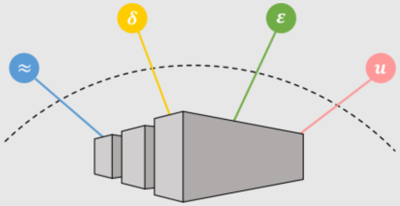


Balancing Inexactness in Large-Scale Matrix Computations

Erin C. Carson
Charles University

Nordic Numerical Linear Algebra Meeting 2024
June 17, 2024

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

Co-funded by the
European Union

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

The Exascale Era

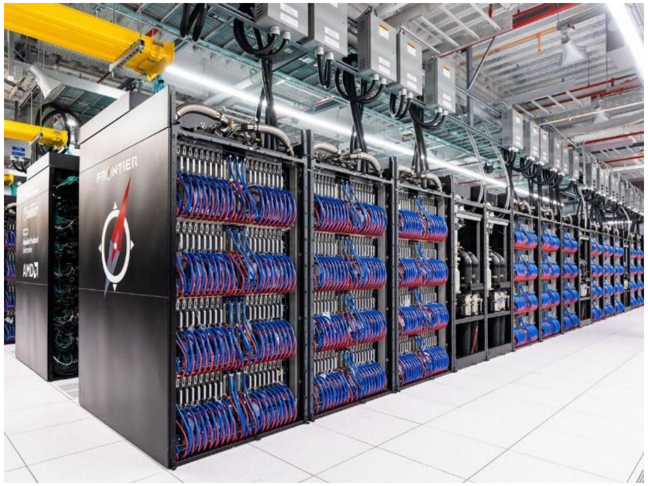
We have now entered the “Exascale Era”

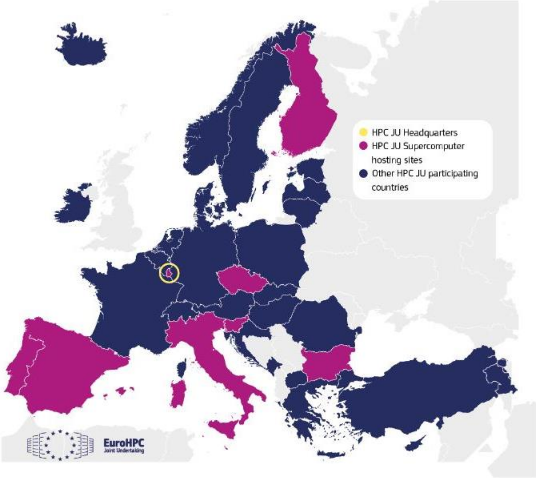
- 10^{18} floating point operations per second

The Exascale Era

We have now entered the “Exascale Era”

- 10^{18} floating point operations per second



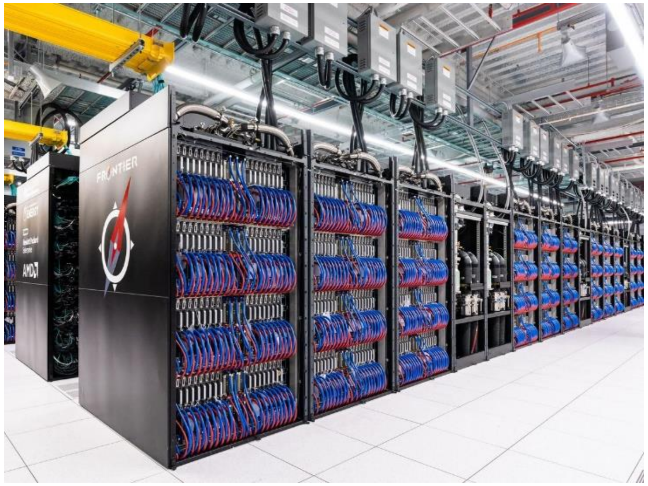

● HPC JU Headquarters
● HPC JU Supercomputer hosting sites
● Other HPC JU participating countries
EuroHPC
Joint Undertaking

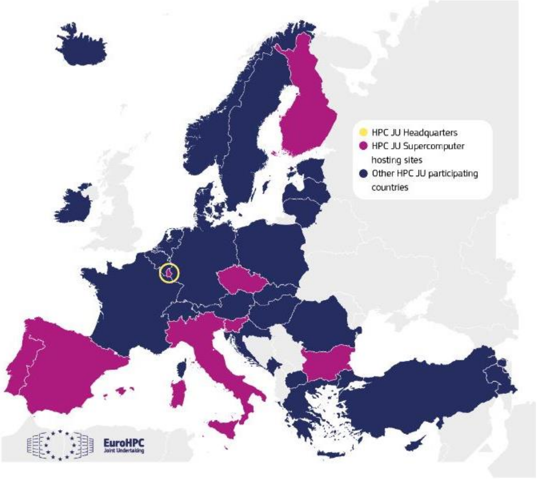
https://eurohpc-ju.europa.eu/pictures

The Exascale Era

We have now entered the “Exascale Era”

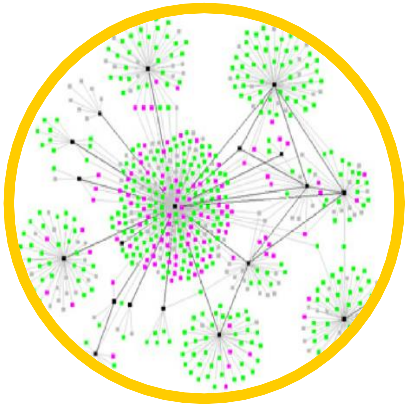
- 10^{18} floating point operations per second

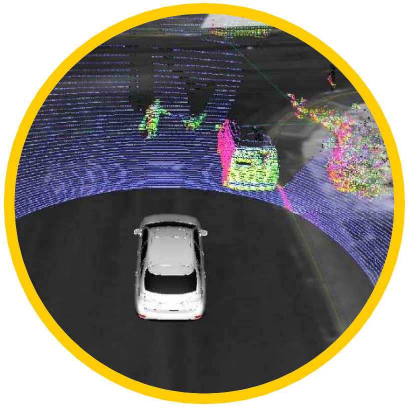


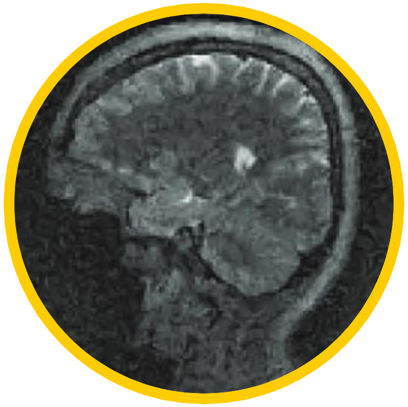

● HPC JU Headquarters
● HPC JU Supercomputer hosting sites
● Other HPC JU participating countries
EuroHPC Joint Undertaking

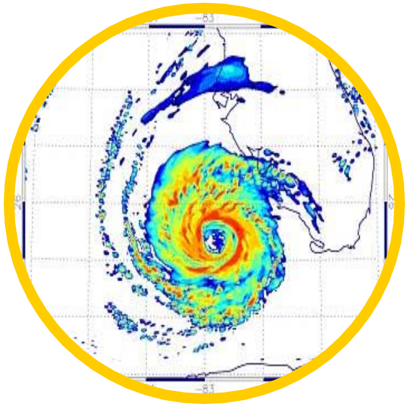
https://eurohpc-ju.europa.eu/pictures

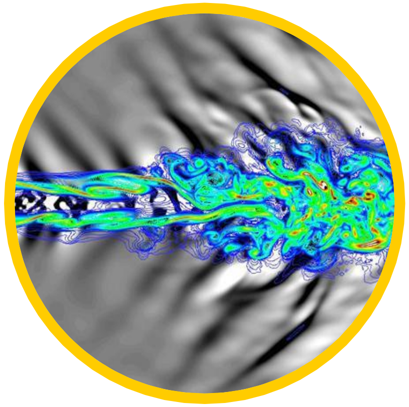
Significant opportunity … Significant challenges

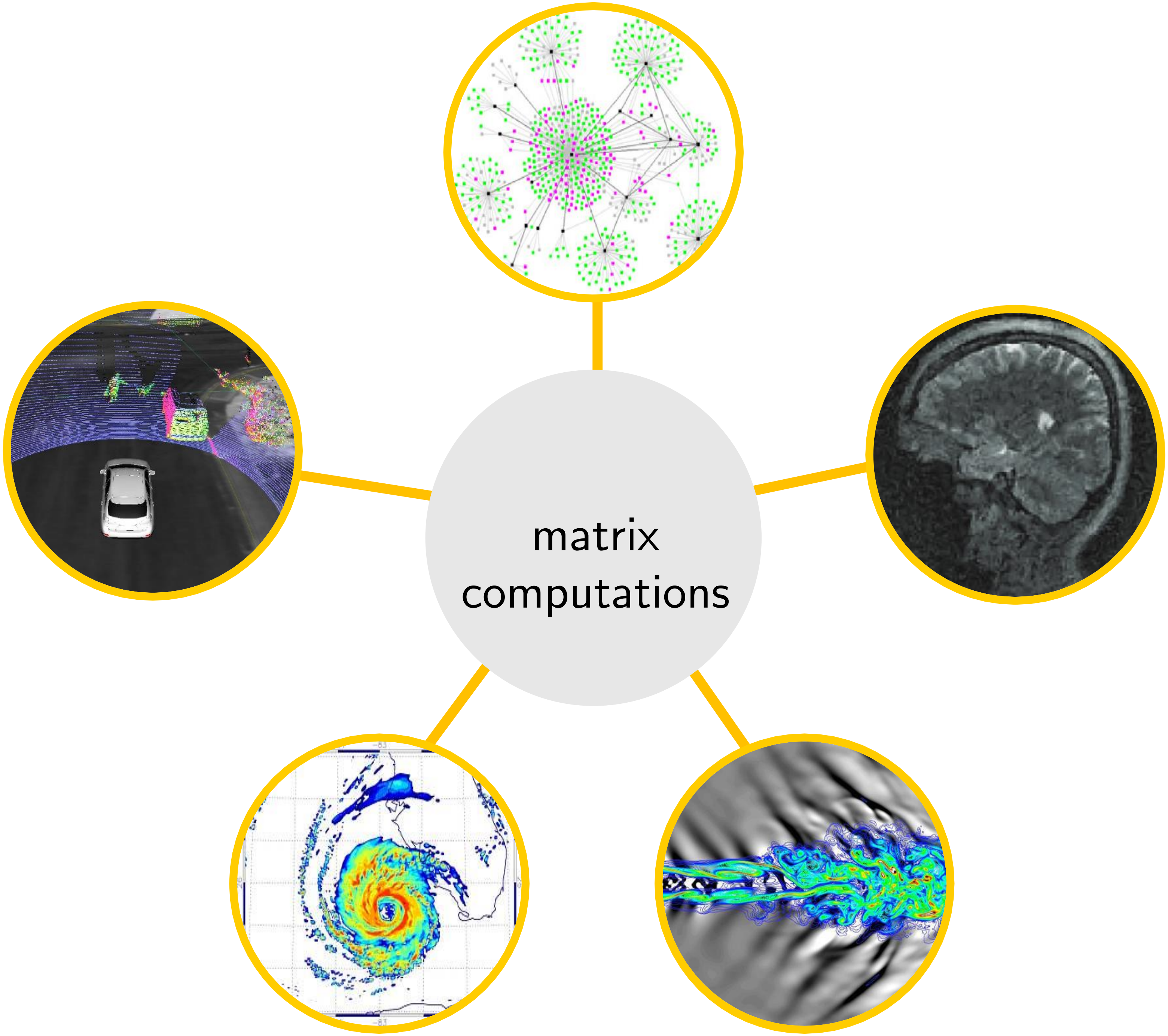




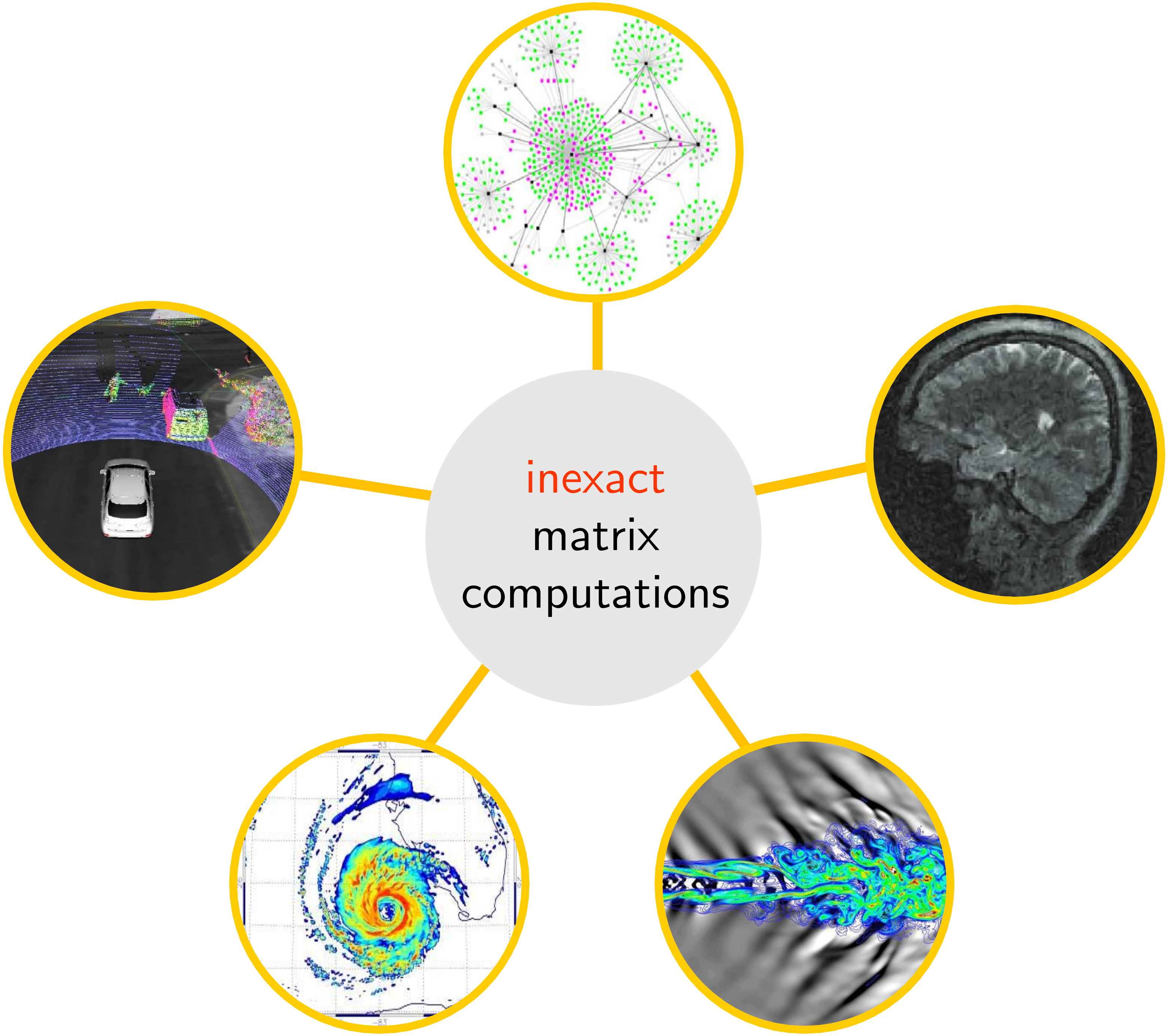






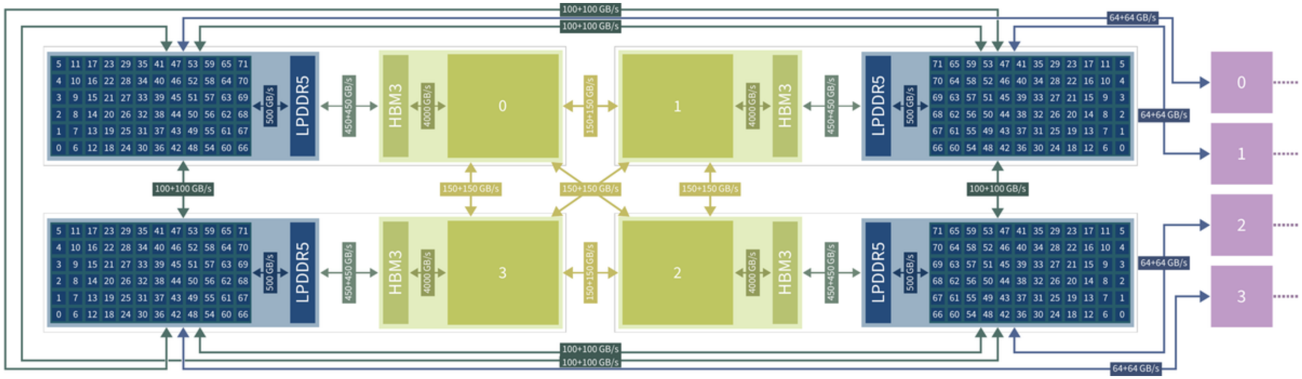


matrix computations



inexact
matrix
computations

Exascale Hardware



5	11	17	23	29	35	41	47	53	59	65	71
4	10	16	22	28	34	40	46	52	58	64	70
3	9	15	21	27	33	39	45	51	57	63	69
2	8	14	20	26	32	38	44	50	56	62	68
1	7	13	19	25	31	37	43	49	55	61	67
0	6	12	18	24	30	36	42	48	54	60	66

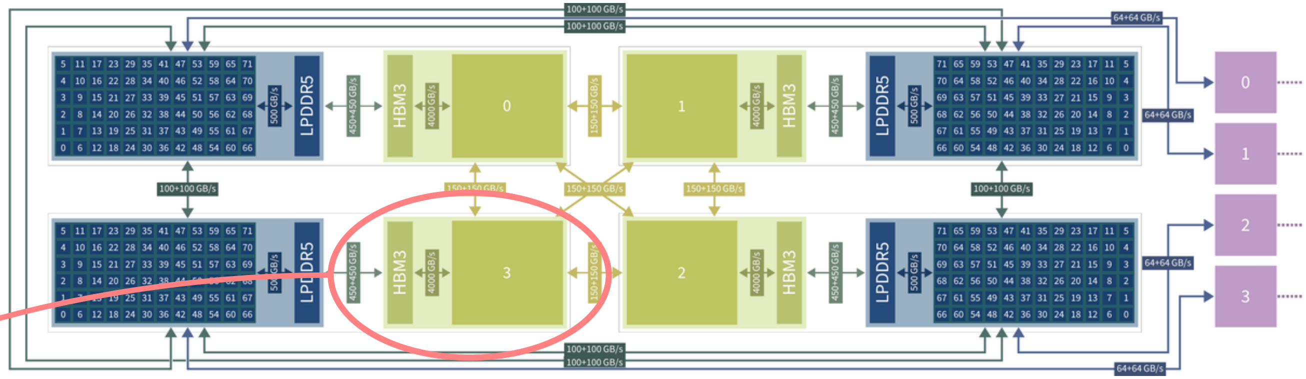
71	65	59	53	47	41	35	29	23	17	11	5
70	64	58	52	46	40	34	28	22	16	10	4
69	63	57	51	45	39	33	27	21	15	9	3
68	62	56	50	44	38	32	26	20	14	8	2
67	61	55	49	43	37	31	25	19	13	7	1
66	60	54	48	42	36	30	24	18	12	6	0

5	11	17	23	29	35	41	47	53	59	65	71
4	10	16	22	28	34	40	46	52	58	64	70
3	9	15	21	27	33	39	45	51	57	63	69
2	8	14	20	26	32	38	44	50	56	62	68
1	7	13	19	25	31	37	43	49	55	61	67
0	6	12	18	24	30	36	42	48	54	60	66

71	65	59	53	47	41	35	29	23	17	11	5
70	64	58	52	46	40	34	28	22	16	10	4
69	63	57	51	45	39	33	27	21	15	9	3
68	62	56	50	44	38	32	26	20	14	8	2
67	61	55	49	43	37	31	25	19	13	7	1
66	60	54	48	42	36	30	24	18	12	6	0

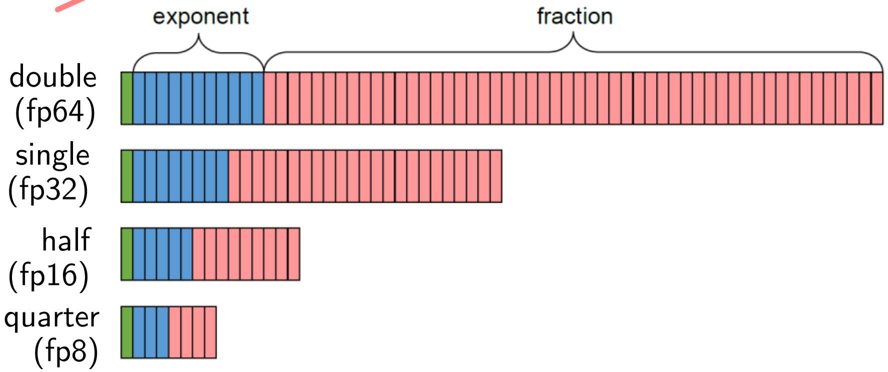
https://www.fz-juelich.de/en/ias/jsc/jupiter/tech

Exascale Hardware



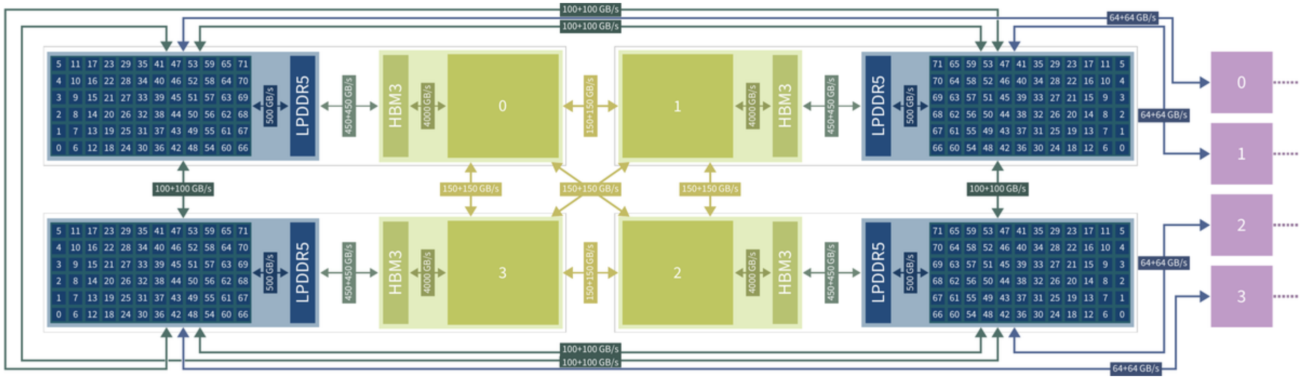
https://www.fz-juelich.de/en/ias/jsc/jupiter/tech

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



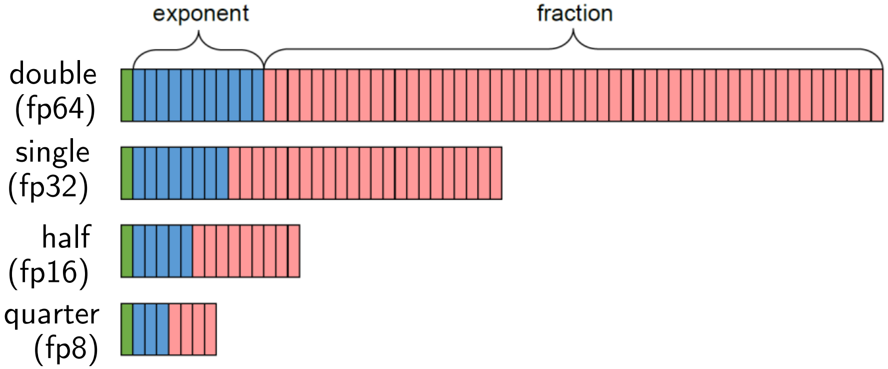
	size (bits)	range	u	perf. (NVIDIA H100)
fp64	64	$10^{\pm 308}$	1×10^{-16}	60 Tflops/s
fp32	32	$10^{\pm 38}$	6×10^{-8}	1 Pflop/s
fp16	16	$10^{\pm 5}$	5×10^{-4}	2 Pflops/s
bfloat16	16	$10^{\pm 38}$	4×10^{-3}	
fp8-e5m2	8	$10^{\pm 5}$	1×10^{-1}	4 Pflops/s
fp8-e4m3	8	$10^{\pm 2}$	6×10^{-2}	

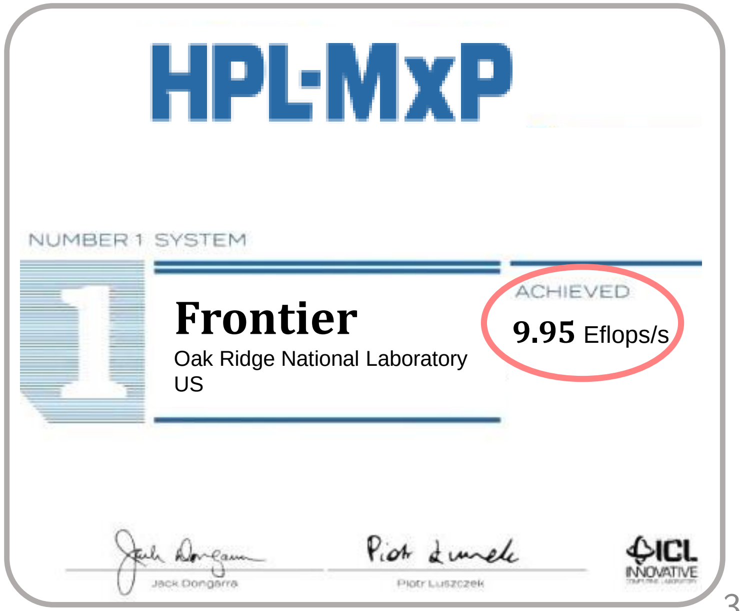
Exascale Hardware



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

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




HPL-MxP

NUMBER 1 SYSTEM

Frontier

Oak Ridge National Laboratory
US

ACHIEVED

9.95 Eflops/s

Jack Dongarra

Piotr Luszczek

ICL INNOVATIVE

Mixed precision in NLA

- BLAS: cuBLAS, MAGMA, [Agullo et al. 2009], [Abdelfattah et al., 2019], [Haidar et al., 2018]
- Iterative refinement:
  - Long history: [Wilkinson, 1963], [Moler, 1967], [Stewart, 1973], ...
  - More recently: [Langou et al., 2006], [C., Higham, 2017], [C., Higham, 2018], [C., Higham, Pranesh, 2020], [Amestoy et al., 2021]
- Matrix factorizations: [Haidar et al., 2017], [Haidar et al., 2018], [Haidar et al., 2020], [Abdelfattah et al., 2020]
- Eigenvalue problems: [Dongarra, 1982], [Dongarra, 1983], [Tisseur, 2001], [Davies et al., 2001], [Petschow et al., 2014], [Alvermann et al., 2019]
- Sparse direct solvers: [Buttari et al., 2008]
- Orthogonalization: [Yamazaki et al., 2015]
- Multigrid: [Tamstorf et al., 2020], [Richter et al., 2014], [Sumiyoshi et al., 2014], [Ljungkvist, Kronbichler, 2017, 2019]
- (Preconditioned) Krylov subspace methods: [Emans, van der Meer, 2012], [Yamagishi, Matsumura, 2016], [C., Gergelits, Yamazaki, 2021], [Clark, 2019], [Anzt et al., 2019], [Clark et al., 2010], [Gratton et al., 2020], [Arioli, Duff, 2009], [Hogg, Scott, 2010]

For survey and references, see [Abdelfattah et al., IJHPC, 2021], [Higham and Mary, 2022]

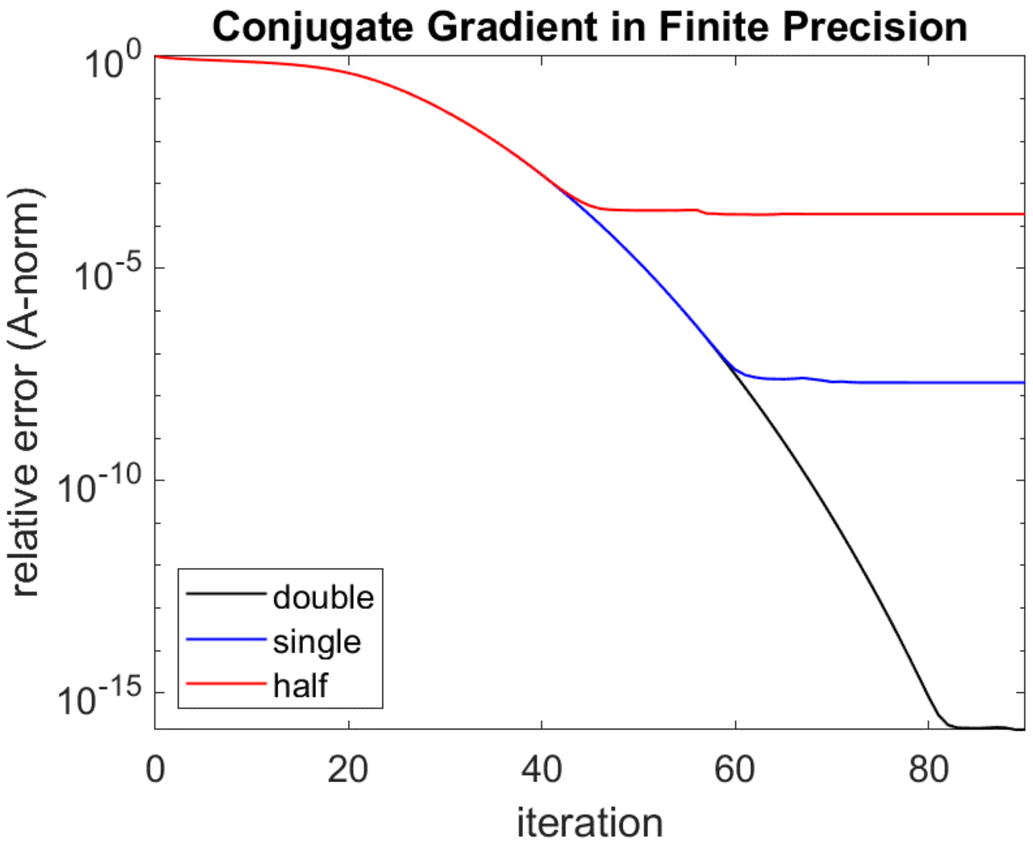
When Can I Use Low Precision?

1. When low accuracy is needed

When Can I Use Low Precision?

1. When low accuracy is needed

```
A = diag(linspace(.001,1,100));
b = ones(n,1);
```



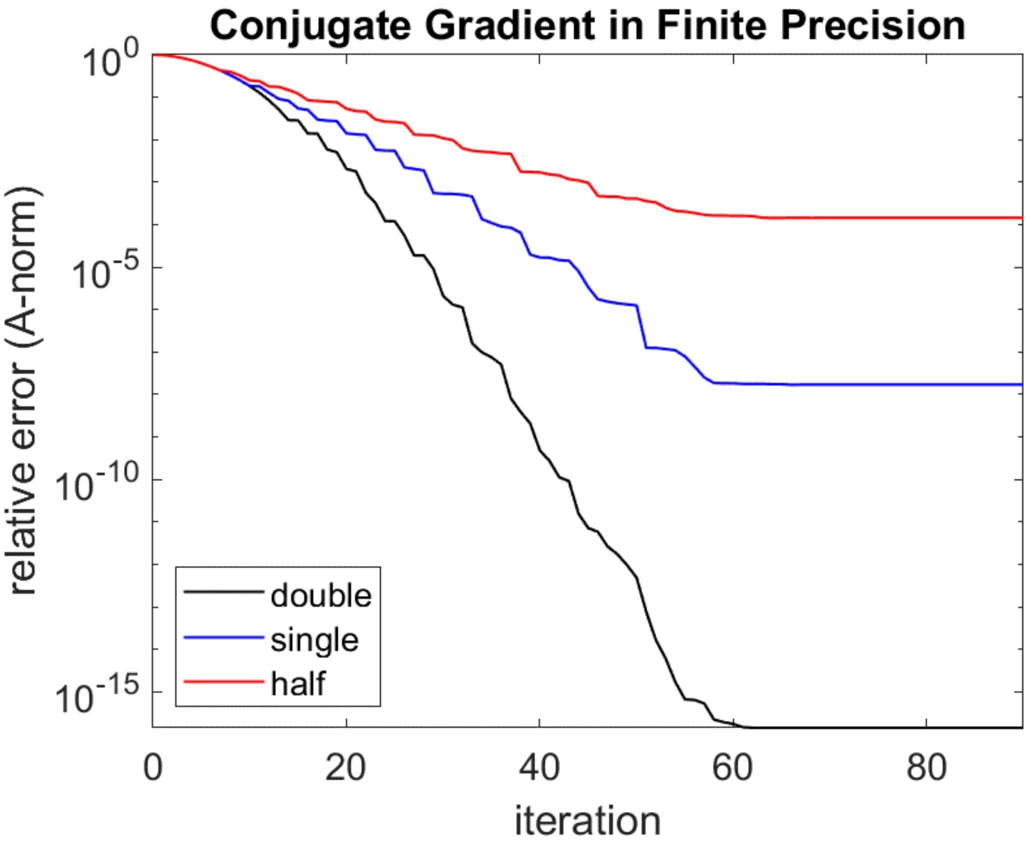
When Can I Use Low Precision?

1. When low accuracy is needed

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

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

```
b = ones(n,1);
```



When Can I Use Low Precision?

1. When low accuracy is needed
2. When a self-correction mechanism is available

When Can I Use Low Precision?

1. When low accuracy is needed
2. When a self-correction mechanism is available

Example: Iterative refinement

```
Solve $Ax_0 = b$ by LU factorization          (in precision $\boldsymbol{u_f}$)
for $i = 0\colon \mathrm{maxit}$
        $r_i = b - Ax_i$                        (in precision $\boldsymbol{u_r}$)
        Solve $Ad_i = r_i$                      (in precision $\boldsymbol{u_s}$)
        $x_{i+1} = x_i + d_i$                   (in precision $\boldsymbol{u}$)
```

e.g., [Langou et al., 2006], [Arioli and Duff, 2009], [Hogg and Scott, 2010], [Abdelfattah et al., 2016], [C. and Higham, 2018], [Amestoy et al., 2021]

When Can I Use Low Precision?

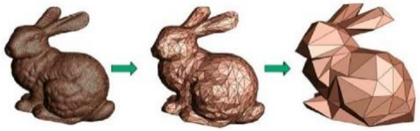
1. When low accuracy is needed
2. When a self-correction mechanism is available
3. When there are other significant sources of inexactness

When Can I Use Low Precision?

1. When low accuracy is needed
2. When a self-correction mechanism is available
3. When there are other significant sources of inexactness

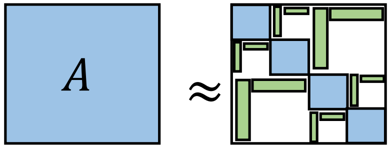
- E.g., reduced models, sparsification, low-rank approximations, randomization

Model Reduction

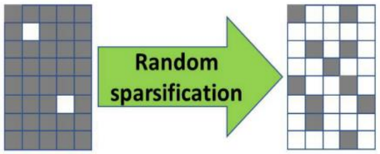


[Schilders, van der Vorst, Rommes, 2008]

Low-rank approximation



Sparsification, randomization



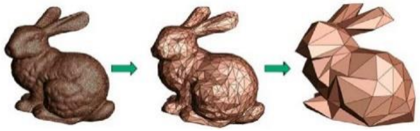
[Sinha, 2018]

When Can I Use Low Precision?

1. When low accuracy is needed
2. When a self-correction mechanism is available
3. When there are other significant sources of inexactness

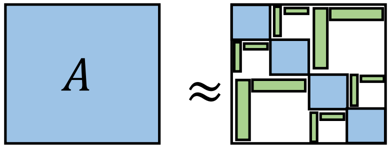
- E.g., reduced models, sparsification, low-rank approximations, randomization

Model Reduction

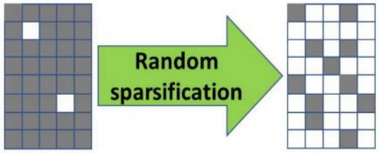


[Schilders, van der Vorst, Rommes, 2008]

Low-rank approximation



Sparsification, randomization



[Sinha, 2018]

Mixed Precision Sparse Approximate Inverse Preconditioners

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 $\boldsymbol{\varepsilon}$
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 $\|r_k\|_2 = \|e_k - Am_k\|_2 \leq \boldsymbol{\varepsilon}$
        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 $\varepsilon$
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 $\|r_k\|_2 = \|e_k - Am_k\|_2 \leq \varepsilon$
        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, Tůma, 1999], [He, Yin, Gao, 2020]

SPAI Preconditioners in Low Precision

What is the effect of using low precision in SPAI construction?

Notes and assumptions:

- We will assume that the SPAI construction is performed in some precision $\boldsymbol{u_f}$
- We will denote quantities computed in finite precision with hats
- In our application, we want a left preconditioner, so we will run the algorithm on A^T and get M^T.
- We will assume that the QR factorization of the submatrix of A^T is computed fully using HouseholderQR/TSQR

SPAI Preconditioners in Low Precision

Two interesting questions:

1. Assuming we impose no maximum sparsity pattern on $\widehat{M}$, under what constraint on $\boldsymbol{u_f}$ can we guarantee that $\|\hat{r}_k\|_2 \leq \boldsymbol{\varepsilon}$, with $\hat{r}_k = fl_{\boldsymbol{u_f}}(e_k - A^T \widehat{m}_k^T)$ for the computed $\widehat{m}_k^T$?

SPAI Preconditioners in Low Precision

Two interesting questions:

1. Assuming we impose no maximum sparsity pattern on $\widehat{M}$, under what constraint on $\boldsymbol{u_f}$ can we guarantee that $\|\hat{r}_k\|_2 \leq \boldsymbol{\varepsilon}$, with $\hat{r}_k = fl_{\boldsymbol{u_f}}(e_k - A^T \widehat{m}_k^T)$ for the computed $\widehat{m}_k^T$?

2. Assume that when M is computed in exact arithmetic, we quit as soon as $\|r_k\| \leq \boldsymbol{\varepsilon}$. For $\widehat{M}$ computed in precision $\boldsymbol{u_f}$ with the same sparsity pattern as M, what is $\left\|e_k - A^T \widehat{m}_k^T\right\|_2$?

SPAI Preconditioning in Low Precision

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

$$\|\hat{r}_k\|_2 \leq n^3 \boldsymbol{u_f} \big\| |e_k| + |A^T| |\widehat{m}_k^T| \big\|_2.$$

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

$$n^3 \boldsymbol{u_f} \big\| |e_k| + |A^T| |\widehat{m}_k^T| \big\|_2 \leq \boldsymbol{\varepsilon}.$$

SPAI Preconditioning in Low Precision

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

$$\|\hat{r}_k\|_2 \leq n^3 \boldsymbol{u_f} \Big\| |e_k| + |A^T| |\widehat{m}_k^T| \Big\|_2.$$

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

$$n^3 \boldsymbol{u_f} \Big\| |e_k| + |A^T| |\widehat{m}_k^T| \Big\|_2 \leq \boldsymbol{\varepsilon}.$$

→ problem must not be so ill-conditioned WRT $\boldsymbol{u_f}$ that we incur an error greater than $\boldsymbol{\varepsilon}$ just computing the residual

SPAI Preconditioning in Low Precision

Can turn this into the looser but more descriptive a priori bound:

$$\mathrm{cond}_2(A^T) \lesssim \boldsymbol{\varepsilon} \boldsymbol{u_f}^{-1},$$

where $\mathrm{cond}_2(A^T) = \| |A^{-T}||A^T| \|_2$.

SPAI Preconditioning in Low Precision

Can turn this into the looser but more descriptive a priori bound:

$$\mathrm{cond}_2(A^T) \lesssim \boldsymbol{\varepsilon} \boldsymbol{u}_{\boldsymbol{f}}^{-1},$$

where $\mathrm{cond}_2(A^T) = \||A^{-T}||A^T|\|_2$.

Another view: with a given matrix A and a given precision $\boldsymbol{u}_{\boldsymbol{f}}$, one must set $\boldsymbol{\varepsilon}$ such that

$$\boldsymbol{\varepsilon} \geq \boldsymbol{u}_{\boldsymbol{f}} \mathrm{cond}_2(A^T).$$

Confirms intuition: **The more approximate the inverse, the lower the precision we can use without noticing it**.

SPAI Preconditioning in Low Precision

Can turn this into the looser but more descriptive a priori bound:

$$\mathrm{cond}_2(A^T) \lesssim \boldsymbol{\varepsilon} \boldsymbol{u}_{\boldsymbol{f}}^{-1},$$

where $\mathrm{cond}_2(A^T) = \| |A^{-T}| |A^T| \|_2$.

Another view: with a given matrix A and a given precision $\boldsymbol{u}_{\boldsymbol{f}}$, one must set $\boldsymbol{\varepsilon}$ such that

$$\boldsymbol{\varepsilon} \geq \boldsymbol{u}_{\boldsymbol{f}} \mathrm{cond}_2(A^T).$$

Confirms intuition: **The more approximate the inverse, the lower the precision we can use without noticing it**.

Resulting bounds for $\widehat{M}$:

$$\left\| I - \widehat{M}A \right\|_F \leq 2\sqrt{n}\boldsymbol{\varepsilon}, \qquad \left\| I - \widehat{M}A \right\|_\infty \leq 2n\boldsymbol{\varepsilon}$$

Second Question

Assume that when M *is computed in exact arithmetic, we quit as soon as* $\|r_k\| \le \boldsymbol{\varepsilon}$*. For* $\widehat{M}$ *computed in precision* $\boldsymbol{u_f}$ *with the same sparsity pattern as* M*, what is* $\left\|e_k - A^T \widehat{m}_k^T\right\|_2$*?*

Second Question

Assume that when M is computed in exact arithmetic, we quit as soon as $\|r_k\| \le \boldsymbol{\varepsilon}$. For $\widehat{M}$ computed in precision $\boldsymbol{u_f}$ with the same sparsity pattern as M, what is $\left\|e_k - A^T \widehat{m}_k^T\right\|_2$?

In this case, we obtain the bound

$$\boxed{\left\|I - \widehat{M}A\right\|_\infty \le n\left(\boldsymbol{\varepsilon} + n^{7/2}\boldsymbol{u_f}\kappa_\infty(A)\right).}$$

→ If $\kappa_\infty(A) \gg \boldsymbol{\varepsilon}\boldsymbol{u_f}^{-1}$, then computed $\widehat{M}$ with same sparsity structure as M can be of much lower quality.

Krylov-Based Iterative Refinement

```
Solve $Ax_0 = b$ by LU factorization          (in precision $\boldsymbol{u_f}$)
for $i = 0\colon \text{maxit}$
        $r_i = b - Ax_i$                      (in precision $\boldsymbol{u_r}$)
        Solve $Ad_i = r_i$                    (in precision $\boldsymbol{u_s}$)
        $x_{i+1} = x_i + d_i$                 (in precision $\boldsymbol{u}$)
```

Krylov-Based Iterative Refinement

GMRES-IR [C. and Higham, SISC 39(6), 2017]

To compute the updates d_i, apply GMRES to $\overbrace{\hat{U}^{-1}\hat{L}^{-1}A}^{\tilde{A}}d_i = \overbrace{\hat{U}^{-1}\hat{L}^{-1}r_i}^{\tilde{r}_i}$

```
Solve $Ax_0 = b$ by LU factorization                                  (in precision $\boldsymbol{u_f}$)
for $i = 0$: maxit
        $r_i = b - Ax_i$                                              (in precision $\boldsymbol{u_r}$)
        Solve $Ad_i = r_i$ via GMRES on $\tilde{A}d_i = \tilde{r}_i$  (in precision $\boldsymbol{u_s}$)
        $x_{i+1} = x_i + d_i$                                         (in precision $\boldsymbol{u}$)
```

For related work, see references in [Higham, Mary, 2022], [Vieuble, 2022]

GMRES-IR with Inexact Preconditioners

- Most existing analyses of GMRES-IR assume we use full LU factors
- In practice, often want to use approximate preconditioners (ILU, SPAI, etc.)
- [Amestoy et al., 2022]
  - Analysis of **block low-rank (BLR) LU** within GMRES-IR
  - Analysis of use of **static pivoting** in LU within GMRES-IR
- [C., Khan, 2023]
  - Analysis of **sparse approximate inverse (SPAI) preconditioners** within GMRES-IR

SPAI-GMRES-IR

SPAI-GMRES-IR [C. and Khan, SISC 45(3), 2023]

To compute the updates d_i, apply GMRES to $\overbrace{\widehat{M}A}^{\tilde{A}}d_i = \overbrace{\widehat{M}r_i}^{\tilde{r}_i}$

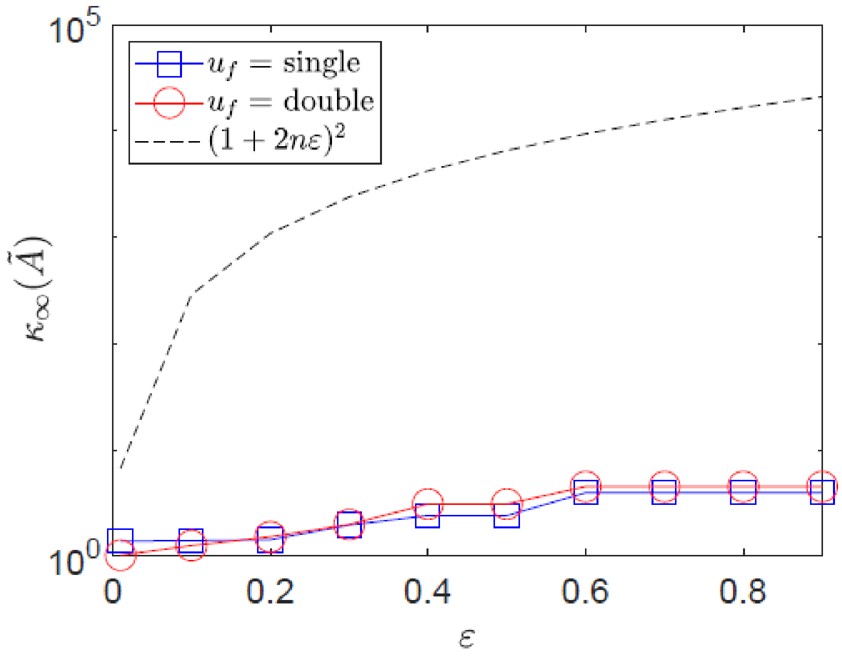
```
Compute SPAI $\widehat{M}$; solve $\widehat{M}Ax_0 = \widehat{M}b$          (in precision $u_f$)
for $i = 0$: maxit
    $r_i = b - Ax_i$                                                        (in precision $u_r$)
    Solve $Ad_i = r_i$ via GMRES on $\widehat{M}Ad_i = \widehat{M}r_i$      (in precision $u_s$)
    $x_{i+1} = x_i + d_i$                                                   (in precision $u$)
```

Low Precision SPAI within GMRES-IR

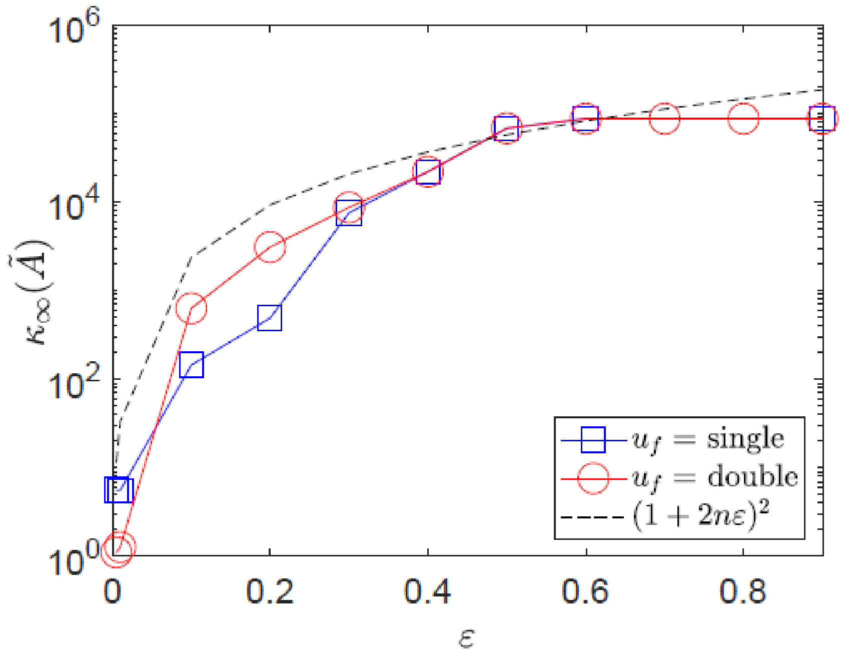
Using $\widehat{M}$ computed in precision $\boldsymbol{u_f}$, for the preconditioned system $\tilde{A} = \widehat{M}A$,

$$\kappa_\infty(\tilde{A}) \lesssim (1 + 2n\boldsymbol{\varepsilon})^2.$$

steam3



saylr1



Low Precision SPAI within GMRES-IR

To guarantee that both SPAI construction will complete and the GMRES-based iterative refinement scheme will converge, we must have roughly

$$n\boldsymbol{u_f}\,\mathrm{cond}_2(A^T) \lesssim n\boldsymbol{\varepsilon} \lesssim \boldsymbol{u}^{-1/2}.$$

Low Precision SPAI within GMRES-IR

To guarantee that both SPAI construction will complete and the GMRES-based iterative refinement scheme will converge, we must have roughly

$$\underbrace{n\boldsymbol{u_f}\,\mathrm{cond}_2(A^T) \lesssim n\boldsymbol{\varepsilon}}_{\widehat{M}\text{ can be constructed}} \lesssim \boldsymbol{u}^{-1/2}.$$

Low Precision SPAI within GMRES-IR

To guarantee that both SPAI construction will complete and the GMRES-based iterative refinement scheme will converge, we must have roughly

$$\underbrace{n \boldsymbol{u_f} \mathrm{cond}_2(A^T) \lesssim n \boldsymbol{\varepsilon}}_{\widehat{M}\text{ can be constructed}} \underbrace{\lesssim \boldsymbol{u}^{-1/2}}_{\widehat{M}\text{ is a good enough preconditioner}}.$$

Low Precision SPAI within GMRES-IR

To guarantee that both SPAI construction will complete and the GMRES-based iterative refinement scheme will converge, we must have roughly

$$\underbrace{n\boldsymbol{u}_{\boldsymbol{f}}\,\mathrm{cond}_2(A^T) \lesssim n\boldsymbol{\varepsilon}}_{\widehat{M}\text{ can be constructed}}\underbrace{\lesssim \boldsymbol{u}^{-1/2}}_{\widehat{M}\text{ is a good enough preconditioner}}.$$

If $\boldsymbol{\varepsilon}$ satisfies these constraints, then the constraints on condition number for forward and backward errors to converge are the same as for GMRES-IR with full LU factorization.

Low Precision SPAI within GMRES-IR

To guarantee that both SPAI construction will complete and the GMRES-based iterative refinement scheme will converge, we must have roughly

$$n\boldsymbol{u_f}\,\mathrm{cond}_2(A^T) \lesssim n\boldsymbol{\varepsilon} \lesssim \boldsymbol{u}^{-1/2}.$$

$\widehat{M}$ can be constructed

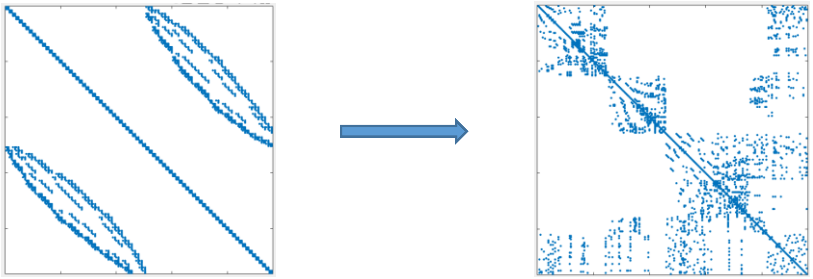
$\widehat{M}$ is a good enough preconditioner

If $\boldsymbol{\varepsilon}$ satisfies these constraints, then the constraints on condition number for forward and backward errors to converge are the same as for GMRES-IR with full LU factorization.

Compared to GMRES-IR with full LU factorization, in general expect **slower convergence, but much sparser preconditioner**.

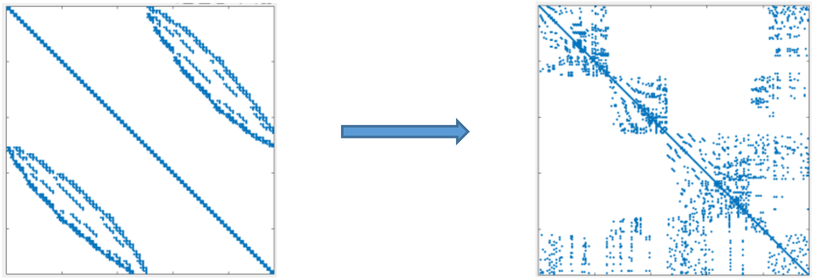
SPAI-GMRES-IR Example

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

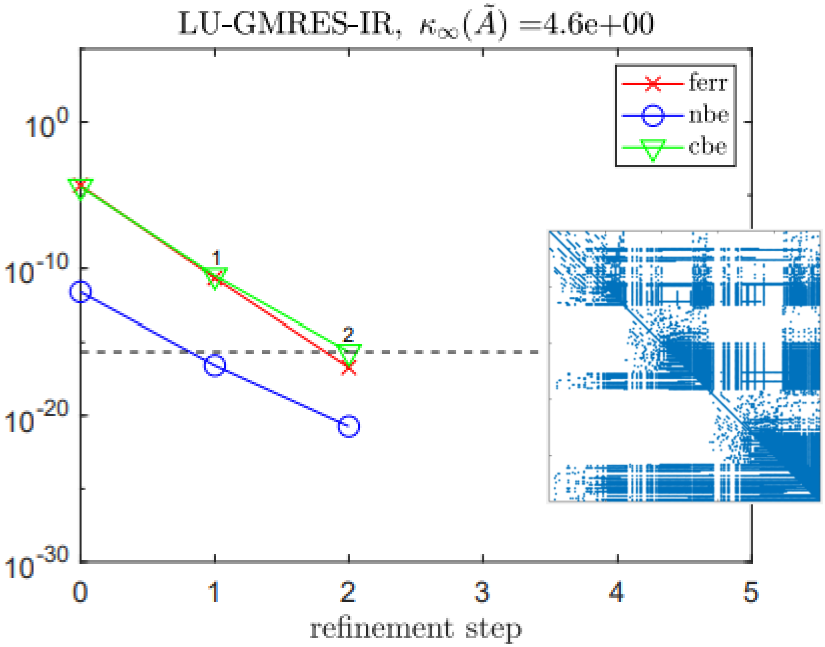


SPAI-GMRES-IR Example

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



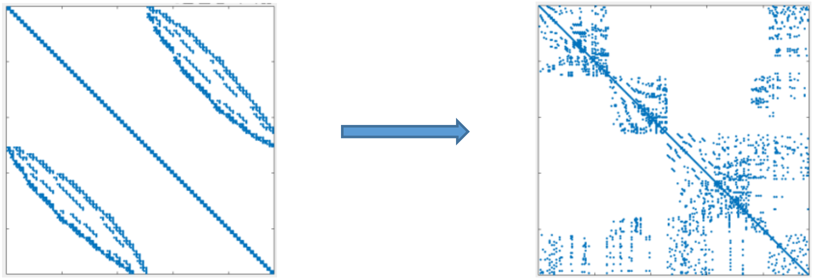
$(\boldsymbol{u_f}, \boldsymbol{u}, \boldsymbol{u_r})$ = (single, double, quad)



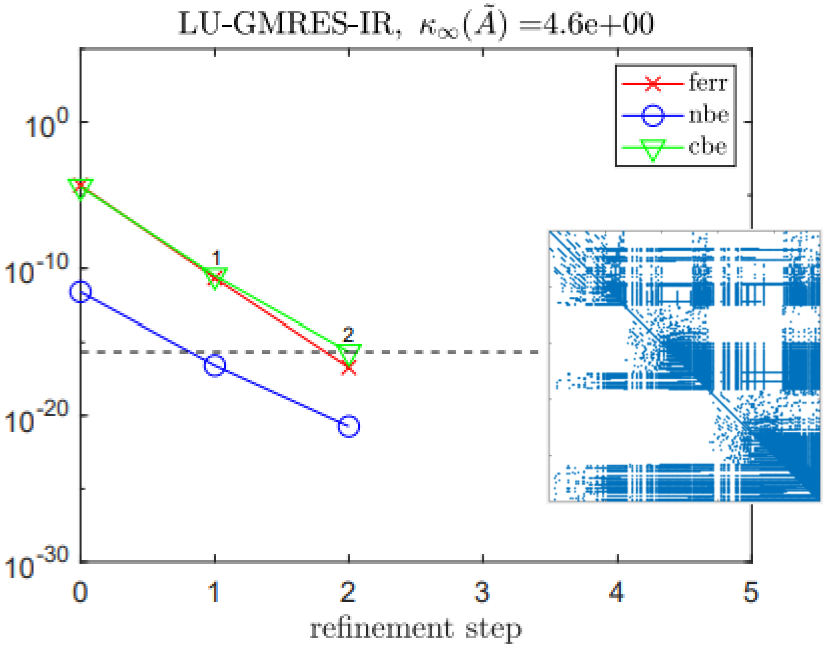
nnz$(L + U) = 13{,}765$

SPAI-GMRES-IR Example

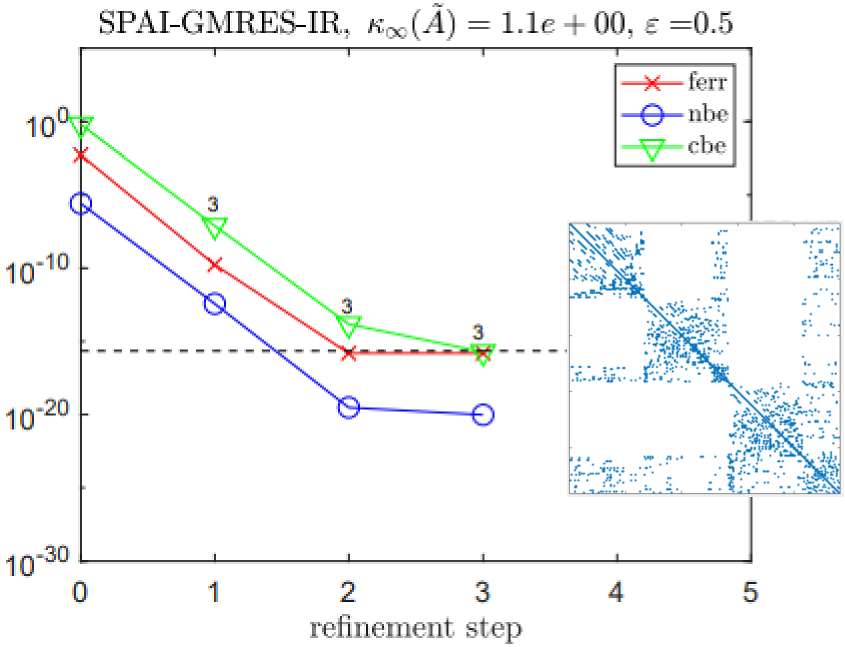
Matrix: steam1, $n = 240$, nnz $= 2{,}248$, $\kappa_\infty(A) = 3 \cdot 10^7$, cond$(A^T) = 3 \cdot 10^3$



$(\boldsymbol{u_f}, \boldsymbol{u}, \boldsymbol{u_r})$ = (single, double, quad)



nnz$(L+U) = 13{,}765$



nnz$(M) = 2{,}248$

Ongoing and Future Work

- Incorporate mixed-precision storage of $\widehat{M}$ and **adaptive-precision SpMV** to apply $\widehat{M}$ using the work of [Graillat et al., 2022]

- Theoretical analysis of **incomplete factorization preconditioners** in mixed precision (with J. Scott and M. Tůma)
  - Experimental work shows that half precision works well in practice [Scott, Tůma, 2023]

Randomized Preconditioners for GMRES-Based Least Squares Iterative Refinement

Least Squares Problems

- Want to solve

$$\min_x \|b - Ax\|_2$$

where $A \in \mathbb{R}^{m \times n}$ $(m > n)$ has rank n

- Commonly solved using QR factorization:

$$A = QR = [Q_1, Q_2] \begin{bmatrix} U \\ 0 \end{bmatrix}$$

where Q is an $m \times m$ orthogonal matrix and U is upper triangular.

$$x = U^{-1} Q_1^T b, \qquad \|b - Ax\|_2 = \left\|Q_2^T b\right\|_2$$

- As in linear system case, for ill-conditioned problems, iterative refinement often needed to improve accuracy and stability

Least Squares Iterative Refinement

- For inconsistent systems, must simultaneously refine both solution and residual
- (Björck, 1967): Least squares problem can be written as a linear system with square matrix of size $(m+n)$:

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

Least Squares Iterative Refinement

- For inconsistent systems, must simultaneously refine both solution and residual
- (Björck, 1967): Least squares problem can be written as a linear system with square matrix of size $(m+n)$:

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

- Refinement proceeds as follows:

1. Compute "residuals"

$$\begin{bmatrix} f_i \\ g_i \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r_i \\ x_i \end{bmatrix} = \begin{bmatrix} b - r_i - Ax_i \\ -A^T r_i \end{bmatrix}$$

2. Solve for corrections

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \Delta r_i \\ \Delta x_i \end{bmatrix} = \begin{bmatrix} f_i \\ g_i \end{bmatrix}$$

3. Update "solution":

$$\begin{bmatrix} r_{i+1} \\ x_{i+1} \end{bmatrix} = \begin{bmatrix} r_i \\ x_i \end{bmatrix} + \begin{bmatrix} \Delta r_i \\ \Delta x_i \end{bmatrix}$$

GMRES-LSIR

- For inconsistent systems, must simultaneously refine both solution and residual
- (Björck, 1967): Least squares problem can be written as a linear system with square matrix of size $(m+n)$:

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

[C., Higham, Pranesh, 2020]: Compute QR factorization in $\boldsymbol{u_f}$, use as preconditioner for GMRES

- Refinement proceeds as follows:

1. Compute "residuals"

$$\begin{bmatrix} f_i \\ g_i \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r_i \\ x_i \end{bmatrix} = \begin{bmatrix} b - r_i - Ax_i \\ -A^T r_i \end{bmatrix}$$

(in precision $\boldsymbol{u_r}$)

2. Solve for corrections

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \Delta r_i \\ \Delta x_i \end{bmatrix} = \begin{bmatrix} f_i \\ g_i \end{bmatrix}$$

via preconditioned GMRES (in precision $\boldsymbol{u_s}$)

3. Update "solution":

$$\begin{bmatrix} r_{i+1} \\ x_{i+1} \end{bmatrix} = \begin{bmatrix} r_i \\ x_i \end{bmatrix} + \begin{bmatrix} \Delta r_i \\ \Delta x_i \end{bmatrix}$$

(in precision $\boldsymbol{u}$)

GMRES-LSIR Analysis

- Using the preconditioner

$$M = \begin{bmatrix} \alpha I & \hat{Q}_1 \hat{R} \\ \hat{R}^T \hat{Q}_1^T & 0 \end{bmatrix}$$

we can prove that for the left-preconditioned system,

$$\kappa\left(M^{-1}\tilde{A}\right) \leq \left(1 + \boldsymbol{u_f} c\, \kappa(A)\right)^2$$

where $c = O(m^2)$.

[C., Higham, Pranesh, SISC 2020]

GMRES-LSIR Analysis

- Using the preconditioner

$$M = \begin{bmatrix} \alpha I & \hat{Q}_1 \hat{R} \\ \hat{R}^T \hat{Q}_1^T & 0 \end{bmatrix}$$

we can prove that for the left-preconditioned system,

$$\kappa\left(M^{-1}\tilde{A}\right) \leq \left(1 + \boldsymbol{u_f} c\, \kappa(A)\right)^2$$

where $c = O(m^2)$.

- So for GMRES-based LSIR, expect convergence of forward error when $\kappa_\infty(A) < \boldsymbol{u}^{-1/2}\boldsymbol{u_f}^{-1}$.

[C., Higham, Pranesh, SISC 2020]

GMRES-LSIR Analysis

- Using the preconditioner

$$M = \begin{bmatrix} \alpha I & \hat{Q}_1 \hat{R} \\ \hat{R}^T \hat{Q}_1^T & 0 \end{bmatrix}$$

Can we use other preconditioners?

we can prove that for the left-preconditioned system,

$$\kappa\left(M^{-1}\tilde{A}\right) \leq \left(1 + \boldsymbol{u_f} c\, \kappa(A)\right)^2$$

where $c = O(m^2)$.

- So for GMRES-based LSIR, expect convergence of forward error when $\kappa_\infty(A) < \boldsymbol{u}^{-1/2}\boldsymbol{u_f}^{-1}$.

[C., Higham, Pranesh, SISC 2020]

Randomized Preconditioning for LS

“Sketch-and-precondition” [Rokhlin, Tygert, 2008]:

1. Randomly sketch A

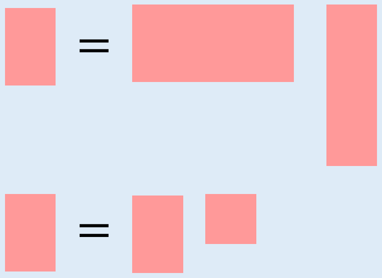
$$S = \Omega A, \text{ where } \Omega \in \mathbb{R}^{s\times m},\ s \geq n$$

2. Compute economic QR

$$S = QR$$

3. Solve via LSQR preconditioned with R

$$\min_y \|b - AR^{-1}y\|_2\,, \quad \text{where } y = Rx$$



[Avron, Maymounkov, Toledo, 2010]: Efficient implementation (Blendenpik) in one precision

Randomized Preconditioning for LS

$u = u_{QR} \leq u_s$

"Sketch-and-precondition" [Rokhlin, Tygert, 2008]:

1. Randomly sketch A

   $S = \Omega A$, where $\Omega \in \mathbb{R}^{s \times m}$, $s \geq n$ (in precision u_s)

2. Compute economic QR

   $S = QR$ (in precision u_{QR})

3. Solve via LSQR preconditioned with R

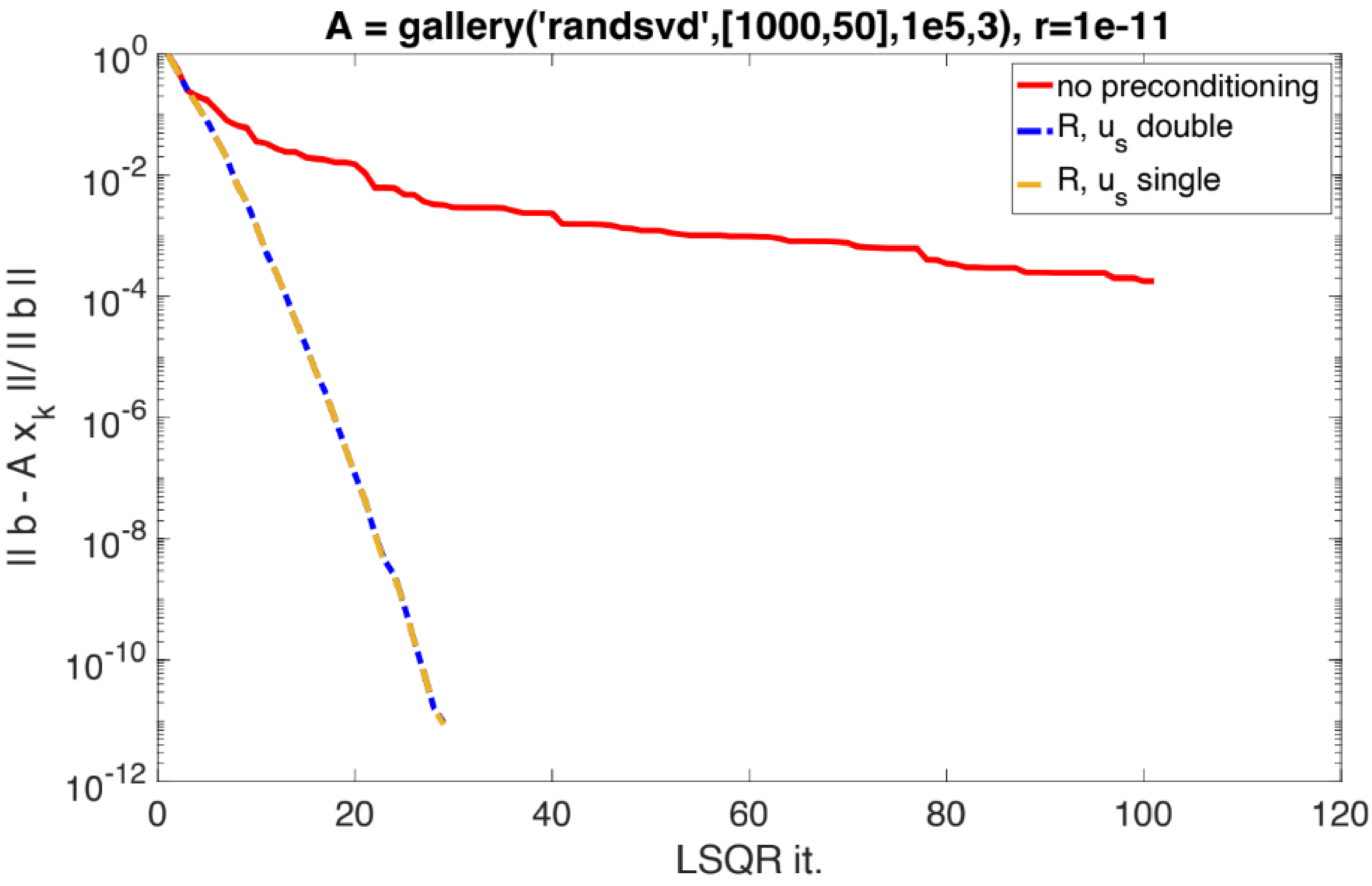
   $\min_y \|b - AR^{-1}y\|_2$, where $y = Rx$ (in precision u)

[Avron, Maymounkov, Toledo, 2010]: Efficient implementation (Blendenpik) in one precision

[Georgiou, Boutsikas, Drineas, Anzt, 2023]: Experimental results that show R can be computed in mixed precision

Example

$$u = u_{QR} = \text{double}$$



Randomized Preconditioning

"Sketch-and-apply" [Meier, Nakatsukasa, Townsend, Webb, 2023]

1. Compute R as in [Rokhlin, Tygert, 2008]
2. Explicitly form preconditioned matrix

   $$Y = AR^{-1}$$

3. Solve via (unpreconditioned) LSQR

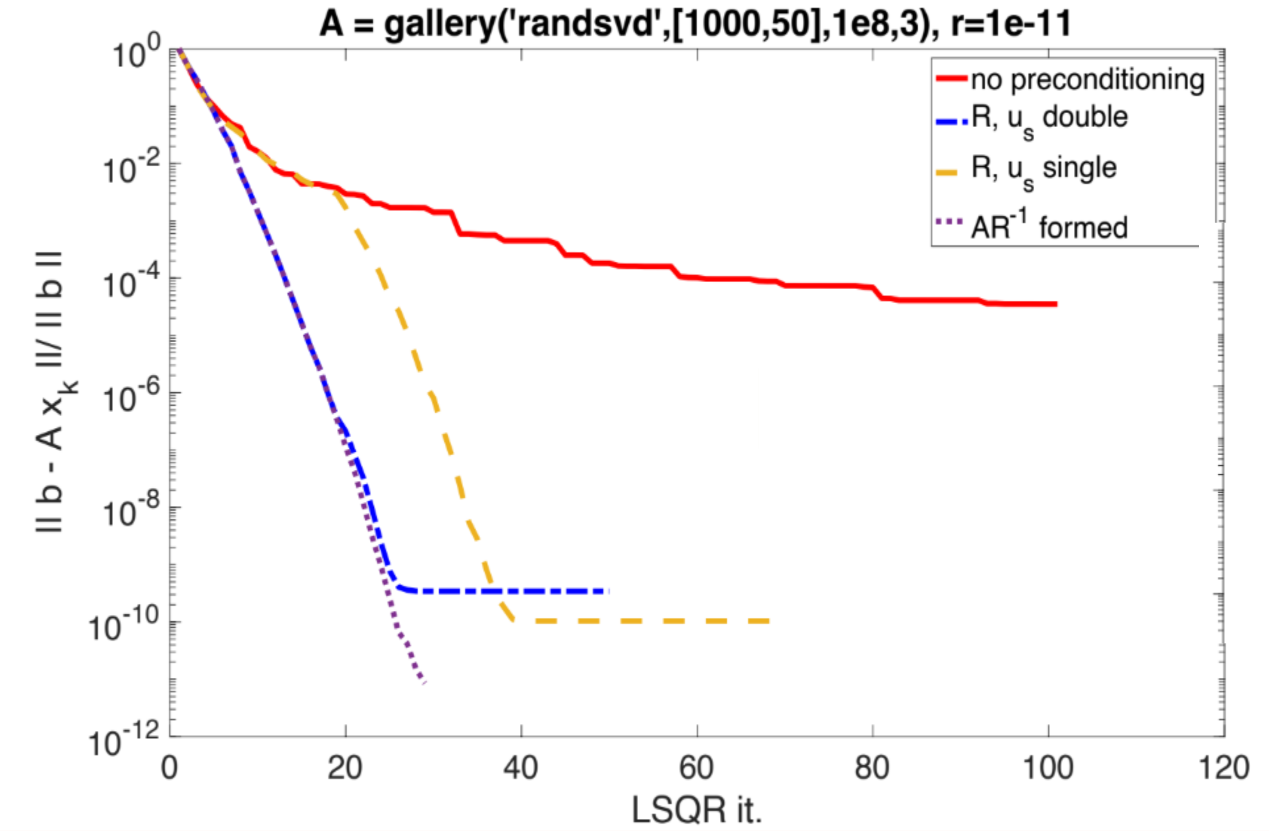
   $$\min_z \|b - Yz\|_2$$

4. Recover x

   $$Rx = z$$

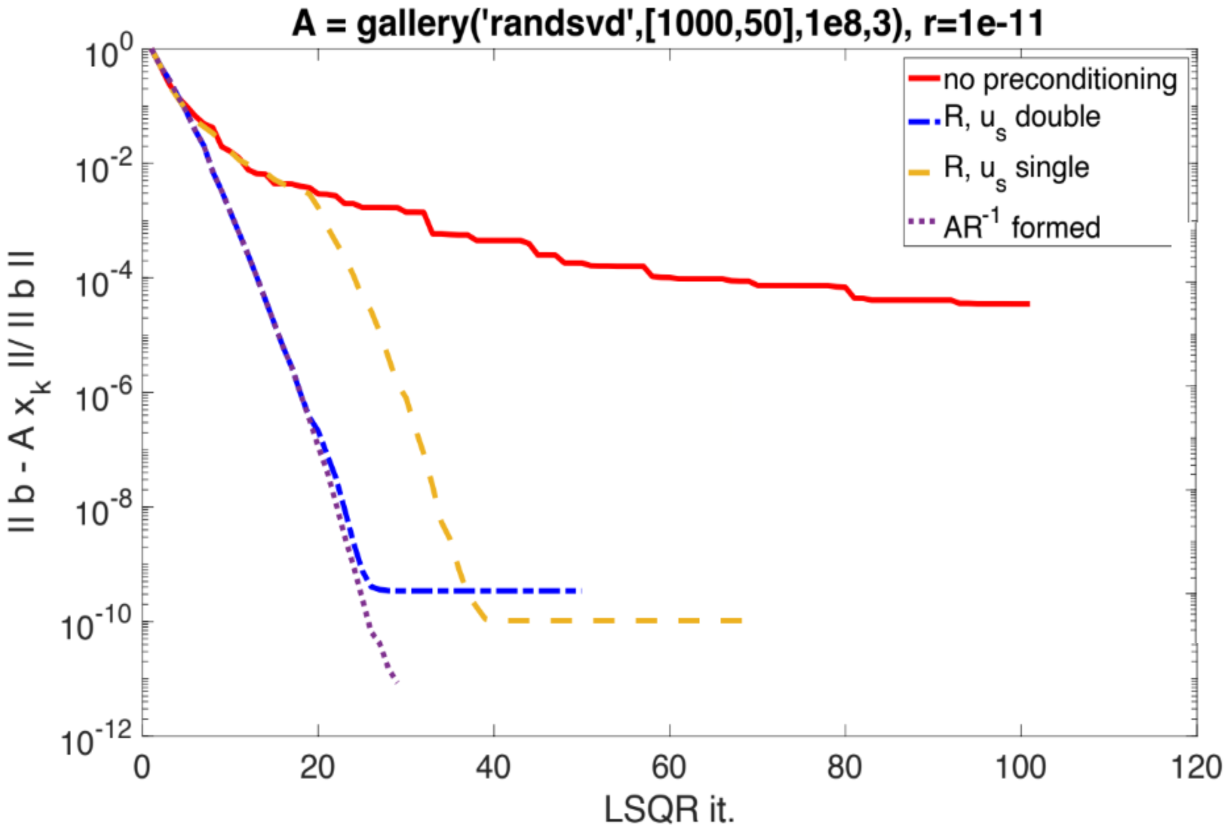
Example

$u = u_{QR} =$ double



Example

$u = u_{QR} =$ double



Relative forward error:

R, u_s double: 4×10^{-8}
R, u_s double: 3×10^{-8}
Formed AR^{-1}: 2×10^{-8}

“Sketch-and-Precondition” GMRES-LSIR

Compute $\hat{R}$ factor of QR decomposition of randomly sketched A using precision $\boldsymbol{u_s}$ (sketching step) and $\boldsymbol{u_{QR}}$ (QR step).

“Sketch-and-Precondition” GMRES-LSIR

Compute $\hat{R}$ factor of QR decomposition of randomly sketched A using precision $\boldsymbol{u_s}$ (sketching step) and $\boldsymbol{u_{QR}}$ (QR step).

Solve $\min_x \|b - Ax\|_2$via LSQR preconditioned with $\hat{R}$ in precision $\boldsymbol{u}$ to get initial solution x_0 and residual r_0.

“Sketch-and-Precondition” GMRES-LSIR

Compute $\hat{R}$ factor of QR decomposition of randomly sketched A using precision $\boldsymbol{u_s}$ (sketching step) and $\boldsymbol{u_{QR}}$ (QR step).

Solve $\min_x \|b - Ax\|_2$via LSQR preconditioned with $\hat{R}$ in precision $\boldsymbol{u}$ to get initial solution x_0 and residual r_0.

for $i = 0, \ldots,$ until convergence

Compute residual $\begin{bmatrix} f_i \\ g_i \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r_i \\ x_i \end{bmatrix}$ and $h_i = \hat{R}^{-T} g_i$ in precision $\boldsymbol{u_r}$.

Solve via FGMRES in (effective) precision $\boldsymbol{u_s}$:

$$\begin{bmatrix} I & 0 \\ 0 & \hat{R}^{-T} \end{bmatrix} \begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & \hat{R}^{-1} \end{bmatrix} \begin{bmatrix} \delta r_i \\ \delta z_i \end{bmatrix} = \begin{bmatrix} f_i \\ h_i \end{bmatrix},$$

where $\hat{R} \delta x_i = \delta z_i$.

Update in precision $\boldsymbol{u}$:

$$\begin{bmatrix} r_{i+1} \\ x_{i+1} \end{bmatrix} = \begin{bmatrix} r_i \\ x_i \end{bmatrix} + \begin{bmatrix} \delta r_i \\ \delta x_i \end{bmatrix}$$

“Sketch-and-Precondition” GMRES-LSIR

Compute $\hat{R}$ factor of QR decomposition of randomly sketched A using precision $\boldsymbol{u_s}$ (sketching step) and $\boldsymbol{u_{QR}}$ (QR step).

Solve $\min\limits_x \|b - Ax\|_2$ via LSQR preconditioned with $\hat{R}$ in precision $\boldsymbol{u}$ to get initial solution x_0 and residual r_0.

for $i = 0, \dots,$ until convergence

Compute residual $\begin{bmatrix} f_i \\ g_i \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r_i \\ x_i \end{bmatrix}$ and $h_i = \hat{R}^{-T} g_i$ in precision $\boldsymbol{u_r}$.

Solve via FGMRES in (effective) precision $\boldsymbol{u_s}$:

$$\begin{bmatrix} I & 0 \\ 0 & \hat{R}^{-T} \end{bmatrix} \begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & \hat{R}^{-1} \end{bmatrix} \begin{bmatrix} \delta r_i \\ \delta z_i \end{bmatrix} = \begin{bmatrix} f_i \\ h_i \end{bmatrix},$$

where $\hat{R}\delta x_i = \delta z_i$.

Update in precision $\boldsymbol{u}$:

$$\begin{bmatrix} r_{i+1} \\ x_{i+1} \end{bmatrix} = \begin{bmatrix} r_i \\ x_i \end{bmatrix} + \begin{bmatrix} \delta r_i \\ \delta x_i \end{bmatrix}$$

[C., Daužickaitė, 2024]: Analysis of four-precision split-preconditioned FGMRES

“Sketch-and-Precondition” GMRES-LSIR

Theoretical analysis suggests how to choose precisions:

- For generating preconditioner, $\boldsymbol{u_s} \approx \boldsymbol{u_{QR}}$ (although $\boldsymbol{u_{QR}} < \boldsymbol{u_s}$ is inexpensive and may help avoid overflow)

- For FGMRES, apply left preconditioner and matrix to a vector in precision $\leq \boldsymbol{u}$ (can be less careful with right preconditioner)

“Sketch-and-Apply” GMRES-LSIR

Compute $\hat{R}$ factor of QR decomposition of randomly sketched A using precision $\boldsymbol{u_s}$ (sketching step) and $\boldsymbol{u_{QR}}$ (QR step).

“Sketch-and-Apply” GMRES-LSIR

Compute $\hat{R}$ factor of QR decomposition of randomly sketched A using precision $\boldsymbol{u_s}$ (sketching step) and $\boldsymbol{u_{QR}}$ (QR step).

Form $Y = A\hat{R}^{-1}$ in precision $\boldsymbol{u_Y}$.

“Sketch-and-Apply” GMRES-LSIR

Compute $\hat{R}$ factor of QR decomposition of randomly sketched A using precision $\boldsymbol{u_s}$ (sketching step) and $\boldsymbol{u_{QR}}$ (QR step).

Form $Y = A\hat{R}^{-1}$ in precision $\boldsymbol{u_Y}$.

Solve $\min\limits_{z} \|b - Yz\|_2$ via LSQR in precision $\boldsymbol{u}$ and solve $Rx = z$ in precision $\boldsymbol{u_x}$ to get initial solution x_0 and residual r_0.

“Sketch-and-Apply” GMRES-LSIR

```
Compute $\hat{R}$ factor of $QR$ decomposition of randomly sketched $A$ using precision $\boldsymbol{u_s}$ (sketching step) and $\boldsymbol{u_{QR}}$ (QR step).

Form $Y = A\hat{R}^{-1}$ in precision $\boldsymbol{u_Y}$.

Solve $\min_z \|b - Yz\|_2$ via LSQR in precision $\boldsymbol{u}$ and solve $Rx = z$ in precision $\boldsymbol{u_x}$ to get initial solution $x_0$ and residual $r_0$.

for $i = 0, \ldots,$ until convergence
    Compute residual $\begin{bmatrix} f_i \\ g_i \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r_i \\ x_i \end{bmatrix}$ and $h_i = \hat{R}^{-T} g_i$ in precision $\boldsymbol{u_r}$.
    Solve via unpreconditioned GMRES in precision $\boldsymbol{u}$:
        $\begin{bmatrix} I & Y \\ Y^T & 0 \end{bmatrix} \begin{bmatrix} \delta r_i \\ \delta z_i \end{bmatrix} = \begin{bmatrix} f_i \\ h_i \end{bmatrix}$
    Solve $\hat{R}\delta x_i = \delta z_i$ in precision $\boldsymbol{u_x}$.
    Update in precision $\boldsymbol{u}$:
        $\begin{bmatrix} r_{i+1} \\ x_{i+1} \end{bmatrix} = \begin{bmatrix} r_i \\ x_i \end{bmatrix} + \begin{bmatrix} \delta r_i \\ \delta x_i \end{bmatrix}$
```

“Sketch-and-Apply” GMRES-LSIR

Theoretical analysis suggests how to choose precisions:

- For generating preconditioner, $\boldsymbol{u_s} \approx \boldsymbol{u_{QR}}$ (although $\boldsymbol{u_{QR}} < \boldsymbol{u_s}$ is inexpensive and may help avoid overflow)

- Triangular solves: Want $\boldsymbol{u_x}\kappa(A) < 1$
- GMRES: Want $\boldsymbol{u}\kappa(A)\kappa(Y) < 1$
- Forming Y: Want $\boldsymbol{u_Y}\kappa(A)^2\kappa(Y) < 1$

Ongoing work: Collaboration on high-performance implementation with V. Georgiou and H. Anzt

Mixed Precision Randomized Nyström Approximation

Randomized Nyström Approximation

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

Nyström approximation:

$$\boxed{A_N = (A\Omega)(\Omega^T A\Omega)^\dagger (A\Omega)^T}$$

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

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

Randomized Nyström Approximation

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

Nyström approximation:

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

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

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

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

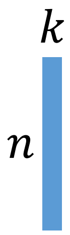
→ Can use single-pass version of the Nyström method [Tropp et al., 2017].

Single-Pass Nyström Approximation

Given sym. PSD matrix A, target rank k

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

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



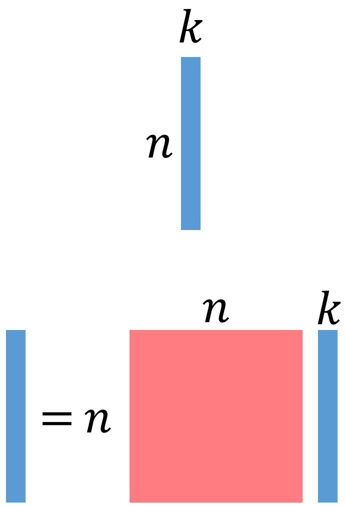
Single-Pass Nyström Approximation

Given sym. PSD matrix A, target rank k

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

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

$\boldsymbol{Y} = \boldsymbol{AQ}$



Single-Pass Nyström Approximation

Given sym. PSD matrix A, target rank k

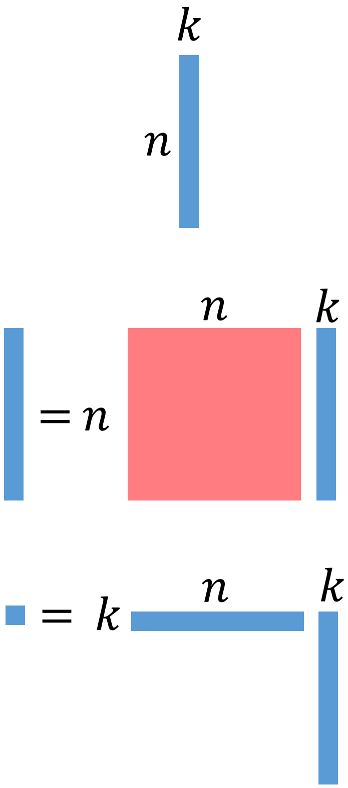
$G = \text{randn}(n, k)$

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

$\boldsymbol{Y = AQ}$

Compute shift ν; $Y_\nu = Y + \nu Q$

$B = Q^T Y_\nu$



Single-Pass Nyström Approximation

Given sym. PSD matrix A, target rank k

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

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

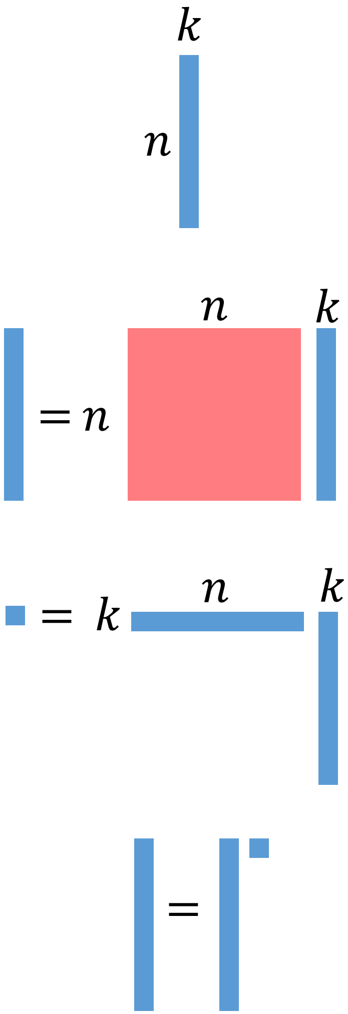
$\boldsymbol{Y = AQ}$

Compute shift ν; $Y_\nu = Y + \nu Q$

$B = Q^T Y_\nu$

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

Solve $F = Y_\nu / C$



Single-Pass Nyström Approximation

Given sym. PSD matrix A, target rank k

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

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

$\boldsymbol{Y = AQ}$

Compute shift ν; $Y_\nu = Y + \nu Q$

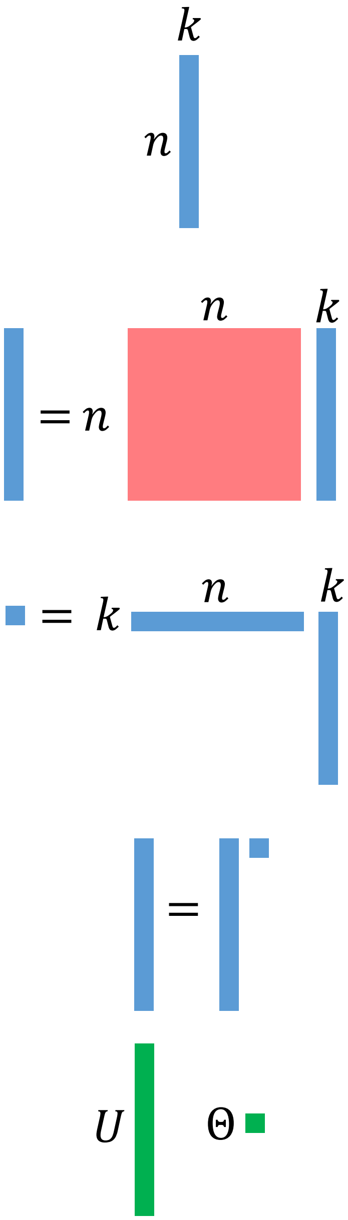
$B = Q^T Y_\nu$

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

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

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

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

$\boldsymbol{Y = AQ}$

Compute shift ν; $Y_\nu = Y + \nu Q$

$B = Q^T Y_\nu$

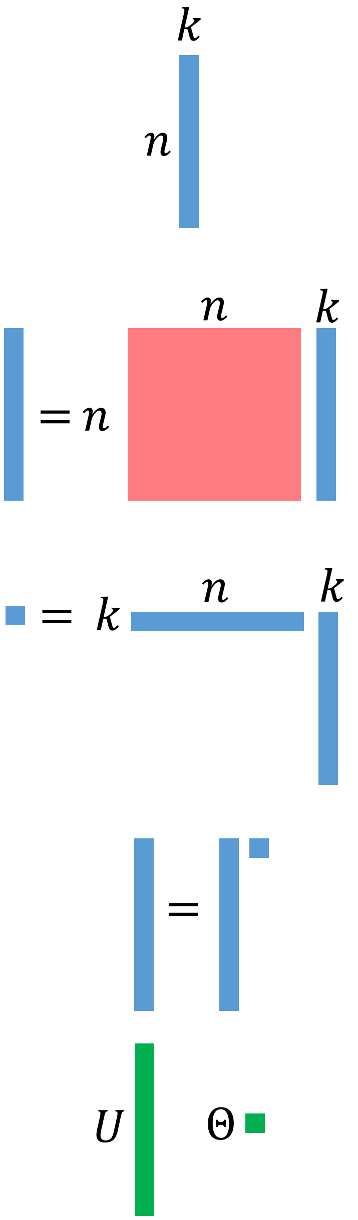
$C = \text{chol}((B + B^T)/2)$

Solve $F = Y_\nu / C$

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

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

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



Single-Pass Nyström Approximation

Given sym. PSD matrix A, target rank k

$\boldsymbol{u} \ll \boldsymbol{u_p}$

```
$G = \text{randn}(n, k)$
$[Q, \sim] = \text{qr}(G, 0)$                              (precision $\boldsymbol{u}$)
$\boldsymbol{Y = AQ}$                                      (precision $\boldsymbol{u_p}$)
Compute shift $\nu$; $Y_\nu = Y + \nu Q$                   (precision $\boldsymbol{u}$)
$B = Q^T Y_\nu$                                            (precision $\boldsymbol{u}$)
$C = \text{chol}((B + B^T)/2)$                             (precision $\boldsymbol{u}$)
Solve $F = Y_\nu / C$                                      (precision $\boldsymbol{u}$)
$[U, \Sigma, \sim] = \text{svd}(F, 0)$                     (precision $\boldsymbol{u}$)
$\Theta = \max(0, \Sigma^2 - \nu I)$                       (precision $\boldsymbol{u}$)
```

Error Bounds

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

exact Nyström
approximation

Nyström approximation
computed in
finite precision

Error Bounds

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

Error Bounds

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

Deterministic bound [Gittens, Mahoney, 2016]

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

Error Bounds

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

[C., Daužickaitė, 2024]: With failure probability at most $e^{-t^2/2} + c_1\alpha$,

$$\boxed{\left\|A_N-\hat{A}_N\right\|_2 \lesssim \alpha^{-1} n^{1/2} k\left(n^{1/2}+k^{1/2}+t\right)^2 \boldsymbol{u_p} \|A\|_2 \kappa(A_k)}$$

where A_k is the best rank-k approximation of A.

Error Bounds

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

[C., Daužickaitė, 2022]: With failure probability at most $e^{-t^2/2} + c_1\alpha$,

$$\left\|A_N - \hat{A}_N\right\|_2 \lesssim \alpha^{-1} n^{1/2} k \left(n^{1/2} + k^{1/2} + t\right)^2 \boldsymbol{u_p} \|A\|_2 \kappa(A_k)$$

where A_k is the best rank-k approximation of A.

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

$$\frac{\lambda_{k+1}}{\lambda_1} \lesssim \sqrt{n}\,\boldsymbol{u_p}$$

Error Bounds

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

[C., Daužickaitė, 2022]: With failure probability at most $e^{-t^2/2} + c_1\alpha$,

$$\left\|A_N-\hat{A}_N\right\|_2 \lesssim \alpha^{-1} n^{1/2} k \left(n^{1/2}+k^{1/2}+t\right)^2 \boldsymbol{u_p} \|A\|_2 \kappa(A_k)$$

where A_k is the best rank-k approximation of A.

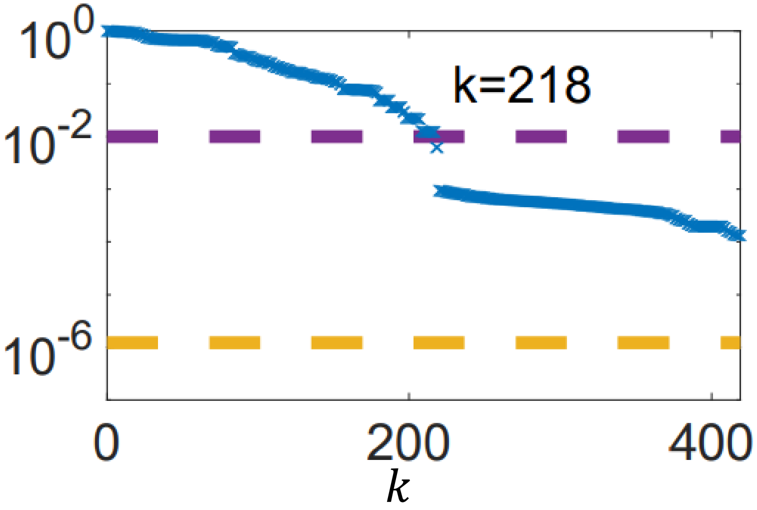
Interpretation: Likely that $\left\|A_N-\hat{A}_N\right\|_2 \gtrsim \|A-A_N\|_2$ when

$$\frac{\lambda_{k+1}}{\lambda_1} \lesssim \sqrt{n}\boldsymbol{u_p}$$

The worse the low-rank representation, the lower the precision we can use!

Numerical Experiment

Matrix: bcsstm07, $n = 420$



🟦 λ_{k+1}/λ_1

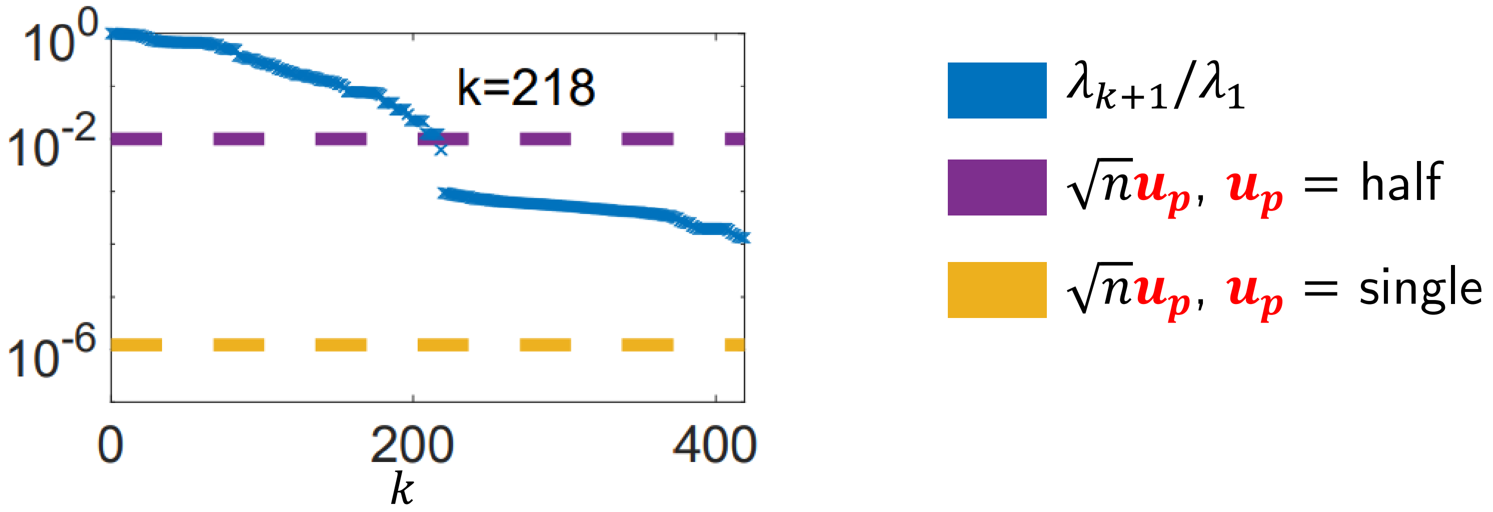
🟪 $\sqrt{n}\boldsymbol{u_p}$, $\boldsymbol{u_p}$ = half

🟨 $\sqrt{n}\boldsymbol{u_p}$, $\boldsymbol{u_p}$ = single

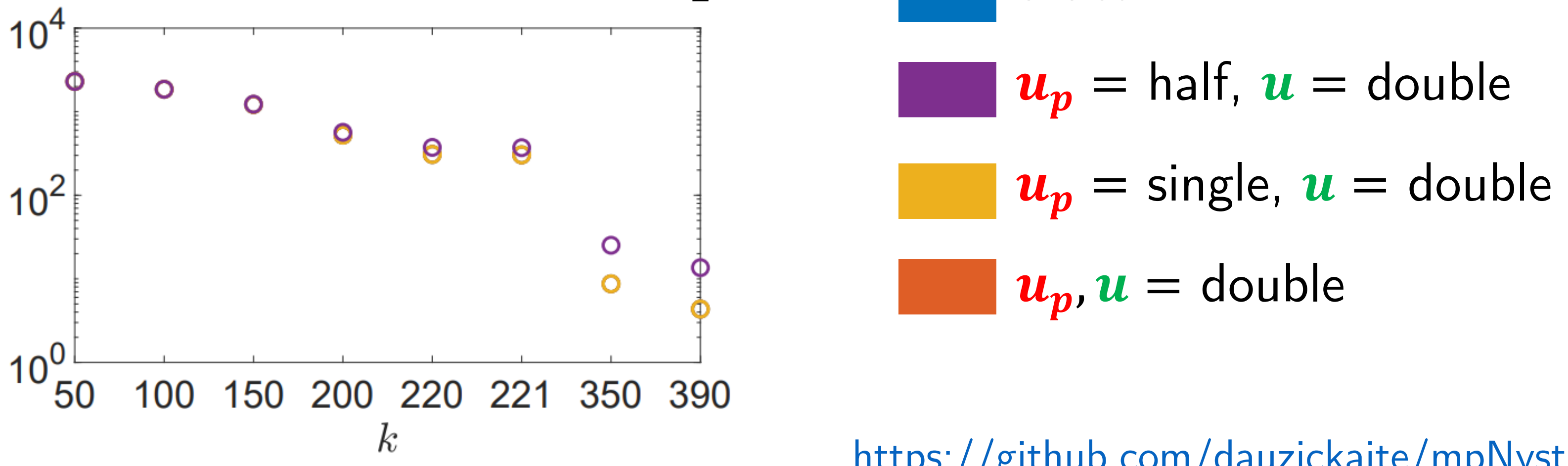
https://github.com/dauzickaite/mpNystrom

Numerical Experiment

Matrix: bcsstm07, $n = 420$

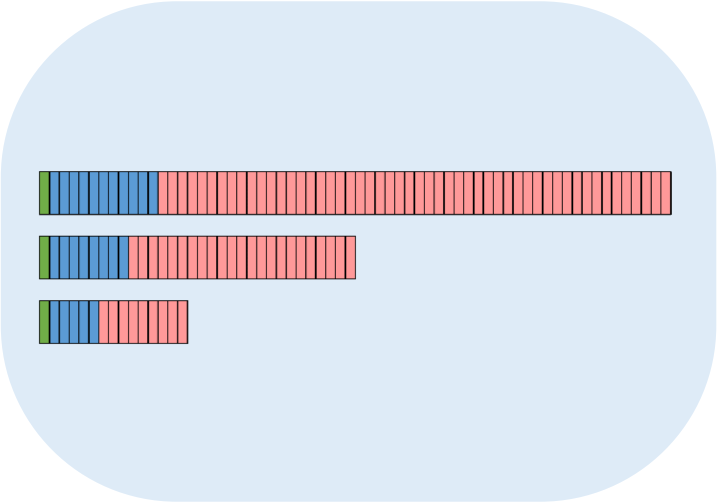


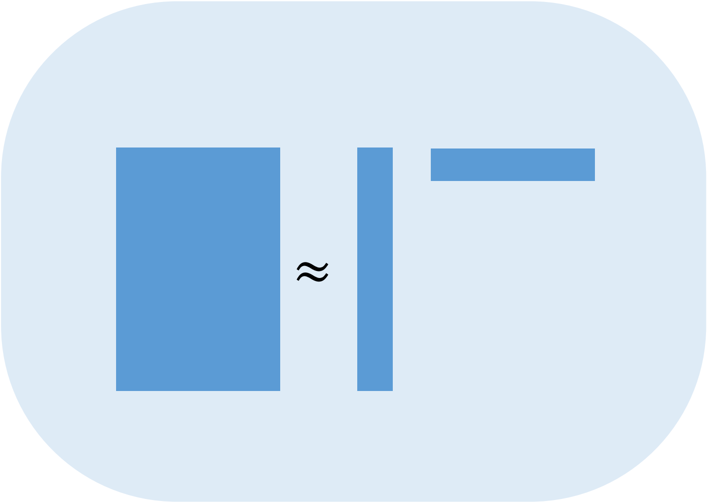
mean total error, $\|A - \hat{A}_N\|_2$



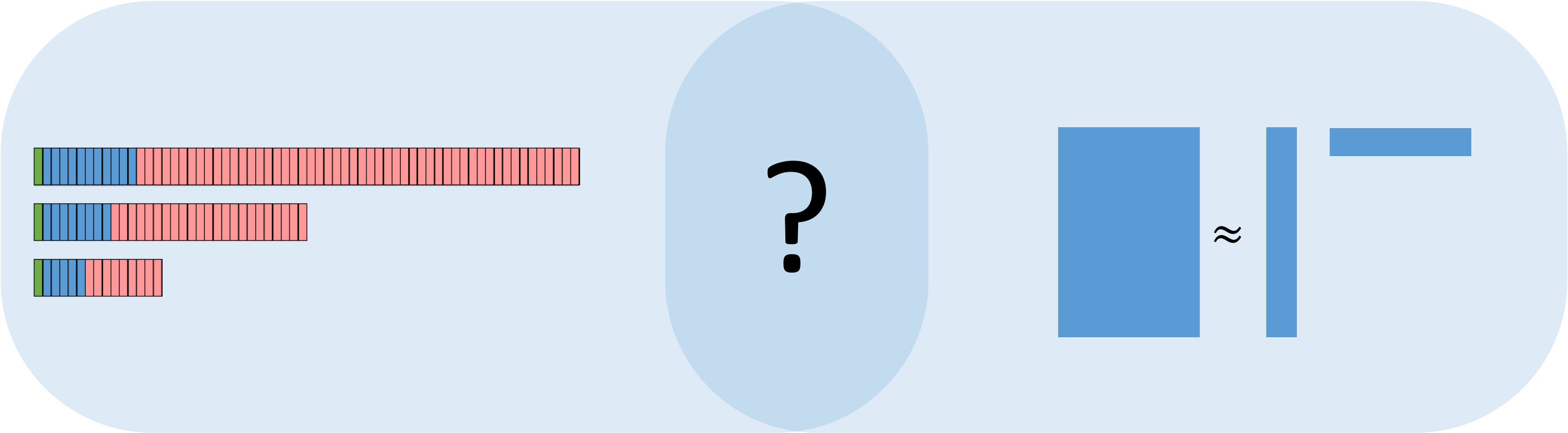
https://github.com/dauzickaite/mpNystrom

Summary and Takeaway

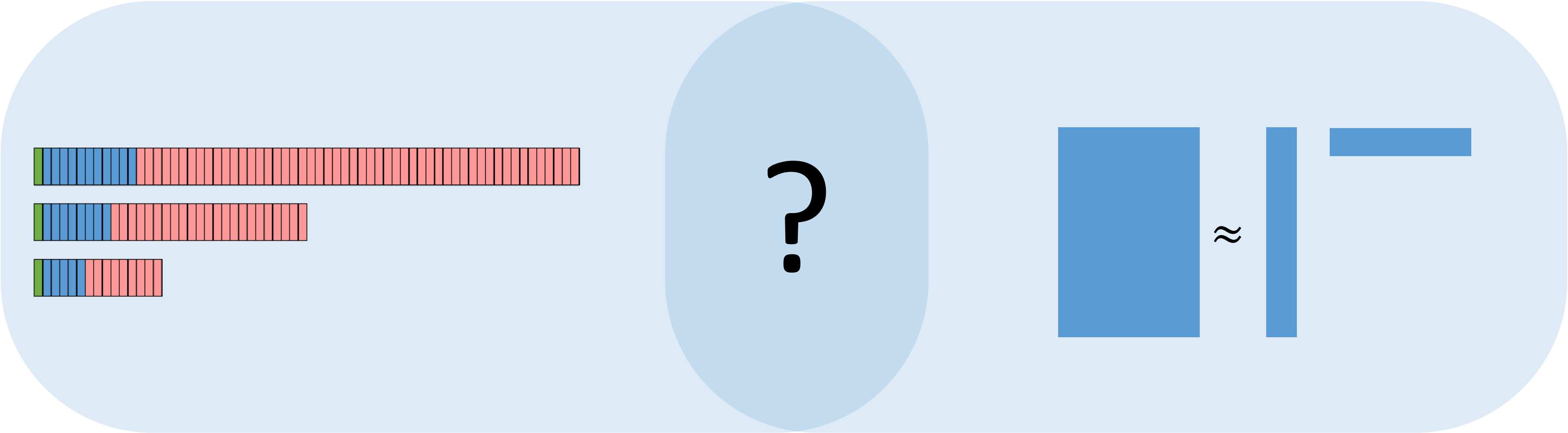




Summary and Takeaway



Summary and Takeaway



Where can ***you*** use mixed or low precision?

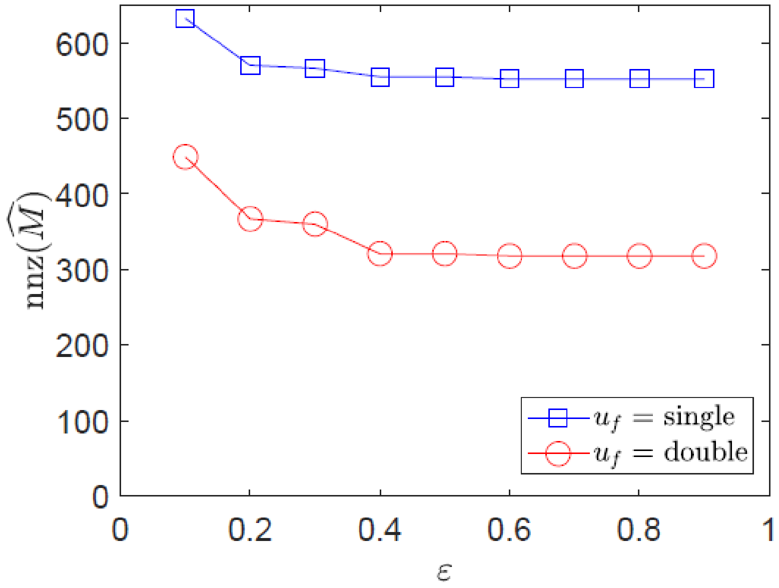
Thank You!

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

Size of SPAI Preconditioner in Low Precision

How does precision used affect the number of nonzeros in $\widehat{M}$?

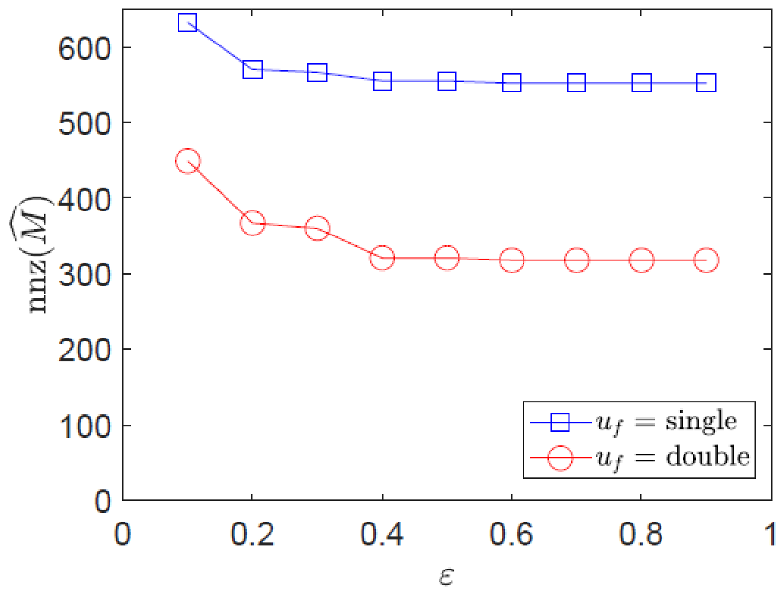
steam3



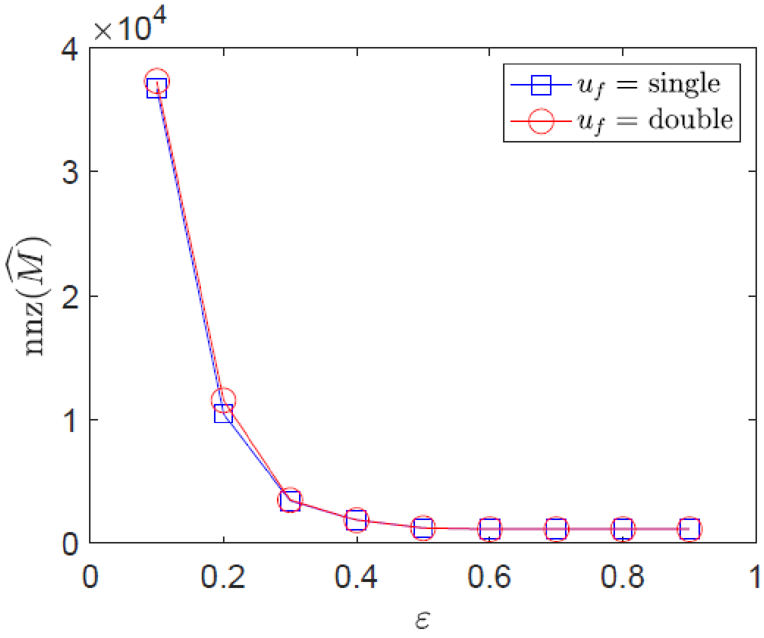
Size of SPAI Preconditioner in Low Precision

How does precision used affect the number of nonzeros in $\widehat{M}$?

steam3



saylr1



A Question

Is there a point in using precision higher than that dictated by $\boldsymbol{u_f}\mathrm{cond}_2(A^T) \leq \boldsymbol{\varepsilon}$?

Matrix: bfwa782, $n = 782$, nnz $= 7514$, $\kappa_\infty(A) = 7 \cdot 10^3$, $\mathrm{cond}(A^T) = 1 \cdot 10^3$

$(\boldsymbol{u_f}, \boldsymbol{u}, \boldsymbol{u_r})$ = (**half**, single, double)

Preconditioner	$\kappa_\infty(\tilde{A})$	Precond. nnz	GMRES-IR steps/iteration
SPAI ($\boldsymbol{\varepsilon} = 0.2$)	$2.1e + 02$	28053	67 (31, 36)
SPAI ($\boldsymbol{\varepsilon} = 0.5$)	$9.7e + 02$	7528	153 (71, 82)

A Question

Is there a point in using precision higher than that dictated by $\boldsymbol{u_f}\mathrm{cond}_2(A^T) \leq \boldsymbol{\varepsilon}$?

Matrix: bfwa782, $n = 782$, nnz $= 7514$, $\kappa_\infty(A) = 7 \cdot 10^3$, $\mathrm{cond}(A^T) = 1 \cdot 10^3$

$(\boldsymbol{u_f}, \boldsymbol{u}, \boldsymbol{u_r})$ = (**half**, single, double)

Preconditioner	$\kappa_\infty(\tilde{A})$	Precond. nnz	GMRES-IR steps/iteration
SPAI ($\boldsymbol{\varepsilon} = 0.2$)	$2.1e+02$	28053	67 (31, 36)
SPAI ($\boldsymbol{\varepsilon} = 0.5$)	$9.7e+02$	7528	153 (71, 82)

$(\boldsymbol{u_f}, \boldsymbol{u}, \boldsymbol{u_r})$ = (**single**, single, double)

Preconditioner	$\kappa_\infty(\tilde{A})$	Precond. nnz	GMRES-IR steps/iteration
SPAI ($\boldsymbol{\varepsilon} = 0.2$)	$2.2e+02$	26801	69 (32, 37)
SPAI ($\boldsymbol{\varepsilon} = 0.5$)	$9.7e+02$	7529	153 (71, 82)

Summary and Takeaway

- To efficiently use modern exascale machines, we **need to use mixed precision hardware**

- **Understanding the interaction and balance of errors** from finite precision and sources of algorithmic approximation is thus crucial

- Careful analysis can reveal **not only limitations, but opportunities!**

Summary and Takeaway

- To efficiently use modern exascale machines, we **need to use mixed precision hardware**

- **Understanding the interaction and balance of errors** from finite precision and sources of algorithmic approximation is thus crucial

- Careful analysis can reveal **not only limitations, but opportunities!**

Where can you use mixed or low precision?