GMRES-based Iterative Refinement in Five Precisions for Sparse Direct Solvers

Theo Mary, CNRS

Joint work with P. Amestoy, A. Buttari, N. Higham, J.-Y. L'Excellent, and B. Vieublé

NLA group meeting, Manchester, ~~26 March~~ 6 May 2020

Iterative refinement in three precisions

Algorithm 1 Iterative refinement in 3 precisions (IR3)

Compute $A = LU$ (precision u_f)
Solve $Ax_0 = b$ (precision u_f)
while not converged **do**
  Compute $r_i = b - Ax_i$ (precision u_r)
  Solve $Ad_i = r_i$ (precision u_f)
  Compute $x_{i+1} = x_i + d_i$ (precision u)
end while

Carson & Higham, *Accelerating the solution of linear systems by iterative refinement in three precisions*, 2018

- Converges to b'err of order u and f'err of order $u + \kappa(A)u_r$ provided that $\kappa(A)u_f \ll 1$

The specificity of sparse direct solvers

- Sparse direct solvers are subject to the fill-in phenomenon: $\text{nnz}(A) \ll \text{nnz}(LU)$
- For an $n \times n$ matrix coming from a regular 3D geometry reordered by nested dissection, $\text{nnz}(A) = O(n)$ and $\text{nnz}(LU) = O(n^{4/3})$

The specificity of sparse direct solvers

- Sparse direct solvers are subject to the fill-in phenomenon: $\text{nnz}(A) \ll \text{nnz}(LU)$
- For an $n \times n$ matrix coming from a regular 3D geometry reordered by nested dissection, $\text{nnz}(A) = O(n)$ and $\text{nnz}(LU) = O(n^{4/3})$

$\Rightarrow$ Therefore, sparse solvers have some important specificities:

- Products with A are much cheaper than triangular solves with $LU \Rightarrow$ setting $u_r \ll u$ has limited impact on performance (even for very small u_r, ex: fp128)

The specificity of sparse direct solvers

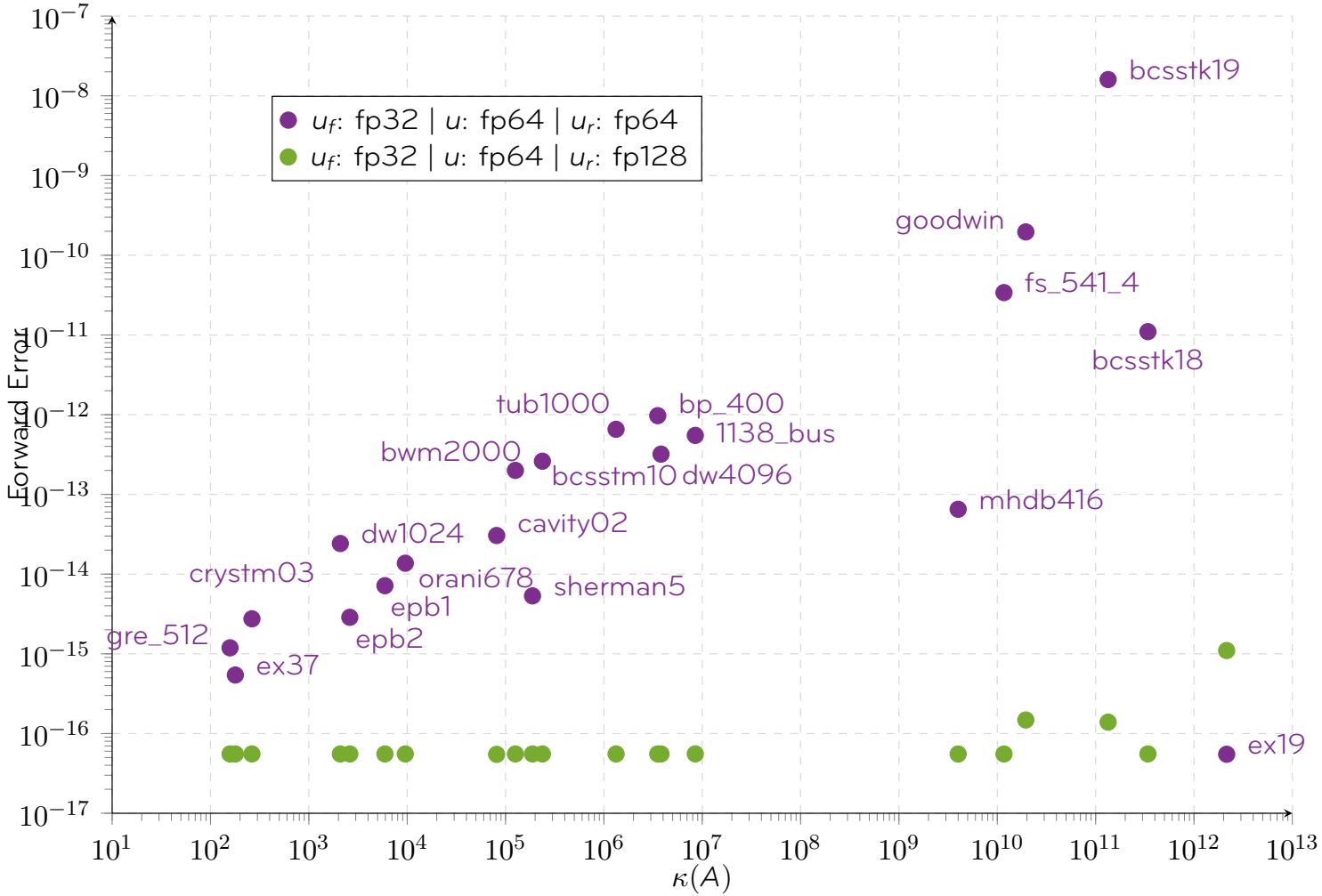
- Sparse direct solvers are subject to the fill-in phenomenon: $\text{nnz}(A) \ll \text{nnz}(LU)$
- For an $n \times n$ matrix coming from a regular 3D geometry reordered by nested dissection, $\text{nnz}(A) = O(n)$ and $\text{nnz}(LU) = O(n^{4/3})$

$\Rightarrow$ Therefore, sparse solvers have some important specificities:

- Products with A are much cheaper than triangular solves with $LU \Rightarrow$ setting $u_r \ll u$ has limited impact on performance (even for very small u_r, ex: fp128)
- Iterative refinement **saves memory**! Compare memory cost of IR3 and standard LU in precision u:

Dense matrix			Sparse matrix		
LU	IR3	ratio	LU	IR3	ratio
n^2u	$n^2u_f + n^2u_r$	$\times 1.25$–2.5	$n^{4/3}u$	$n^{4/3}u_f + nu_r$	$\div 2$–4

Experiments with IR3 in MUMPS (1/2)



Experiments with IR3 in MUMPS (2/2)

Results on Queen_4147 problem

Sequential timings		IR3 with MUMPS in fp32	IR3 with MUMPS in fp64
Analysis		65	65
Factorization		2580	5070
Solve		8×13	2×26
Residual	$u_r = \text{fp64}$	8×0.6	2×0.6
	$u_r = \text{fp128}$	8×17	2×17
Forward error	$u_r = \text{fp64}$	10^{-10}	10^{-10}
	$u_r = \text{fp128}$	10^{-15}	10^{-15}
Storage (GB)	$u_r = \text{fp64}$	85	169
	$u_r = \text{fp128}$	86	170

GMRES-based IR3

Algorithm 2 GMRES-based IR3

Compute $A = LU$ (precision u_f)
Solve $Ax_0 = b$ (precision u_f)
while not converged **do**
  Compute $r_i = b - Ax_i$ (precision u_r)
  Solve $U^{-1}L^{-1}Ad_i = U^{-1}L^{-1}r_i$ by GMRES at precision u with products with A and LU solves at precision u_r
  Compute $x_{i+1} = x_i + d_i$ (precision u)
end while

Carson & Higham, *A new analysis of iterative refinement and its application to accurate solution of ill-conditioned sparse linear systems*, 2017

- Same b'err and f'err bounds as IR3 for a wider range of problems: convergence only requires $\kappa(A)u_f u^{1/2} \ll 1$

The main issue with GMRES-IR3

- GMRES-IR3 assumes the LU triangular solves are performed at precision $u_r \ll u$: this is a major practical issue
- Increases cost per iteration, especially if u_r is fp128
- Requires to cast the LU factors from precision u_f to precision u_r $\Rightarrow$ huge memory increase
  - Alternative: cast LU factors on the fly, chunk by chunk $\Rightarrow$ cost per iteration explodes
- These issues are especially bad for sparse solvers

The main issue with GMRES-IR3

- GMRES-IR3 assumes the LU triangular solves are performed at precision $u_r \ll u$: this is a major practical issue
- Increases cost per iteration, especially if u_r is fp128
- Requires to cast the LU factors from precision u_f to precision u_r $\Rightarrow$ huge memory increase
  - Alternative: cast LU factors on the fly, chunk by chunk $\Rightarrow$ cost per iteration explodes
- These issues are especially bad for sparse solvers

- Our idea: perform the LU solves at precision u_p in between u_f and u_r

$\Rightarrow$ What can we say about the behavior of GMRES-IR in this case?

Introducing a fourth precision u_p

Case $u_p = u$ analyzed in the technical report Higham, *Error Analysis For Standard and GMRES-Based Iterative Refinement in Two and Three-Precisions*, 2019.

- A more general analysis of GMRES-IR shows that a tighter convergence condition is $\kappa(U^{-1}L^{-1}A)\kappa(A)u \ll 1$

  Using $\kappa(U^{-1}L^{-1}A) \leq (1+\kappa(A)u_f)^2$ yields the sufficient condition $\kappa(A)u^{1/3}u_f^{2/3} \ll 1$

Introducing a fourth precision u_p

Case $u_p = u$ analyzed in the technical report Higham, *Error Analysis For Standard and GMRES-Based Iterative Refinement in Two and Three-Precisions*, 2019.

- A more general analysis of GMRES-IR shows that a tighter convergence condition is $\kappa(U^{-1}L^{-1}A)\kappa(A)u \ll 1$
  Using $\kappa(U^{-1}L^{-1}A) \leq (1+\kappa(A)u_f)^2$ yields the sufficient condition $\kappa(A)u^{1/3}u_f^{2/3} \ll 1$

- The analysis carries on to general u_p (not necessarily equal to u) giving the condition: $\boxed{\kappa(A)u_p^{1/3}u_f^{2/3} \ll 1}$

- Note $u_p = u_r$ yields a better bound than the one from Carson & Higham thanks to tighter analysis

Choice of u_p: discussion

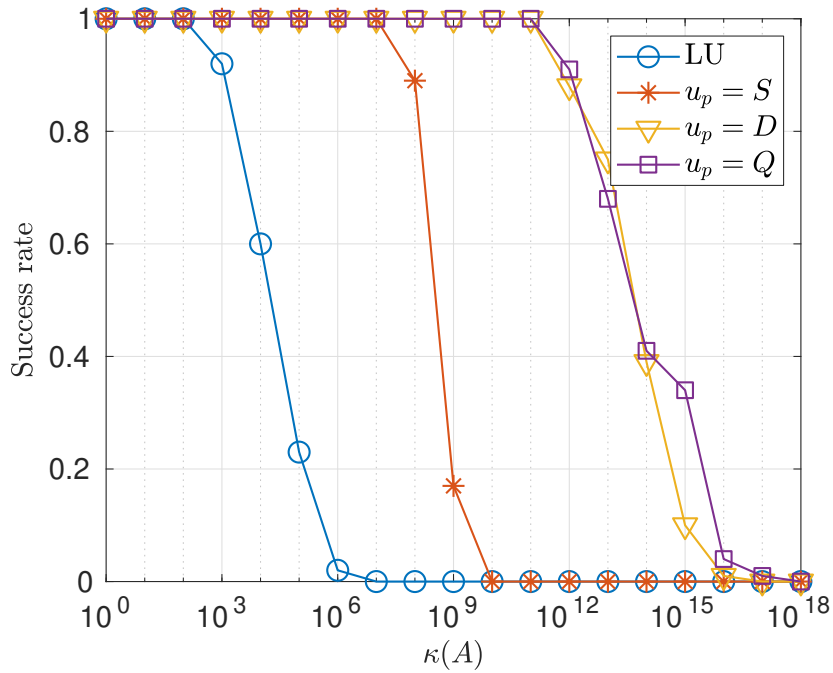
- Take $u_f =$ fp16, $u =$ fp64, $u_r =$ fp128 as an example

Bound on $\kappa(A)$ to ensure fp64 f'err			
LU-IR	GMRES-IR		
	$u_p =$ fp32	$u_p =$ fp64	$u_p =$ fp128
2×10^3	4×10^4	3×10^7	3×10^{13}

- We obtain new variants allowing different compromises between robustness and performance
- Note the case $u_p =$ fp32 has *four* different precisions in play!
  - This is a natural case to consider with GPU tensor cores, which perform the LU factorization in fp16 but (sometimes) store the LU factors in fp32
- We expect the GMRES-IR bounds to be pessimistic because of the use of $\kappa(U^{-1}L^{-1}A) \leq (1 + \kappa(A)u_f)^2$

Experiments with GMRES-IR4 (1/3)

We consider randsvd matrices (mode 2) with varying $\kappa(A)$. For each value of $\kappa(A)$ we generate 100 different matrices and plot the ratio for which IR converged to fp64 f'err



	LU-IR	GMRES-IR		
		$u_p =$ fp32	$u_p =$ fp64	$u_p =$ fp128
Bound	2×10^3	4×10^4	3×10^7	3×10^{13}
Experiment	10^3	10^8	10^{12}	10^{12}

Experiments with GMRES-IR4 (2/3)

Total number of GMRES iterations (averaged over 100 matrices)

$\kappa(A)$	10^0	10^1	10^2	10^3	10^4	10^5	10^6	10^7
$u_p =$ fp128	6.4	7.8	8.0	8.0	9.6	10.2	14.1	12.5
$u_p =$ fp64	6.4	7.8	8.0	8.0	9.6	10.2	14.1	12.5
$u_p =$ fp32	9.5	11.7	12.0	13.1	16.8	23.8	36.5	72.0

For moderate $\kappa(A)$ values:

- GMRES convergence is unaffected by reducing u_p from fp128 to fp64
- Moderate increase of number of iterations if u_p further reduced to fp32, but could still be beneficial:
  - in terms of performance, if iterations are less than doubled
  - in terms of memory, almost always

Experiments with GMRES-IR4 (3/3)

Total number of iterations to reach f'err $< 10^{-14}$

	LU-IR	GMRES-IR		
		$u_p =$ fp32	$u_p =$ fp64	$u_p =$ fp128
saylr1	–	63	20	20
494_bus	443	64	22	22
utm300	94	88	20	22
lund_a	–	85	16	16
nos1	–	150	36	36
fs_183_1	–	–	30	30
plskz362	56	96	26	26
cz308	31	35	14	14
hangGlider_1	–	117	22	22
orbitRaising_1	–	120	26	26
ww_36_pmec_36	–	198	14	14

- Subset of SuiteSparse matrices chosen because LU-IR struggles (on most matrices, LU-IR behaves surprisingly well)
- $u_p =$ fp64 best performance/robustness tradeoff
- $u_p =$ fp32 still able to converge in most cases for half the memory

Block low-rank factorization

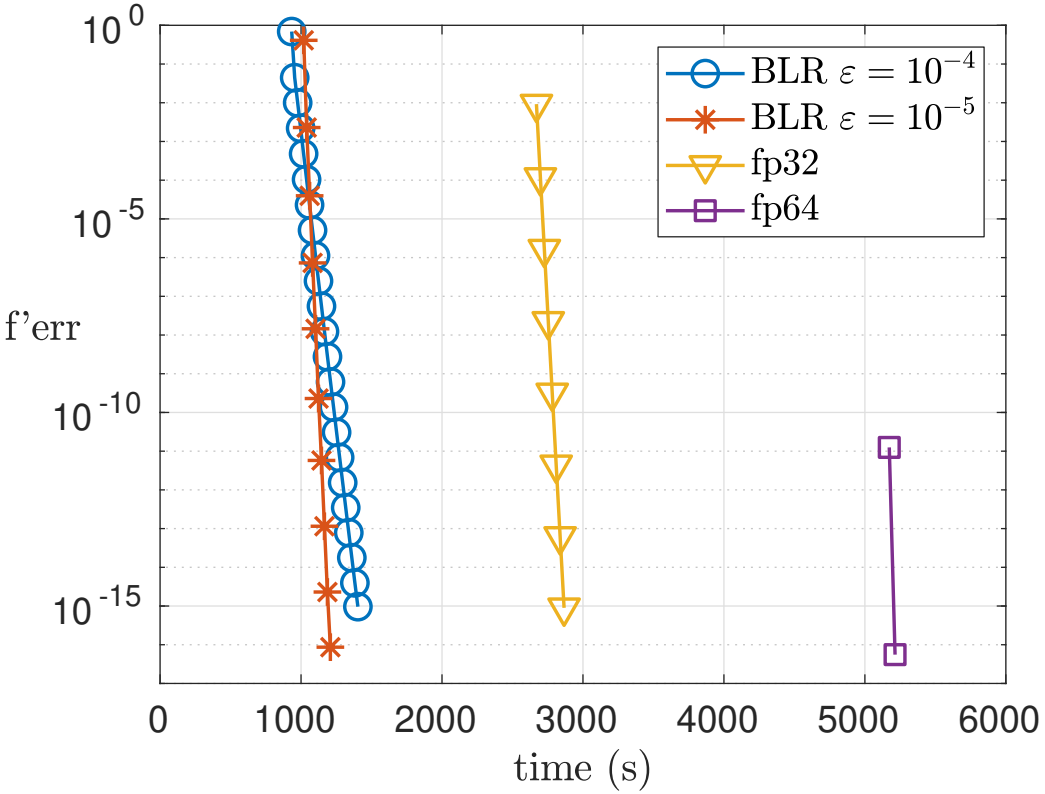
- **Block low-rank** (BLR) approximations can be used to compute approximate LU factors at precision ε ($:= u_f$)

BLR LU is backward stable with respect to ε (Higham and Mary, *Solving Block Low-Rank Linear Systems by LU Factorization is Numerically Stable*, 2019)

Block low-rank factorization

- **Block low-rank** (BLR) approximations can be used to compute approximate LU factors at precision ε ($:= u_f$)

BLR LU is backward stable with respect to ε (Higham and Mary, *Solving Block Low-Rank Linear Systems by LU Factorization is Numerically Stable*, 2019)

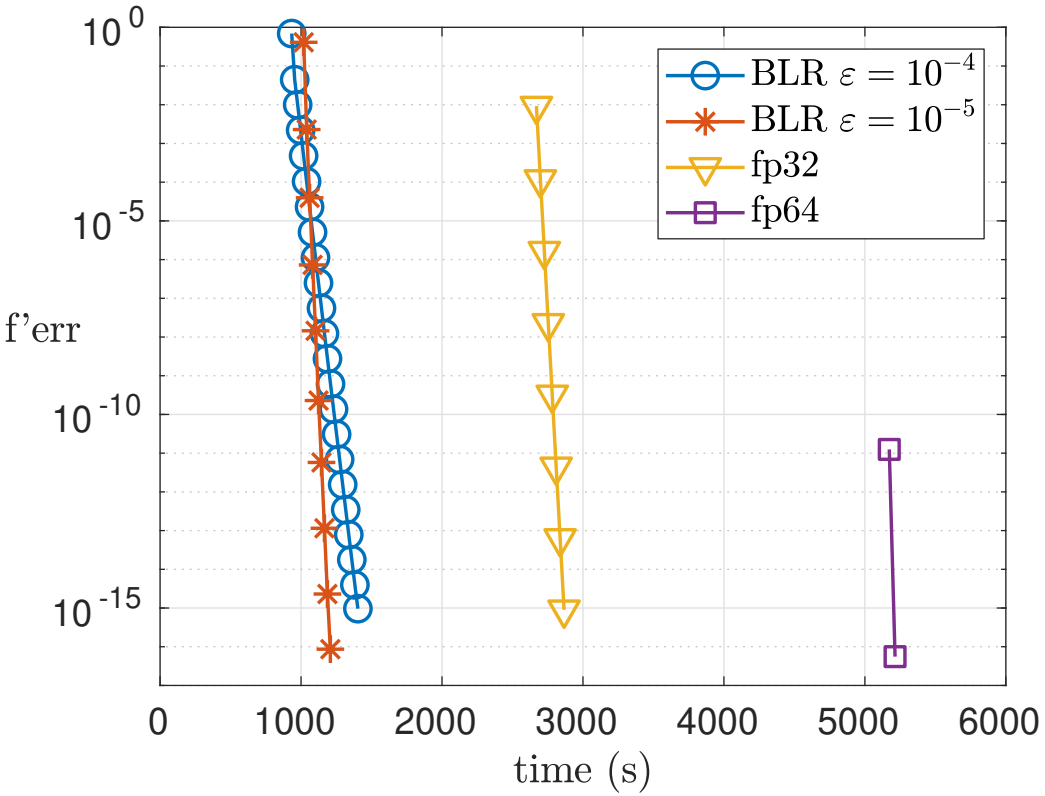


- Sequential timings of LU-IR on Queen_4147

Block low-rank factorization

- **Block low-rank** (BLR) approximations can be used to compute approximate LU factors at precision ε ($:= u_f$)

BLR LU is backward stable with respect to ε (Higham and Mary, *Solving Block Low-Rank Linear Systems by LU Factorization is Numerically Stable*, 2019)



- Sequential timings of LU-IR on Queen_4147
- LU-IR does not converge for $\varepsilon \geq 10^{-3}$
- GMRES-IR (not shown) does converge but is slower than LU-IR with smaller ε

Introducing a fifth precision u_g

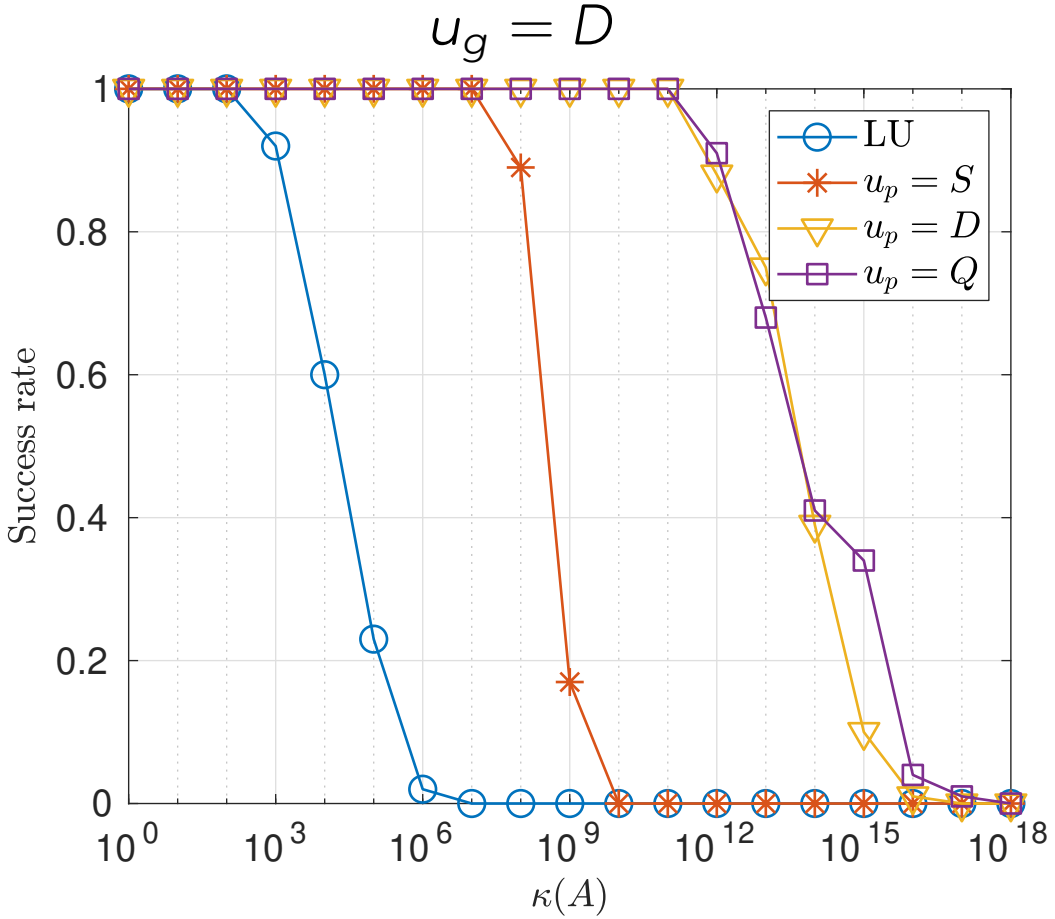
A secondary issue with GMRES-IR is the need to store the (dense) Krylov basis in precision u, which may be significant for sparse direct solvers

What if we use a different precision u_g to store the Krylov basis and perform other GMRES-related computations?

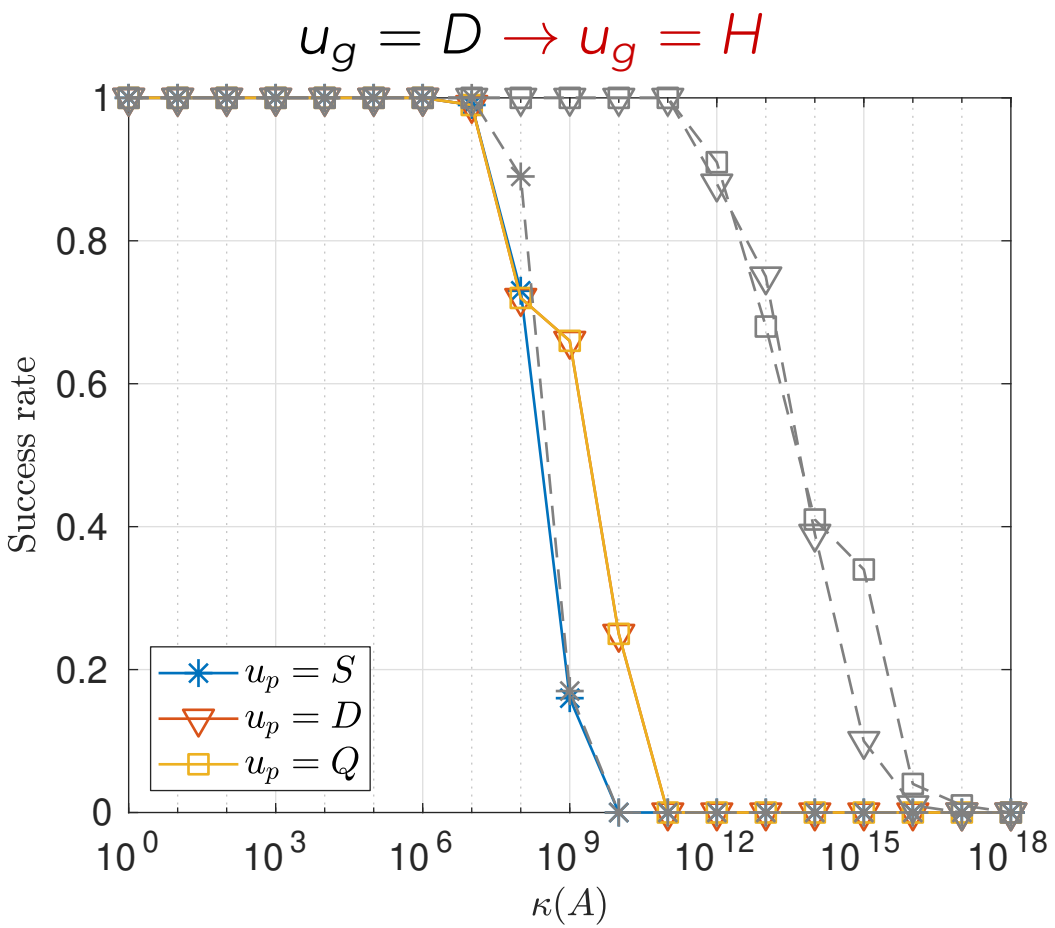
The analysis once again generalizes and gives the condition:

$$\kappa(U^{-1}L^{-1}A)(\kappa(A)u_p + u_g) \ll 1$$

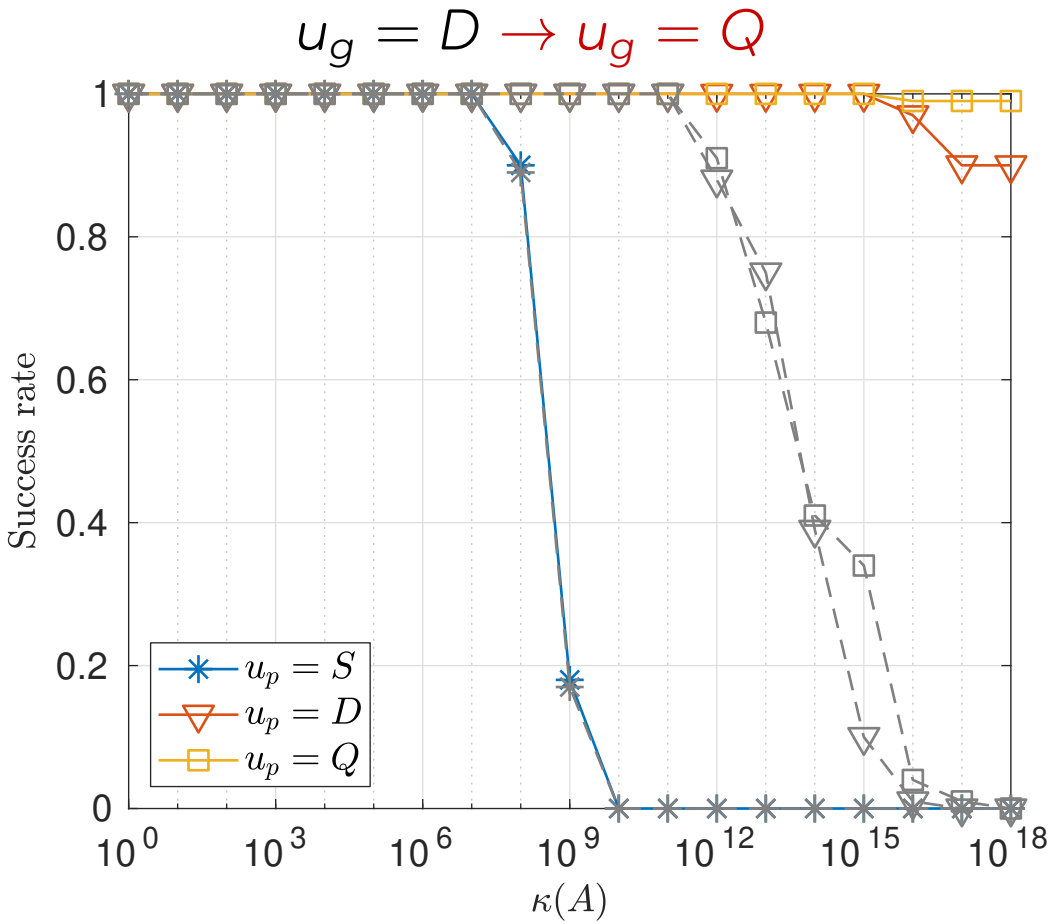
Experiments: influence of u_g



Experiments: influence of u_g



Experiments: influence of u_g



Final algorithm: GMRES-IR5

Algorithm 3 GMRES-IR5

Compute $A = LU$ (precision u_f)
Solve $Ax_0 = b$ (precision u_f)
while not converged **do**
  Compute $r_i = b - Ax_i$ (precision u_r)
  Solve $U^{-1}L^{-1}Ad_i = U^{-1}L^{-1}r_i$ by GMRES at precision u_g with products with A and LU solves at precision u_p
  Compute $x_{i+1} = x_i + d_i$ (precision u)
end while

Final algorithm: GMRES-IR5

Algorithm 4 GMRES-IR5

Compute $A = LU$ (precision u_f)
Solve $Ax_0 = b$ (precision u_f)
while not converged **do**
  Compute $r_i = b - Ax_i$ (precision u_r)
  Solve $U^{-1}L^{-1}Ad_i = U^{-1}L^{-1}r_i$ by GMRES at precision u_g with products with A and LU solves at precision u_p
  Compute $x_{i+1} = x_i + d_i$ (precision u)
end while

Concluding question: is GMRES-IR mathematically and/or numerically equivalent to restarted GMRES?