A convergence rates comparison of iterative refinement and GMRES for the refinement of inaccurate solvers

Alfredo Buttari, Xin Liu, Théo Mary, **Bastien Vieublé***

SIAM PP26

03/03/2026

Approximate computing for linear systems

$$Ax = b,$$

$$A \in \mathbb{R}^{n \times n}, \quad b \in \mathbb{R}^n, \quad x \in \mathbb{R}^n$$

Approximate computing for linear systems

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

Approximate computing: **deliberately approximate the computations** in order **to improve the performance** at the cost of **introducing a perturbation**.

- ➤ The perturbed problem should be close to the original one and should **reduce time and/or memory**!
- ➤ In general **the larger the perturbations the larger the savings**...
- ➤ BUT **large perturbations = low accuracy**!

Examples: low precision, randomization/sketching, low-rank approximations, tensor compression, etc.

Approximate computing for linear systems

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

Approximate computing: **deliberately approximate the computations** in order **to improve the performance** at the cost of **introducing a perturbation**.

- ➤ The perturbed problem should be close to the original one and should **reduce time and/or memory**!
- ➤ In general **the larger the perturbations the larger the savings**...
- ➤ BUT **large perturbations = low accuracy**!

Examples: low precision, randomization/sketching, low-rank approximations, tensor compression, etc.

$\Rightarrow$ **Iterative refinement and GMRES can correct cheaply the inaccuracies introduced by approximate computing.**

Background on iterative refinement

Algorithm: Iterative refinement

1: **while** not converged **do**
2: Compute the residual $r_k = b - Ax_k$.
3: Solve $Ad_k = r_k$ **inaccurately** and **cheaply**.
4: Compute the next iterate $x_{k+1} = x_k + d_k$.
5: **end while**

Background on iterative refinement

Algorithm: Iterative refinement

1: **while** not converged **do**
2: Compute the residual $r_k = b - Ax_k$.
3: Solve $Ad_k = r_k$ **inaccurately** and **cheaply**.
4: Compute the next iterate $x_{k+1} = x_k + d_k$.
5: **end while**

- Accelerate direct solvers:

  "*Using Mixed Precision for Sparse Matrix Computations to Enhance the Performance while Achieving 64-bit Accuracy*" by **Buttari et al.**, 2008.
- Accelerate iterative solvers:

  "*Mixed Precision Iterative Refinement Methods for Linear Systems: Convergence Analysis Based on Krylov Subspace Methods.*" by **Anzt et al.**, 2012.
- Accelerate multigrid solvers:

  "*Performance and accuracy of hardware-oriented native-, emulated- and mixed-precision solvers in FEM simulations*" by **Göddeke et al.**, 2007.

Background on GMRES

Consider 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
    $h_{k+1,k} = \|w_k\|, v_{k+1} = w_k/h_{k+1,k}$
    $V_k = [v_1, \ldots, v_k]$
    $H_k = \{h_{i,j}\}_{1\le i\le j+1; 1\le j\le k}$
    $y_k = \text{argmin}_y \|\beta e_1 - H_k y\|$
    $k = k + 1$
until $\|\beta e_1 - H_k y_k\| \le \tau$
$x_k = x_0 + V_k y_k$

Background on GMRES

Consider 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 $i = 1, \ldots, k$ **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 $\|\beta e_1 - H_k y_k\| \le \tau$
$x_k = x_0 + V_k y_k$

Background on GMRES

Consider 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
  $h_{k+1,k} = \|w_k\|, v_{k+1} = w_k/h_{k+1,k}$
  $V_k = [v_1, \ldots, v_k]$
  $H_k = \{h_{i,j}\}_{1\le i\le j+1; 1\le j\le k}$
  $y_k = \text{argmin}_y \|\beta e_1 - H_k y\|$
  $k = k + 1$
until $\|\beta e_1 - H_k y_k\| \le \tau$
$x_k = x_0 + V_k y_k$

Background on GMRES

Consider 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 $\|\beta e_1 - H_k y_k\| \le \tau$
$x_k = x_0 + V_k y_k$

Background on GMRES

Consider 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 satisfactory 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
  $h_{k+1,k} = \|w_k\|, v_{k+1} = w_k/h_{k+1,k}$
  $V_k = [v_1, \ldots, v_k]$
  $H_k = \{h_{i,j}\}_{1\le i\le j+1; 1\le j\le k}$
  $y_k = \text{argmin}_y \|\beta e_1 - H_k y\|$
  $k = k + 1$
until $\|\beta e_1 - H_k y_k\| \le \tau$
$x_k = x_0 + V_k y_k$

Preconditioned GMRES as a refinement

Preconditioner are ubiquitous to accelerate Krylov-based iterative solvers. Here are three ways to preconditioned GMRES with a preconditioner M:

- **Left-preconditioning (LGMRES):**

$$M^{-1}Ax = M^{-1}b.$$

- **Right-preconditioning (RGMRES):**

$$AM^{-1}u = b, \qquad x = M^{-1}u.$$

- **Flexible-preconditioning (FGMRES):** A variant of RGMRES storing an additional basis and allowing the preconditioner $M^{(k)}$ to vary from an iteration to another.

Preconditioned GMRES as a refinement

Preconditioner are ubiquitous to accelerate Krylov-based iterative solvers. Here are three ways to preconditioned GMRES with a preconditioner M:

- **Left-preconditioning (LGMRES):**

$$M^{-1}Ax = M^{-1}b.$$

- **Right-preconditioning (RGMRES):**

$$AM^{-1}u = b, \qquad x = M^{-1}u.$$

- **Flexible-preconditioning (FGMRES):** A variant of RGMRES storing an additional basis and allowing the preconditioner $M^{(k)}$ to vary from an iteration to another.

$\Rightarrow$ **GMRES can also refine an inaccurate linear solver by using it as a preconditioner.**

Which is the better refinement?

Consider an inaccurate solver that act as a constant operator M^{-1} and that can be refined by either iterative refinement and GMRES.

$\Rightarrow$**Can we say something about which is faster?**

Which is the better refinement?

Consider an inaccurate solver that act as a constant operator M^{-1} and that can be refined by either iterative refinement and GMRES.

$\Rightarrow$**Can we say something about which is faster?**

What we know from the literature:

- Experimental observation that FGMRES needs less iterations:

  "*Using FGMRES to obtain backward stability in mixed precision*" by **Arioli and Duff**, 2008.

- Inner-outer FGMRES-GMRES can be expected faster than restarted GMRES:

  "*Flexible Inner-Outer Krylov Subspace Methods*" by **Simoncini and Szyld**, 2002.

- Stationary iterative methods converge slower than Krylov subspace methods:

  "*An Introduction to Domain Decomposition Meth- ods: Algorithms, Theory, and Parallel Implementation*" by **Dolean et al.**, Chap. 3, 2015.

A misleading argument

➤ Iterative refinement is a stationary iteration:

$$x_{k+1} = x_k + M^{-1}(b - Ax_k).$$

➤ Taking $x_0 = 0$, the kth iterative refinement iterate satisfies

$$\begin{aligned} x_k &\in \mathcal{K}_k(M^{-1}A, M^{-1}b) \\ &\in span\{M^{-1}b, (M^{-1}A)(M^{-1}b), \ldots, (M^{-1}A)^{k-1}(M^{-1}b)\}. \end{aligned}$$

➤ On the other hand, LGMRES looks at each iteration for an optimal iterate x_k in the same space $\mathcal{K}_k(M^{-1}A, M^{-1}b)$:

$$\min_{x_k \in \mathcal{K}_k(M^{-1}A, M^{-1}b)} \|M^{-1}b - M^{-1}Ax_k\|$$

A misleading argument

➤ Iterative refinement is a stationary iteration:

$$x_{k+1} = x_k + M^{-1}(b - Ax_k).$$

➤ Taking $x_0 = 0$, the kth iterative refinement iterate satisfies

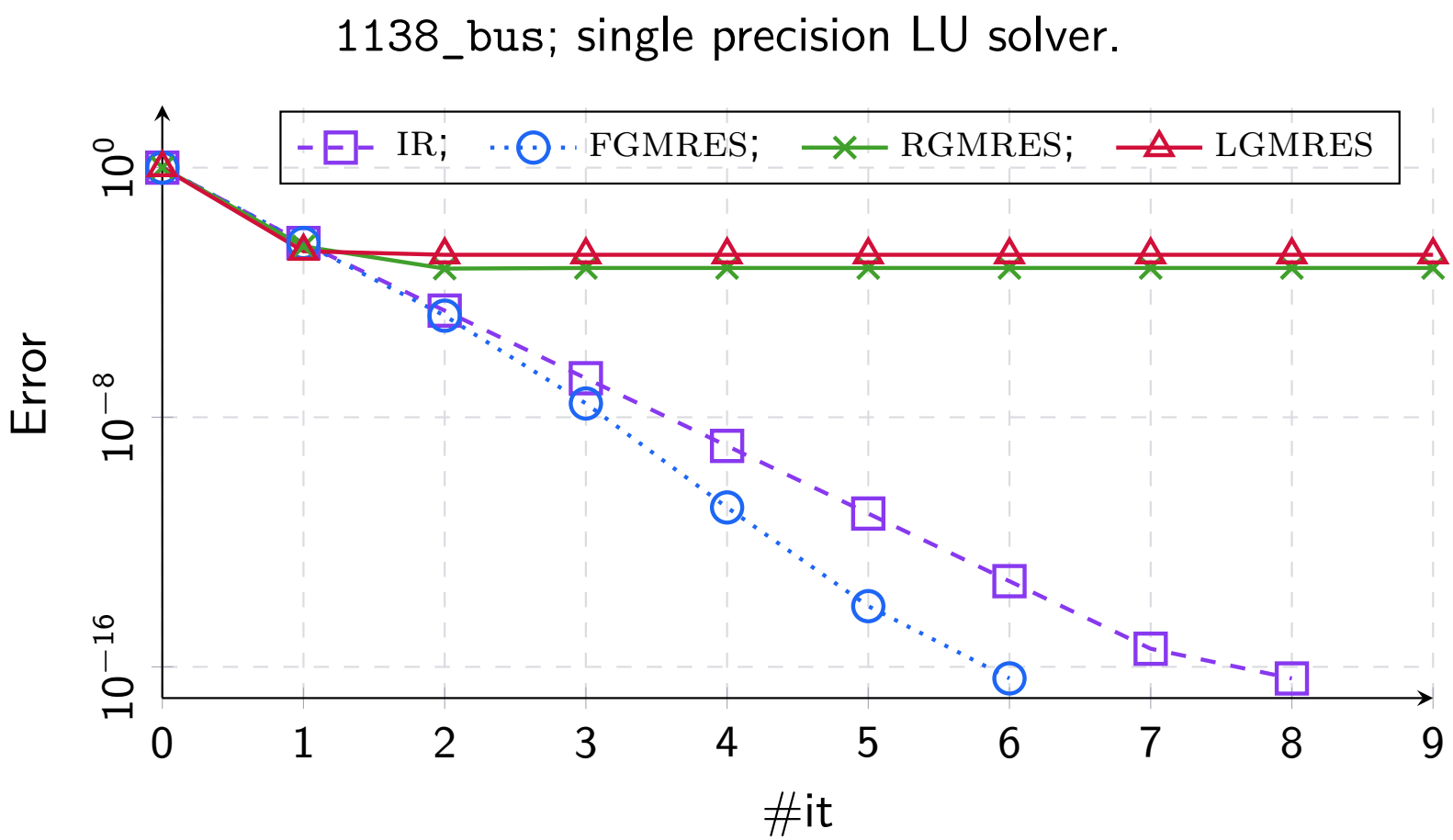
$$\begin{aligned} x_k &\in \mathcal{K}_k(M^{-1}A, M^{-1}b) \\ &\in span\{M^{-1}b, (M^{-1}A)(M^{-1}b), \ldots, (M^{-1}A)^{k-1}(M^{-1}b)\}. \end{aligned}$$

➤ On the other hand, LGMRES looks at each iteration for an optimal iterate x_k in the same space $\mathcal{K}_k(M^{-1}A, M^{-1}b)$:

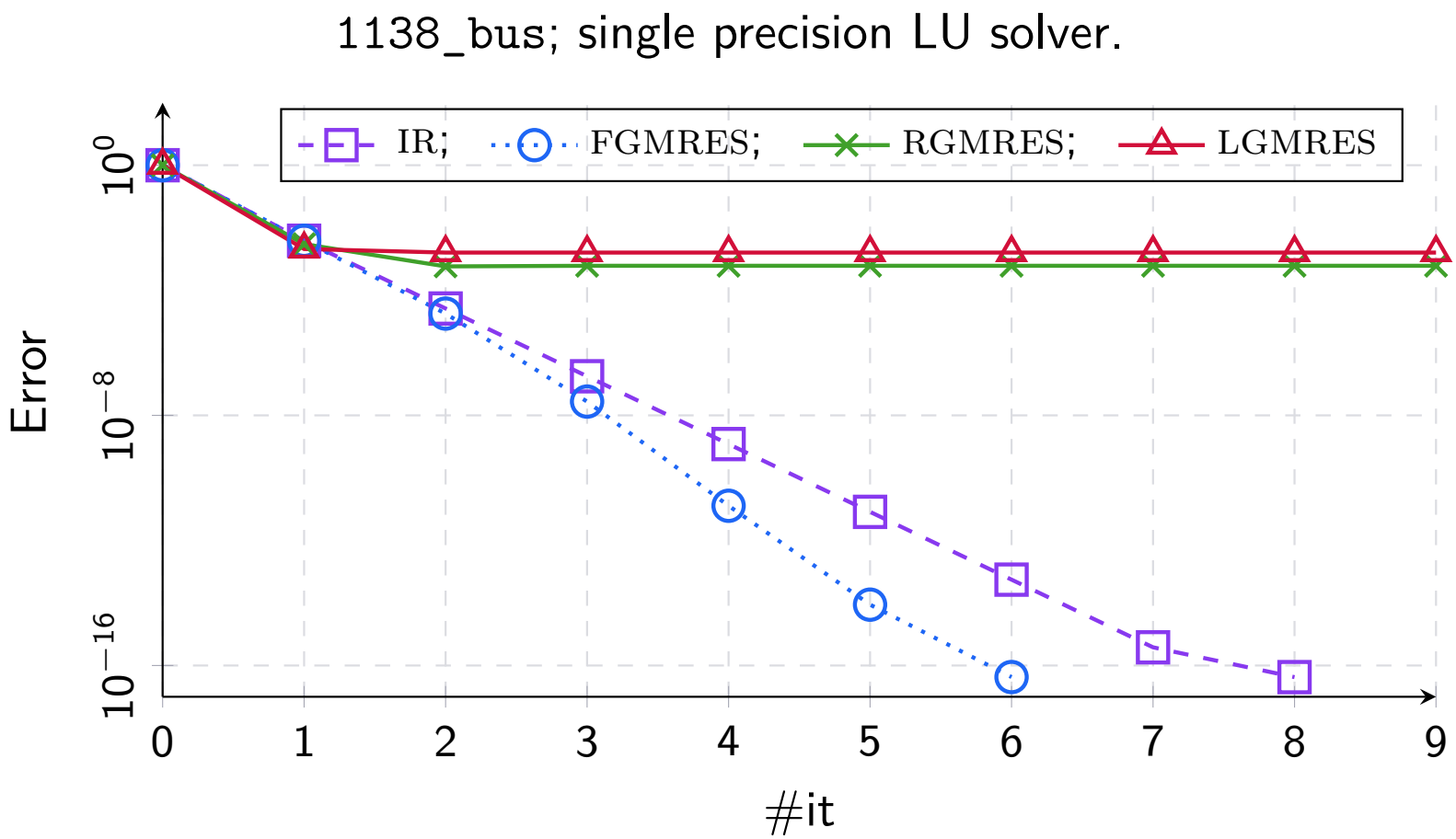
$$\min_{x_k \in \mathcal{K}_k(M^{-1}A, M^{-1}b)} \|M^{-1}b - M^{-1}Ax_k\|$$

$\Rightarrow$ We should expect LGMRES (and RGMRES/FGMRES) to converge faster than iterative refinement.

A counterexample



A counterexample



Why: Attainable accuracies of LGMRES and RGMRES are sensitive to the accuracy at which M^{-1} is applied.

"*Mixed precision strategies for preconditioned GMRES: a comprehensive analysis*" by **Buttari et al.**, 2025.

About comparing FGMRES and iterative refinement

Can we guarantee that FGMRES always refine faster than iterative refinement?

About comparing FGMRES and iterative refinement

Can we guarantee that FGMRES always refine faster than iterative refinement?

Sharp convergence rate for GMRES in the general case is an ongoing difficult problem:

- "*Any nonincreasing convergence curve is possible for GMRES*" by **Greenbaum et al.**, 1996.
- "*Any nonincreasing convergence curves are simultaneously possible for GMRES and weighted GMRES, as well as for left and right preconditioned GMRES*" by **Matalon and Spillane**, 2025.
- "*GMRES convergence bounds that depend on the right-hand-side vector*" by **Titley-Peloquin et al.**, 2013.

About comparing FGMRES and iterative refinement

Can we guarantee that FGMRES always refine faster than iterative refinement?

Sharp convergence rate for GMRES in the general case is an ongoing difficult problem:

- "*Any nonincreasing convergence curve is possible for GMRES*" by **Greenbaum et al.**, 1996.
- "*Any nonincreasing convergence curves are simultaneously possible for GMRES and weighted GMRES, as well as for left and right preconditioned GMRES*" by **Matalon and Spillane**, 2025.
- "*GMRES convergence bounds that depend on the right-hand-side vector*" by **Titley-Peloquin et al.**, 2013.

BUT we have strong assumptions on the preconditioner/inaccurate solver:

$$\widehat{d}_k - d_k = \boldsymbol{u_s} E_k d_k, \qquad \boldsymbol{u_s}\|E_k\| \leq 1.$$

Studying the convergence with a proxy algorithm

The $(k+1)$th iterate of iterative refinement satisfies:

$$x_{k+1} = x_1 + \qquad d_1 + \qquad d_2 + \cdots + \qquad d_k,$$

Studying the convergence with a proxy algorithm

The $(k+1)$th iterate of iterative refinement satisfies:

$$x_{k+1} = x_1 + \alpha_1^{(k)} d_1 + \alpha_2^{(k)} d_2 + \cdots + \alpha_k^{(k)} d_k,$$

with $y_k = [\alpha_1^{(k)}, \ldots, \alpha_k^{(k)}]^T = [1, \ldots, 1]^T$.

Studying the convergence with a proxy algorithm

The $(k+1)$th iterate of iterative refinement satisfies:

$$x_{k+1} = x_1 + \alpha_1^{(k)} d_1 + \alpha_2^{(k)} d_2 + \cdots + \alpha_k^{(k)} d_k,$$

with $y_k = [\alpha_1^{(k)}, \ldots, \alpha_k^{(k)}]^T = [1, \ldots, 1]^T$.

BUT we could choose better:

$$\underset{\alpha_1^{(k)},\ldots,\alpha_k^{(k)}}{\arg\min} \|b - A(x_1 + \alpha_1^{(k)} d_1 + \cdots + \alpha_k^{(k)} d_k)\|.$$

Studying the convergence with a proxy algorithm

The $(k+1)$th iterate of iterative refinement satisfies:

$$x_{k+1} = x_1 + \alpha_1^{(k)} d_1 + \alpha_2^{(k)} d_2 + \cdots + \alpha_k^{(k)} d_k,$$

with $y_k = [\alpha_1^{(k)}, \ldots, \alpha_k^{(k)}]^T = [1, \ldots, 1]^T$.

BUT we could choose better:

$$\underset{\alpha_1^{(k)},\ldots,\alpha_k^{(k)}}{\arg\min} \|b - A(x_1 + \alpha_1^{(k)} d_1 + \cdots + \alpha_k^{(k)} d_k)\|.$$

Algorithm: Optimalized iterative refinement

1: **while** not converged **do**
2: Compute the residual $r_k = b - Ax_k$.
3: Solve $Ad_k = r_k$ and store $D_k = [D_{k-1}, d_k]$.
4: Compute the next iterate $x_{k+1} = x_1 + D_k y_k$ where $y_k = \text{argmin}_y \|r_1 - AD_k y\|$.
5: **end while**

Convergence analysis

Theorem

The ratio between the (backward error) convergence rates of optimal iterative refinement and iterative refinement for all iteration $k \geq 1$ is

$$\frac{\|(I - Q_k Q_k^T)(A\widehat{d}_k - \widehat{r}_k)\|}{\|A\widehat{d}_k - \widehat{r}_k\|} \leq 1,$$

where $(I - Q_k Q_k^T)$ is an orthogonal projection on the complement of $span\{A[\widehat{d}_1, \ldots, \widehat{d}_k]\}$.

Convergence analysis

Theorem

The ratio between the (backward error) convergence rates of optimal iterative refinement and iterative refinement for all iteration $k \geq 1$ is

$$\frac{\|(I - Q_k Q_k^T)(A\widehat{d}_k - \widehat{r}_k)\|}{\|A\widehat{d}_k - \widehat{r}_k\|} \leq 1,$$

where $(I - Q_k Q_k^T)$ is an orthogonal projection on the complement of $span\{A[\widehat{d}_1, \ldots, \widehat{d}_k]\}$.

Equivalently, and under our assumptions on the solver, we write

$$(I - Q_k Q_k^T)(A\widehat{d}_k - \widehat{r}_k) = (I - Q_k Q_k^T)A(\widehat{d}_k - d_k) = \boldsymbol{u_s}(I - Q_k Q_k^T)AE_k d_k.$$

Convergence analysis

Theorem

The ratio between the (backward error) convergence rates of optimal iterative refinement and iterative refinement for all iteration $k \geq 1$ is

$$\frac{\|(I - Q_k Q_k^T)(A\widehat{d}_k - \widehat{r}_k)\|}{\|A\widehat{d}_k - \widehat{r}_k\|} \leq 1,$$

where $(I - Q_k Q_k^T)$ is an orthogonal projection on the complement of $span\{A[\widehat{d}_1, \ldots, \widehat{d}_k]\}$.

Equivalently, and under our assumptions on the solver, we write

$$(I - Q_k Q_k^T)(A\widehat{d}_k - \widehat{r}_k) = (I - Q_k Q_k^T)A(\widehat{d}_k - d_k) = \boldsymbol{u_s}(I - Q_k Q_k^T)AE_k d_k.$$

$\Rightarrow$ **The more** $span\{\widehat{d}_1, \ldots, \widehat{d}_k\}$ **spans** $E_k d_k$**, the faster is optimal iterative refinement over iterative refinement.**

The link to FGMRES

Algorithm: Optimal IR as Flexible Simpler GMRES

Initialize x_1.
$r_1 = b - Ax_1$
repeat
    Solve $Ad_i = r_i/\|r_i\|$ and form $D_i = [D_{i-1}, d_i]$
    $w_i = Ad_i$
    for $l = 1 : i-1$ **do**
        $h_{l,i} = v_l^T w_i$
        $w_i = w_i - h_{l,i} v_l$
    end for
    $h_{i,i} = \|w_i\|_2$
    Compute $v_i = w_i/h_{i,i}$ and store $V_i = [V_{i-1},\ v_i]$.
    Compute $r_{i+1} = r_i - \beta_i v_i$, where $\beta_i = r_i^T v_i$ and $b_i = [b_{i-1}, \beta_i]$.
until convergence
Solve the triangular system $H_i t_i = b_i$.
$x_i = x_1 + D_i t_i$

Numerical experiments

Number of refinement iterations; LU factors in BFloat16; $n \leq 1000$; $\|x - \widehat{x}_k\|/\|x\| \leq 10^{-14}$.

Matrix	IR	Simpler FGMRES	FGMRES
pores_1	25	21	**19**
fs_680_1	18	**11**	**11**
cz148	32	**15**	**15**
ck104	32	**20**	**20**
cz308	117	**19**	22
fs_680_3	33	**17**	**17**
Si2	51	**19**	**19**
LFAT5	28	**12**	**12**
tub100	73	**31**	33
bcsstk34	37	**21**	**21**

A truncated scalable version

For very large scale problems, there might be instances where storing the full dense basis of correction vectors $[\widehat{d}_1, \ldots, \widehat{d}_k]$ is too expensive.

$\Rightarrow$ We can **optimalize on a subset of j corrections** instead!

A truncated scalable version

For very large scale problems, there might be instances where storing the full dense basis of correction vectors $[\widehat{d}_1, \ldots, \widehat{d}_k]$ is too expensive.

$\Rightarrow$ We can **optimalize on a subset of j corrections** instead!

Algorithm: Truncated optimalized iterative refinement

1: **while** not converged **do**
2: Compute the residual $r_k = b - Ax_k$.
3: Solve $Ad_k = r_k$ and store $\widetilde{D}_k = [d_{k+1-j}, \ldots, d_k]$.
4: Compute the next iterate $x_{k+1} = x_{k+1-j} + \widetilde{D}_k y_k$ where $y_k = \operatorname{argmin}_y \|r_1 - A(x_{k+1-j} + \widetilde{D}_k y)\|$.
5: **end while**

A truncated scalable version

For very large scale problems, there might be instances where storing the full dense basis of correction vectors $[\widehat{d}_1, \ldots, \widehat{d}_k]$ is too expensive.

$\Rightarrow$ We can **optimalize on a subset of j corrections** instead!

Theorem

The ratio between the (backward error) convergence rates of truncated iterative refinement and iterative refinement is

$$\frac{\|(I - \widetilde{Q}_k \widetilde{Q}_k^T)(A\widehat{d}_k - \widehat{r}_k)\|}{\|A\widehat{d}_k - \widehat{r}_k\|} \leq 1,$$

where $(I - \widetilde{Q}_k \widetilde{Q}_k^T)$ is an orthogonal projection on the complement of $span\{A[\widehat{d}_{k+1-j}, \ldots, \widehat{d}_k]\}$.

Numerical experiments

Number of refinement iterations; LU factors in BFloat16; $n \leq 1000$; $\|x - \widehat{x}_k\|/\|x\| \leq 10^{-14}$.

Matrix	IR	$j = 1$	$j = 3$	$j = 10$	FGMRES
pores_1	25	21	21	21	**19**
fs_680_1	18	15	13	11	**11**
cz148	32	26	20	16	**15**
ck104	32	25	22	21	**20**
cz308	117	83	40	23	**22**
fs_680_3	33	23	19	18	**17**
Si2	51	35	26	21	**19**
LFAT5	28	—	—	12	**12**
tub100	73	51	41	38	**33**
bcsstk34	37	29	23	22	**21**

Conclusion

Use FGMRES instead of iterative refinement!

"*A convergence rates comparison of iterative refinement and GMRES for the refinement of inaccurate solvers*" by **Buttari et al.**, Coming soon.


juliacon
global 2026
Minisymposium
Approximate Computing in Numerical Linear Algebra
Jonas Schulze, Bastien Vieublé, Nicolas Venkovic, Andreas Varga, Mantas Mikaitis, and you?
This minisymposium features talks from researchers and practitioners describing their use of Julia for studying the numerical behavior or leveraging the computational benefits of approximate computing techniques. The purpose of this minisymposium is to provide a space where people can share and talk about the newest Julia software developments in approximate computing for numerical linear algebra (and beyond).
We wish to promote Julia as a tool for driving advancements in approximate computing and highlight the latest exciting Julia packages in this field. We will end the minisymposium with a panel discussion around the development of an approximate computing ecosystem in Julia.
Johannes Gutenberg University
Mainz, Germany
August 10-15 2026
visit https://juliacon.org/