A new backward error analysis framework for GMRES and its application to GMRES preconditioned with MUMPS in mixed precision

Speaker: Bastien Vieublé
Co-authors: Patrick Amestoy, Alfredo Buttari, Nick Higham, Jean-Yves L'Excellent, and Théo Mary

20/06/2023

The University of Manchester, UK

What is GMRES?

Throughout the presentation, we focus on the Generalized Minimal RESidual (GMRES) algorithm.

Algorithm: GMRES(A, b, x_0, τ)

Require: $A \in \mathbb{R}^{n \times n}$, $b, x_0 \in \mathbb{R}^n$, $\tau \in \mathbb{R}$

$r_0 = b - Ax_0$
$\beta = \|r_0\|,\ v_1 = r_0/\beta,\ k = 1$
repeat
  $w_k = Av_k$
  for $i = 1, \ldots, k$ **do**
    $h_{i,k} = v_i^T w_k$
    $w_k = w_k - h_{i,k} v_i$
  end for
  $h_{k+1,k} = \|w_k\|, v_{k+1} = w_k / h_{k+1,k}$
  $V_k = [v_1, \ldots, v_k]$
  $H_k = \{h_{i,j}\}_{1 \le i \le j+1; 1 \le j \le k}$
  $y_k = \mathrm{argmin}_y \|\beta e_1 - H_k y\|$
  $k = k + 1$
until $\|\beta e_1 - H_k y_k\| \le \tau$
$x_k = x_0 + V_k y_k$

What is GMRES?

Throughout the presentation, we focus on the Generalized Minimal RESidual (GMRES) algorithm.

➤ GMRES = Krylov-based iterative solver for the solution of general square linear systems $Ax = b$.

Algorithm: GMRES(A, b, x_0, τ)

Require: $A \in \mathbb{R}^{n\times n}$, $b, x_0 \in \mathbb{R}^n$, $\tau \in \mathbb{R}$

$r_0 = b - Ax_0$

$\beta = \|r_0\|,\ v_1 = r_0/\beta,\ k = 1$

repeat

  $w_k = Av_k$

  for $i = 1, \dots, k$ **do**

    $h_{i,k} = v_i^T w_k$

    $w_k = w_k - h_{i,k} v_i$

  end for

  $h_{k+1,k} = \|w_k\|, v_{k+1} = w_k/h_{k+1,k}$

  $V_k = [v_1, \dots, v_k]$

  $H_k = \{h_{i,j}\}_{1\leq i\leq j+1; 1\leq j\leq k}$

  $y_k = \textbf{argmin}_y \|\beta e_1 - H_k y\|$

  $k = k + 1$

until $\|\beta e_1 - H_k y_k\| \leq \tau$

$x_k = x_0 + V_k y_k$

What is GMRES?

Throughout the presentation, we focus on the Generalized Minimal RESidual (GMRES) algorithm.

- ➤ GMRES = Krylov-based iterative solver for the solution of **general square linear systems $Ax = b$**.
- ➤ Computes iteratively an orthonormal **Krylov basis V_k** through an Arnoldi process.

Algorithm: GMRES(A, b, x_0, τ)

Require: $A \in \mathbb{R}^{n \times n}$, $b, x_0 \in \mathbb{R}^n$, $\tau \in \mathbb{R}$

$r_0 = b - Ax_0$
$\beta = \|r_0\|, \; v_1 = r_0/\beta, \; k = 1$
repeat
  $w_k = Av_k$
  for $i = 1, \ldots, k$ **do**
    $h_{i,k} = v_i^T w_k$
    $w_k = w_k - h_{i,k} v_i$
  end for
  $h_{k+1,k} = \|w_k\|, v_{k+1} = w_k / h_{k+1,k}$
  $V_k = [v_1, \ldots, v_k]$
  $H_k = \{h_{i,j}\}_{1 \le i \le j+1; 1 \le j \le k}$
  $y_k = \mathbf{argmin}_y \|\beta e_1 - H_k y\|$
  $k = k + 1$
until $\|\beta e_1 - H_k y_k\| \le \tau$
$x_k = x_0 + V_k y_k$

What is GMRES?

Throughout the presentation, we focus on the Generalized Minimal RESidual (GMRES) algorithm.

- GMRES = Krylov-based iterative solver for the solution of **general square linear systems $Ax = b$**.
- Computes iteratively an orthonormal **Krylov basis V_k** through an Arnoldi process.
- Chooses the vector x_k in $span\{V_k\}$ that **minimizes $\|Ax_k - b\|$**.

Algorithm: GMRES(A, b, x_0, τ)

Require: $A \in \mathbb{R}^{n\times n}$, $b, x_0 \in \mathbb{R}^n$, $\tau \in \mathbb{R}$

$r_0 = b - Ax_0$
$\beta = \|r_0\|,\ v_1 = r_0/\beta,\ k = 1$
repeat
  $w_k = Av_k$
  for $i = 1, \ldots, k$ **do**
    $h_{i,k} = v_i^T w_k$
    $w_k = w_k - h_{i,k} v_i$
  end for
  $h_{k+1,k} = \|w_k\|, v_{k+1} = w_k / h_{k+1,k}$
  $V_k = [v_1, \ldots, v_k]$
  $H_k = \{h_{i,j}\}_{1\le i\le j+1; 1\le j\le k}$
  $y_k = \text{argmin}_y \|\beta e_1 - H_k y\|$
  $k = k + 1$
until $\|\beta e_1 - H_k y_k\| \le \tau$
$x_k = x_0 + V_k y_k$

What is GMRES?

Throughout the presentation, we focus on the Generalized Minimal RESidual (GMRES) algorithm.

- GMRES = Krylov-based iterative solver for the solution of **general square linear systems $Ax = b$**.
- Computes iteratively an orthonormal **Krylov basis V_k** through an Arnoldi process.
- Chooses the vector x_k in $span\{V_k\}$ that **minimizes $\|Ax_k - b\|$**.
- **Reiterate** until x_k is a satisfying approximant of x.

Algorithm: GMRES(A, b, x_0, τ)

Require: $A \in \mathbb{R}^{n \times n}$, $b, x_0 \in \mathbb{R}^n$, $\tau \in \mathbb{R}$

$r_0 = b - Ax_0$
$\beta = \|r_0\|, \; v_1 = r_0/\beta, \; k = 1$
repeat
  $w_k = Av_k$
  for $i = 1, \ldots, k$ **do**
    $h_{i,k} = v_i^T w_k$
    $w_k = w_k - h_{i,k} v_i$
  end for
  $h_{k+1,k} = \|w_k\|, v_{k+1} = w_k / h_{k+1,k}$
  $V_k = [v_1, \ldots, v_k]$
  $H_k = \{h_{i,j}\}_{1 \leq i \leq j+1; 1 \leq j \leq k}$
  $y_k = \textbf{argmin}_y \|\beta e_1 - H_k y\|$
  $k = k + 1$
until $\|\beta e_1 - H_k y_k\| \leq \tau$
$x_k = x_0 + V_k y_k$

GMRES comes in many flavors

Preconditioning

GMRES might converge too slowly. It is essential to use a preconditioner M that transforms $Ax = b$ into an "easier" linear system to solve.

$$M^{-1}Ax = M^{-1}b \quad \text{(left)}, \qquad Au = b, \quad u = Mx \quad \text{(right)}$$

More possibilities: split preconditioning, non-constant preconditioners (FGMRES).

Example of M: ILU, polynomial, block Jacobi, approximate inverse, an iterative method, ...

Restart

The cost in memory and execution time of an iteration grows with k.

Principle: under a chosen restart criterion, stop the iteration, erase V_k, restart GMRES with the initial guess $x_0 = x_k \Rightarrow$ Cumulate more iterations while bounding the cost.

Orthogonalization

The Arnoldi process can be constructed with any orthogonalization procedures: Householder QR, CGS, MGS, CGS2, ...

Warning: Different tradeoffs between numerical stability and performance!

What is a backward error analysis?

Backward and forward errors

Even for $k = n$, GMRES computed in finite precision won't deliver the exact solution. We quantify the quality of the computed solution $\widehat{x}_k$ by the quantities

$$bwd = \frac{\|A\widehat{x}_k - b\|}{\|A\|\|\widehat{x}_k\| + \|b\|}, \qquad fwd = \frac{\|x - \widehat{x}_k\|}{\|x\|}.$$

> "*The process of bounding the backward error of a computed solution is called backward error analysis*" N. J. Higham, Accuracy and Stability of Numerical Algorithms.

Why we care?

- Formal proof that the computed solution will always be correct.
- Reveals how each operation contributes to the final accuracy of the computed solution.
- Is needed to derive a backward error analysis of an algorithm using GMRES.

Existing backward error analysis of GMRES

Bounding the backward and forward error of GMRES is **NOT EASY**:

- GMRES is a complex algorithm made of different sub-algorithms $\rightarrow$ we need a **backward error analysis on every sub-algorithm**.
- GMRES is an iterative process, **bounds** on the errors are only **valid from a certain k** $\rightarrow$ we need to prove the existence of k where the errors are satisfying.

Existing backward error analysis of GMRES

1995 — Householder GMRES

"*Numerical stability of GMRES*" by **J. Drkošová, A. Greenbaum, M. Rozložník and Z. Strakoš**, BIT Numerical Mathematics.

Existing backward error analysis of GMRES

1995 Householder GMRES

"Numerical stability of GMRES" by **J. Drkošová, A. Greenbaum, M. Rozložník and Z. Strakoš**, BIT Numerical Mathematics.

2006 MGS GMRES

"Modified Gram-Schmidt (mgs), least squares, and backward stability of MGS-GMRES" by **C. C. Paige, M. Rozložník, and Z. Strakoš**, 2006, SIAM SIMAX.

Existing backward error analysis of GMRES

1995 — **Householder GMRES**

“*Numerical stability of GMRES*” by **J. Drkošová, A. Greenbaum, M. Rozložník and Z. Strakoš**, BIT Numerical Mathematics.

2006 — **MGS GMRES**

“*Modified Gram-Schmidt (mgs), least squares, and backward stability of MGS-GMRES*” by **C. C. Paige, M. Rozložník, and Z. Strakoš**, 2006, SIAM SIMAX.

2007-2008 — **Flexible MGS GMRES**

“*A Note on GMRES Preconditioned by a Perturbed LDL^T Decomposition with Static Pivoting*” by **M. Arioli, I. S. Duff, S. Gratton, and S. Pralet**, SIAM SISC.

“*Using FGMRES to obtain backward stability in mixed precision*” by **M. Arioli and I. S. Duff**, ETNA.

Our experience of using these analyses

In a previous work of mine:

📕 *"Five-Precision GMRES-based iterative refinement"* by **P. Amestoy, A. Buttari, N. J. Higham, J-Y L'Excellent, T. Mary, B. Vieublé**, Preprint.

We needed a result on the backward stability of **MGS GMRES left-preconditioned by LU** factors computed in low precision.

PROBLEM: The previous backward error analysis of MGS-GMRES does not hold with left-preconditioner and it **CANNOT be straightforwardly adapted**.

Our experience of using these analyses

In a previous work of mine:

"*Five-Precision GMRES-based iterative refinement*" by **P. Amestoy, A. Buttari, N. J. Higham, J-Y L'Excellent, T. Mary, B. Vieublé**, Preprint.

We needed a result on the backward stability of **MGS GMRES left-preconditioned by LU** factors computed in low precision.

PROBLEM: The previous backward error analysis of MGS-GMRES does not hold with left-preconditioner and it **CANNOT be straightforwardly adapted**.

$\Rightarrow$ Because of this tiny change, we had to **REDO the analysis** for this specific variant of GMRES!

Why do we need a new backward error analysis?

The range of possible variants of GMRES is astonishing!

Number of variants =

Why do we need a new backward error analysis?

The range of possible variants of GMRES is astonishing!

Number of variants =

A plethora of preconditioners...

Why do we need a new backward error analysis?

The range of possible variants of GMRES is astonishing!

Number of variants =

A plethora of preconditioners...

× Four ways to apply them: left, right, split, flexible.

Why do we need a new backward error analysis?

The range of possible variants of GMRES is astonishing!

Number of variants =

A plethora of preconditioners...

× Four ways to apply them: left, right, split, flexible.

× Restart or not.

Why do we need a new backward error analysis?

The range of possible variants of GMRES is astonishing!

Number of variants =

A plethora of preconditioners...

× Four ways to apply them: left, right, split, flexible.

× Restart or not.

× Possible orthogonalization methods: CGS, MGS, CGS2, Householder, ...

Why do we need a new backward error analysis?

The range of possible variants of GMRES is astonishing!

Number of variants =

A plethora of preconditioners...

× Four ways to apply them: left, right, split, flexible.

× Restart or not.

× Possible orthogonalization methods: CGS, MGS, CGS2, Householder, ...

× All the "more exotic" techniques: recycling, randomization, mixed precision, compression of the basis, ...

Why do we need a new backward error analysis?

The range of possible variants of GMRES is astonishing!

Number of variants =

A plethora of preconditioners...

× Four ways to apply them: left, right, split, flexible.

× Restart or not.

× Possible orthogonalization methods: CGS, MGS, CGS2, Householder, ...

× All the "more exotic" techniques: recycling, randomization, mixed precision, compression of the basis, ...

$\Rightarrow$ An almost infinite number of variants...

Why do we need a new backward error analysis?

... BUT only a tiny subset of them are covered by the previous analyses.

Why do we need a new backward error analysis?

… BUT only a tiny subset of them are covered by the previous analyses.

In addition:

- These analyses were **not** made to be **modular** ⇒ Changing one element requires redoing a big part of the analysis.
- They are very smart, long, and hard ⇒ Understanding and **adapting them is a challenge**.

Why do we need a new backward error analysis?

... BUT only a tiny subset of them are covered by the previous analyses.

In addition:

- These analyses were **not** made to be **modular** $\Rightarrow$ Changing one element requires redoing a big part of the analysis.
- They are very smart, long, and hard $\Rightarrow$ Understanding and **adapting them is a challenge**.

Consequences:

- A few GMRES variants have error bounds on their computed solution
- Bounding errors of a new variant is inconvenient and tedious.

Toward a generic and modular tool

Can we provide an analysis...

Toward a generic and modular tool

Can we provide an analysis…

- … that gives the sharpest error bounds?

Toward a generic and modular tool

Can we provide an analysis...

- ... that gives the sharpest error bounds?
- ... that is generic enough to cover “a lot” of possible GMRES variants (i.e., different preconditioners, orthogonalization, restart, mixed precision, ...)?

Toward a generic and modular tool

Can we provide an analysis...

- ... that gives the sharpest error bounds?
- ... that is generic enough to cover “a lot” of possible GMRES variants (i.e., different preconditioners, orthogonalization, restart, mixed precision, ...)?
- ... that is modular (if you change the preconditioner, you do not need to redo all the analysis)?

Toward a generic and modular tool

Can we provide an analysis...

- ... that gives the sharpest error bounds?
- ... that is generic enough to cover "a lot" of possible GMRES variants (i.e., different preconditioners, orthogonalization, restart, mixed precision, ...)?
- ... that is modular (if you change the preconditioner, you do not need to redo all the analysis)?
- ... that is easy to use to some extent?

Toward a generic and modular tool

Can we provide an analysis...

- ... that gives the sharpest error bounds?
- ... that is generic enough to cover "a lot" of possible GMRES variants (i.e., different preconditioners, orthogonalization, restart, mixed precision, ...)?
- ... that is modular (if you change the preconditioner you do not need to redo all the analysis)?
- ... that is easy to use to some extent?

$\Rightarrow$ We aim to propose a modular and generic backward error analysis tool for GMRES.

Generic GMRES: an abstract algorithm

Algorithm: GEN-GMRES(A, b, M_l, k)

1: Initialize $Z_k = [z_1, \ldots, z_k]$.
2: Compute $C_k = \widetilde{A} Z_k$ where $\widetilde{A} = M_l^{-1} A$.
3: Compute $\widetilde{b} = M_l^{-1} b$.
4: Solve $y_k = \mathbf{argmin}_y \, \|\widetilde{b} - C_k y\|$.
5: Compute the approximant $x_k = Z_k y_k$.

Generic GMRES: an abstract algorithm

Algorithm: GEN-GMRES(A, b, M_l, k)

1: Initialize $Z_k = [z_1, \ldots, z_k]$.
2: Compute $C_k = \widetilde{A} Z_k$ where $\widetilde{A} = M_l^{-1} A$.
3: Compute $\widetilde{b} = M_l^{-1} b$.
4: Solve $y_k = \textbf{argmin}_y \|\widetilde{b} - C_k y\|$.
5: Compute the approximant $x_k = Z_k y_k$.

Principle: Finding $x_k \in span\{Z_k\}$ minimizing the left-preconditioned residual $\|\widetilde{b} - \widetilde{A}x\|$.

- Do not assume Arnoldi process.
- Not presented as an iterative process.
- Z_k can be any basis of rank k.
- Little assumptions on the operations.

➤ Can be seen as a left-preconditioned Flexible GMRES where the left-preconditioner M_l, the preconditioned basis Z_k, and the least squares solver are not specified.

Generic GMRES: an abstract algorithm

Algorithm: GEN-GMRES(A, b, M_l, k)

1: Initialize $Z_k = [z_1, \ldots, z_k]$.
2: Compute $C_k = \widetilde{A} Z_k$ where $\widetilde{A} = M_l^{-1} A$.
3: Compute $\widetilde{b} = M_l^{-1} b$.
4: Solve $y_k = \mathbf{argmin}_y \|\widetilde{b} - C_k y\|$.
5: Compute the approximant $x_k = Z_k y_k$.

Specialization to:

Algorithm: MGS GMRES

1: Consider the computed Arnoldi basis $\widehat{V}_k = [\hat{v}_1, \ldots, \hat{v}_k]$.
2: Compute $C_k = A\widehat{V}_k$, where $M_l = I$.
3:
4: Solve $y_k = \mathbf{argmin}_y \|b - A\widehat{V}_k y\|$ by MGS Arnoldi.
5: Compute the approximant $x_k = \widehat{V}_k y_k$.

Generic GMRES: an abstract algorithm

Algorithm: GEN-GMRES(A, b, M_l, k)

1: Initialize $Z_k = [z_1, \dots, z_k]$.
2: Compute $C_k = \widetilde{A} Z_k$ where $\widetilde{A} = M_l^{-1} A$.
3: Compute $\widetilde{b} = M_l^{-1} b$.
4: Solve $y_k = \mathrm{argmin}_y \|\widetilde{b} - C_k y\|$.
5: Compute the approximant $x_k = Z_k y_k$.

Specialization to:

Algorithm: MGS GMRES with left- LU preconditioner

1: Consider the computed Arnoldi basis $\widehat{V}_k = [\hat{v}_1, \dots, \hat{v}_k]$.
2: Compute $C_k = \widetilde{A} \widehat{V}_k$ where $\widetilde{A} = U \backslash L \backslash A$.
3: Compute $\widetilde{b} = U \backslash L \backslash b$.
4: Solve $y_k = \mathrm{argmin}_y \|\widetilde{b} - \widetilde{A} \widehat{V}_k y\|$ by MGS Arnoldi.
5: Compute the approximant $x_k = \widehat{V}_k y_k$.

Generic GMRES: an abstract algorithm

Algorithm: GEN-GMRES(A, b, M_l, k)

1: Initialize $Z_k = [z_1, \dots, z_k]$.
2: Compute $C_k = \widetilde{A} Z_k$ where $\widetilde{A} = M_l^{-1} A$.
3: Compute $\widetilde{b} = M_l^{-1} b$.
4: Solve $y_k = \mathrm{argmin}_y \|\widetilde{b} - C_k y\|$.
5: Compute the approximant $x_k = Z_k y_k$.

Specialization to:

Algorithm: MGS GMRES with flexible LU preconditioner

1: Consider the preconditioned Arnoldi basis $Z_k = U \backslash L \backslash \widehat{V}_k$.
2: Compute $C_k = A Z_k$.
3:
4: Solve $y_k = \mathrm{argmin}_y \|b - A Z_k y\|$ by MGS Arnoldi.
5: Compute the approximant $x_k = Z_k y_k$.

Generic GMRES: an abstract algorithm

Algorithm: GEN-GMRES(A, b, M_l, k)

1: Initialize $Z_k = [z_1, \ldots, z_k]$.
2: Compute $C_k = \widetilde{A} Z_k$ where $\widetilde{A} = M_l^{-1} A$.
3: Compute $\widetilde{b} = M_l^{-1} b$.
4: Solve $y_k = \mathbf{argmin}_y \|\widetilde{b} - C_k y\|$.
5: Compute the approximant $x_k = Z_k y_k$.

GEN-GMRES is an abstract generic algorithm that can be specialized to many GMRES algorithms $\Rightarrow$ Any result on GEN-GMRES holds for its specializations.

Our goal: Make a backward error analysis of GEN-GMRES.

One analysis to rule them all!

Generic rounding error model

The terms $\epsilon_{\tilde{A}}$, ϵ_{b}, ϵ_{LS}, and ϵ_{Z} quantify the accuracies of every operation and are unspecified. They are only specified for a given specialization of GEN-GMRES.

Matrix–matrix product with the basis (step 2)

$$fl(\widetilde{A}Z_k) = \widetilde{A}Z_k + \Delta_{\widetilde{A}Z_k}, \qquad \|\Delta_{\widetilde{A}Z_k}\| \le \epsilon_{\tilde{A}}\|\widetilde{A}Z_k\|.$$

Preconditioned RHS (step 3)

$$fl(M_l^{-1}b) = \widetilde{b} + \Delta\widetilde{b}, \qquad \|\Delta\widetilde{b}\| \le \epsilon_{b}\|\widetilde{b}\|.$$

Least squares solution (step 4)

$$\widehat{y}_k = \text{argmin}_y \|\widetilde{b} + \Delta b' - (fl(AZ_k) + \Delta'_{\widetilde{A}Z_k})\|$$

$$\|[\Delta\widetilde{b}', \Delta'_{\widetilde{A}Z_k}]e_j\| \le \epsilon_{LS}\|[\widetilde{b}, fl(AZ_k)]e_j\|$$

Compute the kth approximant (step 5)

$$\widehat{x}_k = fl(Z_k\widehat{y}_k) = (Z_k + \Delta Z_k)\widehat{y}_k, \qquad \|\Delta Z_k\| \le \epsilon_{Z}\|Z_k\|$$

A key dimension(/iteration)

We need to define the special dimension(/iteration) k at which we can demonstrate that the computed solution has attained a satisfying error.

Key dimension

We define the key dimension k as the first $k \leq n$ such that, for all $\phi > 0$, we have

$$\sigma_{\min}([\widetilde{b}\phi, \widetilde{A}Z_k]) \leq \epsilon_{\text{LS}} \|[\widetilde{b}\phi, \widetilde{A}Z_k]\|_F$$

and

$$\sigma_{\min}(\widetilde{A}Z_k) \gg (\epsilon_{\tilde{A}} + \epsilon_{\text{b}} + \epsilon_{\text{LS}}) \|\widetilde{A}Z_k\|_F.$$

The philosophy of these conditions is to capture the exact moment where $\widetilde{b}$ lies in the range of $\widetilde{A}Z_k$, which is the moment where the basis Z_k contains the solution.

"Modified Gram-Schmidt (mgs), least squares, and backward stability of MGS-GMRES" by **C. C. Paige, M. Rozložník, and Z. Strakoš**, 2006, SIAM SIMAX.

Error bounds of GEN-GMRES

Theorem

Consider the solution of a nonsingular linear system

$$Ax = b, \quad A \in \mathbb{R}^{n\times n}, \quad 0 \neq b \in \mathbb{R}^n,$$

with GEN-GMRES under the previous **error model**. **If there exists a key dimension** k as defined previously, then, GEN-GMRES produces a computed solution $\widehat{x}_k$ whose **backward** and **forward** error satisfies respectively

$$\frac{\|b - A\widehat{x}_k\|}{\|b\| + \|A\|\|\widehat{x}_k\|} \lesssim \Phi\kappa(M_l), \qquad \frac{\|\widehat{x}_k - x\|}{\|x\|} \lesssim \Phi\kappa(\widetilde{A}),$$

where

$$\Phi \equiv \alpha\epsilon_{\tilde{A}} + \beta\epsilon_{b} + \beta\epsilon_{LS} + \lambda\epsilon_{Z}$$

with

$$\alpha \equiv \sigma_{\min}^{-1}(Z_k)\frac{\|\widetilde{A}Z_k\|}{\|\widetilde{A}\|}, \quad \beta \equiv \max(1, \sigma_{\min}^{-1}(Z_k)\frac{\|\widetilde{A}Z_k\|}{\|\widetilde{A}\|}), \quad \lambda \equiv \sigma_{\min}^{-1}(Z_k)\|Z_k\|.$$

How to use?

How to use the previous result to **derive** forward and backward error bounds **for real GMRES algorithms**?

How to use?

How to use the previous result to **derive** forward and backward error bounds **for real GMRES algorithms**?

Using the previous theorem requires some work:

- Show that your algorithm is a **specialization of GEN-GMRES**.
- **Determine** $\epsilon_{\tilde{A}}$, ϵ_{b}, ϵ_{LS}, and ϵ_{Z}. The difficulty of this step varies according to the existing literature of the methods used.
- Show the existence of **the key dimension**. The difficulty also varies according to the existing literature.

How to use?

How to use the previous result to derive forward and backward error bounds for real GMRES algorithms?

Using the previous theorem requires some work:

- Show that your algorithm is a specialization of GEN-GMRES.
- Determine $\epsilon_{\tilde{A}}$, ϵ_{b}, ϵ_{LS}, and ϵ_{Z}. The difficulty of this step varies according to the existing literature of the methods used.
- Show the existence of the key dimension. The difficulty also varies according to the existing literature.

This Theorem is backward compatible with the previous analyses: Applying it on Householder GMRES, MGS GMRES, and Flexible MGS GMRES gives the same results as the existing analyses.

Error model for restarted GEN-GMRES

Algorithm: Restarted GEN-GMRES(A, b, M_l)

```
Initialize x_0
repeat
    Compute r_i = Ax_i − b.
    Solve Ad_i = r_i with GEN-GMRES.
    Compute the approximant x_{i+1} = x_i + d_i.
until convergence
```

Error model for restarted GEN-GMRES

Algorithm: Restarted GEN-GMRES(A, b, M_l)

1: Initialize x_0
2: **repeat**
3: Compute $r_i = Ax_i - b$.
4: Solve $Ad_i = r_i$ with GEN-GMRES.
5: Compute the approximant $x_{i+1} = x_i + d_i$.
6: **until** convergence

Residual computation (step 3)

$$\widehat{r}_i = b - A\widehat{x}_i + \Delta r_i, \qquad |\Delta r_i| \le \epsilon_R(|b| + |A||\widehat{x}_i|).$$

Restart update (step 5)

$$\widehat{x}_{i+1} = \widehat{x}_i + \widehat{d}_i + \Delta x_i, \qquad |\Delta x_i| \le \epsilon_U|\widehat{x}_{i+1}|.$$

Mixed precision introduction

Commonly available arithmetics

	ID	Signif. bits	Exp. bits	Range	Unit roundoff u
fp128	Q	113	15	$10^{\pm 4932}$	1×10^{-34}
double-fp64	DD	107	11	$10^{\pm 308}$	6×10^{-33}
fp64	D	53	11	$10^{\pm 308}$	1×10^{-16}
fp32	S	24	8	$10^{\pm 38}$	6×10^{-8}
tfloat32	T	11	8	$10^{\pm 38}$	5×10^{-4}
fp16	H	11	5	$10^{\pm 5}$	5×10^{-4}
bfloat16	B	8	8	$10^{\pm 38}$	4×10^{-3}
fp8 (E4M3)	R	4	4	$10^{\pm 2}$	6.3×10^{-2}
fp8 (E5M2)	R*	3	5	$10^{\pm 5}$	1.3×10^{-1}

Mixed precision introduction

Commonly available arithmetics

	ID	Signif. bits	Exp. bits	Range	Unit roundoff u
fp128	Q	113	15	$10^{\pm 4932}$	1×10^{-34}
double-fp64	DD	107	11	$10^{\pm 308}$	6×10^{-33}
fp64	D	53	11	$10^{\pm 308}$	1×10^{-16}
fp32	S	24	8	$10^{\pm 38}$	6×10^{-8}
tfloat32	T	11	8	$10^{\pm 38}$	5×10^{-4}
fp16	H	11	5	$10^{\pm 5}$	5×10^{-4}
bfloat16	B	8	8	$10^{\pm 38}$	4×10^{-3}
fp8 (E4M3)	R	4	4	$10^{\pm 2}$	6.3×10^{-2}
fp8 (E5M2)	R*	3	5	$10^{\pm 5}$	1.3×10^{-1}

The low precision arithmetics are less accurate BUT are faster, consumes less memory and energy.

Specialization to mixed precision GMRES

Algorithm: Restart loop

Compute $A \approx \widehat{LU}$ u_f
repeat
  $x_{i+1} = \text{GMRES}(A, \widehat{LU}, b, x_i, \tau)$
until convergence

Algorithm: $\text{GMRES}(A, \widehat{LU}, b, x_0, \tau)$

Require: $A, M^{-1} \in \mathbb{R}^{n \times n}$, $b, x_0 \in \mathbb{R}^n$, $\tau \in \mathbb{R}$

$r_0 = b - Ax$ u_r
$s_0 = \widehat{U} \backslash \widehat{L} \backslash r_0$ u_p
$\beta = \|s_0\|,\ v_1 = s_0/\beta,\ k = 1$ u_g
repeat
  $z_k = Av_k$ u_p
  $w_k = \widehat{U} \backslash \widehat{L} \backslash z_k$ u_p
  for $i = 1, \ldots, k$ **do**
    $h_{i,k} = v_i^T w_k$ u_g
    $w_k = w_k - h_{i,k} v_i$ u_g
  end for
  $h_{k+1,k} = \|w_k\|, v_{k+1} = w_k / h_{k+1,k}$ u_g
  $V_k = [v_1, \ldots, v_k]$
  $H_k = \{h_{i,j}\}_{1 \le i \le j+1; 1 \le j \le k}$
  $y_k = \text{argmin}_y \|\beta e_1 - H_k y\|$ u_g
  $k = k + 1$
until $\|\beta e_1 - H_k y_k\| \le \tau$
$x_k = x_0 + V_k y_k$ u

Specialization to mixed precision GMRES

Algorithm: Restart loop

Compute $A \approx \widehat{LU}$ $\quad u_f$
repeat
  $x_{i+1} = \text{GMRES}(A, \widehat{LU}, b, x_i, \tau)$
until convergence

- Restarted LU-left-preconditioned GMRES with MGS Arnoldi.

Algorithm: $\text{GMRES}(A, \widehat{LU}, b, x_0, \tau)$

Require: $A, M^{-1} \in \mathbb{R}^{n \times n}$, $b, x_0 \in \mathbb{R}^n$, $\tau \in \mathbb{R}$

$r_0 = b - Ax$ $\quad u_r$
$s_0 = \widehat{U} \backslash \widehat{L} \backslash r_0$ $\quad u_p$
$\beta = \|s_0\|, \; v_1 = s_0/\beta, \; k = 1$ $\quad u_g$
repeat
  $z_k = Av_k$ $\quad u_p$
  $w_k = \widehat{U} \backslash \widehat{L} \backslash z_k$ $\quad u_p$
  for $i = 1, \ldots, k$ **do**
    $h_{i,k} = v_i^T w_k$ $\quad u_g$
    $w_k = w_k - h_{i,k} v_i$ $\quad u_g$
  end for
  $h_{k+1,k} = \|w_k\|, v_{k+1} = w_k / h_{k+1,k}$ $\quad u_g$
  $V_k = [v_1, \ldots, v_k]$
  $H_k = \{h_{i,j}\}_{1 \le i \le j+1; 1 \le j \le k}$
  $y_k = \text{argmin}_y \|\beta e_1 - H_k y\|$ $\quad u_g$
  $k = k + 1$
until $\|\beta e_1 - H_k y_k\| \le \tau$
$x_k = x_0 + V_k y_k$ $\quad u$

Specialization to mixed precision GMRES

Algorithm: Restart loop

```
Compute A ≈ \widehat{LU}                          u_f
repeat
    x_{i+1} = GMRES(A, \widehat{LU}, b, x_i, τ)
until convergence
```

- Restarted LU-left-preconditioned GMRES with MGS Arnoldi.
- 5 precisions: $u_f \geq u_g \geq u_p \geq u \geq u_r$.

Algorithm: GMRES($A, \widehat{LU}, b, x_0, \tau$)

Require: $A, M^{-1} \in \mathbb{R}^{n \times n}$, $b, x_0 \in \mathbb{R}^n$, $\tau \in \mathbb{R}$

$r_0 = b - Ax$ — u_r

$s_0 = \widehat{U} \backslash \widehat{L} \backslash r_0$ — u_p

$\beta = \|s_0\|,\ v_1 = s_0/\beta,\ k = 1$ — u_g

repeat

  $z_k = A v_k$ — u_p

  $w_k = \widehat{U} \backslash \widehat{L} \backslash z_k$ — u_p

  for $i = 1, \ldots, k$ **do**

    $h_{i,k} = v_i^T w_k$ — u_g

    $w_k = w_k - h_{i,k} v_i$ — u_g

  end for

  $h_{k+1,k} = \|w_k\|, v_{k+1} = w_k / h_{k+1,k}$ — u_g

  $V_k = [v_1, \ldots, v_k]$

  $H_k = \{h_{i,j}\}_{1 \leq i \leq j+1; 1 \leq j \leq k}$

  $y_k = \textbf{argmin}_y \|\beta e_1 - H_k y\|$ — u_g

  $k = k + 1$

until $\|\beta e_1 - H_k y_k\| \leq \tau$

$x_k = x_0 + V_k y_k$ — u

Specialization to mixed precision GMRES

Algorithm: Restart loop

Compute $A \approx \widehat{L}\widehat{U}$ — u_f
repeat
  $x_{i+1} = \text{GMRES}(A, \widehat{L}\widehat{U}, b, x_i, \tau)$
until convergence

- Restarted LU-left-preconditioned GMRES with MGS Arnoldi.
- 5 precisions: $u_f \geq u_g \geq u_p \geq u \geq u_r$.
- Aims to compute a solution to accuracy u.

Algorithm: $\text{GMRES}(A, \widehat{L}\widehat{U}, b, x_0, \tau)$

Require: $A, M^{-1} \in \mathbb{R}^{n \times n}$, $b, x_0 \in \mathbb{R}^n$, $\tau \in \mathbb{R}$

$r_0 = b - Ax$ — u_r
$s_0 = \widehat{U} \backslash \widehat{L} \backslash r_0$ — u_p
$\beta = \|s_0\|,\ v_1 = s_0/\beta,\ k = 1$ — u_g
repeat
  $z_k = Av_k$ — u_p
  $w_k = \widehat{U} \backslash \widehat{L} \backslash z_k$ — u_p
  for $i = 1, \ldots, k$ **do**
    $h_{i,k} = v_i^T w_k$ — u_g
    $w_k = w_k - h_{i,k} v_i$ — u_g
  end for
  $h_{k+1,k} = \|w_k\|, v_{k+1} = w_k / h_{k+1,k}$ — u_g
  $V_k = [v_1, \ldots, v_k]$
  $H_k = \{h_{i,j}\}_{1 \leq i \leq j+1; 1 \leq j \leq k}$
  $y_k = \text{argmin}_y \|\beta e_1 - H_k y\|$ — u_g
  $k = k + 1$
until $\|\beta e_1 - H_k y_k\| \leq \tau$
$x_k = x_0 + V_k y_k$ — u

Specialization to mixed precision GMRES

Algorithm: Restart loop

```
Compute A ≈ \widehat{LU}                                  u_f
repeat
    x_{i+1} = GMRES(A, \widehat{LU}, b, x_i, τ)
until convergence
```

- Restarted LU-left-preconditioned GMRES with MGS Arnoldi.
- 5 precisions: $u_f \geq u_g \geq u_p \geq u \geq u_r$.
- Aims to compute a solution to accuracy u.
- GMRES iterations and costly preconditioner computed in low precisions (u_g, u_f, and u_p).

Algorithm: $\mathrm{GMRES}(A, \widehat{LU}, b, x_0, \tau)$

Require: $A, M^{-1} \in \mathbb{R}^{n\times n}$, $b, x_0 \in \mathbb{R}^n$, $\tau \in \mathbb{R}$

```
r_0 = b − Ax                                              u_r
s_0 = \hat{U}\\hat{L}\r_0                                 u_p
β = ‖s_0‖, v_1 = s_0/β, k = 1                             u_g
repeat
    z_k = A v_k                                           u_p
    w_k = \hat{U}\\hat{L}\z_k                             u_p
    for i = 1, ..., k do
        h_{i,k} = v_i^T w_k                               u_g
        w_k = w_k − h_{i,k} v_i                           u_g
    end for
    h_{k+1,k} = ‖w_k‖, v_{k+1} = w_k/h_{k+1,k}            u_g
    V_k = [v_1, ..., v_k]
    H_k = {h_{i,j}}_{1≤i≤j+1; 1≤j≤k}
    y_k = argmin_y ‖βe_1 − H_k y‖                         u_g
    k = k + 1
until ‖βe_1 − H_k y_k‖ ≤ τ
x_k = x_0 + V_k y_k                                       u
```

Specialization to mixed precision GMRES

Algorithm: Restart loop

```
Compute A ≈ \widehat{L}\widehat{U}      [u_f]
repeat
    x_{i+1} = GMRES(A, \widehat{L}\widehat{U}, b, x_i, τ)
until convergence
```

- Restarted LU-left-preconditioned GMRES with MGS Arnoldi.
- 5 precisions: $u_f \geq u_g \geq u_p \geq u \geq u_r$.
- Aims to compute a solution to accuracy u.
- GMRES iterations and costly preconditioner computed in low precisions (u_g, u_f, and u_p).
- Restart computed in high precisions to recover accuracy (u and u_r).

Algorithm: GMRES($A, \widehat{L}\widehat{U}, b, x_0, \tau$)

Require: $A, M^{-1} \in \mathbb{R}^{n\times n}$, $b, x_0 \in \mathbb{R}^n$, $\tau \in \mathbb{R}$

```
r_0 = b − Ax                                   [u_r]
s_0 = \widehat{U}\\widehat{L}\r_0              [u_p]
β = ||s_0||, v_1 = s_0/β, k = 1                [u_g]
repeat
    z_k = Av_k                                 [u_p]
    w_k = \widehat{U}\\widehat{L}\z_k          [u_p]
    for i = 1, ..., k do
        h_{i,k} = v_i^T w_k                    [u_g]
        w_k = w_k − h_{i,k} v_i                [u_g]
    end for
    h_{k+1,k} = ||w_k||, v_{k+1} = w_k/h_{k+1,k}   [u_g]
    V_k = [v_1, ..., v_k]
    H_k = {h_{i,j}}_{1≤i≤j+1; 1≤j≤k}
    y_k = argmin_y ||βe_1 − H_k y||            [u_g]
    k = k + 1
until ||βe_1 − H_k y_k|| ≤ τ
x_k = x_0 + V_k y_k                            [u]
```

Stability of restarted left-preconditioned GMRES

Using the theorem on restarted GEN-GMRES on the previous algorithm delivers the following stability result.

Theorem

Let $Ax = b$ be solved by the previous mixed precision restarted LU-left-preconditioned GMRES. Provided that

$$\kappa(A)u_p < 1 \quad \text{and} \quad \sigma_{\min}(\widetilde{A}) \gg (u_p\kappa(A) + u_g)\|\widetilde{A}\|,$$

the ***forward error***

$$\frac{\|\hat{x} - x\|}{\|x\|} \leq n u_r \operatorname{cond}(A, x) + u \quad \text{if} \quad (u_g + u_p\kappa(A))(1 + \kappa(A)^2 u_f^2) \ll 1,$$

and the ***backward error***

$$\frac{\|A\hat{x} - b\|}{\|A\|\|x\| + \|b\|} \leq n u_r + u, \quad \text{if} \quad (u_g + u_p\kappa(A))(1 + \kappa(A)u_f)\kappa(A) \ll 1.$$

"Five-Precision GMRES-based Iterative Refinement" by **P. R. Amestoy, A. Buttari, N. J. Higham, J-Y. L'Excellent, T. Mary, B. Vieublé**, Preprint.

Foretaste of performance study on real-life applications

Name	N	NNZ	Arith.	Sym.	$\kappa(A)$	Fact. (flops)	Slv. (flops)
ElectroPhys10M	1.02E+07	1.41E+08	R	1	1.10E+01	4E+14	9E+10
DrivAer6M	6.11E+06	4.97E+07	R	1	9.40E+05	6E+13	3E+10
Queen_4147	4.14E+06	3.28E+08	R	1	4.30E+06	3E+14	6E+10
tminlet3M	2.84E+06	1.62E+08	C	0	2.70E+07	1E+14	2E+10
perf009ar	5.41E+06	2.08E+08	R	1	3.70E+08	2E+13	2E+10
elasticity-3d	5.18E+06	1.16E+08	R	1	3.60E+09	2E+14	5E+10
lfm_aug5M	5.52E+06	3.71E+07	C	1	5.80E+11	2E+14	5E+10
CarBody25M	2.44E+07	7.06E+08	R	1	8.60E+12	1E+13	3E+10
thmgas	5.53E+06	3.71E+07	R	0	8.28E+13	1E+14	4E+10

Set of **industrial** and SuiteSparse matrices.

➤ The matrices are **ordered in increasing $\kappa(A)$**, the higher $\kappa(A)$ is, the slower the convergence (if reached at all).

Foretaste of performance study on real-life applications

Name	N	NNZ	Arith.	Sym.	$\kappa(A)$	Fact. (flops)	Slv. (flops)
ElectroPhys10M	1.02E+07	1.41E+08	R	1	1.10E+01	4E+14	9E+10
DrivAer6M	6.11E+06	4.97E+07	R	1	9.40E+05	6E+13	3E+10
Queen_4147	4.14E+06	3.28E+08	R	1	4.30E+06	3E+14	6E+10
tminlet3M	2.84E+06	1.62E+08	C	0	2.70E+07	1E+14	2E+10
perf009ar	5.41E+06	2.08E+08	R	1	3.70E+08	2E+13	2E+10
elasticity-3d	5.18E+06	1.16E+08	R	1	3.60E+09	2E+14	5E+10
lfm_aug5M	5.52E+06	3.71E+07	C	1	5.80E+11	2E+14	5E+10
CarBody25M	2.44E+07	7.06E+08	R	1	8.60E+12	1E+13	3E+10
thmgas	5.53E+06	3.71E+07	R	0	8.28E+13	1E+14	4E+10

Set of **industrial** and SuiteSparse matrices.

➤ We run on OLYMPE supercomputer nodes (two Intel 18-cores Skylake/node), 1 node **(2MPI×18threads)** or 2 nodes **(4MPI×18threads)** depending on the matrix size.

Foretaste of performance study on real-life applications

Name	N	NNZ	Arith.	Sym.	$\kappa(A)$	Fact. (flops)	Slv. (flops)
ElectroPhys10M	1.02E+07	1.41E+08	R	1	1.10E+01	4E+14	9E+10
DrivAer6M	6.11E+06	4.97E+07	R	1	9.40E+05	6E+13	3E+10
Queen_4147	4.14E+06	3.28E+08	R	1	4.30E+06	3E+14	6E+10
tminlet3M	2.84E+06	1.62E+08	C	0	2.70E+07	1E+14	2E+10
perf009ar	5.41E+06	2.08E+08	R	1	3.70E+08	2E+13	2E+10
elasticity-3d	5.18E+06	1.16E+08	R	1	3.60E+09	2E+14	5E+10
lfm_aug5M	5.52E+06	3.71E+07	C	1	5.80E+11	2E+14	5E+10
CarBody25M	2.44E+07	7.06E+08	R	1	8.60E+12	1E+13	3E+10
thmgas	5.53E+06	3.71E+07	R	0	8.28E+13	1E+14	4E+10

Set of **industrial** and SuiteSparse matrices.

- $u_p = u_g = u =$ D and $u_r =$ Q.
- LU factors are computed in single precision ($u_f =$ S), with low-rank approximation and static pivoting.

Implementation details and design choices

- We use the MUMPS multifrontal sparse solvers for factorization and solve. MUMPS supports BLR, static pivoting, and threshold partial pivoting.

Implementation details and design choices

- ➤ We use the MUMPS multifrontal sparse solvers for factorization and solve. MUMPS supports BLR, static pivoting, and threshold partial pivoting.
- ➤ We cast in-place the factors fully from fp32 to fp64.

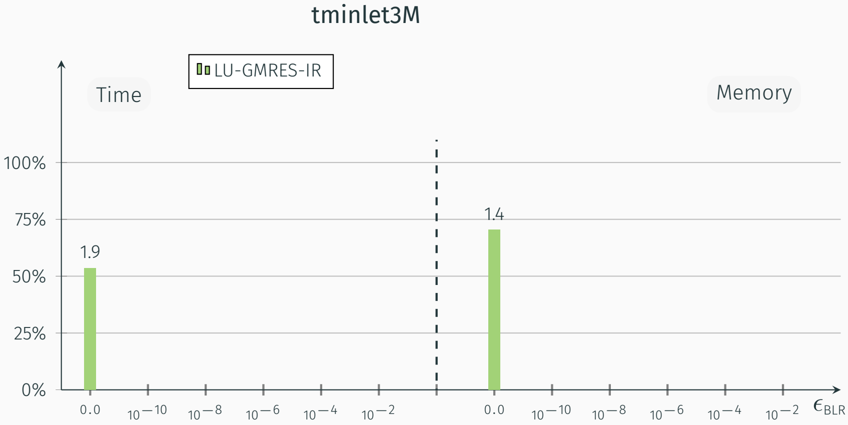
Implementation details and design choices

- ➤ We use the MUMPS multifrontal sparse solvers for factorization and solve. MUMPS supports BLR, static pivoting, and threshold partial pivoting.
- ➤ We cast in-place the factors fully from fp32 to fp64.
- ➤ In-house GMRES implementation and SpMV kernel running in parallel on the master MPI process.

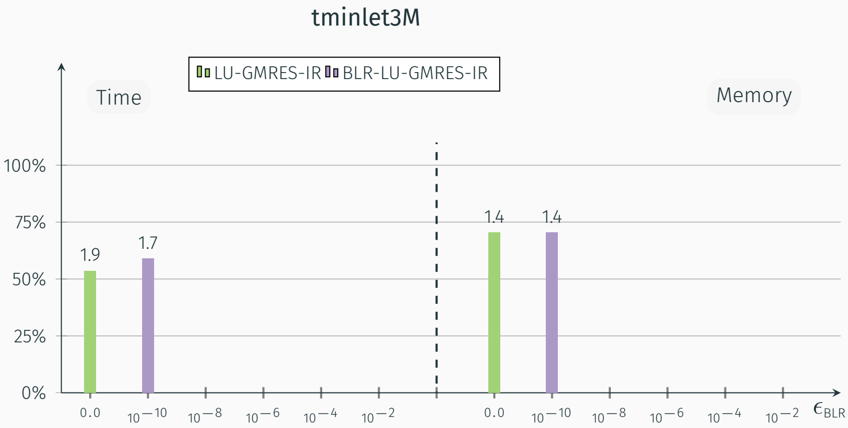
Implementation details and design choices

- We use the MUMPS multifrontal sparse solvers for factorization and solve. MUMPS supports BLR, static pivoting, and threshold partial pivoting.
- We cast in-place the factors fully from fp32 to fp64.
- In-house GMRES implementation and SpMV kernel running in parallel on the master MPI process.
- The MUMPS factorization and solve are distributed over the MPI processes.

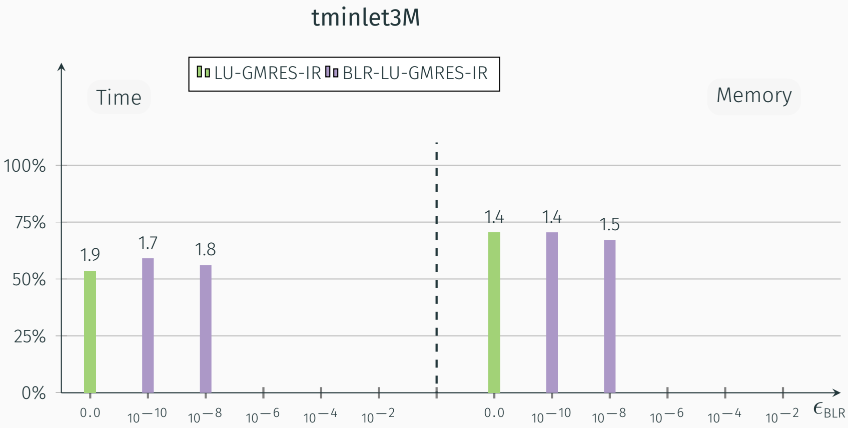
Time and memory performance with BLR w.r.t. DMUMPS



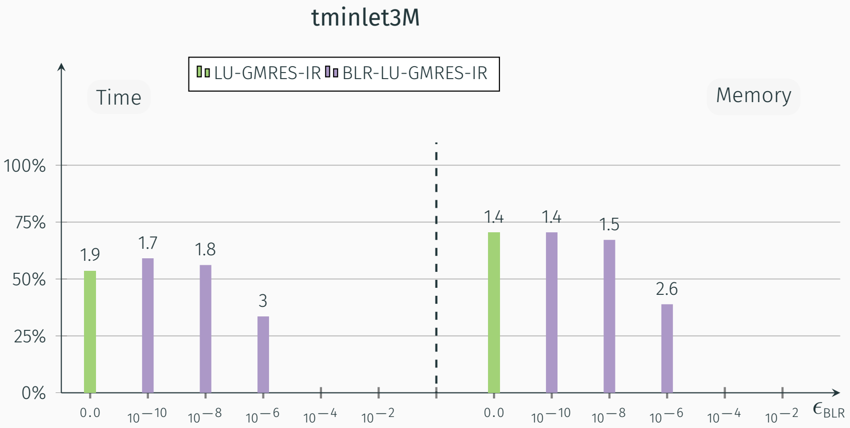
Time and memory performance with BLR w.r.t. DMUMPS



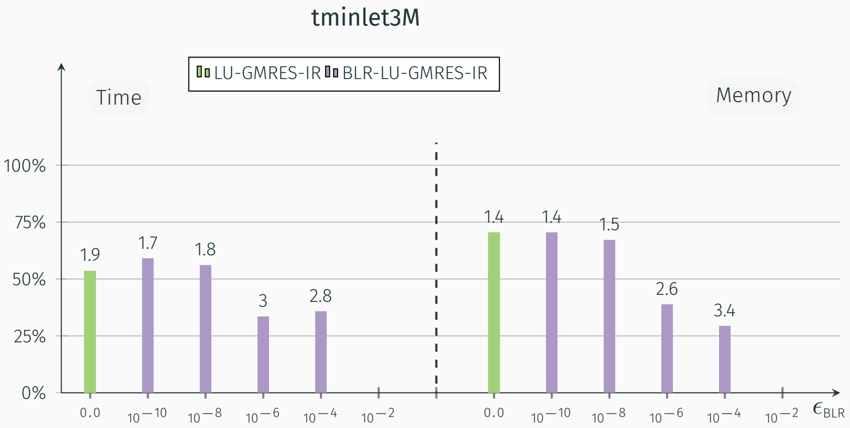
Time and memory performance with BLR w.r.t. DMUMPS



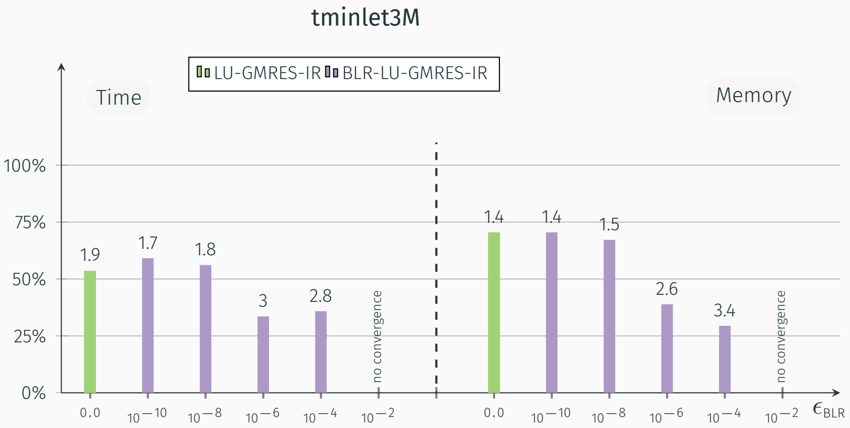
Time and memory performance with BLR w.r.t. DMUMPS



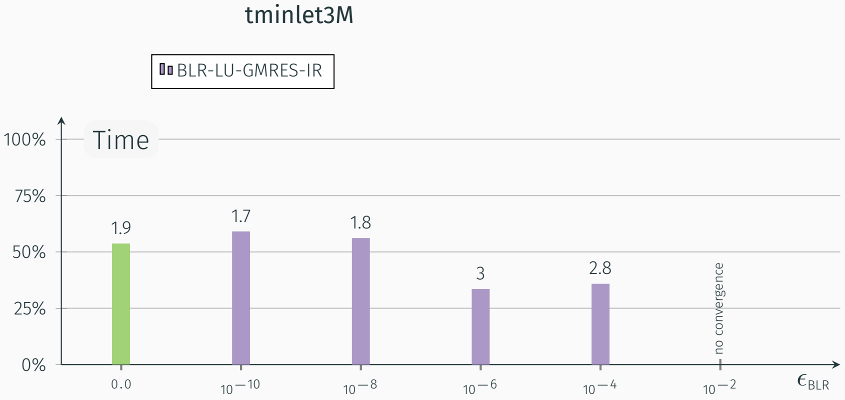
Time and memory performance with BLR w.r.t. DMUMPS



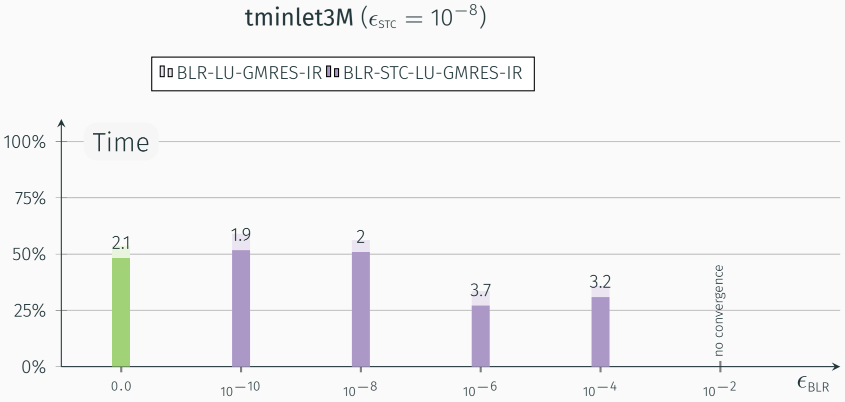
Time and memory performance with BLR w.r.t. DMUMPS



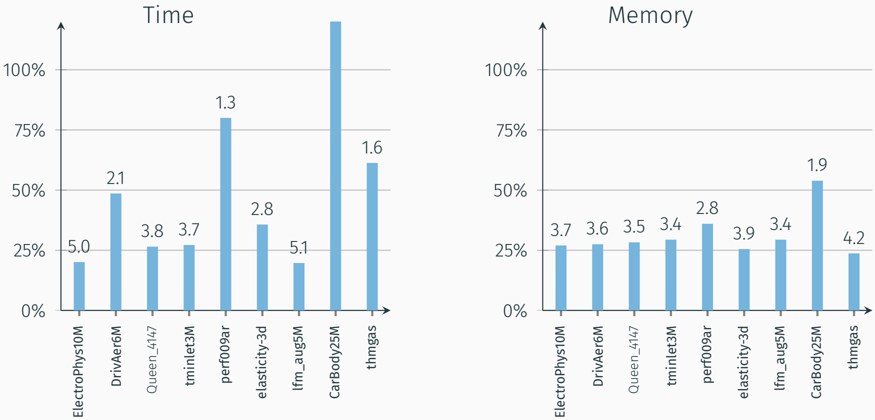
Time performance with BLR + static pivoting w.r.t. DMUMPS



Time performance with BLR + static pivoting w.r.t. DMUMPS



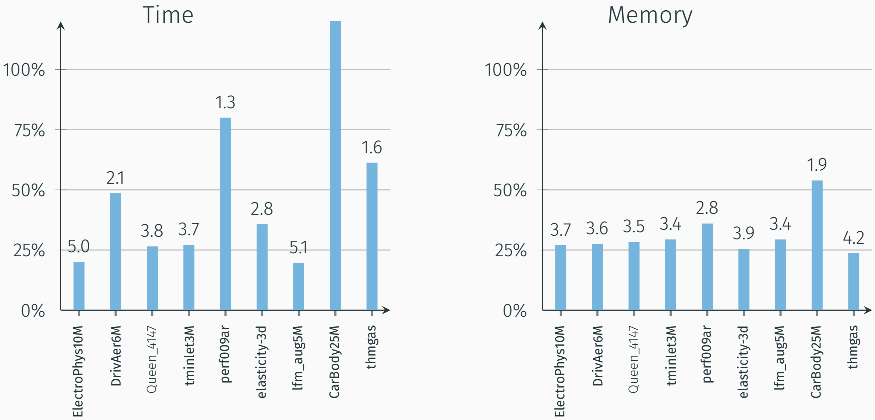
Best time and memory w.r.t. DMUMPS



Compared to a LU direct solver in double precision without approximations and with threshold partial pivoting.

$\Rightarrow$ Up to 5.1$\times$ faster and 4.2$\times$ less memory with the same accuracy on the solution than DMUMPS!

Best time and memory w.r.t. DMUMPS



"Combining sparse approximate factorizations with mixed precision iterative refinement" by **P. Amestoy, A. Buttari, N. J. Higham, J-Y L'Excellent, T. Mary, B. Vieublé**, ACM TOMS.

Conclusion

Takeaways

- Many GMRES variants not covered by a backward error analysis.
- We propose a backward error analysis framework to efficiently derive error bounds on new variants.
- We can apply this framework to a five precisions GMRES algorithms.

It is still an ongoing work. No preprint available yet.

"*Five-Precision GMRES-based iterative refinement*" by **P. Amestoy, A. Buttari, N. J. Higham, J-Y L'Excellent, T. Mary, B. Vieublé**, Preprint.

"*Combining sparse approximate factorizations with mixed precision iterative refinement*" by **P. Amestoy, A. Buttari, N. J. Higham, J-Y L'Excellent, T. Mary, B. Vieublé**, ACM TOMS.

"*Mixed precision iterative refinement for the solution of large sparse linear systems*" by **B. Vieublé**, Ph.D. Thesis.