

# Mixed-Precision Computing: High Accuracy with Low Precision

Erin C. Carson  
Faculty of Mathematics and Physics  
Charles University

HPCSE 2026  
May 21, 2026

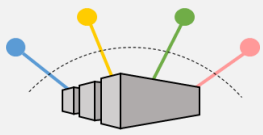


FACULTY  
OF MATHEMATICS  
AND PHYSICS  
Charles University



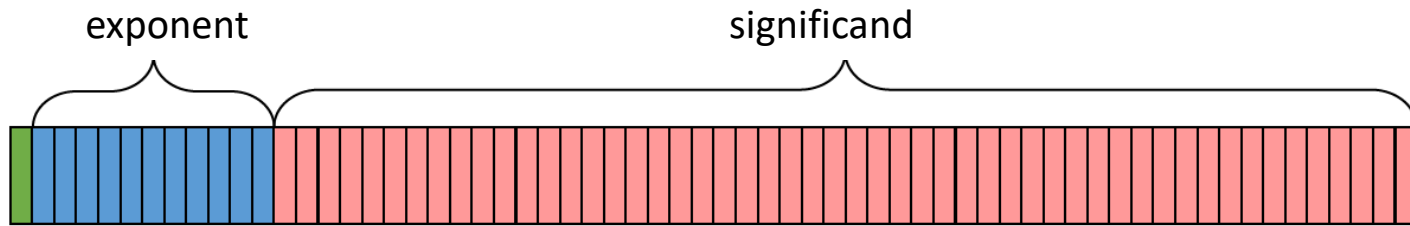
Co-funded by the  
European Union

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.



# Floating Point Numbers

$$(-1)^{\text{sign}} \times 2^{(\text{exponent}-\text{offset})} \times 1.\text{significantand}$$



IEEE 754 standard: double precision: 1 sign bit, 11 exponent bits, 52(+1) significand bits;  
offset = 1023

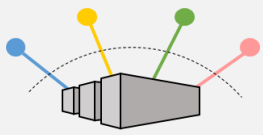
Largest representable number:  $2^{(2^{\#\text{exponent bits}}-1)-1023} = 2^{2047-1023} = 2^{1024} \approx 2 \times 10^{308}$

Smallest (normal) representable number:  $2^{1-1023} = 2^{-1022} \approx 2 \times 10^{-308}$

Unit roundoff:  $u = 2^{-\#\text{significand bits}} = 2^{-53} \approx 10^{-16}$

Standard model of floating point arithmetic:

$$fl(x \text{ op } y) = (x \text{ op } y)(1 + \delta), \quad |\delta| \leq u, \quad \text{op} = +, -, *, /$$

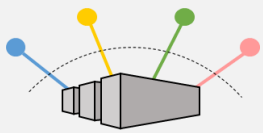


# The Exascale Era

We are in the “Exascale Era”

- $10^{18}$  floating point operations per second

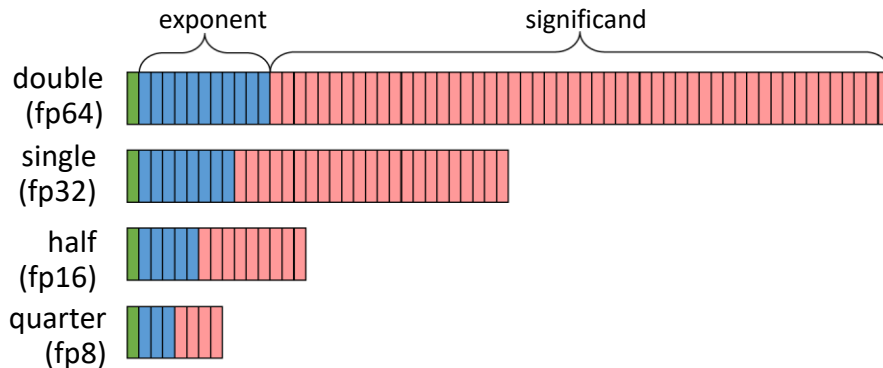




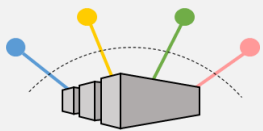
# Exascale Hardware



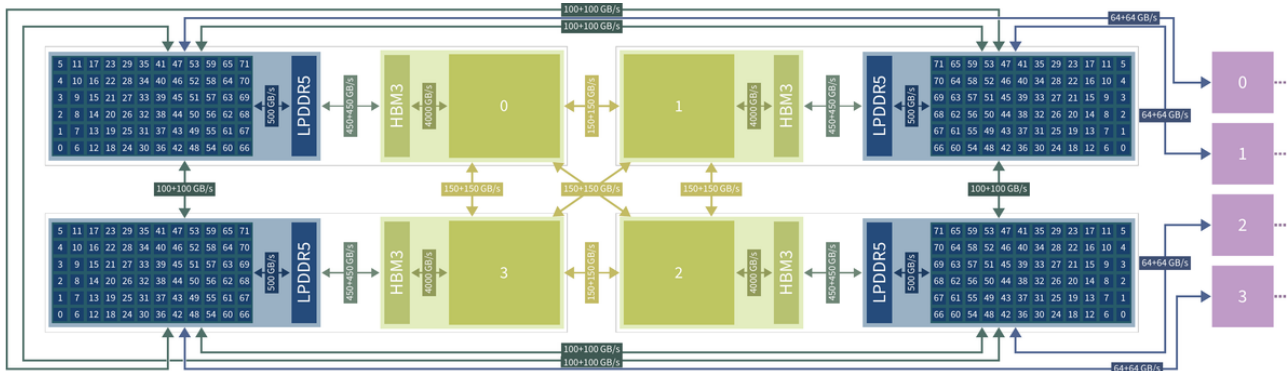
<https://www.fz-juelich.de/en/ias/jsc/jupiter/tech>



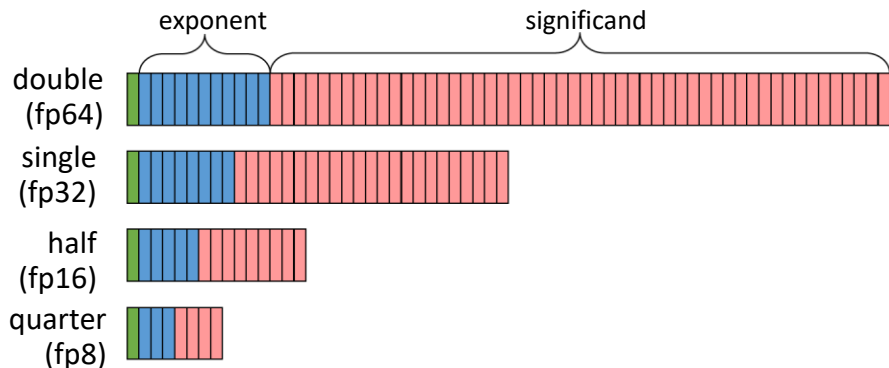
	size (bits)	range	$u$	perf. (NVIDIA H100 TC)
fp64	64	$10^{\pm 308}$	$1 \times 10^{-16}$	67 Tflops/s
fp32	32	$10^{\pm 38}$	$6 \times 10^{-8}$	989 Tflop/s
fp16	16	$10^{\pm 5}$	$5 \times 10^{-4}$	1979 Tflops/s
bfloat16	16	$10^{\pm 38}$	$4 \times 10^{-3}$	
fp8-e5m2	8	$10^{\pm 5}$	$1 \times 10^{-1}$	3958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



# Exascale Hardware



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



## HPL-MxP

NUMBER 1 SYSTEM

1

### El Capitan

LLNL,  
US

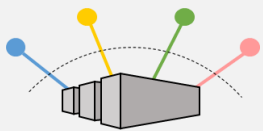
ACHIEVED

16.7 Eflops/s

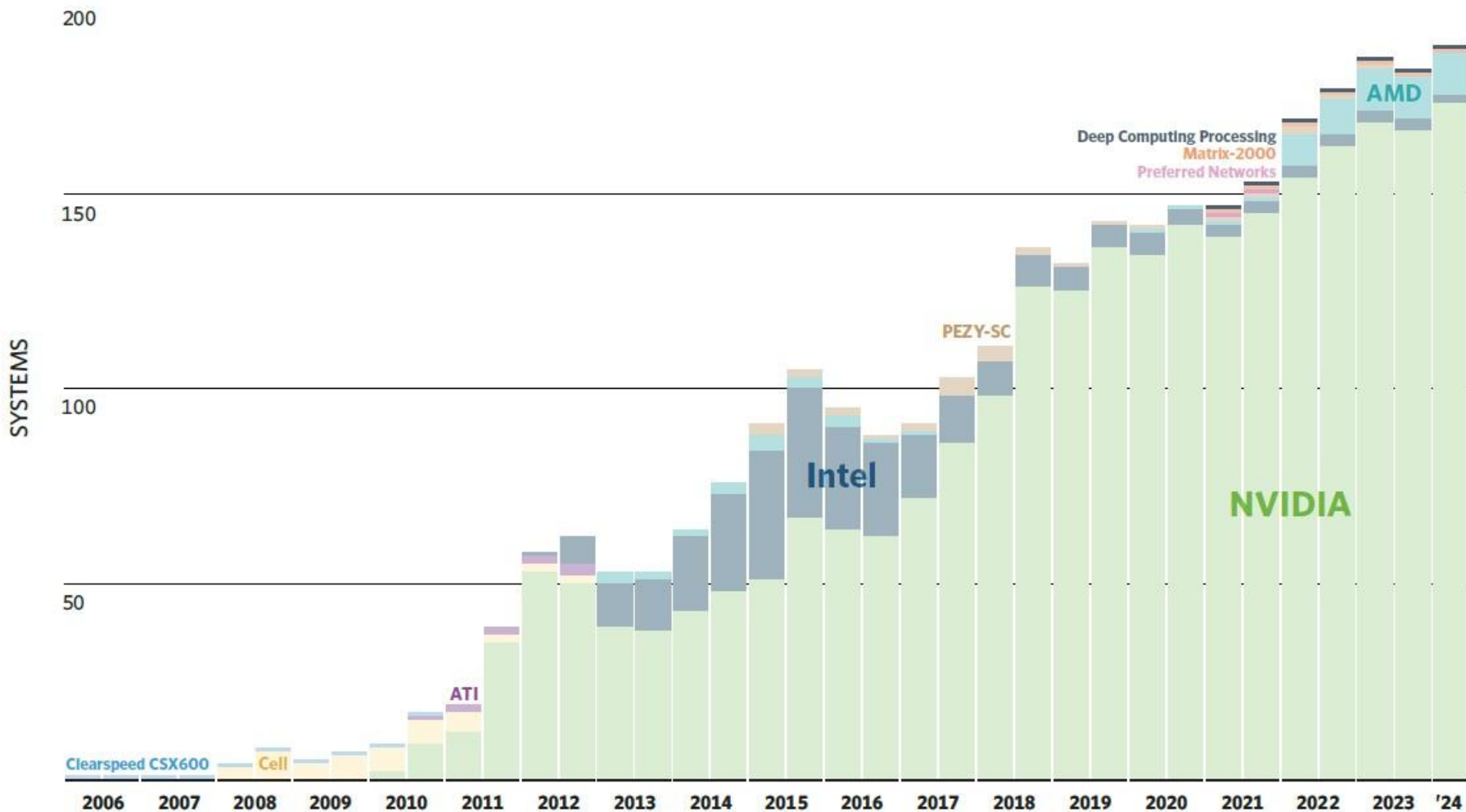
Jack Dongarra

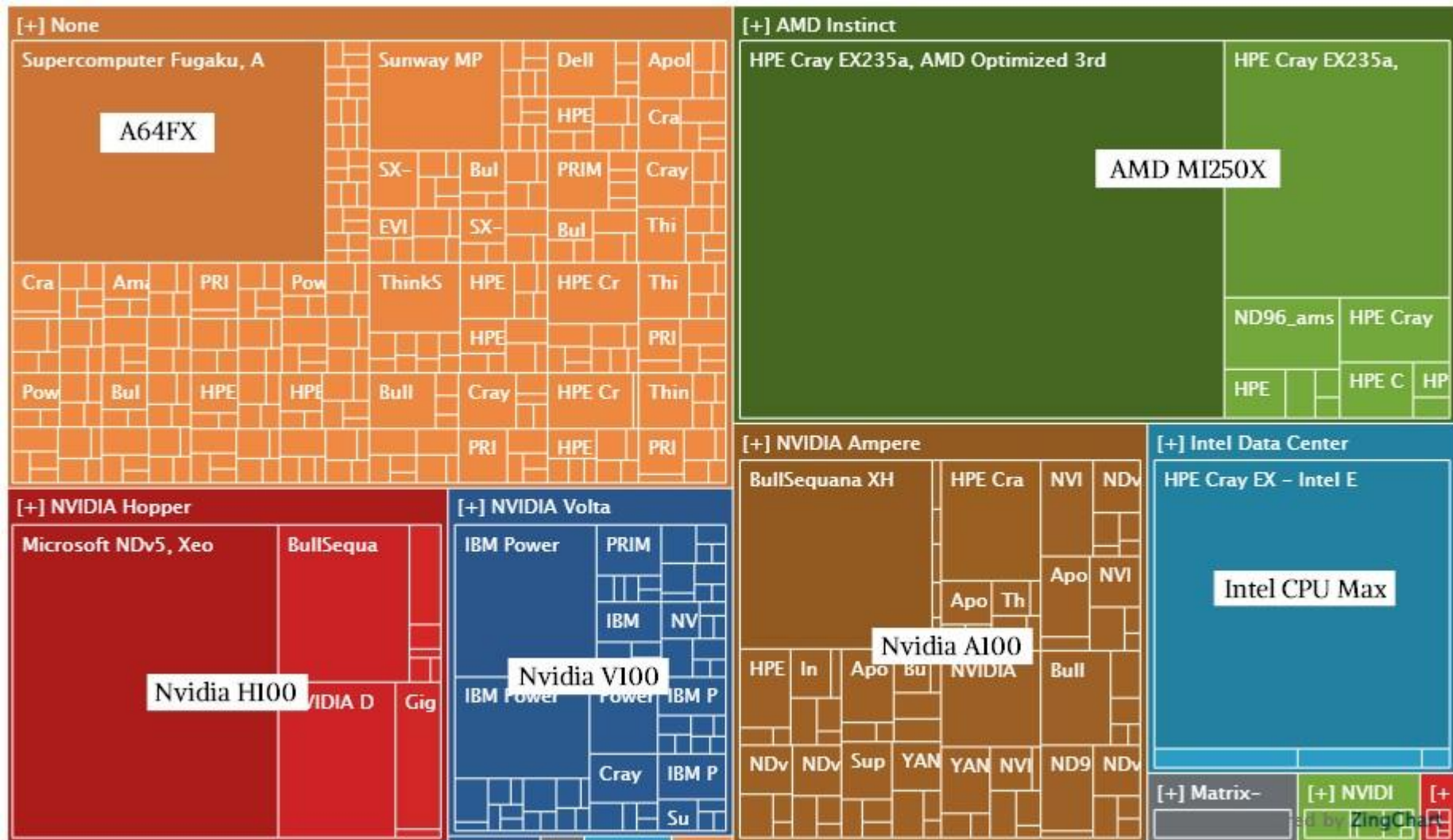
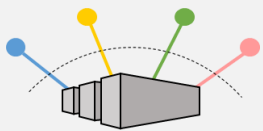
Piotr Luszczek

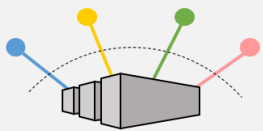
ICL  
INNOVATIVE  
FOR THE ENERGY



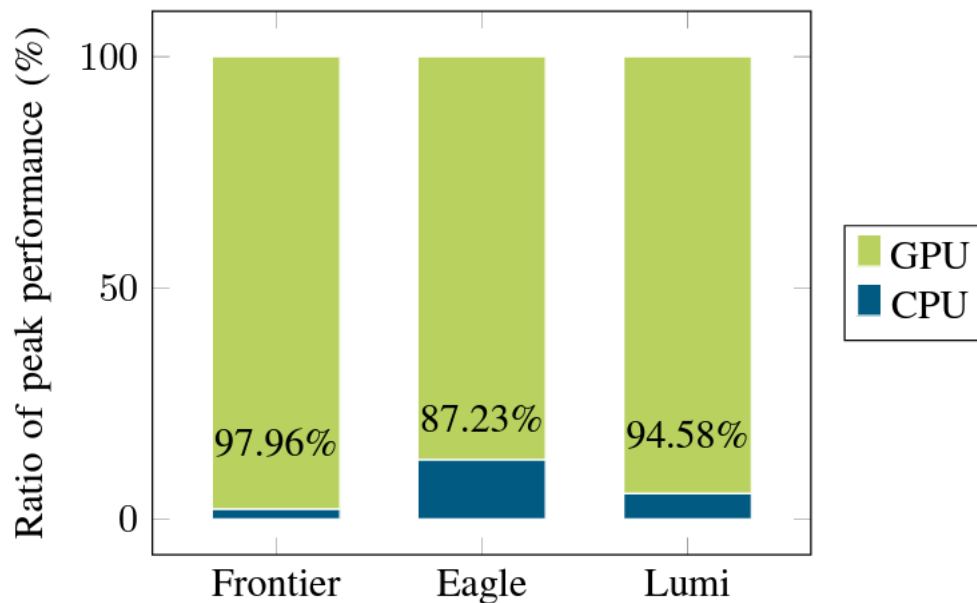
# Growth of Accelerated Systems





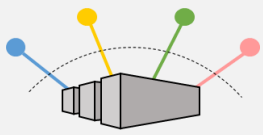


# Performance comes from Accelerators



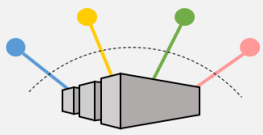
We MUST develop hardware-aware algorithms for matrix computations that can take advantage of mixed precision capabilities!

# Challenges of Using Low/Mixed Precision



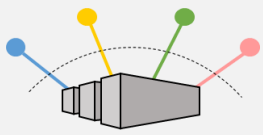
# 1. Need for Descriptive Bounds and Adaptive Approaches

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



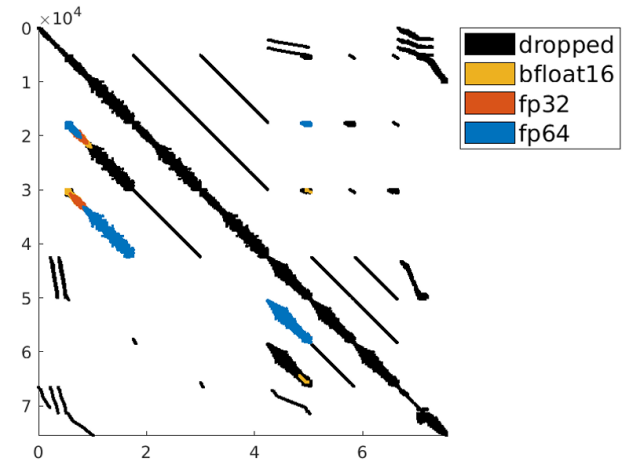
# 1. Need for Descriptive Bounds and Adaptive Approaches

- 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**

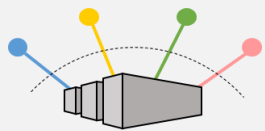


# 1. Need for Descriptive Bounds and Adaptive Approaches

- 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

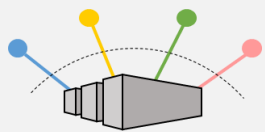
- Error bounds derived the typical way involve constraints like  $nu < 1$  or  $\kappa(A)u < 1$

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  $\epsilon$  ( $\sigma$  means singular value):

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

propose the first provably stable one-synchronization reorthogonalized BCGS variant, demonstrating that it has  $O(\mathbf{u})$  loss of orthogonality under the condition  $O(\mathbf{u})\kappa^2(\mathcal{X}) \leq 1/2$ , 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  $\kappa(A)u < 1$

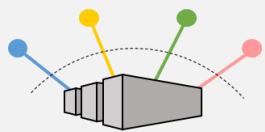
to show that the MGS-GMRES method does eventually produce a backward stable approximate solution when applied to any member of the following class of linear systems with floating point arithmetic unit roundoff  $\epsilon$  ( $\sigma$  means singular value):

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

propose the first provably stable one-synchronization reorthogonalized BCGS variant, demonstrating that it has  $O(\mathbf{u})$  loss of orthogonality under the condition  $O(\mathbf{u})\kappa^2(\mathcal{X}) \leq 1/2$ , 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  $\epsilon$  ( $\sigma$  means singular value):

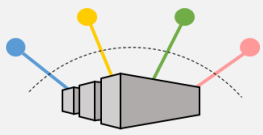
$$(1.1) \quad Ax = b \neq 0, \quad A \in \mathbf{R}^{n \times n}, \quad b \in \mathbf{R}^n, \quad \sigma_{\min}(A) \gg n^2 \epsilon \|A\|_F;$$

propose the first provably stable one-synchronization reorthogonalized BCGS variant, demonstrating that it has  $O(\mathbf{u})$  loss of orthogonality under the condition  $O(\mathbf{u})\kappa^2(\mathcal{X}) \leq 1/2$ , 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  $ku < 1$ .

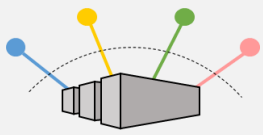
- Is this reality or just an effect of worst-case analysis?
- One solution: probabilistic approach [Higham, Mary, 2019], [Higham, Mary, 2020]

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



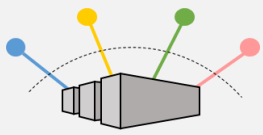
## 3. Working with a Limited Range

- We usually assume that no overflow or underflow occurs
  - Overflow: large magnitude values become  $\pm\infty$ 
    - Complete failure
  - Underflow: small magnitude values become 0
    - Potential loss of important numerical properties (e.g., nonsingularity, positive definiteness)



## 3. Working with a Limited Range

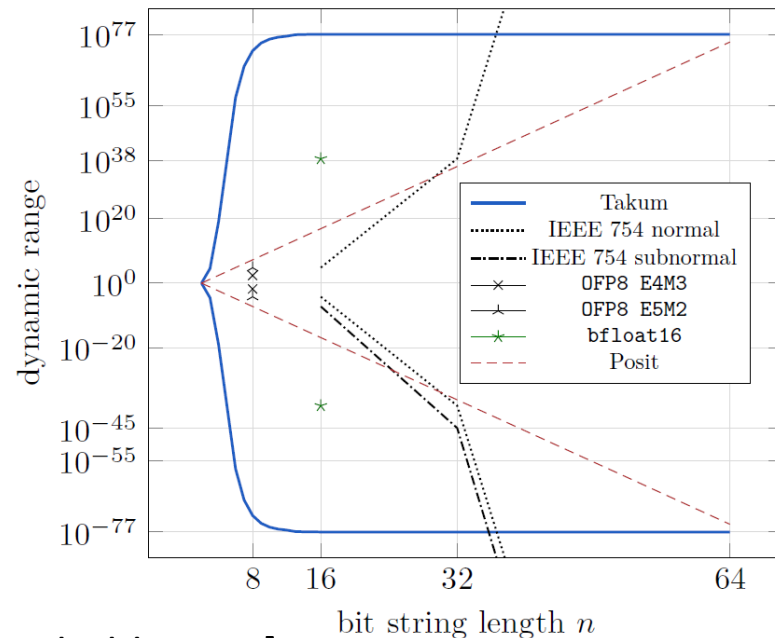
- We usually assume that no overflow or underflow occurs
  - Overflow: large magnitude values become  $\pm\infty$ 
    - Complete failure
  - Underflow: small magnitude values become 0
    - Potential loss of important numerical properties (e.g., nonsingularity, positive definiteness)
- Maybe reasonable 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: large magnitude values become  $\pm\infty$ 
    - Complete failure
  - Underflow: small magnitude values become 0
    - 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  $10^{\pm 5}$ )

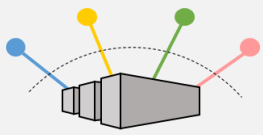
- Some solutions:
  - Scaling and shifting approach [Higham, Pranesh, 2019]
  - Rigorous error handling [Scott, Tuma, 2025]
  - New number formats, e.g., bfloat16, takum, block floating point



[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

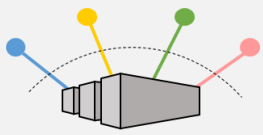
Solve  $Ax_0 = b$  by LU factorization precision  $u$

for  $i = 0$ : maxit

$$r_i = b - Ax_i \quad \text{precision } u^2$$

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

$$x_{i+1} = x_i + d_i \quad \text{precision } u$$



# 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

$$r_i = b - Ax_i \quad \text{precision } u^2$$

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

$$x_{i+1} = x_i + d_i \quad \text{precision } u$$

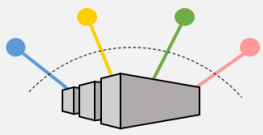
Theorem [Wilkinson]: If

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

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  $u^2$

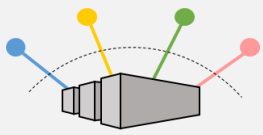
Solve  $Ad_i = r_i$  via GMRES on  $U^{-1}L^{-1}Ad_i = U^{-1}L^{-1}r_i$

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

Theorem: If

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

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



# Mixed Precision Iterative Refinement

[C., Higham, 2018]

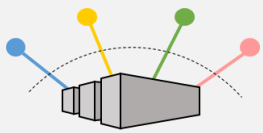
**Extended and improved analysis to allow for 3 potentially different precisions**

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

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



# Mixed Precision Iterative Refinement

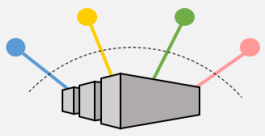
**HPL-MxP**

[PEOPLE](#) [LINKS](#) [PUBLICATIONS](#) [COLLABORATORS](#) [RULES](#) [SOFTWARE](#) [RESULTS](#) [UPLOAD](#)

## HPL-MXP MIXED-PRECISION BENCHMARK

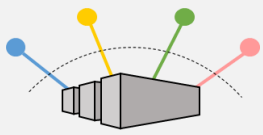
The HPL-MxP benchmark seeks to highlight the emerging convergence of high-performance computing (HPC) and artificial intelligence (AI) workloads. While traditional HPC focused on simulation runs for modeling phenomena in physics, chemistry, biology, and so on, the mathematical models that drive these 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  $i = 1:i_{max}$

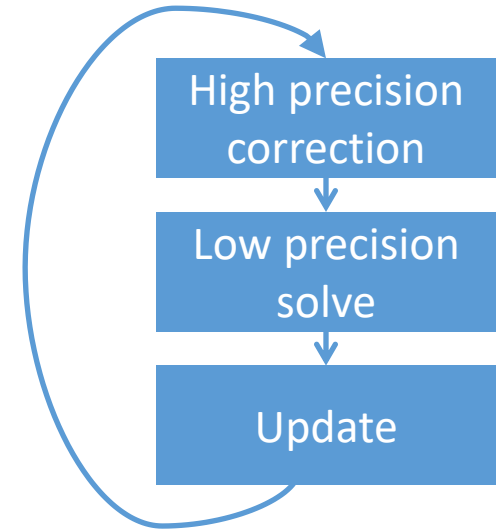
Compute  $f_i = F(x_i)$  in precision  $u_r$

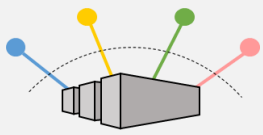
Solve  $J(x_i)d_i = -f_i$  in precision  $u_\ell$

$x_{i+1} = x_i + d_i$  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  $i = 1:i_{max}$

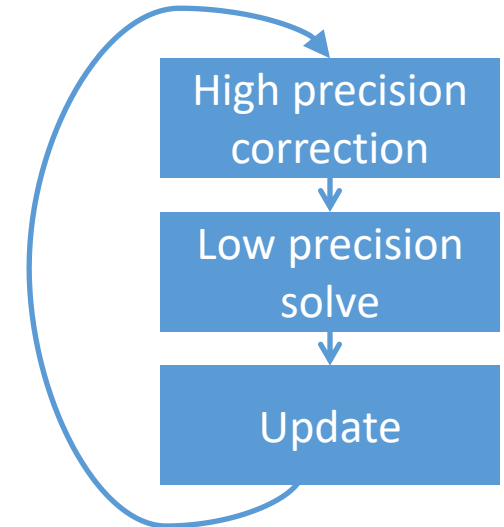
Compute  $f_i = F(x_i)$  in precision  $u_r$

Solve  $J(x_i)d_i = -f_i$  in precision  $u_\ell$

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

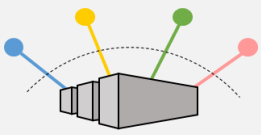
end

[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

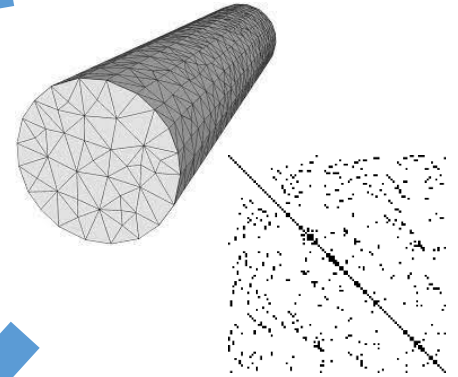


modeling error

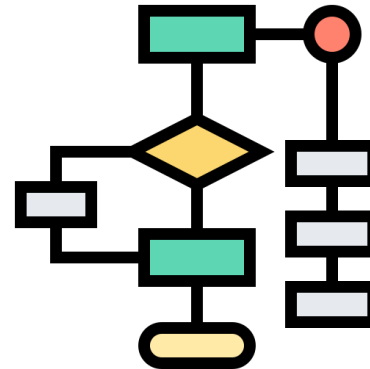


$$\begin{aligned} \frac{\partial u}{\partial t} + (u \cdot \nabla)u \\ = \nu \nabla^2 u - \frac{1}{\rho} \nabla p + f \\ \nabla \cdot u = 0 \end{aligned}$$

discretization/ linearization error,  
measurement error

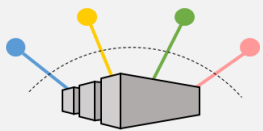


algorithmic approximation



rounding error/  
truncation





# Randomized Nyström Approximation

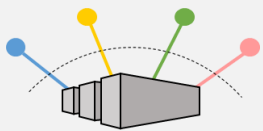
Want to compute a rank- $k$  approximation of sym. PSD  $A \approx U\Theta U^T$  via the randomized Nyström method.

Nyström approximation:

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

where  $\Omega$  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:

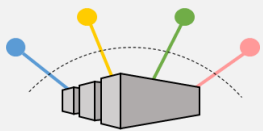
$$A_N = (A\Omega)(\Omega^T A\Omega)^\dagger (A\Omega)^T$$

where  $\Omega$  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].



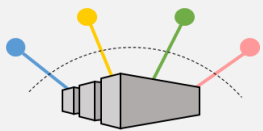
# Single-Pass Nyström Approximation

Given sym. PSD matrix  $A$ , target rank  $k$

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

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





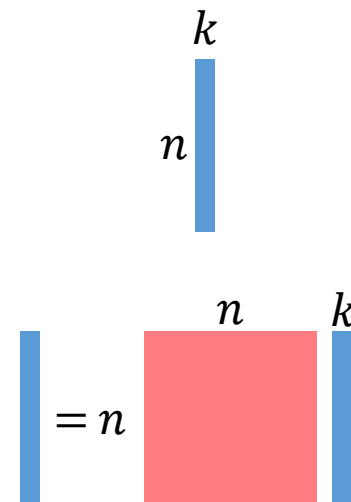
# Single-Pass Nyström Approximation

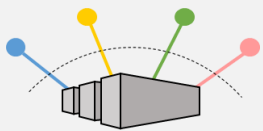
Given sym. PSD matrix  $A$ , target rank  $k$

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

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

$$Y = A\Omega$$





# Single-Pass Nyström Approximation

Given sym. PSD matrix  $A$ , target rank  $k$

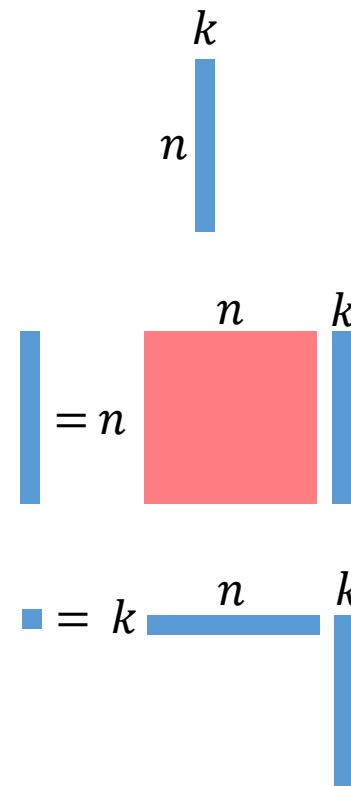
$$G = \text{randn}(n, k)$$

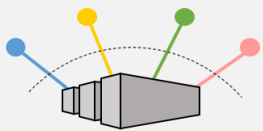
$$[\Omega, \sim] = \text{qr}(G, 0)$$

$$Y = A\Omega$$

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

$$B = \Omega^T Y_\nu$$





# Single-Pass Nyström Approximation

Given sym. PSD matrix  $A$ , target rank  $k$

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

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

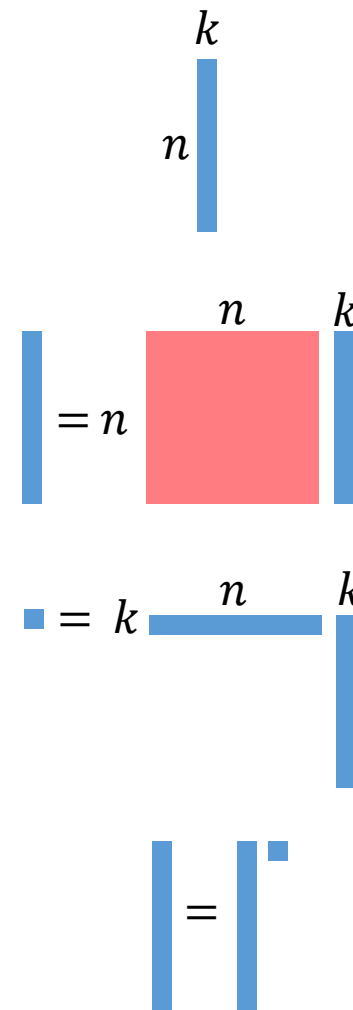
$$Y = A\Omega$$

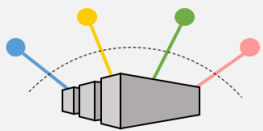
Compute shift  $\nu$ ;  $Y_\nu = Y + \nu\Omega$

$$B = \Omega^T Y_\nu$$

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

$$\text{Solve } F = Y_\nu / C$$





# Single-Pass Nyström Approximation

Given sym. PSD matrix  $A$ , target rank  $k$

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

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

$$Y = A\Omega$$

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

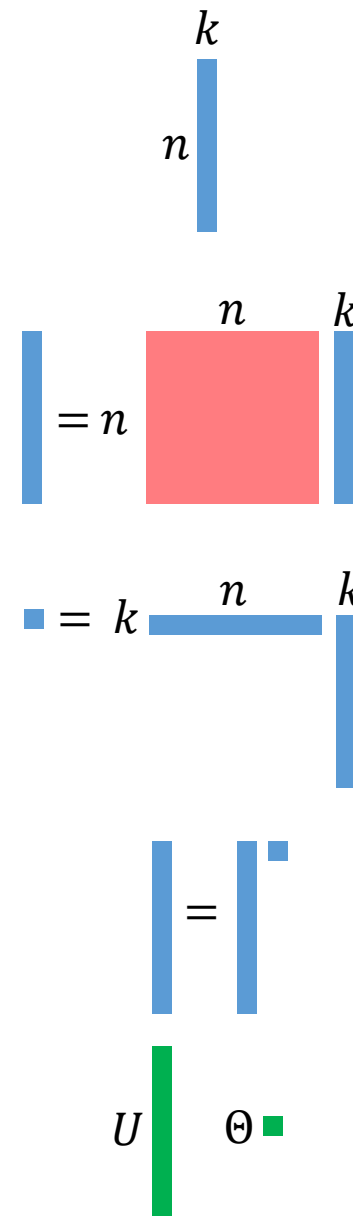
$$B = \Omega^T Y_\nu$$

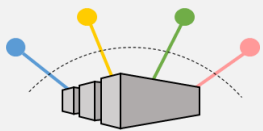
$$C = \text{chol}((B + B^T)/2)$$

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  $k$

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

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

$$Y = A\Omega$$

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

$$B = \Omega^T Y_\nu$$

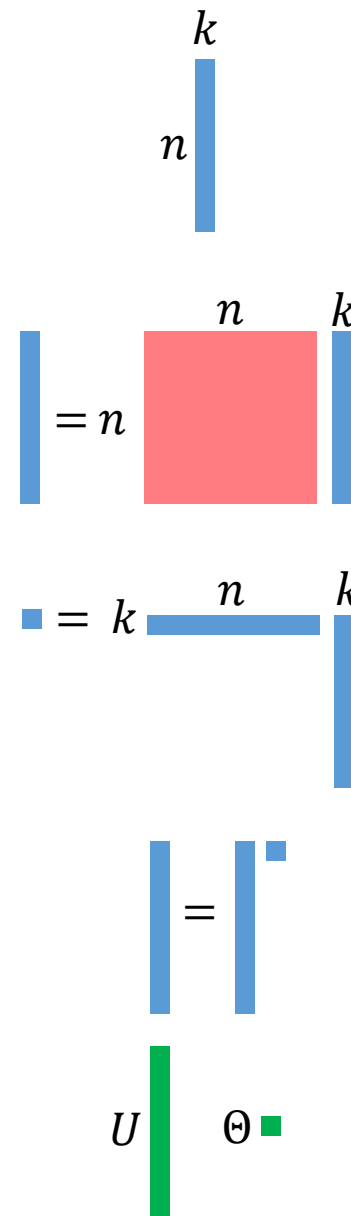
$$C = \text{chol}((B + B^T)/2)$$

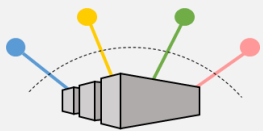
$$\text{Solve } F = Y_\nu / C$$

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

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

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  $k$

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

$$u \leq u_p$$

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

(precision  $u$ )

$$Y = A\Omega$$

(precision  $u_p$ )

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

(precision  $u$ )

$$B = \Omega^T Y_\nu$$

(precision  $u$ )

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

(precision  $u$ )

Solve  $F = Y_\nu/C$

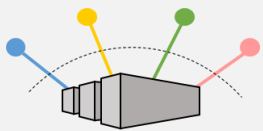
(precision  $u$ )

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

(precision  $u$ )

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

(precision  $u$ )

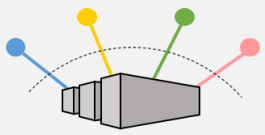


# Error Bounds

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

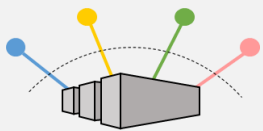
exact Nyström  
approximation

Nyström approximation  
computed in  
finite precision



# Error Bounds

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



# Error Bounds

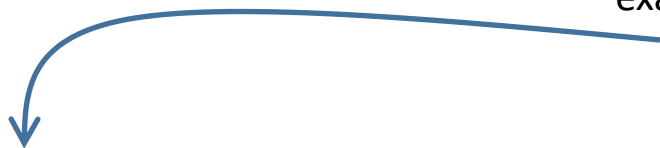
$$\|A - \hat{A}_N\|_2 = \|A - A_N + A_N - \hat{A}_N\|_2 \leq \|A - A_N\|_2 + \|A_N - \hat{A}_N\|_2$$



exact approximation  
error

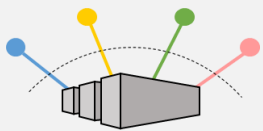


finite precision  
error



Deterministic bound [Gittens, Mahoney, 2016]

Expected value bound [Frangella, Tropp, Udell, 2021]



# Error Bounds

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

Theorem: If we choose precisions  $u_p$  and  $u$  such that

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

then

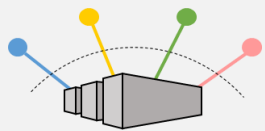
$$\|A_N - \hat{A}_N\|_F \lesssim \|A - A_N\|_F + k^{1/2} n u_p \kappa_2(A_k) \tilde{\kappa}(\Omega)^2 \|A\|_F$$

where

$A_k$  is the best rank- $k$  approximation of  $A$

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

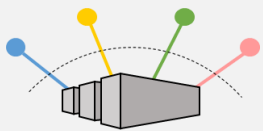
[C., Daužickaitė, 2024]



# A Practical Heuristic

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

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



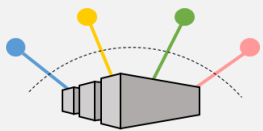
# A Practical Heuristic

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

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



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



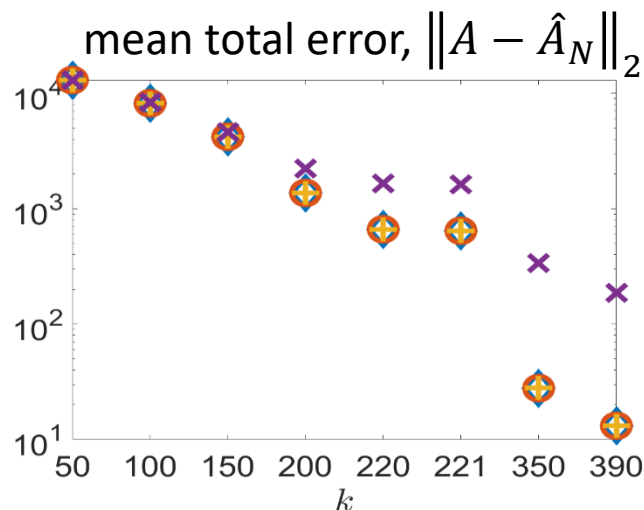
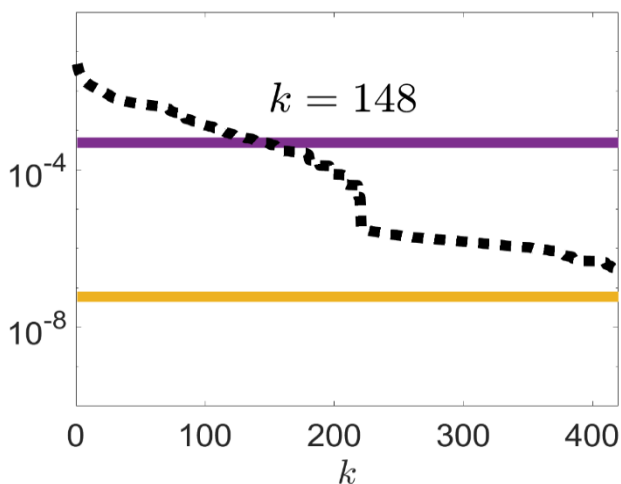
# A Practical Heuristic

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

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



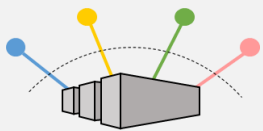
**Interpretation: The greater the low-rank approximation error, the lower the precision we can use!**



$\lambda_k/\lambda_1$ 
  $\sqrt{n}u_p, u_p = \text{half}$   
  $\sqrt{n}u_p, u_p = \text{single}$

exact
   $u_p = \text{half}, u = \text{double}$   
  $u_p = \text{single}, u = \text{double}$ 
  $u_p, u = \text{double}$

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

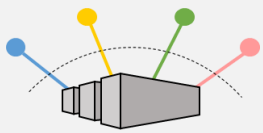
$$fl(a + b) = (a + b)(1 + \delta), \quad |\delta| \leq u.$$

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

Then

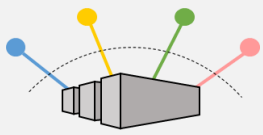
$$\begin{aligned} fl(a + \tilde{b}) &= (a + \tilde{b})(1 + \delta) \\ &= (a + b(1 + \delta_{low}))(1 + \delta) \\ &= (a + b)(1 + \delta) \underbrace{\left(1 + \frac{b}{a+b} \delta_{low}\right)} \end{aligned}$$

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

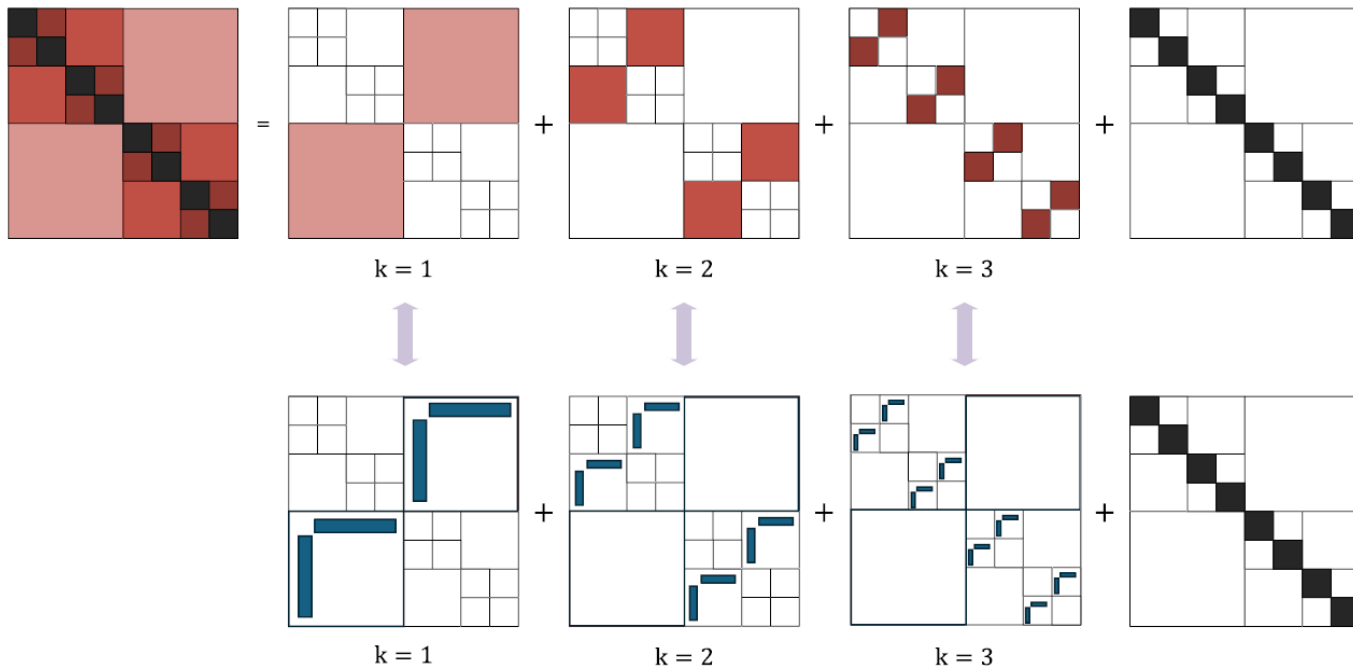


# Sensitivity of subparts and adaptive algorithms

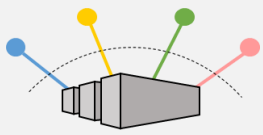
- Takeaway: computations performed on data of small magnitude need not use very high precision.
- Adaptive approaches: adapt precision to be inversely proportional to the weight of the data
  - Weight can be magnitude, norm, condition number, etc.



# Example: HODLR Matrices



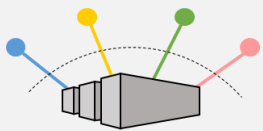
- Hierarchical Off-Diagonal Low-Rank (HODLR) matrices
- Fixed hierarchical block structure, low-rank ( $p$ ) approximation of all off-diagonal blocks
- Enables efficient computation, e.g.,  $O(n^2)$  computations become  $O(pn \log n)$



# Mixed Precision HODLR Matrices

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

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



# Mixed Precision HODLR Matrices

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

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

Theorem:

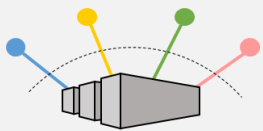
$$\frac{\|H - \hat{H}\|_F}{\|H\|_F} \lesssim \left( 2\sqrt{2} \left( \sum_{k=1}^{\ell} 2^k \xi_k^2 u_k^2 \right)^{1/2} + \varepsilon \right)$$

where

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

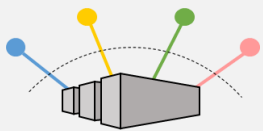
**Relative (norm-based) weight** of off-diagonal blocks in level  $k$

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



# Mixed Precision HODLR Matrices

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



# Mixed Precision HODLR Matrices

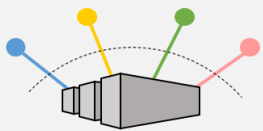
$$\frac{\|H - \hat{H}\|_F}{\|H\|_F} \lesssim \left( 2\sqrt{2} \left( \sum_{k=1}^{\ell} 2^k \xi_k^2 \mathbf{u}_k^2 \right)^{1/2} + \varepsilon \right)$$

If we choose

$$\mathbf{u}_k \leq \frac{\varepsilon}{2^{k/2} \xi_k}$$

then the bound becomes

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



# Mixed Precision HODLR Matrices

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

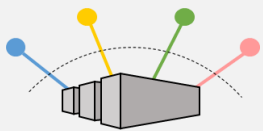
If we choose

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

Less important blocks  
(smaller  $\xi_k$ ) can be stored in  
lower precision (larger  $u_k$ )

then the bound becomes

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



# Mixed Precision HODLR Matrices

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

If we choose

$$\mathbf{u}_k \leq \frac{\varepsilon}{2^{k/2} \xi_k}$$

**Adaptive precision approach:**

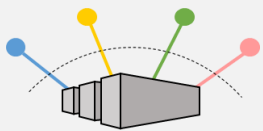
Store off-diagonal blocks in different precisions based on their relative norm

then the bound becomes

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

4. When low precision hardware can efficiently emulate high precision

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

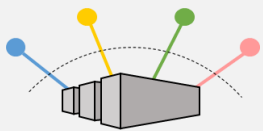


### TFLOPs/TOPs for recent GPUs

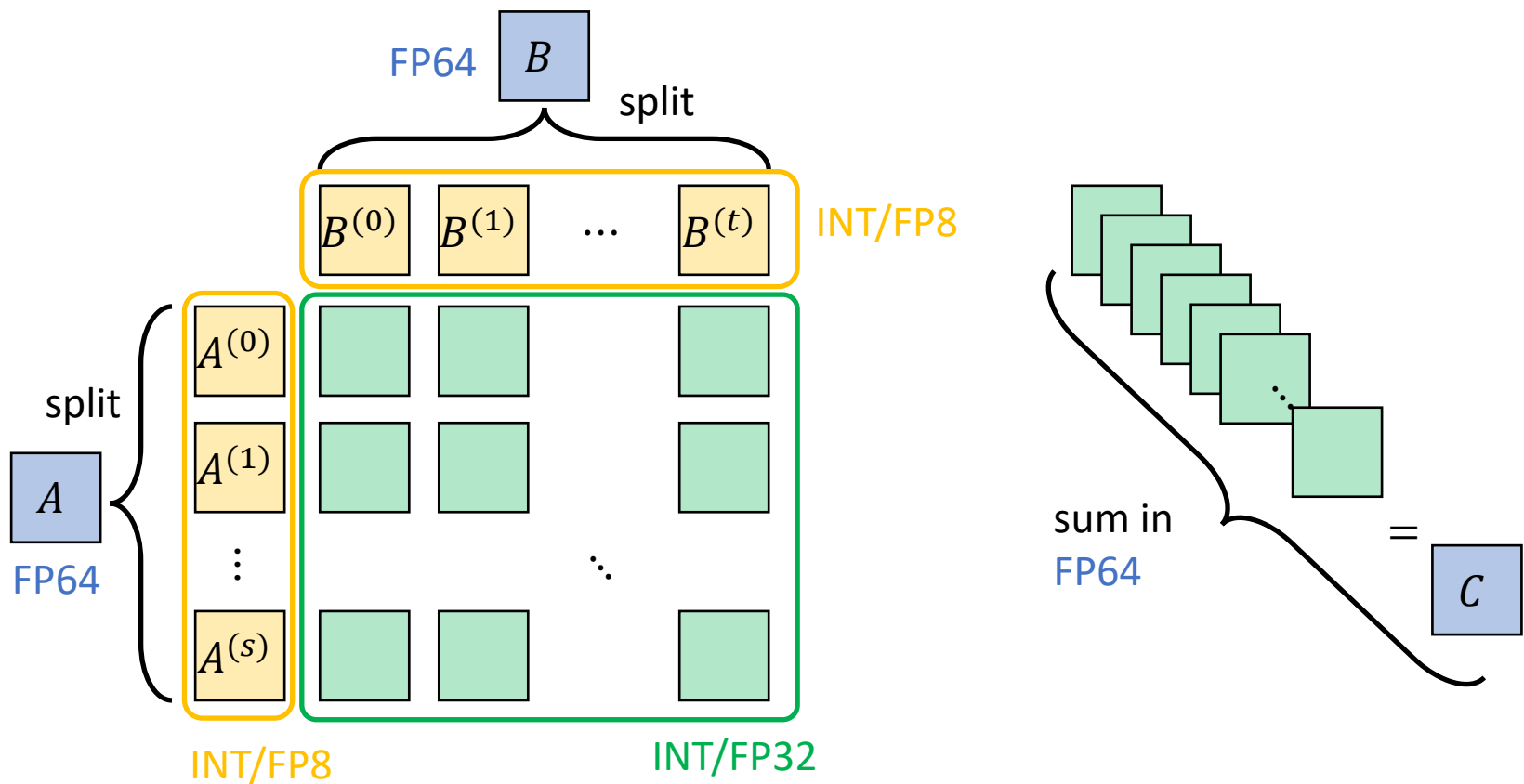
	RTX 4090	H200	B200
FP64	1.29	34	40
FP64 TC	-	67	40
FP32	82.6	67	80
TF32 TC	82.6	494	1100
BF16 TC	165.2	989	2250
FP16 TC	165.2	989	2250
INT8 TC	660.6	1979	4500
FP8 TC	660.6	1979	4500
FP6 TC	-	-	4500
FP4 TC	-	-	9000

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

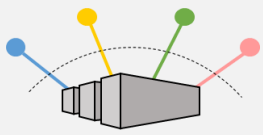
⇒ **Emulate** higher precision with lower precision!



# 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

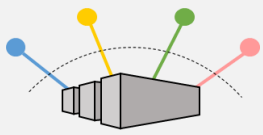
$$\begin{aligned} A &\approx D_1^{-1}D_1A_1 + D_2^{-1}D_2A_2 + \cdots + D_k^{-1}D_kA_k, \\ B &\approx B_1E_1E_1^{-1} + B_2E_2E_2^{-1} + \cdots + B_kE_kE_k^{-1}, \end{aligned}$$

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  $1 \leq i \leq k$  can be represented in INT8.

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

Approximation:

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



# The Ozaki Scheme

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

Write the splitting

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

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  $1 \leq i \leq k$  can be represented in INT8.

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

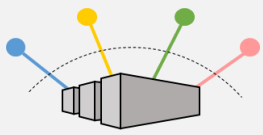
Approximation:

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

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

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

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  $1 \leq i \leq k$  can be represented in INT8.

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

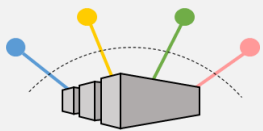
Approximation:

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

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

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

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  $1 \leq i \leq k$  can be represented in INT8.

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

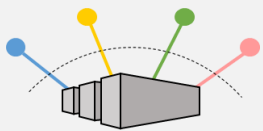
Approximation:

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

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

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

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  $1 \leq i \leq k$  can be represented in INT8.

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

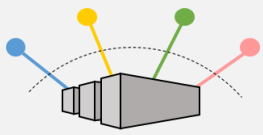
Approximation:

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

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

**HPC**WIRE

SECTORS ▾

TOPICS ▾

RESOURCES ▾

SPECIALS ▾

AI

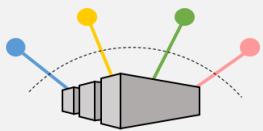
Have You Heard About the Ozaki Scheme? You Will

ACADEMIA & RESEARCH

by Doug Eadline | April 17, 2025



- 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<sup>1</sup>

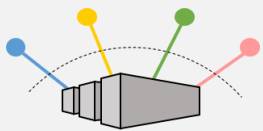
<b>NVFP4 Inference</b>	50 PFLOPS
<b>NVFP4 Training<sup>2</sup></b>	35 PFLOPS
<b>FP8/FP6 Training<sup>2</sup></b>	17.5 PFLOPS
<b>INT8<sup>2</sup></b>	0.25 POPS
<b>FP16/BF16<sup>2</sup></b>	4 PFLOPS
<b>TF32<sup>2</sup></b>	2 PFLOPS
<b>FP32</b>	130 TFLOPS
<b>FP64</b>	33 TFLOPS
<b>FP32 SGEMM<sup>3</sup>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

1. Preliminary information. All values are up to and subject to change.

2. Dense specification.

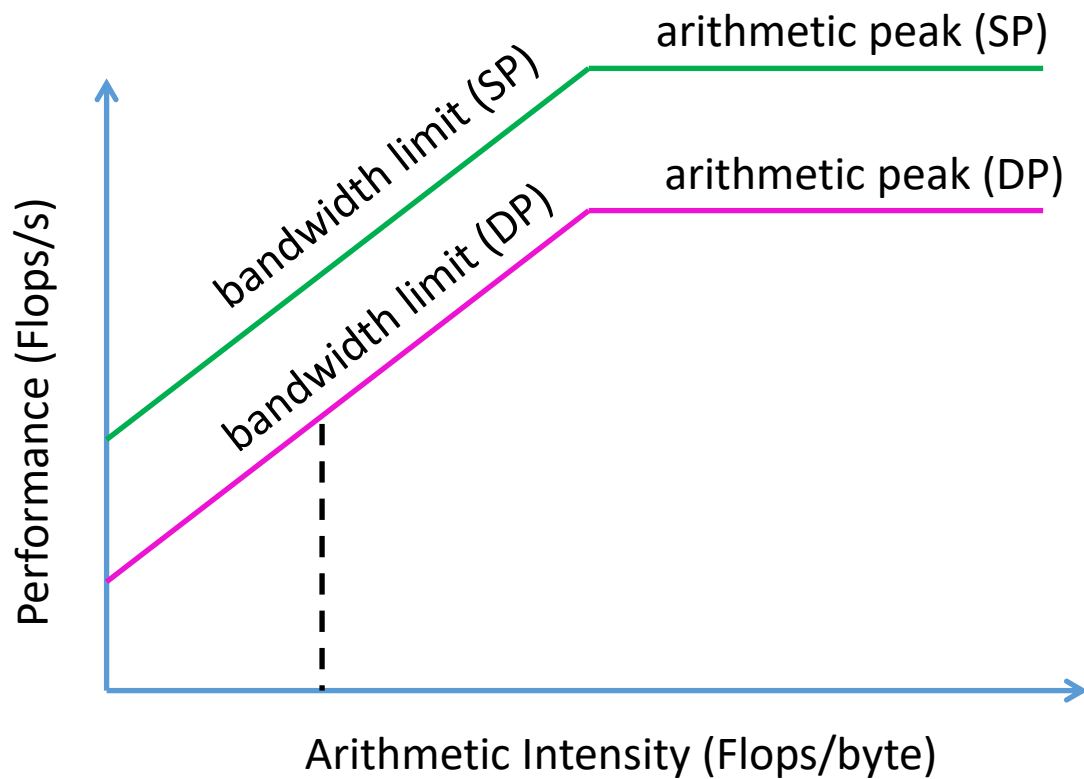
3. Peak performance using Tensor Core-based emulation algorithms.

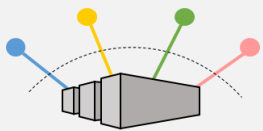
## 5. When data transfers are the bottleneck



# The Roofline Model

[Williams, Waterman, Patterson, 2009]

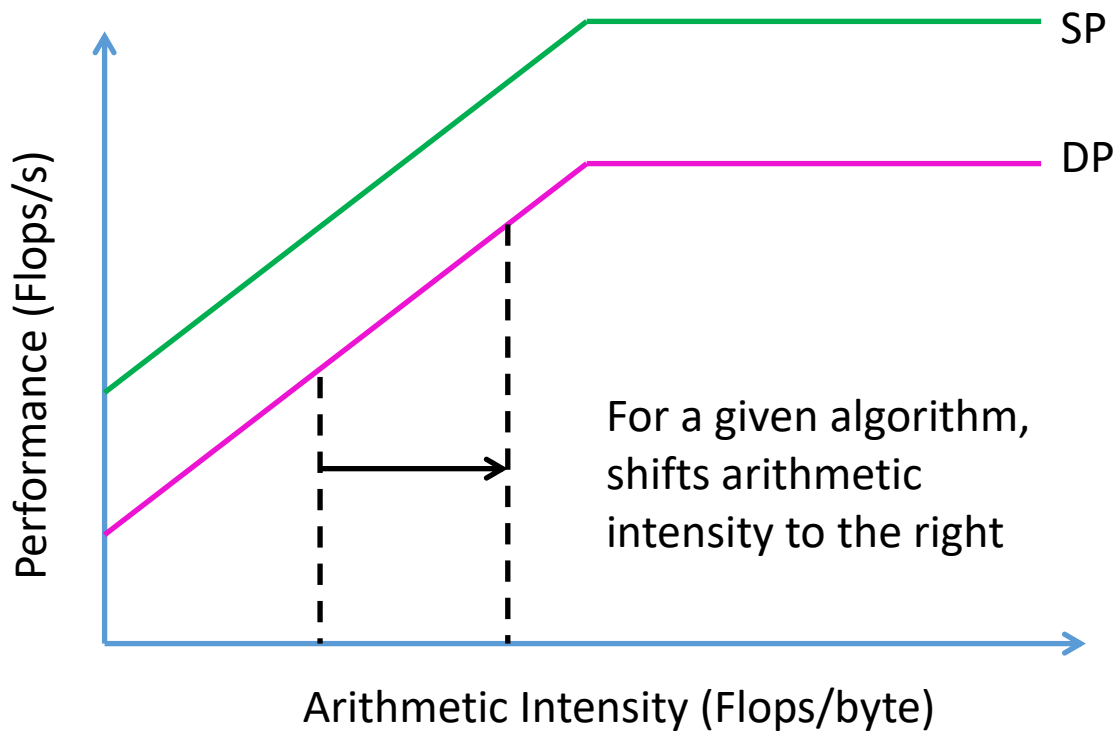


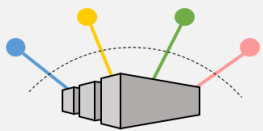


# Idea: Decouple Storage and Compute Precision

Compute in higher precision, store in lower precision

⇒ Same flops; move fewer bytes

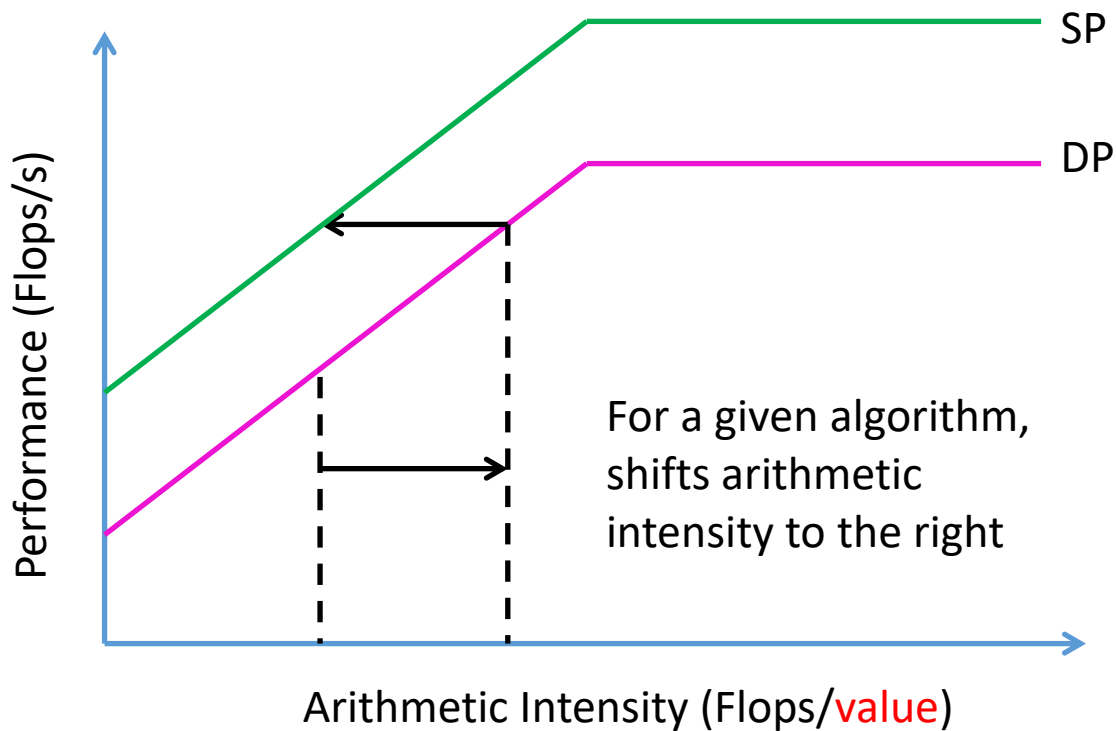


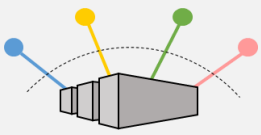


# Idea: Decouple Storage and Compute Precision

Compute in higher precision, store in lower precision

⇒ Same flops; move fewer bytes

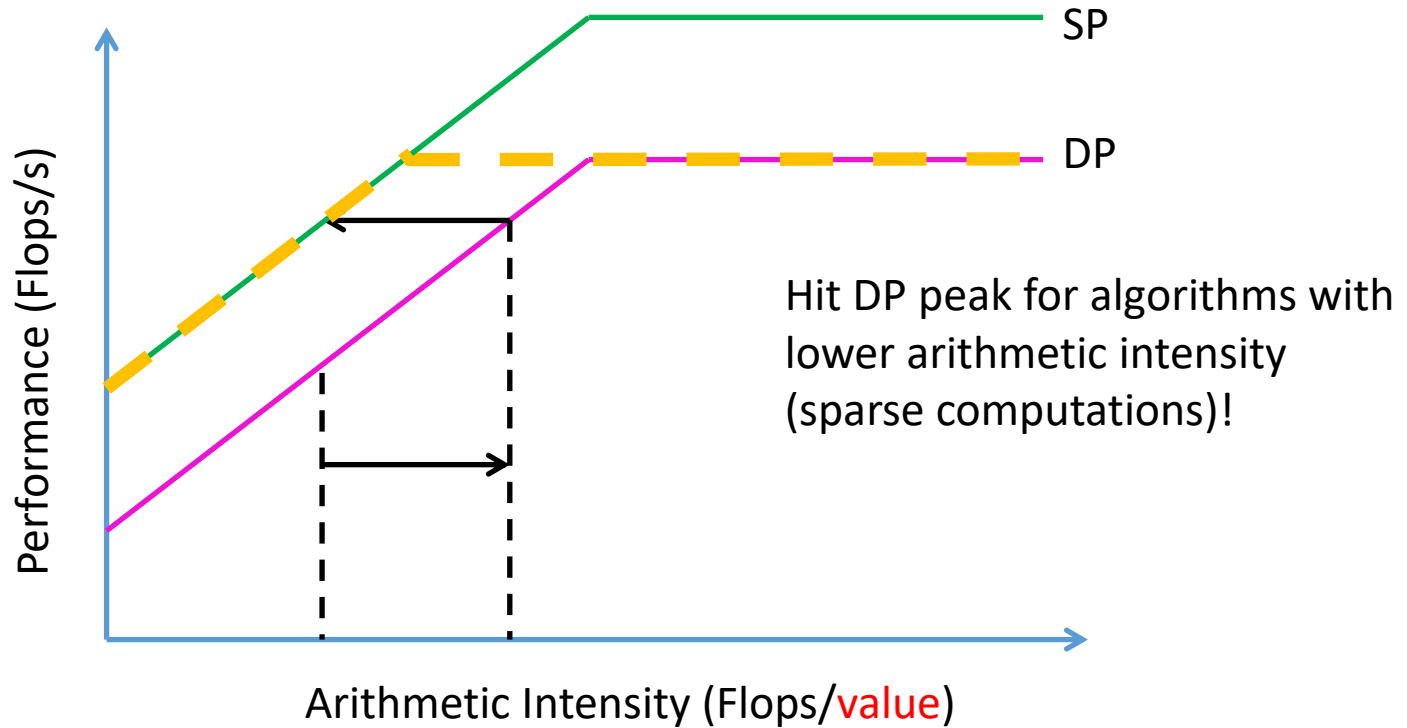


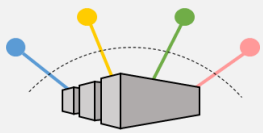


# Idea: Decouple Storage and Compute Precision

Compute in higher precision, store in lower precision

⇒ Same flops; move fewer bytes

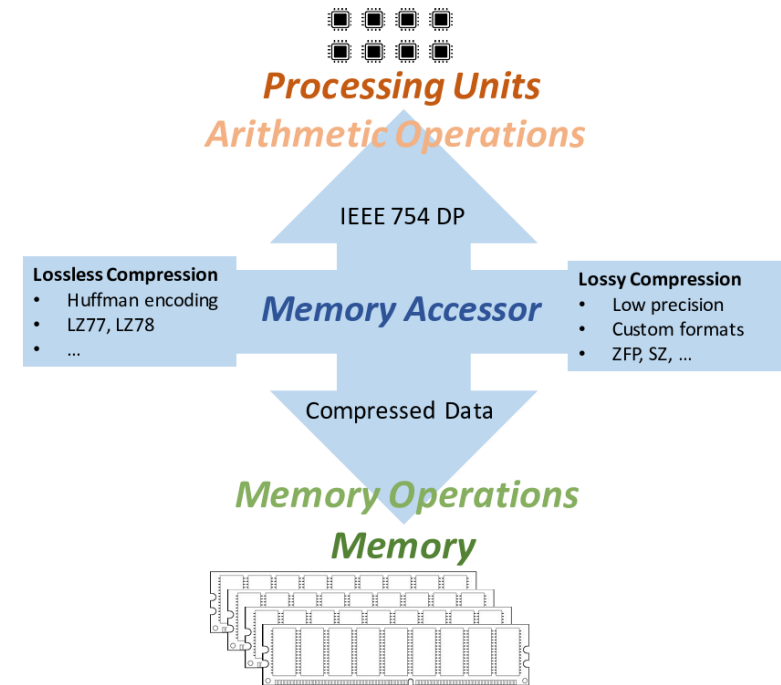


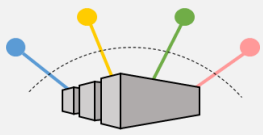


# Memory Accessor

[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]



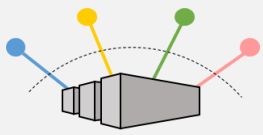


# Other Recent and Ongoing Work

- 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, Rūde, 2025]
- Mixed precision multilevel sampling methods [Martínek, C., Scheichl, 2025]

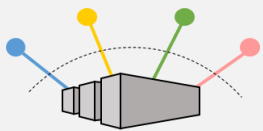
Surveys on mixed precision NLA: [Abdelfattah et al., IJHPC, 2021], [Higham and Mary, 2022]

Topic of today's lecture: [C., Mary, 2025, SIAM News]



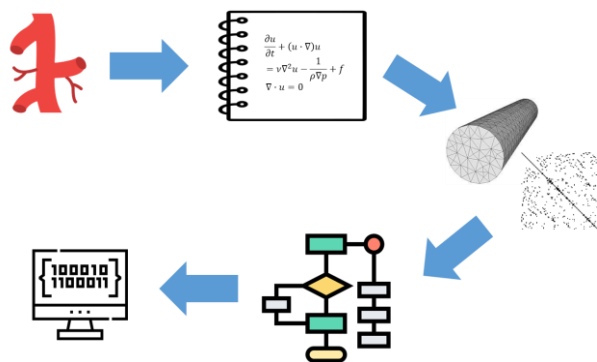
# Looking Forward

- 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 error analysis?
- Emerging hardware – approximate hardware, quantum, neuromorphic, etc.



# 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 error analysis?
- Emerging hardware – approximate hardware, quantum, neuromorphic, etc.

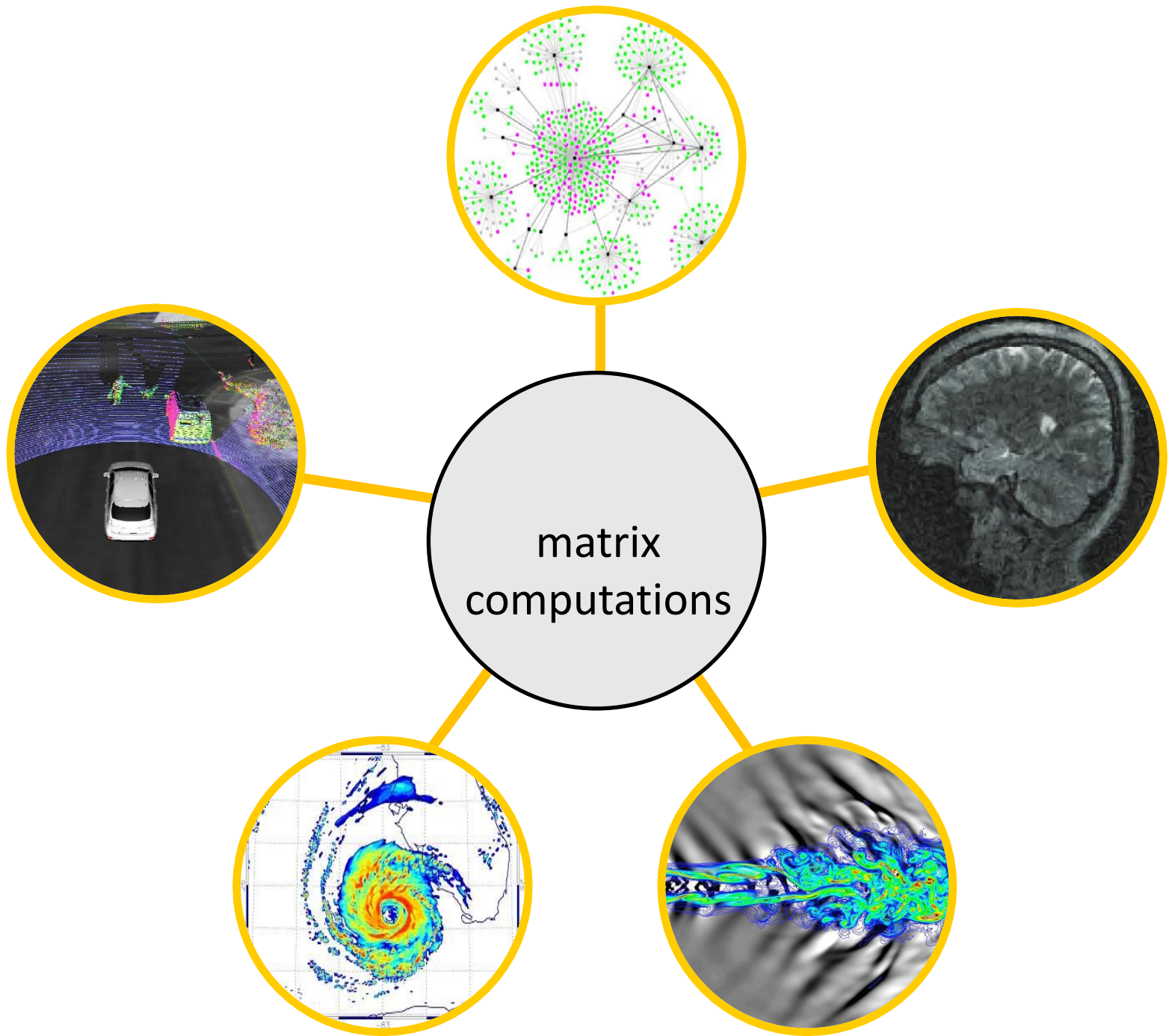


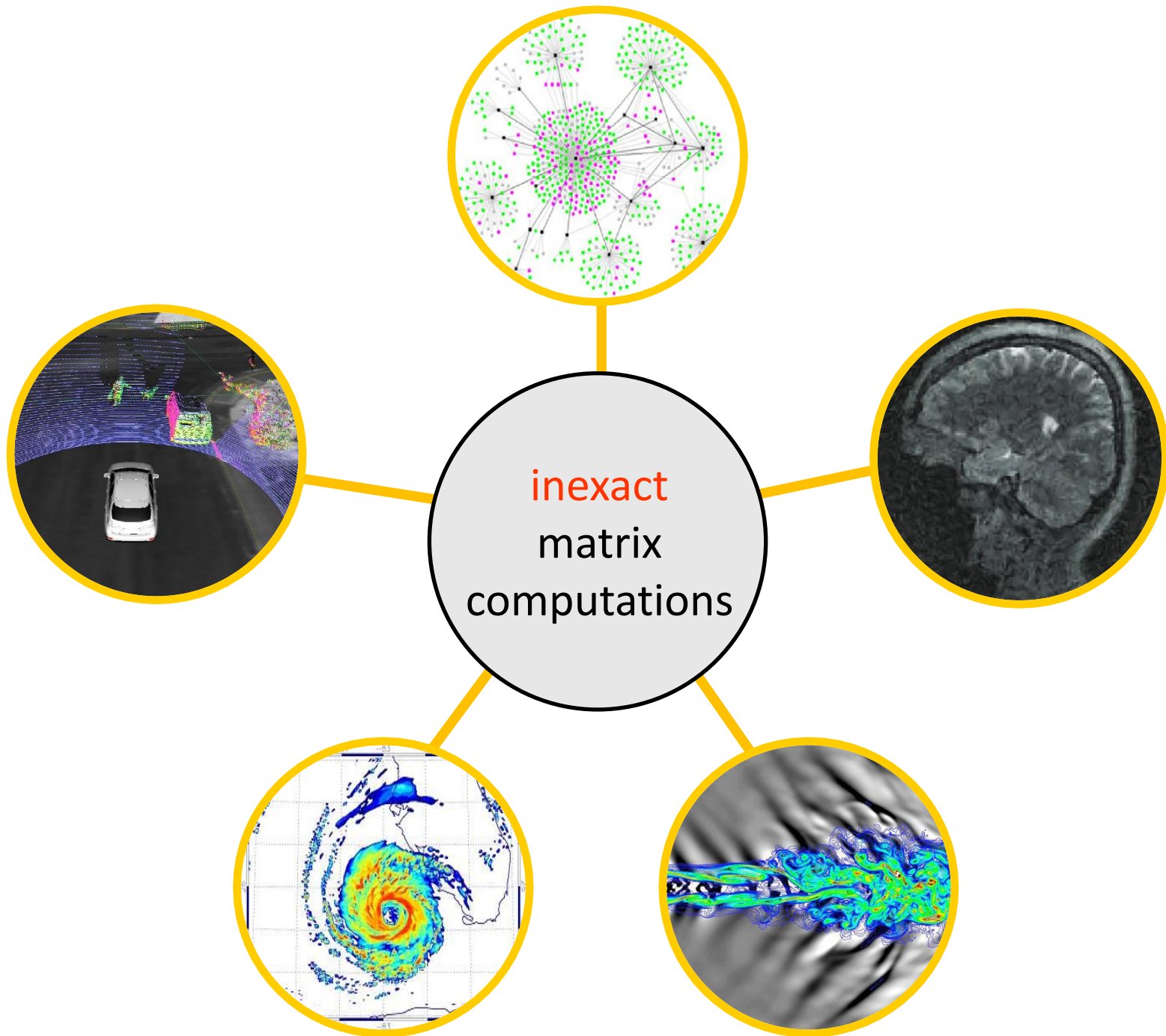
**Challenge:** where can *you* use low precision in your computations?

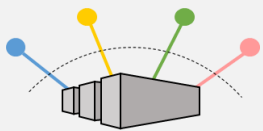
# Thank You!

carson@karlin.mff.cuni.cz  
[www.karlin.mff.cuni.cz/~carson/](http://www.karlin.mff.cuni.cz/~carson/)

# Extra Slides

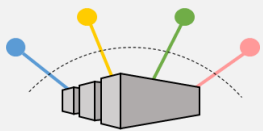






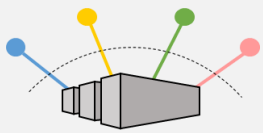
# Challenges of low precision

- Smaller range of representable numbers
  - Limited range of lower precision might cause overflow when rounding
  - Quantities rounded to lower precision may lose important numerical properties (e.g., positive definiteness)
  - One solution: scaling and shifting approach [Higham, Pranesh, 2019]



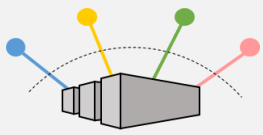
# Challenges of low precision

- Smaller range of representable numbers
  - Limited range of lower precision might cause overflow when rounding
  - Quantities rounded to lower precision may lose important numerical properties (e.g., positive definiteness)
  - One solution: scaling and shifting approach [Higham, Pranesh, 2019]
- Larger unit roundoff
  - Lose something small when storing:  $fl(x) = x(1 + \delta)$ ,  $|\delta| \leq u$
  - Lose something small when computing:  $fl(x \text{ op } y) = (x \text{ op } y)(1 + \delta)$ ,  $|\delta| \leq u$



# Challenges of low precision

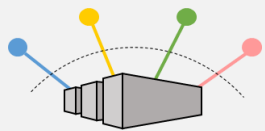
- Smaller range of representable numbers
  - Limited range of lower precision might cause overflow when rounding
  - Quantities rounded to lower precision may lose important numerical properties (e.g., positive definiteness)
  - One solution: scaling and shifting approach [Higham, Pranesh, 2019]
- Larger unit roundoff
  - Lose something small when storing:  $fl(x) = x(1 + \delta)$ ,  $|\delta| \leq u$
  - Lose something small when computing:  $fl(x \text{ op } y) = (x \text{ op } y)(1 + \delta)$ ,  $|\delta| \leq u$
- Do error bounds still apply?
  - Error bound with constant  $nu$  provides no information if  $nu > 1$
  - One solution: probabilistic approach [Higham, Mary, 2019], [Higham, Mary, 2020]



# Challenges of low precision

- Smaller range of representable numbers
  - Limited range of lower precision might cause overflow when rounding
  - Quantities rounded to lower precision may lose important numerical properties (e.g., positive definiteness)
  - One solution: scaling and shifting approach [Higham, Pranesh, 2019]
- Larger unit roundoff
  - Lose something small when storing:  $fl(x) = x(1 + \delta)$ ,  $|\delta| \leq u$
  - Lose something small when computing:  $fl(x \text{ op } y) = (x \text{ op } y)(1 + \delta)$ ,  $|\delta| \leq u$
- Do error bounds still apply?
  - Error bound with constant  $nu$  provides no information if  $nu > 1$
  - One solution: probabilistic approach [Higham, Mary, 2019], [Higham, Mary, 2020]

Does it matter?

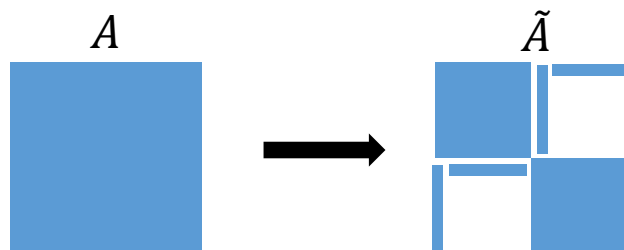


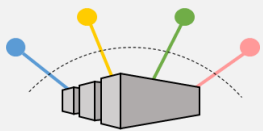
# Example: Block Low-Rank Approximation

Inverse multiquadratic kernel:

$$A(i, j) = \frac{1}{\sqrt{1 + 0.1\|x - y\|^2}}, \quad x, y \in \mathbb{R}^2$$

A is SPD. Low-rank approximation of A should also be SPD!



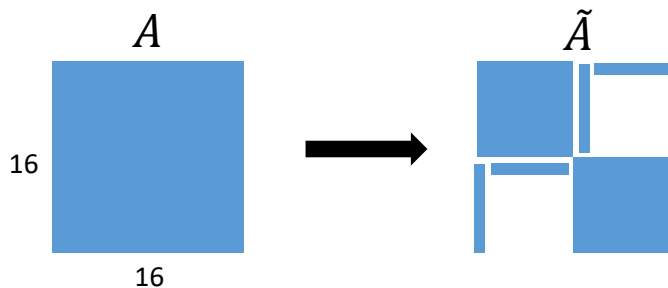


# Example: Block Low-Rank Approximation

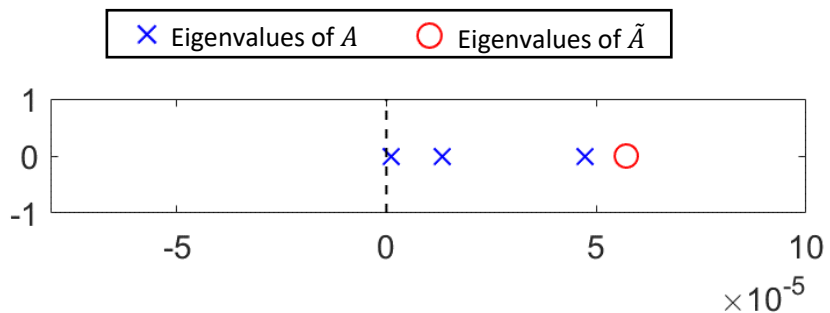
Inverse multiquadratic kernel:

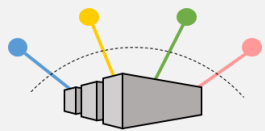
$$A(i, j) = \frac{1}{\sqrt{1 + 0.1\|x - y\|^2}}, \quad x, y \in \mathbb{R}^2$$

A is SPD. Low-rank approximation of A should also be SPD!



Exact arithmetic SVD:



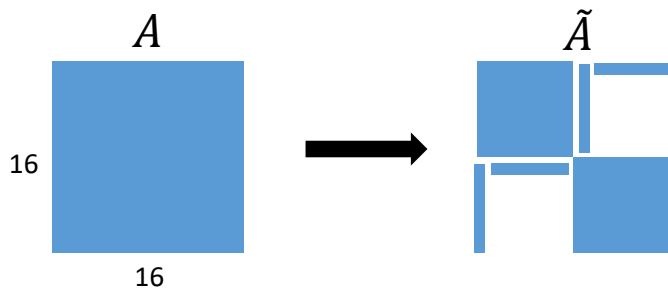


# Example: Block Low-Rank Approximation

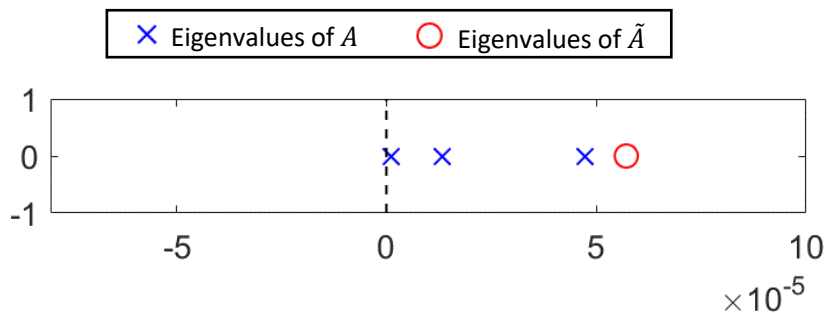
Inverse multiquadratic kernel:

$$A(i, j) = \frac{1}{\sqrt{1 + 0.1\|x - y\|^2}}, \quad x, y \in \mathbb{R}^2$$

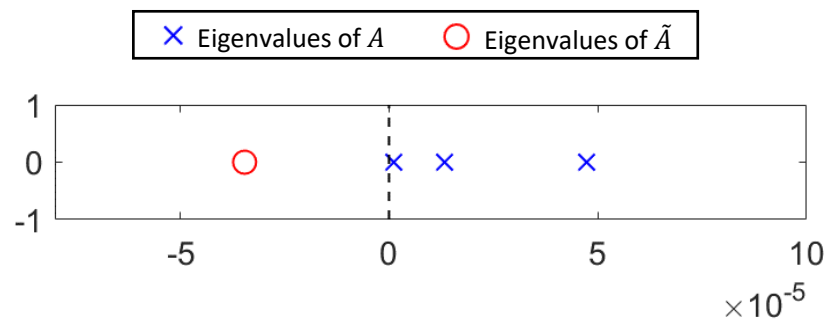
A is SPD. Low-rank approximation of A should also be SPD!

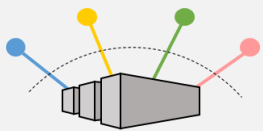


Exact arithmetic SVD:



Half precision SVD:



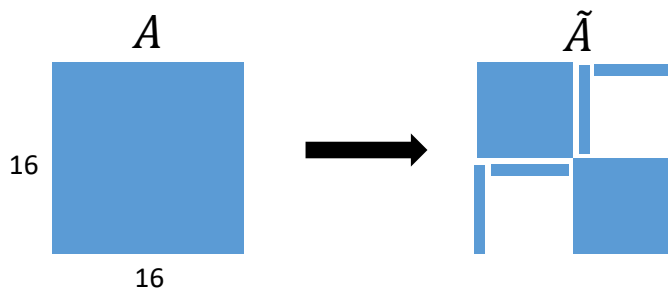


# Example: Block Low-Rank Approximation

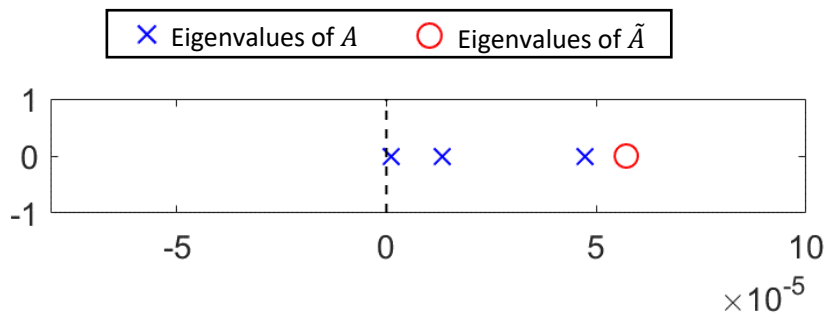
Inverse multiquadratic kernel:

$$A(i, j) = \frac{1}{\sqrt{1 + 0.1\|x - y\|^2}}, \quad x, y \in \mathbb{R}^2$$

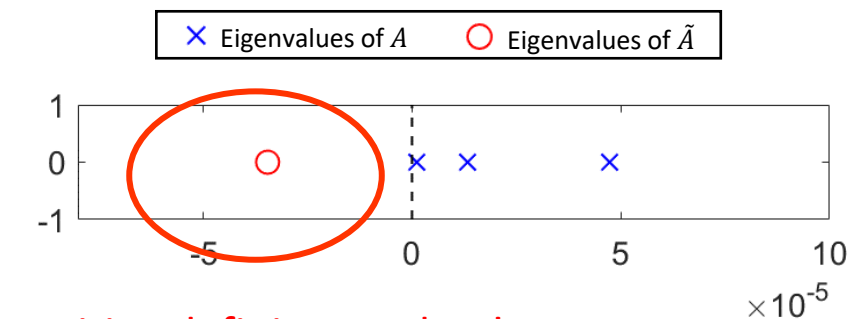
A is SPD. Low-rank approximation of A should also be SPD!



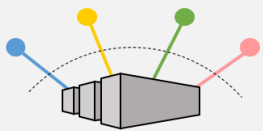
Exact arithmetic SVD:



Half precision SVD:



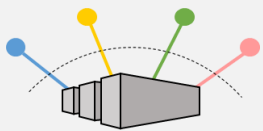
Positive definiteness lost!



# Example: Randomized Algorithms

- Given  $m \times n$   $A$ , want truncated SVD with parameter  $k$

$$A \approx \hat{U} \hat{\Sigma} \hat{V}^T$$



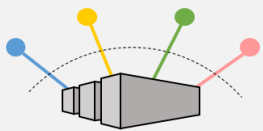
# Example: Randomized Algorithms

- Given  $m \times n$   $A$ , want truncated SVD with parameter  $k$

$$A \approx \hat{U} \hat{\Sigma} \hat{V}^T$$

- Randomized SVD:

$$A \Omega = Y = QR$$



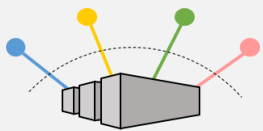
# Example: Randomized Algorithms

- Given  $m \times n$   $A$ , want truncated SVD with parameter  $k$

$$A \approx \hat{U} \hat{\Sigma} \hat{V}^T$$

- Randomized SVD:

$$A \Omega = Y = QR \rightarrow Q^T A = B$$



# Example: Randomized Algorithms

- Given  $m \times n$   $A$ , want truncated SVD with parameter  $k$

$$A \approx \hat{U} \hat{\Sigma} \hat{V}^T$$

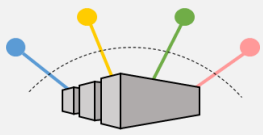
- Randomized SVD:

$$A \Omega = Y = Q R \quad \rightarrow \quad Q^T A = B = \tilde{U} \hat{\Sigma} \hat{V}^T$$

$$\hat{U} = Q \tilde{U}$$

Assuming exact arithmetic:

If  $Q$  satisfies  $\|A - QQ^T A\| \leq \varepsilon$ , then  $\|A - \hat{U} \hat{\Sigma} \hat{V}^T\| \leq \varepsilon$



# What happens in finite precision?

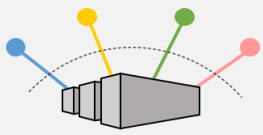
Let's try different types of randsvd matrices from the MATLAB gallery:

```
A = gallery('randsvd', [100, 40], 1e6, mode); k=15;
```

$[U, S, V]$  = rsvd( $A$ ) : exact arithmetic

$[U_d, S_d, V_d]$  = rsvd( $A$ ) : double precision

$[U_h, S_h, V_h, ]$  = rsvd( $A$ ) : half precision



# What happens in finite precision?

Let's try different types of randsvd matrices from the MATLAB gallery:

```
A = gallery('randsvd', [100, 40], 1e6, mode); k=15;
```

$[U, S, V]$  = rsvd(A) : exact arithmetic

$[U_d, S_d, V_d]$  = rsvd(A) : double precision

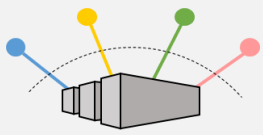
$[U_h, S_h, V_h, ]$  = rsvd(A) : half precision

Mode 3: Geometrically distributed  
singular values

$$\|A - USV^T\|_2 = 4.92e-03$$

$$\|A - U_d S_d V_d^T\|_2 = 4.92e-03$$

$$\|A - U_h S_h V_h^T\|_2 = 4.92e-03$$



# What happens in finite precision?

Let's try different types of randsvd matrices from the MATLAB gallery:

```
A = gallery('randsvd', [100, 40], 1e6, mode); k=15;
```

$[U, S, V]$  = rsvd(A) : exact arithmetic

$[U_d, S_d, V_d]$  = rsvd(A) : double precision

$[U_h, S_h, V_h, ]$  = rsvd(A) : half precision

Mode 3: Geometrically distributed  
singular values

$$\|A - USV^T\|_2 = 4.92e-03$$

$$\|A - U_d S_d V_d^T\|_2 = 4.92e-03$$

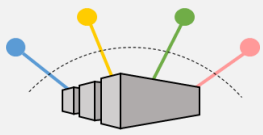
$$\|A - U_h S_h V_h^T\|_2 = 4.92e-03$$

Mode 1: one large singular value

$$\|A - USV^T\|_2 = 1.17e-06$$

$$\|A - U_d S_d V_d^T\|_2 = 1.17e-06$$

$$\|A - U_h S_h V_h^T\|_2 = 1.11e-05$$



# What happens in finite precision?

Let's try different types of randsvd matrices from the MATLAB gallery:

```
A = gallery('randsvd', [100, 40], 1e6, mode); k=15;
```

$[U, S, V]$  = rsvd(A) : exact arithmetic

$[U_d, S_d, V_d]$  = rsvd(A) : double precision

$[U_h, S_h, V_h, ]$  = rsvd(A) : half precision

Mode 3: Geometrically distributed  
singular values

$$\|A - USV^T\|_2 = 4.92e-03$$

$$\|A - U_d S_d V_d^T\|_2 = 4.92e-03$$

$$\|A - U_h S_h V_h^T\|_2 = 4.92e-03$$

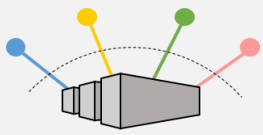
Mode 1: one large singular value

$$\|A - USV^T\|_2 = 1.17e-06$$

$$\|A - U_d S_d V_d^T\|_2 = 1.17e-06$$

$$\|A - U_h S_h V_h^T\|_2 = 1.11e-05$$

Use of low precision leads to an order magnitude  
loss of accuracy!



# What happens in finite precision?

Let's try different types of randsvd matrices from the MATLAB gallery:

```
A = gallery('randsvd', [100, 40], 1e6, mode); k=15;
```

$[U, S, V]$  = rsvd(A) : exact arithmetic

$[U_d, S_d, V_d]$  = rsvd(A) : double precision

$[U_h, S_h, V_h, ]$  = rsvd(A) : half precision

If  $Q$  satisfies  $\|A - QQ^T A\| \leq \varepsilon$ , then  $\|A - \hat{U}\hat{\Sigma}\hat{V}^T\| \leq \varepsilon$

Mode 3: Geometrically distributed  
singular values

$$\|A - USV^T\|_2 = 4.92e-03$$

$$\|A - U_d S_d V_d^T\|_2 = 4.92e-03$$

$$\|A - U_h S_h V_h^T\|_2 = 4.92e-03$$

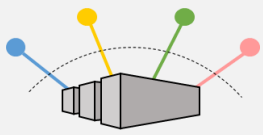
Mode 1: one large singular value

$$\|A - USV^T\|_2 = 1.17e-06$$

$$\|A - U_d S_d V_d^T\|_2 = 1.17e-06$$

$$\|A - U_h S_h V_h^T\|_2 = 1.11e-05$$

$$\|A - Q_h Q_h^T A\|_2 = 3.59e-06$$



# What happens in finite precision?

Let's try different types of randsvd matrices from the MATLAB gallery:

```
A = gallery('randsvd', [100, 40], 1e6, mode); k=15;
```

$[U, S, V]$  = rsvd(A) : exact arithmetic

$[U_d, S_d, V_d]$  = rsvd(A) : double precision

$[U_h, S_h, V_h, ]$  = rsvd(A) : half precision

Error bound no longer holds!

~~If  $Q$  satisfies  $\|A - QQ^T A\| \leq \varepsilon$ , then  $\|A - \hat{U}\hat{\Sigma}\hat{V}^T\| \leq \varepsilon$~~

Mode 3: Geometrically distributed  
singular values

$$\|A - USV^T\|_2 = 4.92e-03$$

$$\|A - U_d S_d V_d^T\|_2 = 4.92e-03$$

$$\|A - U_h S_h V_h^T\|_2 = 4.92e-03$$

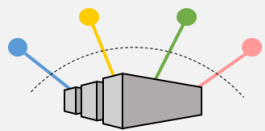
Mode 1: one large singular value

$$\|A - USV^T\|_2 = 1.17e-06$$

$$\|A - U_d S_d V_d^T\|_2 = 1.17e-06$$

$$\|A - U_h S_h V_h^T\|_2 = 1.11e-05$$

$$\|A - Q_h Q_h^T A\|_2 = 3.59e-06$$

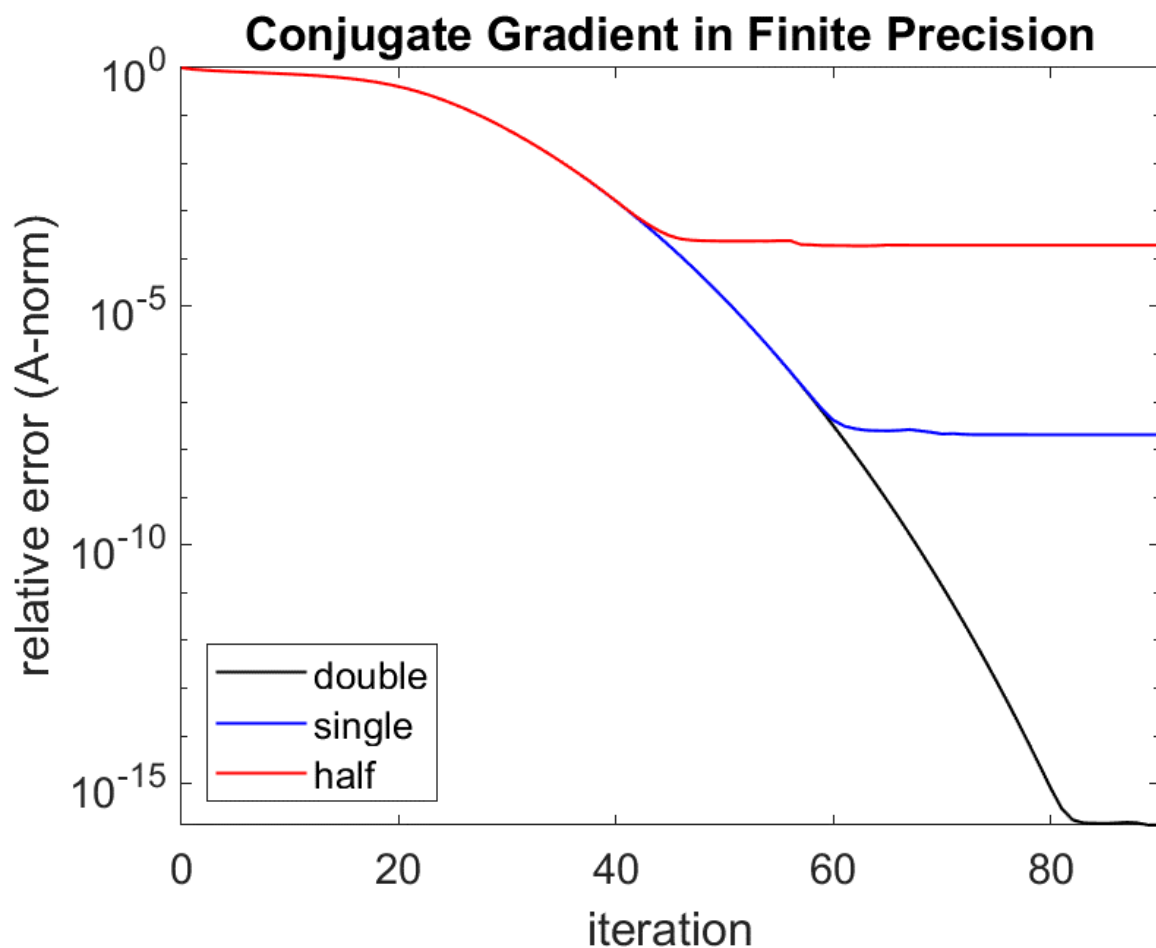


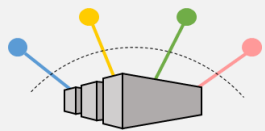
# Example: Iterative Methods

$n = 100, \lambda_1 = 10^{-3}, \lambda_n = 1$

$A = \text{diag}(\text{linspace}(\lambda_1, \lambda_n, n));$

$b = \text{ones}(n, 1);$



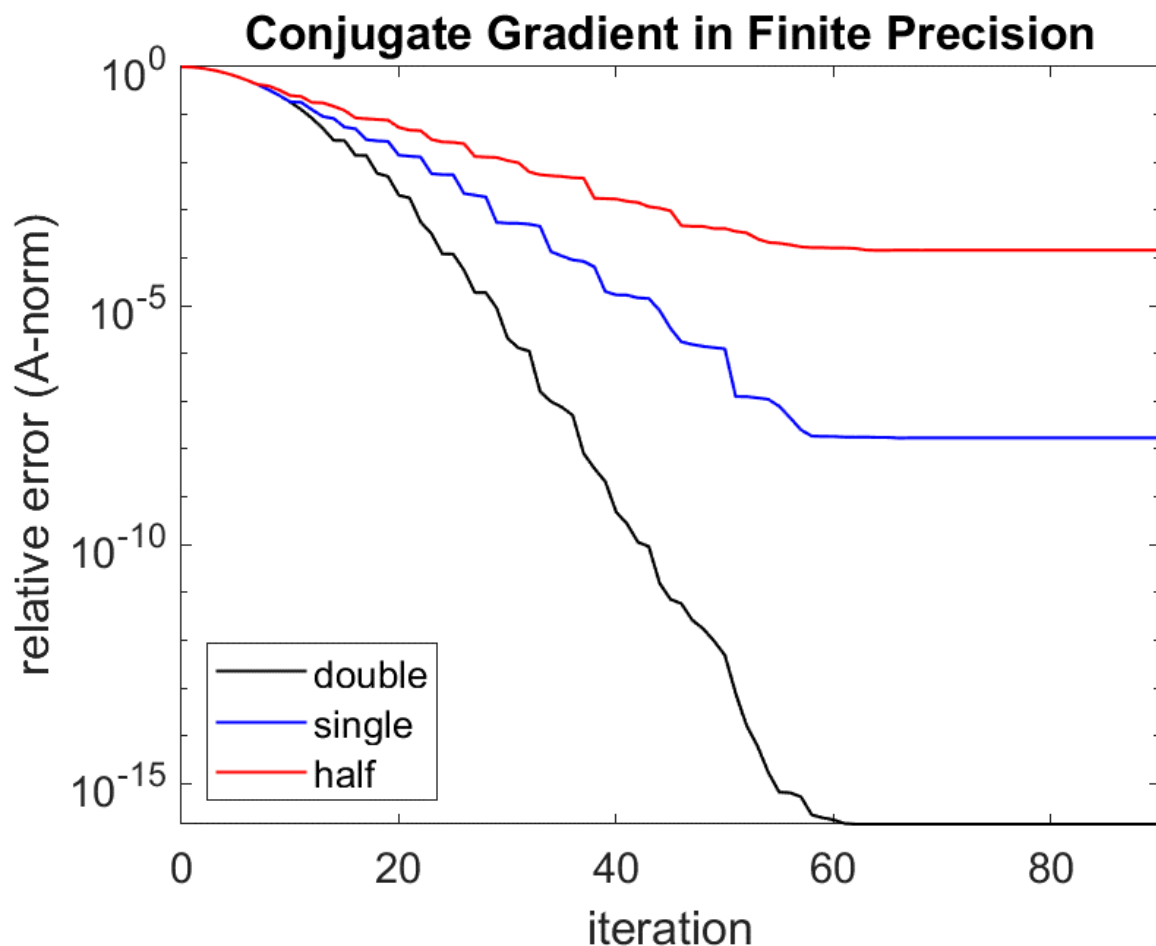


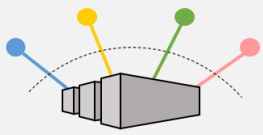
# Example: Iterative Methods

$$n = 100, \lambda_1 = 10^{-3}, \lambda_n = 1$$

$$\lambda_i = \lambda_1 + \left(\frac{i-1}{n-1}\right)(\lambda_n - \lambda_1)(0.65)^{n-i}, \quad i = 2, \dots, n-1$$

$$b = \text{ones}(n, 1);$$



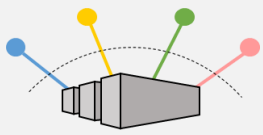


# Mixed precision algorithms

- Goals
  - Improve performance by using potentially low precision in expensive parts
  - While still achieving desired accuracy
- Challenges
  - Need a **rigorous understanding** of how the finite precision computation in each part of an algorithm affects numerical properties (stability, convergence, accuracy)
  - Need adaptive approaches
    - Selection of precisions used **must adapt to the problem input**

# Mixed Precision Sparse Approximate Inverse Preconditioners

C. and Khan. (2023). Mixed precision iterative refinement with sparse approximate inverse preconditioning. *SIAM Journal on Scientific Computing*, 45(3), C131-C153.



# SPAI Preconditioners

Goal: Construct sparse matrix  $M \approx A^{-1}$  (for survey see [Benzi, 2002])

Approach of [Grote, Huckle, 1997]: Construct columns  $m_k$  of  $M$  dynamically

Given matrix  $A$ , initial sparsity structure  $J$ , and tolerance  $\tau$

For each column  $k$ :

Compute QR factorization of submatrix of  $A$  defined by  $J$

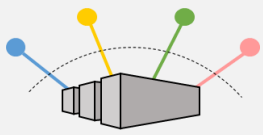
Use QR factorization to solve  $\min_{m_k} \|e_k - Am_k\|_2$

If  $\|r_k\|_2 = \|e_k - Am_k\|_2 \leq \tau$

break;

Else

add select nonzeros to  $J$ , repeat.



# SPAI Preconditioners

Goal: Construct sparse matrix  $M \approx A^{-1}$  (for survey see [Benzi, 2002])

Approach of [Grote, Huckle, 1997]: Construct columns  $m_k$  of  $M$  dynamically

Given matrix  $A$ , initial sparsity structure  $J$ , and tolerance  $\tau$

For each column  $k$ :

Compute QR factorization of submatrix of  $A$  defined by  $J$

Use QR factorization to solve  $\min_{m_k} \|e_k - Am_k\|_2$

If  $\|r_k\|_2 = \|e_k - Am_k\|_2 \leq \tau$

break;

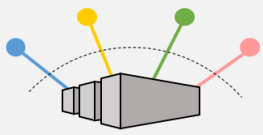
Else

add select nonzeros to  $J$ , repeat.

Benefits: Highly parallelizable

But **construction can still be costly**, esp. for large-scale problems

[Gao, Chen, He, 2021], [Chao, 2001], [Benzi, Tuma, 1999], [He, Yin, Gao, 2020]



# SPAI Preconditioners in Low Precision

Assume we want to compute the sparse approximate inverse in some precision  $\mathbf{u}_s$  and let  $\hat{r}_k = fl_{\mathbf{u}_s}(e_k - A\hat{m}_k)$  for the computed  $\hat{m}_k$ .

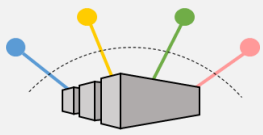
Using standard rounding error analysis and perturbation results for LS problems, we have

$$\|\hat{r}_k\|_2 \leq n^3 \mathbf{u}_s (\|e_k\| + \|A\|\|\hat{m}_k\|)_2.$$

So in order to guarantee we eventually reach a solution with  $\|\hat{r}_k\|_2 \leq \tau$ , we need

$$n^3 \mathbf{u}_s (\|e_k\| + \|A\|\|\hat{m}_k\|)_2 \leq \tau.$$

→ problem must not be so ill-conditioned WRT  $\mathbf{u}_s$  that we incur an error greater than  $\tau$  just computing the residual

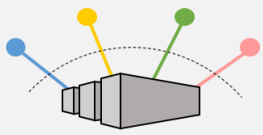


## Mixed Precision SPAI Preconditioners

Theorem: The sparse approximate inverse can be constructed in precision  $u_s$  as long as

$$u_s \text{cond}_2(A) \lesssim \tau,$$

where  $\text{cond}_2(A) = \|A^{-1}\|A\|_2$ .



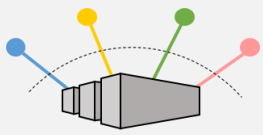
# Mixed Precision SPAI Preconditioners

Theorem: The sparse approximate inverse can be constructed in precision  $u_s$  as long as

$$u_s \text{cond}_2(A) \lesssim \tau,$$

where  $\text{cond}_2(A) = \|A^{-1}\|A\|_2$ .





## Mixed Precision SPAI Preconditioners

Theorem: The sparse approximate inverse can be constructed in precision  $\mathbf{u}_s$  as long as

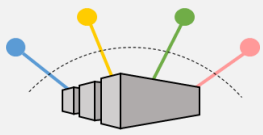
$$\mathbf{u}_s \text{cond}_2(A) \lesssim \boldsymbol{\tau},$$

where  $\text{cond}_2(A) = \|A^{-1}\|A\|_2$ .



Theorem: Convergence of SPAI-GMRES-IR to working precision  $\mathbf{u}$  guaranteed if  $\mathbf{u}_s$ ,  $\mathbf{u}$ , and  $\boldsymbol{\tau}$  are chosen such that

$$n\mathbf{u}_s \text{cond}_2(A^T) \lesssim n\boldsymbol{\tau} \lesssim \mathbf{u}^{-1/2}.$$



# Mixed Precision SPAI Preconditioners

Theorem: The sparse approximate inverse can be constructed in precision  $u_s$  as long as

$$u_s \text{cond}_2(A) \lesssim \tau,$$

where  $\text{cond}_2(A) = \|A^{-1}\|A\|_2$ .

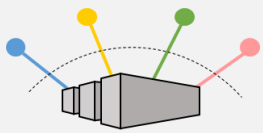


Theorem: Convergence of SPAI-GMRES-IR to working precision  $u$  guaranteed if  $u_s$ ,  $u$ , and  $\tau$  are chosen such that

$$n u_s \text{cond}_2(A^T) \lesssim n \tau \lesssim u^{-1/2}.$$

$\hat{M}$  can be  
constructed

$\hat{M}$  is a good enough  
preconditioner



# Mixed Precision SPAI Preconditioners

Theorem: The sparse approximate inverse can be constructed in precision  $\mathbf{u}_s$  as long as

$$\mathbf{u}_s \text{cond}_2(A) \lesssim \boldsymbol{\tau},$$

where  $\text{cond}_2(A) = \|A^{-1}\|A\|_2$ .

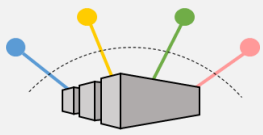


Theorem: Convergence of SPAI-GMRES-IR to working precision  $\mathbf{u}$  guaranteed if  $\mathbf{u}_s$ ,  $\mathbf{u}$ , and  $\boldsymbol{\tau}$  are chosen such that

$$n\mathbf{u}_f \text{cond}_2(A^T) \lesssim n\boldsymbol{\tau} \lesssim \mathbf{u}^{-1/2}.$$

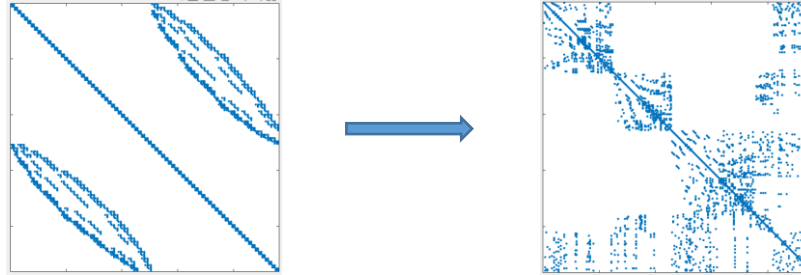
$\hat{M}$  can be  
constructed

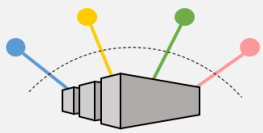
$\hat{M}$  is a good enough  
preconditioner



# Example: Iterative Refinement

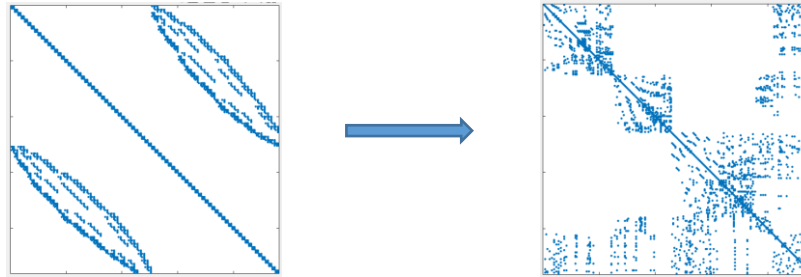
Matrix: steam1,  $n = 240$ ,  $\text{nnz} = 2,248$ ,  $\kappa_{\infty}(A) = 3 \cdot 10^7$ ,  $\text{cond}(A^T) = 3 \cdot 10^3$



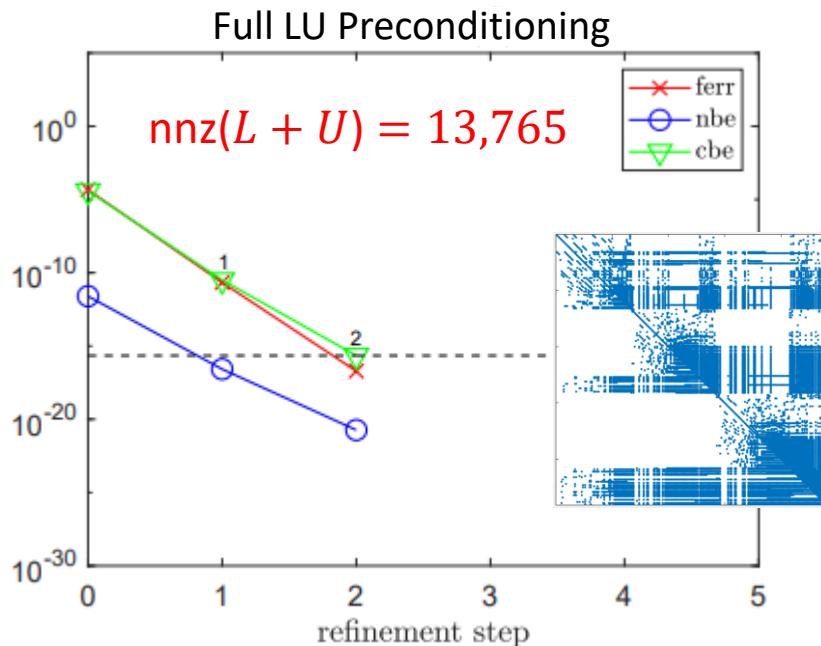


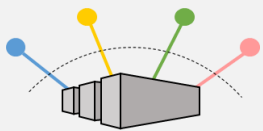
# Example: Iterative Refinement

Matrix: steam1,  $n = 240$ ,  $\text{nnz} = 2,248$ ,  $\kappa_{\infty}(A) = 3 \cdot 10^7$ ,  $\text{cond}(A^T) = 3 \cdot 10^3$



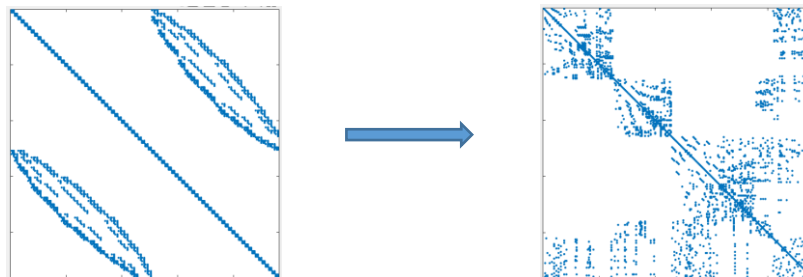
$u_s = \text{single}$ ,  $u = \text{double}$



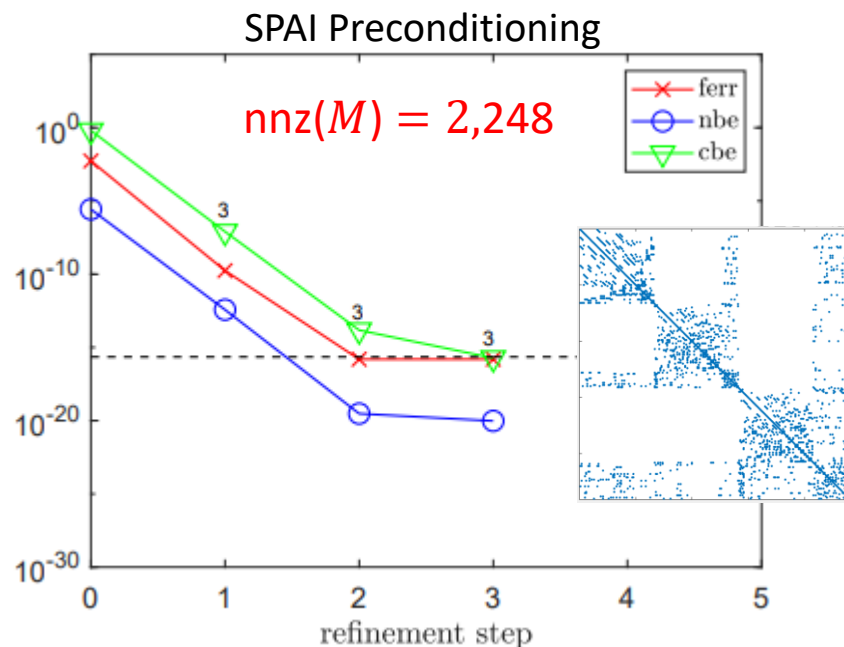
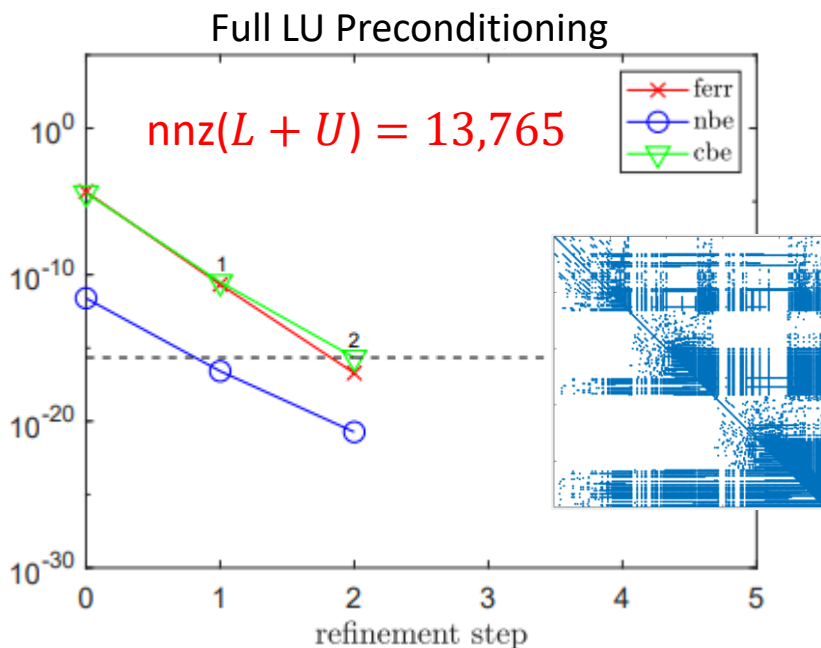


# Example: Iterative Refinement

Matrix: steam1,  $n = 240$ ,  $\text{nnz} = 2,248$ ,  $\kappa_\infty(A) = 3 \cdot 10^7$ ,  $\text{cond}(A^T) = 3 \cdot 10^3$

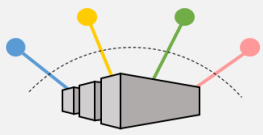


$u_s = \text{single}$ ,  $u = \text{double}$



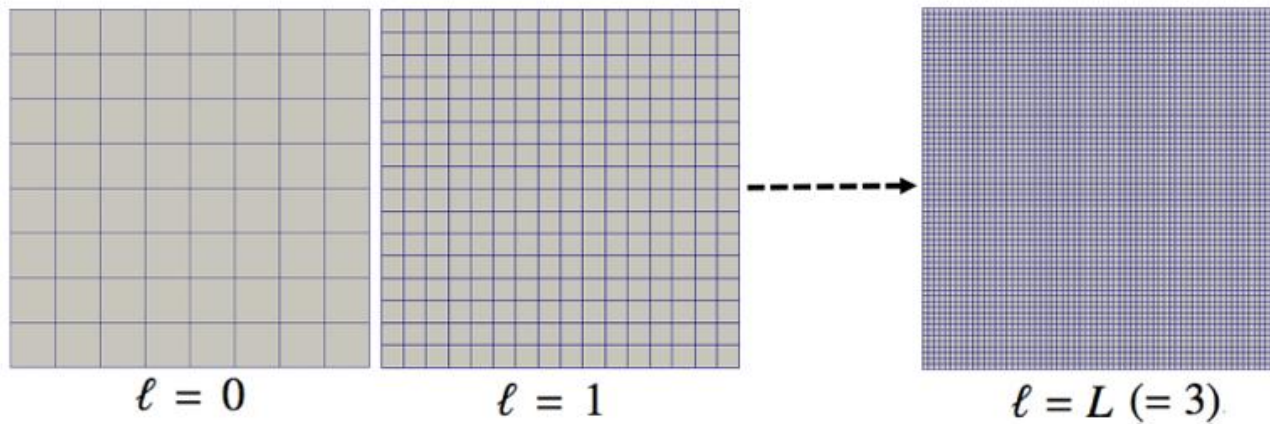
# Inexact Computations in Multilevel Monte Carlo

Martínek, C., and Scheichl. (2025). Exploiting Inexact Computations in Multilevel Sampling Methods. *arXiv preprint arXiv:2503.05533*.

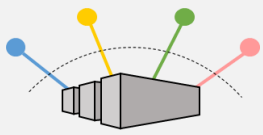


# Multilevel Monte Carlo Motivation

- Setting: want to sample from a probability distribution of a quantity  $Q$  which depends on the (infinite dimensional) solution of a PDE or SDE
- Basic approach: Use numerical model to obtain a high-fidelity finite-dimensional approximation  $Q_L$  of  $Q$ , use this model for sampling
- Multilevel approach (MLMC): Use hierarchy of models  $Q_0, \dots, Q_L$  with increasing accuracy; take a relatively small number of samples from the higher accuracy models

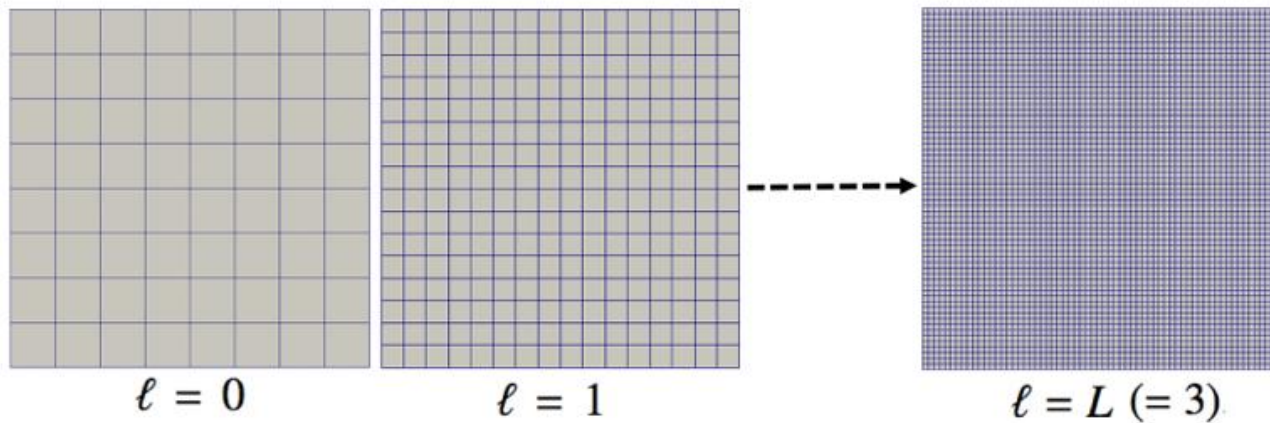


[Dodwell et al., 2021]



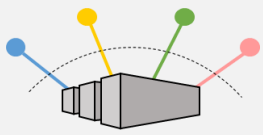
# Multilevel Monte Carlo Motivation

- Setting: want to sample from a probability distribution of a quantity  $Q$  which depends on the (infinite dimensional) solution of a PDE or SDE
- Basic approach: Use numerical model to obtain a high-fidelity finite-dimensional approximation  $Q_L$  of  $Q$ , use this model for sampling
- Multilevel approach (MLMC): Use hierarchy of models  $Q_0, \dots, Q_L$  with increasing accuracy; take a relatively small number of samples from the higher accuracy models



[Dodwell et al., 2021]

- **Insight:** Since we incur a larger amount of discretization error on coarser grids anyway, we can likely use inexact computations here without detriment



# Basic Approach

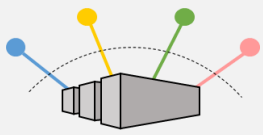
Model problem: PDE with random coefficients

$$-\nabla \cdot (a(\cdot, \omega) \nabla u(\cdot, \omega)) = f(\cdot, \omega)$$

$\omega$ : random parameter,  $a, f$ : infinite dimensional random fields

Quantity of interest  $Q$  is the mean value of a function of the solution

- Use FEM to construct computable approximations  $Q_\ell$  of  $Q$ 
  - Introduces discretization error
- Use Monte Carlo methods to approximate  $\mathbb{E}[Q_L]$ 
  - Introduces sampling error



# Basic Approach

Model problem: PDE with random coefficients

$$-\nabla \cdot (a(\cdot, \omega) \nabla u(\cdot, \omega)) = f(\cdot, \omega)$$

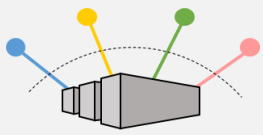
$\omega$ : random parameter,  $a, f$ : infinite dimensional random fields

Quantity of interest  $Q$  is the mean value of a function of the solution

- Use FEM to construct computable approximations  $Q_\ell$  of  $Q$ 
  - Introduces discretization error
- Use Monte Carlo methods to approximate  $\mathbb{E}[Q_L]$ 
  - Introduces sampling error

4 main steps:

1. Sample random fields  $a(\cdot, \omega), f(\cdot, \omega), \omega \in \Omega$
2. Assemble FEM stiffness matrix
3. Solve associated linear system
4. Compute MLMC estimate



# Basic Approach

Model problem: PDE with random coefficients

$$-\nabla \cdot (a(\cdot, \omega) \nabla u(\cdot, \omega)) = f(\cdot, \omega)$$

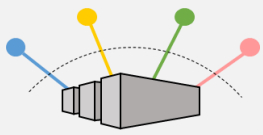
$\omega$ : random parameter,  $a, f$ : infinite dimensional random fields

Quantity of interest  $Q$  is the mean value of a function of the solution

- Use FEM to construct computable approximations  $Q_\ell$  of  $Q$ 
  - Introduces discretization error
- Use Monte Carlo methods to approximate  $\mathbb{E}[Q_L]$ 
  - Introduces sampling error

4 main steps:

1. Sample random fields  $a(\cdot, \omega), f(\cdot, \omega), \omega \in \Omega$
2. Assemble FEM stiffness matrix
3. Solve associated linear system ← Dominant cost
4. Compute MLMC estimate



# Basic Approach

Model problem: PDE with random coefficients

$$-\nabla \cdot (a(\cdot, \omega) \nabla u(\cdot, \omega)) = f(\cdot, \omega)$$

$\omega$ : random parameter,  $a, f$ : infinite dimensional random fields

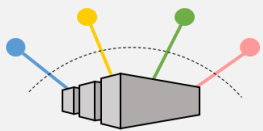
Quantity of interest  $Q$  is the mean value of a function of the solution

- Use FEM to construct computable approximations  $Q_\ell$  of  $Q$ 
  - Introduces discretization error
- Use Monte Carlo methods to approximate  $\mathbb{E}[Q_L]$ 
  - Introduces sampling error

4 main steps:

1. Sample random fields  $a(\cdot, \omega), f(\cdot, \omega), \omega \in \Omega$
2. Assemble FEM stiffness matrix
3. Solve associated linear system ← Dominant cost
4. Compute MLMC estimate

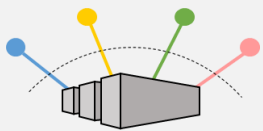
**Goal: Balance the computational error in solving the linear system with the sampling and discretization errors**



# Existing Results on Error in MLMC

MLMC estimator:

$$\hat{Q}_{L, \{N_\ell\}} = \sum_{\ell=0}^L \frac{1}{N_\ell} \sum_{k=1}^{N_\ell} (Q_\ell^{(k)} - Q_{\ell-1}^{(k)})$$



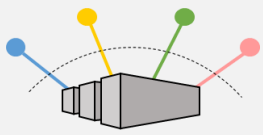
# Existing Results on Error in MLMC

MLMC estimator:

$$\hat{Q}_{L, \{N_\ell\}} = \sum_{\ell=0}^L \frac{1}{N_\ell} \sum_{k=1}^{N_\ell} (Q_\ell^{(k)} - Q_{\ell-1}^{(k)})$$

Mean Square Error (MSE): (under certain assumptions) [Cliffe et al., 2011]

$$\begin{aligned} \mathbb{E} \left[ (\mathbb{E}[Q] - \hat{Q}_{L, \{N_\ell\}})^2 \right] &= (\mathbb{E}[Q - Q_L])^2 + \sum_{\ell=0}^L \frac{\text{Var}[Q_\ell - Q_{\ell-1}]}{N_\ell} \\ &= C \left( h_L^{2\alpha_1} + \sum_{\ell=0}^L \frac{h_\ell^{\beta_1}}{N_\ell} \right) \end{aligned}$$



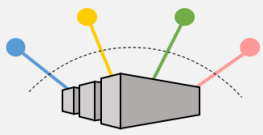
# Existing Results on Error in MLMC

MLMC estimator:

$$\hat{Q}_{L, \{N_\ell\}} = \sum_{\ell=0}^L \frac{1}{N_\ell} \sum_{k=1}^{N_\ell} (Q_\ell^{(k)} - Q_{\ell-1}^{(k)})$$

Mean Square Error (MSE): (under certain assumptions) [Cliffe et al., 2011]

$$\begin{aligned} \mathbb{E} \left[ (\mathbb{E}[Q] - \hat{Q}_{L, \{N_\ell\}})^2 \right] &= (\mathbb{E}[Q - Q_L])^2 + \sum_{\ell=0}^L \frac{\text{Var}[Q_\ell - Q_{\ell-1}]}{N_\ell} \\ &= C \left( h_L^{2\alpha_1} + \sum_{\ell=0}^L \frac{h_\ell^{\beta_1}}{N_\ell} \right) \\ &= \text{bias} + \text{variance} \end{aligned}$$



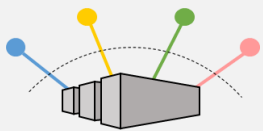
# Existing Results on Error in MLMC

MLMC estimator:

$$\hat{Q}_{L, \{N_\ell\}} = \sum_{\ell=0}^L \frac{1}{N_\ell} \sum_{k=1}^{N_\ell} (Q_\ell^{(k)} - Q_{\ell-1}^{(k)})$$

Mean Square Error (MSE): (under certain assumptions) [Cliffe et al., 2011]

$$\begin{aligned} \mathbb{E} \left[ (\mathbb{E}[Q] - \hat{Q}_{L, \{N_\ell\}})^2 \right] &= (\mathbb{E}[Q - Q_L])^2 + \sum_{\ell=0}^L \frac{\text{Var}[Q_\ell - Q_{\ell-1}]}{N_\ell} \\ &= C \left( h_L^{2\alpha_1} + \sum_{\ell=0}^L \frac{h_\ell^{\beta_1}}{N_\ell} \right) \\ &= \text{bias} + \text{variance} \\ &\quad \downarrow \\ &\sim \text{discretization error} \end{aligned}$$



# Existing Results on Error in MLMC

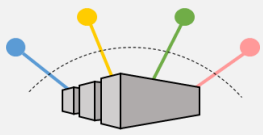
MLMC estimator:

$$\hat{Q}_{L, \{N_\ell\}} = \sum_{\ell=0}^L \frac{1}{N_\ell} \sum_{k=1}^{N_\ell} (Q_\ell^{(k)} - Q_{\ell-1}^{(k)})$$

Mean Square Error (MSE): (under certain assumptions) [Cliffe et al., 2011]

$$\begin{aligned} \mathbb{E} \left[ (\mathbb{E}[Q] - \hat{Q}_{L, \{N_\ell\}})^2 \right] &= (\mathbb{E}[Q - Q_L])^2 + \sum_{\ell=0}^L \frac{\text{Var}[Q_\ell - Q_{\ell-1}]}{N_\ell} \\ &= C \left( h_L^{2\alpha_1} + \sum_{\ell=0}^L \frac{h_\ell^{\beta_1}}{N_\ell} \right) \\ &= \text{bias} + \text{variance} \\ &\quad \downarrow \\ &\sim \text{discretization error} \end{aligned}$$

**This assumes the linear systems are solved exactly!**



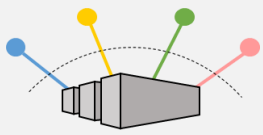
# Computational error in MLMC

Let  $A_\ell x_\ell = b_\ell$  be the linear system from the FEM corresponding to level  $\ell$  (with mesh parameter  $h_\ell$ ). Let  $\hat{x}_\ell$  be the computed solution of this system satisfying

$$\frac{\|b_\ell - A_\ell \hat{x}_\ell\|_2}{\|b_\ell\|_2} \leq \delta_\ell.$$

Then

$$MSE \leq C \left( h_L^{2\alpha_1} + \sum_{\ell=0}^L \frac{h_\ell^{\beta_1}}{N_\ell} \right) + \left( h_L^{\alpha_1} \delta_L^{2\alpha_2} + \delta_L^{2\alpha_2} + \sum_{\ell=0}^L \frac{\delta_\ell^{\beta_2}}{N_\ell} \right)$$



# Computational error in MLMC

Let  $A_\ell x_\ell = b_\ell$  be the linear system from the FEM corresponding to level  $\ell$  (with mesh parameter  $h_\ell$ ). Let  $\hat{x}_\ell$  be the computed solution of this system satisfying

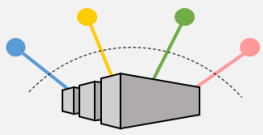
$$\frac{\|b_\ell - A_\ell \hat{x}_\ell\|_2}{\|b_\ell\|_2} \leq \delta_\ell.$$

Then

$$MSE \leq C \left( h_L^{2\alpha_1} + \sum_{\ell=0}^L \frac{h_\ell^{\beta_1}}{N_\ell} \right) + \left( h_L^{\alpha_1} \delta_L^{2\alpha_2} + \delta_L^{2\alpha_2} + \sum_{\ell=0}^L \frac{\delta_\ell^{\beta_2}}{N_\ell} \right)$$

standard MLMC error

computational error



# Computational error in MLMC

Let  $A_\ell x_\ell = b_\ell$  be the linear system from the FEM corresponding to level  $\ell$  (with mesh parameter  $h_\ell$ ). Let  $\hat{x}_\ell$  be the computed solution of this system satisfying

$$\frac{\|b_\ell - A_\ell \hat{x}_\ell\|_2}{\|b_\ell\|_2} \leq \delta_\ell.$$

Then

$$MSE \leq C \left( h_L^{2\alpha_1} + \sum_{\ell=0}^L \frac{h_\ell^{\beta_1}}{N_\ell} \right) + \left( h_L^{\alpha_1} \delta_L^{2\alpha_2} + \delta_L^{2\alpha_2} + \sum_{\ell=0}^L \frac{\delta_\ell^{\beta_2}}{N_\ell} \right)$$

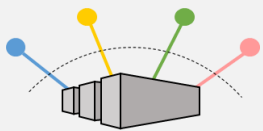
standard MLMC error

computational error

→ Want to choose accuracy  $\delta_\ell$  on level  $\ell$  such that total MSE is not greater than a constant  $k_p \in (0,1)$  of the standard MLMC error:

$$\delta_\ell = \left( k_p h_\ell^{\beta_1} \right)^{1/\beta_2}, \quad \ell = 0, \dots, L-1$$

$$\delta_L = \min \left\{ \left( k_p h_L^{\beta_1} \right)^{1/\beta_2}, \left( k_p h_L^{\alpha_1} \right)^{1/\alpha_2} \right\}$$



# Computational error in MLMC

Let  $A_\ell x_\ell = b_\ell$  be the linear system from the FEM corresponding to level  $\ell$  (with mesh parameter  $h_\ell$ ). Let  $\hat{x}_\ell$  be the computed solution of this system satisfying

$$\frac{\|b_\ell - A_\ell \hat{x}_\ell\|_2}{\|b_\ell\|_2} \leq \delta_\ell.$$

Then

$$MSE \leq C \left( h_L^{2\alpha_1} + \sum_{\ell=0}^L \frac{h_\ell^{\beta_1}}{N_\ell} \right) + \left( h_L^{\alpha_1} \delta_L^{2\alpha_2} + \delta_L^{2\alpha_2} + \sum_{\ell=0}^L \frac{\delta_\ell^{\beta_2}}{N_\ell} \right)$$

standard MLMC error

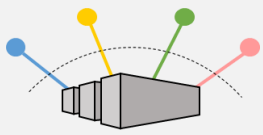
computational error

→ Want to choose accuracy  $\delta_\ell$  on level  $\ell$  such that total MSE is not greater than a constant  $k_p \in (0,1)$  of the standard MLMC error:

$$\delta_\ell = \left( k_p h_\ell^{\beta_1} \right)^{1/\beta_2}, \quad \ell = 0, \dots, L-1$$

$$\delta_L = \min \left\{ \left( k_p h_L^{\beta_1} \right)^{1/\beta_2}, \left( k_p h_L^{\alpha_1} \right)^{1/\alpha_2} \right\}$$



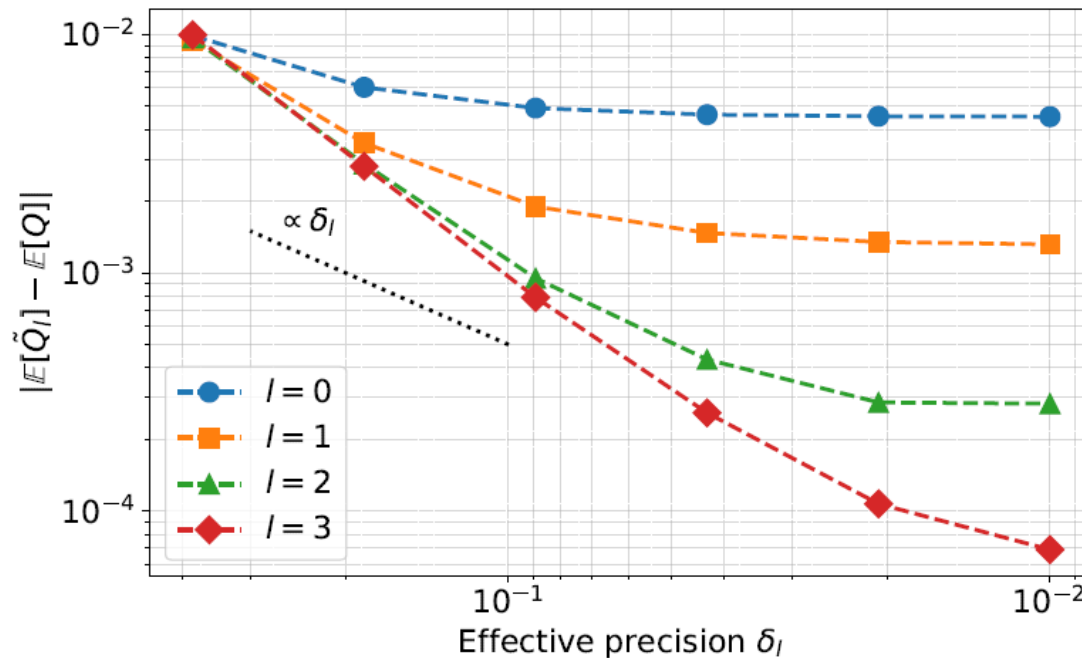


# Results: Elliptic PDE with lognormal coefficients

Variation on model problem: 
$$-\nabla \cdot (a(\cdot, \omega) \nabla u(\cdot, \omega)) = f \quad \text{on } D,$$

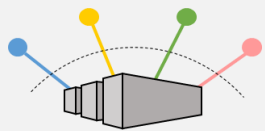
$$u(\cdot, \omega) = 0 \quad \text{on } \partial D$$

$f \equiv 1$  on  $D = (0,1)^2$ ,  $h_\ell = \frac{1}{8} \times 2^{-\ell}$ , discretized using FEM via FEniCSx  
 $\alpha_1 = 2$ ,  $\beta_1 = 4$ ,  $\alpha_2 = 1$ ,  $\beta_2 = 2$



Line flattens =  
 discretization error  
 starts to dominate

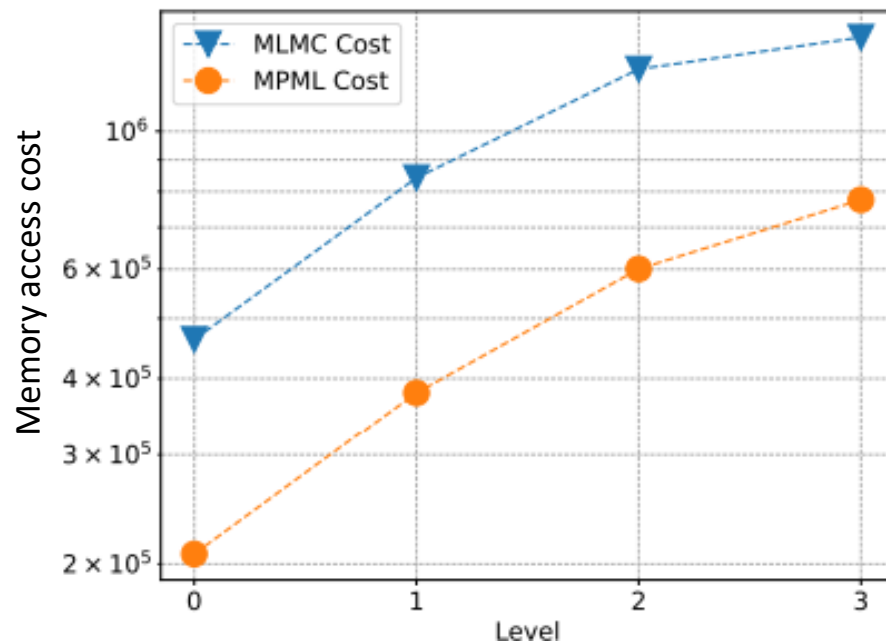
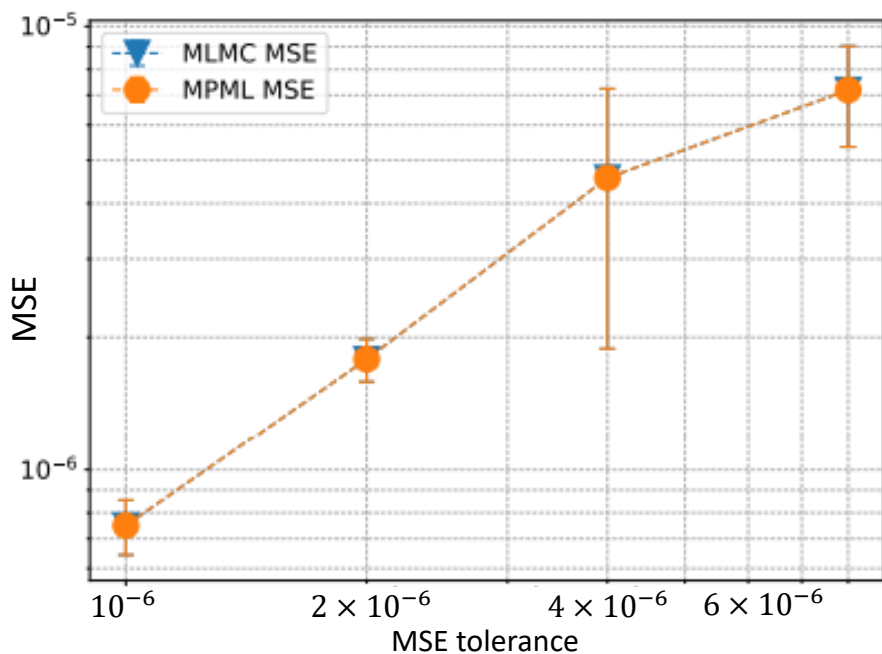
Solver: MINRES with stopping criterion set to  $\delta_\ell$



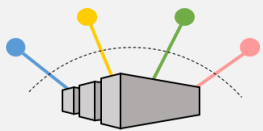
# Results: Adaptive approach w/ mixed precision direct solver

MLMC Solver: Double precision Cholesky  
MPML Solver: Low precision Cholesky + IR,  
 $k_p = 0.05$

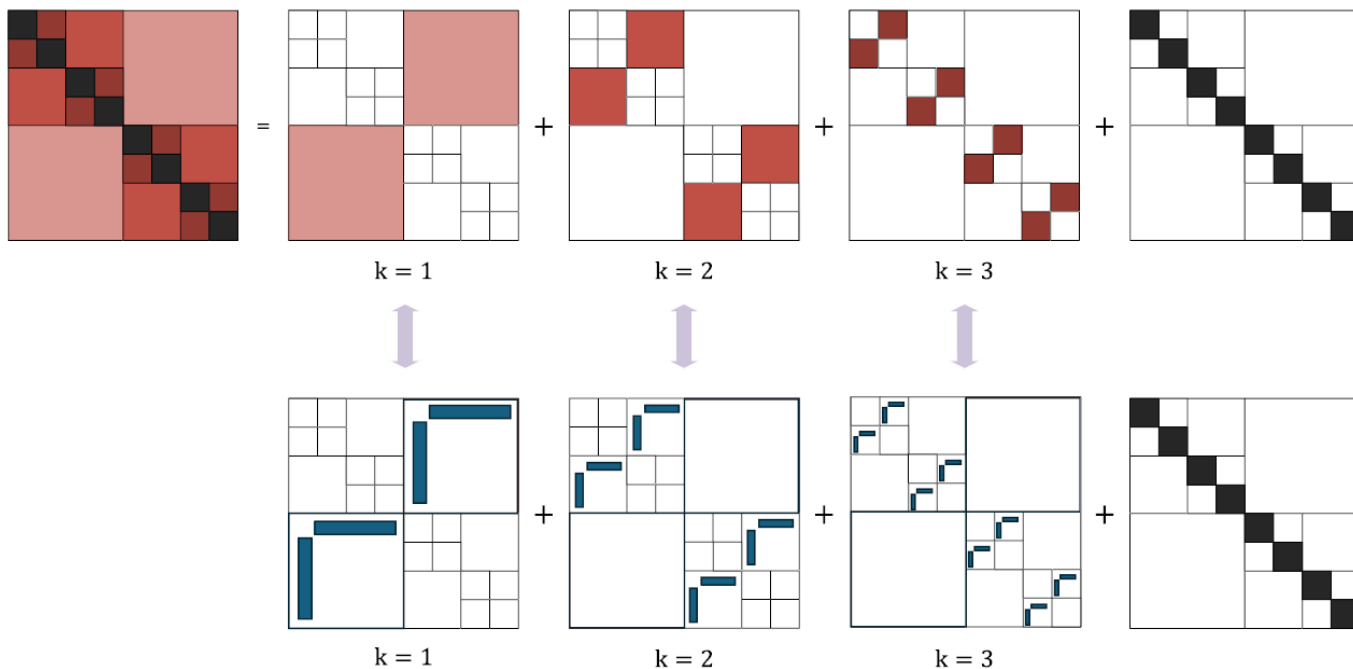
$\ell$	$(\mathbf{u}_f, \mathbf{u}, \mathbf{u}_r)$
0	HSS
1	SSS
2+	SDD



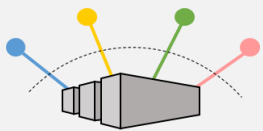
# Mixed Precision Hierarchical Matrix Approximations



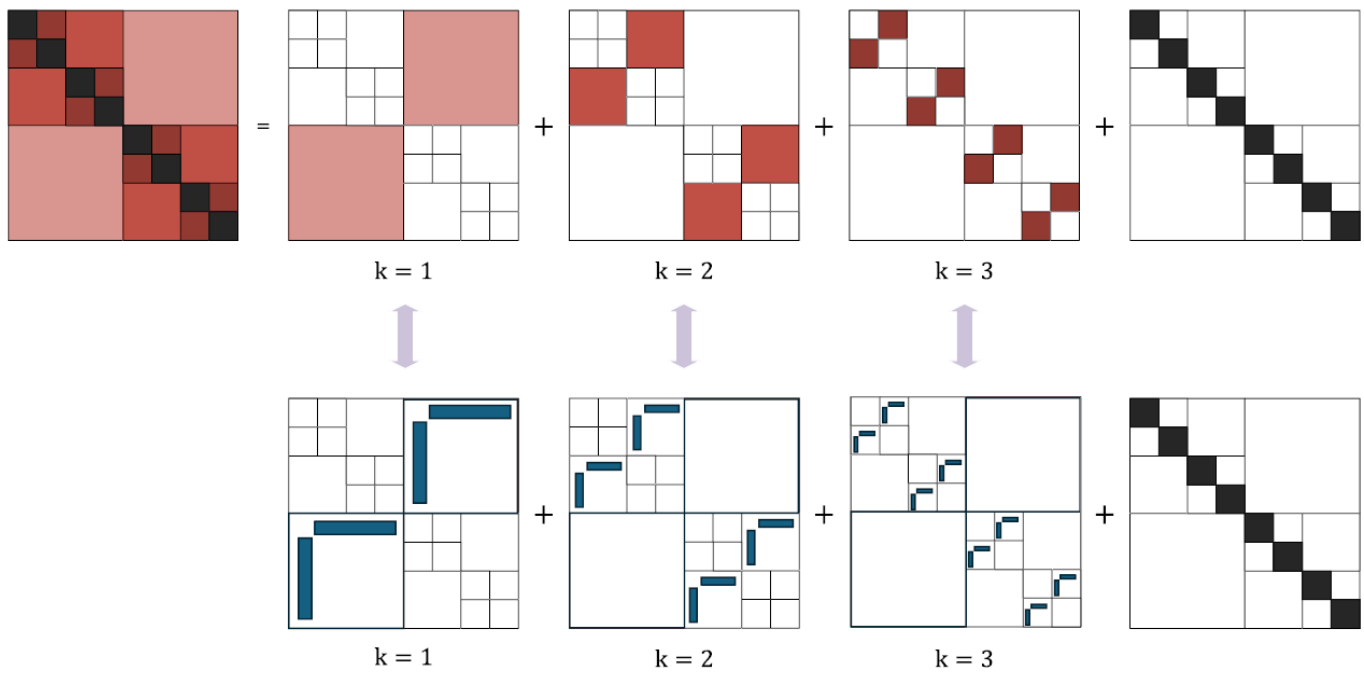
# HODLR Matrix Format



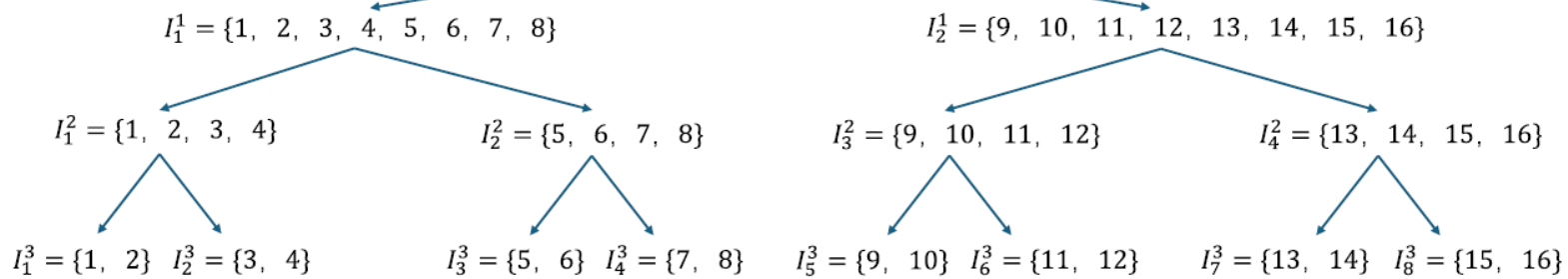
- Hierarchical Off-Diagonal Low-Rank (HODLR) matrices
- Fixed hierarchical block structure, low-rank ( $p$ ) approximation of all off-diagonal blocks
- Enables efficient computation, e.g.,  $O(n^2)$  computations become  $O(pn \log n)$

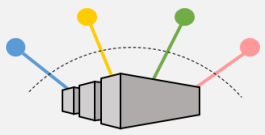


# HODLR Matrix Format



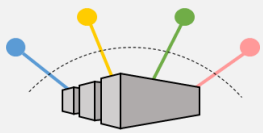
$I = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}$





# Computations with HODLR matrices

How should we set the working precision  $u$  in computations with mixed precision HODLR matrices so that the backward error of the computation does not exceed the error resulting from inexact representation of the matrix?



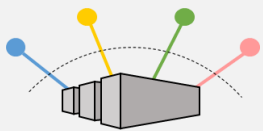
# Computations with HODLR matrices

How should we set the working precision  $u$  in computations with mixed precision HODLR matrices so that the backward error of the computation does not exceed the error resulting from inexact representation of the matrix?

Theorem: For both matrix-vector products and LU factorizations with HODLR matrices stored in the mixed precision HODLR representation, as long as

$$u \leq \frac{\epsilon}{n},$$

the backward error in these computations is dominated by the matrix representation error.



# Computations with HODLR matrices

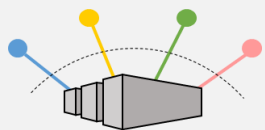
How should we set the working precision  $u$  in computations with mixed precision HODLR matrices so that the backward error of the computation does not exceed the error resulting from inexact representation of the matrix?

Theorem: For both matrix-vector products and LU factorizations with HODLR matrices stored in the mixed precision HODLR representation, as long as

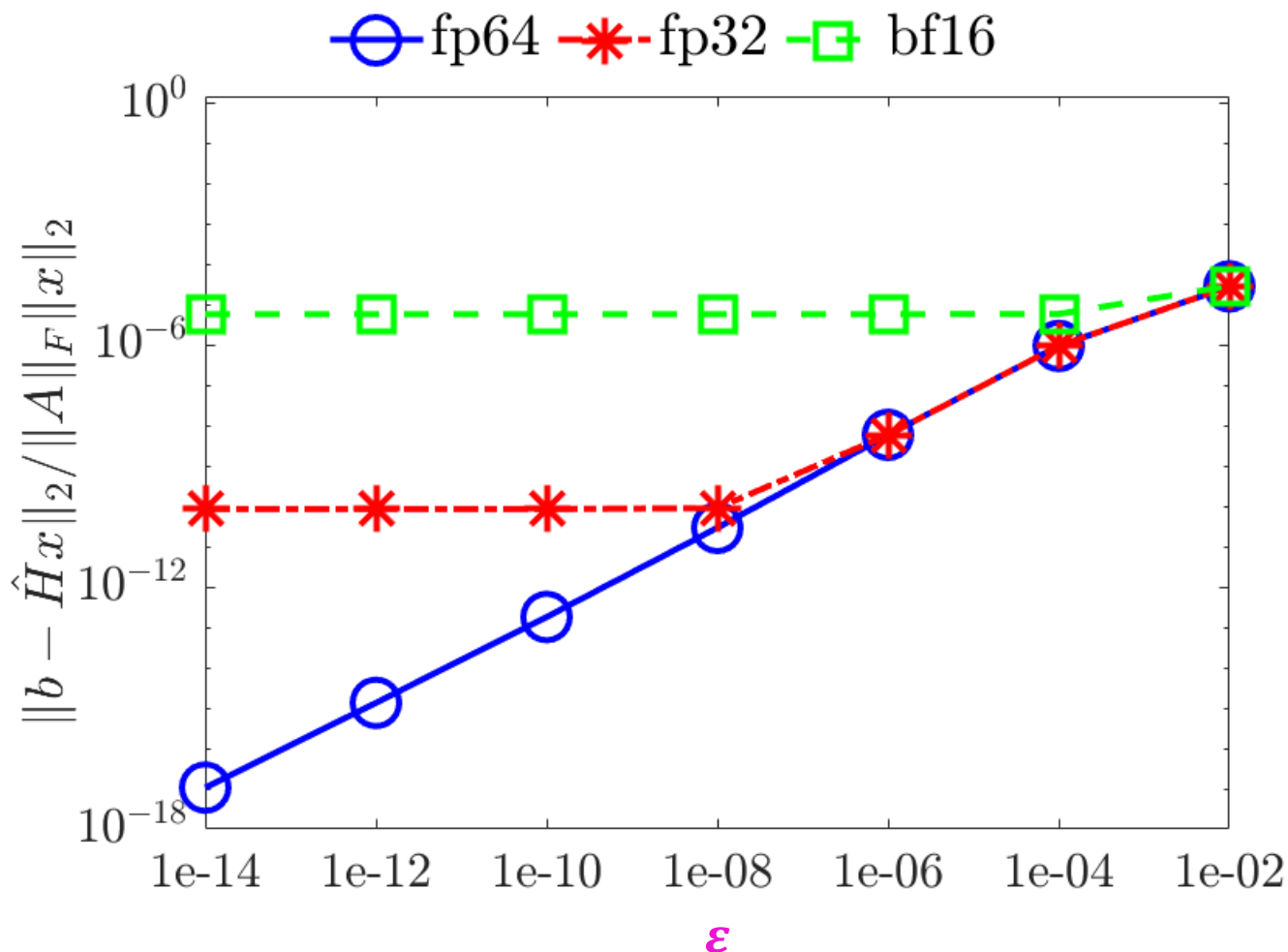
$$u \leq \frac{\varepsilon}{n},$$

the backward error in these computations is dominated by the matrix representation error.

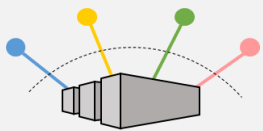




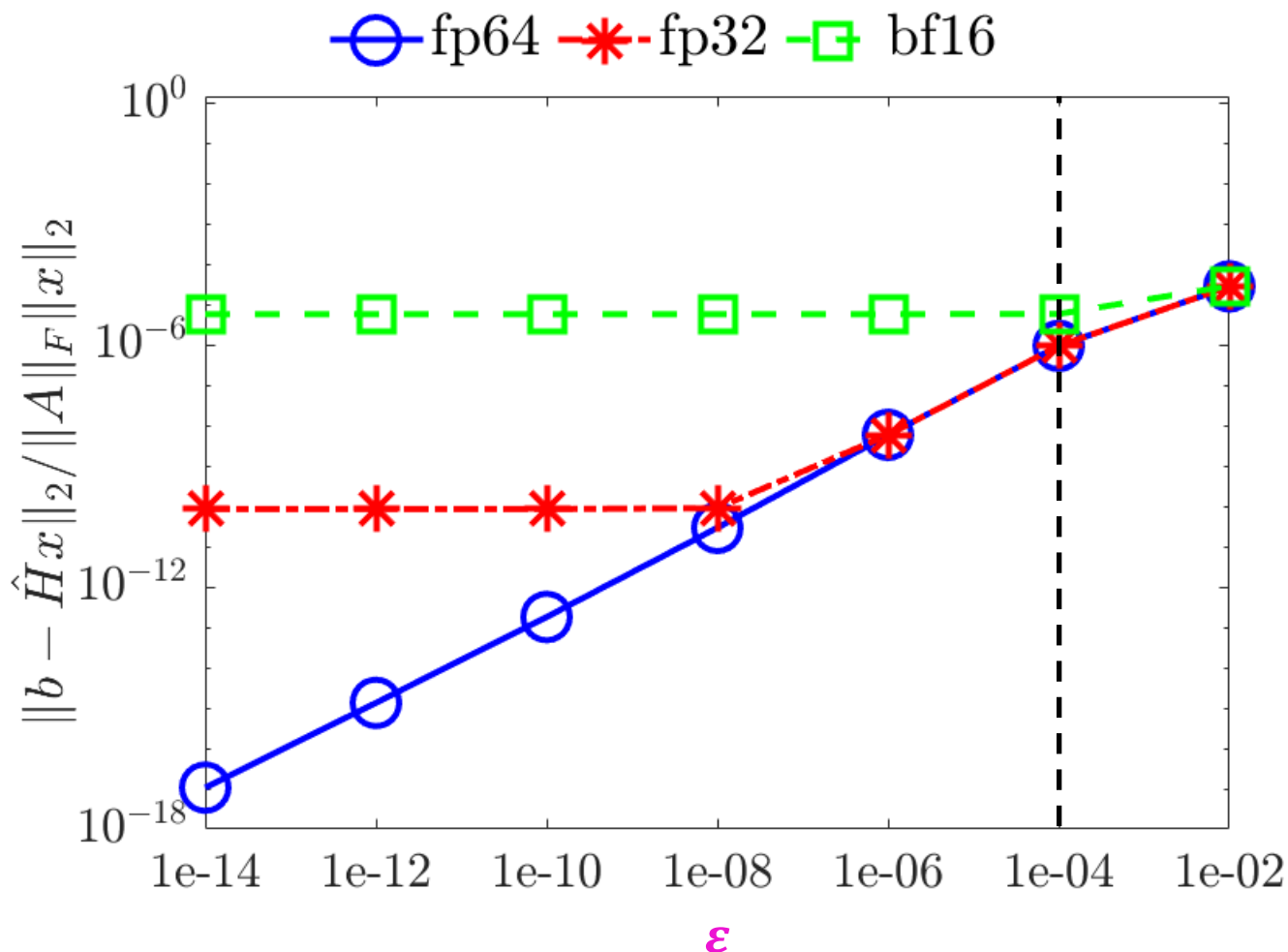
# Example: Matrix-vector products



matrix: Gaussian radial basis function evaluated at set of uniform grid points in  $[-1,1] \times [-1,1]$ ,  $n = 2000$ ,  $\ell = 8$

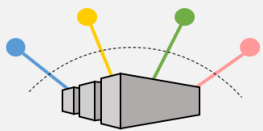


# Example: Matrix-vector products



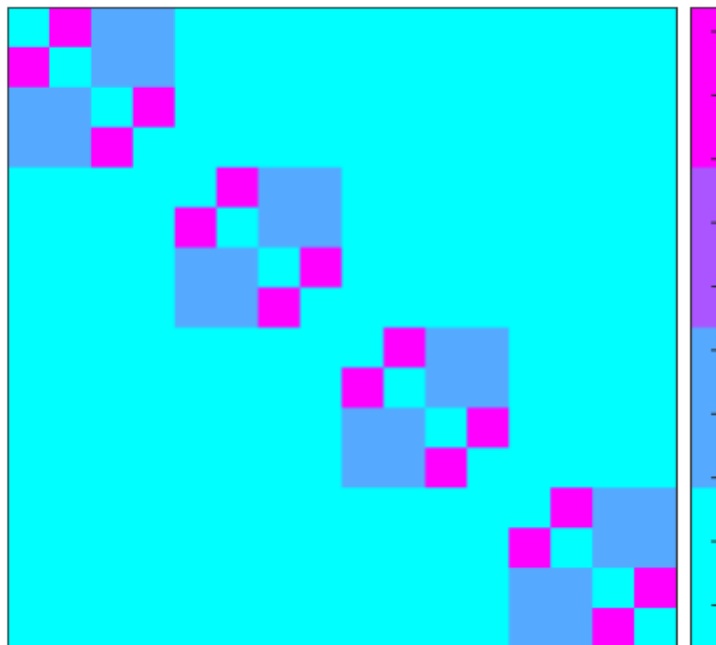
**If  $\epsilon \geq un$ , then approximation error will dominate.**

matrix: Gaussian radial basis function evaluated at set of uniform grid points in  $[-1,1] \times [-1,1]$ ,  $n = 2000$ ,  $\ell = 8$



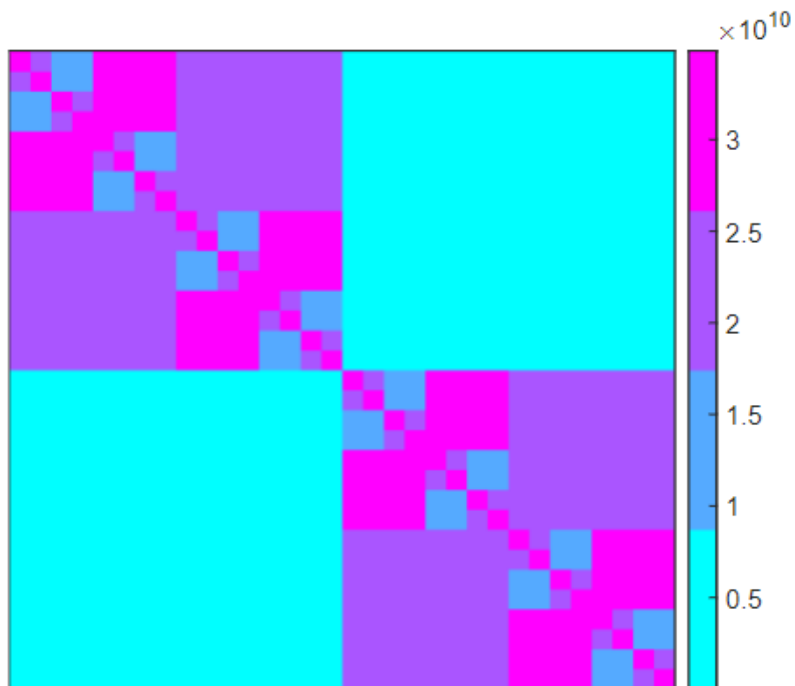
# Example

$$\varepsilon = 10^{-4}, \ell = 6$$



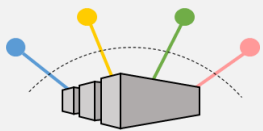
(a) saylr3

{bf16, fp16, fp16, fp32, fp32, fp32}

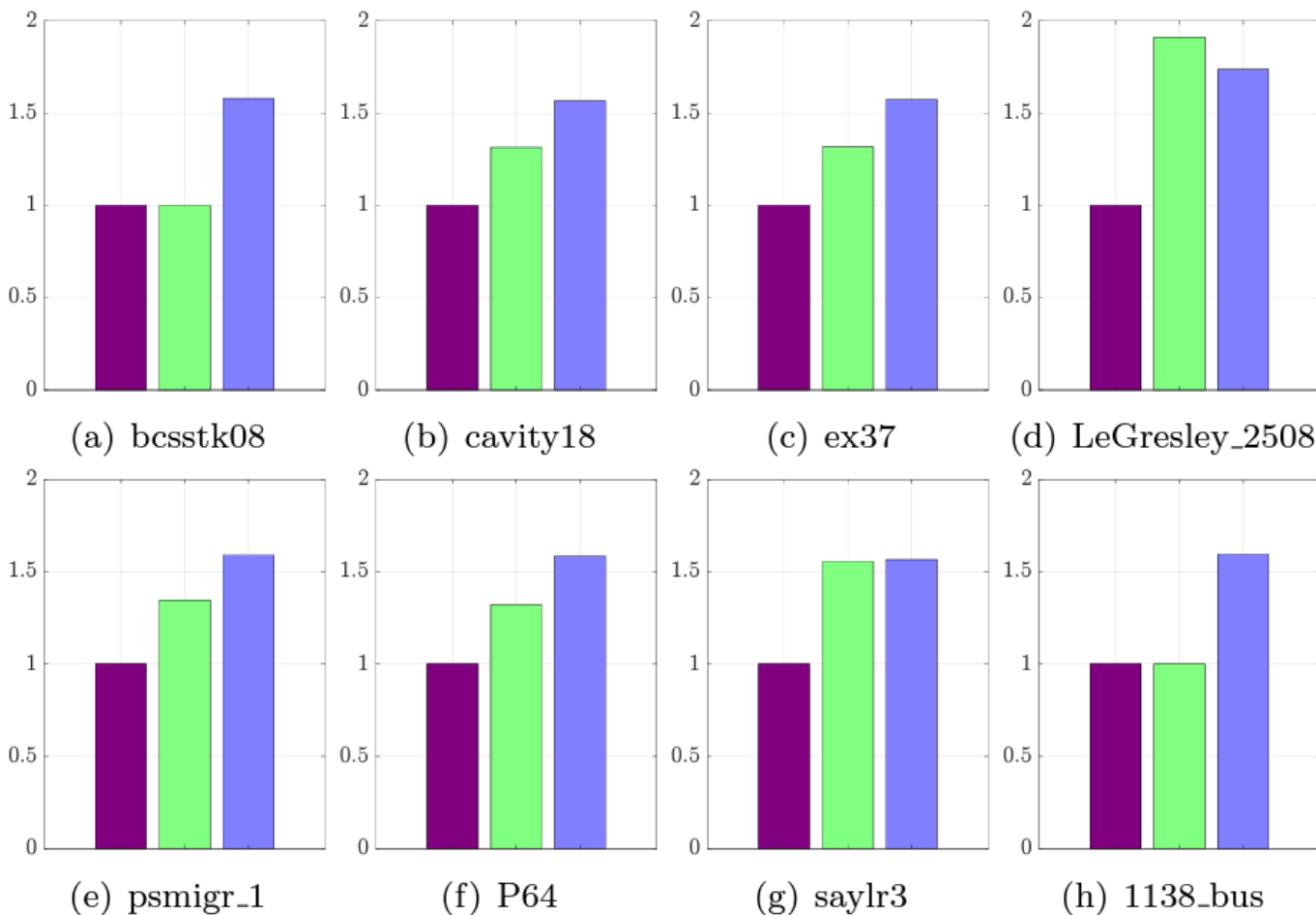


(b) LeGresley\_2508

{q52, fp32, fp32, fp32, fp32, fp32}



# Storage savings



Storage savings of adaptive precision HODLR matrices relative to uniform (double) precision

$\epsilon = 10^{-7}$     $\epsilon = 10^{-4}$     $\epsilon = 10^{-1}$

[https://github.com/inEXASCALE/mphodlr\\_exp](https://github.com/inEXASCALE/mphodlr_exp)