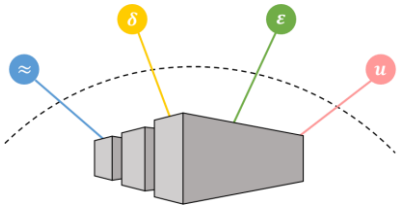


Mixed-Precision Computing: High Accuracy with Low Precision

Erin C. Carson
Faculty of Mathematics and Physics
Charles University

ARITH 2026
June 29, 2026

FACULTY OF MATHEMATICS AND PHYSICS
Charles University

Co-funded by the European Union

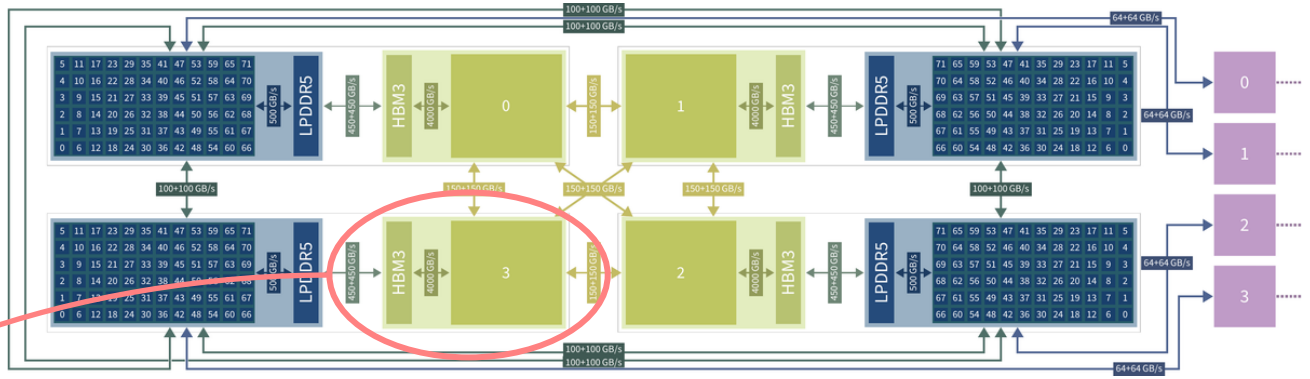
We acknowledge funding from ERC Starting Grant No. 101075632 and the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Admin. Views and opinions expressed are however those of the author only and do not necessarily reflect those of the European Union or the ERC. Neither the European Union nor the granting authority can be held responsible for them.
This work has been supported by Charles University Research Centre program No. UNCE/24/SCI/005.

The Exascale Era

We are in the “Exascale Era” of scientific computing

- 10^{18} floating point operations per second

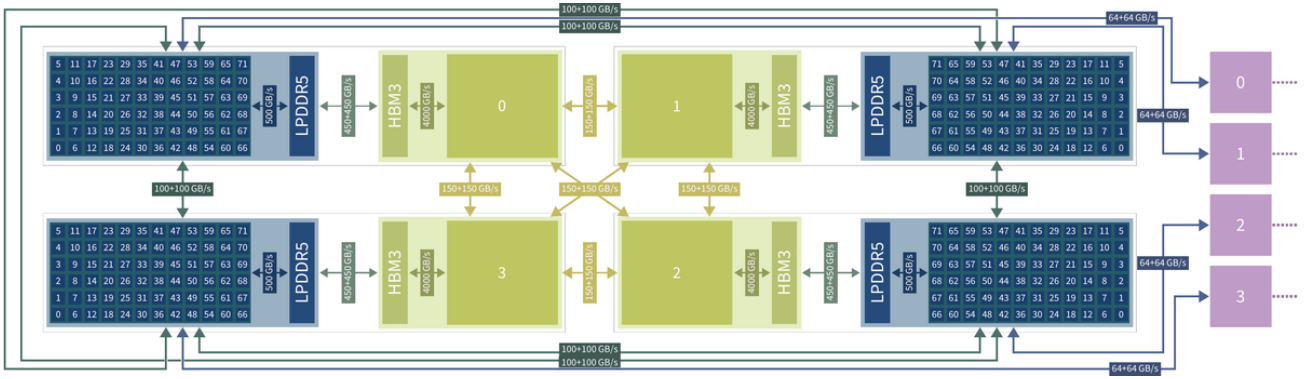
Exascale Hardware



https://www.fz-juelich.de/en/ias/jsc/jupiter/tech

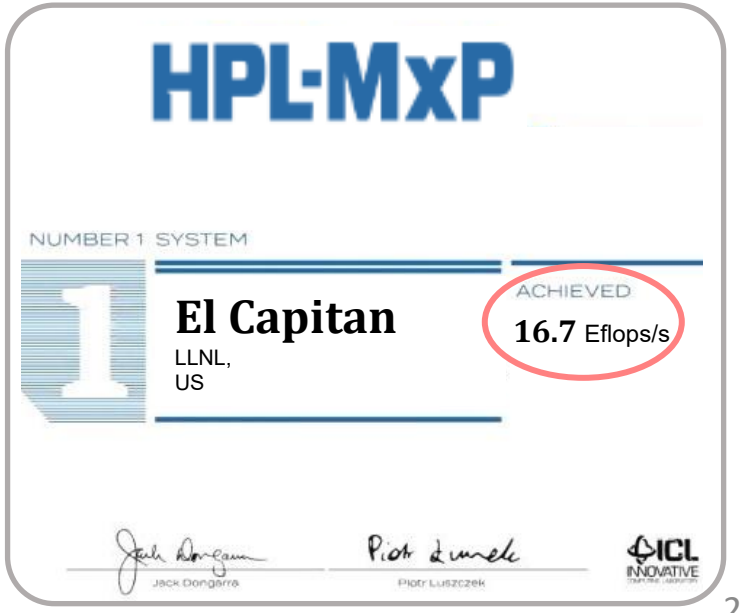
	size (bits)	range	u	perf. (NVIDIA H100 TC)
fp64	64	$10^{\pm 308}$	1×10^{-16}	67 Tflops/s
fp32	32	$10^{\pm 38}$	6×10^{-8}	989 Tflop/s
fp16	16	$10^{\pm 5}$	5×10^{-4}	1979 Tflops/s
bfloat16	16	$10^{\pm 38}$	4×10^{-3}	
fp8-e5m2	8	$10^{\pm 5}$	1×10^{-1}	3958 Tflops/s
fp8-e4m3	8	$10^{\pm 2}$	6×10^{-2}	

Exascale Hardware

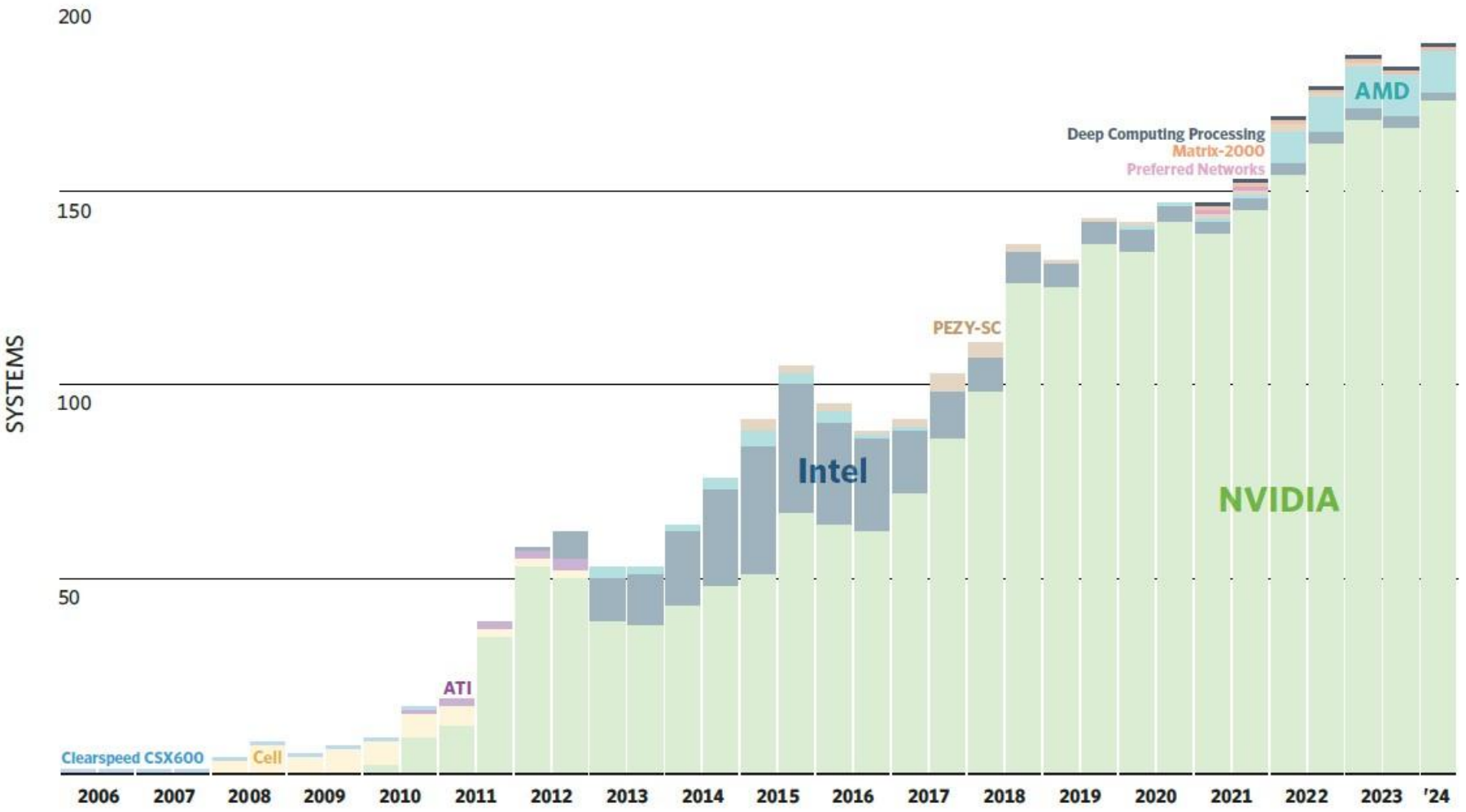


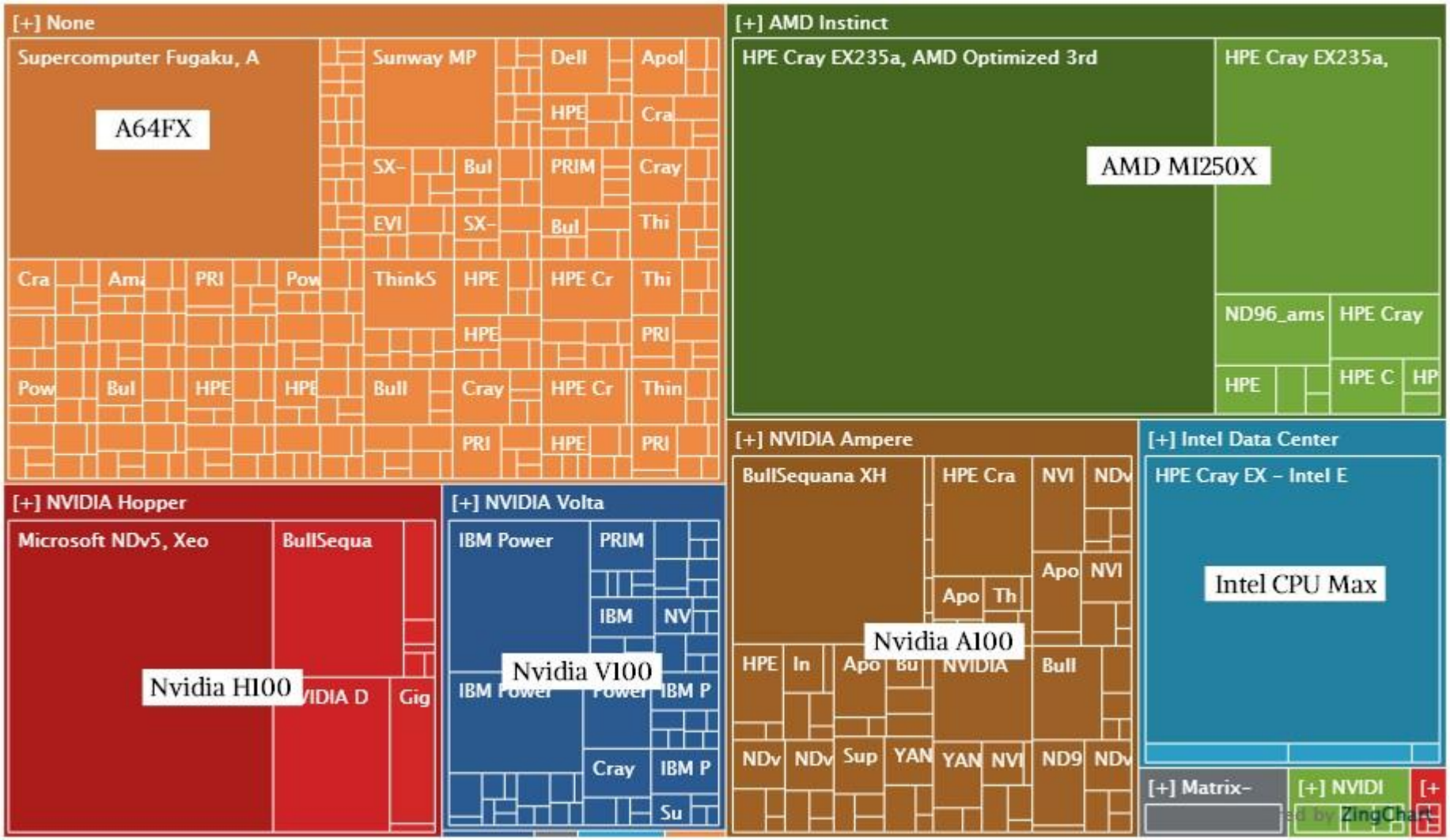
https://www.fz-juelich.de/en/ias/jsc/jupiter/tech

	size (bits)	range	u	perf. (NVIDIA H100 TC)
fp64	64	$10^{\pm 308}$	1×10^{-16}	67 Tflops/s
fp32	32	$10^{\pm 38}$	6×10^{-8}	989 Tflop/s
fp16	16	$10^{\pm 5}$	5×10^{-4}	1979 Tflops/s
bfloat16	16	$10^{\pm 38}$	4×10^{-3}	
fp8-e5m2	8	$10^{\pm 5}$	1×10^{-1}	3958 Tflops/s
fp8-e4m3	8	$10^{\pm 2}$	6×10^{-2}	



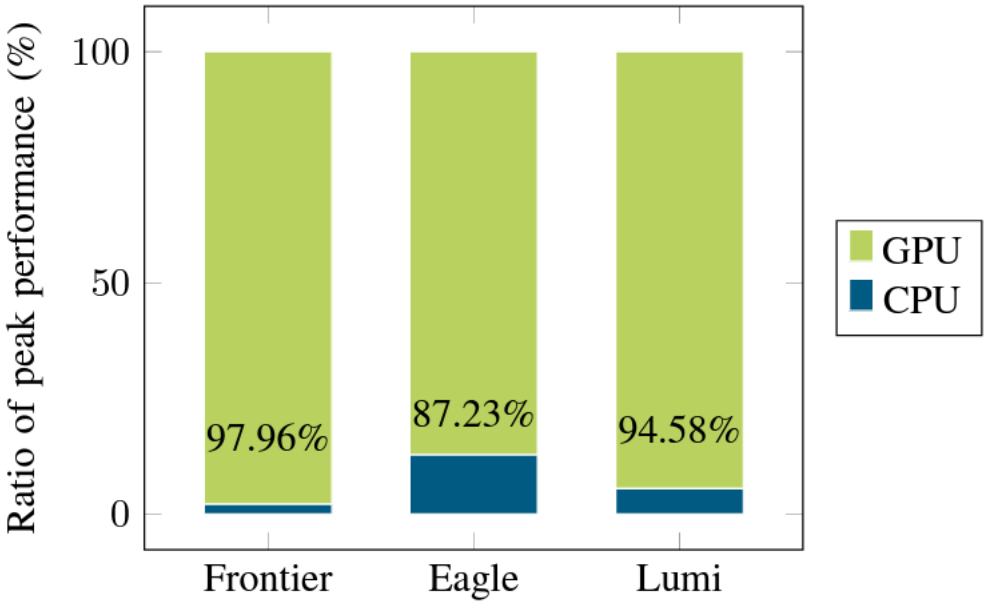
Growth of Accelerated Systems





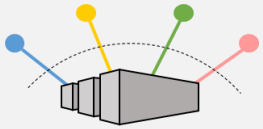
https://www.nextplatform.com/2023/11/13/top500-supercomputers-who-gets-the-most-out-of-peak-performance/

Performance comes from Accelerators



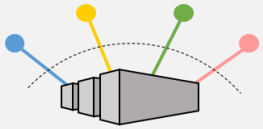
We MUST develop hardware-aware algorithms for matrix computations that can take advantage of mixed precision capabilities!

Challenges of Using Low/Mixed Precision



1. Need for Descriptive Bounds and Adaptive Approaches

- Goal:
  - Improve performance by using potentially low precision in expensive parts
  - While still achieving desired accuracy



1. Need for Descriptive Bounds and Adaptive Approaches

- Goal:
  - Improve performance by using potentially low precision in expensive parts
  - While still achieving desired accuracy

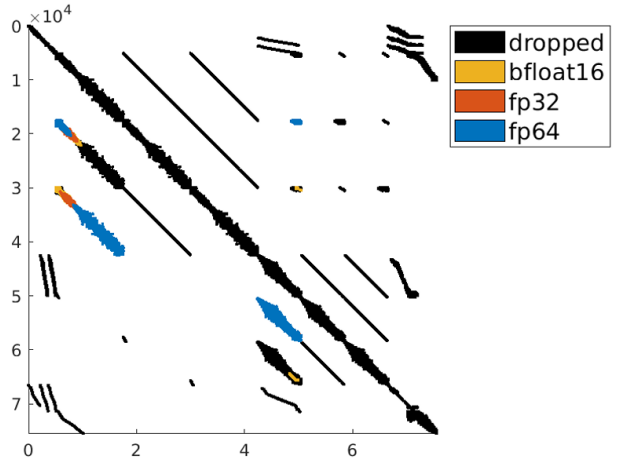
- Need a **rigorous understanding** of how the finite precision computation in each part of an algorithm affects numerical properties (stability, convergence, accuracy)
  - Requires, e.g., **knowledge about the behavior of nonlinear iterative solvers**

1. Need for Descriptive Bounds and Adaptive Approaches

- Goal:
  - Improve performance by using potentially low precision in expensive parts
  - While still achieving desired accuracy

- Need a **rigorous understanding** of how the finite precision computation in each part of an algorithm affects numerical properties (stability, convergence, accuracy)
  - Requires, e.g., **knowledge about the behavior of nonlinear iterative solvers**

- Error bounds will necessarily depend on the data (problem dimension, conditioning, norms of various quantities, etc.)
  ⇒ Can't choose the right precisions *a priori*
  ⇒ Need adaptive approaches



[Molina, Graillat, Jézéquel, Mary, 2022]

2. The in(validity) of bounds at scale

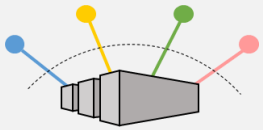
- Error bounds derived the typical way involve constraints like $nu < 1$ or $\kappa(A)u < 1$

to show that the MGS-GMRES method does eventually produce a backward stable approximate solution when applied to any member of the following class of linear systems with floating point arithmetic unit roundoff ϵ (σ means singular value):

$$Ax = b \neq 0, \quad A \in \mathbf{R}^{n\times n}, \quad b \in \mathbf{R}^n, \quad \sigma_{min}(A) \gg n^2\epsilon\|A\|_F; \tag{1.1}$$

propose the first provably stable one-synchronization reorthogonalized BCGS variant, demonstrating that it has $O(\boldsymbol{u})$ loss of orthogonality under the condition $O(\boldsymbol{u})\kappa^2(\boldsymbol{\mathcal{X}}) \leq 1/2$, where $\kappa(\cdot)$ represents the condition number. By incorporating one additional synchronization point, we develop a two-

We also define $\gamma_k = ku/(1-ku)$ for $ku < 1$.



2. The in(validity) of bounds at scale

- Error bounds derived the typical way involve constraints like $nu < 1$ or $\kappa(A)u < 1$

to show that the MGS-GMRES method does eventually produce a backward stable approximate solution when applied to any member of the following class of linear systems with floating point arithmetic unit roundoff ϵ (σ means singular value):

$$Ax = b \neq 0, \quad A \in \mathbf{R}^{n\times n}, \quad b \in \mathbf{R}^n, \quad \sigma_{min}(A) \gg n^2\epsilon\|A\|_F; \tag{1.1}$$

propose the first provably stable one-synchronization reorthogonalized BCGS variant, demonstrating that it has $O(\boldsymbol{u})$ loss of orthogonality under the condition $O(\boldsymbol{u})\kappa^2(\boldsymbol{\mathcal{X}}) \leq 1/2$, where $\kappa(\cdot)$ represents the condition number. By incorporating one additional synchronization point, we develop a two-

We also define $\gamma_k = ku/(1-ku)$ for $ku < 1$.

- Is this reality or just an effect of worst-case analysis?

2. The in(validity) of bounds at scale

- Error bounds derived the typical way involve constraints like $nu < 1$ or $\kappa(A)u < 1$

to show that the MGS-GMRES method does eventually produce a backward stable approximate solution when applied to any member of the following class of linear systems with floating point arithmetic unit roundoff ϵ (σ means singular value):

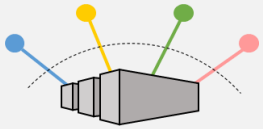
$$Ax = b \neq 0, \quad A \in \mathbf{R}^{n\times n}, \quad b \in \mathbf{R}^n, \quad \sigma_{min}(A) \gg n^2\epsilon\|A\|_F; \tag{1.1}$$

propose the first provably stable one-synchronization reorthogonalized BCGS variant, demonstrating that it has $O(\boldsymbol{u})$ loss of orthogonality under the condition $O(\boldsymbol{u})\kappa^2(\boldsymbol{\mathcal{X}}) \leq 1/2$, where $\kappa(\cdot)$ represents the condition number. By incorporating one additional synchronization point, we develop a two-

We also define $\gamma_k = ku/(1-ku)$ for $ku < 1$.

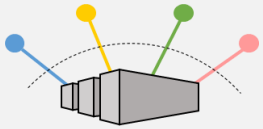
- Is this reality or just an effect of worst-case analysis?
- One solution: probabilistic approach [Higham, Mary, 2019], [Higham, Mary, 2020]

$$nu < 1 \Rightarrow \sqrt{n}u < 1 \text{ or even } u < 1$$



3. Working with a Limited Range

- We usually assume that no overflow or underflow occurs
  - Overflow: Complete failure
  - Underflow: Potential loss of important numerical properties (e.g., nonsingularity, positive definiteness)



3. Working with a Limited Range

- We usually assume that no overflow or underflow occurs
  - Overflow: Complete failure
  - Underflow: Potential loss of important numerical properties (e.g., nonsingularity, positive definiteness)

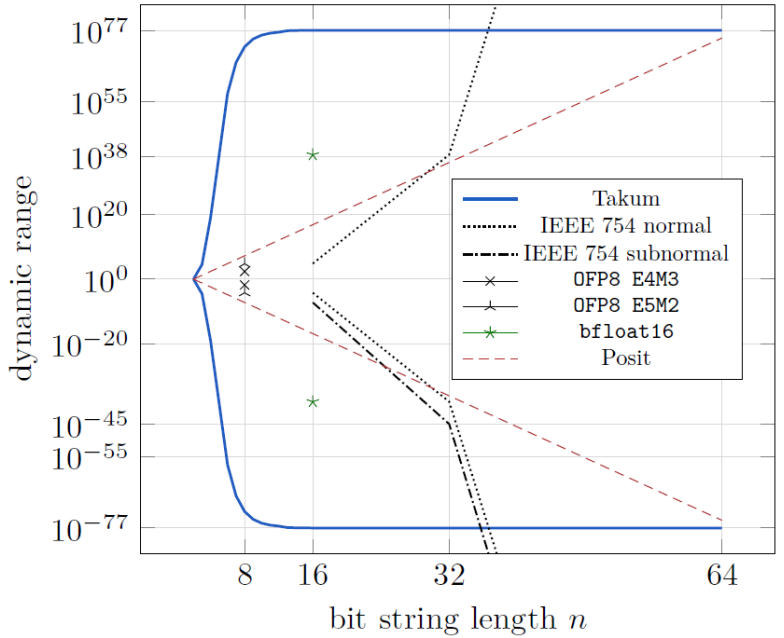
- Maybe reasonable for fp64 ($10^{\pm 308}$), not for lower precisions! (e.g., fp16 with range $10^{\pm 5}$)

3. Working with a Limited Range

- We usually assume that no overflow or underflow occurs
  - Overflow: Complete failure
  - Underflow: Potential loss of important numerical properties (e.g., nonsingularity, positive definiteness)

- Maybe reasonable for fp64 ($10^{\pm 308}$), not for lower precisions! (e.g., fp16 with range $10^{\pm 5}$)

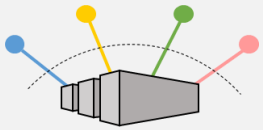
- Some solutions:
  - Scaling and shifting approach [Higham, Pranesh, 2019]
  - Rigorous error handling [Scott, Tůma, 2025]
  - New number formats, e.g., bfloat16, takum, block floating point



[Hunhold, 2025]

When Mixed Precision Makes Sense

1. When the algorithm is self-correcting



Mixed Precision Iterative Refinement

- Mixed precision iterative refinement used by Wilkinson, Turing, and others as early as 1948 (“Progress Report on the Automatic Computing Engine”)
- First analyzed by Wilkinson in 1963
- Hardware at the time: exact scalar product in precision u^2 of two numbers in precision u for free

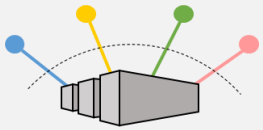
Solve $Ax_0 = b$ by LU factorization — precision u

for $i = 0$: maxit

$r_i = b - Ax_i$ — precision u^2

Solve $Ad_i = r_i$ via $d_i = U^{-1}(L^{-1}r_i)$ — precision u

$x_{i+1} = x_i + d_i$ — precision u



Mixed Precision Iterative Refinement

- Mixed precision iterative refinement used by Wilkinson, Turing, and others as early as 1948 ("Progress Report on the Automatic Computing Engine")
- First analyzed by Wilkinson in 1963
- Hardware at the time: exact scalar product in precision u^2 of two numbers in precision u for free

Solve $Ax_0 = b$ by LU factorization — precision u

for $i = 0$: maxit

$r_i = b - Ax_i$ — precision u^2

Solve $Ad_i = r_i$ via $d_i = U^{-1}(L^{-1}r_i)$ — precision u

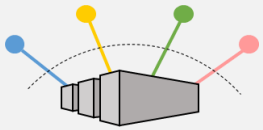
$x_{i+1} = x_i + d_i$ — precision u

Theorem [Wilkinson]: If

$$3nu\kappa_\infty(A) < 1,$$

the limiting relative forward and backward errors are on the order $O(u)$.

$\|x_i - x\|/\|x\|$ $\|b - Ax_i\|/(\|A\|\|x_i\| + \|b\|)$



Mixed Precision Iterative Refinement

[C., Higham, 2017]

Change solver → Can solve more ill-conditioned linear systems

Solve $Ax_0 = b$ by LU factorization — precision u

for $i = 0$: maxit

$r_i = b - Ax_i$ — precision u^2

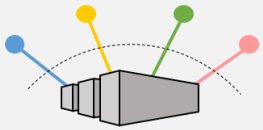
Solve $Ad_i = r_i$ via GMRES on $U^{-1}L^{-1}Ad_i = U^{-1}L^{-1}r_i$

$x_{i+1} = x_i + d_i$ — precision u

Theorem: If

$$3nu\kappa_\infty(U^{-1}L^{-1}A) < 1,$$

the limiting relative forward and backward errors are on the order $O(u)$.



Mixed Precision Iterative Refinement

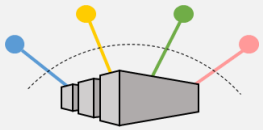
[C., Higham, 2018]

Extended and improved analysis to allow for 3 potentially different precisions

Solve $Ax_0 = b$	precision $\boldsymbol{u_f}$
for $i = 0$: maxit	
$r_i = b - Ax_i$	precision $\boldsymbol{u_r}$
Solve $Ad_i = r_i$	"effective precision" $\boldsymbol{u_s}$
$x_{i+1} = x_i + d_i$	precision $\boldsymbol{u}$

Combines Wilkinson-type iterative refinement with low-precision factorization variants [Langou et al., 2006], [Arioli and Duff, 2009], [Hogg and Scott, 2010], Abdelfattah et al., 2016]

Theorem example:
For choice $\boldsymbol{u_f}$ = half, $\boldsymbol{u}$ = single, $\boldsymbol{u_r}$ = double, forward and backward errors converge to the level $O(\boldsymbol{u})$, as long as $\kappa_\infty(A) \leq 10^8$ for GMRES-based IR



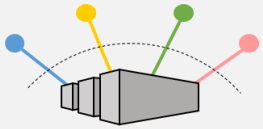
Mixed Precision Iterative Refinement

HPL-MxP PEOPLE LINKS PUBLICATIONS COLLABORATORS RULES SOFTWARE RESULTS UPLOAD

HPL-MXP MIXED-PRECISION BENCHMARK

The HPL-MxP benchmark seeks to highlight the emerging convergence of high-performance computing (HPC) and artificial intelligence (AI) workloads. While traditional HPC focused on simulation runs for modeling phenomena in physics, chemistry, biology, and so on, the mathematical models that drive these computations require, for the most part, 64-bit accuracy. On the other hand, the machine learning methods that fuel advances in AI achieve desired results at 32-bit and even lower floating-point precision formats. This lesser demand for accuracy fueled a resurgence of interest in new hardware platforms that deliver a mix of unprecedented performance levels and energy savings to achieve the classification and recognition fidelity afforded by higher-accuracy formats.

HPL-MxP strives to unite these two realms by delivering a blend of modern algorithms and contemporary hardware while simultaneously connecting to the solver formulation of the decades-old HPL framework of benchmarking the largest supercomputing installations in the world. The solver method of choice is a combination of LU factorization and iterative refinement performed afterwards to bring the solution back to 64-bit accuracy. The innovation of HPL-MxP lies in dropping the requirement of 64-bit computation throughout the entire solution process and instead opting for low-precision (likely 16-bit) accuracy for LU, and a sophisticated iteration to recover the accuracy lost in factorization. The iterative method guaranteed to be numerically stable is the generalized minimal residual method (GMRES), which uses application of the L and U factors to serve as a preconditioner. The combination of these algorithms is demonstrably sufficient for high accuracy and may be implemented in a way that takes advantage of the current and upcoming devices for accelerating AI workloads.



Other Examples of Self-Correction

- “Self-correcting”: repeatedly compute residuals or enforce invariants

Other Examples of Self-Correction

- "Self-correcting": repeatedly compute residuals or enforce invariants
- Newton's method for $F(x) = 0$ with starting vector x_1, precisions $u_r \leq u \leq u\_\ell$

for $i = 1 : i_{max}$

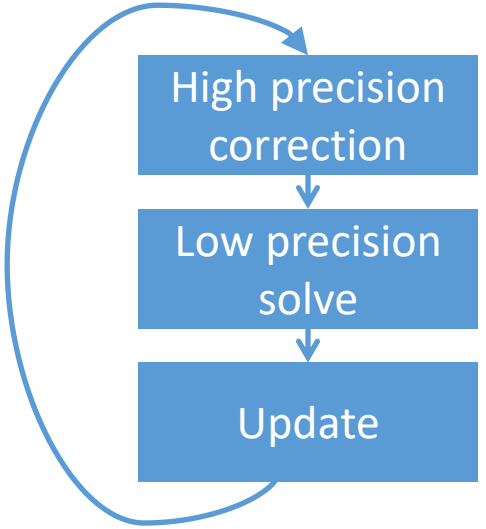
Compute $f_i = F(x_i)$ in precision u_r

Solve $J(x_i)d_i = -f_i$ in precision u_ℓ

$x_{i+1} = x_i + d_i$ in precision u

end

[Tisseur, 2001], [Kelley, 2022], [Higham, Mary, 2022]



Other Examples of Self-Correction

- "Self-correcting": repeatedly compute residuals or enforce invariants
- Newton's method for $F(x) = 0$ with starting vector x_1, precisions $u_r \leq u \leq u\_\ell$

for $i = 1 : i_{max}$

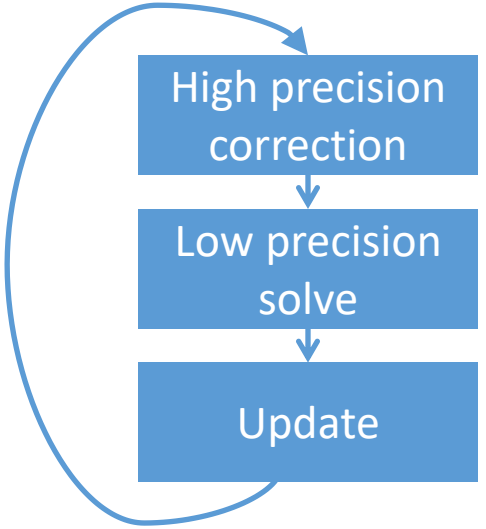
Compute $f_i = F(x_i)$ in precision u_r

Solve $J(x_i)d_i = -f_i$ in precision u_ℓ

$x_{i+1} = x_i + d_i$ in precision u

end

[Tisseur, 2001], [Kelley, 2022], [Higham, Mary, 2022]

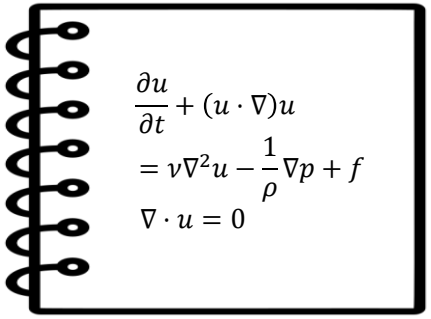


- Other examples:
  - Other stationary iterative methods
  - Preconditioned (or restarted) Krylov subspace methods
    e.g., [C., Daužickaitė, 2024]
  - Rayleigh quotient iteration (and related algorithms like Jacobi Davidson)
    e.g., [Oktay, C., 2024]

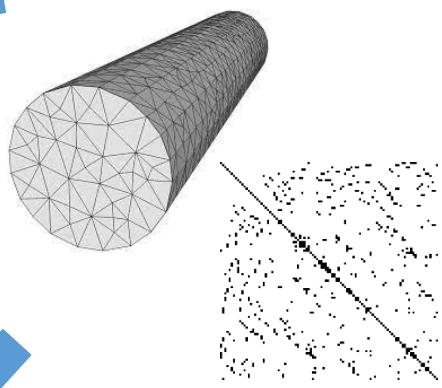
2. When finite precision error is small compared to other sources of error

Sources of Errors

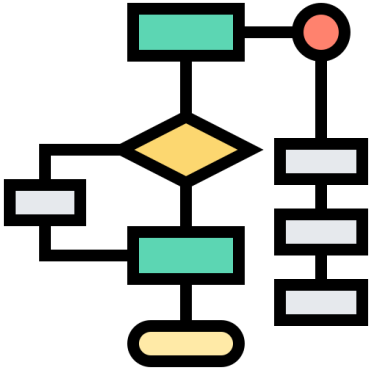
modeling error



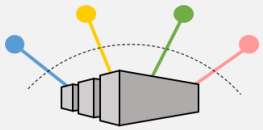
discretization/ linearization error, measurement error



algorithmic approximation



rounding error/ truncation



Randomized Nyström Approximation

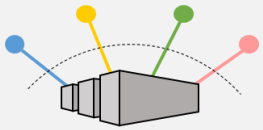
Want to compute a rank-k approximation of sym. PSD $A \approx U\Theta U^T$ via the randomized Nyström method.

Nyström approximation:

$$A_N = (A\Omega)(\Omega^T A\Omega)^\dagger (A\Omega)^T$$

where Ω is an $n \times k$ sampling matrix

Many applications: approximation of kernel matrices, spectral limited memory preconditioners, etc.



Randomized Nyström Approximation

Want to compute a rank-k approximation of sym. PSD $A \approx U\Theta U^T$ via the randomized Nyström method.

Nyström approximation:

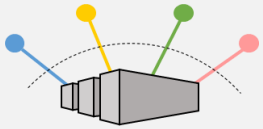
$$A_N = (A\Omega)(\Omega^T A\Omega)^\dagger (A\Omega)^T$$

where Ω is an $n \times k$ sampling matrix

Many applications: approximation of kernel matrices, spectral limited memory preconditioners, etc.

In the case that A is very large, matrix-matrix products with A are the bottleneck.

→ Can use single-pass version of the Nyström method [Tropp et al., 2017].

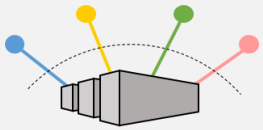


Single-Pass Nyström Approximation

Given sym. PSD matrix A, target rank k

$G = \text{randn}(n, k)$

$[\Omega, \sim] = \text{qr}(G, 0)$



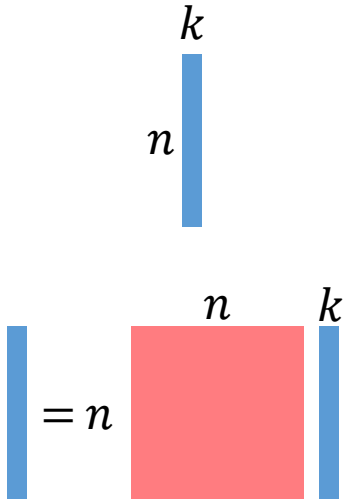
Single-Pass Nyström Approximation

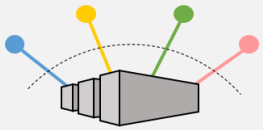
Given sym. PSD matrix A, target rank k

$G = \text{randn}(n, k)$

$[\Omega, \sim] = \text{qr}(G, 0)$

$\boldsymbol{Y = A\Omega}$





Single-Pass Nyström Approximation

Given sym. PSD matrix A, target rank k

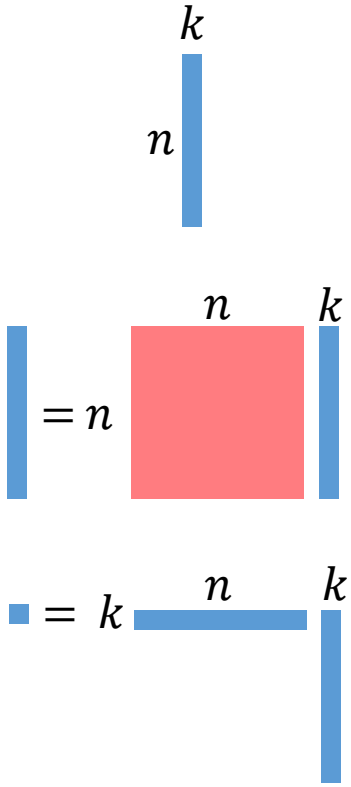
$G = \text{randn}(n, k)$

$[\Omega, \sim] = \text{qr}(G, 0)$

$\boldsymbol{Y = A\Omega}$

Compute shift ν; $Y_\nu = Y + \nu\Omega$

$B = \Omega^T Y_\nu$



Single-Pass Nyström Approximation

Given sym. PSD matrix A, target rank k

$G = \text{randn}(n, k)$

$[\Omega, \sim] = \text{qr}(G, 0)$

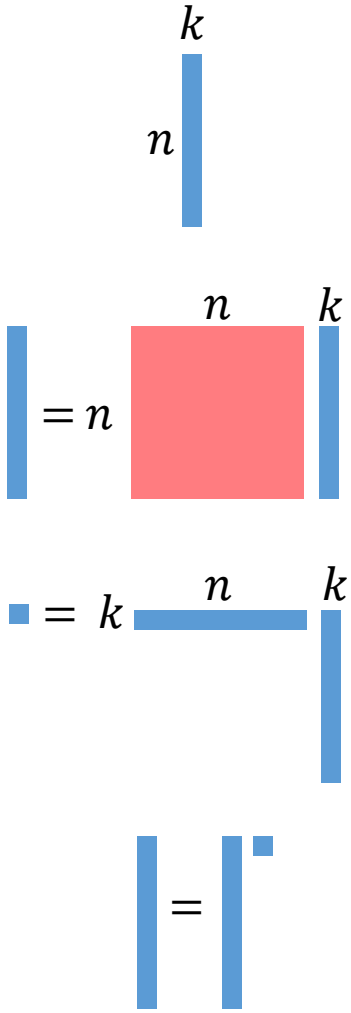
$\boldsymbol{Y = A\Omega}$

Compute shift ν; $Y_\nu = Y + \nu\Omega$

$B = \Omega^T Y_\nu$

$C = \text{chol}((B + B^T)/2)$

Solve $F = Y_\nu / C$



Single-Pass Nyström Approximation

Given sym. PSD matrix A, target rank k

$G = \text{randn}(n, k)$

$[\Omega, \sim] = \text{qr}(G, 0)$

$\boldsymbol{Y = A\Omega}$

Compute shift ν; $Y_\nu = Y + \nu\Omega$

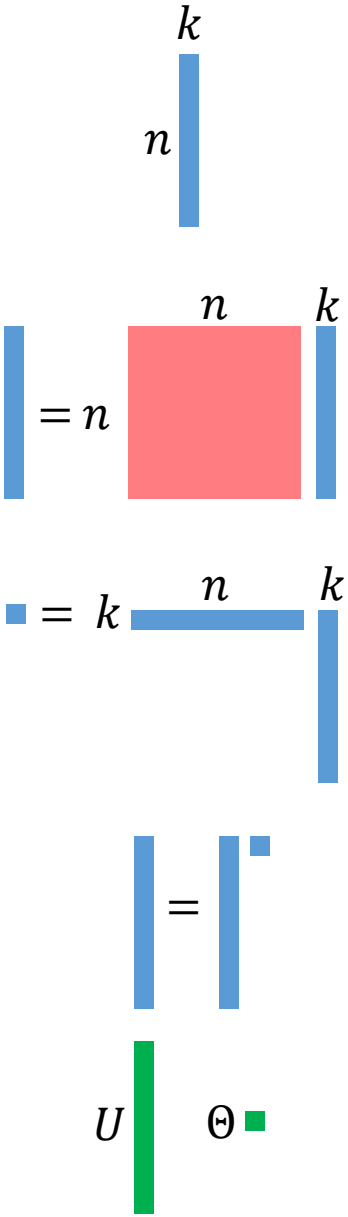
$B = \Omega^T Y_\nu$

$C = \text{chol}((B + B^T)/2)$

Solve $F = Y_\nu / C$

$[U, \Sigma, \sim] = \text{svd}(F, 0)$

$\Theta = \max(0, \Sigma^2 - \nu I)$



Single-Pass Nyström Approximation

Given sym. PSD matrix A, target rank k

$G = \text{randn}(n, k)$

$[\Omega, \sim] = \text{qr}(G, 0)$

$\boldsymbol{Y = A\Omega}$

Compute shift ν; $Y_\nu = Y + \nu\Omega$

$B = \Omega^T Y_\nu$

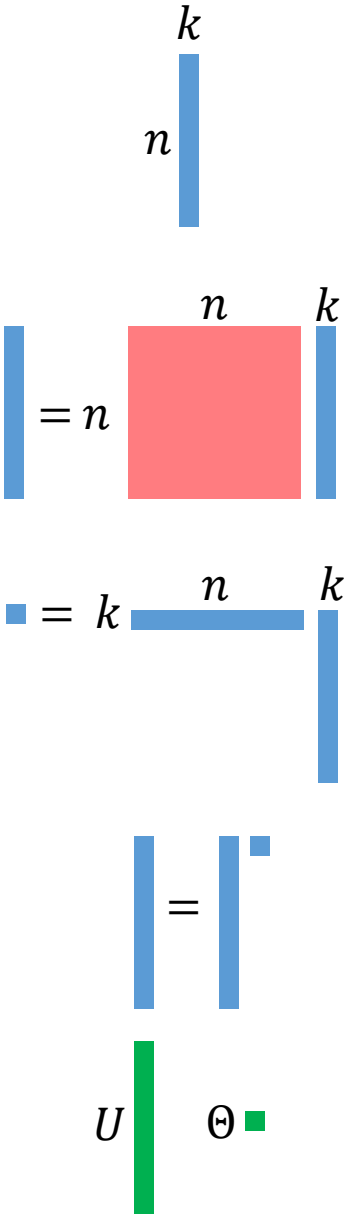
$C = \text{chol}((B + B^T)/2)$

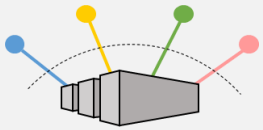
Solve $F = Y_\nu / C$

$[U, \Sigma, \sim] = \text{svd}(F, 0)$

$\Theta = \max(0, \Sigma^2 - \nu I)$

Can we further reduce the cost of the matrix-matrix product with A by using low precision?





Single-Pass Nyström Approximation

Given sym. PSD matrix A, target rank k

$\boldsymbol{u \leq u_p}$

$G = \text{randn}(n, k)$

$[\Omega, \sim] = \text{qr}(G, 0)$ (precision $\boldsymbol{u}$)

$\boldsymbol{Y = A\Omega}$ (precision $\boldsymbol{u_p}$)

Compute shift ν; $Y_\nu = Y + \nu\Omega$ (precision $\boldsymbol{u}$)

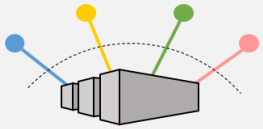
$B = \Omega^T Y_\nu$ (precision $\boldsymbol{u}$)

$C = \text{chol}((B + B^T)/2)$ (precision $\boldsymbol{u}$)

Solve $F = Y_\nu / C$ (precision $\boldsymbol{u}$)

$[U, \Sigma, \sim] = \text{svd}(F, 0)$ (precision $\boldsymbol{u}$)

$\Theta = \max(0, \Sigma^2 - \nu I)$ (precision $\boldsymbol{u}$)

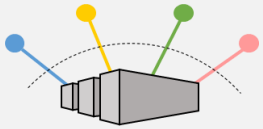


Error Bounds

$$\left\|A - \hat{A}_N\right\|_2 = \left\|A - A_N + A_N - \hat{A}_N\right\|_2 \leq \|A - A_N\|_2 + \left\|A_N - \hat{A}_N\right\|_2$$

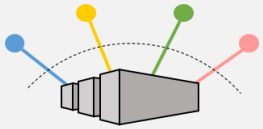
exact Nyström approximation

Nyström approximation computed in finite precision



Error Bounds

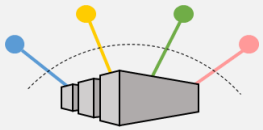
$$\left\|A - \hat{A}_N\right\|_2 = \left\|A - A_N + A_N - \hat{A}_N\right\|_2 \leq \underbrace{\|A - A_N\|_2}_{\text{exact approximation error}} + \underbrace{\left\|A_N - \hat{A}_N\right\|_2}_{\text{finite precision error}}$$



Error Bounds

$$\left\|A - \hat{A}_N\right\|_2 = \left\|A - A_N + A_N - \hat{A}_N\right\|_2 \leq \underbrace{\|A - A_N\|_2}_{\text{exact approximation error}} + \underbrace{\left\|A_N - \hat{A}_N\right\|_2}_{\text{finite precision error}}$$

Deterministic bound [Gittens, Mahoney, 2016]
Expected value bound [Frangella, Tropp, Udell, 2021]



Error Bounds

$$\left\|A - \hat{A}_N\right\|_2 = \left\|A - A_N + A_N - \hat{A}_N\right\|_2 \leq \underbrace{\|A - A_N\|_2}_{\text{exact approximation error}} + \underbrace{\left\|A_N - \hat{A}_N\right\|_2}_{\text{finite precision error}}$$

Theorem: If we choose precisions $\boldsymbol{u_p}$ and $\boldsymbol{u}$ such that

$$\kappa_2(A_k)\tilde{\kappa}(\Omega)^2 \ll \boldsymbol{u_p}^{-2}, \qquad \kappa_2(A_k)\tilde{\kappa}(\Omega)^2 \ll \boldsymbol{u}^{-1},$$

then

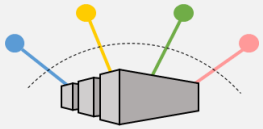
$$\left\|A_N - \hat{A}_N\right\|_F \lesssim \|A - A_N\|_F + k^{1/2}\, n\boldsymbol{u_p}\kappa_2(A_k)\tilde{\kappa}(\Omega)^2\|A\|_F$$

where

A_k is the best rank-k approximation of A

$\tilde{\kappa}(\Omega) = \|\Omega\|_F\left\|(W_1^T\Omega)^\dagger\right\|_2$, W_1 are eigenvectors for leading k eigenvalues of A

[C., Daužickaitė, 2024]



A Practical Heuristic

Heuristic: Likely that $\left\|A_N - \hat{A}_N\right\|_2 \leq \|A - A_N\|_2$ when

$$\boldsymbol{u_p} \leq n^{-1/2} \frac{\lambda_k}{\lambda_1}$$

A Practical Heuristic

Heuristic: Likely that $\left\|A_N - \hat{A}_N\right\|_2 \leq \|A - A_N\|_2$ when

$$\boldsymbol{u_p} \leq n^{-1/2} \frac{\lambda_k}{\lambda_1}$$

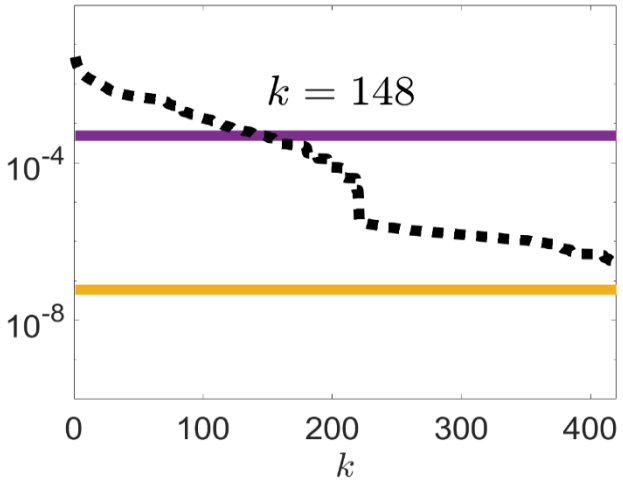
Interpretation: The greater the low-rank approximation error, the lower the precision we can use!

A Practical Heuristic

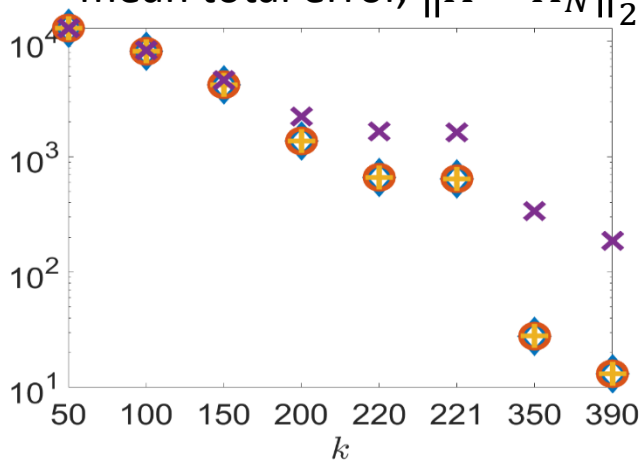
Heuristic: Likely that $\left\|A_N - \hat{A}_N\right\|_2 \leq \|A - A_N\|_2$ when

$$\boldsymbol{u_p} \leq n^{-1/2} \frac{\lambda_k}{\lambda_1}$$

Interpretation: The greater the low-rank approximation error, the lower the precision we can use!



mean total error, $\left\|A - \hat{A}_N\right\|_2$

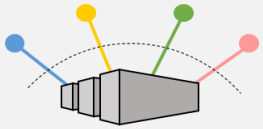


λ_k/λ_1 $\sqrt{n}\boldsymbol{u_p}$, $\boldsymbol{u_p}$ = half exact $\boldsymbol{u_p}$ = half, $\boldsymbol{u}$ = double

$\sqrt{n}\boldsymbol{u_p}$, $\boldsymbol{u_p}$ = single $\boldsymbol{u_p}$ = single, $\boldsymbol{u}$ = double $\boldsymbol{u_p}$, $\boldsymbol{u}$ = double

https://github.com/dauzickaite/mpNystrom

3. When the computation naturally contains less sensitive subparts

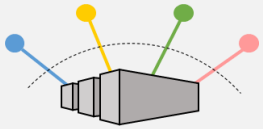


An Example

Consider computing the sum of double precision numbers $a + b$ in double precision with unit roundoff u, where $|b| \ll |a|$.

Then

$$fl(a + b) = (a + b)(1 + \delta), \qquad |\delta| \leq u.$$



An Example

Consider computing the sum of double precision numbers $a + b$ in double precision with unit roundoff u, where $|b| \ll |a|$.

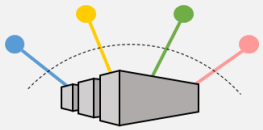
Then

$$fl(a + b) = (a + b)(1 + \delta), \qquad |\delta| \leq u.$$

Now consider rounding b to lower precision: $\tilde{b} = fl_{low}(b) = b(1 + \delta_{low})$.

Then

$$\begin{aligned} fl\left(a + \tilde{b}\right) &= (a + \tilde{b})(1 + \delta) \\ &= \left(a + b(1 + \delta_{low})\right)(1 + \delta) \\ &= (a + b)(1 + \delta)\left(1 + \frac{b}{a+b}\delta_{low}\right) \end{aligned}$$



An Example

Consider computing the sum of double precision numbers $a + b$ in double precision with unit roundoff u, where $|b| \ll |a|$.

Then

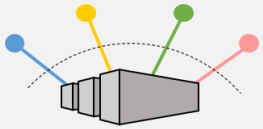
$$fl(a + b) = (a + b)(1 + \delta), \qquad |\delta| \leq u.$$

Now consider rounding b to lower precision: $\tilde{b} = fl_{low}(b) = b(1 + \delta_{low})$.

Then

$$\begin{aligned} fl\left(a + \tilde{b}\right) &= (a + \tilde{b})(1 + \delta) \\ &= \left(a + b(1 + \delta_{low})\right)(1 + \delta) \\ &= (a + b)(1 + \delta)\left(1 + \frac{b}{a+b}\delta_{low}\right) \end{aligned}$$

This term is insignificant if $|b|u_{low} \ll |a + b|u$!

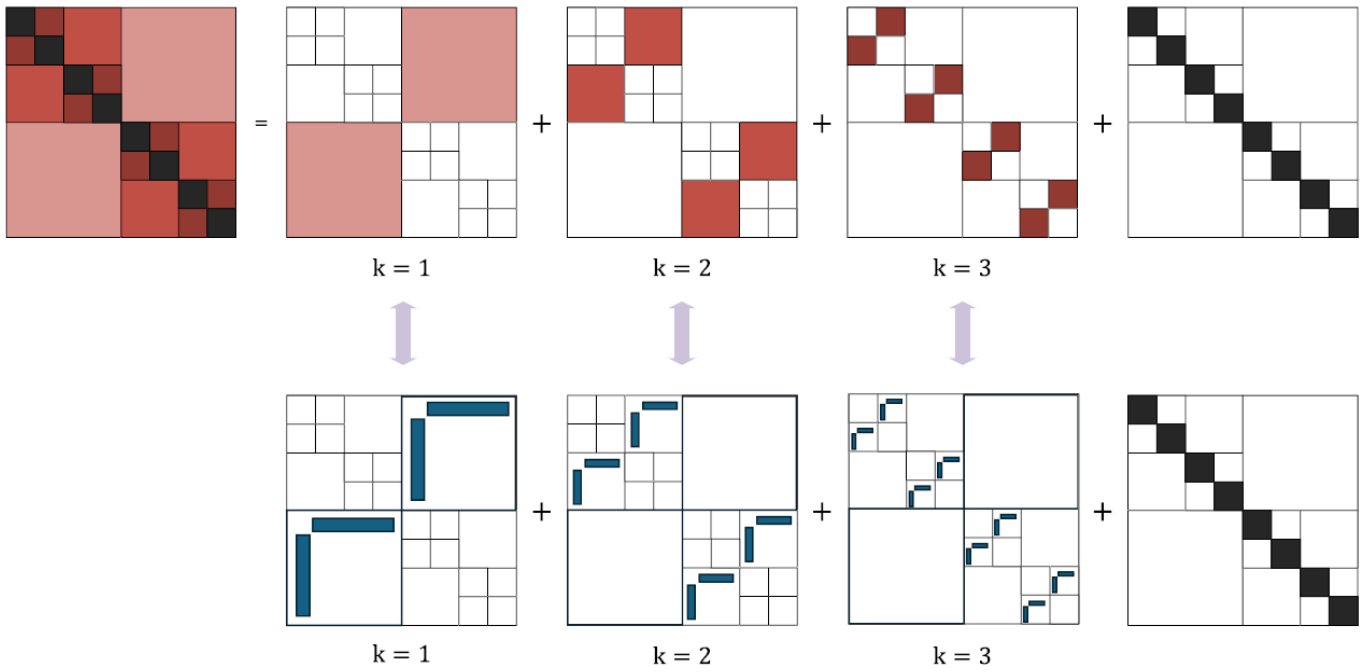


Sensitivity of subparts and adaptive algorithms

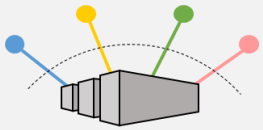
- Takeaway: computations performed on data of small magnitude need not use very high precision.

- Adaptive approaches: adapt precision to be inversely proportional to the weight of the data
  - Weight can be magnitude, norm, condition number, etc.

Example: HODLR Matrices



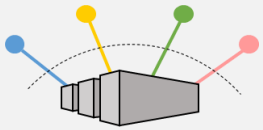
- Hierarchical Off-Diagonal Low-Rank (HODLR) matrices
- Fixed hierarchical block structure, low-rank (p) approximation of all off-diagonal blocks
- Enables efficient computation, e.g., $O(n^2)$ computations become $O(pn \log n)$



Mixed Precision HODLR Matrices

Let ε be the maximum relative error in the approximation of the off-diagonal blocks of a HODLR matrix H.

Let $\widehat{H}$ be the mixed precision representation of H where the off-diagonal blocks in level k are stored in precision $\boldsymbol{u_k}$ and the diagonal blocks in the final level are stored in a working precision $\boldsymbol{u} < \boldsymbol{u_k}$.



Mixed Precision HODLR Matrices

Let ε be the maximum relative error in the approximation of the off-diagonal blocks of a HODLR matrix H.

Let $\widehat{H}$ be the mixed precision representation of H where the off-diagonal blocks in level k are stored in precision $\boldsymbol{u_k}$ and the diagonal blocks in the final level are stored in a working precision $\boldsymbol{u} < \boldsymbol{u_k}$.

Theorem:

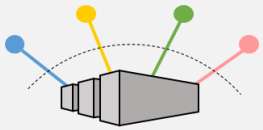
$$\frac{\|H - \widehat{H}\|_F}{\|H\|_F} \lesssim \left(2\sqrt{2} \left(\sum_{k=1}^{\ell} 2^k \xi_k^2 u_k^2 \right)^{1/2} + \varepsilon \right)$$

where

$$\xi_k = \frac{\max_{|i-j|=1} \left\| H_{ij}^{(k)} \right\|_F}{\|H\|_F}$$

and $\widetilde{H}_{ij}^{(k)}$, $|i-j| = 1$ denotes any off-diagonal block from the kth level.

[C., Chen, Liu, 2025]



Mixed Precision HODLR Matrices

Let ε be the maximum relative error in the approximation of the off-diagonal blocks of a HODLR matrix H.

Let $\widehat{H}$ be the mixed precision representation of H where the off-diagonal blocks in level k are stored in precision $\boldsymbol{u_k}$ and the diagonal blocks in the final level are stored in a working precision $\boldsymbol{u} < \boldsymbol{u_k}$.

Theorem:

$$\frac{\|H - \widehat{H}\|_F}{\|H\|_F} \lesssim \left(2\sqrt{2} \left(\sum_{k=1}^{\ell} 2^k \xi_k^2 u_k^2 \right)^{1/2} + \varepsilon \right)$$

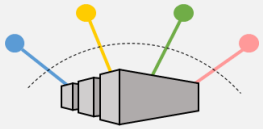
where

$$\xi_k = \frac{\max_{|i-j|=1} \left\| H_{ij}^{(k)} \right\|_F}{\|H\|_F}$$

→ **Relative (norm-based) weight** of off-diagonal blocks in level k

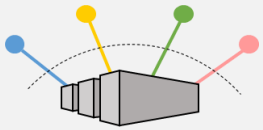
and $\widetilde{H}_{ij}^{(k)}$, $|i-j| = 1$ denotes any off-diagonal block from the kth level.

[C., Chen, Liu, 2025]



Mixed Precision HODLR Matrices

$$\frac{\|H-\widehat{H}\|_F}{\|H\|_F} \lesssim \left(2\sqrt{2}\left(\sum_{k=1}^{\ell} 2^k \xi_k^2 u_k^2\right)^{1/2} + \varepsilon \right)$$



Mixed Precision HODLR Matrices

$$\frac{\|H-\widehat{H}\|_F}{\|H\|_F} \lesssim \left(2\sqrt{2}\left(\sum_{k=1}^{\ell} 2^k \xi_k^2 u_k^2 \right)^{1/2} + \varepsilon \right)$$

If we choose

$$u_k \leq \frac{\varepsilon}{2^{k/2}\xi_k}$$

then the bound becomes

$$\frac{\|H-\widehat{H}\|_F}{\|H\|_F} \lesssim \left(2\sqrt{2\ell}+1\right)\varepsilon.$$

Mixed Precision HODLR Matrices

$$\frac{\|H-\widehat{H}\|_F}{\|H\|_F} \lesssim \left(2\sqrt{2}\left(\sum_{k=1}^{\ell} 2^k \xi_k^2 u_k^2\right)^{1/2} + \varepsilon\right)$$

If we choose

$$u_k \leq \frac{\varepsilon}{2^{k/2}\xi_k}$$

→ **Less important blocks (smaller ξ_k) can be stored in lower precision (larger u_k)**

then the bound becomes

$$\frac{\|H-\widehat{H}\|_F}{\|H\|_F} \lesssim \left(2\sqrt{2\ell}+1\right)\varepsilon.$$

Mixed Precision HODLR Matrices

$$\frac{\|H-\widehat{H}\|_F}{\|H\|_F} \lesssim \left(2\sqrt{2}\left(\sum_{k=1}^{\ell} 2^k \xi_k^2 u_k^2\right)^{1/2} + \varepsilon\right)$$

If we choose

$$u_k \leq \frac{\varepsilon}{2^{k/2}\xi_k}$$

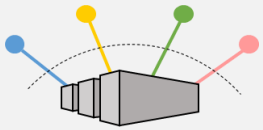
Adaptive precision approach: Store off-diagonal blocks in different precisions based on their relative norm

then the bound becomes

$$\frac{\|H-\widehat{H}\|_F}{\|H\|_F} \lesssim \left(2\sqrt{2\ell}+1\right)\varepsilon.$$

4. When low precision hardware can efficiently emulate high precision

Question:
How much faster is FP8 TC than FP64 TC on NVIDIA Blackwell?



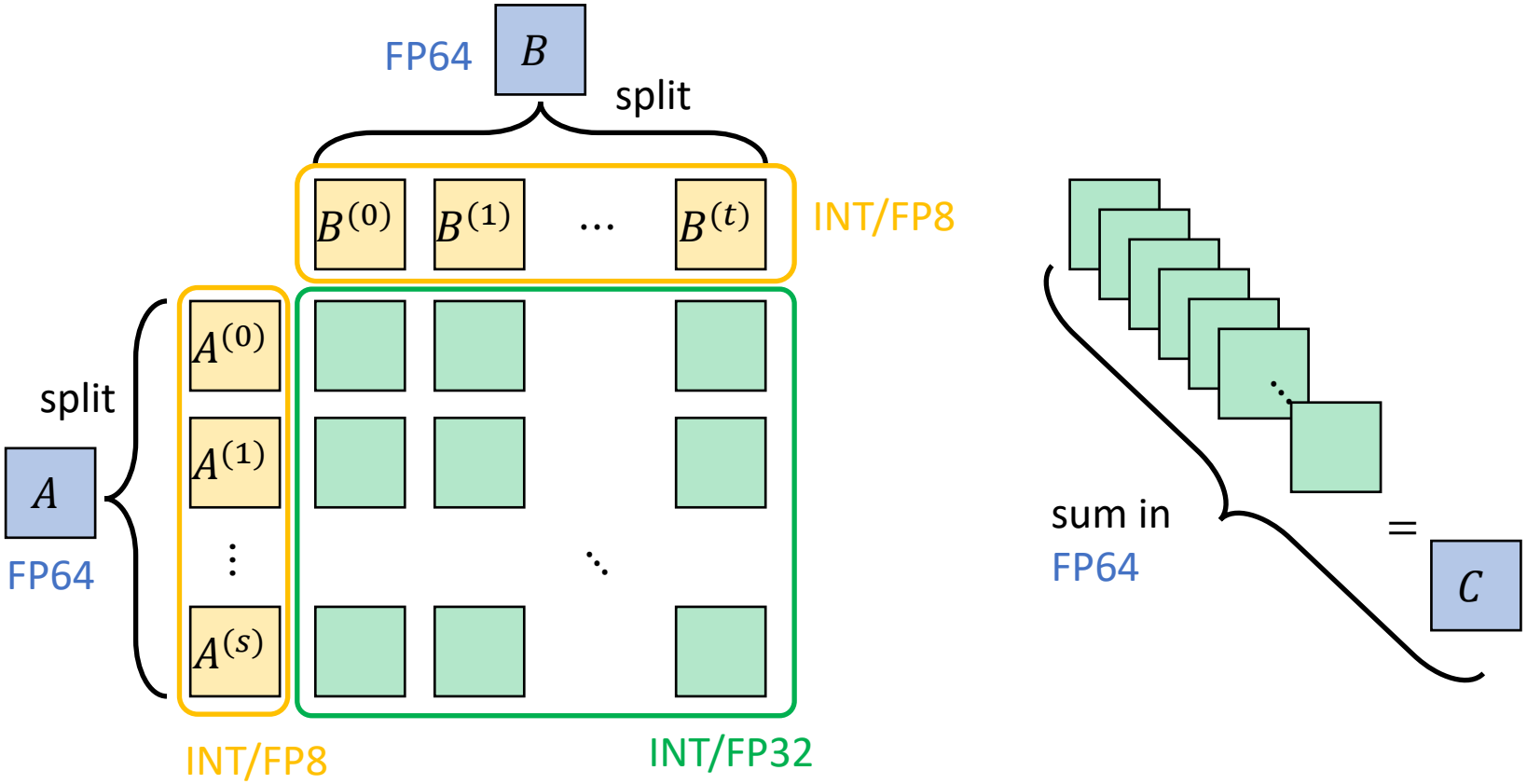
TFLOPs/TOPs for recent GPUs

	RTX 4090	H200	B200
FP64	1.29	34	40
FP64 TC	-	67	40
FP32	82.6	67	80
TF32 TC	82.6	494	1100
BF16 TC	165.2	989	2250
FP16 TC	165.2	989	2250
INT8 TC	660.6	1979	4500
FP8 TC	660.6	1979	4500
FP6 TC	-	-	4500
FP4 TC	-	-	9000

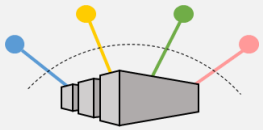
8-bit formats are 112x faster than 64-bit formats!

⇒ **Emulate** higher precision with lower precision!

The Ozaki Scheme



[Ozaki, Ogita, Oishi, Rump, 2012], [Mukunoki, Ozaki, Ogita, Imamura, 2020], [Ootomo, Ozaki, Yokota, 2024], [Uchino, Ozaki, Imamura, 2025], [Ozaki, Uchino, Imamura, 2025]



The Ozaki Scheme

Given $A \in \mathbb{F}^{p\times q}$, $B \in \mathbb{F}^{q\times r}$.

Write the splitting

$$A \approx D_1^{-1}D_1A_1 + D_2^{-1}D_2A_2 + \cdots + D_k^{-1}D_kA_k,$$
$$B \approx B_1E_1E_1^{-1} + B_2E_2E_2^{-1} + \cdots + B_kE_kE_k^{-1},$$

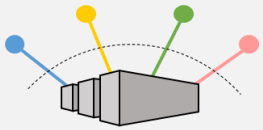
where $D_i \in \mathbb{F}^{p\times p}$ and $E_i \in \mathbb{F}^{r\times r}$ are diagonal matrices with entries powers of two,

Where all elements in D_iA_i and B_iE_i for all $1 \le i \le k$ can be represented in INT8.

$\Rightarrow (D_iA_i)(B_jE_j)$ can be computed without rounding error in INT8-TC.

Approximation:

$$\tilde{C} = fl\left(\sum_{i+j\le k+1} D_i^{-1}\left((D_iA_i)(B_jE_j)\right)E_j^{-1}\right)$$



The Ozaki Scheme

Given $A \in \mathbb{F}^{p\times q}$, $B \in \mathbb{F}^{q\times r}$.

Write the splitting

$$A \approx D_1^{-1}D_1A_1 + D_2^{-1}D_2A_2 + \cdots + D_k^{-1}D_kA_k,$$
$$B \approx B_1E_1E_1^{-1} + B_2E_2E_2^{-1} + \cdots + B_kE_kE_k^{-1},$$

where $D_i \in \mathbb{F}^{p\times p}$ and $E_i \in \mathbb{F}^{r\times r}$ are diagonal matrices with entries powers of two,

Where all elements in D_iA_i and B_iE_i for all $1 \le i \le k$ can be represented in INT8.

$\Rightarrow (D_iA_i)(B_jE_j)$ can be computed without rounding error in INT8-TC.

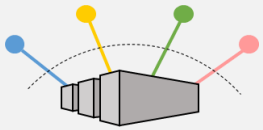
Approximation:

$$\tilde{C} = fl\left(\sum_{i+j\le k+1} D_i^{-1}\left((D_iA_i)(B_jE_j)\right)E_j^{-1}\right)$$

Step 1: Split matrices into k slices; $O(pq) + O(qr)$

Step 2: Compute $k(k+1)/2$ matrix products using INT8-TC; $k(k+1)pqr + O(pr)$

Step 3: Sum up computed products in FP64; $O(pr)$



The Ozaki Scheme

Given $A \in \mathbb{F}^{p\times q}$, $B \in \mathbb{F}^{q\times r}$.

Write the splitting

$$A \approx D_1^{-1}D_1A_1 + D_2^{-1}D_2A_2 + \cdots + D_k^{-1}D_kA_k,$$
$$B \approx B_1E_1E_1^{-1} + B_2E_2E_2^{-1} + \cdots + B_kE_kE_k^{-1},$$

where $D_i \in \mathbb{F}^{p\times p}$ and $E_i \in \mathbb{F}^{r\times r}$ are diagonal matrices with entries powers of two,

Where all elements in D_iA_i and B_iE_i for all $1 \le i \le k$ can be represented in INT8.

$\Rightarrow (D_iA_i)(B_jE_j)$ can be computed without rounding error in INT8-TC.

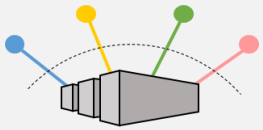
Approximation:

$$\tilde{C} = fl\left(\sum_{i+j\le k+1} D_i^{-1}\left((D_iA_i)(B_jE_j)\right)E_j^{-1}\right)$$

Step 1: Split matrices into k slices; $O(pq) + O(qr)$

Step 2: Compute $k(k+1)/2$ matrix products using INT8-TC; $k(k+1)pqr + O(pr)$

Step 3: Sum up computed products in FP64; $O(pr)$



The Ozaki Scheme

Given $A \in \mathbb{F}^{p\times q}$, $B \in \mathbb{F}^{q\times r}$.

Write the splitting

$$A \approx D_1^{-1}D_1A_1 + D_2^{-1}D_2A_2 + \cdots + D_k^{-1}D_kA_k,$$
$$B \approx B_1E_1E_1^{-1} + B_2E_2E_2^{-1} + \cdots + B_kE_kE_k^{-1},$$

where $D_i \in \mathbb{F}^{p\times p}$ and $E_i \in \mathbb{F}^{r\times r}$ are diagonal matrices with entries powers of two,

Where all elements in D_iA_i and B_iE_i for all $1 \le i \le k$ can be represented in INT8.

$\Rightarrow (D_iA_i)(B_jE_j)$ can be computed without rounding error in INT8-TC.

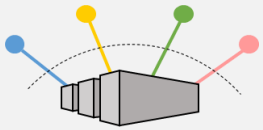
Approximation:

$$\tilde{C} = fl\left(\sum_{i+j\le k+1} D_i^{-1}\left((D_iA_i)(B_jE_j)\right)E_j^{-1}\right)$$

Step 1: Split matrices into k slices; $O(pq) + O(qr)$

Step 2: Compute $k(k+1)/2$ matrix products using INT8-TC; $k(k+1)pqr + O(pr)$

Step 3: Sum up computed products in FP64; $O(pr)$



The Ozaki Scheme

Given $A \in \mathbb{F}^{p\times q}$, $B \in \mathbb{F}^{q\times r}$.

Write the splitting

$$A \approx D_1^{-1}D_1A_1 + D_2^{-1}D_2A_2 + \cdots + D_k^{-1}D_kA_k,$$
$$B \approx B_1E_1E_1^{-1} + B_2E_2E_2^{-1} + \cdots + B_kE_kE_k^{-1},$$

where $D_i \in \mathbb{F}^{p\times p}$ and $E_i \in \mathbb{F}^{r\times r}$ are diagonal matrices with entries powers of two,

Where all elements in D_iA_i and B_iE_i for all $1 \le i \le k$ can be represented in INT8.

$\Rightarrow (D_iA_i)(B_jE_j)$ can be computed without rounding error in INT8-TC.

Approximation:

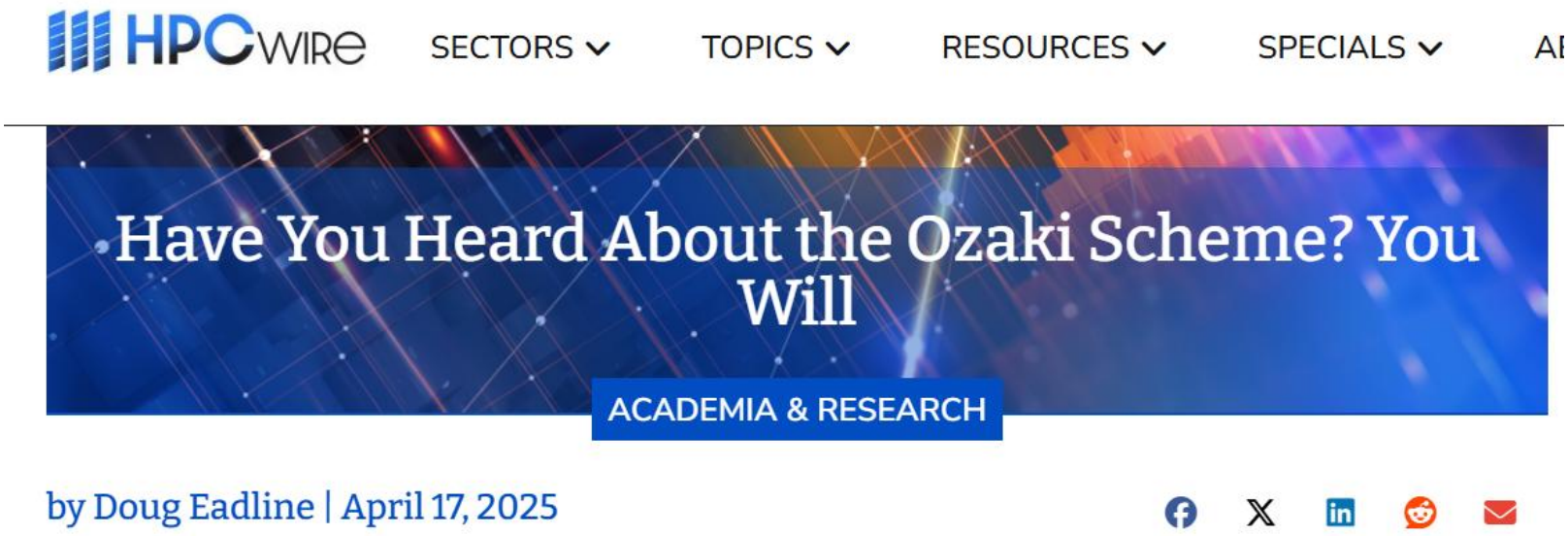
$$\tilde{C} = fl\left(\sum_{i+j\le k+1} D_i^{-1}\left((D_iA_i)(B_jE_j)\right)E_j^{-1}\right)$$

Step 1: Split matrices into k slices; $O(pq) + O(qr)$

Step 2: Compute $k(k+1)/2$ matrix products using INT8-TC; $k(k+1)pqr + O(pr)$

Step 3: Sum up computed products in FP64; $O(pr)$

The Ozaki Scheme



- HPL using Ozaki scheme achieves 2.3x speedup versus DGEMM [Dongarra et al., 2025]
- Used in NVIDIA HPC-Benchmarks 25.04

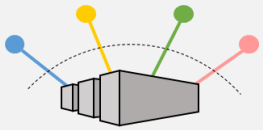
NVIDIA Rubin GPU (late 2026)

Specifications[1]

NVFP4 Inference	50 PFLOPS
NVFP4 Training[2]	35 PFLOPS
FP8/FP6 Training[2]	17.5 PFLOPS
INT8[2]	0.25 POPS
FP16/BF16[2]	4 PFLOPS
TF32[2]	2 PFLOPS
FP32	130 TFLOPS
FP64	33 TFLOPS
FP32 SGEMM[3]	400 TFLOPS
FP64 DGEMM[3]	200 TFLOPS

1. Preliminary information. All values are up to and subject to change.
2. Dense specification.
3. Peak performance using Tensor Core-based emulation algorithms.

https://www.nvidia.com/en-us/data-center/vera-rubin-nvl72/



NVIDIA Rubin GPU (late 2026)

Specifications[1]

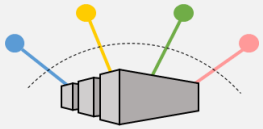
NVFP4 Inference	50 PFLOPS
NVFP4 Training[2]	35 PFLOPS
FP8/FP6 Training[2]	17.5 PFLOPS
INT8[2]	0.25 POPS
FP16/BF16[2]	4 PFLOPS
TF32[2]	2 PFLOPS
FP32	130 TFLOPS
FP64	33 TFLOPS
FP32 SGEMM[3]	400 TFLOPS
FP64 DGEMM[3]	200 TFLOPS

1. Preliminary information. All values are up to and subject to change.
2. Dense specification.
3. Peak performance using Tensor Core-based emulation algorithms.

Important point: Ozaki does not satisfy componentwise error bounds for GEMM!

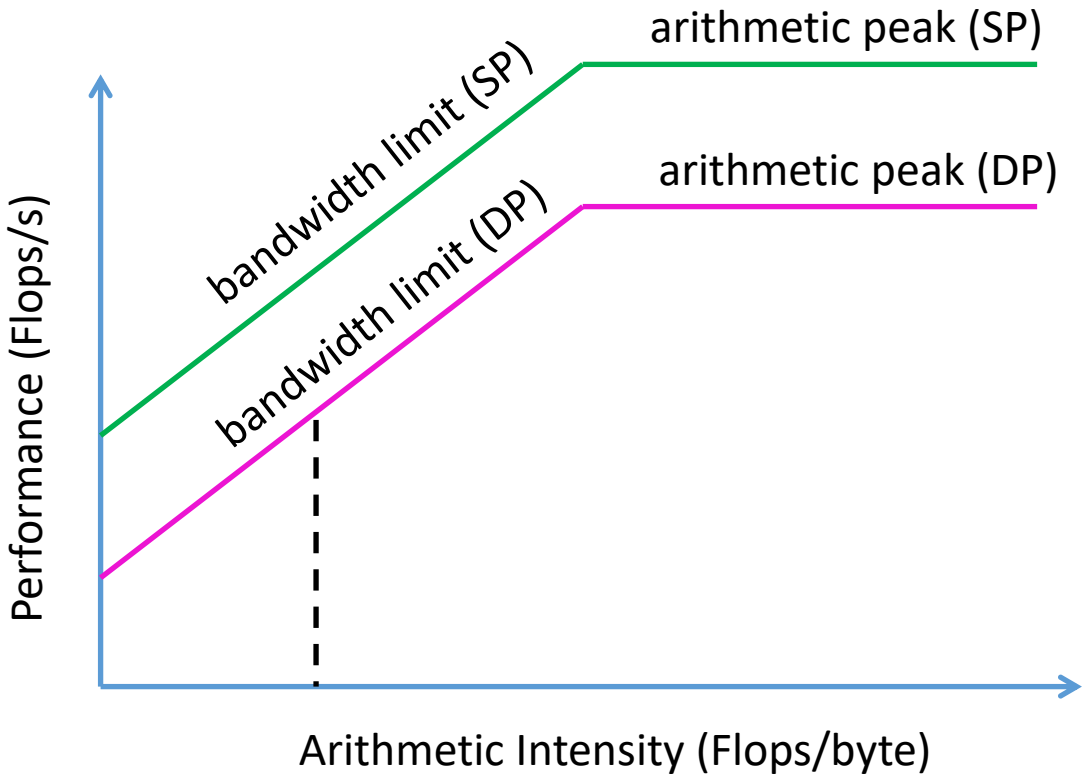
https://www.nvidia.com/en-us/data-center/vera-rubin-nvl72/

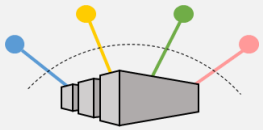
5. When data transfers are the bottleneck



The Roofline Model

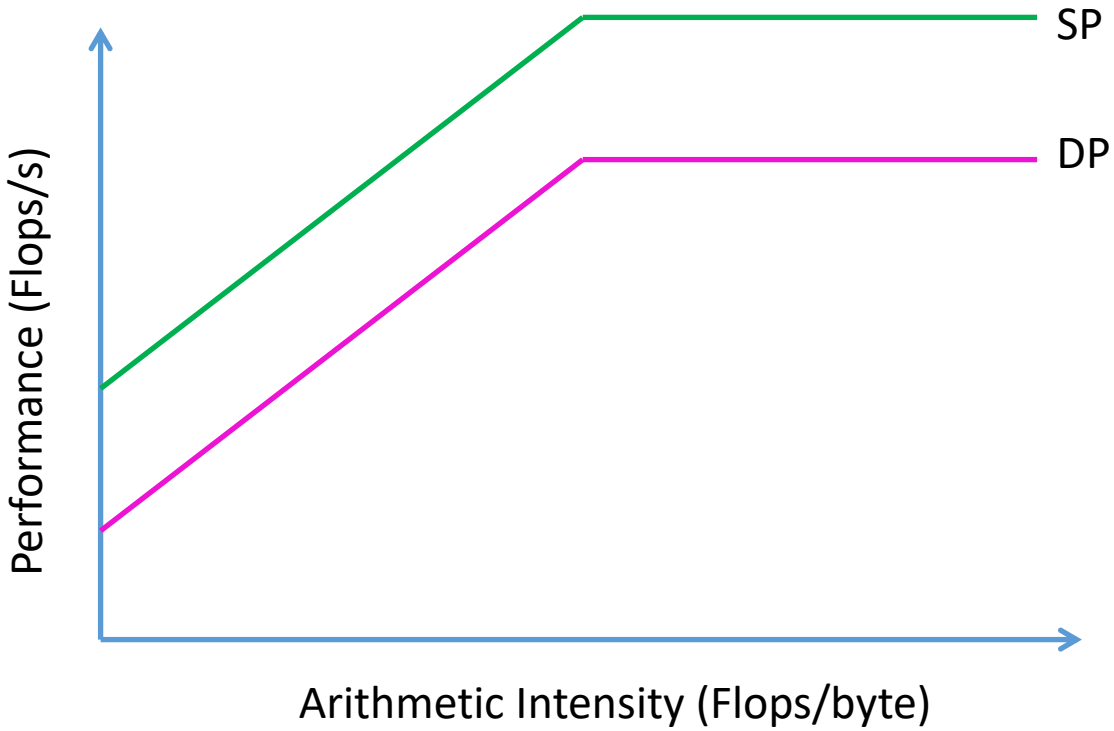
[Williams, Waterman, Patterson, 2009]

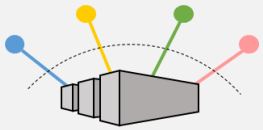




Idea: Decouple Storage and Compute Precision

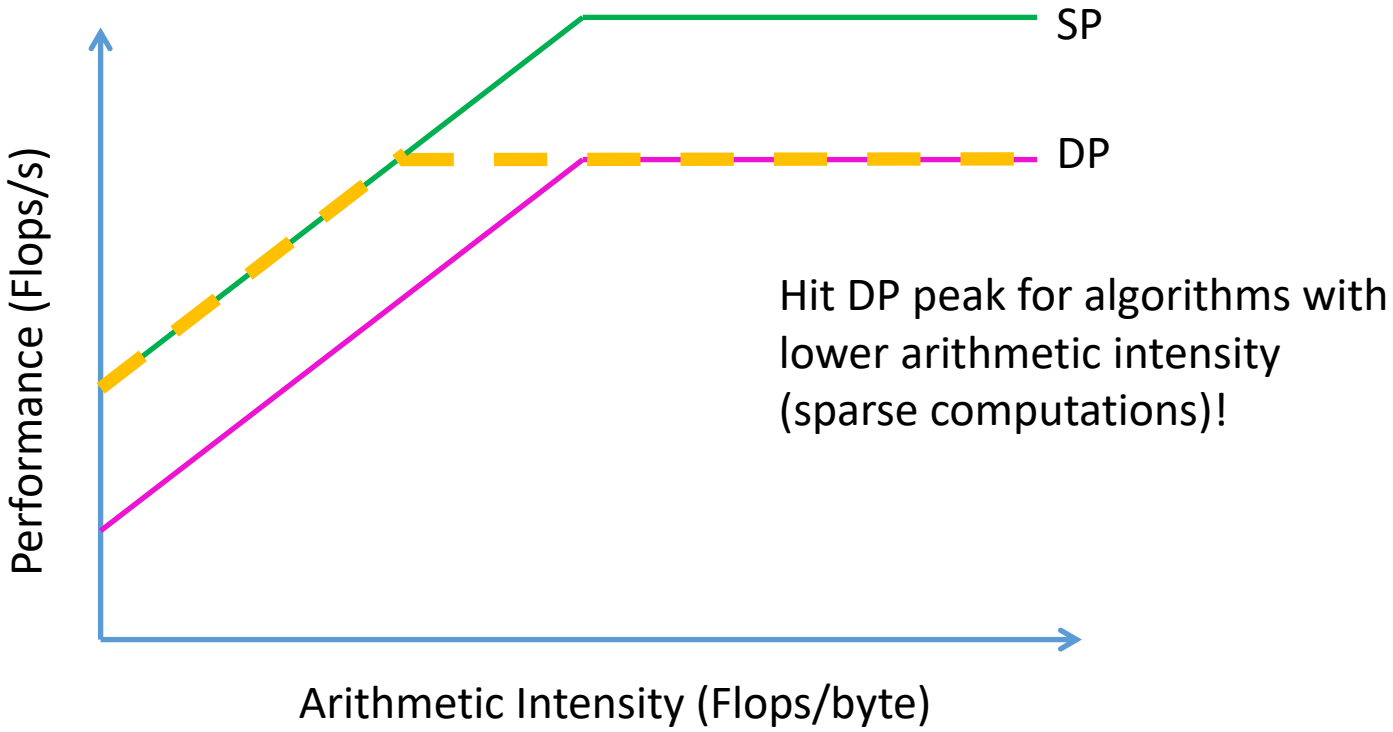
Compute in higher precision, store in lower precision
⇒ Same flops; move fewer bytes





Idea: Decouple Storage and Compute Precision

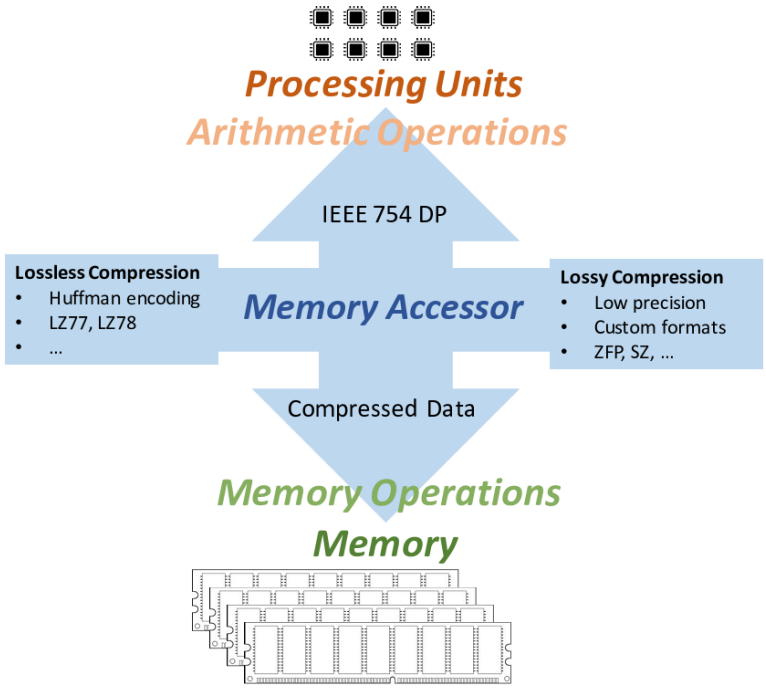
Compute in higher precision, store in lower precision
⇒ Same flops; move fewer bytes



Memory Accessor

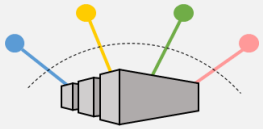
[Anzt, Flegar, Grützmacher, Quintana-Ortí, 2019]

- Store and access data in low precision, computations in higher precision
- Memory Accessor: Access data in low precision and convert to high precision on-the-fly
- Note: enables use of custom number formats!
- Implemented in Ginkgo library



- Example application: Compressed Basis GMRES, up to 50% performance improvement over standard GMRES [Aliaga, Anzt, Grützmacher, Quintana-Ortí, Tomás, 2023]

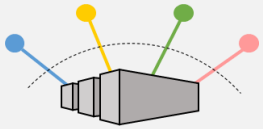
- Other references: [Grützmacher, Anzt, Quintana-Ortí, 2021], [Mukunoki, Kawai, Imamura, 2023], [Kriemann, 2023, 2024], [Amestoy, Jego, L'Excellent, Mary, Pichon, 2025]



Other Recent and Ongoing Work

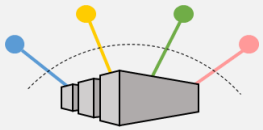
- Mixed precision iterative refinement for least-squares problems
  - 3-precision variant [C., Higham, Pranesh, 2020]
  - With randomized preconditioners [C., Daužickaitė, 2024]
  - For weighted least squares [C., Oktay, 2024]
  - A comparison of approaches [C., Daužickaitė, 2025]
- Mixed precision RQI for total least-squares problems [C., Oktay, 2023]
- Constructing sparse approximate inverses [Khan, C., 2025], [C., Khan, 2023]
- Mixed precision Krylov subspace methods
  - Four-precision preconditioned Flexible GMRES [C., Daužickaitė, 2024]
  - Preconditioned CG: [Bake, C., Ma, 2025]
  - Using extra precision to regain stability in s-step Krylov subspace methods [C., Gergelits, Yamazaki, 2022], [Yamazaki, C., Kelley, 2022]
- Mixed precision multigrid methods [C., Vacek, Tsai, Anzt, Kohl, Rüde, 2025]
- Mixed precision multilevel sampling methods [Martínek, C., Scheichl, 2025]

Surveys on mixed precision NLA: [Abdelfattah et al., IJHPC, 2021], [Higham and Mary, 2022]
Topic of today's lecture: [C., Mary, 2025, SIAM News]



Looking Forward

- Era of extreme heterogeneity (accelerators)
  - New (IEEE and non-IEEE number) formats
    - IEEE P3109 standard (400 possible formats); OCP standard
  - Emulation? Stochastic rounding?
  - *How can we do meaningful error analysis?*



Looking Forward

- Era of extreme heterogeneity (accelerators)
  - New (IEEE and non-IEEE number) formats
    - IEEE P3109 standard (400 possible formats); OCP standard
  - Emulation? Stochastic rounding?
  - *How can we do meaningful error analysis?*
- Emerging hardware – approximate hardware, quantum, neuromorphic, etc.

- Overall goal:
  - Develop **tight error bounds** to rigorously determine how many bits each part of a given computation **needs and deserves**

Thank You!

carson@karlin.mff.cuni.cz
www.karlin.mff.cuni.cz/~carson/