Harnessing inexactness in scientific computing

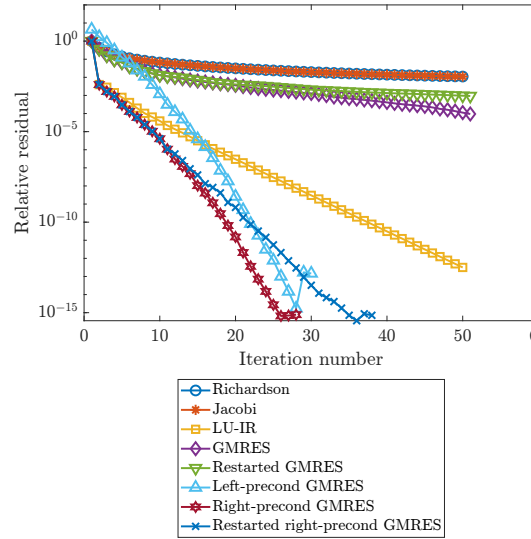
Lecture 9: iterative methods for $Ax = b$

Theo Mary (CNRS)
theo.mary@lip6.fr
https://perso.lip6.fr/Theo.Mary/

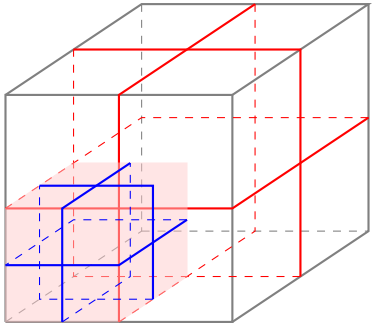
Elisa Riccietti (ENS Lyon)
elisa.riccietti@ens-lyon.fr
https://perso.ens-lyon.fr/elisa.riccietti/

M2 course at ENS Lyon, 2024–2025

Slides available on course webpage



Sparse direct methods complexity



Regular problems (nested dissection)	2D $N \times N$ grid	3D $N \times N \times N$ grid
Nonzeros in original matrix	$O(N^2)$	$O(N^3)$
Nonzeros in factors	$O(N^2 \log N)$	$O(N^4)$
Floating-point ops	$O(N^3)$	$O(N^6)$

Highly superlinear complexities!

Bound on parallelism

Regular problems (nested dissection)	2D $N \times N$ grid	3D $N \times N \times N$ grid
Sequential	$O(N^3)$	$O(N^6)$
Tree parallelism	$O(N^3)$	$O(N^6)$
Node parallelism	$O(N^2)$	$O(N^3)$
Tree and node parallelism	$O(N)$	$O(N^2)$

Top of the tree dominates!

- **Dense systems:**

Theorem 9.3, Higham

$$A + \Delta A = \widehat{L}\widehat{U}, \quad |\Delta A| \leq \gamma_n |\widehat{L}||\widehat{U}|$$

- **Sparse systems:** n should be replaced with the maximum number of operations a given entry of the original matrix can be involved in $\rightarrow$ exploit independence of operations in different branches!
  - **Regular 2D problem**: $n \rightarrow O(N) = O(\sqrt{n})$
  - **Regular 3D problem**: $n \rightarrow O(N^2) = O(n^{2/3})$

Fixed-point iterations

Iterative refinement

GMRES

GMRES-IR

Adaptive precision GMRES

Fixed-point iterations

Iterative refinement

GMRES

GMRES-IR

Adaptive precision GMRES

Fixed-point iteration

- A fixed-point iteration to solve a linear system $Ax = b$ is

$$x^{(k+1)} = x^{(k)} + C(b - Ax^{(k)})$$

- Let $e^{(k)} = x^{(k)} - x$. Then $e^{(k+1)} = (I - CA)e^{(k)} = Re^{(k)}$. R is called the iteration matrix.
- The following statements are equivalent:
  1. The iteration converges when $k \to \infty$
  2. $\lim_{k\to\infty} R^k = 0$
  3. $\rho(R) < 1$, where $\rho(R) = \max_i |\lambda_i(R)|$ is the spectral radius of R
     (proof: $2 \to 3$: $\lambda_i(R)^k \to 0$ for all i; $3 \to 2$: use Jordan normal form)
- How to choose C?

Richardson method

- Take $C = \omega I$

$$x^{(k+1)} = x^{(k)} + \omega(b - Ax^{(k)})$$

- If A is SPD, this method converges if

$$\rho(I - \omega A) = \max_i |1 - \omega\lambda_i(A)| < 1$$

- For $\omega = 1/\lambda_{\max}(A)$,

$$\rho(I - \omega A) = 1 - \frac{1}{\kappa(A)}$$

- Thus after n iterations, the initial error $e^{(0)}$ has been reduced by a factor

$$(1 - \frac{1}{\kappa(A)})^n \approx 1 - \frac{n}{\kappa(A)}$$

  $\rightarrow$ excruciatingly slow convergence if $\kappa(A)$ is large
- With optimal choice of ω, $\rho(I - \omega A)$ can be reduced to $1 - \frac{2}{\kappa(A)+1}$, which is still very bad

Matrix splitting

- Define the splitting $A = M - N$ and the iteration $Mx^{(k+1)} = Nx^{(k)} + b$, i.e.,

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b = x^{(k)} + M^{-1}(b - Ax^{(k)})$$

- We thus have $C = M^{-1}$ and the iteration matrix is $R = I - M^{-1}A$
- How to choose C?
  - Should contain as much information from A as possible
  - Should be easy to invert
- Examples:
  - Richardson: $M = \frac{1}{\omega}I$
  - Jacobi: $M = D \rightarrow$ weighted Jacobi: $M = \frac{1}{\omega}D$
  - Gauss-Seidel: $M = D + L$
  - Block variants
- Convergence speed: $\rho(R) \approx 1 - \frac{1}{\kappa(M^{-1}A)}$, still bad in general
- **MATLAB demo** (Richardson, Jacobi, weighted Jacobi)

Link with gradient descent

- Consider solving $\min_x F(x) = \frac{1}{2}\|Ax - b\|_2^2$
- The gradient is $\nabla F(x) = A^T(Ax - b) = A^T Ax - A^T b = A'x - b'$
- A step of gradient descent is

$$x_{k+1} = x_k - t\nabla F(x_k) = x_k - t(A'x_k - b')$$

$\Rightarrow$ equivalent to Richardson method applied to $A'x = b'$ with $t = \omega$

Fixed-point iterations

Iterative refinement

GMRES

GMRES-IR

Adaptive precision GMRES

while Not converged **do**
  $r_i = b - Ax_i$
  Solve $Ad_i = r_i$
  $x_{i+1} = x_i + d_i$
end while

- If $Ad_i = r_i$ is solved exactly, converges in 1 iteration
- But if knew how to solve $Ad_i = r_i$ exactly, we would rather solve $Ax = b$!

Link with Newton's method

- Consider solving $\min_x F(x) = \frac{1}{2}\|Ax - b\|_2^2$
- The gradient is $\nabla F(x) = A^T(Ax - b) = A^T Ax - A^T b = A'x - b'$
- The Hessian is $\nabla^2 F(x) = A^T A = A'$
- A step of Newton's method is

$$x_{k+1} = x_k - t(\nabla^2 F(x_k))^{-1} \nabla F(x_k) = x_k - t(A')^{-1}(A'x_k - b')$$

$\Rightarrow$ equivalent to iterative refinement applied to $A'x = b'$ with $t = 1$

Convergence of iterative refinement

while Not converged **do**
  $r_i = b - Ax_i$
  Solve $Ad_i \approx r_i$ such that $d_i = A^{-1}r_i + f_i$, $\|f_i\| \leq \phi_i \|d_i\|$
  $x_{i+1} = x_i + d_i$
end while

- $x_i - x = f_i \Rightarrow \|x_i - x\| \lesssim \phi^i \|x_0 - x\|$
- If $Ad_i = r_i$ is solved with a backward error ε, then $\phi \approx \kappa(A)\varepsilon \Rightarrow$ error reduced by factor $(\kappa(A)\varepsilon)^i$ after i iterations
- Much faster than previous methods if $\varepsilon \ll 1$

Convergence of iterative refinement, with rounding errors

while Not converged **do**
  $r_i = b - Ax_i$ in precision $\mathbf{u_r}$
  Solve $Ad_i \approx r_i$ such that $d_i = A^{-1}r_i + f_i$, $\|f_i\| \leq \phi_i \|d_i\|$
  $x_{i+1} = x_i + d_i$ in precision $\mathbf{u}$
end while

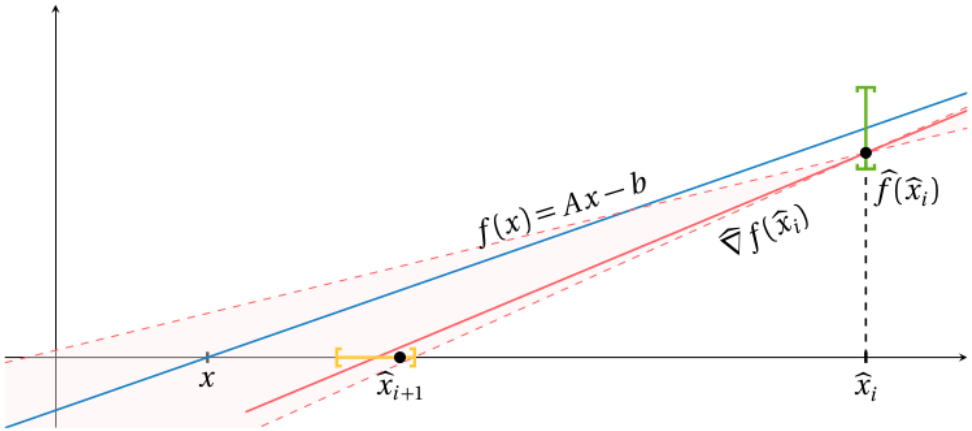
Many variants over the years, depending on choice of precisions and solver for $Ad_i = r_i$

Theorem (simplified from Carson and Higham, 2018)

Under the condition $\phi_i < 1$, the forward error is reduced at each step by a factor ϕ_i until it reaches its limiting value

$$\frac{\|\widehat{x} - x\|}{\|x\|} \lesssim 2n\kappa(A)\mathbf{u_r} + \mathbf{u}$$

Rounding errors in Newton's method illustration



Proof

Assuming

- $\widehat{r}_i = b - A\widehat{x}_i + e_i, \quad \|e_i\| \leq n\mathbf{u_r}(\|A\|\|\widehat{x}_i\| + \|b\|)$
- $k_i = d_i + f_i, \quad \|f_i\| \leq \phi_i\|d_i\|$
- $\widehat{x}_{i+1} = \widehat{x}_i + k_i + g_i, \quad \|g_i\| \leq \mathbf{u}\|\widehat{x}_{i+1}\|$

we have $\widehat{x}_{i+1} = x + A^{-1}e_i + f_i + g_i$ and thus

$$\|x - \widehat{x}_{i+1}\| \lesssim (2n\kappa(A)\mathbf{u_r} + \mathbf{u})\|x\| + \phi_i\|x - \widehat{x}_i\|$$

Choice of solver determines ϕ_i:

- **LU solver:** $d_i = U^{-1}L^{-1}r_i$

$$\frac{\|\widehat{d_i} - d_i\|}{\|\widehat{d_i}\|} \lesssim f(n)\||A^{-1}||\widehat{L}||\widehat{U}\|\mathbf{u_f} \lesssim f(n)\rho_n\kappa(A)\mathbf{u_f}$$

  where ρ_n is the growth factor: the maximum magnitude of the elements appearing during LU factorization

- For stable pivoting strategies (e.g., partial pivoting), ρ_n is almost always $O(1)$

$\Rightarrow$ $\phi_i < 1 \Leftrightarrow \kappa(A)\mathbf{u_f} \ll 1$

LU-based refinement (LU-IR)

Factorize $A = LU$
Solve $Ax_0 = b$ via $x_0 = U^{-1}(L^{-1}b)$
repeat
    $r_i = b - Ax_i$
    Solve $Ad_i = r_i$ via $d_i = U^{-1}(L^{-1}r_i)$
    $x_{i+1} = x_i + d_i$
until converged

LU-based refinement (LU-IR)

Factorize $A = LU$ in precision $\mathbf{u}$
Solve $Ax_0 = b$ via $x_0 = U^{-1}(L^{-1}b)$ in precision $\mathbf{u}$
repeat
  $r_i = b - Ax_i$ in precision $\mathbf{u}^2$
  Solve $Ad_i = r_i$ via $d_i = U^{-1}(L^{-1}r_i)$ in precision $\mathbf{u}$
  $x_{i+1} = x_i + d_i$ in precision $\mathbf{u}$
until converged

Wilkinson (1948) Moler (1967)

- Convergence speed: $\phi = O(\kappa(A)\mathbf{u})$
- Attainable accuracy: $O(\kappa(A)\mathbf{u}^2) = O(\mathbf{u})$

Fixed precision LU-IR

Factorize $A = LU$ in precision **u**
Solve $Ax_0 = b$ via $x_0 = U^{-1}(L^{-1}b)$ in precision **u**
repeat
  $r_i = b - Ax_i$ in precision **u**
  Solve $Ad_i = r_i$ via $d_i = U^{-1}(L^{-1}r_i)$ in precision **u**
  $x_{i+1} = x_i + d_i$ in precision **u**
until converged

Jankowski and Wozniakowski (1977) Skeel (1980)

- Convergence speed: $\phi = O(\kappa(A)\mathbf{u})$
- Attainable accuracy: $O(\kappa(A)\mathbf{u})$
- Implemented in LAPACK and various sparse direct solvers
- Used to remedy unstable factorization, especially related to the use of relaxed pivoting

LU-IR with fp32 LU

Factorize $A = LU$ in precision $\mathbf{u_f}$
Solve $Ax_0 = b$ via $x_0 = U^{-1}(L^{-1}b)$ in precision $\mathbf{u_f}$
repeat
  $r_i = b - Ax_i$ in precision $\mathbf{u}$
  Solve $Ad_i = r_i$ via $d_i = U^{-1}(L^{-1}r_i)$ in precision $\mathbf{u_f}$
  $x_{i+1} = x_i + d_i$ in precision $\mathbf{u}$
until converged

with $\mathbf{u_f} \equiv$ fp32 and $\mathbf{u} \equiv$ fp64

Langou et al (2006) Buttari et al (2007) Baboulin et al (2009)

LU-IR with fp32 LU

Factorize $A = LU$ in precision $\mathbf{u_f}$
Solve $Ax_0 = b$ via $x_0 = U^{-1}(L^{-1}b)$ in precision $\mathbf{u_f}$
repeat
  $r_i = b - Ax_i$ in precision $\mathbf{u}$
  Solve $Ad_i = r_i$ via $d_i = U^{-1}(L^{-1}r_i)$ in precision $\mathbf{u_f}$
  $x_{i+1} = x_i + d_i$ in precision $\mathbf{u}$
until converged

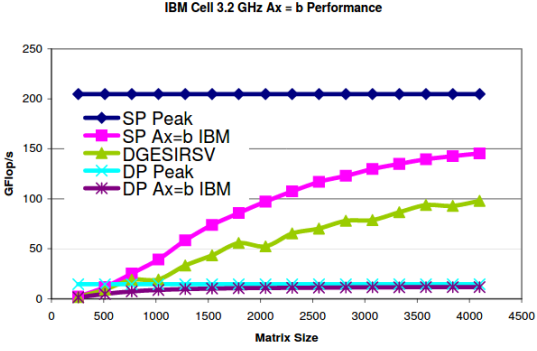
with $\mathbf{u_f} \equiv$ fp32 and $\mathbf{u} \equiv$ fp64

Langou et al (2006) Buttari et al (2007) Baboulin et al (2009)

- For $n \times n$ matrices:
  - $O(n^3)$ flops in fp32
  - $O(n^2)$ flops per iteration in fp64
- Convergence speed: $\phi = O(\kappa(A)\mathbf{u_f})$
- Attainable accuracy: $O(\kappa(A)\mathbf{u})$

LU-IR with CELL processor

CELL processor (2006–2008)
fp64 peak: 21 GFLOPS
fp32 peak: 205 GFLOPS
$\Rightarrow$ 10$\times$ speedup!



Three-precision LU-IR

Factorize $A = LU$ in precision $\mathbf{u_f}$
Solve $Ax_0 = b$ via $x_0 = U^{-1}(L^{-1}b)$ in precision $\mathbf{u_f}$
repeat
  $r_i = b - Ax_i$ in precision $\mathbf{u_r}$
  Solve $Ad_i = r_i$ via $d_i = U^{-1}(L^{-1}r_i)$ in precision $\mathbf{u_f}$
  $x_{i+1} = x_i + d_i$ in precision $\mathbf{u}$
until converged

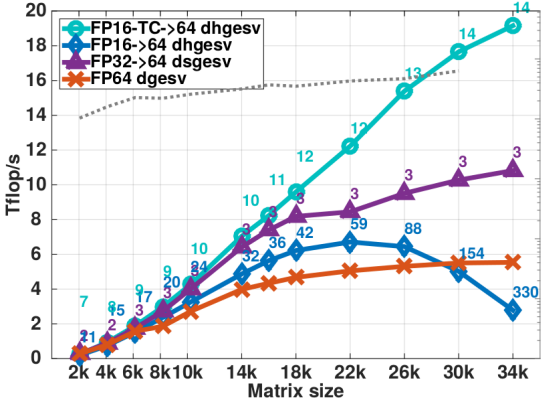
e.g., with $\mathbf{u_f} \equiv$ fp16, $\mathbf{u} \equiv$ fp32, and $\mathbf{u_r} \equiv$ fp64
or $\mathbf{u_f} \equiv$ fp16, $\mathbf{u} \equiv$ fp64, and $\mathbf{u_r} \equiv$ fp128

Carson and Higham (2018)

- Convergence speed: $\phi = O(\kappa(A)\mathbf{u_f})$
- Attainable accuracy: $O(\mathbf{u} + \kappa(A)\mathbf{u_r})$
- Three-precision LU-IR is as general (as modular) as possible

LU-IR with GPU tensor cores

Results from Haidar et al. (2018)



- GPU tensor cores use fp16 storage but fp32 accumulation, providing an accuracy boost which can be critical! (more on this in Lecture 16)

Scaling and shifting fp16 LU

Use of fp16 presents two risks:

- Overflow/underflow in the LU factors
  - $|||L||U||| \leq f(n)\rho_n\|A\| \Rightarrow$ even if A fits in the range, its LU factors may not
  - Higham, Pranesh, Zounon (2019) : two-sided diagonal scaling $A' \leftarrow D_r A D_c$ so that $\|A\| \leq c$
  - To minimize underflow and better utilize the range of fp16, helpful to take c as close as possible to maximum safe value
- Loss of positive definiteness
  - Rounding a posdef A to fp16 might make it indefinite $\Rightarrow$ Cholesky factorization breaks down
  - Higham & Pranesh (2021) : factorize $A + \sigma D$ instead ($D = \mathrm{diag}(A)$, $\sigma = O(u_{16})$)
- **MATLAB demo** (LU-IR)

Fixed-point iterations

Iterative refinement

GMRES

GMRES-IR

Adaptive precision GMRES

Krylov subspace

- We define the Krylov subspace of dimension m associated with B and y as

$$\mathcal{K}_m(B, y) = \operatorname{span}\left\{y, By, B^2y, \ldots, B^{m-1}y\right\}.$$

- Fixed-point iterations build x by successive additive corrections: thus, x can be expressed as a linear combination of vectors belonging to a certain Krylov subspace.
- Proof:
  - $x_m = x_0 + \sum_{k=0}^{m-1} Cr_k$
  - $r_k = (I - AC)^k r_0$
  - $C(I - AC)^k = (I - CA)^k C$
  - $x_m = x_0 + \sum_{k=0}^{m-1} (I - CA)^k Cr_0$

$\Rightarrow$ Conclusion: $x_m \in x_0 + \mathcal{K}_m(CA, Cr_0)$, where

$$\mathcal{K}_m(CA, Cr_0) = \operatorname{span}\left\{Cr_0, CACr_0, (CA)^2 Cr_0, \ldots, (CA)^{m-1} Cr_0\right\}.$$

- Hower, the combination computed fixed-point iteration is not optimal. Can we find the optimal one?

Arnoldi iteration

- The Arnoldi iteration. builds a basis for $\mathcal{K}_m(B, y)$ by repeated multiplication with B and orthonormalization (here, MGS).
- We obtain the Arnoldi relation $BV_m = V_{m+1}H_m$, where
  - $V_{m+1} \in \mathbb{R}^{n\times(m+1)}$ has orthonormal columns
  - $H_m \in \mathbb{R}^{(m+1)\times m}$ is an Hessenberg matrix

$\beta = \|y\|_2$
$v_1 = y/\beta$
for $k = 1\colon m$ **do**
  $w_k = Bv_k$
  for $j = 1\colon k$ **do**
    $h_{jk} = w_k^T v_j$
    $w_k = w_k - h_{jk} v_j$
  end for
  $h_{k+1,k} = \|w_k\|_2$
  If $h_{k+1,k} = 0$, stop.
  $v_{k+1} = w_k / h_{k+1,k}$
end for

(Unpreconditioned) GMRES

The GMRES (generalized miminum residual) method builds a basis for $\mathcal{K}_m(A, r_0)$ with the Arnoldi iteration.

At iteration k:

- Step k of Arnoldi yields: $AV_k = V_{k+1}H_k$
- Find $x_k \in x_0 + \mathcal{K}_k$ minimizing $r_k = Ax_k - b$
  - $x_k \in x_0 + \mathcal{K}_k \Rightarrow x_k = x_0 + V_k y_k$ and thus

$$\begin{aligned} r_k &= b - Ax_k \\ &= b - A(x_0 + V_k y_k) \\ &= r_0 - AV_k y_k \\ &= \beta v_1 - V_{k+1} H_k y_k \\ &= V_{k+1}(\beta e_1 - H_k y_k) \end{aligned}$$

- Stop if $\|r_k\|$ is small enough

$r_0 = b - Ax_0$
$\beta = \|r_0\|$
$v_1 = r_0/\beta$
repeat
  $w_k = Av_k$
  for $j = 1 \colon k$ **do**
    $h_{jk} = q_j^T w_k$
    $w_k = w_k - h_{jk} v_j$
  end for
  $h_{k+1,k} = \|w_k\|$
  $v_{k+1} = w_k / h_{k+1,k}$
  $y_k = \arg\min_y \|\beta e_1 - H_k y\|$.
until $\|r_k\|$ is small enough
$x_k = x_0 + V_k y_k$

MGS-GMRES is backward stable Paige, Rozloznik, Strakos (2006)

Theorem

If unpreconditioned GMRES run in precision u, there exists an iteration $k \leq n$ at which the iterate $\widehat{x}_k$ satisfies

$$(A + \Delta A)\widehat{x}_k = b, \quad \|\Delta A\| \leq O(u)\|A\|$$

and so

$$\|\widehat{x}_k - x\| \lesssim \kappa(A)u\|x\|.$$

This is an existence result: no guarantee that k will be small (might be as large as n!)

Restarted GMRES

$x = x_0$
$r = b - Ax$ and $\beta = \|r\|$
while β is not small enough **do**
  $v_1 = r/\beta$
  for $k = 1 \colon m$ **do**
    $w_k = Av_k$
    for $j = 1 \colon k$ **do**
      $h_{jk} = v_j^T w_k$
      $w_k = w_k - h_{jk} v_j$
    end for
    $h_{k+1,k} = \|w_k\|$
    $v_{k+1} = w_k / h_{k+1,k}$
    $y_k = \arg\min_y \|\beta e_1 - H_k y\|$.
  end for
  $x = x + V_m y_m$
  $r = b - Ax$ and $\beta = \|r\|$
end while

- Cost of SpMV: $\mathrm{nnz}(A)$ per iteration
- Cost of building $\mathcal{K}_m$:
  - $O(nm^2)$ flops
  - $O(nm)$ storage

  $\Rightarrow$ unaffordable as m increases
- Restarted GMRES: limit size of Krylov basis to small m
  - Stop after m inner iterations
  - Update x and restart
  - Repeat until $\|r\|$ is small enough
- Slower convergence but bounded cost per iteration
- **MATLAB demo** (GMRES, restarted GMRES)

Preconditioned GMRES

- Convergence of GMRES strongly depends on matrix $\Rightarrow$ preconditioning is needed
- Preconditioned GMRES: apply GMRES to

$$MAx = Mb \qquad \text{(left preconditioning)}$$
$$AMy = b, \quad My = x \qquad \text{(right preconditioning)}$$

  where $M \approx A^{-1}$
- Some examples of preconditioners:
  - Jacobi: $M = \text{diag}(A)^{-1}$, Gauss-Seidel, etc.
  - LU preconditioner: $M = U^{-1}L^{-1} \Rightarrow$ requires triangular solves at each iteration. Exact LU is expensive and would not require an iterative method. Use approximate LU instead: low precision, incomplete factorization, block low-rank approximations (Lecture 14), etc.
- Right preconditioning does not change the residual and is thus often preferred: $\|MAy - b\| = \|Ax - b\| \neq \|MAx - Mb\|$
- **MATLAB demo** (preconditioned GMRES)

Fixed-point iterations

Iterative refinement

GMRES

GMRES-IR

Adaptive precision GMRES

GMRES-based IR (GMRES-IR)

repeat
$r_i = b - Ax_i$ in precision $\mathbf{u}_r$
Solve $Ad_i = r_i$ with GMRES in precision $\mathbf{u_g}$
$x_{i+1} = x_i + d_i$ in precision $\mathbf{u}$
until converged

- GMRES is stable $\Rightarrow \phi = \kappa(A)\mathbf{u_g}$
- Can be interpreted as mixed precision restarted GMRES
- Inner GMRES is unpreconditioned $\Rightarrow$ might take too many iterations!

Factorize $A = LU$ in precision $\mathbf{u_f}$
Solve $Ax_0 = b$ via $x_0 = U^{-1}(L^{-1}b)$ in precision $\mathbf{u_f}$
repeat
  $r_i = b - Ax_i$ in precision $\mathbf{u_r}$
  Solve $U^{-1}L^{-1}Ad_i = U^{-1}L^{-1}r_i$ with GMRES in precision $\mathbf{u_g}$
  $x_{i+1} = x_i + d_i$ in precision $\mathbf{u}$
until converged

Rationale for replacing LU solver by preconditioned GMRES:

- GMRES can be asked to converge to accuracy $\mathbf{u_g} \ll \mathbf{u_f}$
- $\kappa(\widetilde{A}) = \kappa(U^{-1}L^{-1}A)$ often smaller than $\kappa(A)$
- If $\widetilde{A}d_i = \widetilde{r}_i$ were solved with accuracy $\phi_i = \kappa(\widetilde{A})\mathbf{u_g}$, convergence condition would be improved from $\kappa(A)\mathbf{u_f} < 1$ to $\kappa(\widetilde{A})\mathbf{u_g} < 1 \ldots$
- . . . but there is a catch!

Stability of preconditioned GMRES

- As mentioned previously unpreconditioned GMRES is stable... but what about preconditioned GMRES?
- One key difference: the matrix–vector products are performed with $\widehat{U}^{-1}$, $\widehat{L}^{-1}$, and A separately, not directly with $\widetilde{A}$ (which is never formed)
  - $y = \widetilde{A}x \Rightarrow \|\widehat{y} - y\| \leq n\mathbf{u_g}\|\widetilde{A}\|\|x\|$
  - $y = \widehat{U}^{-1}\widehat{L}^{-1}Ax \Rightarrow \|\widehat{y} - y\| \leq f(n)u\|A\|\|\widehat{U}^{-1}\|\|\widehat{L}^{-1}\|\|x\| \lesssim \kappa(A)f(n)\mathbf{u_g}\|\widetilde{A}\|\|x\|$

  $\Rightarrow$ extra $\kappa(A)$ term appears, it is as if GMRES was run in "precision" $\kappa(A)\mathbf{u_g}$

- Overall: $\phi_i = \kappa(\widetilde{A})\kappa(A)\mathbf{u_g}$

$\Rightarrow$ Potentially better than $\phi = \kappa(A)\mathbf{u_f}$ if $\kappa(\widetilde{A})$ is small

- We have the (pessimistic) bound $\kappa(\widetilde{A}) \leq \mathbf{u_f}^2\kappa(A)^2$

Factorize $A = LU$ in precision $\mathbf{u_f}$
Solve $Ax_0 = b$ via $x_0 = U^{-1}(L^{-1}b)$ in precision $\mathbf{u_f}$
repeat
  $r_i = b - Ax_i$ in precision $\mathbf{u_r}$
  Solve $U^{-1}L^{-1}Ad_i = U^{-1}L^{-1}r_i$ with GMRES in precision $\mathbf{u_g}$
    except products with $U^{-1}L^{-1}A$ in precision $\mathbf{u_p}$
  $x_{i+1} = x_i + d_i$ in precision $\mathbf{u}$
until converged

- Perform matvecs with $\widetilde{A}$ in precision $\mathbf{u_p} \leq \mathbf{u_g}$ to reduce $\kappa(A)$ dependence
- Convergence speed: $\phi = O(\kappa(\widetilde{A})(\mathbf{u_g} + \kappa(A)\mathbf{u_p})) = O(\kappa(A)^2\mathbf{u_f}^2(\mathbf{u_g} + \kappa(A)\mathbf{u_p}))$
- Attainable accuracy: $O(\mathbf{u} + \kappa(A)\mathbf{u_r})$
- Modular error analysis (parameterize every line by independent precisions) reveals the numerical structure of the algorithm!

Meaningful combinations

With five arithmetics (fp8, fp16, fp32, fp64, fp128) there are over **3000 different combinations** of GMRES-IR5!

They are not all relevant !

Meaningful combinations: those where none of the precisions can be lowered without worsening either the limiting accuracy or the convergence condition.

Filtering rules

- $\mathbf{u}^2 \leq \mathbf{u_r} \leq \mathbf{u} \leq \mathbf{u_f}$
- $\mathbf{u_p} \leq \mathbf{u_g}$
- $\mathbf{u_p} < \mathbf{u_f}$
- $\mathbf{u_p} < \mathbf{u}$, $\mathbf{u_p} = \mathbf{u}$, $\mathbf{u_p} > \mathbf{u}$ all possible
- $\mathbf{u_g} \geq \mathbf{u}$
- $\mathbf{u_g} < \mathbf{u_f}$, $\mathbf{u_g} = \mathbf{u_f}$, $\mathbf{u_g} > \mathbf{u_f}$ all possible

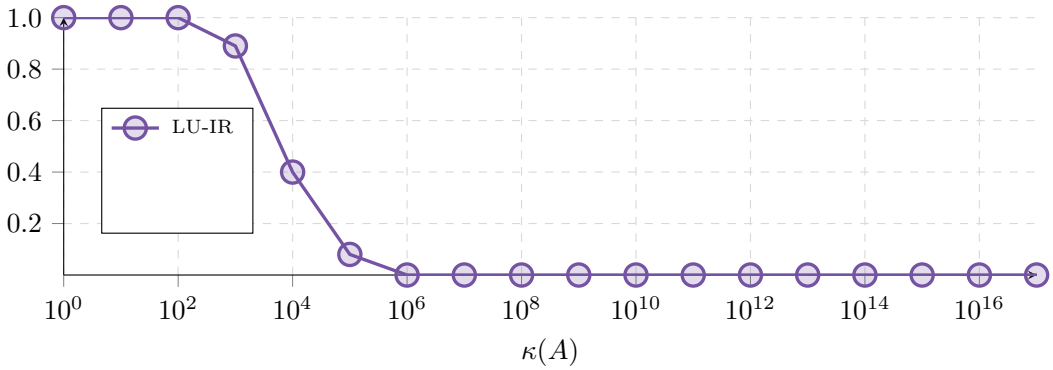
Meaningful combinations of GMRES-IR5 for $\mathbf{u_f} \equiv$ fp16 and $\mathbf{u} \equiv$ fp64

$\mathbf{u_g}$	$\mathbf{u_p}$	Convergence Condition $\max(\kappa(A))$
LU-IR		2×10^3
fp8	fp32	8×10^3
fp16	fp32	4×10^4
fp16	fp64	9×10^4
fp32	fp64	8×10^6
fp64	fp64	3×10^7
fp64	fp128	2×10^{11}

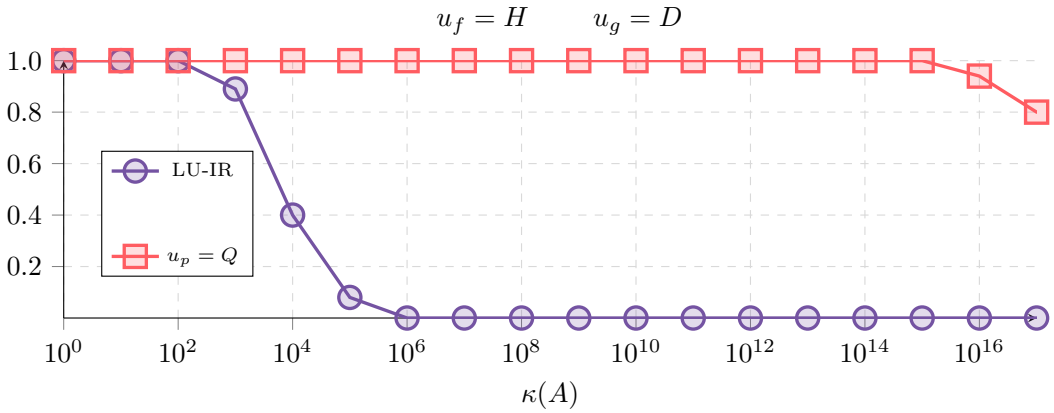
Six meaningful combinations $\Rightarrow$ **flexible** precisions choice to fit at best the **hardware constraints** and the **problem difficulty**.

Take 100 random matrices with specified $\kappa(A)$ and measure the success rate: the percentage of matrices for which GMRES-IR5 converges to a small forward error

$u_f = H \qquad u_g = D$

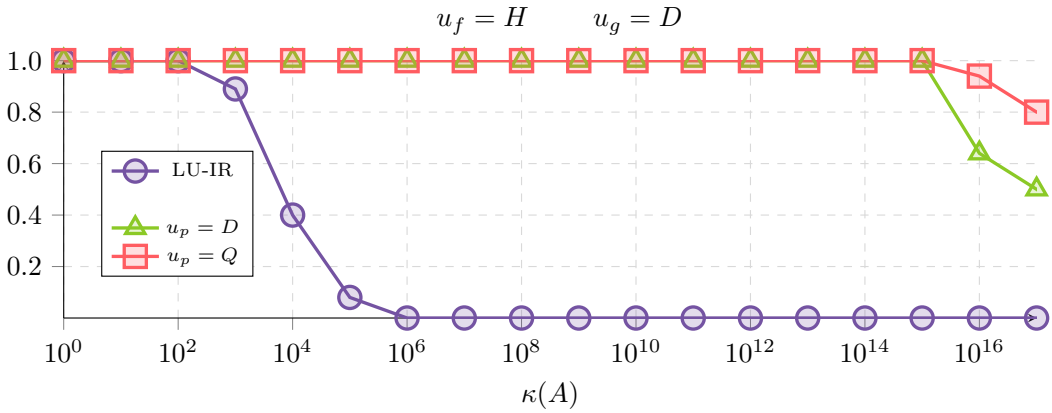


Take 100 random matrices with specified $\kappa(A)$ and measure the success rate: the percentage of matrices for which GMRES-IR5 converges to a small forward error

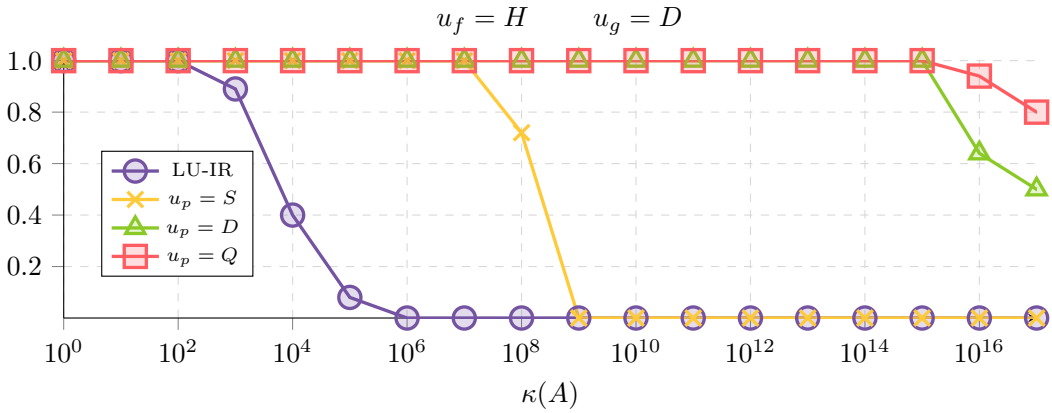


Experimental results

Take 100 random matrices with specified $\kappa(A)$ and measure the success rate: the percentage of matrices for which GMRES-IR5 converges to a small forward error

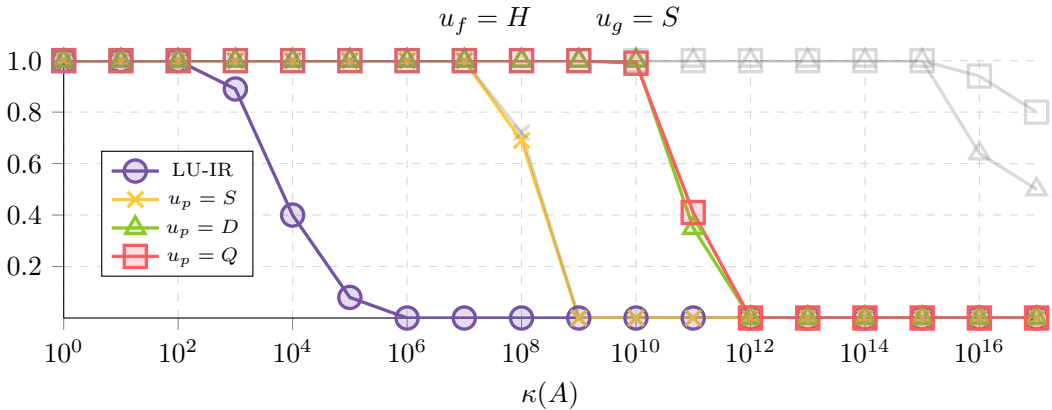


Take 100 random matrices with specified $\kappa(A)$ and measure the success rate: the percentage of matrices for which GMRES-IR5 converges to a small forward error

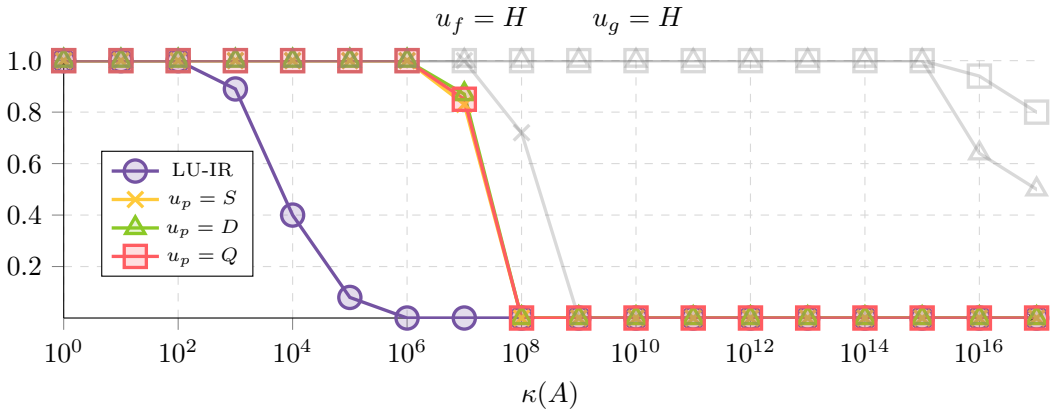


Experimental results

Take 100 random matrices with specified $\kappa(A)$ and measure the success rate: the percentage of matrices for which GMRES-IR5 converges to a small forward error



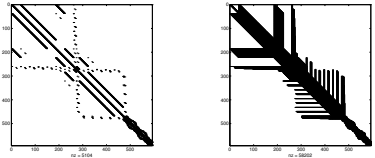
Take 100 random matrices with specified $\kappa(A)$ and measure the success rate: the percentage of matrices for which GMRES-IR5 converges to a small forward error



Memory consumption of IR

- **Dense systems:** cannot overwrite A with LU factors, need to keep A for evaluating residuals $\Rightarrow$ IR costs *more* memory
- **Sparse systems:** typically $\mathrm{nnz}(LU) \gg \mathrm{nnz}(A)$, so original copy of A is negligible; LU-IR allows for storing the LU factors in low precision and thus *saves memory*!

Original matrix *Factorized matrix*



- Same applies to the GMRES basis, which requires mn entries but can be stored in low precision with GMRES-IR

Comparison on industrial problems

A: fp64 LU B: fp32 LU + LU-IR C: fp32 LU + GMRES-IR

Matrix	time (s)			memory (GB)		
	A	B	C	A	B	C
ElectroPhys10M	265.2	**154.0**	166.5	272.0	**138.0**	171.3
Bump_2911	205.4	**129.3**	144.5	135.7	**68.4**	77.8
DrivAer6M	91.8	**67.6**	77.9	81.6	**41.7**	52.9
Queen_4147	284.2	**165.2**	184.7	178.0	**89.8**	114.5
tminlet3M	294.5	**136.2**	157.9	241.1	**121.0**	169.9
perf009ar	**46.1**	57.5	52.0	55.6	**28.9**	38.1
elasticity-3d	156.7	—	**118.6**	153.0	—	**103.6**
lfm_aug5M	536.2	**254.5**	269.3	312.0	**157.0**	187.5
Long_Coup_dt0	67.2	**46.6**	49.0	52.9	**26.7**	33.1
CarBody25M	**62.9**	—	109.8	77.6	—	**54.3**
thmgaz	97.6	**65.4**	79.8	192.0	**97.7**	141.7

- Up to 2× **time and memory reduction**, even for ill-conditioned problems
- GMRES-IR usually more expensive than LU-IR, but more robust ⇒ overall good compromise on a wide range of matrices

Fixed-point iterations

Iterative refinement

GMRES

GMRES-IR

Adaptive precision GMRES

Adaptive precision GMRES

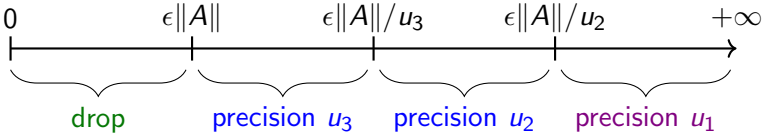
- So far, we have considered introducing mixed precision via restarts and preconditioning, but not in GMRES itself
- Does it make sense to vary precisions within unpreconditioned, unrestarted GMRES?
- YES, both:
  - Spatially: change precisions across different matrix coefficients
  - Temporally: change precisions across different iterations

Adaptive precision SpMV, reminder

In Lecture 2 we saw how to compute SpMV using p precisions $u_1 < \varepsilon < u_2 < \ldots < u_p$ by partitioning $A = \sum_{k=1}^{p} A^{(k)}$ where

$$a_{ij}^{(k)} = \begin{cases} \mathrm{fl}_k(a_{ij}) & \text{if } |a_{ij}| \in (\varepsilon\|A\|/u_k, \varepsilon\|A\|/u_{k+1}] \\ 0 & \text{otherwise} \end{cases}$$

$\Rightarrow$ the precision of each element is chosen inversely proportional to its magnitude



$$\begin{pmatrix} \times & & \times \\ \times & \times & \\ & \times & \times \end{pmatrix} = \begin{pmatrix} d & & \\ & d & \\ & & d \end{pmatrix} + \begin{pmatrix} & & s \\ & & \\ & s & \end{pmatrix} + \begin{pmatrix} h & & \\ & & \\ & & \end{pmatrix}$$

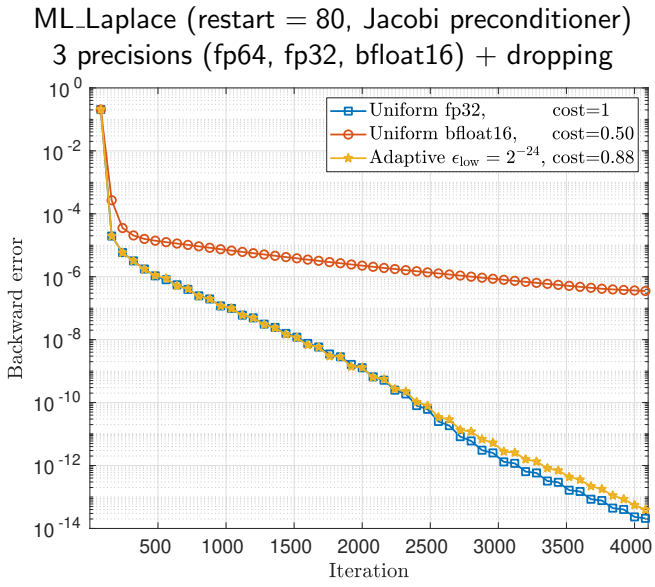
GMRES-IR with adaptive precision SpMV

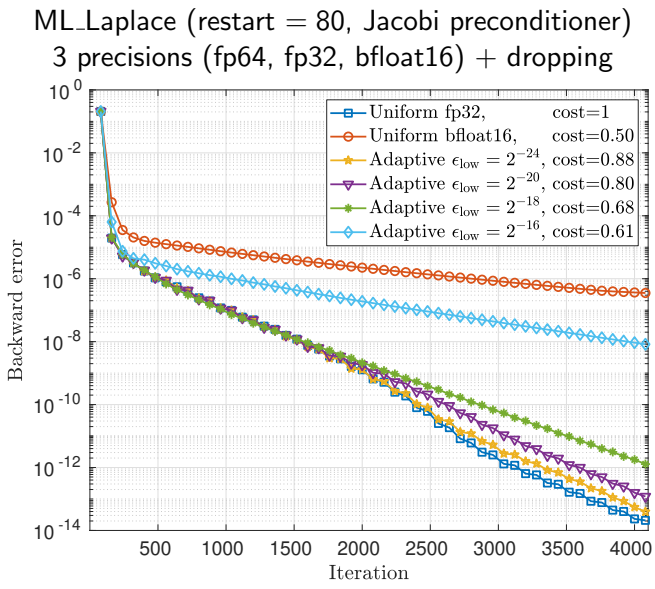
Build adaptive precision representation $\widetilde{A}$
$x = x_0$
$r = b - Ax \rightarrow$ high precision $\mathbf{u}$
$\beta = \|r\|$
while β is not small enough **do**
  $v_1 = r/\beta$
  for $k = 1\colon m$ **do**
    $w_k = Av_k \rightarrow$ adaptive precision $\mathbf{u_g}$
    for $j = 1\colon k$ **do**
      $h_{jk} = v_j^T w_k$
      $w_k = w_k - h_{jk} v_j$
    end for
    $h_{k+1,k} = \|w_k\|$
    $v_{k+1} = w_k / h_{k+1,k}$
    $y_k = \arg\min_y \|\beta e_1 - H_k y\|$.
  end for
  $x = x + V_m y_m$
  $r = b - Ax \rightarrow$ high precision $\mathbf{u}$
  $\beta = \|r\|$
end while

Benefits of adaptive precision SpMV in a GMRES context:

- SpMV is one of the most costly operations so accelerating it is useful
- Adaptive representation $\widetilde{A}$ does not depend on vector $v_k \rightarrow$ only need to build $\widetilde{A}$ once at the beginning
- As mentioned previously the inner GMRES can be switched to low precision $\mathbf{u_g} \rightarrow$ can build $\widetilde{A}$ at precision $\mathbf{u_g} \ll \mathbf{u}$
- Moreover $\widetilde{A}$ can target any accuracy $\varepsilon_{\mathbf{g}}$ not necessarily corresponding to an available arithmetic

GMRES-IR with adaptive precision SpMV




ML_Laplace (restart = 80, Jacobi preconditioner)
3 precisions (fp64, fp32, bfloat16) + dropping
Uniform fp32, cost=1
Uniform bfloat16, cost=0.50
Adaptive $\epsilon_{low} = 2^{-24}$, cost=0.88
Adaptive $\epsilon_{low} = 2^{-20}$, cost=0.80
Adaptive $\epsilon_{low} = 2^{-18}$, cost=0.68
Adaptive $\epsilon_{low} = 2^{-16}$, cost=0.61
Backward error
Iteration
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
500
1000
1500
2000
2500
3000
3500
4000

Relaxed GMRES

- How much error can the SpMV $w_k = Av_k$ at iteration k tolerate?
- Assume inexact SpMV satisfying a relaxed bound $w_k = (A + E_k)v_k$ (note that E_k depends on v_k and thus on k)
- Ignoring sources of inexactness other than the SpMV (e.g., rounding errors in orthonormalization), we obtain a modified Arnoldi relation

$$\widetilde{A}_k V_k = V_{k+1} H_k, \quad \text{where } \widetilde{A}_k = A + G_k V_k^T \quad \text{and } G_k = [E_1 v_1, \ldots, E_k v_k]$$

- Therefore k steps of relaxed GMRES are equivalent to k steps of exact GMRES applied to $\widetilde{A}_k \rightarrow$ since GMRES is monotone the relaxed residual $\widetilde{r}_k = b - \widetilde{A}_k x_k$ decreases... but how far is it from the true residual $r_k = b - Ax_k$?
- $\widetilde{r}_k - r_k = (\widetilde{A}_k - A)x_k = G_k V_k^T x_k$, where $x_k = V_k y_k$

$\Rightarrow$ $\widetilde{r}_k - r_k = G_k y_k = \sum_{i=1}^{k} y_{k,i} E_i v_i$

- $\widetilde{r}_k - r_k = G_k y_k = \sum_{i=1}^{k} y_{k,i} E_i v_i$
- $y_{k,i} \propto \|\widetilde{r}_{k-1}\|$ (intuition)
- Formal result:

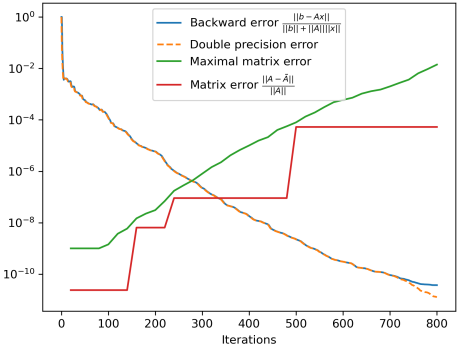
Relaxed GMRES (Giraud, Gratton, Langou, 2009)

Stability up to $O(\varepsilon)$ is maintained if, at each iteration k, the matvec is performed with $\widetilde{A}_k = A + E_k$ such that

$$\frac{\|E_k\|}{\|A\|} \leq \frac{1}{n\kappa(A)} \frac{\|b\|}{\|\widetilde{r}_{k-1}\|} \varepsilon$$

- Matvec precision can be reduced to be inversely proportional to the residual norm
  $\Rightarrow$ lower and lower precision as iterations progress

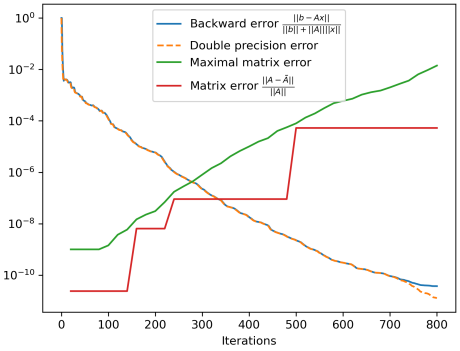
Relaxed GMRES with adaptive SpMV



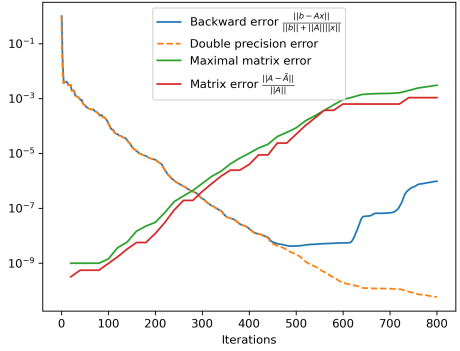
Relaxed

- Can switch to precision $\mathbf{u_g}$ as soon as $\frac{c\varepsilon}{||\widetilde{r}_{k-1}||} \geq \mathbf{u_g} \Rightarrow$ the more precisions available, the more fine tuning we can do

Relaxed GMRES with adaptive SpMV



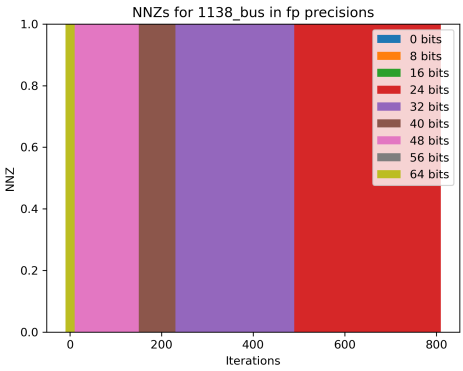
Relaxed



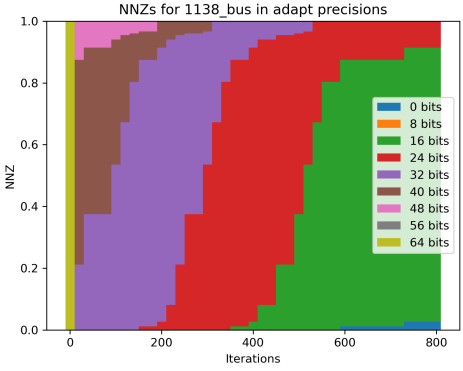
Relaxed+adaptive

- Can switch to precision $\mathbf{u_g}$ as soon as $\frac{c\varepsilon}{\|\widetilde{r}_{k-1}\|} \geq \mathbf{u_g} \Rightarrow$ the more precisions available, the more fine tuning we can do
- Adaptive precision SpMV can allow continuous variations of accuracy $\varepsilon_{\mathbf{g}}$!

Relaxed GMRES with adaptive SpMV



Relaxed



Relaxed+adaptive

- Need to reevaluate the potential of relaxed GMRES in light of evolutions in hardware (more precisions) and algorithms (adaptive SpMV)

$\Rightarrow$ Internship/PhD available (see here for details)