y}^n + \Delta \boldsymbol{y}^n = \boldsymbol{y}^n + \sum_{i=1}^{s} b_i \boldsymbol{k}_i,$$

$$\boldsymbol{k}_1 = \Delta t A \boldsymbol{y}^n + \Delta t \boldsymbol{g}(t^n, \boldsymbol{y}^n),$$

$$\boldsymbol{k}_i = \boldsymbol{k}_1 + \Delta t \hat{A} \sum_{j=1}^{i-1} a_{ij} \boldsymbol{k}_j + \Delta t \widehat{\Delta \boldsymbol{g}^i}, \quad i = 2, \ldots, s.$$

$$\Delta \boldsymbol{g}^i = \boldsymbol{g}\left(t^n + c_i \Delta t,\ \boldsymbol{y}^n + \sum_{j=1}^{i-1} a_{ij} \boldsymbol{k}_j \right) - \boldsymbol{g}(t^n,\ \boldsymbol{y}^n) = O(\Delta t)$$

For a 1-order-preserving method we need to make sure all rounding errors are $O(\Delta t^2)$.

$\Rightarrow$ Now the $\Delta \boldsymbol{g}^i$ are $O(\Delta t)$. Provided that they **stay $\boldsymbol{O(\Delta t)}$ in low precision**, we can evaluate them in low. This can be done in the previously mentioned cases.

What about second order? Set $\tilde{\boldsymbol{k}}_i = \boldsymbol{k}_i - \boldsymbol{k}_1$, $\boldsymbol{k}_1 = \Delta t A \boldsymbol{y}^n + \Delta t \boldsymbol{g}(t^n, \boldsymbol{y}^n)$. Then,

$$\boldsymbol{y}^{n+1} = \boldsymbol{y}^n + \boldsymbol{k}_1 + \sum_{i=2}^{s} b_i \tilde{\boldsymbol{k}}_i,$$

$$\tilde{\boldsymbol{k}}_i = \Delta t \hat{A} \sum_{j=2}^{i-1} a_{ij} \tilde{\boldsymbol{k}}_j + c_i \Delta t A \boldsymbol{k}_1 + \Delta t \Delta \boldsymbol{g}^i, \quad i = 1, \dots, s.$$

$$\Delta \boldsymbol{g}^i = \boldsymbol{g}\left(t^n + c_i \Delta t,\ \boldsymbol{y}^n + c_i \boldsymbol{k}_1 + \sum_{j=2}^{i-1} a_{ij} \tilde{\boldsymbol{k}}_j \right) - \boldsymbol{g}(t^n, \boldsymbol{y}^n) = O(\Delta t)$$

For a 2-order-preserving method we need to make sure all rounding errors are $O(\Delta t^3)$.

$\Rightarrow$ If $\boldsymbol{g}$ is cheap to evaluate compute all orange terms exactly.

What if $\boldsymbol{g}$ is expensive? Then,

$$\boldsymbol{y}^{n+1} = \boldsymbol{y}^n + \boldsymbol{k}_1 + \sum_{i=2}^{s} b_i \tilde{\boldsymbol{k}}_i,$$

$$\tilde{\boldsymbol{k}}_i = \Delta t \hat{A} \sum_{j=2}^{i-1} a_{ij} \tilde{\boldsymbol{k}}_j + c_i \Delta t A \boldsymbol{k}_1 + \Delta t \widehat{\Delta_1 \boldsymbol{g}^i} + \Delta t \Delta_2 \boldsymbol{g}^i, \quad i = 1, \dots, s.$$

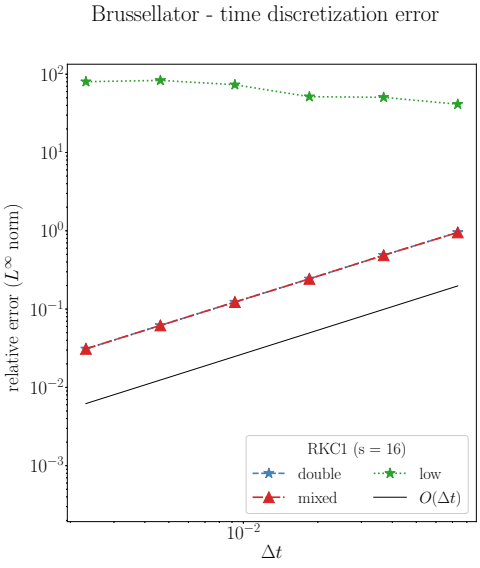
$$\Delta_1 \boldsymbol{g}^i = \boldsymbol{g}\left(\boldsymbol{y}^n + c_i \boldsymbol{k}_1 + \sum_{j=2}^{i-1} a_{ij} \tilde{\boldsymbol{k}}_j\right) - \boldsymbol{g}(\boldsymbol{y}^n + c_i \boldsymbol{k}_1) = O(\Delta t^2)$$

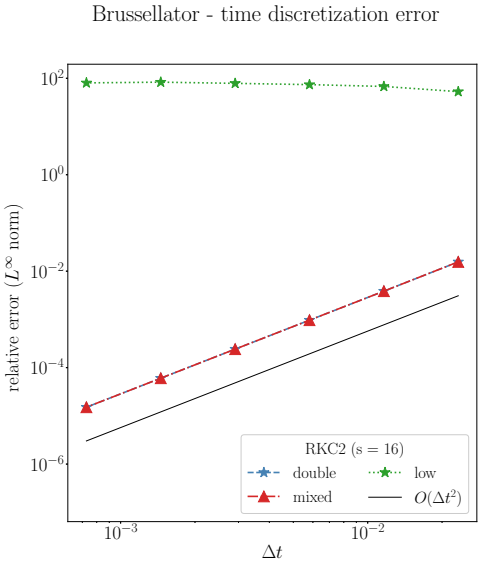
$$\Delta_2 \boldsymbol{g}^i = \boldsymbol{g}(\boldsymbol{y}^n + c_i \boldsymbol{k}_1) - \boldsymbol{g}(\boldsymbol{y}^n) = O(\Delta t)$$

For a 2-order-preserving method we need to make sure all rounding errors are $O(\Delta t^3)$.

$\Rightarrow$ Now the $\Delta_1 \boldsymbol{g}^i$ are $O(\Delta t^2)$. Provided that they **stay $\boldsymbol{O(\Delta t^2)}$ in low precision**, we can evaluate them in low. This can be done in the previously mentioned cases.

Numerical results - convergence





Overview

Introduction and background

Linear problems

Nonlinear problems

Conclusions

Outlook

- A naïve mixed-precision implementation can harm convergence.
- We can construct mixed-precision methods that will retain at least 1^{st} or 2^{nd} order convergence and already reduce the overall error by orders of magnitude.
- It might be possible to extend to $q > 2$. However, handling the nonlinear terms becomes increasingly tricky and we have not found a solution yet.
- For $O(N)$ cost RHS-evaluations we save between $40 - 60\%$ of the cost of standard RK. Even more for $O(N^2)$ evals or if we accout for memory/caching-related costs.
- We can make ESRK methods as cheap as their low-precision counterpart.

Current/future research directions

- Finish analysis and paper write-up.
- Extensions to SSPRK methods.
- Larger q, generic nonlinear terms?

Thank you for listening!

UNIVERSITY OF OXFORD

References

[1] M. Croci and G. R. de Souza. Order-preserving mixed-precision Runge-Kutta methods, In preparation (2021).

[2] A. Abdelfattah, H. Anzt, E. G. Boman, E. Carson, T. Cojean, J. Dongarra, M. Gates, T. Grützmacher, N. J. Higham, S. Li, et al. A survey of numerical methods utilizing mixed precision arithmetic. Technical report, 2020.

[3] N. J. Higham. *Functions of Matrices - Theory and Computation*. SIAM, 2008.

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

[5] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II*, volume 14. Springer-Verlag Berlin Heidelberg, 1996.

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

[7] 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.

Thanks a lot NA Group for these amazing 7 years!

APPENDIX

The mixed-precision scheme is cheaper by roughly a factor

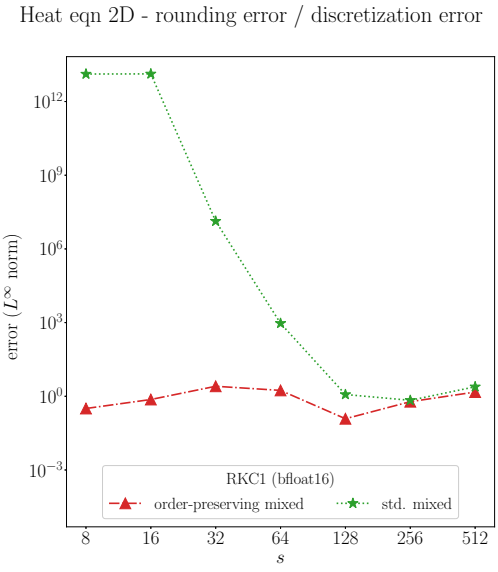
$$\varrho = \frac{(s-q)(r-1)}{sr}, \quad \text{where} \quad r = \frac{\text{Cost of RHS evals in high}}{\text{Cost of RHS evals in low}}.$$

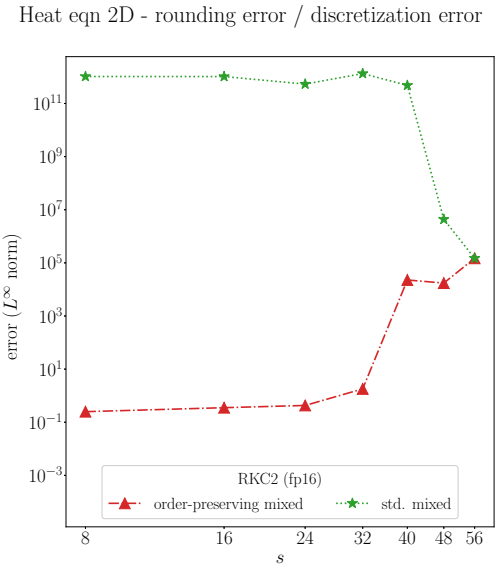
A scheme in double/half yields $r = 4^\ell$ for $O(N^\ell)$-cost RHS evaluations.

- 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.

Numerical results - error vs number of stages





Numerical results - error vs number of stages