Challenges and Opportunities in Mixed Precision Numerical Linear Algebra

Erin Carson

Charles University

ICL Lunch Talk

May 13, 2022

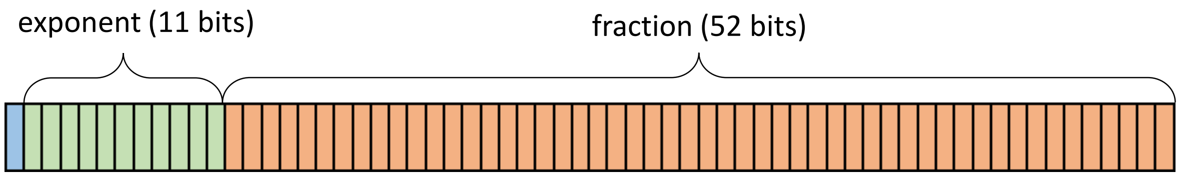
FACULTY OF MATHEMATICS AND PHYSICS
Charles University

We acknowledge funding from Charles Univ. PRIMUS project No. PRIMUS/19/SCI/11, Charles Univ. Research Program No. UNCE/SCI/023, 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.

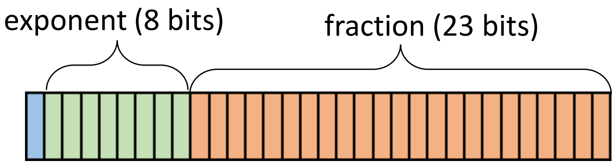
Floating Point Formats

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

IEEE double (FP64)

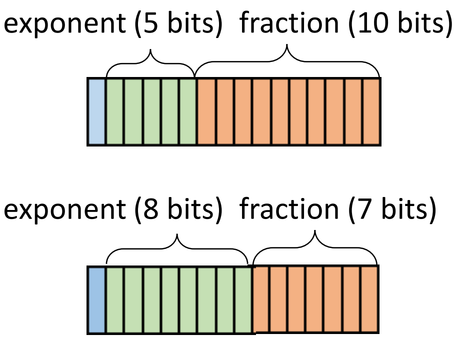


IEEE single (FP32)



IEEE half (FP16)

bfloat16



	size	range	u
fp64	64 bits	$10^{\pm 308}$	1×10^{-16}
fp32	32 bits	$10^{\pm 38}$	6×10^{-8}
fp16	16 bits	$10^{\pm 5}$	5×10^{-4}
bfloat16	16 bits	$10^{\pm 38}$	4×10^{-3}

Hardware Support for Multiprecision Computation

Use of low precision in machine learning has driven emergence of low-precision capabilities in hardware:

- Half precision (FP16) defined as storage format in 2008 IEEE standard
- ARM NEON: SIMD architecture, instructions for 8x16-bit, 4x32-bit, 2x64-bit
- AMD Radeon Instinct MI25 GPU, 2017:
  - single: 12.3 TFLOPS, half: 24.6 TFLOPS
- NVIDIA Tesla P100, 2016: native ISA support for 16-bit FP arithmetic
- NVIDIA Tesla V100, 2017: tensor cores for half precision;

  4x4 matrix multiply in one clock cycle
  - double: 7 TFLOPS, half+tensor: 112 TFLOPS (**16x!**)
- Google's Tensor processing unit (TPU)
- NVIDIA A100, 2020: tensor cores with multiple supported precisions: FP16, FP64, Binary, INT4, INT8, bfloat16
- NVIDIA H100, 2022: now with quarter-precision (FP8) tensor cores
- Future exascale supercomputers: (~2021) Expected extensive support for reduced-precision arithmetic (32/16/8-bit)

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]

HPL-AI Benchmark

- Like HPL, solves dense Ax=b, results still to double precision accuracy
- Achieves this via mixed-precision iterative refinement

HPL-AI Benchmark

Rank	Site	Computer	Cores	HPL-AI (Eflop/s)	TOP500 Rank	HPL Rmax (Eflop/s)	Speedup
1	RIKEN	Fugaku	7,630,848	2.000	1	0.4420	4.5
2	DOE/SC/ORNL	Summit	2,414,592	1.411	2	0.1486	9.5
3	NVIDIA	Selene	555,520	0.630	6	0.0630	9.9
4	DOE/SC/LBNL	Perlmutter	761,856	0.590	5	0.0709	8.3
5	FZJ	JUWELS BM	449,280	0.470	8	0.0440	10.0
6	University of Florida	HiPerGator	138,880	0.170	31	0.0170	9.9
7	SberCloud	Christofari Neo	98,208	0.123	44	0.0120	10.3
8	DOE/SC/ANL	Polaris	259,840	0.114	13	0.0238	4.8
9	ITC	Wisteria	368,640	0.100	18	0.0220	4.5
10	NSC	Berzelius	59,520	0.050	95	0.0053	9.5
11	Nagoya	Flow Type I	110,592	0.030	74	0.0066	4.5
12	NVIDIA	Tethys	19,840	0.024	297	0.0023	10.8
13	NVIDIA	DGX Saturn V	87,040	0.022	118	0.0040	5.5
14	CloudMTS	MTS GROM	19,840	0.015	296	0.0023	6.6
15	Calcul Quebec/Compute Canada	Narval	76,320	0.014	84	0.0059	2.4
16	DOE/SC/ANL	ThetaGPU	280,320	0.012	71	0.0069	1.7
17	Indiana University	Big Red 200 GPU	31,744	0.006	216	0.0026	2.4
18	Texas A&M University	Grace GPU	26,400	0.004	335	0.0021	1.7

HPL-AI Benchmark

Rank	Site	Computer	Cores	HPL-AI (Eflop/s)	TOP500 Rank	HPL Rmax (Eflop/s)	Speedup
1	RIKEN	Fugaku	7,630,848	2.000	1	0.4420	4.5
2	DOE/SC/ORNL	Summit	2,414,592	1.411	2	0.1486	9.5
3	NVIDIA	Selene	555,520	0.630	6	0.0630	9.9
4	DOE/SC/LBNL	Perlmutter	761,856	0.590	5	0.0709	8.3
5	FZJ	JUWELS BM	449,280	0.470	8	0.0440	10.0
6	University of Florida	HiPerGator	138,880	0.170	31	0.0170	9.9
7	SberCloud	Christofari Neo	98,208	0.123	44	0.0120	10.3
8	DOE/SC/ANL	Polaris	259,840	0.114	13	0.0238	4.8
9	ITC	Wisteria	368,640	0.100	18	0.0220	4.5
10	NSC	Berzelius	59,520	0.050	95	0.0053	9.5
11	Nagoya	Flow Type I	110,592	0.030	74	0.0066	4.5
12	NVIDIA	Tethys	19,840	0.024	297	0.0023	10.8
13	NVIDIA	DGX Saturn V	87,040	0.022	118	0.0040	5.5
14	CloudMTS	MTS GROM	19,840	0.015	296	0.0023	6.6
15	Calcul Quebec/Compute Canada	Narval	76,320	0.014	84	0.0059	2.4
16	DOE/SC/ANL	ThetaGPU	280,320	0.012	71	0.0069	1.7
17	Indiana University	Big Red 200 GPU	31,744	0.006	216	0.0026	2.4
18	Texas A&M University	Grace GPU	26,400	0.004	335	0.0021	1.7

HPL-AI Benchmark

Rank	Site	Computer	Cores	HPL-AI (Eflop/s)	TOP500 Rank	HPL Rmax (Eflop/s)	Speedup
1	RIKEN	Fugaku	7,630,848	2.000	1	0.4420	4.5
2	DOE/SC/ORNL	Summit	2,414,592	1.411	2	0.1486	9.5
3	NVIDIA	Selene	555,520	0.630	6	0.0630	9.9
4	DOE/SC/LBNL	Perlmutter	761,856	0.590	5	0.0709	8.3
5	FZJ	JUWELS BM	449,280	0.470	8	0.0440	10.0
6	University of Florida	HiPerGator	138,880	0.170	31	0.0170	9.9
7	SberCloud	Christofari Neo	98,208	0.123	44	0.0120	10.3
8	DOE/SC/ANL	Polaris	259,840	0.114	13	0.0238	4.8
9	ITC	Wisteria	368,640	0.100	18	0.0220	4.5
10	NSC	Berzelius	59,520	0.050	95	0.0053	9.5
11	Nagoya	Flow Type I	110,592	0.030	74	0.0066	4.5
12	NVIDIA	Tethys	19,840	0.024	297	0.0023	10.8
13	NVIDIA	DGX Saturn V	87,040	0.022	118	0.0040	5.5
14	CloudMTS	MTS GROM	19,840	0.015	296	0.0023	6.6
15	Calcul Quebec/Compute Canada	Narval	76,320	0.014	84	0.0059	2.4
16	DOE/SC/ANL	ThetaGPU	280,320	0.012	71	0.0069	1.7
17	Indiana University	Big Red 200 GPU	31,744	0.006	216	0.0026	2.4
18	Texas A&M University	Grace GPU	26,400	0.004	335	0.0021	1.7

Challenges of low precision

- 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

- 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]

- 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

- 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]

- 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: $fl(x) = x(1+\delta), \quad |\delta| \leq u$
  - Lose something small when computing: $fl(x \text{ op } y) = (x \text{ op } y)(1+\delta), \quad |\delta| \leq u$

Challenges of low precision

- 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]

- 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]

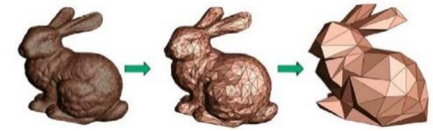
- Larger unit roundoff
  - Lose something small when storing: $fl(x) = x(1+\delta), \ \ |\delta| \leq u$
  - Lose something small when computing: $fl(x \text{ op } y) = (x \text{ op } y)(1+\delta), \ \ |\delta| \leq u$

Does it matter?

Inexact computations

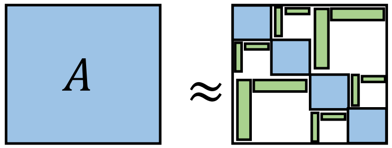
- In real computations we have many sources of inexactness
  - Imperfect data, measurement error
  - Modeling error, discretization error
  - Intentional approximation to improve performance
    - Reduced models, Low-rank representations, sparsification, randomization

Model Reduction



[Schilders, van der Vorst, Rommes, 2008]

Low-rank (hierarchical) approximation



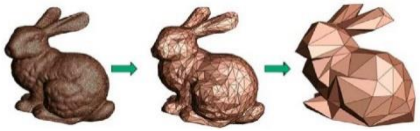
Sparsification, Randomized algorithms

[Sinha, 2018]

Inexact computations

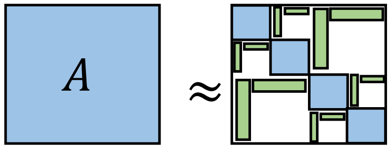
- In real computations we have many sources of inexactness
  - Imperfect data, measurement error
  - Modeling error, discretization error
  - Intentional approximation to improve performance
    - Reduced models, Low-rank representations, sparsification, randomization

Model Reduction

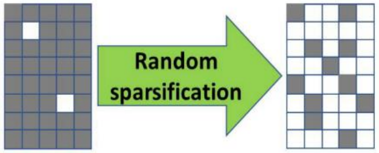


[Schilders, van der Vorst, Rommes, 2008]

Low-rank (hierarchical) approximation



Sparsification, Randomized algorithms

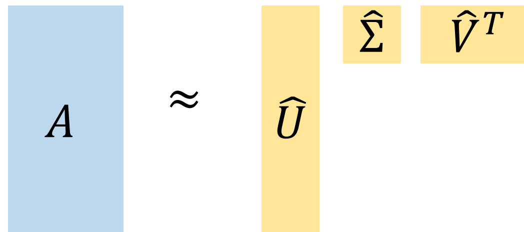


[Sinha, 2018]

- Given that we are already working with so much inexactness, does it matter if we use lower precision?
  - Analysis of accuracy in techniques that use intentional approximation ***almost always*** assume that roundoff error is small enough to be ignored
  - Is this true? Is it true even if we use low precision?

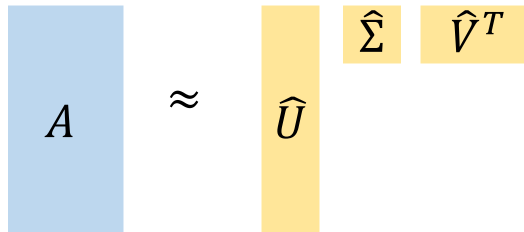
Example: Randomized Algorithms

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

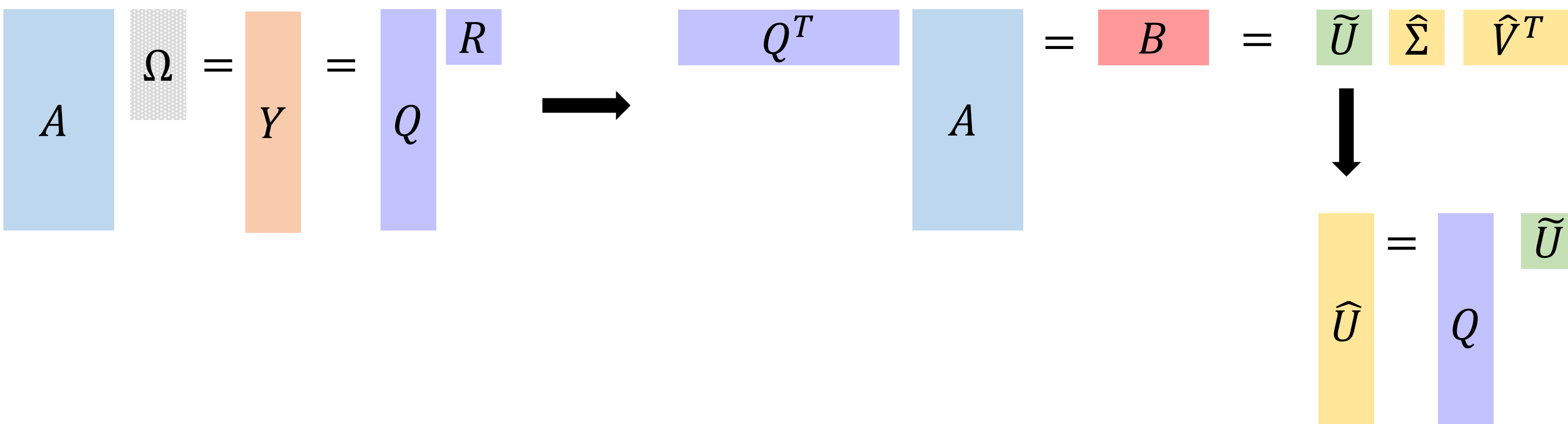


Example: Randomized Algorithms

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



- Randomized SVD:



Assuming exact arithmetic:
If Q satisfies $\|A - QQ^TA\| \le \varepsilon$, then $\left\|A - \hat{U}\hat{\Sigma}\hat{V}^T\right\| \le \varepsilon$

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;
```

$[U,S,V] \quad = \mathrm{svd}(A)$: non-randomized SVD, exact arithmetic

$[\widehat{U},\hat{S},\widehat{V}] \quad = \mathrm{rsvd}(A)$: randomized SVD, exact arithmetic

$[\widehat{U}_d,\hat{S}_d,\widehat{V}_d] = \mathrm{rsvd}(A)$: randomized SVD, double precision

$[\widehat{U}_h,\hat{S}_h,\widehat{V}_h] = \mathrm{rsvd}(A)$: randomized SVD, 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;
```

$[U,S,V] = \text{svd}(A)$: non-randomized SVD, exact arithmetic

$[\hat{U},\hat{S},\hat{V}] = \text{rsvd}(A)$: randomized SVD, exact arithmetic

$[\hat{U}_d,\hat{S}_d,\hat{V}_d] = \text{rsvd}(A)$: randomized SVD, double precision

$[\hat{U}_h,\hat{S}_h,\hat{V}_h] = \text{rsvd}(A)$: randomized SVD, half precision

Mode 3: Geometrically distributed singular values

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

$\|A - \hat{U}\hat{S}\hat{V}^T\|_2$ = 4.92e-03

$\|A - \hat{U}_d\hat{S}_d\hat{V}_d{}^T\|_2$ = 4.92e-03

$\|A - \hat{U}_h\hat{S}_h\hat{V}_h{}^T\|_2$ = 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;
```

$[U,S,V] = \text{svd}(A)$: non-randomized SVD, exact arithmetic

$[\hat{U},\hat{S},\hat{V}] = \text{rsvd}(A)$: randomized SVD, exact arithmetic

$[\hat{U}_d,\hat{S}_d,\hat{V}_d] = \text{rsvd}(A)$: randomized SVD, double precision

$[\hat{U}_h,\hat{S}_h,\hat{V}_h] = \text{rsvd}(A)$: randomized SVD, half precision

Mode 3: Geometrically distributed singular values

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

$\|A - \hat{U}\hat{S}\hat{V}^T\|_2 = 4.92\text{e-}03$

$\|A - \hat{U}_d\hat{S}_d\hat{V}_d^T\|_2 = 4.92\text{e-}03$

$\|A - \hat{U}_h\hat{S}_h\hat{V}_h^T\|_2 = 4.92\text{e-}03$

Mode 1: one large singular value

$\|A - USV^T\|_2 = 1.00\text{e-}06$

$\|A - \hat{U}\hat{S}\hat{V}^T\|_2 = 1.17\text{e-}06$

$\|A - \hat{U}_d\hat{S}_d\hat{V}_d^T\|_2 = 1.17\text{e-}06$

$\|A - \hat{U}_h\hat{S}_h\hat{V}_h^T\|_2 = 1.11\text{e-}05$

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;
```

$[U,S,V] = \text{svd}(A)$: non-randomized SVD, exact arithmetic

$[\hat{U},\hat{S},\hat{V}] = \text{rsvd}(A)$: randomized SVD, exact arithmetic

$[\hat{U}_d,\hat{S}_d,\hat{V}_d] = \text{rsvd}(A)$: randomized SVD, double precision

$[\hat{U}_h,\hat{S}_h,\hat{V}_h] = \text{rsvd}(A)$: randomized SVD, half precision

Mode 3: Geometrically distributed singular values

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

$\|A - \hat{U}\hat{S}\hat{V}^T\|_2 = 4.92\text{e-}03$

$\|A - \hat{U}_d\hat{S}_d\hat{V}_d^T\|_2 = 4.92\text{e-}03$

$\|A - \hat{U}_h\hat{S}_h\hat{V}_h^T\|_2 = 4.92\text{e-}03$

Mode 1: one large singular value

$\|A - USV^T\|_2 = 1.00\text{e-}06$

$\|A - \hat{U}\hat{S}\hat{V}^T\|_2 = 1.17\text{e-}06$

$\|A - \hat{U}_d\hat{S}_d\hat{V}_d^T\|_2 = 1.17\text{e-}06$

$\|A - \hat{U}_h\hat{S}_h\hat{V}_h^T\|_2 = 1.11\text{e-}05$

Use of low precision leads to an order magnitude loss of accuracy! Roundoff error can't be ignored!

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;
```

$[U, S, V] = \mathrm{svd}(A)$: non-randomized SVD, exact arithmetic

$[\hat{U}, \hat{S}, \hat{V}] = \mathrm{rsvd}(A)$: randomized SVD, exact arithmetic

$[\hat{U}_d, \hat{S}_d, \hat{V}_d] = \mathrm{rsvd}(A)$: randomized SVD, double precision

$[\hat{U}_h, \hat{S}_h, \hat{V}_h] = \mathrm{rsvd}(A)$: randomized SVD, half precision

Error bound no longer holds!

Mode 3: Geometrically distributed singular values

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

$\|A - \hat{U}\hat{S}\hat{V}^T\|_2 = 4.92\text{e-}03$

$\left\|A - \hat{U}_d\hat{S}_d\hat{V}_d^{\,T}\right\|_2 = 4.92\text{e-}03$

$\left\|A - \hat{U}_h\hat{S}_h\hat{V}_h^{\,T}\right\|_2 = 4.92\text{e-}03$

Mode 1: one large singular value

$\|A - USV^T\|_2 = 1.00\text{e-}06$

$\|A - \hat{U}\hat{S}\hat{V}^T\|_2 = 1.17\text{e-}06$

$\left\|A - \hat{U}_d\hat{S}_d\hat{V}_d^{\,T}\right\|_2 = 1.17\text{e-}06$

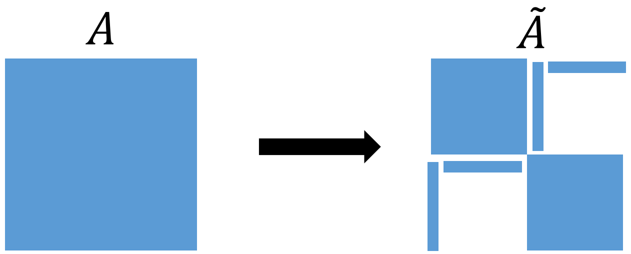
$\left\|A - \hat{U}_h\hat{S}_h\hat{V}_h^{\,T}\right\|_2 = 1.11\text{e-}05$

$\left\|A - Q_h Q_h^T A\right\|_2 = 3.59\text{e-}06$

Use of low precision leads to an order magnitude loss of accuracy! Roundoff error can't be ignored!

Example: Low-Rank Approximation

- Block low-rank approximation and hierarchical matrix representations arise in a variety of applications



- Work on mixed and low precision in block low-rank computations

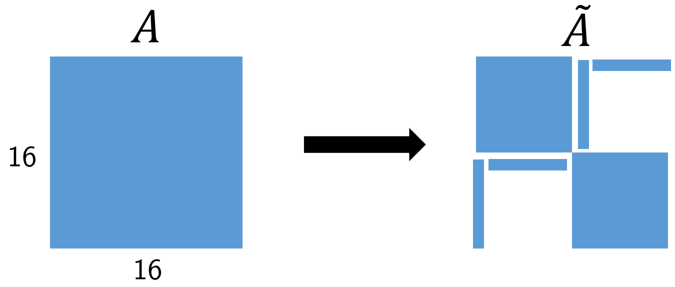
- [Higham, Mary, 2019]: block low-rank LU factorization preconditioner that exploits numerically low-rank structure of the error for LU computed in low precision
- [Higham, Mary, 2019]: Interplay of roundoff error and approximation error in solving block low-rank linear systems using LU
- [Buttari, et al., 2020]: block low-rank single precision coarse grid solves in multigrid
- [Amestoy et al., 2021]: Mixed precision low rank approximation and application to block low-rank LU factorization

Example: Low-Rank Approximation

Inverse multiquadratic kernel:

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

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

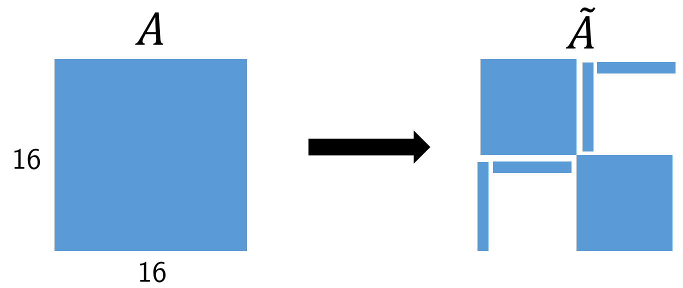


Example: Low-Rank Approximation

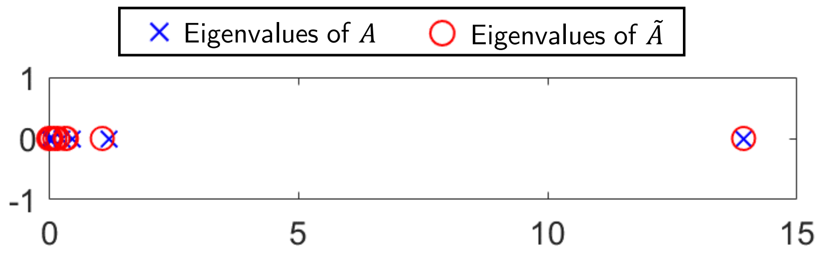
Inverse multiquadratic kernel:

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

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



Exact arithmetic SVD:

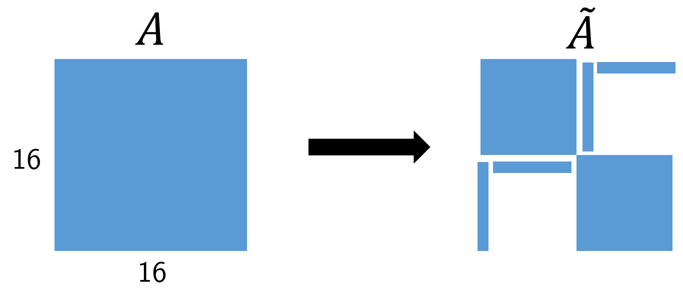


Example: Low-Rank Approximation

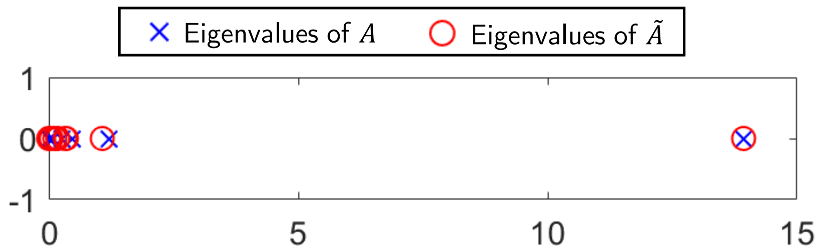
Inverse multiquadratic kernel:

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

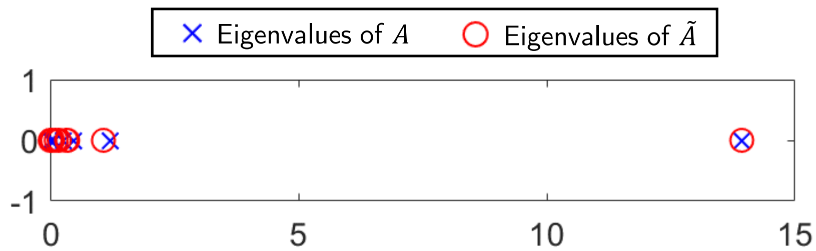
A is SPD. Low-rank approximation of A should also be SPD!



Exact arithmetic SVD:



Half precision SVD:

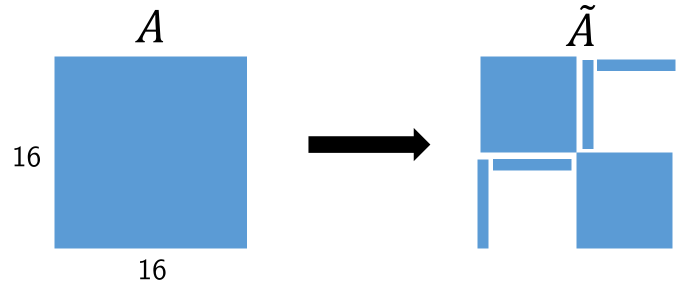


Example: Low-Rank Approximation

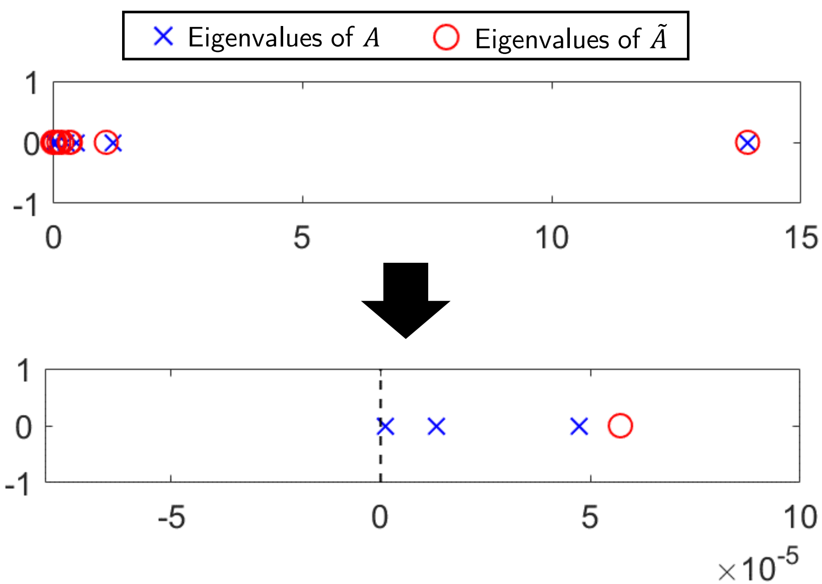
Inverse multiquadratic kernel:

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

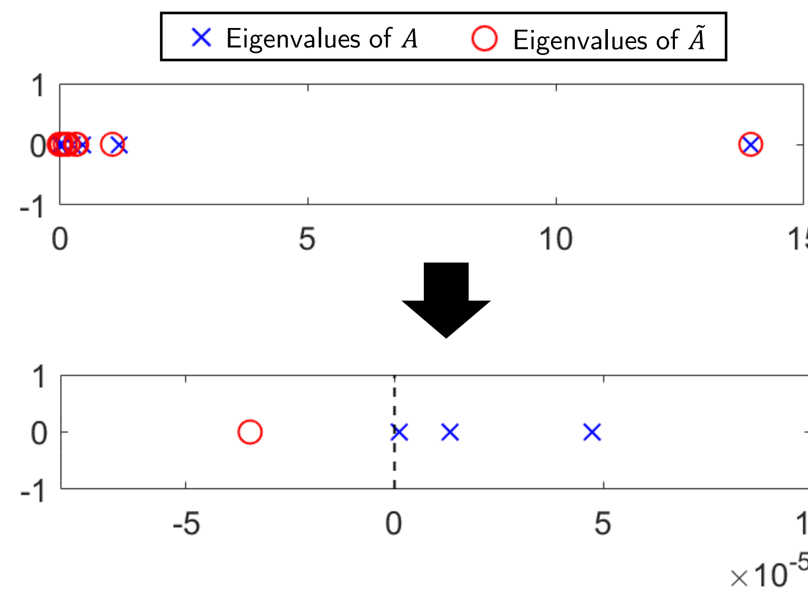
A is SPD. Low-rank approximation of A should also be SPD!



Exact arithmetic SVD:



Half precision SVD:

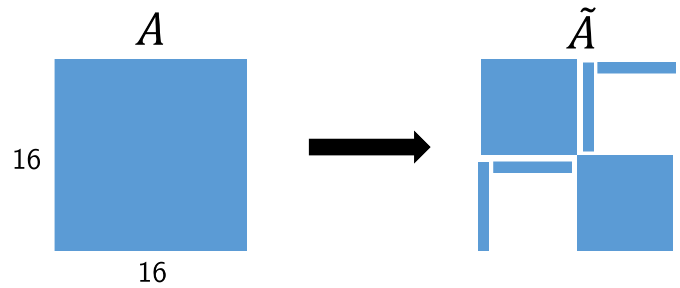


Example: Low-Rank Approximation

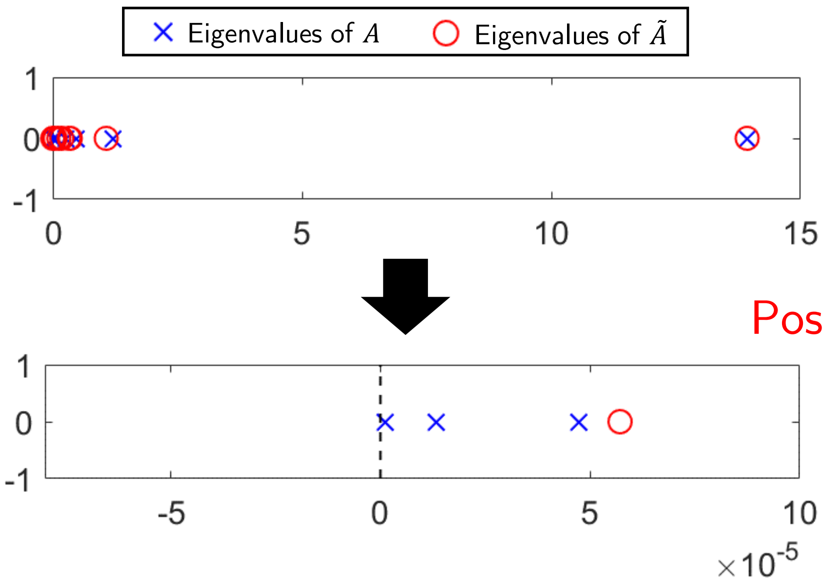
Inverse multiquadratic kernel:

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

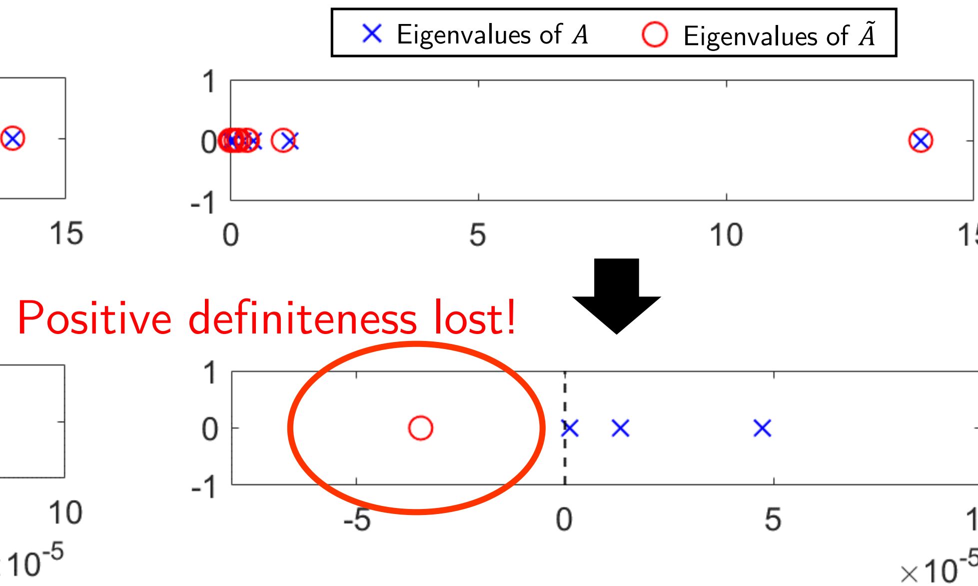
A is SPD. Low-rank approximation of A should also be SPD!



Exact arithmetic SVD:

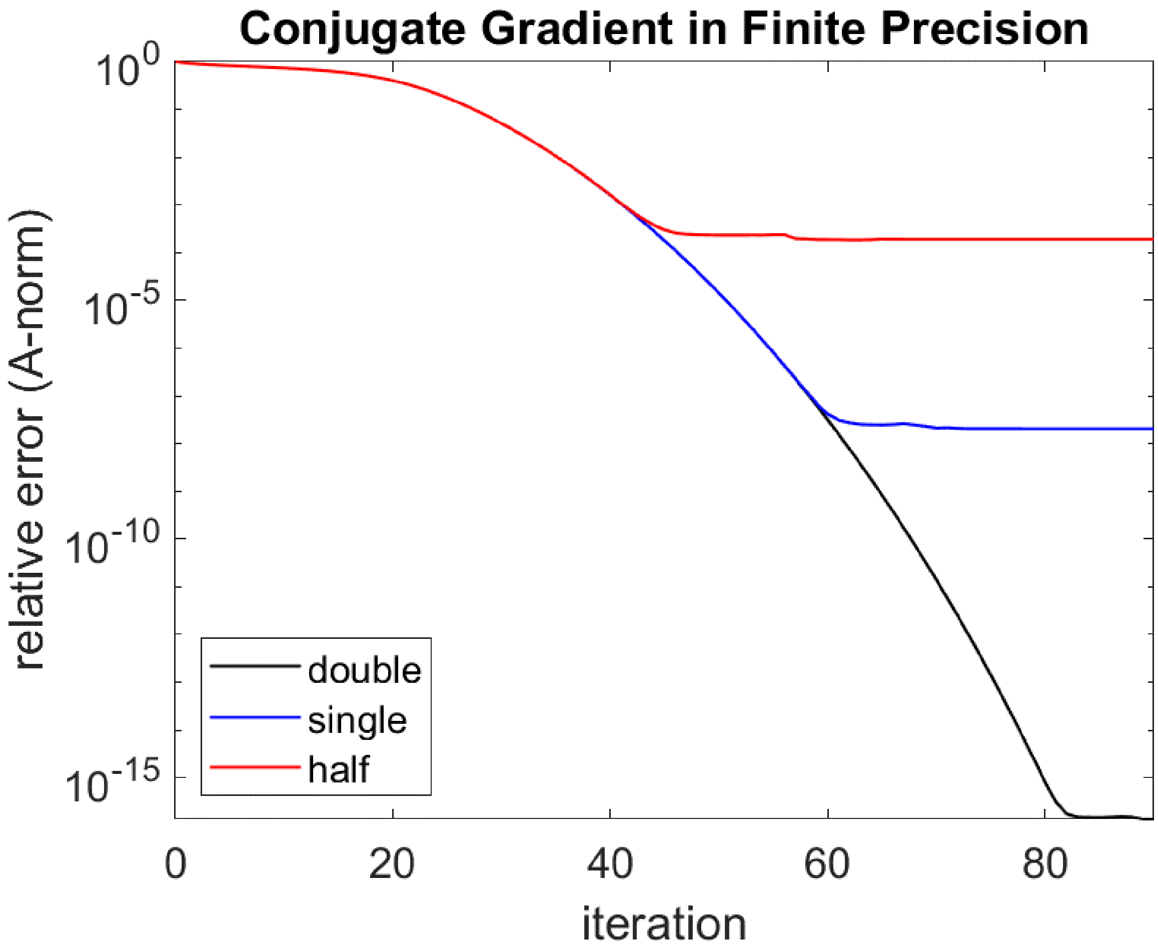


Half precision SVD:



Example: Iterative Methods

```
A = diag(linspace(.001,1,100));
[V,~] = eig(A);
b = V'*ones(n,1);
```

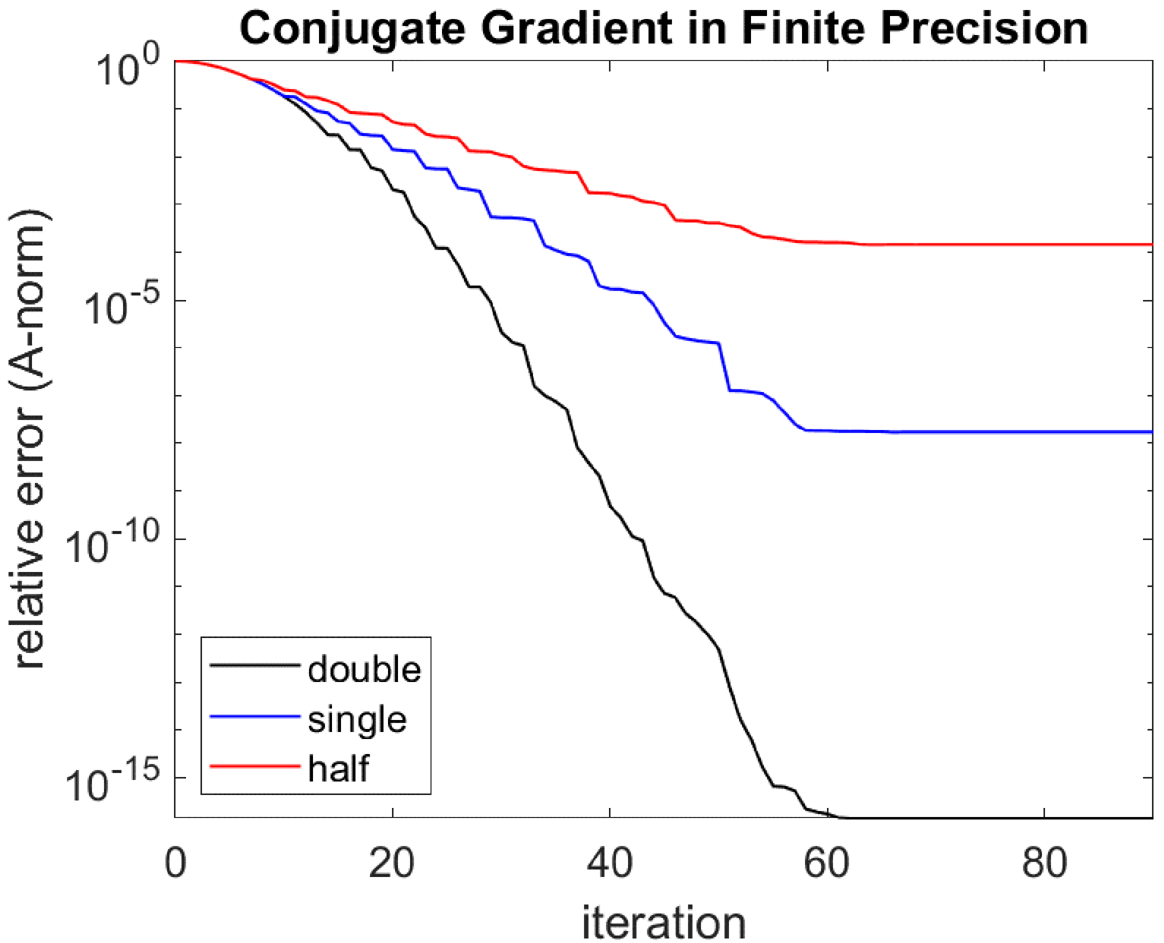


Example: Iterative Methods

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

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

```
[V,~] = eig(A);
b = V'*ones(n,1);
```



Takeaway

- Low precision can have massive performance benefits but must be used with caution!

- Many opportunities for using mixed and low precision computation in scientific applications

- Need to develop a theoretical understanding of how mixed precision algorithms behave; need to revisit analyses of algorithms and techniques that ignore finite precision

Iterative Refinement for $Ax = b$

Iterative refinement: well-established method for improving an approximate solution to $Ax = b$

A is $n \times n$ and nonsingular; u is unit roundoff

Solve $Ax_0 = b$ by LU factorization (in precision u_f)

for $i = 0\colon \mathrm{maxit}$

$r_i = b - Ax_i$ (in precision u_r)

Solve $Ad_i = r_i$ (in precision u_s)

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

Iterative Refinement in 3 Precisions

- 3-precision iterative refinement [C. and Higham, 2018]

u_f = factorization precision, u = working precision, u_r = residual precision

$$u_f \geq u \geq u_r$$

$\boldsymbol{u_s}$ is the *effective precision* of the solve, with $\boldsymbol{u} \leq \boldsymbol{u_s} \leq \boldsymbol{u_f}$

- For triangular solves with LU factors: $\boldsymbol{u_s} = \boldsymbol{u_f}$
- For GMRES preconditioned by LU factors, $\boldsymbol{u_s} = \boldsymbol{u}$ [C. and Higham, 2017]

- New analysis **generalizes** existing types of IR:

Traditional	$u_f = u, u_r = u^2$
Fixed precision	$u_f = u = u_r$
Lower precision factorization	$u_f^2 = u = u_r$

- Enables **new** types of IR: (half, single, double), (half, single, quad), (half, double, quad), etc.

IR3: Summary

Standard (LU-based) IR in three precisions ($u_s = u_f$)

Half $\approx 10^{-4}$, Single $\approx 10^{-8}$, Double $\approx 10^{-16}$, Quad $\approx 10^{-34}$

				Backward error		
u_f	u	u_r	max $\kappa_\infty(A)$	norm	comp	Forward error
H	S	S	10^4	10^{-8}	10^{-8}	$\mathrm{cond}(A,x) \cdot 10^{-8}$
H	S	D	10^4	10^{-8}	10^{-8}	10^{-8}
H	D	D	10^4	10^{-16}	10^{-16}	$\mathrm{cond}(A,x) \cdot 10^{-16}$
H	D	Q	10^4	10^{-16}	10^{-16}	10^{-16}
S	S	S	10^8	10^{-8}	10^{-8}	$\mathrm{cond}(A,x) \cdot 10^{-8}$
S	S	D	10^8	10^{-8}	10^{-8}	10^{-8}
S	D	D	10^8	10^{-16}	10^{-16}	$\mathrm{cond}(A,x) \cdot 10^{-16}$
S	D	Q	10^8	10^{-16}	10^{-16}	10^{-16}

IR3: Summary

Standard (LU-based) IR in three precisions ($u_s = u_f$)

Half $\approx 10^{-4}$, Single $\approx 10^{-8}$, Double $\approx 10^{-16}$, Quad $\approx 10^{-34}$

	u_f	u	u_r	max $\kappa_\infty(A)$	Backward error norm	Backward error comp	Forward error
LP fact.	H	S	S	10^4	10^{-8}	10^{-8}	$\text{cond}(A, x) \cdot 10^{-8}$
	H	S	D	10^4	10^{-8}	10^{-8}	10^{-8}
LP fact.	H	D	D	10^4	10^{-16}	10^{-16}	$\text{cond}(A, x) \cdot 10^{-16}$
	H	D	Q	10^4	10^{-16}	10^{-16}	10^{-16}
	S	S	S	10^8	10^{-8}	10^{-8}	$\text{cond}(A, x) \cdot 10^{-8}$
	S	S	D	10^8	10^{-8}	10^{-8}	10^{-8}
LP fact.	S	D	D	10^8	10^{-16}	10^{-16}	$\text{cond}(A, x) \cdot 10^{-16}$
	S	D	Q	10^8	10^{-16}	10^{-16}	10^{-16}

IR3: Summary

Standard (LU-based) IR in three precisions ($u_s = u_f$)

Half $\approx 10^{-4}$, Single $\approx 10^{-8}$, Double $\approx 10^{-16}$, Quad $\approx 10^{-34}$

	u_f	u	u_r	max $\kappa_\infty(A)$	Backward error norm	Backward error comp	Forward error
LP fact.	H	S	S	10^4	10^{-8}	10^{-8}	$\mathrm{cond}(A,x)\cdot 10^{-8}$
	H	S	D	10^4	10^{-8}	10^{-8}	10^{-8}
LP fact.	H	D	D	10^4	10^{-16}	10^{-16}	$\mathrm{cond}(A,x)\cdot 10^{-16}$
	H	D	Q	10^4	10^{-16}	10^{-16}	10^{-16}
Fixed	S	S	S	10^8	10^{-8}	10^{-8}	$\mathrm{cond}(A,x)\cdot 10^{-8}$
	S	S	D	10^8	10^{-8}	10^{-8}	10^{-8}
LP fact.	S	D	D	10^8	10^{-16}	10^{-16}	$\mathrm{cond}(A,x)\cdot 10^{-16}$
	S	D	Q	10^8	10^{-16}	10^{-16}	10^{-16}

IR3: Summary

Standard (LU-based) IR in three precisions ($u_s = u_f$)

Half $\approx 10^{-4}$, Single $\approx 10^{-8}$, Double $\approx 10^{-16}$, Quad $\approx 10^{-34}$

	u_f	u	u_r	max $\kappa_\infty(A)$	Backward error norm	Backward error comp	Forward error
LP fact.	H	S	S	10^4	10^{-8}	10^{-8}	$\mathrm{cond}(A,x) \cdot 10^{-8}$
	H	S	D	10^4	10^{-8}	10^{-8}	10^{-8}
LP fact.	H	D	D	10^4	10^{-16}	10^{-16}	$\mathrm{cond}(A,x) \cdot 10^{-16}$
	H	D	Q	10^4	10^{-16}	10^{-16}	10^{-16}
Fixed	S	S	S	10^8	10^{-8}	10^{-8}	$\mathrm{cond}(A,x) \cdot 10^{-8}$
Trad.	S	S	D	10^8	10^{-8}	10^{-8}	10^{-8}
LP fact.	S	D	D	10^8	10^{-16}	10^{-16}	$\mathrm{cond}(A,x) \cdot 10^{-16}$
	S	D	Q	10^8	10^{-16}	10^{-16}	10^{-16}

IR3: Summary

Standard (LU-based) IR in three precisions ($u_s = u_f$)

Half $\approx 10^{-4}$, Single $\approx 10^{-8}$, Double $\approx 10^{-16}$, Quad $\approx 10^{-34}$

	u_f	u	u_r	max $\kappa_\infty(A)$	Backward error: norm	Backward error: comp	Forward error
LP fact.	H	S	S	10^4	10^{-8}	10^{-8}	$\text{cond}(A,x)\cdot 10^{-8}$
New	H	S	D	10^4	10^{-8}	10^{-8}	10^{-8}
LP fact.	H	D	D	10^4	10^{-16}	10^{-16}	$\text{cond}(A,x)\cdot 10^{-16}$
New	H	D	Q	10^4	10^{-16}	10^{-16}	10^{-16}
Fixed	S	S	S	10^8	10^{-8}	10^{-8}	$\text{cond}(A,x)\cdot 10^{-8}$
Trad.	S	S	D	10^8	10^{-8}	10^{-8}	10^{-8}
LP fact.	S	D	D	10^8	10^{-16}	10^{-16}	$\text{cond}(A,x)\cdot 10^{-16}$
New	S	D	Q	10^8	10^{-16}	10^{-16}	10^{-16}

IR3: Summary

Standard (LU-based) IR in three precisions ($u_s = u_f$)

Half $\approx 10^{-4}$, Single $\approx 10^{-8}$, Double $\approx 10^{-16}$, Quad $\approx 10^{-34}$

	u_f	u	u_r	max $\kappa_\infty(A)$	Backward error norm	Backward error comp	Forward error
LP fact.	H	S	S	10^4	10^{-8}	10^{-8}	$\mathrm{cond}(A,x) \cdot 10^{-8}$
New	H	S	D	10^4	10^{-8}	10^{-8}	10^{-8}
LP fact.	H	D	D	10^4	10^{-16}	10^{-16}	$\mathrm{cond}(A,x) \cdot 10^{-16}$
New	H	D	Q	10^4	10^{-16}	10^{-16}	10^{-16}
Fixed	S	S	S	10^8	10^{-8}	10^{-8}	$\mathrm{cond}(A,x) \cdot 10^{-8}$
Trad.	S	S	D	10^8	10^{-8}	10^{-8}	10^{-8}
LP fact.	S	D	D	10^8	10^{-16}	10^{-16}	$\mathrm{cond}(A,x) \cdot 10^{-16}$
New	S	D	Q	10^8	10^{-16}	10^{-16}	10^{-16}

$\Rightarrow$ Benefit of IR3 vs. "LP fact.": no $\mathrm{cond}(A,x)$ term in forward error

IR3: Summary

Standard (LU-based) IR in three precisions ($u_s = u_f$)

Half $\approx 10^{-4}$, Single $\approx 10^{-8}$, Double $\approx 10^{-16}$, Quad $\approx 10^{-34}$

	u_f	u	u_r	max $\kappa_\infty(A)$	Backward error norm	Backward error comp	Forward error
LP fact.	H	S	S	10^4	10^{-8}	10^{-8}	$\text{cond}(A,x) \cdot 10^{-8}$
New	H	S	D	10^4	10^{-8}	10^{-8}	10^{-8}
LP fact.	H	D	D	10^4	10^{-16}	10^{-16}	$\text{cond}(A,x) \cdot 10^{-16}$
New	H	D	Q	10^4	10^{-16}	10^{-16}	10^{-16}
Fixed	S	S	S	10^8	10^{-8}	10^{-8}	$\text{cond}(A,x) \cdot 10^{-8}$
Trad.	S	S	D	10^8	10^{-8}	10^{-8}	10^{-8}
LP fact.	S	D	D	10^8	10^{-16}	10^{-16}	$\text{cond}(A,x) \cdot 10^{-16}$
New	S	D	Q	10^8	10^{-16}	10^{-16}	10^{-16}

$\Rightarrow$ Benefit of IR3 vs. traditional IR: As long as $\kappa_\infty(A) \leq 10^4$, can use lower precision factorization w/no loss of accuracy!

GMRES-IR: Summary

GMRES-IR: Solve for d_i via GMRES on $U^{-1}L^{-1}Ad_i = U^{-1}L^{-1}r_i$

GMRES-based IR in three precisions ($u_s = u$)

	u_f	u	u_r	max $\kappa_\infty(A)$	Backward error: norm	Backward error: comp	Forward error
LU-IR	H	S	D	10^4	10^{-8}	10^{-8}	10^{-8}
GMRES-IR	H	S	D	10^8	10^{-8}	10^{-8}	10^{-8}
LU-IR	S	D	Q	10^8	10^{-16}	10^{-16}	10^{-16}
GMRES-IR	S	D	Q	10^{16}	10^{-16}	10^{-16}	10^{-16}
LU-IR	H	D	Q	10^4	10^{-16}	10^{-16}	10^{-16}
GMRES-IR	H	D	Q	10^{12}	10^{-16}	10^{-16}	10^{-16}

$\Rightarrow$With GMRES-IR, lower precision factorization will work for higher $\kappa_\infty(A)$

GMRES-IR: Summary

GMRES-IR: Solve for d_i via GMRES on $U^{-1}L^{-1}Ad_i = U^{-1}L^{-1}r_i$

GMRES-based IR in three precisions ($\boldsymbol{u_s} = \boldsymbol{u}$)

	$\boldsymbol{u_f}$	$\boldsymbol{u}$	$\boldsymbol{u_r}$	max $\kappa_\infty(A)$	Backward error norm	Backward error comp	Forward error
LU-IR	H	S	D	10^4	10^{-8}	10^{-8}	10^{-8}
GMRES-IR	H	S	D	10^8	10^{-8}	10^{-8}	10^{-8}
LU-IR	S	D	Q	10^8	10^{-16}	10^{-16}	10^{-16}
GMRES-IR	S	D	Q	10^{16}	10^{-16}	10^{-16}	10^{-16}
LU-IR	H	D	Q	10^4	10^{-16}	10^{-16}	10^{-16}
GMRES-IR	H	D	Q	10^{12}	10^{-16}	10^{-16}	10^{-16}

→ $\kappa_\infty(A) \le \boldsymbol{u}^{-1/2}\, \boldsymbol{u_f}^{-1}$

⇒With GMRES-IR, lower precision factorization will work for higher $\kappa_\infty(A)$

GMRES-IR: Summary

GMRES-IR: Solve for d_i via GMRES on $U^{-1}L^{-1}Ad_i = U^{-1}L^{-1}r_i$

GMRES-based IR in three precisions ($u_s = u$)

	u_f	u	u_r	max $\kappa_\infty(A)$	Backward error: norm	Backward error: comp	Forward error
LU-IR	H	S	D	10^4	10^{-8}	10^{-8}	10^{-8}
GMRES-IR	H	S	D	10^8	10^{-8}	10^{-8}	10^{-8}
LU-IR	S	D	Q	10^8	10^{-16}	10^{-16}	10^{-16}
GMRES-IR	S	D	Q	10^{16}	10^{-16}	10^{-16}	10^{-16}
LU-IR	H	D	Q	10^4	10^{-16}	10^{-16}	10^{-16}
GMRES-IR	H	D	Q	10^{12}	10^{-16}	10^{-16}	10^{-16}

→ $\kappa_\infty(A) \leq u^{-1/2}\, u_f^{-1}$

⇒ As long as $\kappa_\infty(A) \leq 10^{12}$, can use half precision factorization and still obtain double precision accuracy!

GMRES-IR: Summary

GMRES-IR: Solve for d_i via GMRES on $U^{-1}L^{-1}Ad_i = U^{-1}L^{-1}r_i$

GMRES-based IR in three precisions ($u_s = u$)

	u_f	u	u_r	max $\kappa_\infty(A)$	Backward error norm	Backward error comp	Forward error
LU-IR	H	S	D	10^4	10^{-8}	10^{-8}	10^{-8}
GMRES-IR	H	S	D	10^8	10^{-8}	10^{-8}	10^{-8}
LU-IR	S	D	Q	10^8	10^{-16}	10^{-16}	10^{-16}
GMRES-IR	S	D	Q	10^{16}	10^{-16}	10^{-16}	10^{-16}
LU-IR	H	D	Q	10^4	10^{-16}	10^{-16}	10^{-16}
GMRES-IR	H	D	Q	10^{12}	10^{-16}	10^{-16}	10^{-16}

$\rightarrow \kappa_\infty(A) \leq u^{-1/2} u_f^{-1}$

$\Rightarrow$ As long as $\kappa_\infty(A) \leq 10^{12}$, can use half precision factorization and still obtain double precision accuracy!

Recent work: 5-precision GMRES-IR [Amestoy, et al., 2021]

GMRES-IR: Summary

GMRES-IR: Solve for d_i via GMRES on $U^{-1}L^{-1}Ad_i = U^{-1}L^{-1}r_i$

GMRES-based IR in three precisions ($u_s = u$)

	u_f	u	u_r	max $\kappa_\infty(A)$	Backward error norm	Backward error comp	Forward error
LU-IR	H	S	D	10^4	10^{-8}	10^{-8}	10^{-8}
GMRES-IR	H	S	D	10^8	10^{-8}	10^{-8}	10^{-8}
LU-IR	S	D	Q	10^8	10^{-16}	10^{-16}	10^{-16}
GMRES-IR	S	D	Q	10^{16}	10^{-16}	10^{-16}	10^{-16}
LU-IR	H	D	Q	10^4	10^{-16}	10^{-16}	10^{-16}
GMRES-IR	H	D	Q	10^{12}	10^{-16}	10^{-16}	10^{-16}

→ $\kappa_\infty(A) \leq u^{-1/2}\, u_f^{-1}$

⇒ As long as $\kappa_\infty(A) \leq 10^{12}$, can use half precision factorization and still obtain double precision accuracy!

Recent work: 5-precision GMRES-IR [Amestoy, et al., 2021]

→ $\kappa_\infty(A) \leq u^{-1/3}\, u_f^{-2/3}$

Extension: 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.

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

Extension: 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

Extension: Least Squares Problems

- 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 $(m+n)$:

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

Extension: Least Squares Problems

- 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":

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

Extension: Least Squares Problems

- 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 $(m+n)$:

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

- 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":

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

Extension: Least Squares Problems

- 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 $(m+n)$:

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

- Refinement proceeds as follows:

1. Compute "residuals"

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

2. Solve for corrections

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \Delta r_i \\ \Delta x_i \end{bmatrix} = \begin{bmatrix} f_i \\ g_i \end{bmatrix} \qquad \tilde{A} d_i = \tilde{r}_i$$

3. Update "solution":

$$\begin{bmatrix} r_{i+1} \\ x_{i+1} \end{bmatrix} = \begin{bmatrix} r_i \\ x_i \end{bmatrix} + \begin{bmatrix} \Delta r_i \\ \Delta x_i \end{bmatrix} \qquad \tilde{x}_{i+1} = \tilde{x}_i + d_i$$

Extension: Least Squares Problems

- 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 $(m+n)$:

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

- 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":

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

$$\tilde{r}_i = \tilde{b} - \tilde{A}\tilde{x}_i$$

$$\tilde{A}d_i = \tilde{r}_i$$

$$\tilde{x}_{i+1} = \tilde{x}_i + d_i$$

Results for 3-precision IR for linear systems **also applies to least squares problems [C., Higham, Pranesh, 2020]**

Extension: Multistage Mixed Precision IR

- Many different variants of mixed precision IR
  - "standard IR" (SIR): LU solves
  - SGMRES-IR: preconditioned GMRES entirely in working precision
  - GMRES-IR: preconditioned GMRES with extra precision

cost, reliability

Extension: Multistage Mixed Precision IR

- Many different variants of mixed precision IR
  - "standard IR" (SIR): LU solves
  - SGMRES-IR: preconditioned GMRES entirely in working precision
  - GMRES-IR: preconditioned GMRES with extra precision

cost, reliability

- Problem: constraints for convergence often overly strict in practice
  - Hard to pick the best variant

Extension: Multistage Mixed Precision IR

- Many different variants of mixed precision IR
  - "standard IR" (SIR): LU solves
  - SGMRES-IR: preconditioned GMRES entirely in working precision
  - GMRES-IR: preconditioned GMRES with extra precision

cost, reliability ↓

- Problem: constraints for convergence often overly strict in practice
  - Hard to pick the best variant

- Opportunity: typically implementations increase precisions if lack of convergence detected
  - *Requires recomputing the expensive LU factorization* in higher precision
  - Before resorting to increasing precisions, we can first try using a better inner solver with the existing LU factors!

Extension: Multistage Mixed Precision IR

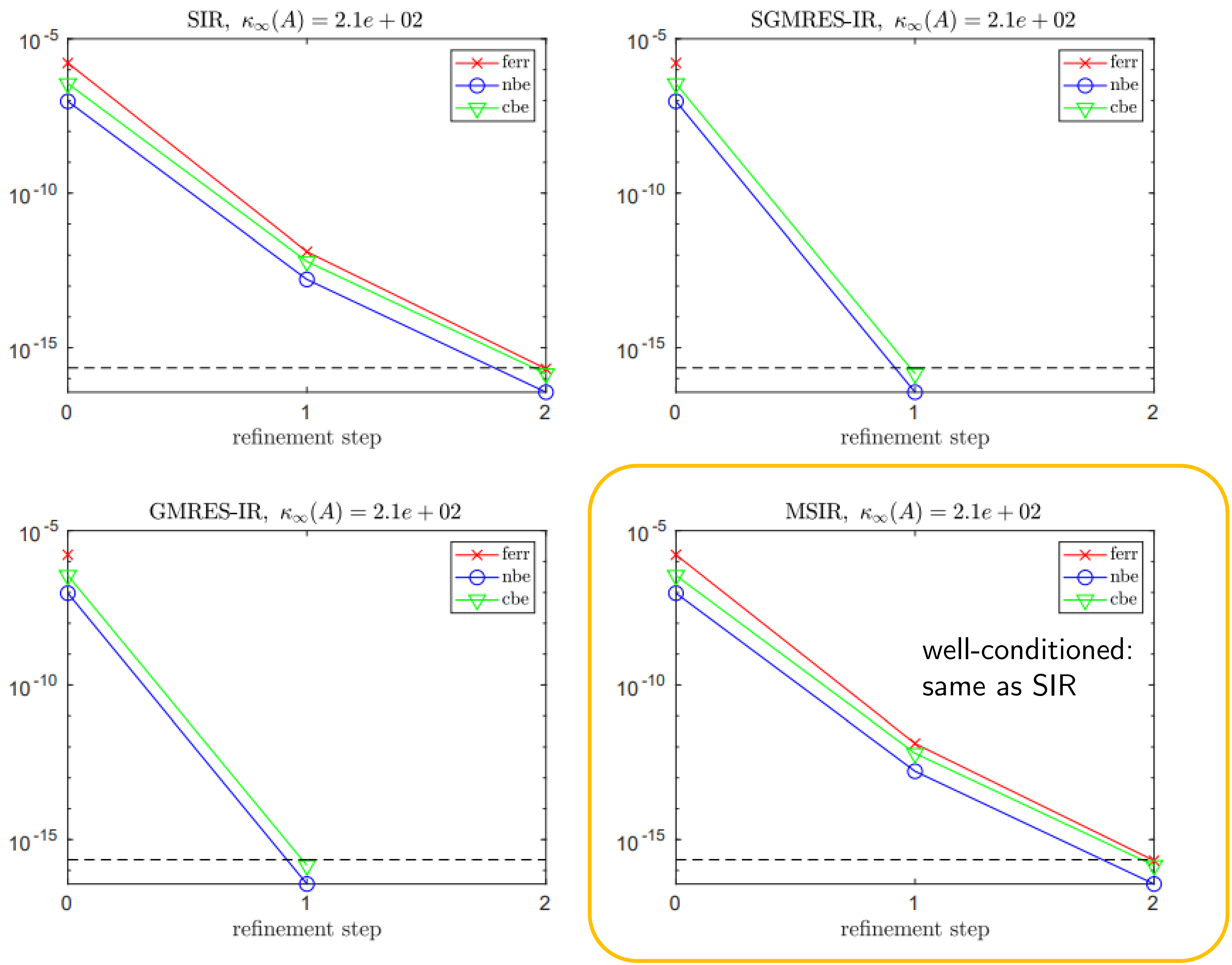
- Many different variants of mixed precision IR
  - "standard IR" (SIR): LU solves
  - SGMRES-IR: preconditioned GMRES entirely in working precision
  - GMRES-IR: preconditioned GMRES with extra precision

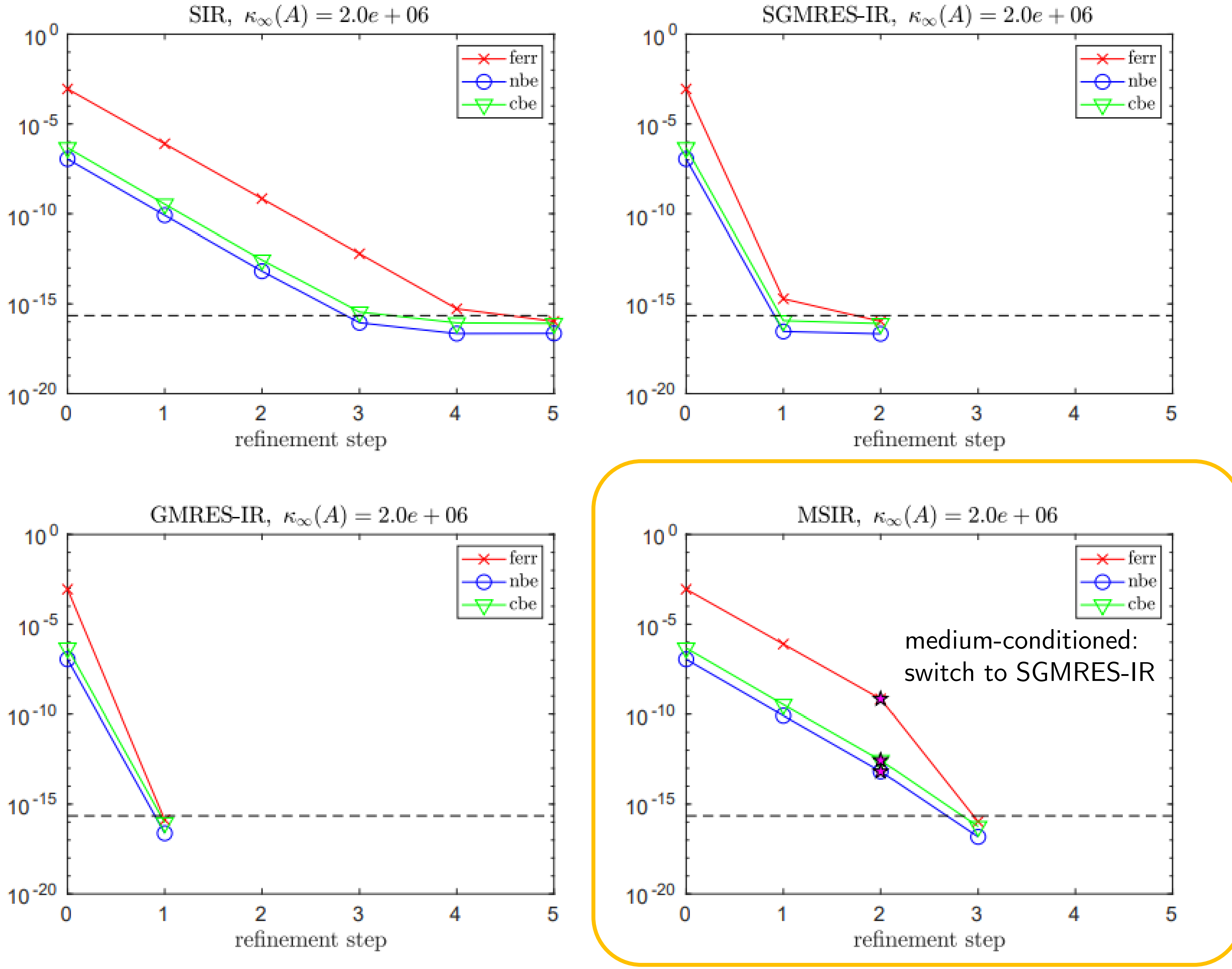
cost, reliability ↓

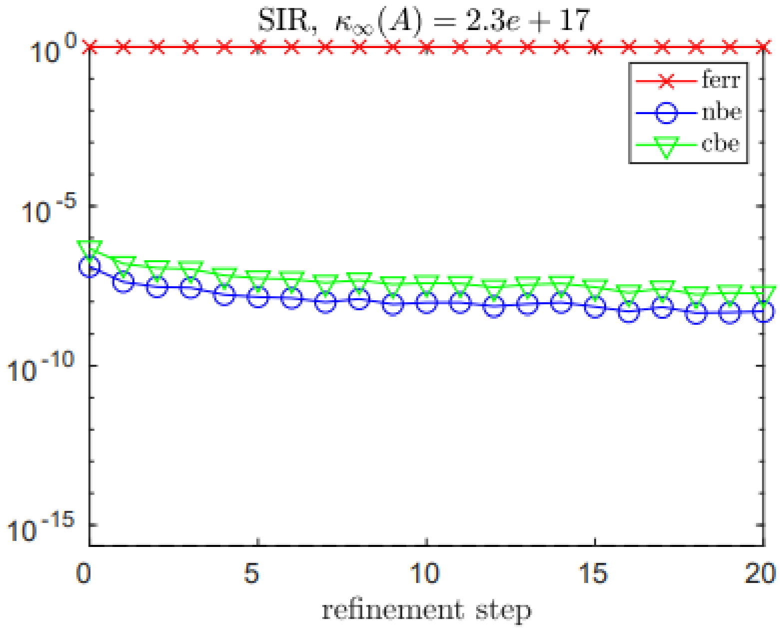
- Problem: constraints for convergence often overly strict in practice
  - Hard to pick the best variant

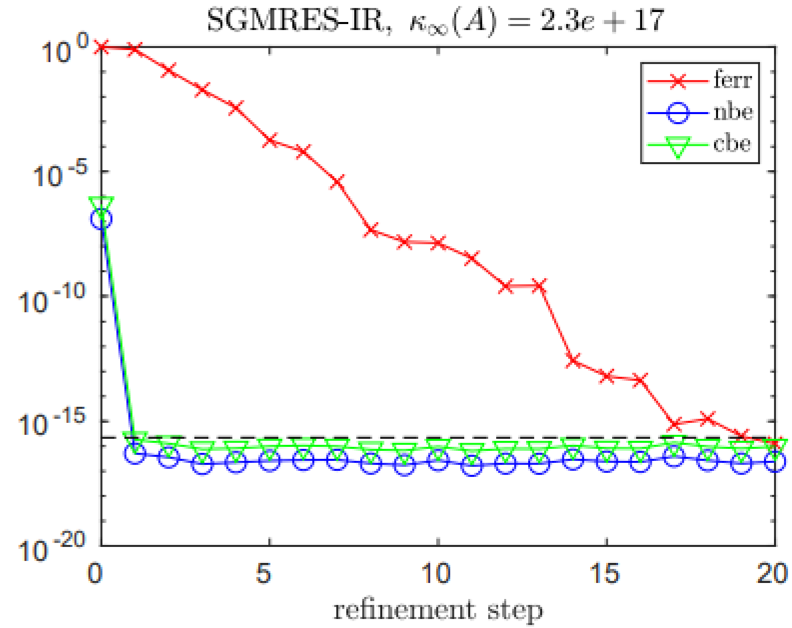
- Opportunity: typically implementations increase precisions if lack of convergence detected
  - *Requires recomputing the expensive LU factorization* in higher precision
  - Before resorting to increasing precisions, we can first try using a better inner solver with the existing LU factors!

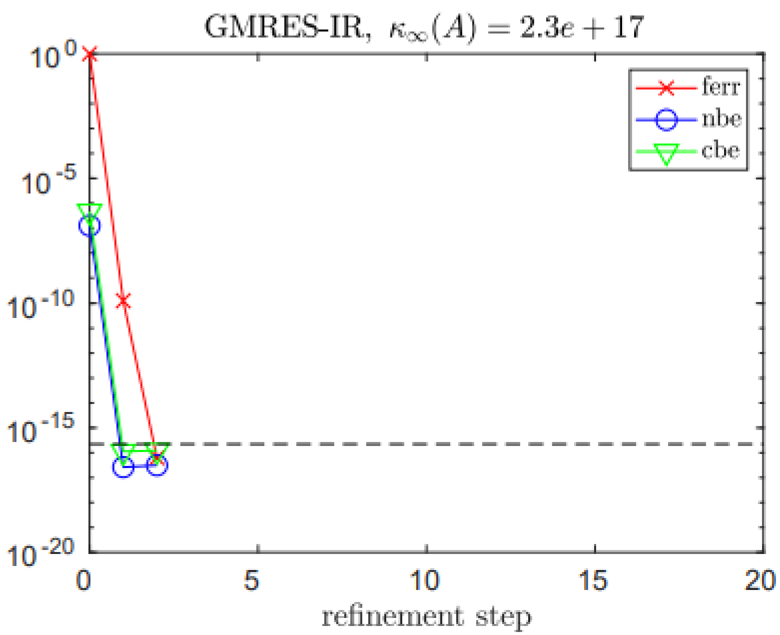
→ **Multistage Iterative Refinement (MSIR)** [Oktay, C., NLAA, 2022]

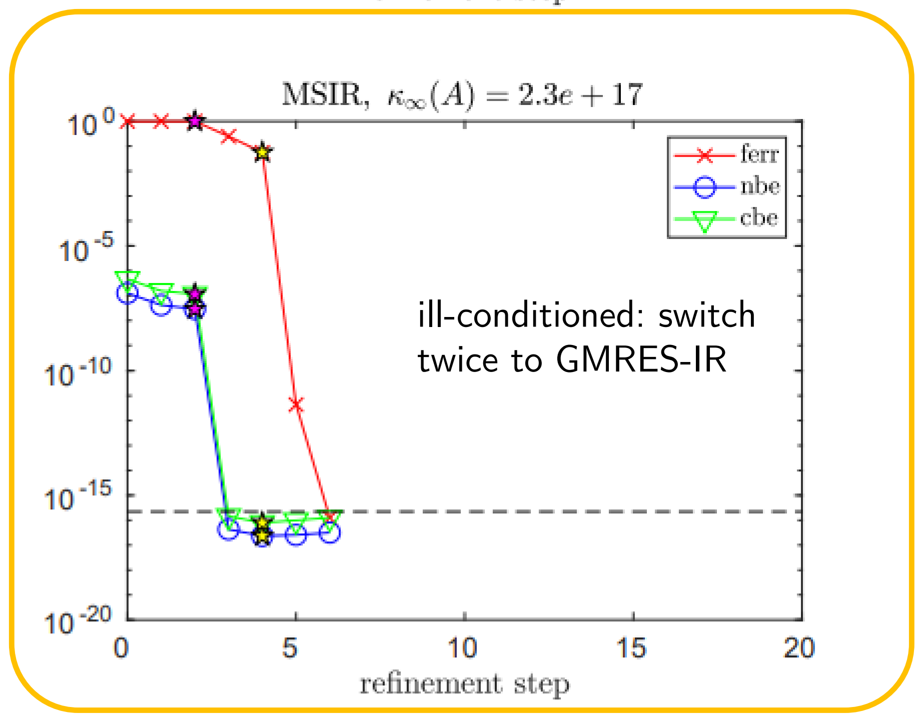

SIR, $\kappa_\infty(A) = 2.1e + 02$
SGMRES-IR, $\kappa_\infty(A) = 2.1e + 02$
GMRES-IR, $\kappa_\infty(A) = 2.1e + 02$
MSIR, $\kappa_\infty(A) = 2.1e + 02$
ferr
nbe
cbe
10^{-5}
10^{-10}
10^{-15}
0
1
2
refinement step
well-conditioned:
same as SIR


SIR, $\kappa_\infty(A) = 2.0e+06$
SGMRES-IR, $\kappa_\infty(A) = 2.0e+06$
GMRES-IR, $\kappa_\infty(A) = 2.0e+06$
MSIR, $\kappa_\infty(A) = 2.0e+06$
ferr
nbe
cbe
10^{0}
10^{-5}
10^{-10}
10^{-15}
10^{-20}
0
1
2
3
4
5
refinement step
medium-conditioned:
switch to SGMRES-IR


SIR, $\kappa_\infty(A) = 2.3e + 17$
ferr
nbe
cbe
refinement step


SGMRES-IR, $\kappa_\infty(A) = 2.3e + 17$
ferr
nbe
cbe
refinement step


GMRES-IR, $\kappa_\infty(A) = 2.3e + 17$
ferr
nbe
cbe
refinement step


MSIR, $\kappa_\infty(A) = 2.3e + 17$
ferr
nbe
cbe
ill-conditioned: switch
twice to GMRES-IR
refinement step

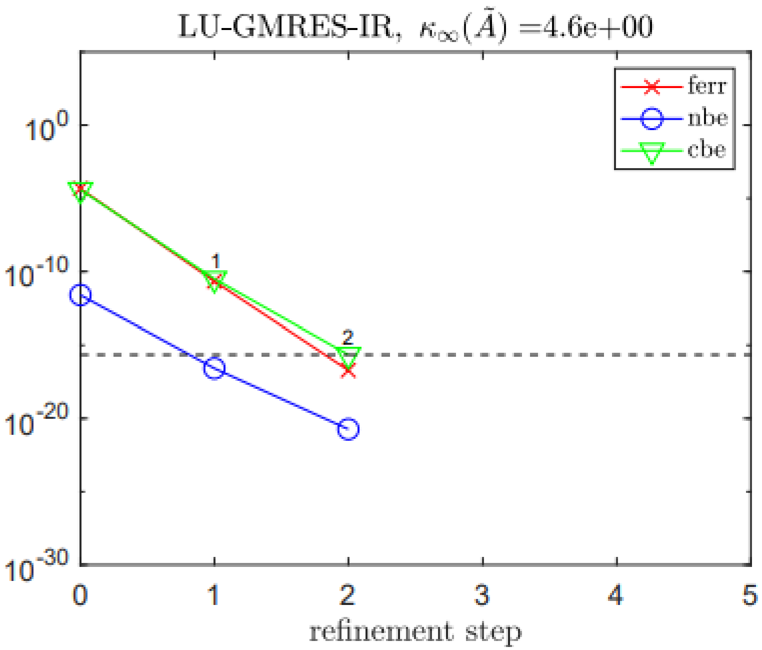
Extension: SPAI-GMRES-IR

- Existing analyses of GMRES-IR assume we use full LU factors
- In practice, often want to use sparse preconditioners (ILU, SPAI, etc.)
- [C., Khan, arXiv:2202.10204, 2022]: analysis of GMRES-IR with SPAI preconditioning)
  - Convergence to limiting accuracy as long as $n u_f \mathit{cond}(A^T) \lesssim n\varepsilon \lesssim u^{-1/2}$

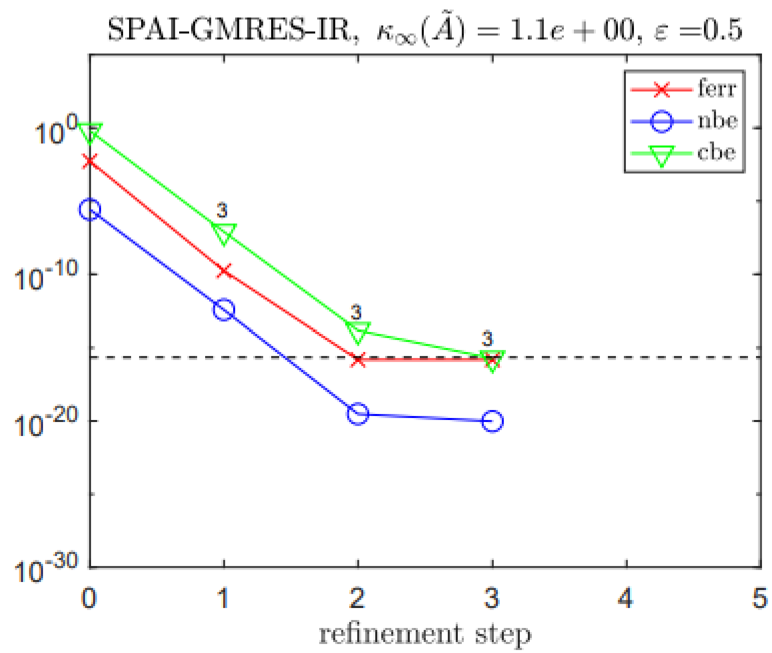
Extension: SPAI-GMRES-IR

- Existing analyses of GMRES-IR assume we use full LU factors
- In practice, often want to use sparse preconditioners (ILU, SPAI, etc.)
- [C., Khan, arXiv:2202.10204, 2022]: analysis of GMRES-IR with SPAI preconditioning)
  - Convergence to limiting accuracy as long as $nu_f cond(A^T) \lesssim n\varepsilon \lesssim u^{-1/2}$

matrix: steam1, (u_f, u, u_r) = (single, double, quad)



nnz$(L+U)$ = 21,657



nnz(M) = 2,248

The rise of multiprecision hardware

- Future machines will support a range of precisions: quarter, half, single, double, quad

The rise of multiprecision hardware

- Future machines will support a range of precisions: quarter, half, single, double, quad
- New, non-IEEE compliant floating point formats will appear in commercially-available hardware
    - e.g., bfloat16 (truncated 16-bit version of single precision), posits

The rise of multiprecision hardware

- Future machines will support a range of precisions: quarter, half, single, double, quad
- New, non-IEEE compliant floating point formats will appear in commercially-available hardware
  - e.g., bfloat16 (truncated 16-bit version of single precision), posits
- Lower-precision arithmetic is faster and more energy efficient, but the potential for its use depends heavily on the particular problem and algorithm

The rise of multiprecision hardware

- Future machines will support a range of precisions: quarter, half, single, double, quad

- New, non-IEEE compliant floating point formats will appear in commercially-available hardware
    - e.g., bfloat16 (truncated 16-bit version of single precision), posits

- Lower-precision arithmetic is faster and more energy efficient, but the potential for its use depends heavily on the particular problem and algorithm

- As numerical analysts, we must determine when and where we can exploit lower-precision hardware to improve performance

Thank you!

carson@karlin.mff.cuni.cz

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