Exploiting Mixed Precision in Numerical Linear Algebra

Erin C. Carson

Faculty of Mathematics and Physics, Charles University

May 4, 2022

47th Spring Lecture Series

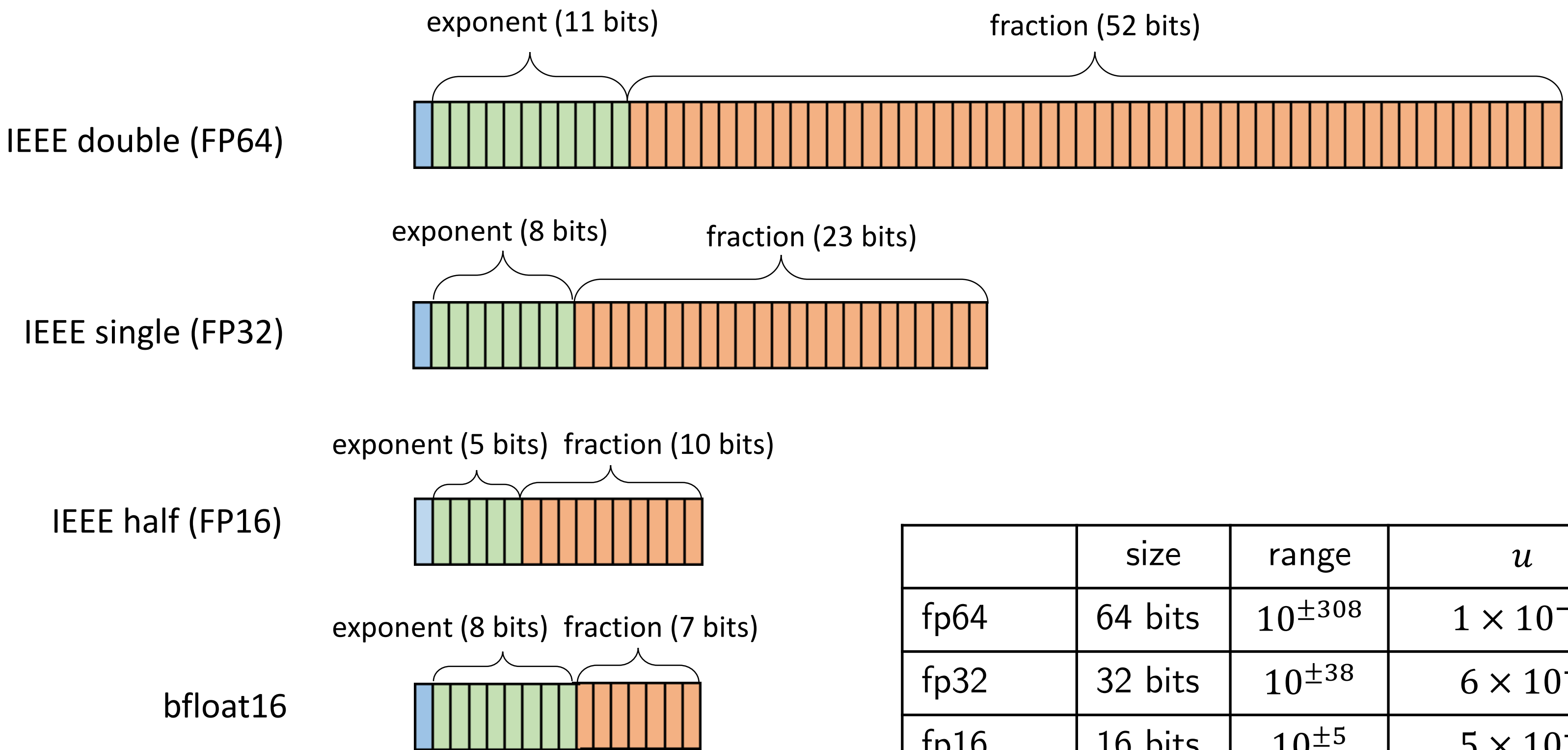
University of Arkansas

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

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



	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}

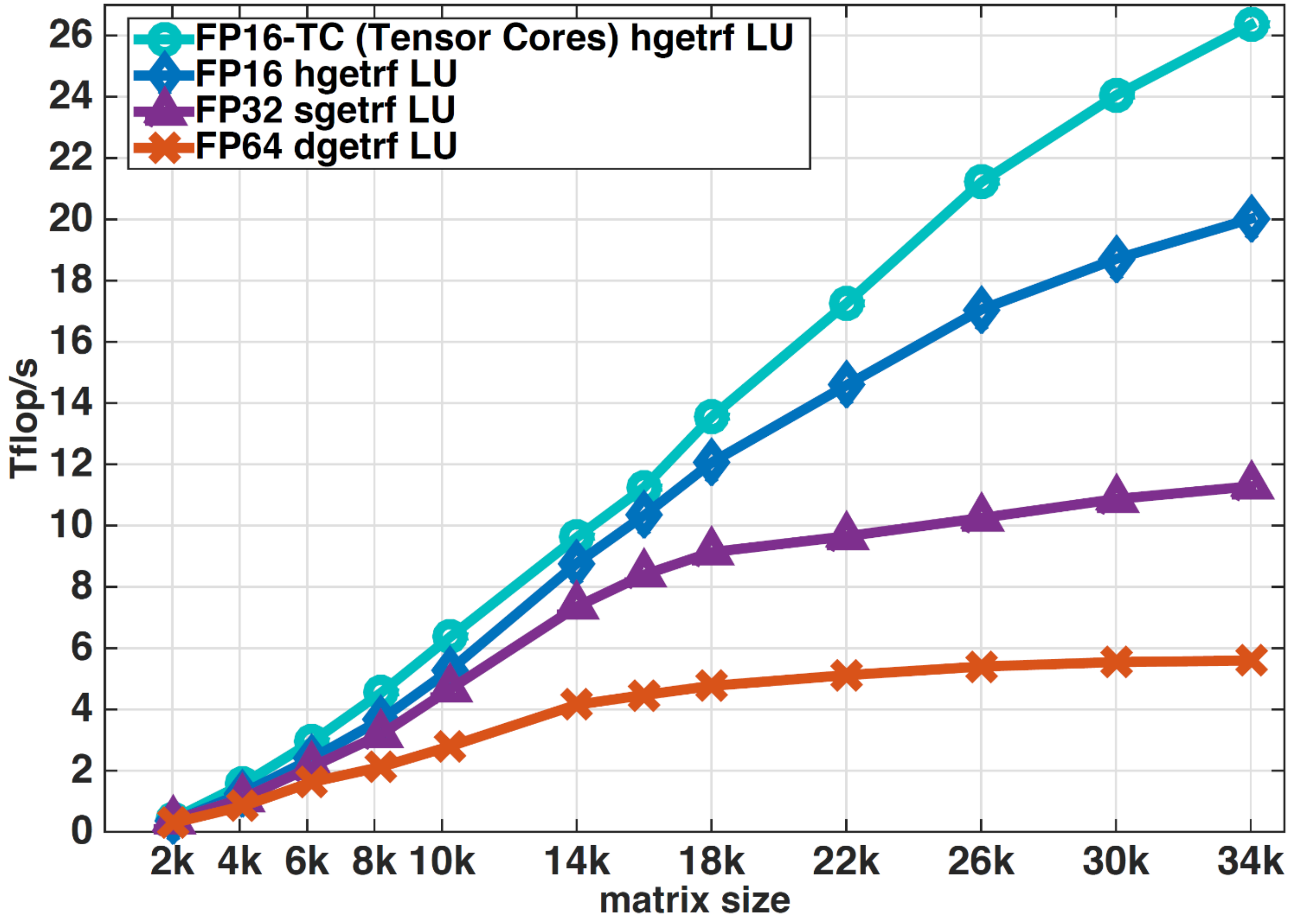
Hardware Support for Multiprecision Computation

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)

Performance of LU factorization on an NVIDIA V100 GPU



[Haidar, Tomov, Dongarra, Higham, 2018]

Mixed Precision Capabilities on Supercomputers

From TOP500:

November 2021

	Accelerator/CP Family	Count	System Share (%)	Rmax (GFlops)	Rpeak (GFlops)	Cores
1	NVIDIA Volta	84	16.8	608,245,890	1,015,712,384	11,475,992
2	NVIDIA Ampere	43	8.6	527,074,700	749,271,060	5,072,700
3	NVIDIA Pascal	8	1.6	54,569,640	80,911,013	1,080,788

27

Mixed Precision Capabilities on Supercomputers

From TOP500:

November 2021

	Accelerator/CP Family	Count	System Share (%)	Rmax (GFlops)	Rpeak (GFlops)	Cores
1	NVIDIA Volta	84	16.8	608,245,890	1,015,712,384	11,475,992
2	NVIDIA Ampere	43	8.6	527,074,700	749,271,060	5,072,700
3	NVIDIA Pascal	8	1.6	54,569,640	80,911,013	1,080,788

27

June 2019

	Accelerator/CP Family	Count	System Share (%)	Rmax (GFlops)	Rpeak (GFlops)	Cores
1	NVIDIA Pascal	61	12.2	106,025,166	179,951,012	2,738,356
3	NVIDIA Volta	12	2.4	224,559,400	360,593,742	4,488,720

14.6

"Exascale": An exaflop of what?

- When will victory be declared?
  - When a supercomputer reaches exaflop performance on the HPL (LINPACK) benchmark (TOP500)
    - Solving dense $Ax = b$ using Gaussian elimination with partial pivoting in double precision (FP64)

"Exascale": An exaflop of what?

- When will victory be declared?
  - When a supercomputer reaches exaflop performance on the HPL (LINPACK) benchmark (TOP500)
    - Solving dense $Ax = b$ using Gaussian elimination with partial pivoting in double precision (FP64)

- HPL benchmark is typically a compute-bound problem ("BLAS-3")
- Not a good indication of performance for a large number of applications!
  - Lots of remaining work even after exascale performance is achieved
  - Has led to incorporation of other benchmarks into the TOP500 ranking
    - e.g., HPCG: Solving sparse $Ax = b$ iteratively using the conjugate gradient method

"Exascale": An exaflop of what?

- HPL doesn't make use of modern mixed precision hardware

- We can *already* achieve "exaflop" performance today if we allow for mixed precision computations

https://www.olcf.ornl.gov/2018/06/08/genomics-code-exceeds-exaops-on-summit-supercomputer/

"Exascale": An exaflop of what?

- HPL doesn't make use of modern mixed precision hardware

- We can *already* achieve "exaflop" performance today if we allow for mixed precision computations

https://www.olcf.ornl.gov/2018/06/08/genomics-code-exceeds-exaops-on-summit-supercomputer/

=>HPL-AI: A new mixed precision benchmark

HPL-AI Benchmark

- Highlights confluence of HPC+AI workloads
  - Like HPL, solves dense Ax=b, results still to double precision accuracy
  - Achieves this via mixed-precision iterative refinement
    - may be implemented in a way that takes advantage of the current and upcoming devices for accelerating AI workloads

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

More information: https://icl.bitbucket.io/hpl-ai/
Reference implementation: https://bitbucket.org/icl/hpl-ai/src/

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

More information: https://icl.bitbucket.io/hpl-ai/
Reference implementation: https://bitbucket.org/icl/hpl-ai/src/

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

More information: https://icl.bitbucket.io/hpl-ai/
Reference implementation: https://bitbucket.org/icl/hpl-ai/src/

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]

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

Challenges of low precision

- 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]

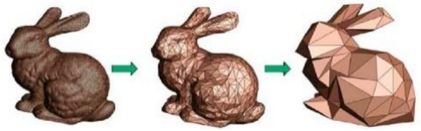
- Larger unit roundoff
  - Lose something small when storing: $fl(x) = x(1+\delta), \;\; |\delta| \le u$
  - Lose something small when computing: $fl(x \text{ op } y) = (x \text{ op } y)(1+\delta), \;\; |\delta| \le u$

Does it matter?

Inexact computations

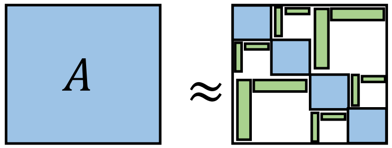
- In real computations we have many sources of inexactness
  - Imperfect data, measurement error
  - Modeling error, discretization error
  - Intentional approximation to improve performance
    - Reduced models, Low-rank representations, sparsification, randomization

Model Reduction



[Schilders, van der Vorst, Rommes, 2008]

Low-rank (hierarchical) approximation



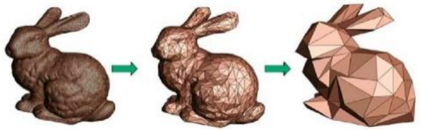
Sparsification, Randomized algorithms

[Sinha, 2018]

Inexact computations

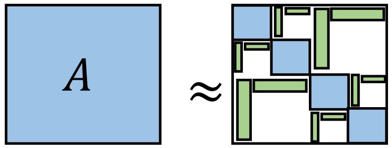
- In real computations we have many sources of inexactness
  - Imperfect data, measurement error
  - Modeling error, discretization error
  - Intentional approximation to improve performance
    - Reduced models, Low-rank representations, sparsification, randomization

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?

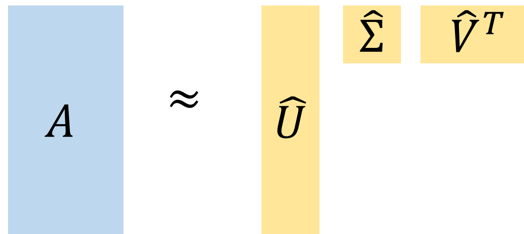
Example: Randomized Algorithms

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

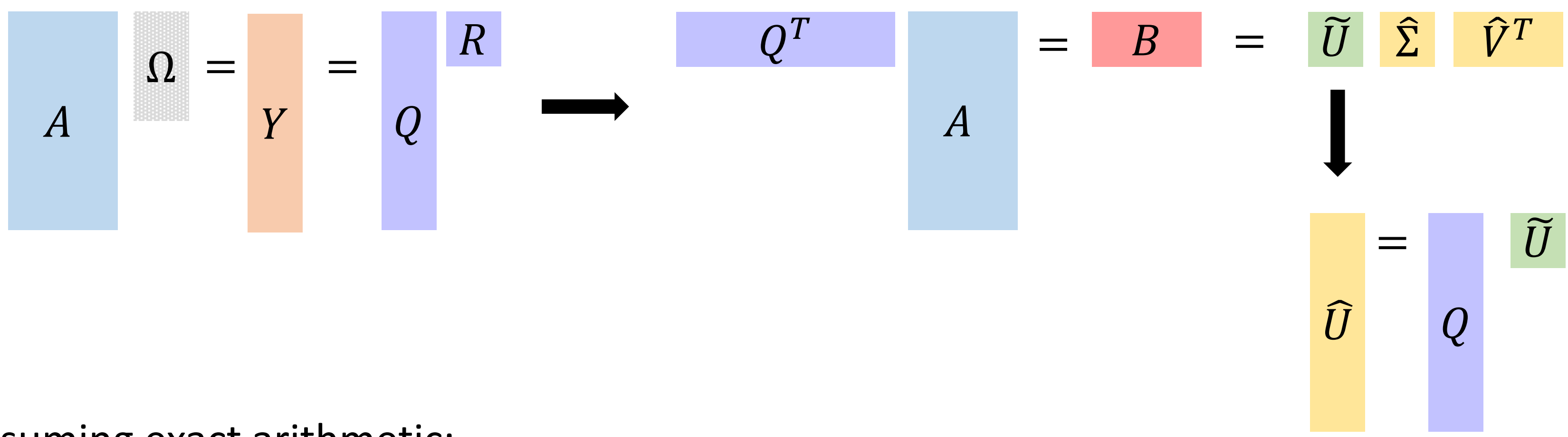
$$A \approx \hat{U} \hat{\Sigma} \hat{V}^T$$

Example: Randomized Algorithms

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



- Randomized SVD:



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

What happens in finite precision?

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

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

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

$[\hat{U}, \hat{S}, \hat{V}] \quad = \mathrm{rsvd}(A)$: 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

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

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

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

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

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

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

$[\hat{U},\hat{S},\hat{V}] \quad = \mathrm{rsvd}(A)$: 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

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

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

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

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

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

Mode 1: one large singular value

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

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

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

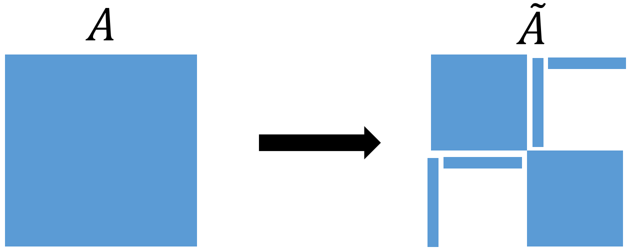
$\|A - \hat{U}_h\hat{S}_h\hat{V}_h^{\,T}\|_2 = 1.11\text{e-}05$

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

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

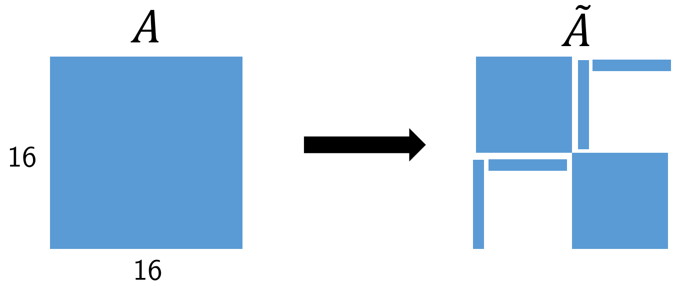
- [Higham, Mary, 2019]: block low-rank LU factorization preconditioner that exploits numerically low-rank structure of the error for LU computed in low precision
- [Higham, Mary, 2019]: Interplay of roundoff error and approximation error in solving block low-rank linear systems using LU
- [Buttari, et 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!

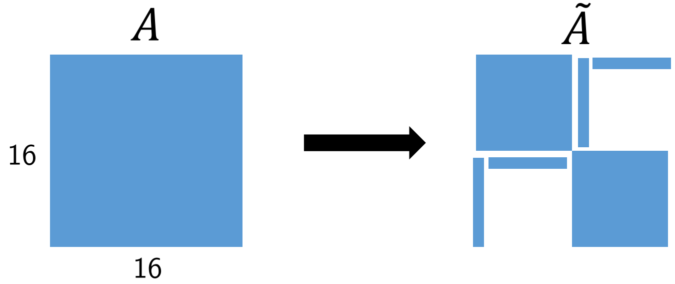


Example: Low-Rank Approximation

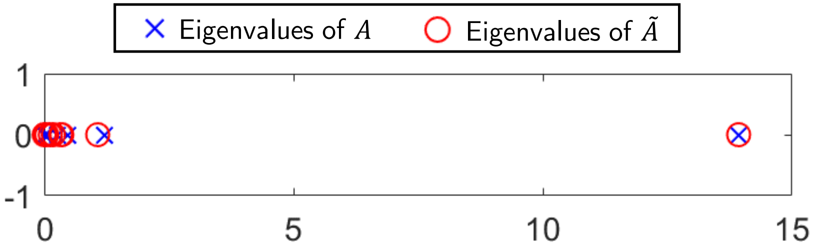
Inverse multiquadratic kernel:

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

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



Exact arithmetic SVD:

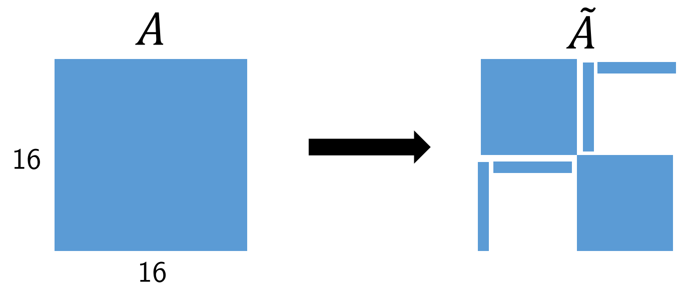


Example: Low-Rank Approximation

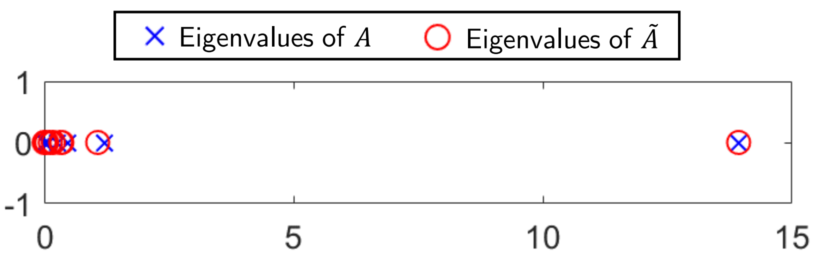
Inverse multiquadratic kernel:

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

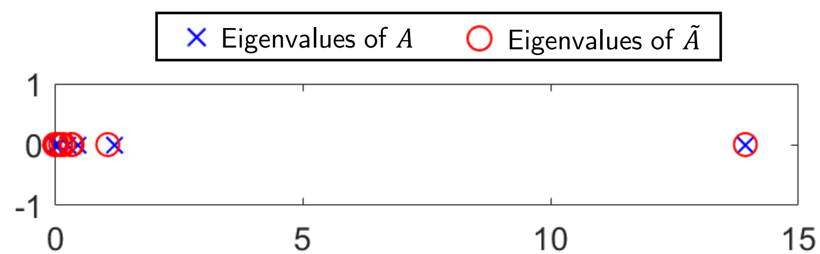
A is SPD. Low-rank approximation of A should also be SPD!



Exact arithmetic SVD:



Half precision SVD:

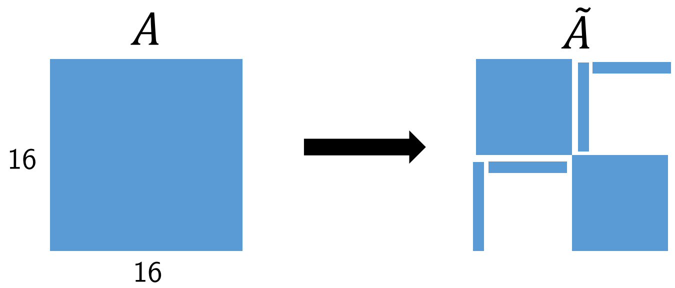


Example: Low-Rank Approximation

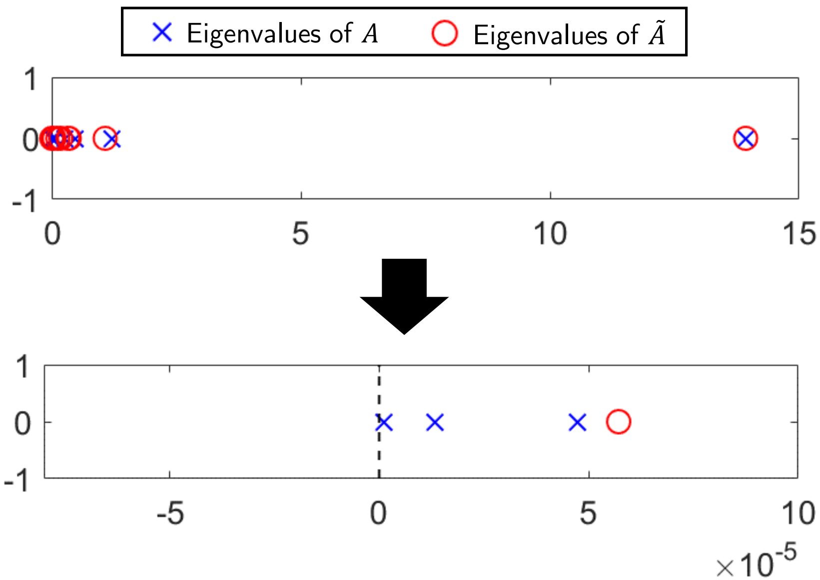
Inverse multiquadratic kernel:

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

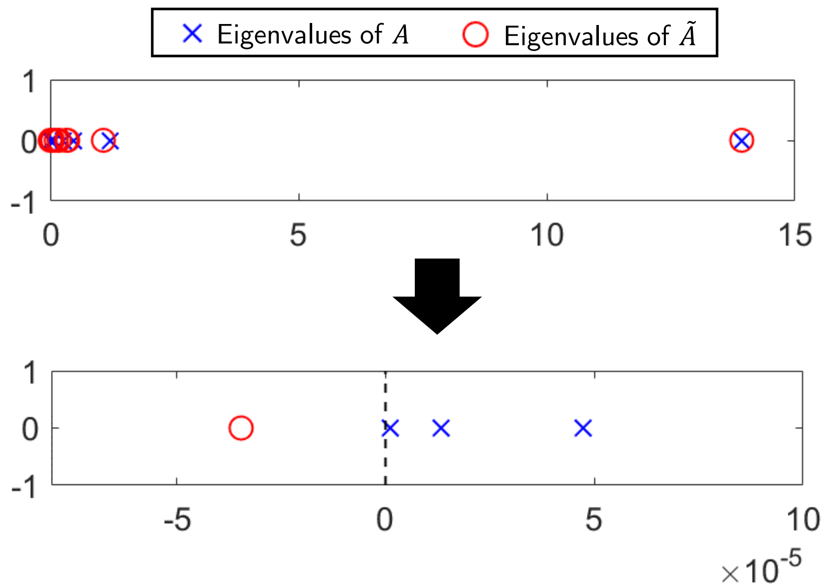
A is SPD. Low-rank approximation of A should also be SPD!



Exact arithmetic SVD:



Half precision SVD:

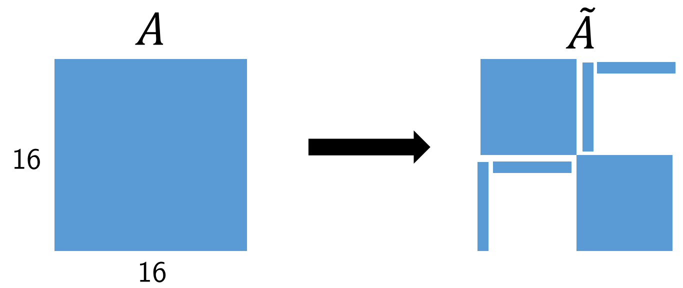


Example: Low-Rank Approximation

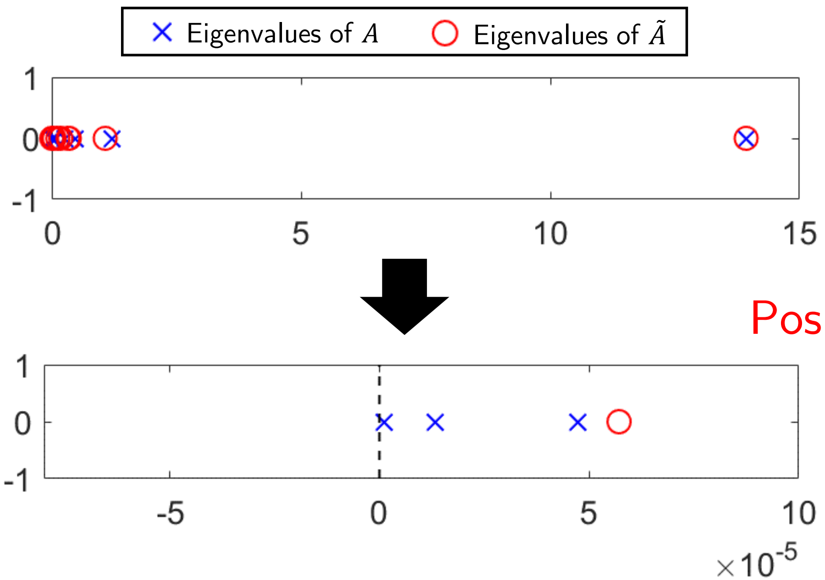
Inverse multiquadratic kernel:

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

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

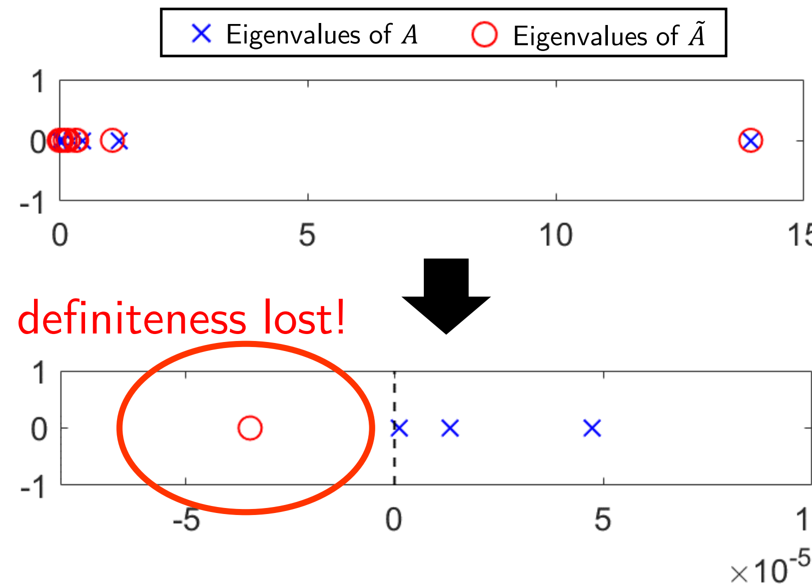


Exact arithmetic SVD:



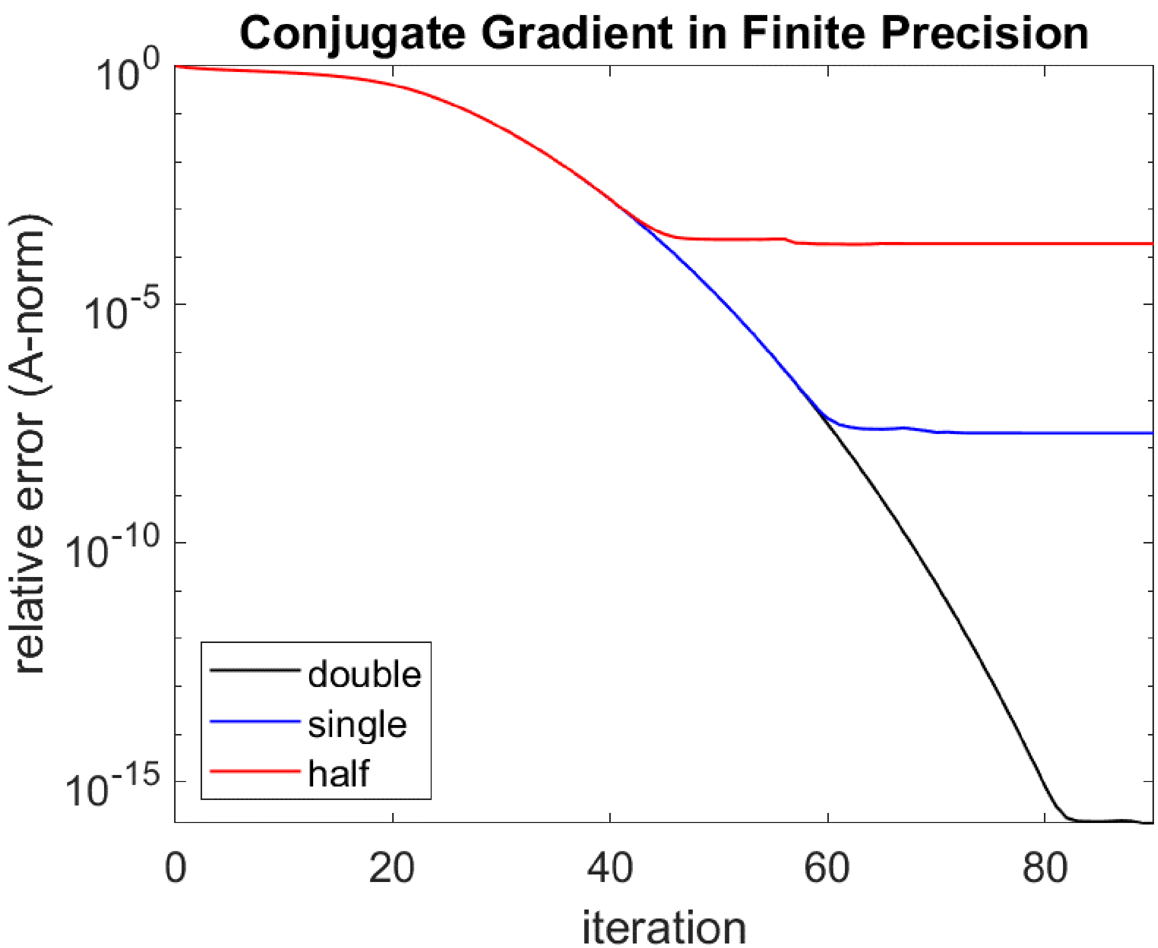
Half precision SVD:

Positive definiteness lost!



Example: Iterative Methods

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

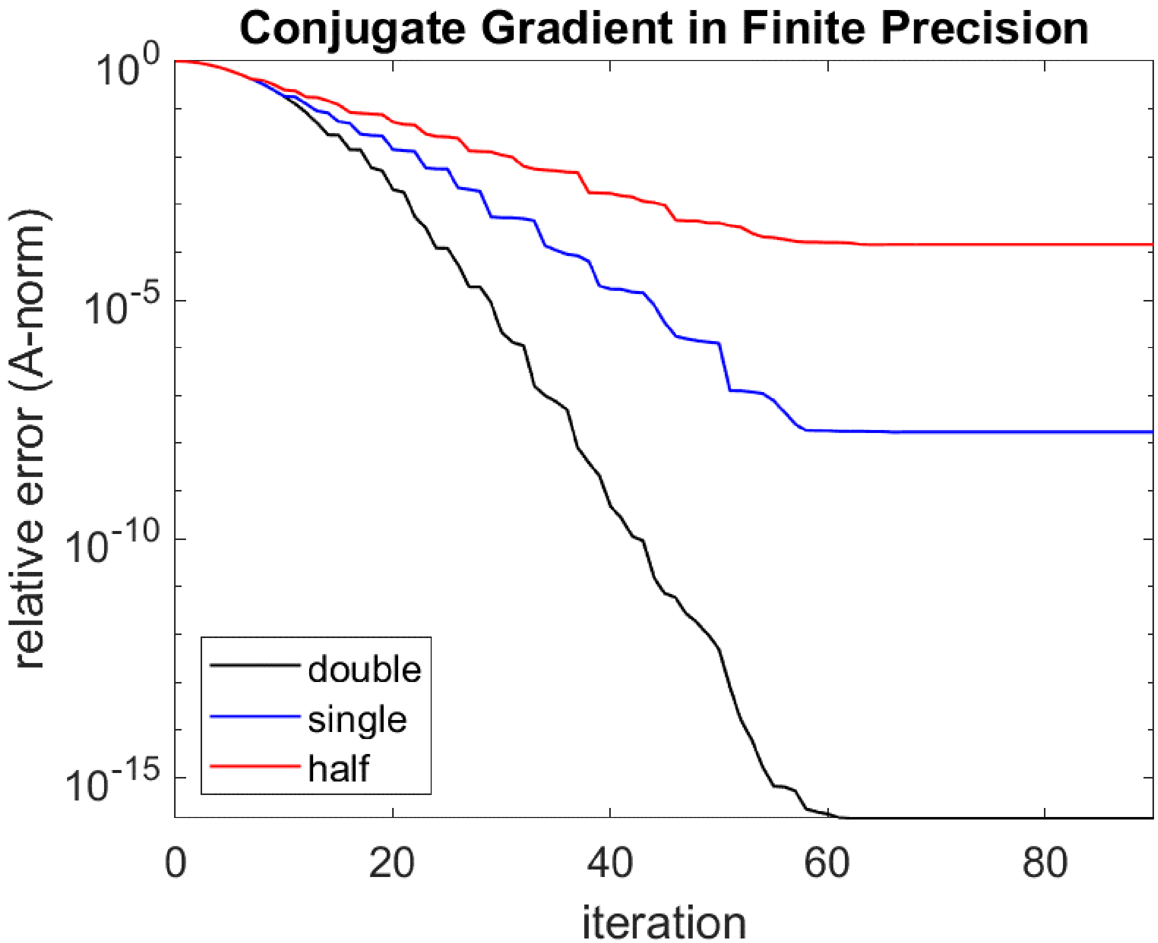


Example: Iterative Methods

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

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

```
b = 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 $i = 0 : \mathrm{maxit}$

$r_i = b - Ax_i$ (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

Existing analyses only support at most two precisions

Can we combine the performance benefits of low-precision factorization IR with the accuracy of traditional IR?

Iterative Refinement in 3 Precisions

Existing analyses only support at most two precisions

Can we combine the performance benefits of low-precision factorization IR with the accuracy of traditional IR?

⇒ **3-precision iterative refinement**

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

$$u_f \geq u \geq u_r$$

Iterative Refinement in 3 Precisions

Existing analyses only support at most two precisions

Can we combine the performance benefits of low-precision factorization IR with the accuracy of traditional IR?

$\Rightarrow$ **3-precision iterative refinement**

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

$$u_f \geq u \geq u_r$$

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

[C. and Higham, SIAM SISC 40(2), 2018]

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

(and **improves** upon existing analyses in some cases)

Iterative Refinement in 3 Precisions

Existing analyses only support at most two precisions

Can we combine the performance benefits of low-precision factorization IR with the accuracy of traditional IR?

⇒ **3-precision iterative refinement**

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

$$u_f \geq u \geq u_r$$

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

[C. and Higham, SIAM SISC 40(2), 2018]

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

(and **improves** upon existing analyses in some cases)

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

Key Aspects of Analysis I

Obtain tighter upper bounds:

Typical bounds used in analysis: $\|A(x - \hat{x}_i)\|_\infty \leq \|A\|_\infty \|x - \hat{x}_i\|_\infty$

Key Aspects of Analysis I

Obtain tighter upper bounds:

Typical bounds used in analysis: $\|A(x - \hat{x}_i)\|_\infty \leq \|A\|_\infty \|x - \hat{x}_i\|_\infty$

Define μ_i: $\|A(x - \hat{x}_i)\|_\infty = \mu_i \|A\|_\infty \|x - \hat{x}_i\|_\infty$

Key Aspects of Analysis I

Obtain tighter upper bounds:

Typical bounds used in analysis: $\|A(x-\hat{x}_i)\|_\infty \leq \|A\|_\infty \|x-\hat{x}_i\|_\infty$

Define μ_i: $\|A(x-\hat{x}_i)\|_\infty = \mu_i \|A\|_\infty \|x-\hat{x}_i\|_\infty$

For a stable refinement scheme, in early stages we expect

$$\frac{\|r_i\|}{\|A\|\|\hat{x}_i\|} \approx u \ll \frac{\|x-\hat{x}_i\|}{\|x\|} \longrightarrow \boxed{\mu_i \ll 1}$$

Key Aspects of Analysis I

Obtain tighter upper bounds:

Typical bounds used in analysis: $\|A(x-\hat{x}_i)\|_\infty \leq \|A\|_\infty \|x-\hat{x}_i\|_\infty$

Define μ_i: $\|A(x-\hat{x}_i)\|_\infty = \mu_i \|A\|_\infty \|x-\hat{x}_i\|_\infty$

For a stable refinement scheme, in early stages we expect

$$\frac{\|r_i\|}{\|A\|\|\hat{x}_i\|} \approx u \ll \frac{\|x-\hat{x}_i\|}{\|x\|} \longrightarrow \mu_i \ll 1$$

But close to convergence,

$$\|r_i\| \approx \|A\|\|x-\hat{x}_i\| \longrightarrow \mu_i \approx 1$$

Key Aspects of Analysis II

Allow for general solver:

Let u_s be the *effective precision* of the solve, with $u \leq u_s \leq u_f$

Key Aspects of Analysis II

Allow for general solver:

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

Assume computed solution $\hat{d}_i$ to $Ad_i = \hat{r}_i$ satisfies:

1. $\hat{d}_i = (I + \boldsymbol{u_s} E_i) d_i, \quad \boldsymbol{u_s} \|E_i\|_\infty < 1$

   → normwise relative forward error is bounded by multiple of u_s and is less than 1

Key Aspects of Analysis II

Allow for general solver:

Let u_s be the *effective precision* of the solve, with $u \leq u_s \leq u_f$

example: LU solve:

Assume computed solution $\hat{d}_i$ to $Ad_i = \hat{r}_i$ satisfies:

1. $\hat{d}_i = (I + u_s E_i)d_i, \quad u_s\|E_i\|_\infty < 1$

   → normwise relative forward error is bounded by multiple of u_s and is less than 1

$u_s\|E_i\|_\infty \leq 3n u_f \left\| |A^{-1}||\hat{L}||\hat{U}| \right\|_\infty$

Key Aspects of Analysis II

Allow for general solver:
Let $\boldsymbol{u_s}$ be the *effective precision* of the solve, with $\boldsymbol{u} \leq \boldsymbol{u_s} \leq \boldsymbol{u_f}$

example: LU solve:

$$\boldsymbol{u_s}\|E_i\|_\infty \leq 3n\boldsymbol{u_f}\left\||A^{-1}||\hat{L}||\hat{U}|\right\|_\infty$$

Assume computed solution $\hat{d}_i$ to $Ad_i = \hat{r}_i$ satisfies:

1. $\hat{d}_i = (I + \boldsymbol{u_s}E_i)d_i, \quad \boldsymbol{u_s}\|E_i\|_\infty < 1$

   → normwise relative forward error is bounded by multiple of u_s and is less than 1

2. $\left\|\hat{r}_i - A\hat{d}_i\right\|_\infty \leq \boldsymbol{u_s}(c_1\|A\|_\infty\left\|\hat{d}_i\right\|_\infty + c_2\|\hat{r}_i\|_\infty)$

   → normwise relative backward error is at most $\max(c_1, c_2)\, u_s$

Key Aspects of Analysis II

Allow for general solver:
Let u_s be the *effective precision* of the solve, with $u \le u_s \le u_f$

example: LU solve:

Assume computed solution $\hat{d}_i$ to $Ad_i = \hat{r}_i$ satisfies:

1. $\hat{d}_i = (I + u_s E_i) d_i, \quad u_s \|E_i\|_\infty < 1$

   → normwise relative forward error is bounded by multiple of u_s and is less than 1

$$u_s \|E_i\|_\infty \le 3n u_f \left\| |A^{-1}| |\hat{L}| |\hat{U}| \right\|_\infty$$

2. $\left\| \hat{r}_i - A\hat{d}_i \right\|_\infty \le u_s \left(c_1 \|A\|_\infty \left\| \hat{d}_i \right\|_\infty + c_2 \|\hat{r}_i\|_\infty \right)$

   → normwise relative backward error is at most $\max(c_1, c_2)\, u_s$

$$\max(c_1, c_2)\, u_s \le \frac{3n u_f \left\| |\hat{L}| |\hat{U}| \right\|_\infty}{\|A\|_\infty}$$

Key Aspects of Analysis II

Allow for general solver:
Let $\boldsymbol{u_s}$ be the *effective precision* of the solve, with $\boldsymbol{u} \le \boldsymbol{u_s} \le \boldsymbol{u_f}$

example: LU solve:

Assume computed solution $\hat{d}_i$ to $Ad_i = \hat{r}_i$ satisfies:

1. $\hat{d}_i = (I + \boldsymbol{u_s} E_i) d_i, \quad \boldsymbol{u_s} \|E_i\|_\infty < 1$

   → normwise relative forward error is bounded by multiple of u_s and is less than 1

$$\boldsymbol{u_s} \|E_i\|_\infty \le 3n\boldsymbol{u_f} \left\| |A^{-1}||\hat{L}||\hat{U}| \right\|_\infty$$

2. $\left\| \hat{r}_i - A\hat{d}_i \right\|_\infty \le \boldsymbol{u_s} \left(c_1 \|A\|_\infty \left\| \hat{d}_i \right\|_\infty + c_2 \|\hat{r}_i\|_\infty \right)$

   → normwise relative backward error is at most $\max(c_1, c_2)\, u_s$

$$\max(c_1, c_2)\, \boldsymbol{u_s} \le \frac{3n\boldsymbol{u_f} \left\| |\hat{L}||\hat{U}| \right\|_\infty}{\|A\|_\infty}$$

3. $\left| \hat{r}_i - A\hat{d}_i \right| \le \boldsymbol{u_s} G_i |\hat{d}_i|$

   → componentwise relative backward error is bounded by a multiple of u_s

E_i, c_1, c_2, and G_i depend on A, $\hat{r}_i$, n, and $\boldsymbol{u_s}$

Key Aspects of Analysis II

Allow for general solver:
Let u_s be the *effective precision* of the solve, with $u \le u_s \le u_f$

example: LU solve:

Assume computed solution $\hat{d}_i$ to $Ad_i = \hat{r}_i$ satisfies:

1. $\hat{d}_i = (I + u_s E_i) d_i, \quad u_s \|E_i\|_\infty < 1$

   → normwise relative forward error is bounded by multiple of u_s and is less than 1

$$u_s \|E_i\|_\infty \le 3n u_f \left\| |A^{-1}| |\hat{L}| |\hat{U}| \right\|_\infty$$

2. $\left\|\hat{r}_i - A\hat{d}_i\right\|_\infty \le u_s (c_1 \|A\|_\infty \left\|\hat{d}_i\right\|_\infty + c_2 \|\hat{r}_i\|_\infty)$

   → normwise relative backward error is at most $\max(c_1, c_2)\, u_s$

$$\max(c_1, c_2)\, u_s \le \frac{3n u_f \left\| |\hat{L}| |\hat{U}| \right\|_\infty}{\|A\|_\infty}$$

3. $\left|\hat{r}_i - A\hat{d}_i\right| \le u_s G_i |\hat{d}_i|$

   → componentwise relative backward error is bounded by a multiple of u_s

$$u_s \|G_i\|_\infty \le 3n u_f \left\| |\hat{L}| |\hat{U}| \right\|_\infty$$

E_i, c_1, c_2, and G_i depend on A, $\hat{r}_i$, n, and u_s

Key Aspects of Analysis II

Allow for general solver:
Let u_s be the *effective precision* of the solve, with $u \leq u_s \leq u_f$

example: LU solve:
$u_s = u_f$

Assume computed solution $\hat{d}_i$ to $Ad_i = \hat{r}_i$ satisfies:

1. $\hat{d}_i = (I + u_s E_i) d_i, \quad u_s \|E_i\|_\infty < 1$

   → normwise relative forward error is bounded by multiple of u_s and is less than 1

$u_s \|E_i\|_\infty \leq 3n u_f \| |A^{-1}| |\hat{L}| |\hat{U}| \|_\infty$

2. $\|\hat{r}_i - A\hat{d}_i\|_\infty \leq u_s (c_1 \|A\|_\infty \|\hat{d}_i\|_\infty + c_2 \|\hat{r}_i\|_\infty)$

   → normwise relative backward error is at most $\max(c_1, c_2)\, u_s$

$$\max(c_1, c_2)\, u_s \leq \frac{3n u_f \| |\hat{L}| |\hat{U}| \|_\infty}{\|A\|_\infty}$$

3. $|\hat{r}_i - A\hat{d}_i| \leq u_s G_i |\hat{d}_i|$

   → componentwise relative backward error is bounded by a multiple of u_s

$u_s \|G_i\|_\infty \leq 3n u_f \| |\hat{L}| |\hat{U}| \|_\infty$

E_i, c_1, c_2, and G_i depend on A, $\hat{r}_i$, n, and u_s

Forward Error for IR3

- Three precisions:
  - u_f: factorization precision
  - u: working precision
  - u_r: residual computation precision

$$\kappa_\infty(A) = \|A^{-1}\|_\infty \|A\|_\infty$$
$$\text{cond}(A) = \| \, |A^{-1}||A| \, \|_\infty$$
$$\text{cond}(A,x) = \| \, |A^{-1}||A||x| \, \|_\infty / \|x\|_\infty$$

Forward Error for IR3

- Three precisions:
  - u_f: factorization precision
  - u: working precision
  - u_r: residual computation precision

$$\kappa_\infty(A) = \|A^{-1}\|_\infty \|A\|_\infty$$
$$\mathrm{cond}(A) = \|\, |A^{-1}||A| \,\|_\infty$$
$$\mathrm{cond}(A,x) = \|\, |A^{-1}||A||x| \,\|_\infty / \|x\|_\infty$$

Theorem [C. and Higham, SISC 40(2), 2018]

For IR in precisions $\boldsymbol{u_f} \geq \boldsymbol{u} \geq \boldsymbol{u_r}$ and effective solve precision $\boldsymbol{u_s}$, if

$$\phi_i \equiv 2\boldsymbol{u_s} \min(\mathrm{cond}(A), \kappa_\infty(A)\mu_i) + \boldsymbol{u_s}\|E_i\|_\infty$$

is less than 1, then the forward error is reduced on the ith iteration by a factor $\approx \phi_i$ until an iterate $\hat{x}_i$ is produced for which

$$\frac{\|x - \hat{x}_i\|_\infty}{\|x\|_\infty} \lesssim 4N\boldsymbol{u_r}\,\mathrm{cond}(A,x) + \boldsymbol{u},$$

where N is the maximum number of nonzeros per row in A.

Forward Error for IR3

- Three precisions:
  - u_f: factorization precision
  - u: working precision
  - u_r: residual computation precision

$$\kappa_\infty(A) = \|A^{-1}\|_\infty \|A\|_\infty$$
$$\text{cond}(A) = \| \, |A^{-1}||A| \, \|_\infty$$
$$\text{cond}(A,x) = \| \, |A^{-1}||A||x| \, \|_\infty / \|x\|_\infty$$

Theorem [C. and Higham, SISC 40(2), 2018]

For IR in precisions $\boldsymbol{u_f} \geq \boldsymbol{u} \geq \boldsymbol{u_r}$ and effective solve precision $\boldsymbol{u_s}$, if

$$\phi_i \equiv 2\boldsymbol{u_s} \min(\text{cond}(A), \kappa_\infty(A)\mu_i) + \boldsymbol{u_s}\|E_i\|_\infty$$

is less than 1, then the forward error is reduced on the ith iteration by a factor $\approx \phi_i$ until an iterate $\hat{x}_i$ is produced for which

$$\frac{\|x - \hat{x}_i\|_\infty}{\|x\|_\infty} \lesssim 4N\boldsymbol{u_r} \, \text{cond}(A,x) + \boldsymbol{u},$$

where N is the maximum number of nonzeros per row in A.

Analogous traditional bounds: $\phi_i \equiv 3n\boldsymbol{u_f}\kappa_\infty(A)$

Normwise Backward Error for IR3

Theorem [C. and Higham, SISC 40(2), 2018]

For IR in precisions $\boldsymbol{u_f} \geq \boldsymbol{u} \geq \boldsymbol{u_r}$ and effective solve precision $\boldsymbol{u_s}$, if

$$\phi_i \equiv (c_1 \kappa_\infty(A) + c_2)\boldsymbol{u_s}$$

is less than 1, then the residual is reduced on the ith iteration by a factor $\approx \phi_i$ until an iterate $\hat{x}_i$ is produced for which

$$\|b - A\hat{x}_i\|_\infty \lesssim N\boldsymbol{u}(\|b\|_\infty + \|A\|_\infty \|\hat{x}_i\|_\infty),$$

where N is the maximum number of nonzeros per row in A.

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}	$\mathrm{cond}(A,x) \cdot 10^{-8}$
	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}
	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}	$\mathrm{cond}(A,x) \cdot 10^{-8}$
	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}	$\text{cond}(A,x) \cdot 10^{-8}$
	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}
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}	$\text{cond}(A,x) \cdot 10^{-16}$
	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}

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. "LP fact.": no $\text{cond}(A,x)$ 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 $\approx 10^{-34}$

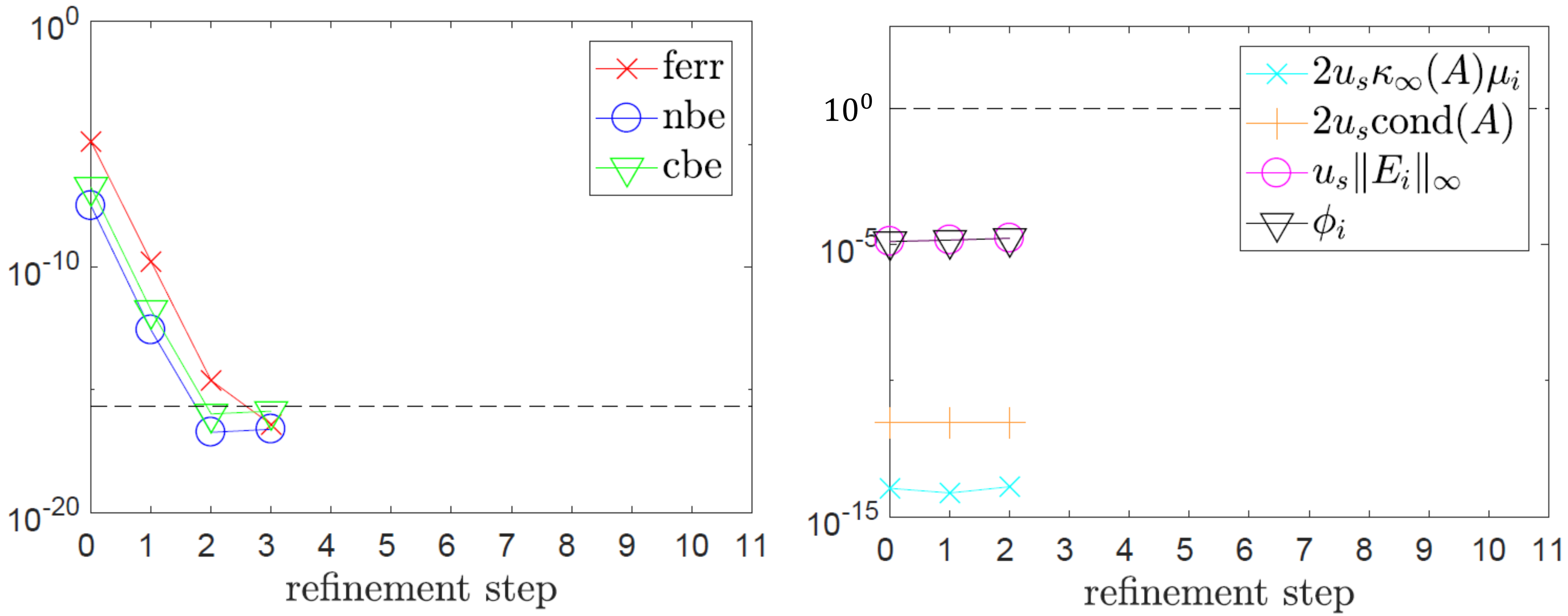
	u_f	u	u_r	max $\kappa_\infty(A)$	Backward error norm	Backward error comp	Forward error
LP fact.	H	S	S	10^4	10^{-8}	10^{-8}	$\mathrm{cond}(A,x)\cdot 10^{-8}$
New	H	S	D	10^4	10^{-8}	10^{-8}	10^{-8}
LP fact.	H	D	D	10^4	10^{-16}	10^{-16}	$\mathrm{cond}(A,x)\cdot 10^{-16}$
New	H	D	Q	10^4	10^{-16}	10^{-16}	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. traditional IR: As long as $\kappa_\infty(A) \leq 10^4$, can use lower precision factorization w/no loss of accuracy!

```
A = gallery('randsvd', 100, 1e3)
b = randn(100,1)
```

$\kappa_\infty(A) \approx$ **1e4**

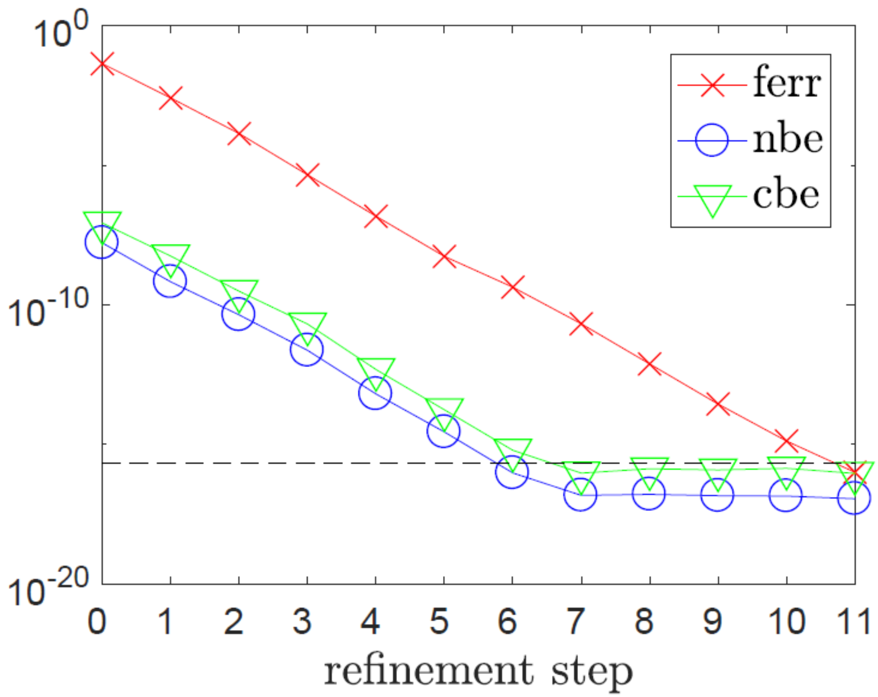
Standard (LU-based) IR with u_f: single, u: double, u_r: quad

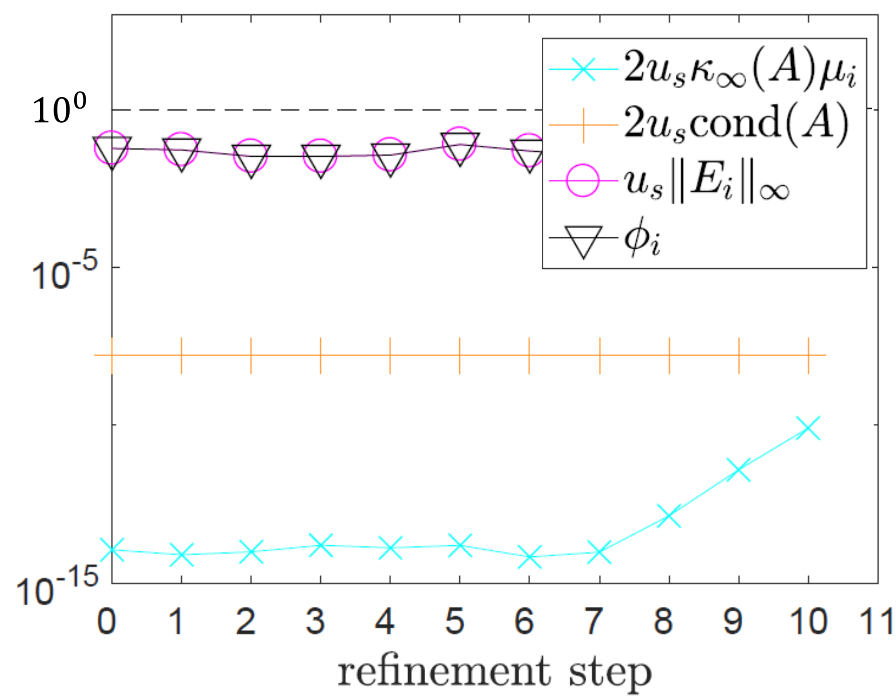


```
A = gallery('randsvd', 100, 1e7)
b = randn(100,1)
```

$\kappa_\infty(A) \approx$ **7e7**

Standard (LU-based) IR with u_f: single, u: double, u_r: quad

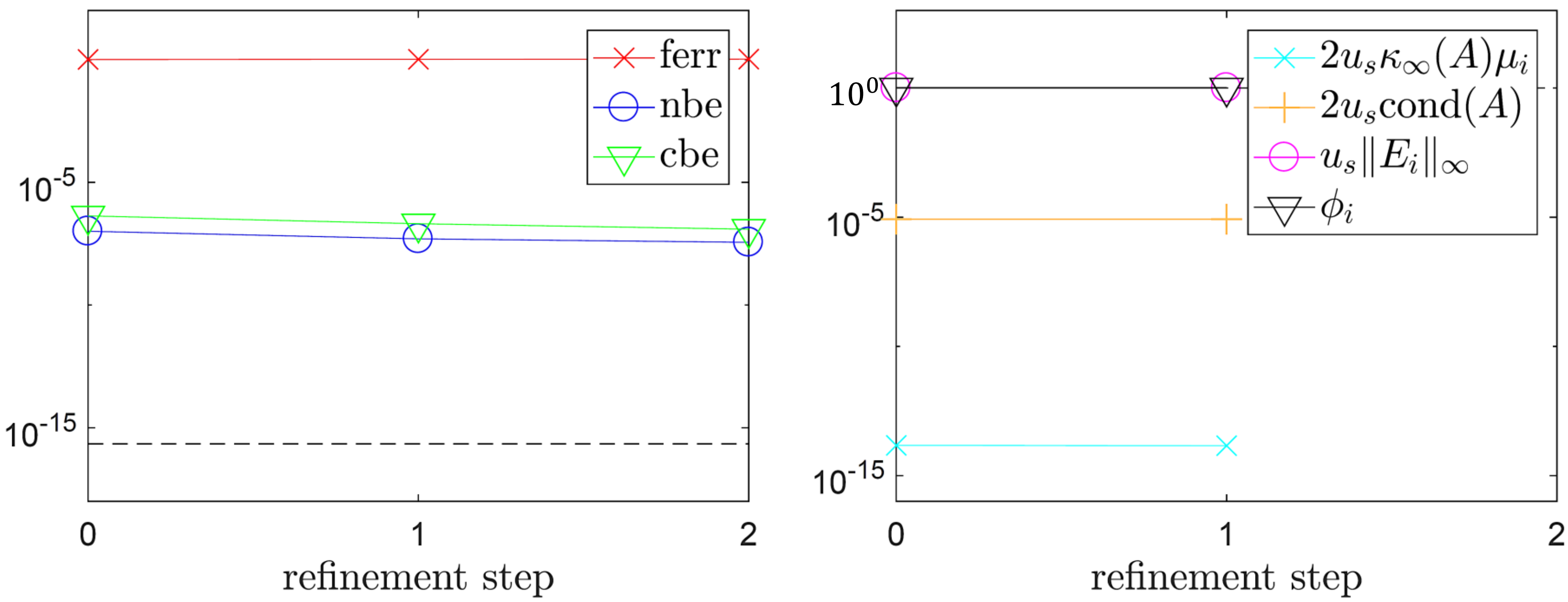




```
A = gallery('randsvd', 100, 1e9)
b = randn(100,1)
```

$\kappa_\infty(A) \approx$ **2e10**

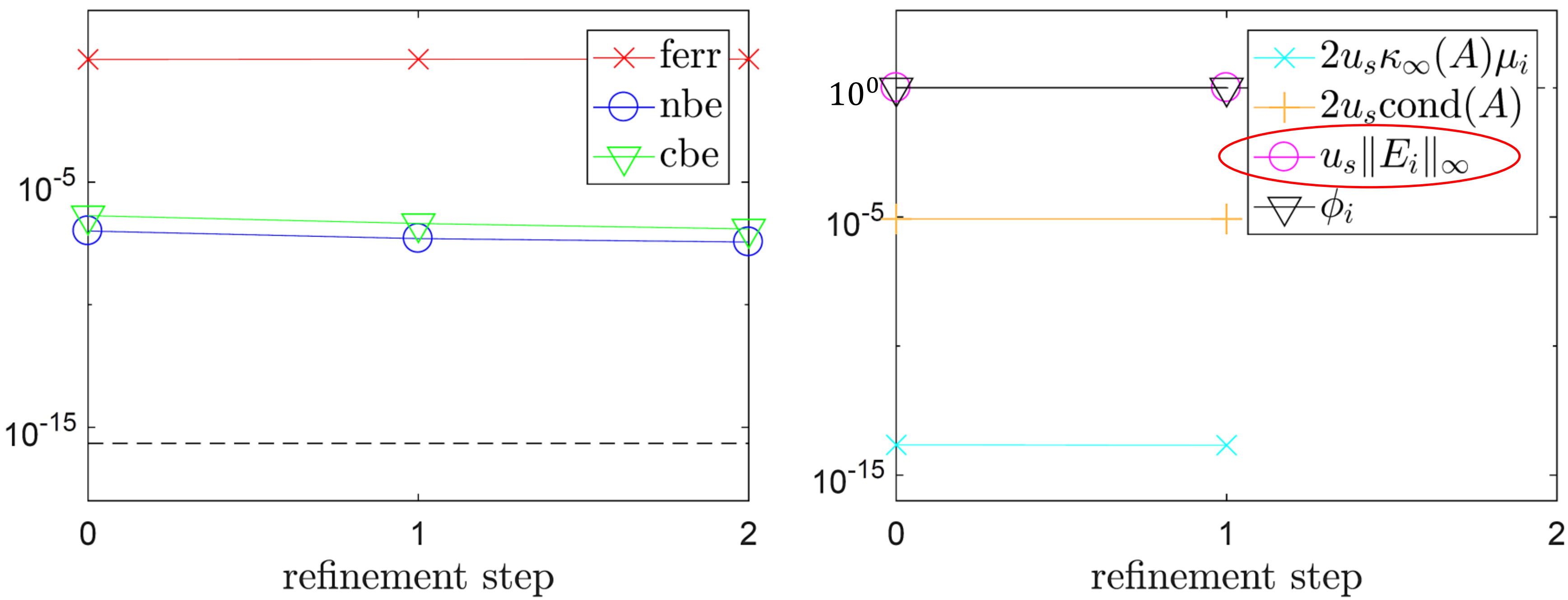
Standard (LU-based) IR with u_f: single, u: double, u_r: quad



```
A = gallery('randsvd', 100, 1e9)
b = randn(100,1)
```

$\kappa_\infty(A) \approx$ **2e10**

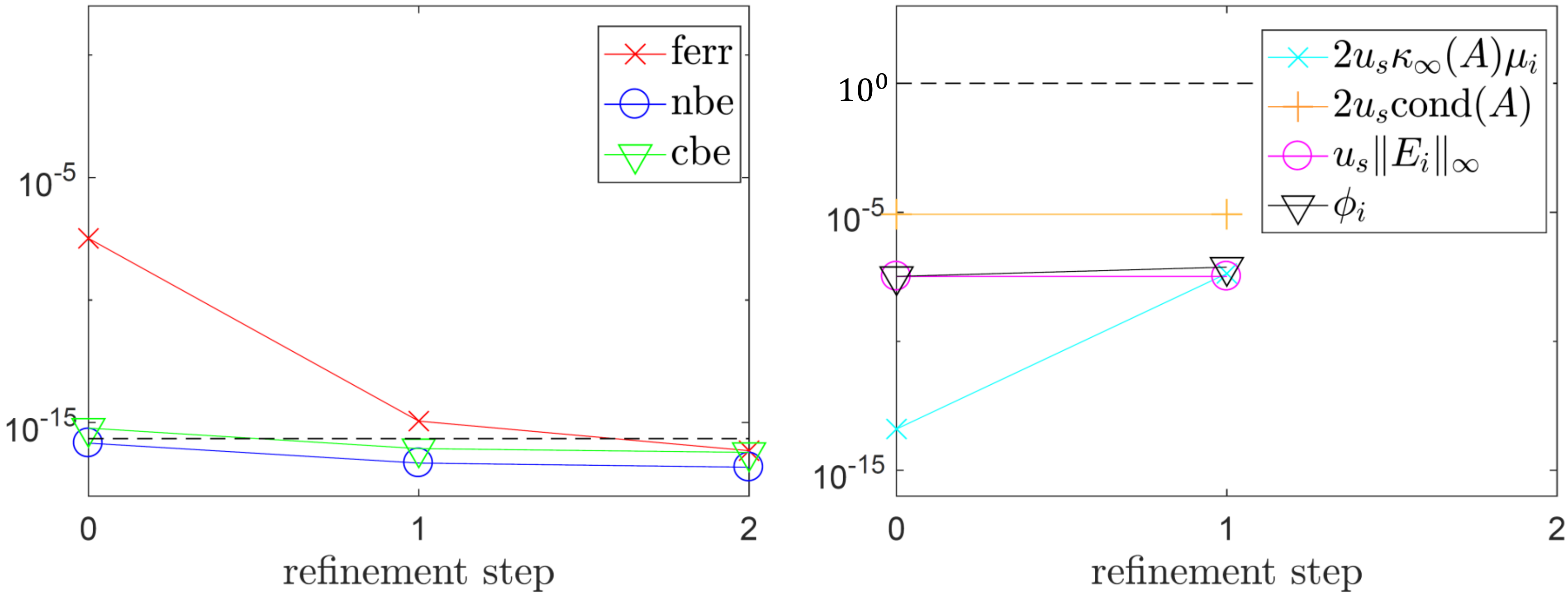
Standard (LU-based) IR with u_f: single, u: double, u_r: quad



```
A = gallery('randsvd', 100, 1e9)
b = randn(100,1)
```

$\kappa_\infty(A) \approx$ **2e10**

Standard (LU-based) IR with u_f: double, u: double, u_r: quad



GMRES-Based Iterative Refinement

- Observation [Rump, 1990]: if $\hat{L}$ and $\hat{U}$ are computed LU factors of A in precision $\boldsymbol{u_f}$, then

$$\kappa_\infty\left(\hat{U}^{-1}\hat{L}^{-1}A\right) \approx 1 + \kappa_\infty(A)\boldsymbol{u_f},$$

even if $\kappa_\infty(A) \gg u_f^{-1}$.

GMRES-Based Iterative Refinement

- Observation [Rump, 1990]: if $\hat{L}$ and $\hat{U}$ are computed LU factors of A in precision $\boldsymbol{u_f}$, then

$$\kappa_\infty\left(\hat{U}^{-1}\hat{L}^{-1}A\right) \approx 1 + \kappa_\infty(A)\boldsymbol{u_f},$$

even if $\kappa_\infty(A) \gg u_f^{-1}$.

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

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

GMRES-Based Iterative Refinement

- Observation [Rump, 1990]: if $\hat{L}$ and $\hat{U}$ are computed LU factors of A in precision $\boldsymbol{u_f}$, then

$$\kappa_\infty(\hat{U}^{-1}\hat{L}^{-1}A) \approx 1 + \kappa_\infty(A)\boldsymbol{u_f},$$

even if $\kappa_\infty(A) \gg u_f^{-1}$.

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

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

Solve $Ax_0 = b$ by LU factorization

for $i = 0$: maxit

$r_i = b - Ax_i$

Solve $Ad_i = r_i$ via GMRES on $\tilde{A}d_i = \tilde{r}_i$

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

GMRES-Based Iterative Refinement

- Observation [Rump, 1990]: if $\hat{L}$ and $\hat{U}$ are computed LU factors of A in precision $\boldsymbol{u_f}$, then

$$\kappa_\infty\big(\hat{U}^{-1}\hat{L}^{-1}A\big) \approx 1 + \kappa_\infty(A)\boldsymbol{u_f},$$

even if $\kappa_\infty(A) \gg u_f^{-1}$.

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

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

Solve $Ax_0 = b$ by LU factorization

for $i = 0$: maxit

$r_i = b - Ax_i$

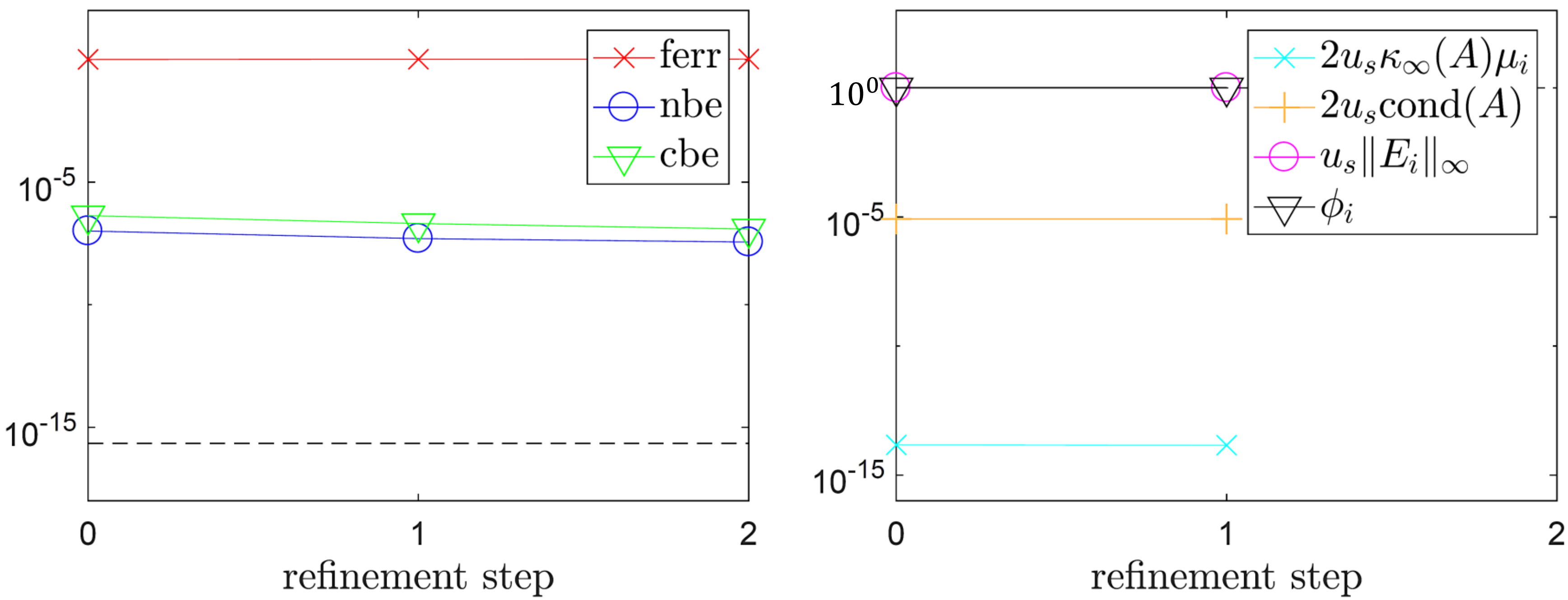
Solve $Ad_i = r_i$ via GMRES on $\tilde{A}d_i = \tilde{r}_i$ → $\boldsymbol{u_s} = \boldsymbol{u}$

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

```
A = gallery('randsvd', 100, 1e9, 2)
b = randn(100,1)
```

$\kappa_\infty(A) \approx$ 2e10, $\text{cond}(A, x) \approx$ 5e9

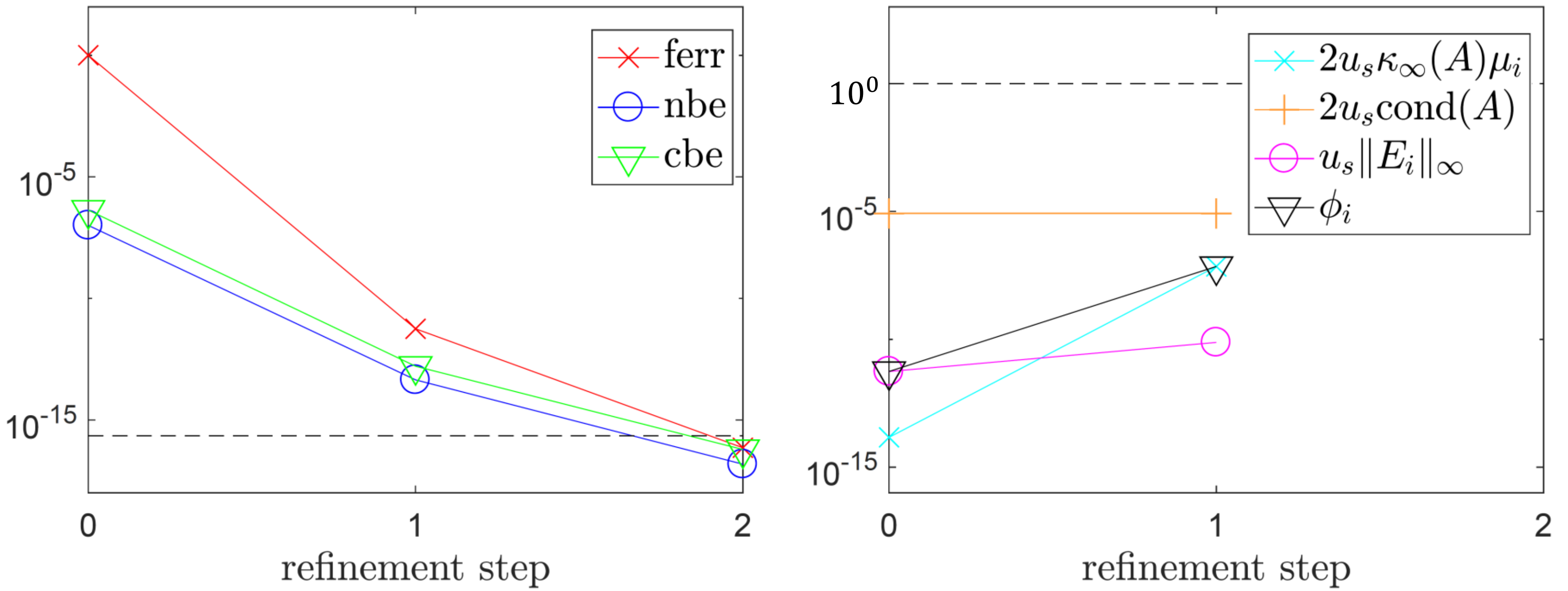
Standard (LU-based) IR with u_f: single, u: double, u_r: quad



```
A = gallery('randsvd', 100, 1e9, 2)
b = randn(100,1)
```

$\kappa_\infty(A) \approx$ 2e10, $\text{cond}(A,x) \approx$ 5e9, $\kappa_\infty(\tilde{A}) \approx$ 2e4

GMRES-IR with u_f: single, u: double, u_r: quad



Number of GMRES iterations: (2,3)

GMRES-IR: Summary

Benefits of GMRES-IR:

	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}

GMRES-IR: Summary

Benefits of GMRES-IR:

	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}

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

GMRES-IR: Summary

Benefits of GMRES-IR:

	u_f	u	u_r	max $\kappa_\infty(A)$	Backward error		Forward error
					norm	comp	
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}

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

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

GMRES-IR: Summary

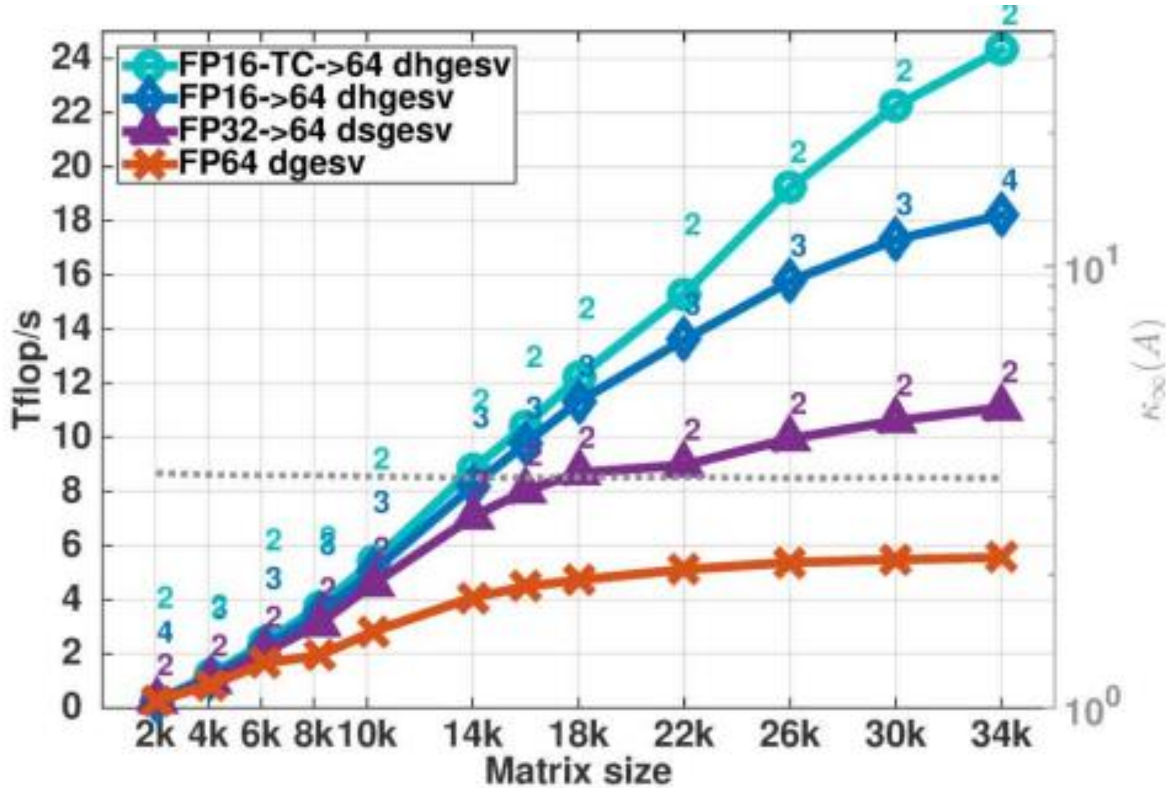
Benefits of GMRES-IR:

	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}

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

Performance Results (MAGMA)

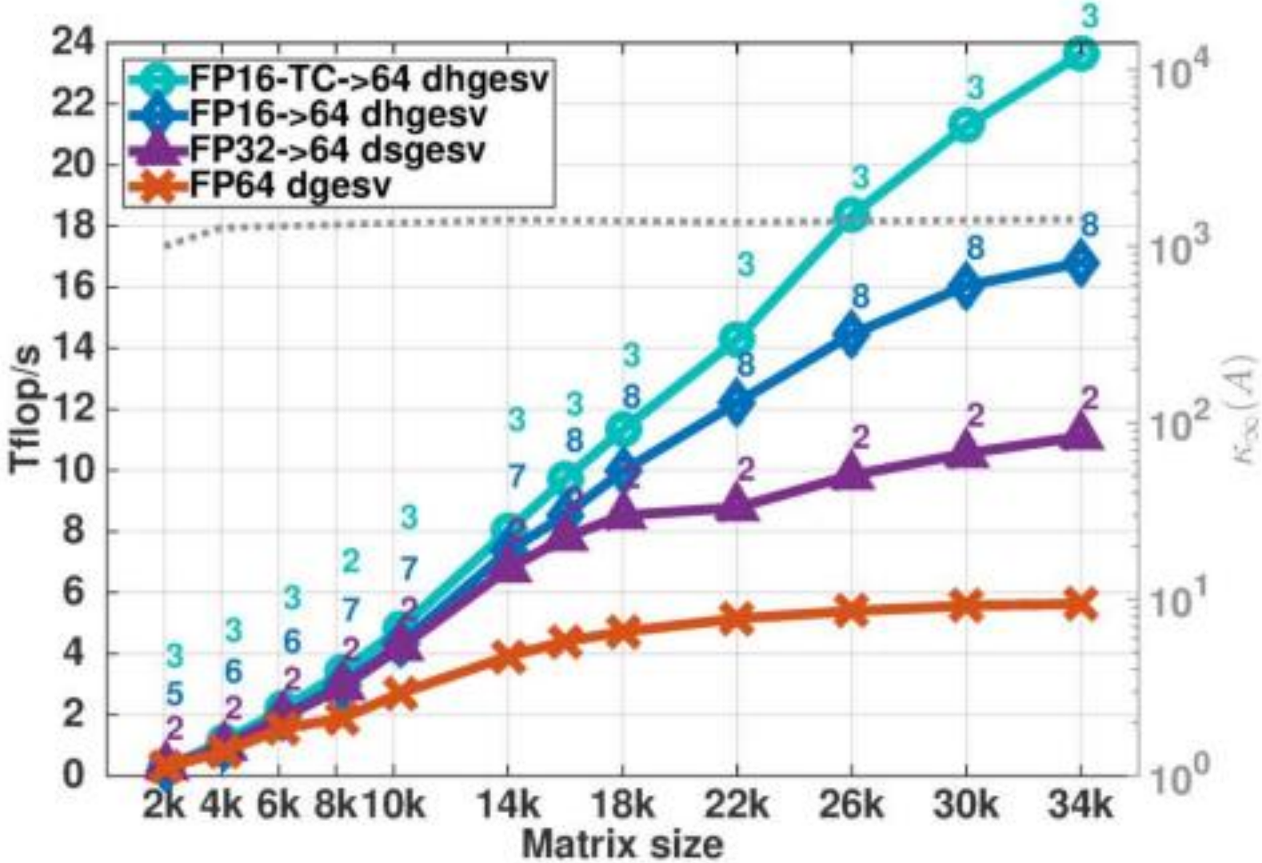
- [Haidar, Tomov, Dongarra, Higham, 2018]
- 2-precision GMRES-IR approach ($u = u_r$) on NVIDIA V100
- IR run to FP64 accuracy, max 400 iterations in GMRES
- Tflops/s measured as $(2n^3/3)$/time



(a) Matrix of type 1: diagonally dominant.

Performance Results (MAGMA)

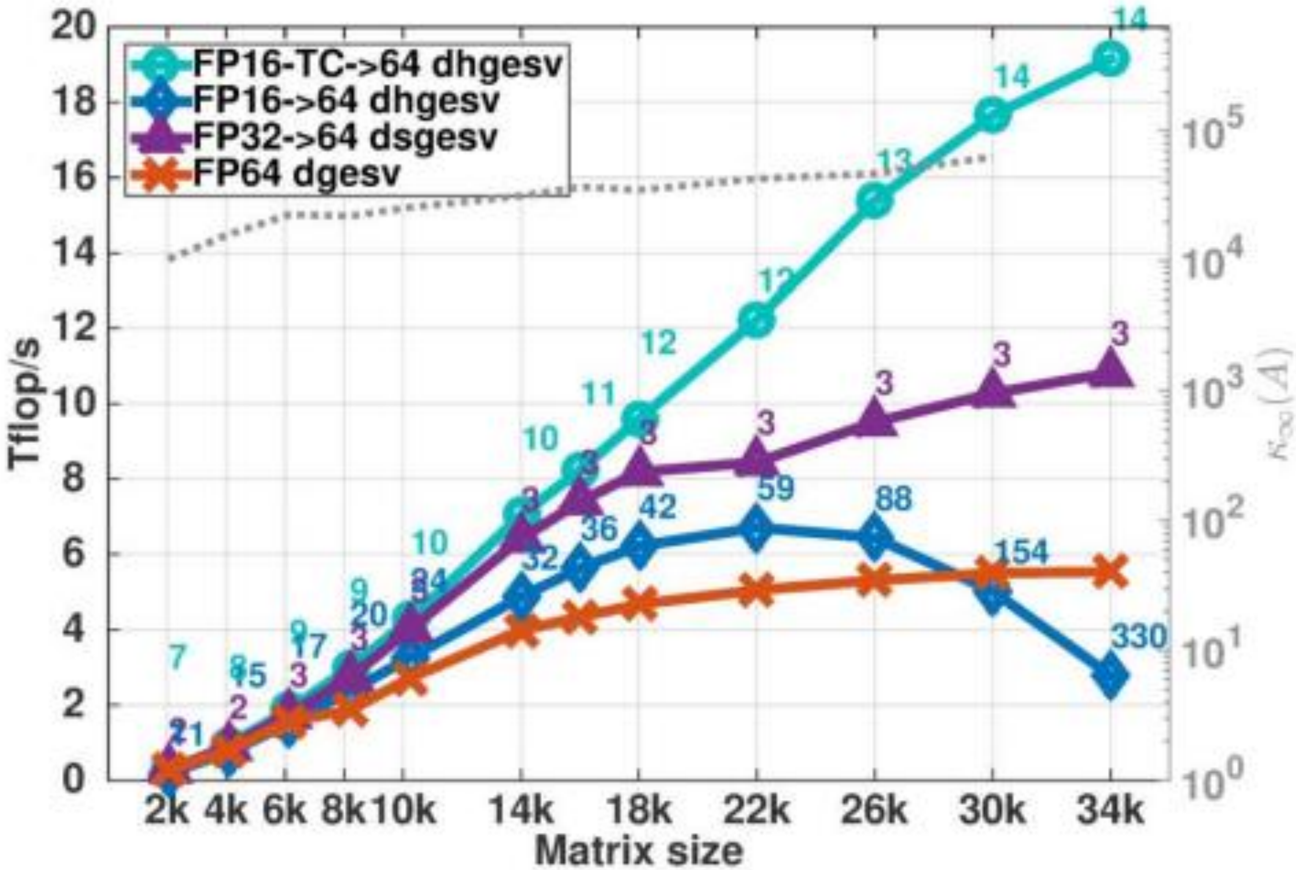
- [Haidar, Tomov, Dongarra, Higham, 2018]



(a) Matrix of type 3: positive λ with clustered singular values, σ_i=(1, $\cdots$, 1, $\frac{1}{cond}$).

Performance Results (MAGMA)

- [Haidar, Tomov, Dongarra, Higham, 2018]



(b) Matrix of type 4: clustered singular values, $\sigma_i = (1, \cdots, 1, \frac{1}{cond})$.

Performance Results

[Haidar, Tomov, Dongarra, Higham, 2018]

Performance for Matrices from SuiteSparse

name	Description	size	$\kappa_\infty(A)$	dgesv	dsgesv		dhgesv		dhgesv-TC		
				time(s)	# iter	time (s)	# iter	time (s)	# iter	time (s)	
em192	radar design	26896	10^6	5.70	3	3.11	40	5.21	10	2.05	2.8×
appu	NASA app benchmark	14000	10^4	0.43	2	0.27	7	0.24	4	0.19	2.3×
ns3Da	3D Navier Stokes	20414	$7.6\ 10^3$	1.12	2	0.69	6	0.54	4	0.43	2.6×
nd6k	ND problem set	18000	$3.5\ 10^2$	0.81	2	0.45	4	0.36	3	0.30	2.7×
nd12k	ND problem set	36000	$4.3\ 10^2$	5.36	2	2.75	3	1.76	3	1.31	4.1×

GMRES-IR in Libraries and Applications

- MAGMA: Dense linear algebra routines for heterogeneous/hybrid architectures

magma / src / dxgesv_gmres_gpu.cpp

```
-------
DSGESV or DHGESV expert interface.
It computes the solution to a real system of linear equations
   A * X = B,  A**T * X = B,  or  A**H * X = B,
where A is an N-by-N matrix and X and B are N-by-NRHS matrices.
the accomodate the Single Precision DSGESV and the Half precision dhgesv API.
precision and iterative refinement solver are specified by facto_type, solver_type.
For other API parameter please refer to the corresponding dsgesv or dhgesv.
```

- NVIDIA's cuSOLVER Library

2.2.1.6. cusolverIRSRefinement_t

The `cusolverIRSRefinement_t` type indicates which solver type would be used for the specific cusolver function. Most of our experimentation shows that CUSOLVER_IRS_REFINE_GMRES is the best option.

`CUSOLVER_IRS_REFINE_GMRES`	GMRES (Generalized Minimal Residual) based iterative refinement solver. In recent study, the GMRES method has drawn the scientific community attention for its ability to be used as refinement solver that outperforms the classical iterative refinement method. based on our experimentation, we recommend this setting.

- In production codes: FK6D/ASGarD code (Oak Ridge National Lab, USA) for tokomak containment problem

Comments and Caveats I

- Convergence tolerance τ for GMRES?
  - Smaller $\tau \rightarrow$ more GMRES iterations, potentially fewer refinement steps
  - Larger $\tau \rightarrow$ fewer GMRES iterations, potentially more refinement steps

Comments and Caveats I

- Convergence tolerance τ for GMRES?
  - Smaller $\tau \rightarrow$ more GMRES iterations, potentially fewer refinement steps
  - Larger $\tau \rightarrow$ fewer GMRES iterations, potentially more refinement steps

- What about overflow, underflow, subnormal numbers?
  - Sophisticated scaling methods can help avoid this
    - "Squeezing a Matrix into Half Precision, with an Application to Solving Linear Systems" [Higham, Pranesh, Zounon, 2019]

Comments and Caveats II

- Convergence rate of GMRES?

Comments and Caveats II

- Convergence rate of GMRES?
  - If A is ill conditioned and LU factorization is performed in very low precision, it can be a poor preconditioner
    - e.g., if $\tilde{A}$ still has cluster of eigenvalues near origin, GMRES can stagnate until n^{th} iteration, regardless of $\kappa_\infty(A)$ [Liesen and Tichý, 2004]
  - Potential remedies: deflation, Krylov subspace recycling [Oktay, C., 2022], using additional preconditioner

Comments and Caveats II

- Convergence rate of GMRES?
    - If A is ill conditioned and LU factorization is performed in very low precision, it can be a poor preconditioner
        - e.g., if $\tilde{A}$ still has cluster of eigenvalues near origin, GMRES can stagnate until n^{th} iteration, regardless of $\kappa_\infty(A)$ [Liesen and Tichý, 2004]
    - Potential remedies: deflation, Krylov subspace recycling [Oktay, C., 2022], using additional preconditioner

- Depending on conditioning of A, applying $\tilde{A}$ to a vector must be done accurately (precision u^2) in each GMRES iteration
    - Recent development of 5-precision GMRES-IR algorithm [Amestoy et al., 2021]
        - Defines working precision u_g for GMRES and u_p for preconditioning within GMRES

Comments and Caveats II

- Convergence rate of GMRES?
  - If A is ill conditioned and LU factorization is performed in very low precision, it can be a poor preconditioner
    - e.g., if $\tilde{A}$ still has cluster of eigenvalues near origin, GMRES can stagnate until n^{th} iteration, regardless of $\kappa_\infty(A)$ [Liesen and Tichý, 2004]
  - Potential remedies: deflation, Krylov subspace recycling [Oktay, C., 2022], using additional preconditioner

- Depending on conditioning of A, applying $\tilde{A}$ to a vector must be done accurately (precision u^2) in each GMRES iteration
  - Recent development of 5-precision GMRES-IR algorithm [Amestoy et al., 2021]
    - Defines working precision u_g for GMRES and u_p for preconditioning within GMRES

- Why GMRES?
  - Theoretical purposes: existing analysis and proof of backward stability [Paige, Rozložník, Strakoš, 2006]
  - In practice, use any solver you want!

Extension to 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 to 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":

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

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 $(m + n)$:

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

- 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":

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

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 $(m+n)$:

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

- 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":

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

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 $(m+n)$:

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

- 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**

Extensions and Current Work

- Multistage mixed precision iterative refinement

  [Oktay, C., 2021]

- Use of inexact preconditioners: SPAI, etc.

  [Amestoy, Buttari, Higham, L'Excellent, Mary, Vieuble, 2022]

  [C., Khan, 2022]

- Use of low-precision randomized preconditioners

  Ongoing work with I. Daužickaitė

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/