An overview of mixed-precision methods in scientific computing

Matteo Croci

Center for Optimization and Statistical Learning Seminar.
Northwestern University, 6 October 2022
Overview

1. Introduction and background

2. Optimization

3. Numerical linear algebra

4. Numerical solution of partial differential equations

5. Conclusions

**Note:** too broad a field to include everything. I will present a few examples per topic.
1. Introduction and background

Main references:


Reduced- and mixed-precision algorithms
### 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.

---

1Review 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 and machine learning communities.
- Many new RP/MP algorithms in scientific computing and data science.
- 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

<table>
<thead>
<tr>
<th>Format</th>
<th>unit roundoff $u$</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>bfloat16 (half)</td>
<td>$2^{-8}$</td>
<td>$10^{38}$</td>
</tr>
<tr>
<td>fp16 (half)</td>
<td>$2^{-11}$</td>
<td>$10^{38}$</td>
</tr>
<tr>
<td>fp32 (single)</td>
<td>$2^{-24}$</td>
<td>$10^{38}$</td>
</tr>
<tr>
<td>fp64 (double)</td>
<td>$2^{-53}$</td>
<td>$10^{308}$</td>
</tr>
</tbody>
</table>

**Important:** don’t just focus on $u$, range is an extremely important factor. Scaling and squeezing techniques are central for a correct reduced-precision implementation.

**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.

**Half vs double max speedups:** $\times 4$ on CPUs, $\times 32$ on A100 NVIDIA GPUs.
Round to nearest

\[ \theta \in [0, 1]. \]

\[ \text{fl}(x) = x(1 + \delta), \quad \text{with} \quad |\delta| \leq u. \]
Stochastic rounding (review article [C. et al. 2022])

\[
\vartheta (x_{k+1} - x_k),
\]

\[
\vartheta \in [0, 1].
\]

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

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

Note: Not my field of expertise. Post-seminar discussions are welcome!

Main references:


- F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In Fifteenth annual conference of the international speech communication association. Microsoft, 2014


Reduced- and mixed-precision first-order methods in machine learning

- The machine learning community has been the main driver of experimentation in this field and GPU tensor cores really help making this efficient.

Reduced- and mixed-precision first-order methods in machine learning

- The machine learning community has been the main driver of experimentation in this field and GPU tensor cores really help making this efficient.
- “Easy” to implement: a single line of code allows to switch to single/half mixed-precision in TensorFlow.

Reduced- and mixed-precision first-order methods in machine learning

- The machine learning community has been the main driver of experimentation in this field and GPU tensor cores really help making this efficient.
- “Easy” to implement: a single line of code allows to switch to single/half mixed-precision in TensorFlow.
- Stochastic rounding has been successfully employed to squeeze stochastic gradient descent into quarter precision, see [Mellempudi et al. 2019].

Reduced- and mixed-precision first-order methods in machine learning

- The machine learning community has been the main driver of experimentation in this field and GPU tensor cores really help making this efficient.
- “Easy” to implement: a single line of code allows to switch to single/half mixed-precision in TensorFlow.
- Stochastic rounding has been successfully employed to squeeze stochastic gradient descent into quarter precision, see [Mellempudi et al. 2019].
- 1-bit Precision has been employed in sign gradient descent, cf. [Seide et al. 2014].

Reduced- and mixed-precision first-order methods in machine learning

- The machine learning community has been the main driver of experimentation in this field and GPU tensor cores really help making this efficient.
- “Easy” to implement: a single line of code allows to switch to single/half mixed-precision in TensorFlow.
- Stochastic rounding has been successfully employed to squeeze stochastic gradient descent into quarter precision, see [Mellempudi et al. 2019].
- 1-bit Precision has been employed in sign gradient descent, cf. [Seide et al. 2014].
- From a theoretical point of view: many open questions.

Reduced-precision second-order optimization \(\subseteq\) optimization with noise?

Limited results in the optimization literature are specific to rounding errors. However, there is work on optimization with noise (see e.g. [Xie, Byrd & Nocedal 2020]).

What I'd be curious to know:

To what extent does the existing theory apply to inexact arithmetic? What are the implementation challenges?

Need to consider:

- Noisy function and derivative evaluations:
  \[
  \hat{f}(x) = f(x) + \varepsilon_f(x), \quad |\varepsilon_f(x)| \leq \varepsilon_0, \quad \forall x.
  \]
  \[
  \hat{\nabla}_i f(x) = \nabla_i f(x) + \varepsilon_i(x), \quad \|\varepsilon_i(x)\| \leq \varepsilon_i, \quad \forall x, i = 1, 2.
  \]

- Inexact Newton system solves, linesearch, local models, subproblems, ...

How does the relative size of the errors affect convergence? Which steps can I perform more or less accurately?
Reduced-precision second-order optimization $\subseteq$ optimization with noise?

Limited results in the optimization literature are specific to rounding errors. However, there is work on optimization with noise (see e.g. [Xie, Byrd & Nocedal 2020]).

**What I’d be curious to know:** To what extent does the existing theory apply to inexact arithmetic? What are the implementation challenges?
Reduced-precision second-order optimization ⊆ optimization with noise?

Limited results in the optimization literature are specific to rounding errors. However, there is work on optimization with noise (see e.g. [Xie, Byrd & Nocedal 2020]).

What I’d be curious to know: To what extent does the existing theory apply to inexact arithmetic? What are the implementation challenges?

Need to consider:

• Noisy function and derivative evaluations:

\[ \hat{f}(x) = f(x) + \varepsilon f(x), \quad \text{with} \quad |\varepsilon_0(x)| \leq \varepsilon_0, \forall x. \]

\[ \hat{\nabla}^i f(x) = \nabla^i f(x) + \varepsilon_i(x), \quad \text{with} \quad ||\varepsilon_i(x)|| \leq \varepsilon_i, \forall x, \ i = 1, 2. \]

• Inexact Newton system solves, linesearch, local models, subproblems, ...

How does the relative size of the errors affect convergence?
Which steps can I perform more or less accurately?
Stochastic rounding may be useful if theory assumes zero-mean independent errors.
Designing mixed-precision optimization methods - some thoughts

- Stochastic rounding may be useful if theory assumes zero-mean independent errors.
- Designing routines for reduced-/mixed-precision derivative evaluations may be problem-dependent and not straightforward in general.
• Stochastic rounding may be useful if theory assumes zero-mean independent errors.

• Designing routines for reduced-/mixed-precision derivative evaluations may be problem-dependent and not straightforward in general.

• Barring underflow/overflow rounding errors are typically linear in $u$ so noise constants are easy to estimate if evaluation routines are type-flexible.
Designing mixed-precision optimization methods - some thoughts

- Stochastic rounding may be useful if theory assumes zero-mean independent errors.
- Designing routines for reduced-/mixed-precision derivative evaluations may be problem-dependent and not straightforward in general.
- Barring underflow/overflow rounding errors are typically linear in $u$ so noise constants are easy to estimate if evaluation routines are type-flexible.
- Mixed-precision NLA methods can be applied and incorporated, e.g. in Newton linear system solves, quasi-Newton updates, trust-region subproblems, ...
Newton’s method in floating-point arithmetic [Tisseur 2001], [Kelley 2022]

Results from [Kelley 2022]. Analysis for “vanilla” Newton: no linesearch.

Assumptions:
2. A backward error bound holds for linear solves (e.g. LU factorization is used).

Newton step:
\[
\hat{x}_{k+1} = \hat{x}_k - (\nabla^2 f(\hat{x}_k) + \epsilon_2 \hat{x}_k + \epsilon_s \hat{x}_k) - 1 (\nabla f(\hat{x}_k) + \epsilon_1 \hat{x}_k) + \epsilon_a \hat{x}_k,
\]

\[\|\epsilon_1 \hat{x}\| \leq \epsilon_1, \quad \text{(gradient error)}\]
\[\|\epsilon_2 \hat{x}\| \leq \epsilon_2, \quad \text{(Hessian error)}\]
\[\|\epsilon_a \hat{x}\| \leq \epsilon_a, \quad \text{(update error)}\]
\[\|\epsilon_s \hat{x}\| \leq \epsilon_s, \quad \text{(linear solve error)}\]

\(\forall \hat{x}\).

Theorem (Kelley 2022)
Under the above assumptions, the error \(e_k = \hat{x}_k - x^*\) satisfies
\[
\|e_{k+1}\| = O(\|e_k\|^2 + (\epsilon_2 + \epsilon_s)\|e_k\| + \epsilon_1 + \epsilon_a).
\]
Newton's method in floating-point arithmetic [Tisseur 2001], [Kelley 2022]

Results from [Kelley 2022]. Analysis for “vanilla” Newton: no linesearch.

Assumptions:

2. A backward error bound holds for linear solves (e.g. LU factorization is used).
Newton’s method in floating-point arithmetic [Tisseur 2001], [Kelley 2022]

Results from [Kelley 2022]. Analysis for “vanilla” Newton: no linesearch.

**Assumptions:**

2. A backward error bound holds for linear solves (e.g. LU factorization is used).

**Newton step:**
Newton’s method in floating-point arithmetic [Tisseur 2001], [Kelley 2022]

Results from [Kelley 2022]. Analysis for “vanilla” Newton: no linesearch.

Assumptions:
2. A backward error bound holds for linear solves (e.g. LU factorization is used).

Newton step:

\[
\hat{x}_{k+1} = \hat{x}_k - (\nabla^2 f(\hat{x}_k) + \varepsilon_2(\hat{x}_k) + \varepsilon_s(\hat{x}_k))^{-1}(\nabla f(\hat{x}_k) + \varepsilon_1(\hat{x}_k)) + \varepsilon_a(\hat{x}_k),
\]

\[
\|\varepsilon_1(x)\| \leq \varepsilon_1, \quad \text{gradient error}, \quad \|\varepsilon_2(x)\| \leq \varepsilon_2, \quad \text{(Hessian error),} \quad \forall x,
\]

\[
\|\varepsilon_a(x)\| \leq \varepsilon_a, \quad \text{(update error)}, \quad \|\varepsilon_s(x)\| \leq \varepsilon_s, \quad \text{(linear solve error),} \quad \forall x.
\]

Theorem (Kelley 2022)

Under the above assumptions, the error \( e_k = \hat{x}_k - x^* \) satisfies

\[
\|e_{k+1}\| = O \left( \|e_k\|^2 + (\varepsilon_2 + \varepsilon_s)\|e_k\| + \varepsilon_1 + \varepsilon_a \right)
\]
Mixed-precision Newton

Theorem (Kelley 2022)

Under the above assumptions, the error $e_k = \hat{x}_k - x^*$ satisfies

$$\|e_{k+1}\| = O (\|e_k\|^2 + (\varepsilon_2 + \varepsilon_s)\|e_k\| + \varepsilon_1 + \varepsilon_a)$$

Note: Inexact Hessian and linear solves impact convergence rate, but not limiting accuracy. Gradient and update errors do not harm rate, but affect limiting accuracy.

Warning: hidden constants proportional to $\|\nabla^2 f(x^*)^{-1}\|$, $\kappa(\nabla^2 f)$, and problem size.
Theorem (Kelley 2022)

Under the above assumptions, the error $e_k = \hat{x}_k - x^*$ satisfies

$$
\|e_{k+1}\| = O\left(\|e_k\|^2 + (\varepsilon_2 + \varepsilon_s)\|e_k\| + \varepsilon_1 + \varepsilon_a\right)
$$

Note: Inexact Hessian and linear solves impact convergence rate, but not limiting accuracy. Gradient and update errors do not harm rate, but affect limiting accuracy.

Warning: hidden constants proportional to $\|\nabla^2 f(x^*)^{-1}\|$, $\kappa(\nabla^2 f)$, and problem size.

Typical mixed-precision strategy: high-precision gradient evaluations and update and low-precision Hessian evaluation/approximation and inversion so that, e.g.

$$
\varepsilon_1, \varepsilon_a = O(u^2); \ \varepsilon_2, \varepsilon_s = O(u) \quad \Rightarrow \quad \|e_{k+1}\| \approx O\left(\|e_k\|^2 + u^2\right).
$$

Since the reduction in the rate occurs when $\|e_k\| \leq O(u)$ for which $\|e_{k+1}\| = O(u^2)$. 
3. Numerical linear algebra

Two topics:

Review articles (citing all mentioned references):


Mixed-precision iterative refinement for $Ax = b$ [Langou et al. 2006], [Carson & Higham 2017-18]
Mixed-precision iterative refinement for $Ax = b$ [Langou et al. 2006], [Carson & Higham 2017-18]

Apply mixed-precision Newton to $Ax - b = 0$. Use two precisions $u$, $u^2$. 

Since $\|e_0\| = O(u)$ the previous theorem gives that $\|e_1\| = O(\|e_0\|^2 + u\|e_0\| + u^2) = O(u^2)$. 

Advantages: LU factorization performed only once in low precision. Limiting accuracy dictated by $\kappa_\infty(A)$ is small enough.
Mixed-precision iterative refinement for $Ax = b$ [Langou et al. 2006], [Carson & Higham 2017-18]

Apply mixed-precision Newton to $Ax - b = 0$. Use two precisions $u, u^2$.

**Mixed-precision iterative refinement**

Solve $Ax_0 = b$ using LU factorization in precision $u$ and store the LU factors. 
**For** $k = 1, 2, \ldots$

1. Compute residual $r_k = b - Ax_k$ at precision $u^2$.
2. Solve $Ad_k = r_k$ at precision $u$ by re-using the LU factors.
3. $x_{k+1} = x_k + d_k$ at precision $u^2$.

Since $\|e_0\| = O(u)$ the previous theorem gives that

$$\|e_1\| = O(\|e_0\|^2 + u\|e_0\| + u^2) = O(u^2).$$
Mixed-precision iterative refinement for $Ax = b$ [Langou et al. 2006], [Carson & Higham 2017-18]

Apply mixed-precision Newton to $Ax - b = 0$. Use two precisions $u, u^2$.

Mixed-precision iterative refinement

Solve $Ax_0 = b$ using LU factorization in precision $u$ and store the LU factors.

For $k = 1, 2, \ldots$

1. Compute residual $r_k = b - Ax_k$ at precision $u^2$.
2. Solve $Ad_k = r_k$ at precision $u$ by re-using the LU factors.
3. $x_{k+1} = x_k + d_k$ at precision $u^2$.

Since $\|e_0\| = O(u)$ the previous theorem gives that

$$\|e_1\| = O(\|e_0\|^2 + u\|e_0\| + u^2) = O(u^2).$$

Advantages: LU factorization performed only once in low precision. Limiting accuracy dictated by $u^2$ provided $\kappa_\infty(A)$ is small enough.
GMRES-IR [Carson & Higham 2017-18, Amestoy et al. 2021]

Now use three precisions: $u_l \geq u \geq u^2$. In [Amestoy et al. 2021] they use five.

---

**GMRES-IR**

Solve $Ax_0 = b$ using LU factorization in precision $u_l$ and store the LU factors.

**For** $k = 1, 2, \ldots$

1. Compute residual $r_k = b - Ax_k$ at precision $u^2$.
2. Solve $Ad_k = r_k$ at precision $u$ by using GMRES with $U^{-1}L^{-1}$ as preconditioner and matrix-vector products performed at precision $u^2$.
3. $x_{k+1} = x_k + d_k$ at precision $u$.
GMRES-IR [Carson & Higham 2017-18, Amestoy et al. 2021]

Now use three precisions: $u_l \geq u \geq u^2$. In [Amestoy et al. 2021] they use five.

---

### GMRES-IR

Solve $Ax_0 = b$ using LU factorization in precision $u_l$ and store the LU factors.

**For** $k = 1, 2, \ldots$

1. Compute residual $r_k = b - Ax_k$ at precision $u^2$.
2. Solve $Ad_k = r_k$ at precision $u$ by using GMRES with $U^{-1}L^{-1}$ as preconditioner and matrix-vector products performed at precision $u^2$.
3. $x_{k+1} = x_k + d_k$ at precision $u$.

---

**Result:**

- Provided that $\kappa_\infty(A) \ll u^{-1}$ we obtain a limiting accuracy of $O(u)$ where the hidden constant is independent from $\kappa_\infty(A)$.
- This approach is efficient since again the LU factorization is performed only once and in low precision, and GMRES typically converges in a handful of iterations.
- GMRES-IR is more robust to ill-conditioning than LU-based iterative refinement.
Mixed-precision iterative refinement is at the heart of many recent mixed-precision developments in numerical linear algebra, including:

- Sparse approximate factorizations (e.g. replace LU with a sparse approximation), cf. [Amestoy et al. 2022].
- Least square problems (see e.g. [Carson et al. 2020]).
- Eigenvalue problems (see e.g. [Tisseur 2001]).
- Multigrid (see e.g. [Tamstorf et al. 2021] and [McCormick et al. 2021]).
Mixed-precision Krylov methods and preconditioning

Complex theory: the theory describing the finite precision behavior of iterative methods is extensive and complex. Review on Lanczos-CG [Meurant & Strakoš 2006].

Practical methods: much work focuses on showing what improves performance in practice rather than on theoretical results.

Three approaches: (see review articles for more details and info):

1. Iterative refinement. Use lower precision in inner solver.
2. MP preconditioning. Apply/implement preconditioner in low precision.
3. MP iterative methods. Adaptively change precision of inner products/matvecs.
Mixed-precision Krylov methods and preconditioning

**Complex theory:** the theory describing the finite precision behavior of iterative methods is extensive and complex. Review on Lanczos-CG: [Meurant & Strakoš 2006].

**Practical methods:** much work focuses on showing what improves performance in practice rather than on theoretical results.
Mixed-precision Krylov methods and preconditioning

**Complex theory:** the theory describing the finite precision behavior of iterative methods is extensive and complex. Review on Lanczos-CG: [Meurant & Strakoš 2006].

**Practical methods:** much work focuses on showing what improves performance in practice rather than on theoretical results.

**Three approaches:** (see review articles for more details and info):

1. **Iterative refinement.** Use lower precision in inner solver.
Mixed-precision Krylov methods and preconditioning

Complex theory: the theory describing the finite precision behavior of iterative methods is extensive and complex. Review on Lanczos-CG: [Meurant & Strakoš 2006].

Practical methods: much work focuses on showing what improves performance in practice rather than on theoretical results.

Three approaches: (see review articles for more details and info):

1. Iterative refinement. Use lower precision in inner solver.
2. MP preconditioning. Apply/implement preconditioner in low precision.
Mixed-precision Krylov methods and preconditioning

**Complex theory:** the theory describing the finite precision behavior of iterative methods is extensive and complex. Review on Lanczos-CG: [Meurant & Strakoš 2006].

**Practical methods:** much work focuses on showing what improves performance in practice rather than on theoretical results.

**Three approaches:** (see review articles for more details and info):

1. **Iterative refinement.** Use lower precision in inner solver.
2. **MP preconditioning.** Apply/implement preconditioner in low precision.
3. **MP iterative methods.** Adaptively change precision of inner products/matvecs.
4. Numerical solution of partial differential equations

Main references:


4a. Towards climate simulations in half precision

Joint with: M. Klöwer and T. N. Palmer (University of Oxford), 
S. Hatfield and P. D. Düben (European Centre for Medium-Range Weather Forecasts).

Algorithm type: reduced-precision (half).

Main references:

Towards climate simulations in half precision [Klöwer et al. 2021]

Shallow-water eqs for 2D oceanic flow:
\[
\begin{align*}
\dot{v} + v \cdot \nabla v + \hat{z} \times v &= -\nabla \eta + \Delta^2 v - v + F, \\
\dot{\eta} + \nabla \cdot (vh) &= 0, \\
\dot{q} + v \cdot \nabla q &= -\tau(q - q_0).
\end{align*}
\]

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

Techniques used for fp16 simulations:
\begin{itemize}
\item Scaling and squeezing.
\item Kahan compensated summation.
\item Performed using A64FX chips on Fugaku (1st in TOP500).
\end{itemize}

Note: all other results in this part of the talk use precision emulation in software.
4b. Solving parabolic PDEs in half precision

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

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

Main references:


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!
RtN vs SR

Why is RtN in low precision bad for parabolic PDEs?

a) **Stagnation:**
   - RtN always stagnates for sufficiently small $\Delta t$.

b) **Global error:**
   - RtN rounding errors are strongly correlated and grow rapidly until stagnation.

**SR fixes all these issues!**
a) Stagnation (heat equation, left 1D, right 2D)

RtN computations are discretization and initial condition dependent. SR works!
b) Global rounding errors [C. and Giles 2020]

Let \( \epsilon^n \in \mathbb{R}^K \) be the vector containing all rounding errors introduced at time step \( n \).
Define the global rounding error \( E^n = \hat{U}^n - U^n \). It can be shown that
\[
E^{n+1} = S E^n + \epsilon^n.
\]

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

We can distinguish two cases:

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

**SR:** the \( \epsilon^n_i \) 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 $E^n = \hat{U}^n - U^n$. It can be shown that

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

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

We can distinguish two cases:

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

**SR:** the $\varepsilon^n_i$ are zero-mean, independent in space and mean-independent in time.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Norm</th>
<th>1D $O(u\Delta t^{-1})$</th>
<th>2D $O(u\Delta t^{-1})$</th>
<th>3D $O(u\Delta t^{-1})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>RtN</td>
<td>$L^2, \infty$</td>
<td>$O(u\Delta t^{-1})$</td>
<td>$O(u\Delta t^{-1})$</td>
<td>$O(u\Delta t^{-1})$</td>
</tr>
<tr>
<td>SR</td>
<td>$E[</td>
<td></td>
<td>\cdot</td>
<td></td>
</tr>
<tr>
<td>SR</td>
<td>$E[</td>
<td></td>
<td>\cdot</td>
<td></td>
</tr>
</tbody>
</table>

Asymptotic global rounding error blow-up rates; $\ell(\Delta t) = |\log(\Delta t)|$. 
b) Global rounding errors (2D heat equation)

Global error (delta form, 2D)

Note: relative error = error \times (u||U^N||)^{-1}
4c. Mixed-precision explicit Runge-Kutta methods

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

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

**Main reference:**

Framework and objective

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

\[ y'(t) = f(t, y(t)), \quad y(0) = y_0, \]

where \( f(t, y) \) is sufficiently smooth, and from now on set \( f = f(y(t)) \) for simplicity.

**Objective**

Evaluate \( f \) in low-precision as much as possible without affecting accuracy or stability.

**Note:** 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 \( |R_s(z)| < 1. \)
Linear stability for RK methods (in practice)
Linear stability for RKC (in practice, $s = 128$, $u = 2^{-8}$)
Order-preserving mixed-precision RK methods

**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.

We saw that RP methods do not converge, hence they are not order-preserving.
Order-preserving mixed-precision RK methods

### 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.

We saw that RP methods do not converge, hence they are not order-preserving.

**Our idea:** store solution in high precision and use only $q$ high-precision function evaluations to obtain a $q$-order-preserving mixed-precision RK method.

We can construct $q$-order preserving RK methods for any $q$ for linear problems, and for $q = 1, 2$ for nonlinear problems. We can prove both stability and convergence.
Order-preserving mixed-precision RK methods

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.

We saw that RP methods do not converge, hence they are not order-preserving.

Our idea: store solution in high precision and use only $q$ high-precision function evaluations to obtain a $q$-order-preserving mixed-precision RK method.

We can construct $q$-order preserving RK methods for any $q$ for linear problems, and for $q = 1, 2$ for nonlinear problems. We can prove both stability and convergence.

Note: We mainly focused on stabilized methods since they are low-order, but use a lot of function evaluations to maximize stability.
Linear problems, i.e. \( f(y) = Ay \)

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

\[
y(\Delta t) = \exp(\Delta t A) y_0 = \sum_{j=0}^{\infty} \frac{(\Delta t A)^j}{j!} y_0,
\]

\[
y_1 = \sum_{j=0}^{p} \frac{(\Delta t A)^j}{j!} y_0 + O(\Delta t^{p+1}).
\]

Giving a local error of \( \tau = \Delta t^{-1} \| y(\Delta t) - y_1 \| = O(\Delta t^p) \).
Linear problems, i.e. $f(y) = Ay$

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

$$y(\Delta t) = \exp(\Delta t A)y_0 = \sum_{j=0}^{\infty} \frac{(\Delta t A)^j}{j!} y_0,$$

$$y_1 = \sum_{j=0}^{p} \frac{(\Delta t A)^j}{j!} y_0 + O(\Delta t^{p+1}).$$

Giving a local error of $\tau = \Delta t^{-1} ||y(\Delta t) - y_1|| = O(\Delta t^p)$.

Evaluating the scheme in finite precision yields:

$$\hat{y}_1 = \varepsilon + y_0 + \sum_{j=1}^{p} \frac{\Delta t^j}{j!} \left( \prod_{k=1}^{j} (A + \Delta A_k) \right) y_0 + O(\Delta t^{p+1}).$$
Linear problems

\[ \tau = \Delta^{-1} \| \hat{y}_1 - y_1 \| = \Delta t^{-1} \left\| \varepsilon + \sum_{j=1}^{p} \frac{\Delta t^j}{j!} \left( \prod_{k=1}^{j} (A + \Delta A_k) - A^j \right) y_0 \right\| + O(\Delta t^p). \]
Linear problems

\[ \tau = \Delta^{-1} \| \hat{y}_1 - y_1 \| = \Delta t^{-1} \left\| \varepsilon + \sum_{j=1}^{p} \frac{\Delta t^j}{j!} \left( \prod_{k=1}^{j} (A + \Delta A_k) - A^j \right) y_0 \right\| + O(\Delta t^p). \]

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

\[ \tau = \Delta^{-1} \| \hat{y}_1 - y_1 \| = \Delta t^{-1} \left\| \varepsilon + \sum_{j=1}^{p} \frac{\Delta t^j}{j!} \left( \prod_{k=1}^{j} (A + \Delta A_k) - A^j \right) y_0 \right\| + O(\Delta t^p). \]

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.
\[ \tau = \Delta^{-1} \| \hat{y}_1 - y_1 \| = \Delta t^{-1} \left\| \varepsilon + \sum_{j=1}^{p} \frac{\Delta t^j}{j!} \left( \prod_{k=1}^{j} (A + \Delta A_k) - A^j \right) y_0 \right\| + O(\Delta t^p). \]

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!
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):

\[
\begin{align*}
\dot{u} &= \alpha \Delta u + u^2 v - (b + 1) u + a \\
\dot{v} &= \alpha \Delta v - u^2 v + b u
\end{align*}
\]
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
To sum up

- Reduced-/mixed-precision algorithms require a careful implementation, but can bring significant memory, cost, and energy savings.
- Many new reduced and mixed-precision algorithms for scientific computing and data science were developed in recent years. Hardware support is growing.
- **Advice for new developers:** find which operations are more costly or more sensitive to rounding errors before designing a mixed-precision method.
- **Advice for new practitioners:** keep GPU and FPGA applications in mind as that’s where most savings can currently be obtained.
To sum up

- Reduced-/mixed-precision algorithms require a careful implementation, but can bring significant memory, cost, and energy savings.
- Many new reduced and mixed-precision algorithms for scientific computing and data science were developed in recent years. Hardware support is growing.
- **Advice for new developers:** find which operations are more costly or more sensitive to rounding errors before designing a mixed-precision method.
- **Advice for new practitioners:** keep GPU and FPGA applications in mind as that’s where most savings can currently be obtained.

Thank you for listening!

Papers, slides, and more info at: https://croci.github.io
Email: matteo.croci@austin.utexas.edu
References


References II


