Accelerating Linear Systems Solution by Exploiting Low-Rank Approximations to Factorization Error

Theo Mary, joint work with Nick Higham
University of Manchester, School of Mathematics

Manchester Numerical Analysis

IMA Conference on Numerical Linear Algebra and Optimization,
Birmingham, 27-29 June 2018

Context

Objective

- Compute solution to linear system $Ax = b$
- $A \in \mathbb{R}^{n \times n}$ is ill conditioned

LU-based preconditioner

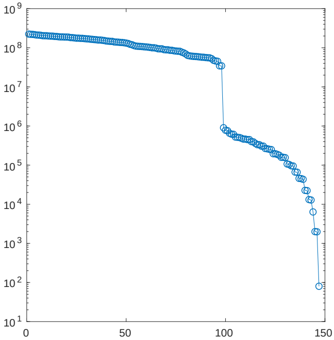
1. Compute approximate factorization $A = \widehat{L}\widehat{U} + \Delta A$
   - Half-precision factorization
   - Incomplete LU factorization
   - Structured matrix factorization: Block Low-Rank, $\mathcal{H}$, HSS,...
2. Solve $\Pi_{LU} A x = \Pi_{LU} b$ with $\Pi_{LU} = \widehat{U}^{-1}\widehat{L}^{-1}$ via some iterative method

- Convergence to solution may be slow or fail

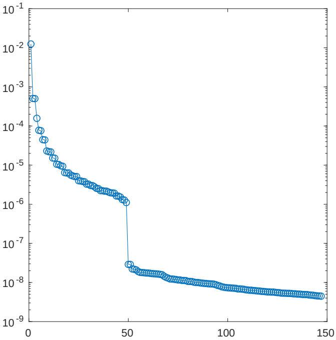
$\Rightarrow$ **Objective: accelerate convergence**

Key observation

Matrix lund_a ($n = 147$, $\kappa(A) = 2.8e{+}06$)



SVD of A



SVD of A^{-1}

- Often, A is ill conditioned due to a small number of small singular values
- Then, A^{-1} is numerically low-rank

Key idea

Factorization error might be low-rank?

Let the error $E = \widehat{U}^{-1}\widehat{L}^{-1}A - I = \widehat{U}^{-1}\widehat{L}^{-1}(\widehat{L}\widehat{U} + \Delta A) - I$

$$= \widehat{U}^{-1}\widehat{L}^{-1}\Delta A \approx A^{-1}\Delta A$$

Does E retain the low-rank property of A^{-1}?

A novel preconditioner

Consider the preconditioner

$$\Pi_{E_k} = (I + E_k)^{-1}\Pi_{LU}$$

with E_k a rank-k approximation to E.

- If $E = E_k$, $\Pi_{E_k} = A^{-1}$
- If $E \approx E_k$ for some small k, Π_{E_k} can be computed cheaply

Preprint

N. J. Higham and T. Mary, *A New Preconditioner that Exploits Low-Rank Approximations to Factorization Error*, MIMS EPrint 2018.10.

Problem statement

Low-rank gap

$$\varepsilon_k(A) = \min_{W_k} \left\{ \frac{\|A - W_k\|}{\|A\|} : \operatorname{rank}(W_k) \le k \right\}$$

Eckart-Young-Mirsky

$$\varepsilon_k(A) = \frac{\sigma_{k+1}(A)}{\sigma_1(A)}$$

Problem statement

Quantify worst-case reduction of the low-rank gap from A^{-1} to $E = \widehat{U}^{-1}\widehat{L}^{-1}\Delta A$, i.e find some β such that

$$\varepsilon_k(E) \le \beta \varepsilon_k(A^{-1})$$

Bound on the low-rankness of E

Theorem

$$\varepsilon_k(E) \le \beta_1 \beta_2 \varepsilon_k(A^{-1})$$

with

$$\varepsilon_k(\widehat{U}^{-1}\widehat{L}^{-1}) \le \beta_1 \varepsilon_k(A^{-1})$$

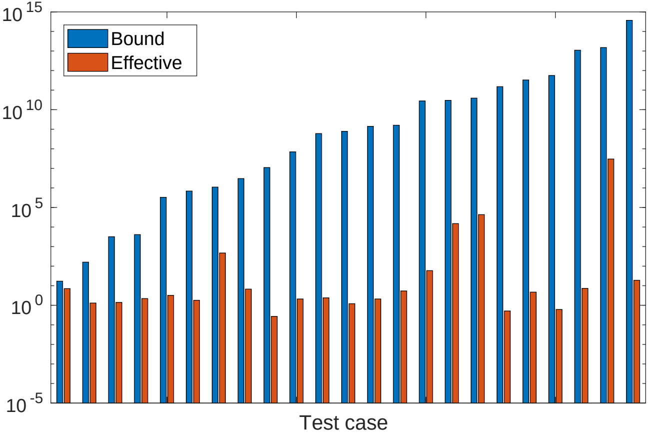
$$\beta_1 = \left(1 + \|A^{-1}\Delta A\|\right)\left(1 + \|\widehat{U}^{-1}\widehat{L}^{-1}\Delta A\|\right)$$

$$\varepsilon_k(E) = \varepsilon_k(\widehat{U}^{-1}\widehat{L}^{-1}\Delta A) \le \beta_2 \varepsilon_k(\widehat{U}^{-1}\widehat{L}^{-1})$$

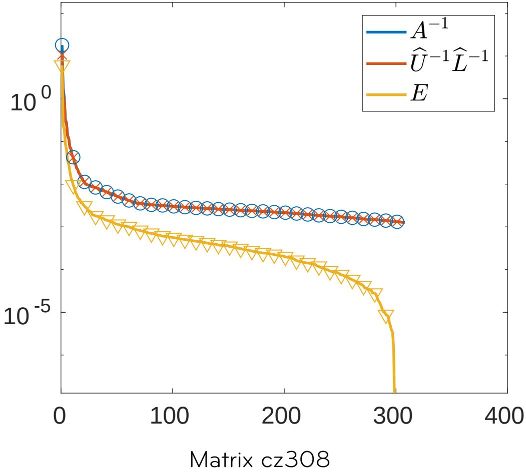
$$\beta_2 = \frac{\|\widehat{U}^{-1}\widehat{L}^{-1}\| \, \|\Delta A\|}{\|\widehat{U}^{-1}\widehat{L}^{-1}\Delta A\|}$$

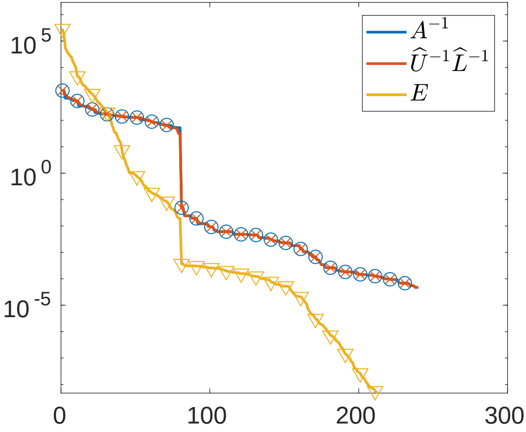
- β_1: maximal deviation of the sing. vals. by additive perturbation
- β_2: should be small for typical ΔA

The bound is pessimistic



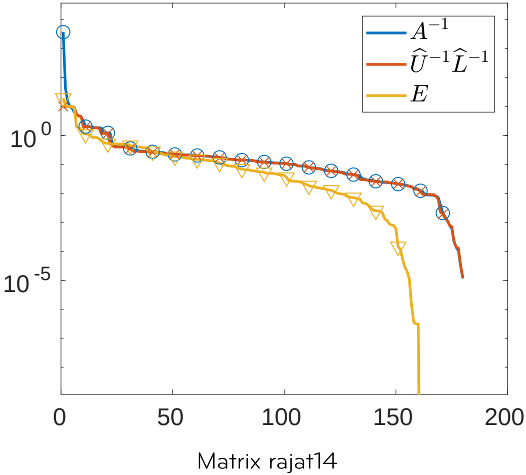
Typical SV distributions of A^{-1} and E



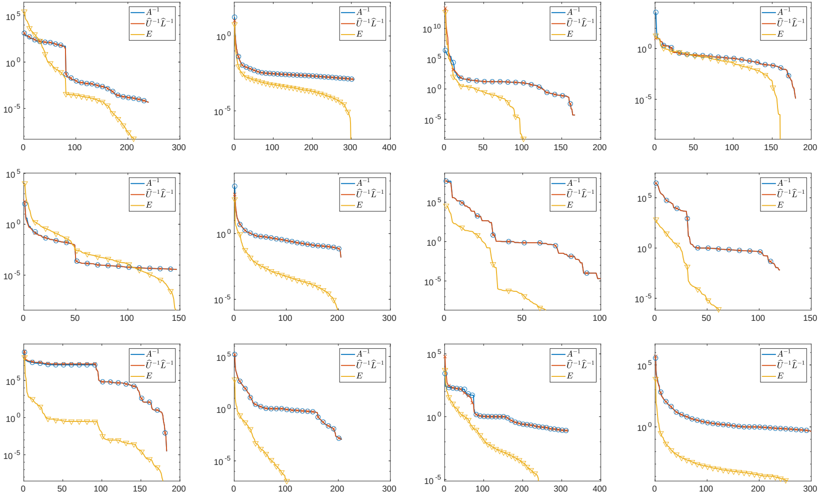


Matrix steam1

Typical SV distributions of A^{-1} and E



Typical SV distributions of A^{-1} and E



We did **not** specifically select matrices for which A^{-1} is low-rank!

Computing E_k

We need to build

$$\Pi_{E_k} = (I + E_k)^{-1}\Pi_{LU} = (I + E_k)^{-1}\widehat{U}^{-1}\widehat{L}^{-1}$$

where E_k is a rank-k approximation of $E = \widehat{U}^{-1}\widehat{L}^{-1}A - I$

E cannot be built explicitly! $\Rightarrow$ Use **randomized** method

Algorithm 1 Randomized SVD via direct SVD of $V^T E$.

1: {Input: the error matrix $E = \widehat{U}^{-1}\widehat{L}^{-1}A - I$, stored implicitly.}
2: Sample E: $S = E\Omega$, with Ω a $n \times (k+p)$ random matrix.
3: Orthonormalize S: $V = \mathrm{qr}(S)$.
4: Compute SVD of $V^T E$: $X\Sigma Y^T = V^T E$.
5: Truncate X, Σ, Y into X_k, Σ_k, Y_k.
6: The SVD of E_k is given by $(VX_k)\Sigma_k Y_k^T$.

Computational cost analysis

Algorithm 1 Randomized SVD via direct SVD of $V^T E$.

1: {Input: the error matrix $E = \widehat{U}^{-1}\widehat{L}^{-1}A - I$, stored implicitly.}
2: Sample E: $S = E\Omega$, with Ω a $n \times (k+p)$ random matrix.
3: Orthonormalize S: $V = \mathrm{qr}(S)$.
4: Compute SVD of $V^T E$: $X\Sigma Y^T = V^T E$.
5: Truncate X, Σ, Y into X_k, Σ_k, Y_k.
6: The SVD of E_k is given by $(VX_k)\Sigma_k Y_k^T$.

$$\ell = k + p$$

	setup	solve
Π_{LU}	$\frac{2}{3}n^3$	$2n^2$
$\Pi_{E_k}^{(1)}$	$\frac{2}{3}n^3 + 8n^2\ell + O(n\ell^2)$	$2n^2 + O(nk)$

Computational cost analysis

Algorithm 1 Randomized SVD via direct SVD of $V^T E$.

1: {Input: the error matrix $E = \widehat{U}^{-1}\widehat{L}^{-1}A - I$, stored implicitly.}
2: Sample E: $S = E\Omega$, with Ω a $n \times (k+p)$ random FFT matrix.
3: Orthonormalize S: $V = \mathrm{qr}(S)$.
4: Compute SVD of $V^T E$: $X \Sigma Y^T = V^T E$.
5: Truncate X, Σ, Y into X_k, Σ_k, Y_k.
6: The SVD of E_k is given by $(VX_k)\Sigma_k Y_k^T$.

$$\ell = k + p$$

	setup	solve
Π_{LU}	$\frac{2}{3}n^3$	$2n^2$
$\Pi_{E_k}^{(1)}$	$\frac{2}{3}n^3 + 8n^2\ell + O(n\ell^2)$	$2n^2 + O(nk)$
$\Pi_{E_k}^{(2)}$	$\frac{2}{3}n^3 + 6n^2\ell + 4n^2 \log \ell + O(n\ell^2)$	$2n^2 + O(nk)$

Computational cost analysis

Algorithm 1 Randomized SVD via row extraction.

1: {Input: the error matrix $E = \widehat{U}^{-1}\widehat{L}^{-1}A - I$, stored implicitly.}
2: Sample E: $S = E\Omega$, with Ω a $n \times (k+p)$ random FFT matrix.
3: Orthonormalize S: $V = \mathrm{qr}(S)$.
4: Compute ID of V: $V = (I_k \;\; W)^T V_{(K,:)}$.
5: Extract $E_{(K,:)}$ and compute a QR factorization $E^T_{(K,:)} = QR$.
6: Compute SVD of $(I_k \;\; W)^T R^T$: $X\Sigma Y^T = (I_k \;\; W)^T R^T$.
7: Truncate X, Σ, Y into X_k, Σ_k, Y_k.
8: The SVD of E_k is given by $(VX_k)\Sigma_k Y_k^T$.

$$\ell = k + p$$

	setup	solve
Π_{LU}	$\frac{2}{3}n^3$	$2n^2$
$\Pi^{(1)}_{E_k}$	$\frac{2}{3}n^3 + 8n^2\ell + O(n\ell^2)$	$2n^2 + O(nk)$
$\Pi^{(2)}_{E_k}$	$\frac{2}{3}n^3 + 6n^2\ell + 4n^2\log\ell + O(n\ell^2)$	$2n^2 + O(nk)$
$\Pi^{(3)}_{E_k}$	$\frac{2}{3}n^3 + 2n^2\ell + 4n^2\log\ell + O(n\ell^2)$	$2n^2 + O(nk)$

Experimental setting

- Three types of approximate LU factorization:
  - Half-precision
  - Incomplete LU with drop tolerance $10^{-5} \leq \tau \leq 10^{-1}$
  - Block Low-Rank with low-rank threshold $10^{-9} \leq \tau \leq 10^{-1}$

Experimental setting

- Three types of approximate LU factorization:
  - Half-precision
  - Incomplete LU with drop tolerance $10^{-5} \le \tau \le 10^{-1}$
  - Block Low-Rank with low-rank threshold $10^{-9} \le \tau \le 10^{-1}$
- Iterative solver is GMRES-based iterative refinement with three precisions
  - FP64 working precision and residual is computed in FP128
  - Max nb of GMRES iterations per IR step is 100
  - Max nb of IR steps is 10

Experimental setting

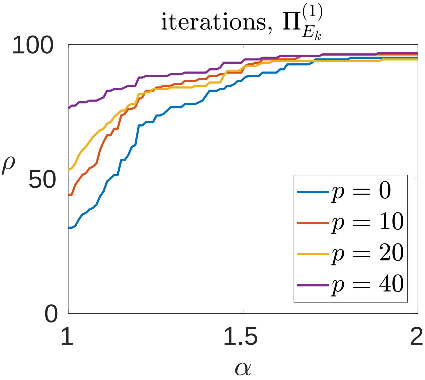
- Three types of approximate LU factorization:
  - Half-precision
  - Incomplete LU with drop tolerance $10^{-5} \leq \tau \leq 10^{-1}$
  - Block Low-Rank with low-rank threshold $10^{-9} \leq \tau \leq 10^{-1}$
- Iterative solver is GMRES-based iterative refinement with three precisions
  - FP64 working precision and residual is computed in FP128
  - Max nb of GMRES iterations per IR step is 100
  - Max nb of IR steps is 10
- Large set of real-life matrices
  - $53 \leq n \leq 494$ and $10^3 \leq \kappa(A) \leq 10^{14}$
  - Most are sparse, but treated as dense
  - 149 tests on 40 different matrices

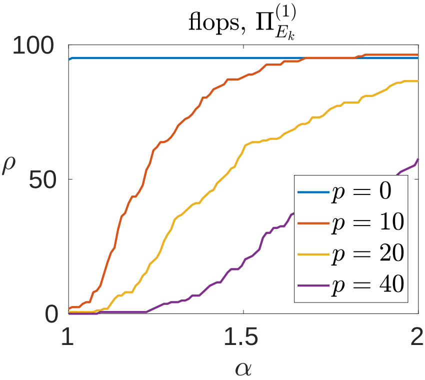
Experimental setting

- Three types of approximate LU factorization:
  - Half-precision
  - Incomplete LU with drop tolerance $10^{-5} \leq \tau \leq 10^{-1}$
  - Block Low-Rank with low-rank threshold $10^{-9} \leq \tau \leq 10^{-1}$
- Iterative solver is GMRES-based iterative refinement with three precisions
  - FP64 working precision and residual is computed in FP128
  - Max nb of GMRES iterations per IR step is 100
  - Max nb of IR steps is 10
- Large set of real-life matrices
  - $53 \leq n \leq 494$ and $10^3 \leq \kappa(A) \leq 10^{14}$
  - Most are sparse, but treated as dense
  - 149 tests on 40 different matrices
- MATLAB code running on laptop
  - We measure nb of iterations and flops
  - Time is only estimated, not measured

Parameter tuning: oversampling p

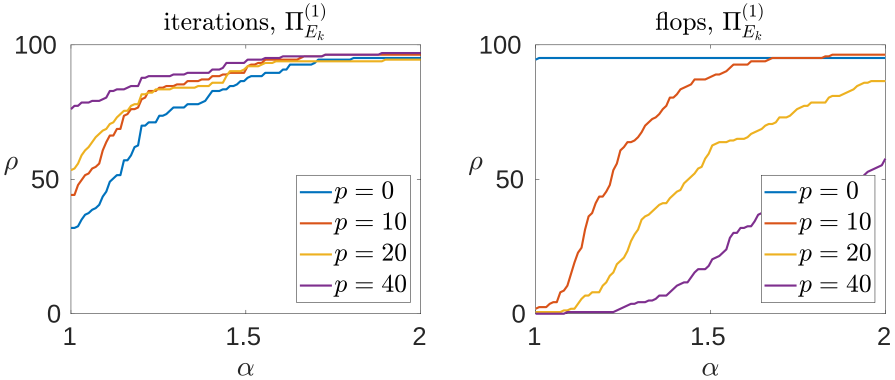
Performance profile: ρ is the percentage of problems solved for less than α × the cost of the best choice ⇒ **higher is better**





Parameter tuning: oversampling p

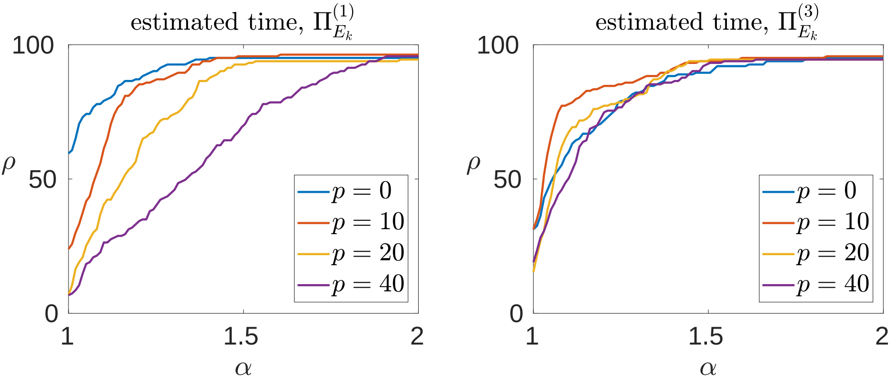
Performance profile: ρ is the percentage of problems solved for less than $\alpha \times$ the cost of the best choice $\Rightarrow$ **higher is better**



We seek a compromise between number of iterations and flops to minimize time, which we estimate assuming BLAS-2 is $10\times$ slower than BLAS-3

Parameter tuning: oversampling p

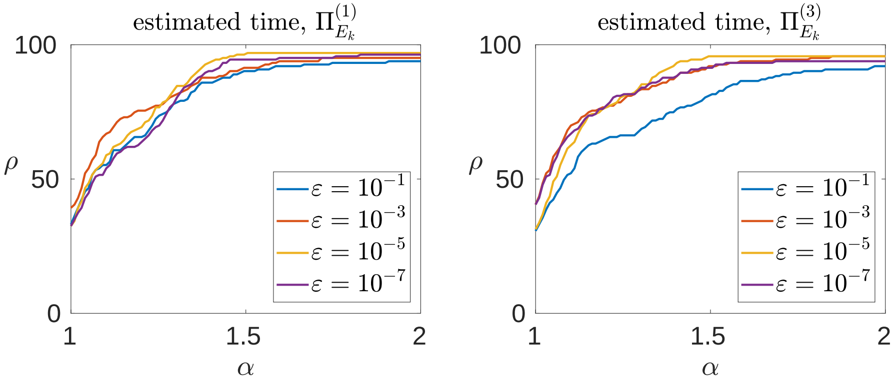
Performance profile: ρ is the percentage of problems solved for less than $\alpha \times$ the cost of the best choice $\Rightarrow$ **higher is better**



Need to set oversampling p differently depending on preconditioner variant

Parameter tuning: ε threshold ($\|E - E_k\| \leq \varepsilon$)

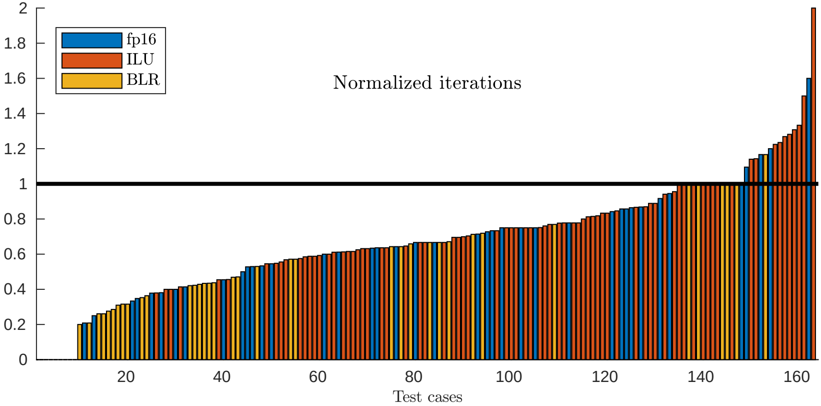
Performance profile: ρ is the percentage of problems solved for less than $\alpha \times$ the cost of the best choice $\Rightarrow$ **higher is better**



Similar trend for low-rank threshold ε

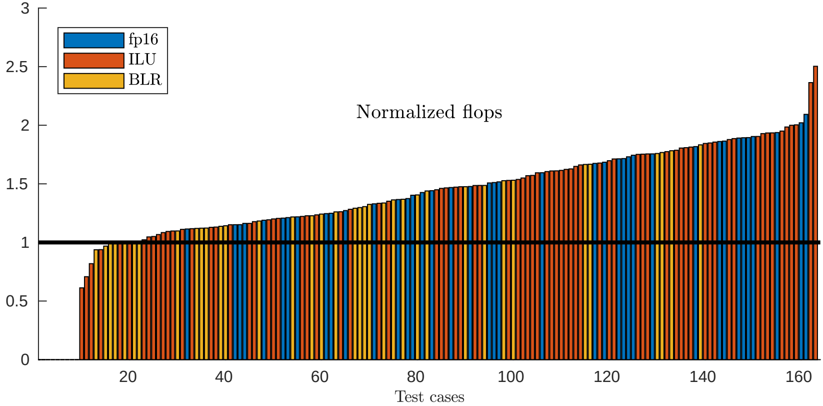
Results with black-box setting

Black-box setting: use $\Pi^{(3)}_{E_k}$ with $p = 10$ and $\varepsilon = 10^{-7}$



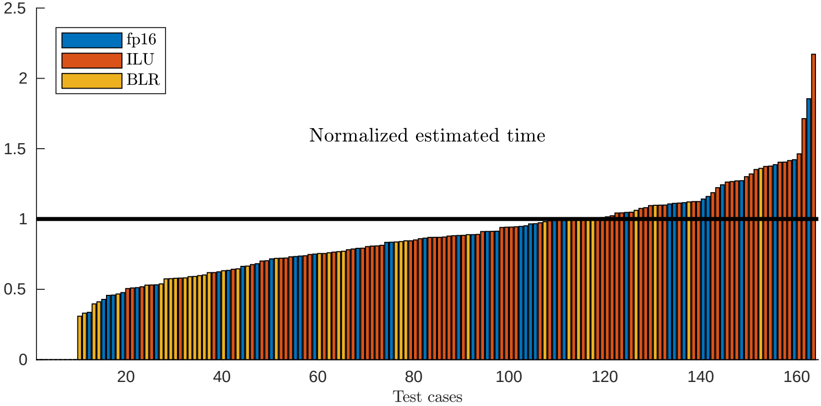
Results with black-box setting

Black-box setting: use $\Pi^{(3)}_{E_k}$ with $p = 10$ and $\varepsilon = 10^{-7}$



Results with black-box setting

Black-box setting: use $\Pi_{E_k}^{(3)}$ with $p = 10$ and $\varepsilon = 10^{-7}$



Application to large-scale, sparse matrices

P. R. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary, *Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures*.

Application to BLR-MUMPS sparse multifrontal solver
BLR threshold $= 10^{-2}$, iterate until converged to accuracy 10^{-9}

Matrix	n	Π_{LU}		Π_{E_k}	
		Iter.	Time	Iter.	Time
audikw_1	1.0M	691	1163	331	625
Bump_2911	2.9M	–	–	284	1708
Emilia_923	0.9M	174	304	136	267
Fault_639	0.6M	–	–	294	345
Ga41As41H72	0.3M	–	–	135	143
Hook_1498	1.5M	417	902	356	808
Si87H76	0.2M	–	–	131	116

Good potential to improve low-precision, low-memory BLR solvers

Conclusion

Summary

- Ill-conditioned matrices often have a numerically low-rank inverse
- Theoretical justification of why the error $E = \widehat{U}^{-1}\widehat{L}^{-1}A - I$ retains this property
- Novel preconditioner based on a low-rank approximation to the error to accelerate linear systems solution

Future work

- High-performance implementation for FP16 and ILU
- Well suited for GPUs (FP16 $8\times$ faster than FP32!)

Slides and paper available here

http://personalpages.manchester.ac.uk/staff/theo.mary/

Backup slides

Ingredient 1: $\widehat{U}^{-1}\widehat{L}^{-1}$ is low-rank if A^{-1} is

Lemma

$$\sigma_i(X + \Delta X) \le \sigma_i(X)\left(1 + \|X^{-1}\Delta X\|\right)$$

Apply lemma twice:

$$X = \widehat{L}\widehat{U} \text{ and } \Delta X = \Delta A \quad \Rightarrow \quad \sigma_i(A) \le \sigma_i(\widehat{L}\widehat{U}) \underbrace{\left(1 + \|\widehat{U}^{-1}\widehat{L}^{-1}\Delta A\|\right)}_{\text{Maximum growth}}$$

$$X = A \text{ and } \Delta X = -\Delta A \quad \Rightarrow \quad \sigma_i(\widehat{L}\widehat{U}) \le \sigma_i(A) \underbrace{\left(1 + \|A^{-1}\Delta A\|\right)}_{\text{Maximum shrinkage}}$$

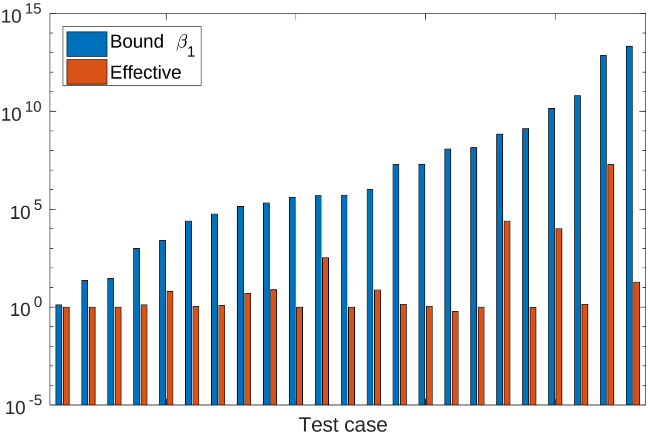
Theorem

$$\varepsilon_k(\widehat{U}^{-1}\widehat{L}^{-1}) \le \beta_1 \varepsilon_k(A^{-1})$$

with

$$\beta_1 = \left(1 + \|A^{-1}\Delta A\|\right)\left(1 + \|\widehat{U}^{-1}\widehat{L}^{-1}\Delta A\|\right)$$

Bound β_1 is pessimistic



Ingredient 2: $\widehat{U}^{-1}\widehat{L}^{-1}\Delta A$ is low-rank if $\widehat{U}^{-1}\widehat{L}^{-1}$ is

Theorem

$$\varepsilon_k(\widehat{U}^{-1}\widehat{L}^{-1}\Delta A) \le \beta_2 \varepsilon_k(\widehat{U}^{-1}\widehat{L}^{-1})$$

with

$$\beta_2 = \frac{\|\widehat{U}^{-1}\widehat{L}^{-1}\| \, \|\Delta A\|}{\|\widehat{U}^{-1}\widehat{L}^{-1}\Delta A\|}$$

Corollary

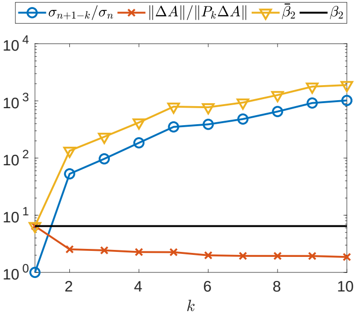
$$\varepsilon_k(E) \le \beta_1 \beta_2 \varepsilon_k(A^{-1})$$

Theorem

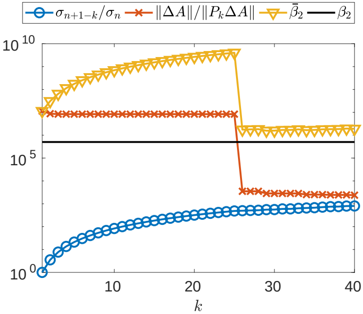
$$\beta_2 \le \bar{\beta}_2 = \frac{\sigma_{n+1-k}(\widehat{L}\widehat{U})}{\sigma_n(\widehat{L}\widehat{U})} \frac{\|\Delta A\|}{\|P_k \Delta A\|}$$

with $P_k = X_k X_k^T$ and X_k the last k left singular vectors of $\widehat{L}\widehat{U}$.

Bound β_2 is also pessimistic



Typical ΔA



Special ΔA