Check for updates

Received: 13 July 2021 | Revised: 12 November 2021 | Accepted: 19 January 2022

DOI: 10.1002/nla.2434

RESEARCH ARTICLE

WILEY

Multistage mixed precision iterative refinement

Eda Oktay | **Erin Carson**

Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Correspondence
Eda Oktay, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic.
Email: oktay@karlin.mff.cuni.cz

Funding information
U.S. Department of Energy; Univerzita Karlova v Praze; Charles University, Grant/Award Numbers: PRIMUS/19/SCI/11, UNCE/SCI/023; Exascale Computing Project, Grant/Award Numbers: 17-SC-20-SC, UNCE/SCI/023

Abstract

Low precision arithmetic, in particular half precision (16-bit) floating point arithmetic, is now available in commercial hardware. Using lower precision can offer significant savings in computation and communication costs with proportional savings in energy. Motivated by this, there has been a renewed interest in mixed precision iterative refinement schemes for solving linear systems $Ax = b$, and new variants of GMRES-based iterative refinement have been developed. Each particular variant with a given combination of precisions leads to different condition number-based constraints for convergence of the backward and forward errors, and each has different performance costs. The constraints for convergence given in the literature are, as an artifact of the analyses, often overly strict in practice, and thus could lead a user to select a more expensive variant when a less expensive one would have sufficed. In this work, we develop a multistage mixed precision iterative refinement solver which aims to combine existing mixed precision approaches to balance performance and accuracy and improve usability. For a user-specified initial combination of precisions, the algorithm begins with the least expensive approach and convergence is monitored via inexpensive computations with quantities produced during the iteration. If slow convergence or divergence is detected using particular stopping criteria, the algorithm switches to use a more expensive, but more reliable variant. A novel aspect of our approach is that, unlike existing implementations, our algorithm first attempts to use "stronger" GMRES-based solvers for the solution update before resorting to increasing the precision(s). In some scenarios, this can avoid the need to refactorize the matrix in higher precision. We perform extensive numerical experiments on a variety of random dense problems and problems from real applications which confirm the benefits of the multistage approach.

KEYWORDS

adaptive algorithms, GMRES, iterative refinement, mixed precision

1 | INTRODUCTION

Iterative refinement (IR) is frequently used in solving linear systems $Ax = b$, where $A \in \mathbb{R}^{n\times n}, x, b \in \mathbb{R}^n$, to improve the accuracy of a computed approximate solution $\hat{x} \in \mathbb{R}^n$. Typically, one computes an initial approximate solution $\hat{x}_0 \in \mathbb{R}^n$

Funding information: We acknowledge funding from the Charles University PRIMUS project No. PRIMUS/19/SCI/11, Charles University Research Program No. UNCE/SCI/023, and the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

 © 2022 John Wiley & Sons, Ltd.

using Gaussian elimination with partial pivoting, saving the approximate factorization $A \approx \hat{L}\hat{U}$, where $\hat{L} \in \mathbb{R}^{n\times n}$ is a lower triangular matrix with unit diagonal, and $\hat{U} \in \mathbb{R}^{n\times n}$ is an upper triangular matrix. After computing the residual $\hat{r} = b - A\hat{x} \in \mathbb{R}^n$ (potentially in higher precision), one reuses $\hat{L}$ and $\hat{U}$ to solve the system $A\hat{d} = \hat{r}$, where $\hat{d} \in \mathbb{R}^n$. The original approximate solution is subsequently refined by adding the corrective term, $\hat{x} = \hat{x} + \hat{d}$. This process can be repeated until either the desired accuracy is reached or the IR process converges to an approximate solution. IR algorithms that exploit multiple different precisions have seen renewed attention recently due to the emergence of mixed precision hardware, and form the basis for the recently developed high performance linpack – artificial intelligence (HPL-AI) benchmark, on which today's top supercomputers exceed exascale performance (see e.g., References 1-3).

Algorithm 1 shows a general mixed precision IR scheme. Following the work of Reference 4, there are four different precisions specified: u_f denotes the factorization precision, u denotes the working precision, u_r denotes the precision for the residual computation, and u_s denotes the effective precision with which the correction equation is solved. This final precision, u_s, is not a hardware precision but will rather depend on the particular solver used in line 5 and the particular precision(s) used within the solver (which may be different from u, u_f, and u_r). It is assumed that $u_f \geq u \geq u_r$.

Algorithm 1. General iterative refinement scheme

Input: $n \times n$ matrix A; right-hand side b; maximum number of refinement steps $i_{\max} \in \mathbb{N}^+$.
Output: Approximate solution x_{i+1} to $Ax = b$.

1: Compute LU factorization $A = LU$ in precision u_f.
2: Solve $Ax_0 = b$ by substitution in precision u_f; store x_0 in precision u.
3: **for** $i = 0$: $i_{\max} - 1$ **do**
4: Compute $r_i = b - Ax_i$ in precision u_r; store in precision u.
5: Solve $Ad_{i+1} = r_i$ in precision u_s; store d_{i+1} in precision u.
6: Compute $x_{i+1} = x_i + d_{i+1}$ in precision u.
7: **if** converged **then** return x_{i+1}. **end if**
8: **end for**

The choice of precisions u, u_f, and u_r, and the choice of a solver to use in line 5 defines different variants of IR. We refer to any variant that solves the correction equation in line 5 using triangular solves with the computed LU factors as "standard iterative refinement" (SIR). For SIR, the effective precision of the correction solve is limited by the precision with which the factorization is computed, and thus we have $u_s = u_f$. The most commonly used SIR variant is what we call "traditional" IR, in which $u_f = u$ and $u_r = u^2$. For instance, if the working precision is single, that is, $u = 5.96 \cdot 10^{-8}$, then for $u_r = u^2 \approx 10^{-16}$, double precision is used. Traditional IR was used already by Wilkinson in 1948,[5] and was analyzed in fixed point arithmetic by Wilkinson in 1963[6] and in floating point arithmetic by Moler in 1967.[7] There have also been analyses of fixed precision IR, in which $u_f = u = u_r$,[8] as well as low precision factorization IR, in which $u_f^2 = u = u_r$.[9] Motivated by the trend of low and mixed precision capabilities in hardware, the authors in Reference 4 developed and analyzed a three-precision IR scheme, in which u_f, u, and u_r may differ, which generalizes (and in some cases, improves the bounds for) many existing variants. For references to analyses of variants of IR, see table 1.1 of Reference 4.

If A is very ill-conditioned or badly scaled, SIR can fail, that is, the error is not eventually bounded by a small multiple of machine precision. In extreme cases of ill-conditioning, the error can grow with each refinement step. In References 4 and 10, the authors developed a mixed precision GMRES-based IR scheme (GMRES-IR), shown in Algorithm 2. The only difference with SIR is the way in which the correction equation is solved. The idea is that instead of using the LU factors to solve for the correction in each step, one can instead use these factors as (left) preconditioners for a preconditioned GMRES method which solves for the correction. In this way, the effective solve precision becomes $u_s = u$, and more ill-conditioned problems can be handled relative to SIR. The GMRES-based refinement approaches have seen much success in practice. Current GMRES-based IR variants are implemented in the MAGMA library (2.5.0) and in the NVIDIA cuSOLVER library. GMRES-IR also forms the basis for the new HPL-AI benchmark, used to rank computers in the TOP500 list.[1] Experiments detailing the performance benefits of three-precision GMRES-IR and SIR can be found in Reference 11.

Algorithm 2. GMRES-IR [10]

Input: $n \times n$ matrix A; right-hand side b; maximum number of refinement steps $i_{\max}$; GMRES convergence tolerance τ.
Output: Approximate solution $\hat{x}$ to $Ax = b$.

1: Compute LU factorization $A = LU$ in precision u_f.
2: Solve $Ax_0 = b$ by substitution in precision u_f; store x_0 in precision u.
3: **for** $i = 0$: $i_{\max} - 1$ **do**
4: Compute $r_i = b - Ax_i$ in precision u_r; store in precision u.
5: Solve $U^{-1}L^{-1}Ad_{i+1} = U^{-1}L^{-1}r_i$ by GMRES in working precision u, with matrix-vector products with $\tilde{A} = U^{-1}L^{-1}A$ computed at precision u^2; store d_{i+1} in precision u.
6: Compute $x_{i+1} = x_i + d_{i+1}$ in precision u.
7: **if** converged **then** return x_{i+1}. **end if**
8: **end for**

Although GMRES-IR can succeed where SIR fails, each step of GMRES-IR is potentially more expensive than each step of SIR (albeit still potentially less expensive than running the entire process in double the working precision). While each SIR refinement step involves only two triangular solves in precision u, each GMRES-IR step involves two triangular solves in precision u^2 in *each* GMRES iteration (in addition to a matrix-vector product in precision u^2 as well as multiple vector operations in precision u). The convergence behavior of each GMRES solve thus plays a large role, and it is important for the performance of GMRES-IR that each call to GMRES converges relatively quickly. It is additionally a performance concern that each GMRES iteration requires computations in higher precision; in order to obtain the needed bound on backward error of the GMRES solve, the authors in References 4, 10 required that the preconditioned system $U^{-1}L^{-1}A$ (not formed explicitly) is applied to a vector in precision u^2.

The authors of References 4, 10 suggested that this required use of extra precision was likely to be overly strict in many practical scenarios. Indeed, most practical implementations of GMRES-based IR do not use this extra precision.[12] This motivated Amestoy et al.[12] to develop an extension of the GMRES-IR algorithm, called GMRES-IR5. The authors in Reference 12 revisit the proof of backward stability for GMRES from Reference 13 and develop a bound on the backward error for a mixed precision GMRES algorithm, where $u_g \geq u$ is the working precision used within GMRES and u_p is the precision in which the preconditioned matrix is applied to a vector. This enables a five-precision GMRES-IR scheme, where u, u_f, u_r, u_g, and u_p may take on different values. A particular improvement over the GMRES-IR scheme of References 4,10 is that the analysis provides bounds on forward and backward errors for a variant of GMRES-IR in which the entire GMRES iteration is carried out in a uniform precision, that is, the analysis does not require that extra precision is used in applying the preconditioned matrix to a vector. This particular variant of the algorithm is thus less expensive than the former GMRES-IR in terms of both time and memory. The drawback is that it is only theoretically applicable to a smaller set of problems due to a tighter constraint on condition number. We call the particular instance of GMRES-IR5 where $u = u_g = u_p$ "SGMRES-IR" (for "simpler"), shown in Algorithm 3. To make more precise the relative costs of each step of SIR, SGMRES-IR, and GMRES-IR, we list their costs in terms of asymptotic computational complexity in Table 1. We discuss the constraints on condition number under which each variant converges later in Section 2.2.

Algorithm 3. SGMRES-IR (a particular variant of GMRES-IR5 [12])

Input: $n \times n$ matrix A; right-hand side b; maximum number of refinement steps $i_{\max}$; GMRES convergence tolerance τ.
Output: Approximate solution x_{i+1} to $Ax = b$.

1: Compute LU factorization $A = LU$ in precision u_f.
2: Solve $Ax_0 = b$ by substitution in precision u_f; store x_0 in precision u.
3: **for** $i = 0$: $i_{\max} - 1$ **do**
4: Compute $r_i = b - Ax_i$ in precision u_r; store in precision u.
5: Solve $U^{-1}L^{-1}Ad_{i+1} = U^{-1}L^{-1}r_i$ by GMRES in working precision u, with matrix-vector products with $\tilde{A} = U^{-1}L^{-1}A$ computed at precision u; store d_{i+1} in precision u.
6: Compute $x_{i+1} = x_i + d_{i+1}$ in precision u.
7: **if** converged **then** return x_{i+1}. **end if**
8: **end for**

TABLE 1 Asymptotic computational complexity of operations in each refinement step for standard iterative refinement (SIR), SGMRES-IR, and GMRES-IR

Once per IR solve (all variants)	$O(n^3)$	In precision u_f	(LU fact.)
SIR step	$O(n^2)$	In precision u_f	(tri. solves)
SGMRES-IR step (k GMRES iterations)	$O(nk^2)$	In precision u	(orthog.)
	$O(nnz \cdot k)$	In precision u	(SpMV)
	$O(n^2k)$	In precision u	(precond.)
GMRES-IR step (k GMRES iterations)	$O(nk^2)$	In precision u	(orthog.)
	$O(nnz \cdot k)$	In precision u^2	(SpMV)
	$O(n^2k)$	In precision u^2	(precond.)
Once per refinement step (all variants)	$O(nnz)$	In precision u_r	(residual comp.)
	$O(n)$	In precision u	(sol. update)

In terms of both cost and range of condition numbers to which the algorithm can be applied, we expect SGMRES-IR to be, in general, somewhere between SIR and GMRES-IR. For example, if we use single precision for u_f, double precision for u, and quadruple precision for u_r, SIR is guaranteed to converge to the level of double precision in both forward and backward errors as long as the infinity-norm condition number of the matrix A, $\kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$, is less than $2 \cdot 10^7$. For SGMRES-IR, this constraint becomes $\kappa_\infty(A) \leq 10^{10}$. GMRES-IR, on the other hand, only requires $\kappa_\infty(A) \leq 2 \cdot 10^{15}$. Thus going from SIR to SGMRES-IR to GMRES-IR, we expect that these algorithms will be increasingly expensive but also expect that they will converge for increasingly ill-conditioned matrices. We note this may not always be the case. For some precision combinations, SGMRES-IR has a tighter constraint on condition number than SIR (see Reference 12), although this may be an artifact of the analysis. Also, the metric of relative "cost" is difficult to determine a priori, in particular between SGMRES-IR and GMRES-IR, since it depends on the number of GMRES iterations required in each refinement step.

It is thus difficult to choose a priori which particular variant of IR is the most appropriate for a particular problem. Even if the matrix condition number meets the constraint for convergence for the chosen IR algorithm, the convergence rate may be unacceptably slow, or each step may require so many GMRES iterations that it becomes impractical. Further, as our experiments show, the condition number constraints in the literature can be too tight, meaning that for some problems a given refinement scheme converges even when the analysis indicates that it may not. This could lead users to select a more expensive IR variant than is actually needed in practice.

In this work, we aim to solve this problem through the development of a multistage, three-precision IR scheme, which we abbreviate MSIR. Our approach automatically switches between solvers and precisions if slow convergence (of the refinement scheme itself or of the inner GMRES solves) is detected using stopping criteria adapted from the work in Reference 14. Two novel aspects of our approach are (1) we attempt to use "stronger" solvers before resorting to increasing the precision of the factorization, and (2) when executing a GMRES-based refinement algorithm, we modify the stopping criteria to also restrict the number of GMRES iterations per refinement step.

Table 1 shows why first switching the solver may be more favorable from a performance perspective; whereas increasing the precision and recomputing the factorization will cost $O(n^3)$ flops in precision u_f^2, where u_f is the current factorization precision, using the existing factorization and performing k total GMRES iterations may be faster, requiring $O(n^2k)$ flops in precision u. This motivates point (2) above. If the number of GMRES iterations k is too large, then one (S)GMRES-IR refinement step is at least as expensive as recomputing the factorization in a higher precision (u for SGMRES-IR or u^2 for GMRES-IR).

Our approach may serve to improve existing multistage IR implementations. For example, the MAGMA library[15] currently uses a variant of SGMRES-IR and if convergence is not detected after the specified maximum number of iterations, the factorization is recomputed in a higher precision and the refinement restarts. Our numerical experiments confirm that it may be beneficial to first try a different solver before resorting to recomputing the factorization.

In Section 2, we present the MSIR algorithm and give a motivating numerical example. We then give details of the stopping criteria used and summarize the analysis for each algorithm variant from[4] and.[12] In Section 3 we present more

thorough numerical experiments on both random dense matrices and matrices from the SuiteSparse collection.[16] We conclude and discuss future extensions in Section 4.

2 | THE MULTISTAGE IR ALGORITHM ALGORITHM

In order to balance reliability and cost, we develop a multistage IR algorithm (MSIR), presented in Algorithm 4. The algorithm starts with three-precision SIR (as in Reference 4) and, using the stopping criteria developed in Reference 14, switches to SGMRES-IR if the algorithm is not converging at an acceptable rate (or not converging at all). Then, using the same stopping criteria along with an additional constraint on the number of GMRES iterations per refinement step, the algorithm may choose to switch a second time to the GMRES-IR algorithm (which uses higher precision in applying the preconditioned matrix to a vector within GMRES). If convergence is still not achieved or is too slow, then as a fail-safe, we increase the factorization precision u_f (and the other precisions if necessary to satisfy $u_f \geq u$, $u_r \leq u^2$), recompute the LU factorization, and begin the process again with SIR. We enforce $u_r \leq u^2$ in order to guarantee the convergence of the forward error to the level of the working precision, but this strategy for increasing precisions could be modified in practice. For instance, if the initial setting is $(u_f, u, u_r) =$ (half, single, double), we would double only u_f and use $(u_f, u, u_r) =$ (single, single, double), and then if we need to increase precisions again, we would use $(u_f, u, u_r) =$ (double, double, quad).

Before explaining the details of the algorithm, we begin with a brief motivating example illustrating how MSIR works. We restrict ourselves to IEEE precisions and use initial precisions $(u_f, u, u_r) =$ (single, double, quad) and test random dense matrices of size 100×100 generated using the MATLAB command `gallery('randsvd',n,kappa(i),2)`, where `kappa` is the array of the desired 2-norm condition numbers. Here we test 2-norm condition numbers 10^1, 10^5, and 10^{16}. We set b to be a vector of normally distributed numbers generated by the MATLAB command `randn`. For reproducibility, we use the MATLAB command `rng(1)` to seed the random number generator before generating each linear system. The GMRES convergence tolerance $\tau \in \mathbb{R}^+$, which also appears in Algorithms 2 and 3, dictates the stopping criterion for the inner GMRES iterations. The algorithm is considered to be converged if the relative (preconditioned) residual norm drops below τ. In all tests here we use $\tau = 10^{-6}$. The particular criteria by which we switch between solvers is discussed in detail in Section 2.1.1.

Figures 1, 2, and 3 show results for condition numbers 10^1, 10^5, and 10^{16}, respectively. The red, blue, and green lines in the figures show the behavior of the forward error `ferr`(red), normwise relative backward error `nbe` (blue), and componentwise relative backward error `cbe`(green). The dotted black line shows the value of the initial working precision u. If there is a switch in MSIR, the step at which it happened is marked with a star. For instance, if MSIR uses both SGMRES-IR and GMRES-IR, then there are two stars: the first one marks the switch from SIR to SGMRES-IR and the second one marks the switch from SGMRES-IR to GMRES-IR.

For figures in this study, forward error is calculated as $\|\hat{x} - x\|_\infty / \|x\|_\infty$. The normwise relative backward error for $\hat{x}$ is calculated as

$$\frac{\|b - A\hat{x}\|}{\|A\|\|\hat{x}\| + \|b\|},$$

whereas for the computation of the componentwise relative backward error,

$$\max_i \frac{|b - A\hat{x}|_i}{(|A||\hat{x}| + |b|)_i},$$

is used.

The number of refinement steps performed by SIR, SGMRES-IR, GMRES-IR, and MSIR for these matrices are presented in Table 2. For SIR, the number in each row gives the number of refinement steps. For SGMRES-IR and GMRES-IR, the number of parenthetical elements gives the total number of refinement steps, and element i in the list gives the number of GMRES iterations performed in refinement step i. For example, (3, 4) indicates that two refinement steps were performed, the first of which took three GMRES iterations and the second of which took four. For the MSIR column, the data for SIR, SGMRES-IR, and GMRES-IR are comma-separated. For example, 2, (2) indicates that two SIR steps were performed, the algorithm then switched to SGMRES-IR, and performed one SGMRES-IR step which required two GMRES iterations. Since there is no second set of parentheses, this indicates that there was no switch to GMRES-IR. In all columns,

Algorithm 4. Multistage iterative refinement (MSIR)

Input: $n \times n$ matrix A; right-hand side b; maximum number of refinement steps of each type $i_{\max}$; GMRES convergence tolerance τ; stopping criteria parameter ρ_{thresh}; maximum GMRES iterations $k_{\max} \in \mathbb{N}^+$; initial factorization precision u_f; initial working precision u; initial residual precision u_r.

Output: Approximate solution x_{i+1} to $Ax = b$, boolean `cged`.

Compute LU factorization $A = LU$ in precision u_f.
Solve $Ax_0 = b$ by substitution in precision u_f; store x_0 in precision u.
Initialize: $d_0 = \infty$; `alg` = `SIR`; `iter` = 0; $i = 0$; `cged` = 0; $\rho_{\max} = 0$.
while not `cged` **do**
  Compute $r_i = b - Ax_i$ in precision u_r; Scale $r_i = r_i/\|r_i\|_\infty$; store in precision u.
  if `alg` = `SIR` **then**
    `iter` = `iter` +1
    Compute $d_{i+1} = U^{-1}(L^{-1}r_i)$ in precision u_f; store d_{i+1} in precision u.
    if d_{i+1} contains `Inf` or `NaN` **then** `alg` = `SGMRES-IR`; `iter` = 0; break. **end if**
    Compute $x_{i+1} = x_i + \|r_i\|_\infty d_{i+1}$ in precision u.
    $z = \|d_{i+1}\|_\infty/\|x_i\|_\infty$; $\quad v = \|d_{i+1}\|_\infty/\|d_i\|_\infty$; $\quad \rho_{\max} = \max(\rho_{\max}, v)$; $\quad \phi_i = z/(1-\rho_{\max})$
    if $z \le u$ or $v \ge \rho_{\text{thresh}}$ or `iter` $> i_{\max}$ or $\phi_i \le \sqrt{n}u$ **then**
      if not converged **then**
        `alg` = `SGMRES-IR`; `iter` = 0.
        if $\phi_i > \phi_0$ **then** $x_{i+1} = x_0$ **end if**
      else
        `cged` = 1
      end if
    end if
  else if `alg` = `SGMRES-IR` **then**
    `iter` = `iter` +1
    Solve $U^{-1}L^{-1}Ad_{i+1} = U^{-1}L^{-1}r_i$ by GMRES in precision u with matrix-vector products with $\tilde{A} = U^{-1}L^{-1}A$ computed at precision u; store d_{i+1} in precision u.
    Compute $x_{i+1} = x_i + \|r_i\|_\infty d_{i+1}$ in precision u.
    $z = \|d_{i+1}\|_\infty/\|x_i\|_\infty$; $\quad v = \|d_{i+1}\|_\infty/\|d_i\|_\infty$; $\quad \rho_{\max} = \max(\rho_{\max}, v)$; $\quad \phi_i = z/(1-\rho_{\max})$
    if $z \le u$ or $v \ge \rho_{\text{thresh}}$ or `iter` $> i_{\max}$ or $k_{\text{GMRES}} > k_{\max}$ or $\phi_i \le \sqrt{n}u$ **then**
      if not converged **then**
        `alg` = `GMRES-IR`; `iter` = 0.
        if $\phi_i > \phi_0$ **then** $x_{i+1} = x_0$ **end if**
      else
        `cged` = 1
      end if
    end if
  else if `alg` = `GMRES-IR` **then**
    `iter` = `iter` +1
    Solve $U^{-1}L^{-1}Ad_{i+1} = U^{-1}L^{-1}r_i$ by GMRES in precision u with matrix-vector products with $\tilde{A} = U^{-1}L^{-1}A$ computed at precision u^2; store d_{i+1} in precision u.
    Compute $x_{i+1} = x_i + \|r_i\|_\infty d_{i+1}$ in precision u.
    $z = \|d_{i+1}\|_\infty/\|x_i\|_\infty$; $\quad v = \|d_{i+1}\|_\infty/\|d_i\|_\infty$; $\quad \rho_{\max} = \max(\rho_{\max}, v)$; $\quad \phi_i = z/(1-\rho_{\max})$
    if $z \le u$ or $v \ge \rho_{\text{thresh}}$ or `iter` $> i_{\max}$ or $k_{\text{GMRES}} > k_{\max}$ or $\phi_i \le \sqrt{n}u$ **then**
      if not converged **then**
        $u_f = u_f^2$; `alg` = `SIR`; `iter` = 0.
        Compute LU factorization $A = LU$ in precision u_f.
        if $u_f < u$ **then** $u = u_f$ **end if**
        if $u_r > u^2$ **then** $u_r = u^2$ **end if**
        if $\phi_i > \phi_0$ **then** $x_{i+1} = x_0$ **end if**
      else
        `cged` = 1
      end if
    end if
  end if
  $i = i + 1$
end while

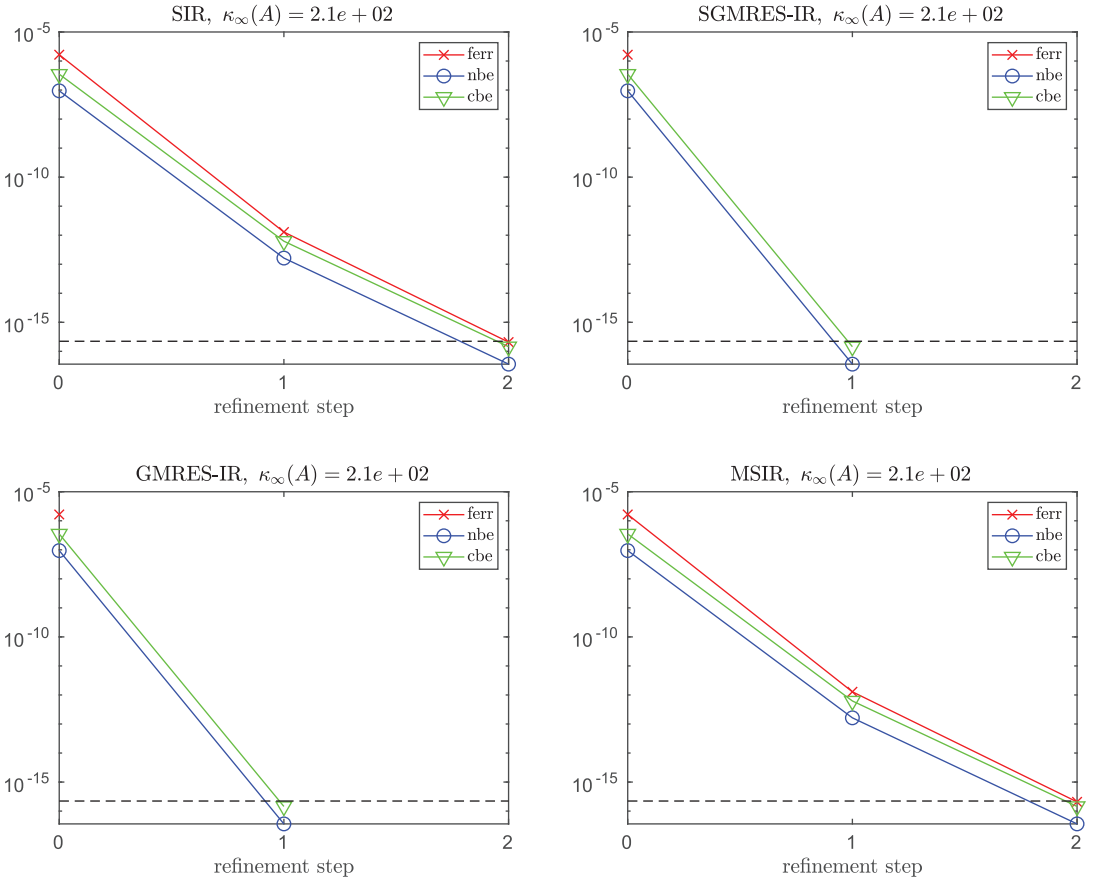

FIGURE 1 Convergence of errors for a 100×100 random dense matrix with $\kappa_2(A) = 10^1$ using standard iterative refinement (SIR) (top left), SGMRES-IR (top right), GMRES-IR (bottom left), and multistage IR (MSIR) (bottom right), with initial precisions $(u_f, u, u_r) =$ (single, double, quad). Note that for SGMRES-IR and GMRES-IR, the forward error is measured as 0 (in double precision) after one refinement step

a dash denotes that the algorithm diverged or made no progress; to enable a fair comparison, in our experiments we set $i_{\max}$ to be a very high value (here $i_{\max} = 2000$) in order to allow all approaches that eventually converge sufficient time to do so.

From Figure 1, we can see that for well-conditioned matrices, SIR quickly converges to the solution. When the condition number increases closer to $u_f^{-1} = (5.96 \cdot 10^8)^{-1} \approx 1.7 \cdot 10^7$ as in Figure 2, however, SIR convergence begins to slow down, and MSIR switches to SGMRES-IR, which then converges in one step. When the matrix becomes very ill-conditioned as in Figure 3, then SIR diverges, SGMRES-IR converges very slowly, and thus MSIR makes the second switch to GMRES-IR.

This illustrates the benefit of the multistage approach. In the case that the problem is well-conditioned enough that SIR suffices, MSIR will only use SIR. For the $\kappa_2(A) = 10^1$ case, MSIR behaves exactly the same as SIR, and is thus less expensive than SGMRES-IR and GMRES-IR since a single GMRES iteration is more expensive than an SIR step. In the extremely ill-conditioned case, both SIR and SGMRES-IR fail to converge or convergence is too slow. MSIR however does converge quickly, although at roughly double the cost of GMRES-IR. This is the inherent trade-off. Compared to GMRES-IR, we expect MSIR to be less expensive when the problem is well or reasonably well-conditioned and thus only SIR or SGMRES-IR are used (see a comparison of costs in Table 1). For cases where the problem is extremely ill-conditioned relative to the working precision and GMRES-IR converges, we expect MSIR to be a constant factor more expensive than GMRES-IR. Compared to SIR and SGMRES-IR, the benefit is clear: when SIR and/or SGMRES-IR converges reasonably quickly, MSIR will also converge at roughly the same cost, but MSIR can converge for problems where where SIR and SGMRES-IR may not.

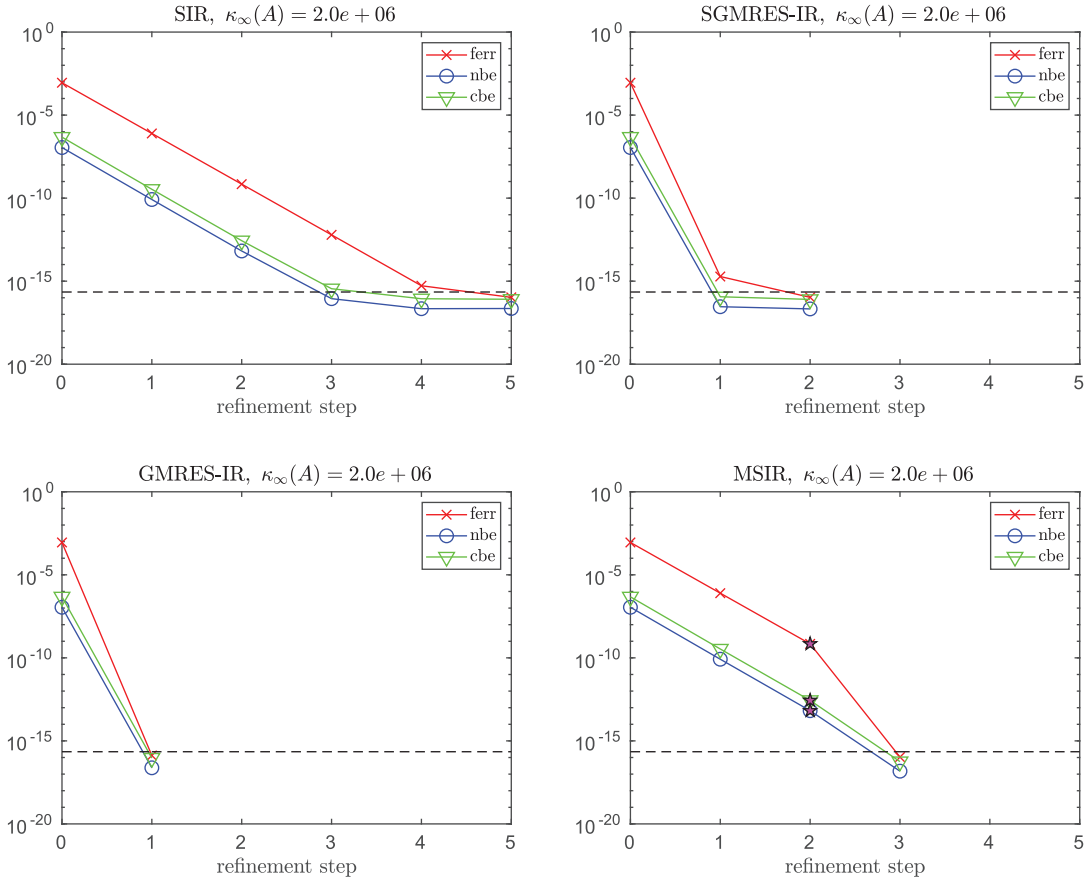

FIGURE 2 Convergence of errors for a 100×100 random dense matrix with $\kappa_2(A) = 10^5$ using standard iterative refinement (SIR) (top left), SGMRES-IR (top right), GMRES-IR (bottom left), and multistage IR (MSIR) (bottom right), with initial precisions (u_f, u, u_r) = (single, double, quad)

2.1 | Algorithm details

We now discuss the MSIR algorithm (Algorithm 4) in more detail. The algorithm uses i to denote the global number of refinement steps of any type. The `iter` variable counts the number of refinement steps of the current refinement variant; for example, it will first count the number of SIR steps, and is then reset to 0 if the algorithm switches to SGMRES-IR. Unlike the single-stage algorithms, here the input parameter $i_{\max}$ specifies the maximum number of refinement steps of each type, meaning, for example, we will perform up to $i_{\max}$ SIR steps before switching to SGMRES-IR.

2.1.1 | Stopping criteria and convergence detection

In Reference 14, Demmel et al. analyze an IR scheme in which extra precision is used in select computations and provide reliable normwise and componentwise error bounds for the computed solution as well as stopping criteria. They devise a new approach that adaptively changes the precision with which the approximate solution is stored based on monitoring the convergence; if consecutive corrections to the approximate solution are not decreasing at a sufficient rate, the precision of the approximate solution is increased. This has the effect of improving the componentwise accuracy.

Following this strategy for monitoring the behavior of IR from Reference 14, the MSIR algorithm will switch to the next variant if any of the following conditions applies:

1. $\frac{\|d_{i+1}\|_\infty}{\|x_i\|_\infty} \leq u$ (the correction d_{i+1} changes solution x_i too little),
2. $\frac{\|d_{i+1}\|_\infty}{\|d_i\|_\infty} \geq \rho_{\text{thresh}}$ (convergence slows down sufficiently),

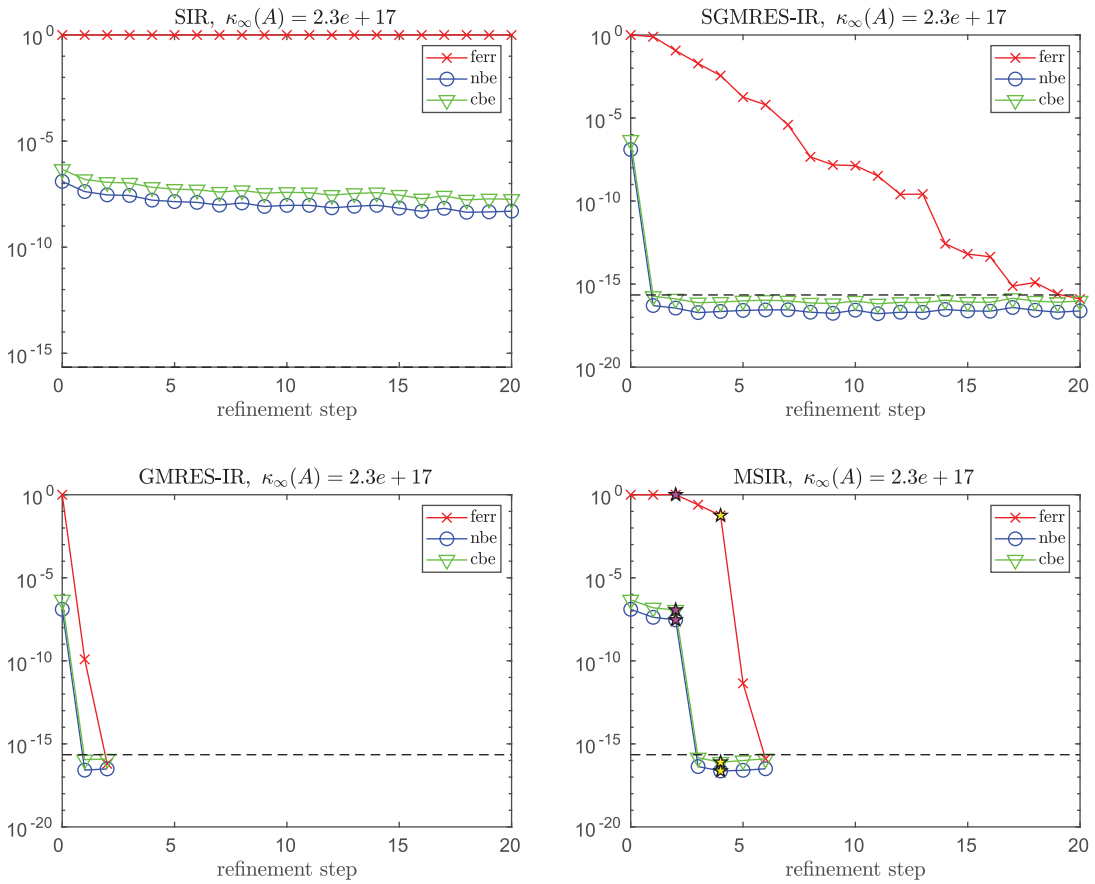

FIGURE 3 Convergence of errors for a 100×100 random dense matrix with $\kappa_2(A) = 10^{16}$ using standard iterative refinement (SIR) (top left), SGMRES-IR (top right), GMRES-IR (bottom left), and multistage (MSIR) (bottom right), with initial precisions (u_f, u, u_r) = (single, double, quad)

TABLE 2 Number of standard iterative refinement (SIR), SGMRES-IR, GMRES-IR, and multistage IR (MSIR) steps with the number of GMRES iterations for each SGMRES-IR and GMRES-IR step for initial precisions (u_f, u, u_r) = (single, double, quad)

$\kappa_\infty(A)$	$\kappa_2(A)$	SIR	SGMRES-IR	GMRES-IR	MSIR
$2 \cdot 10^2$	10^1	2	(2)	(2)	2
$2 \cdot 10^6$	10^5	5	(2,3)	(2)	2, (2)
$2 \cdot 10^{17}$	10^{16}	—	(3,3,3,4,4,4,4,8,100,100,50,12,12,5,4,5,4,4,4,4)	(3,4)	2, (3,3), (3,4)

3. `iter` $\geq i_{\max}$ (too many iterations of a particular variant have been performed), and,
4. $k_{\text{GMRES}} \geq k_{\max}$ (too many GMRES iterations are performed in one step of SGMRES-IR or GMRES-IR)

where d_i is the correction of the solution x_i, $\rho_{\text{thresh}} \in \mathbb{R}^+$ is a threshold for convergence and $\rho_{\text{thresh}} < 1$, $i_{\max} \in \mathbb{N}^+$ is the maximum number of iterations, $k_{\max} \in \mathbb{N}^+$ is the maximum number of GMRES iterations performed per SGMRES-IR step, and u is the working precision.

As is explained in Reference 14, the analyses of Bowdler in 1966[17] and Moler in 1967[7] showed that $\|d_{i+1}\|_\infty$ should decrease by a factor of at most $\rho = O(u_f)\kappa_\infty(A)$ at each step, and thus the solution x_{i+1} should converge like a geometric sum, meaning that $\sum_{j=i+1}^{\infty} \|d_i\|_\infty \leq \|d_{i+1}\|_\infty/(1-\rho)$. This geometric convergence will end when rounding errors become significant. So if we have $\|d_{i+1}\|_\infty/\|d_i\|_\infty \geq \rho_{\text{thresh}}$, it either means that (1) convergence has slowed down due to rounding errors, or (2) convergence is slow from the beginning as the quantity ρ is close to 1 (meaning that the problem is too ill-conditioned with respect to the precision u_f). In the case that $\|d_{i+1}\|_\infty > \|d_i\|_\infty$, this indicates that the IR process is diverging, again because the problem is too ill-conditioned with respect to the precision u_f.

We reiterate that the fourth condition above only applies to the switch from SGMRES-IR to GMRES-IR, or GMRES-IR to SIR with increased precision(s). We note that as in line 9 in Algorithm 4, in case SIR produces a correction d_{i+1} containing Inf or NaN, we immediately switch to SGMRES-IR without performing the solution update. This situation can arise as a result of performing the triangular solves in low precision. We have implemented a simple scaling to help avoid this situation, although in certain scenarios it may still occur; see further details in Section 2.1.2. We note that this is less of a concern for GMRES-based approaches, since within GMRES the triangular solves are performed in either precision u (SGMRES-IR) or u^2 (GMRES-IR).

Moreover, the convergence detection in lines 13, 26, and 39 in Algorithm 4 can be performed using the normwise relative error estimate discussed in Reference 14,

$$\max\left\{\frac{\frac{\|d_{i+1}\|_\infty}{\|x_i\|_\infty}}{1-\rho_{\max}}, \gamma u\right\} \approx \frac{\|x_i - x\|_\infty}{\|x\|_\infty}, \tag{1}$$

where $\gamma = \max(10, \sqrt{n})$, n is the size of the matrix A, and $\rho_{\max} := \max_{j\le i} \frac{\|d_{j+1}\|_\infty}{\|d_j\|_\infty}$ is the maximum ratio of successive corrections. The quantity $(\|d_{i+1}\|_\infty/\|x_i\|_\infty)/(1-\rho_{\max})$ is stored in Algorithm 4 as the quantity φ_i. If the condition in lines 13, 26, and 39 is triggered, one could then test for convergence by checking whether both $\varphi_i \le \sqrt{n}u$ and $\varphi_i \ge 0$, and if so, declare convergence (note that $\varphi_i < 0$ indicates divergence). If convergence is not detected, we must also decide whether to keep the current approximate solution or reset the approximate solution to x_0, since this will be input to the next stage. We test if $\varphi_i > \varphi_0$, and if so, we reset the initial solution. We note that other measures, such as for the componentwise error, could also be used to detect convergence.

We note that the criteria used to determine whether the current algorithm should quit iterating are one iteration behind; in other words, we can only detect convergence (or nonconvergence) in step i after computing the correction term in step $i+1$. For this reason, the MSIR algorithm will generally compute at least two steps of a particular variant before deciding to switch. The two cases where fewer than two steps of a variant will occur are (1) when the computed update contains Infs or NaNs (as in line 9 in Algorithm 4), in which case we switch without updating the current solution, and (2) when, for SGMRES-IR and GMRES-IR, the number of GMRES iterations exceeds $k_{\max}$, since this is detectable immediately in the current step.

In Reference 14, to determine ρ_{thresh}, the authors define "cautious" and "aggressive" settings. Cautious settings produce maximally reliable error bounds for well-conditioned problems, whereas aggressive ones will lead to more steps on the hardest problems and usually, but not always, give error bounds within a factor of 100 of the true error. For cautious mode, the authors suggest ρ_{thresh} should be set to 0.5, which was used by Wilkinson,[6] and for aggressive mode, 0.9. In our experiments we always use the cautious setting, but we note that ρ_{thresh}, $i_{\max}$, and $k_{\max}$ should be set according to the relative costs of the different refinement schemes in practice.

2.1.2 | Scaling

Because of the smaller range of half precision, simply rounding higher precision quantities to a lower precision can cause overflow, underflow, or the introduction of subnormal numbers. In Reference 18, the authors develop a new algorithm for converting single and double precision quantities to half precision. This algorithm involves performing a two-sided scaling for equilibration and then an additional scaling is performed to make full use of the range of half precision before finally rounding to half precision. In our algorithm, when half precision is used for the factorization, we first attempt an LU factorization without scaling. We then test whether the resulting L and U factors contain Inf or NaN; if so (marked with a * in the tables in Section 3), we retry the LU factorization in line 1 using the two-sided scaling algorithm of Reference 18. In all cases, regardless of what precision is used for the factorization, after computing x_0 in line 2 we test whether the initial solution contains Inf or NaN; if so, we simply use the zero vector as the initial approximate solution and proceed.

We also incorporate scaling in each refinement step. After computing the residual r_i, we scale the result to obtain $r_i = r_i/\|r_i\|_\infty$ (line 5 in Algorithm 4). This scaling is then undone when we update the approximate solution, via $x_{i+1} = x_i + \|r_i\|_\infty d_{i+1}$ (lines 10, 23, and 36 in Algorithm 4). As long as $1/\|A\|_\infty$ does not underflow and $\|A^{-1}\|_\infty$ does not overflow, then this scaling avoids the largest element of d_{i+1} overflowing or underflowing.

TABLE 3 Various IEEE precisions and their units roundoff

Precision	Unit roundoff
fp16 (half)	$4.88 \cdot 10^{-4}$
fp32 (single)	$5.96 \cdot 10^{-8}$
fp64 (double)	$1.11 \cdot 10^{-16}$
fp128 (quad)	$9.63 \cdot 10^{-35}$

TABLE 4 Constraints on $\kappa_\infty(A)$ for which the relative forward and normwise backward errors are guarantee to converge to the level u for a given combination of precisions for the different variants of iterative refinement (IR)

u_f	u	u_r	Standard iterative refinement	SGMRES-IR	GMRES-IR
Half	Single	Double	$2 \cdot 10^3$	$4 \cdot 10^4$	$8 \cdot 10^6$
Single	Single	Double	$2 \cdot 10^7$	$2 \cdot 10^7$	$7 \cdot 10^{10}$
Half	Double	Quad	$2 \cdot 10^3$	$3 \cdot 10^7$	$2 \cdot 10^{11}$
Single	Double	Quad	$2 \cdot 10^7$	$1 \cdot 10^{10}$	$2 \cdot 10^{15}$
Double	Double	Quad	$9 \cdot 10^{15}$	$9 \cdot 10^{15}$	$9 \cdot 10^{23}$

2.2 | Error bounds for different variants

There are various scenarios on the usage of precisions which will yield different error bounds and different constraints on condition number. Besides three-precision variants, two precisions or a fixed precision can be used in the algorithms discussed in this study. We summarize the convergence criteria for the precision combinations used in our approach and refer the reader to References 4 and 12 for more general bounds. In particular, we assume that $u_f \geq u$ and $u_r \leq u^2$. We also restrict ourselves to IEEE precisions (see Table 3), although we note that alternative formats like bfloat16[19] could also be used.

Under the assumptions that $u_f \geq u$ and $u_r \leq u^2$, for SIR, both the relative forward and backward errors are guaranteed to converge to the level of the working precision when $\kappa_\infty(A) < u_f^{-1}$. For SGMRES-IR and GMRES-IR, the constraints for convergence of forward and backward errors differ. Summarizing the analysis in Reference 12, for our particular SGMRES-IR variant, the constraint on convergence of the backward error is $\kappa_\infty(A) < u^{-1/3}u_f^{-1/3}$ and the constraint on the convergence of the forward error is $\kappa_\infty(A) < u^{-1/3}u_f^{-2/3}$. It is interesting to note that, as an artifact of the analysis, the constraint for convergence of the backward error for this variant is more strict than that for the forward error. However, since the backward error is bounded by the forward error, the constraint for both backward and forward error to converge to the level u can be given as $\kappa_\infty(A) < u^{-1/3}u_f^{-2/3}$.

In Reference 4, the constraint for convergence of the backward error to level u in GMRES-IR given as is $\kappa_\infty(A) < u^{-1}$. However, the authors in Reference 12 point out that this bound relies on assumptions that may be overly optimistic, and revise this to the tighter constraint $\kappa_\infty(A) < u^{-1/2}u_f^{-1/2}$. The constraint for the convergence of the forward error to this level in GMRES-IR given in Reference 4 is $\kappa_\infty(A) < u^{-1/2}u_f^{-1}$. Thus the tighter constraint for the convergence of backward error in Reference 12 is again more strict than the constraint for the convergence of the forward error, and since the backward error is bounded by the forward error, we can take $\kappa_\infty(A) < u^{-1/2}u_f^{-1}$ to be the constraint for the convergence of both backward and forward error to level u in GMRES-IR.

In Table 4 we quantify these constraints on $\kappa_\infty(A)$ required for convergence of the normwise relative backward and forward errors to the level of the working precision for various precision combinations. To compute the constraints on $\kappa_\infty(A)$, we have used the unit roundoff values in Table 3.

3 | NUMERICAL EXPERIMENTS

In this section we present numerical experiments performed in MATLAB on a number of synthetic and real-world matrices from SuiteSparse[16] for MSIR and single-stage IR variants in three precisions. The problems we test are small and are

meant to demonstrate the numerical behavior of the algorithms. Performance results for SIR and GMRES-based variants for larger problems on modern GPUs can be found in, for example, Reference 11.

For quadruple precision, we use the Advanpix multiprecision computing toolbox for MATLAB.[20] To simulate half precision floating point arithmetic, we use the `chop` library and associated functions from,[21] available at https://github.com/higham/chop and https://github.com/SrikaraPranesh/LowPrecision_Simulation. For single and double precision we use the built-in MATLAB datatypes. All experiments were performed on a computer with an Intel Core i7-9750H processor with 12 CPUs and 16 GB RAM with OS system Ubuntu 20.04.1 using MATLAB 2020a. Our MSIR algorithm and associated functions are available through the repository https://github.com/edoktay/msir, which includes scripts for generating the data and plots in this work.

In all experiments, we use $i_{\max} = 2000$ and $\rho_{\text{thresh}} = 0.5$. We have chosen to use a very high value for $i_{\max}$ in both MSIR and single-stage algorithms as it allows us to see the true convergence behavior of each algorithm (and the benefits of MSIR in cases where convergence for single-stage algorithms happens eventually but is very slow). Of course, in practice, one would use a much smaller value.

For the GMRES convergence tolerance, we use $\tau = 10^{-6}$ when the working precision is single and $\tau = 10^{-10}$ when the working precision is double. We set $k_{\max} = 0.1n$ in lines 25 and 38 of MSIR. For purposes of discerning the actual attainable accuracy of MSIR, in the experiments we explicitly compute and plot the computed forward and backward errors and use these as stopping criteria rather than the error estimate given by (1). For a fair numerical comparison, we also apply the scalings discussed in Section 2.1.2 in SIR, SGMRES-IR, and GMRES-IR.

For each variant, we compare the number of refinement steps required for forward and backward errors to reach the level of accuracy corresponding to the initial working precision. For GMRES-based approaches and MSIR, the number of GMRES iterations per step is given parenthetically (see the explanation in Section 2). For the MSIR results, a semicolon indicates a precision switch: the factorization precision is doubled (and other precisions increased if necessary to ensure $u_f \geq u$ and $u_r \leq u^2$) and the algorithm restarts with SIR. A dash (-) in the tables indicates that forward and/or backward errors are not decreasing. Select convergence plots for MSIR are presented in figures while the number of steps and iterations for all variants are shown in tables.

Again, the red, blue, and green lines in the figures show the behavior of the forward error `ferr` (red), normwise relative backward error `nbe` (blue), and componentwise relative backward error `cbe` (green). The dotted black line shows the value of the initial working precision u. Switches in MSIR are denoted by stars. A magenta star indicates a switch from SIR to SGMRES-IR, a yellow start indicates a switch from SGMRES-IR to GMRES-IR, and a cyan star indicates switch from GMRES-IR to SIR with increased precision(s).

3.1 | Random dense matrices

We first test our algorithm on random dense matrices of dimension $n = 100$ generated in MATLAB via the command `gallery('randsvd',n,kappa(i),2)`(Section 3.1.1) and `gallery('randsvd',n,kappa(i),3)` (Section 3.1.2), where `kappa` is the array of the desired 2-norm condition numbers $\kappa_2(A) = \{10^1, 10^2, 10^4, 10^5, 10^7, 10^9, 10^{11}, 10^{14}\}$, and `2` and `3` stand for the modes which dictate the singular value distribution. Mode 2 generates matrices with one singular value equal to $1/\kappa_2(A)$ and the rest of the singular values equal to 1. Mode 3 generates matrices with geometrically distributed singular values. Mode 3 represents a difficult case for GMRES-based IR methods. It is known that a cluster of singular values near zero causes stagnation of GMRES, and a low-precision preconditioner may fail to effectively shift this cluster away from the origin. We test the algorithms using initial precision combinations (u_f, u, u_r) = (single, double, quad), (u_f, u, u_r) = (half, single, double), and (u_f, u, u_r) = (half, double, quad). For simplicity, b is chosen to be a vector of normally distributed numbers generated by the MATLAB command `randn` for all experiments in this section. For reproducibility, we use the MATLAB command `rng(1)` before generating each linear system.

3.1.1 | Random dense matrices with one small singular value

Table 5 shows the convergence behavior of SIR, SGMRES-IR, GMRES-IR, and MSIR for matrices generated via the MATLAB command `gallery('randsvd',n,kappa(i),2)` using initial precisions (u_f, u, u_r) = (single, double, quad). From Table 5, we can see that SIR no longer converges once $\kappa_\infty(A)$ exceeds 10^{11}. As $\kappa(A)$ increases, the convergence rate of SIR slows down. At the extreme case of $\kappa_2(A) = 10^9$, SIR requires 205 steps to converge! In this case, MSIR thus

TABLE 5 Number of standard iterative refinement (SIR), SGMRES-IR, GMRES-IR, and multistage IR (MSIR) steps with the number of GMRES iterations for each SGMRES-IR and GMRES-IR step for random dense matrices (mode 2) with various condition numbers $\kappa_\infty(A)$, $\kappa_2(A)$, using initial precisions (u_f, u, u_r) = (single, double, quad)

$\kappa_\infty(A)$	$\kappa_2(A)$	SIR	SGMRES-IR	GMRES-IR	MSIR
$2 \cdot 10^2$	10^1	2	(2)	(2)	2
$2 \cdot 10^3$	10^2	3	(2)	(2)	2, (2)
$2 \cdot 10^5$	10^4	5	(2,3)	(2)	2, (2)
$2 \cdot 10^6$	10^5	5	(2,3)	(2)	2, (2)
$2 \cdot 10^8$	10^7	19	(2,3)	(2,3)	2, (2,3)
$2 \cdot 10^{10}$	10^9	205	(2,2)	(2,3)	2, (2,3)
$2 \cdot 10^{12}$	10^{11}	—	(3,3,4)	(3,4)	2, (3,3), (3)
$2 \cdot 10^{15}$	10^{14}	—	(3,3,3,4,4,4,4)	(3,4)	2, (3,3), (3,4)

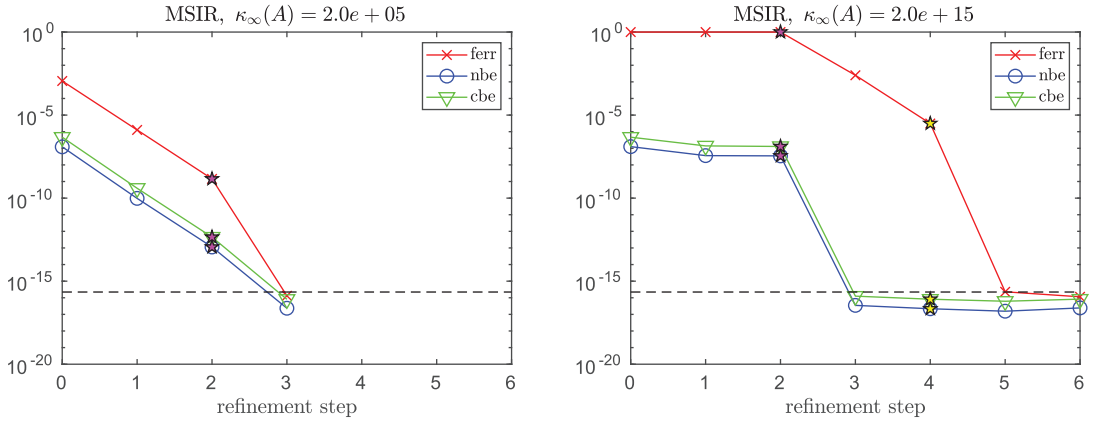

FIGURE 4 Convergence of errors in multistage iterative refinement for random dense matrices (mode 2) with $\kappa_2(A) = 10^4$ (left), and $\kappa_2(A) = 10^{14}$ (right) for initial precisions (u_f, u, u_r) = (single, double, quad); see also Table 5

demonstrates a significant improvement over SIR. Even in cases where only five SIR steps are required, notice that the MSIR algorithm deems this too slow and begins switching to SGMRES-IR after two SIR steps. SGMRES-IR converges in just a few refinement steps with a small numbers of GMRES iterations per step except for the most ill-conditioned problem. For the most ill-conditioned matrix, SGMRES-IR alone requires seven steps to converge. MSIR in this case switches twice: from SIR to SGMRES-IR, and then finally to GMRES-IR. The same is observed when $\kappa_2(A) = 10^{11}$. We plot the convergence trajectories of MSIR for the $\kappa_2(A) = 10^4$ and $\kappa_2(A) = 10^{14}$ problems in Figure 4.

In Table 6, we show data for experiments with (u_f, u, u_r) = (half, single, double). In this case, as predicted in Table 4, SIR fails to converge once $\kappa_\infty(A) > 2 \cdot 10^5$. Moreover, it is seen that MSIR switches to SGMRES-IR even for relatively well-conditioned matrices for which SIR still converges (i.e., for $\kappa_2(A) = 10^2$). Although this switch is technically not necessary, we argue that the difference in cost versus SIR will be a constant factor (SIR requires three LU solves in precision u_f whereas here SGMRES-IR requires two LU solves in precision u_f and three in precision u; see Table 1). SGMRES-IR works well up to condition number $\kappa_\infty(A) = 2 \cdot 10^6$, and thus MSIR only performs one switch in these cases.

It is interesting to notice that the theory from[12] says that we can only expect SGMRES-IR to converge as long as $\kappa_\infty(A) < 4 \cdot 10^4$; see also Table 4. However, as this and other experiments in this work show, SGMRES-IR as well as GMRES-IR actually often perform beyond the limits given by the analysis. This confirms our motivation for the MSIR approach; we cannot rely entirely on the existing theoretical bounds to determine whether or not an algorithm will be effective.

For the condition number $\kappa_\infty(A) = 2 \cdot 10^8$, SGMRES-IR convergence eventually slows down, and as a result MSIR switches a second time to GMRES-IR. Beyond this limit, for $\kappa_\infty(A) = 2 \cdot 10^8$, GMRES-IR still converges (despite that the

TABLE 6 Number of standard iterative refinement (SIR), SGMRES-IR, GMRES-IR, and multistage (MSIR) steps with the number of GMRES iterations for each SGMRES-IR and GMRES-IR step for random dense matrices (mode 2) with various condition numbers $\kappa_\infty(A)$, $\kappa_2(A)$, using initial precisions (u_f, u, u_r) = (half, single, double)

$\kappa_\infty(A)$	$\kappa_2(A)$	SIR	SGMRES-IR	GMRES-IR	MSIR
$2 \cdot 10^2$	10^1	3	(3)	(3)	3
$2 \cdot 10^3$	10^2	3	(3)	(3)	2, (3)
$2 \cdot 10^5$	10^4	19	(3,4)	(3,4)	2, (3,4)
$2 \cdot 10^6$	10^5	—	(3,4)	(3,4)	2, (3,4)
$2 \cdot 10^8$	10^7	—	(5,12,9,17,5,5,5)	(5,5)	2, (5,10), (10)
$2 \cdot 10^{10}$	10^9	—	—	(100,100,100,22,11,3)	2, (5,9), (10); 2, (2,2,2), (2,3,3)
$2 \cdot 10^{12}$	10^{11}	—	—	(100,63,10)	2, (10), (10); 2, (2,2), (2)
$2 \cdot 10^{15}$	10^{14}	—	—	(100,63,10)	2, (10), (10); 2, (2,2), (2)

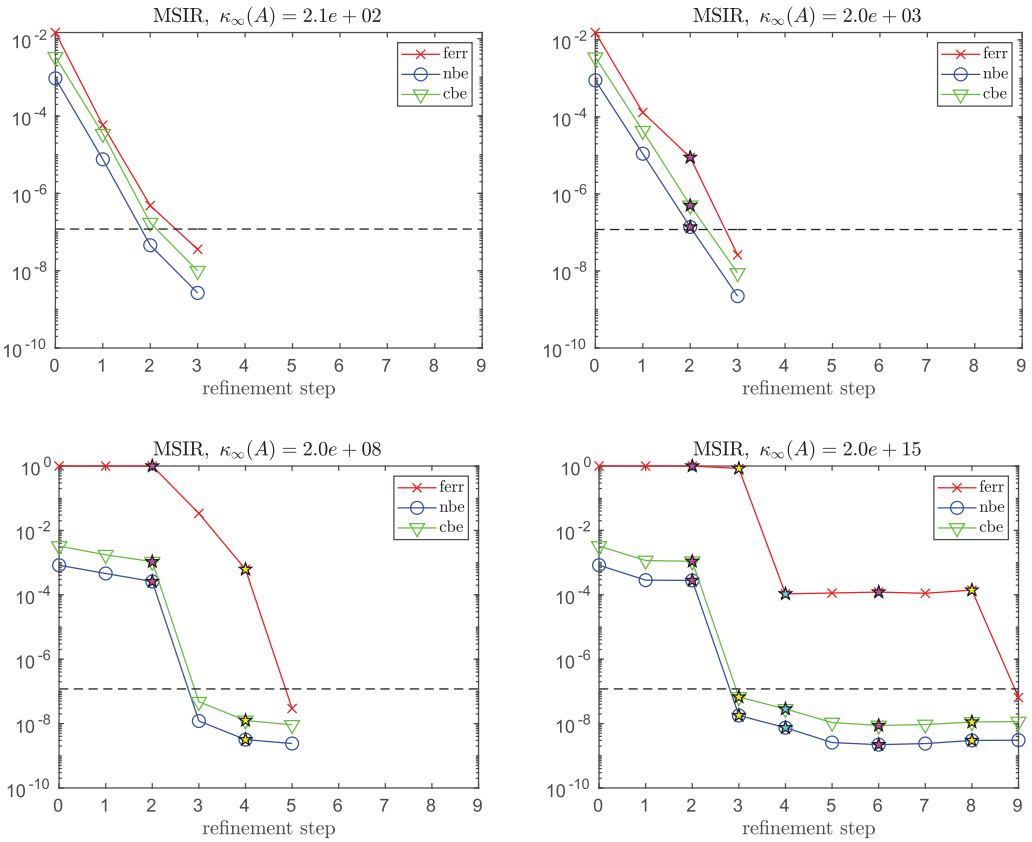

FIGURE 5 Convergence of errors in multistage iterative refinement for random dense matrices (mode 2) with $\kappa_2(A) = 10^1$ (top left), $\kappa_2(A) = 10^2$ (top right), $\kappa_2(A) = 10^7$ (bottom left), and $\kappa_2(A) = 10^{14}$ (bottom right) for initial precisions (u_f, u, u_r) = (half, single, double); see also Table 6

theory only guarantees that GMRES-IR will work up to condition number $8 \cdot 10^6$). Starting at $\kappa_\infty(A) = 2 \cdot 10^{10}$, GMRES convergence slows down significantly (requiring n iterations per refinement step), and thus MSIR switches to precisions (u_f, u, u_r) = (single, single, double) and starts again with SIR. We show plots of MSIR convergence behavior for the cases $\kappa_2(A) \in \{10^1, 10^2, 10^7, 10^{14}\}$ in Figure 5.

Finally, in Table 7, we test initial precisions (u_f, u, u_r) = (half, double, quad). Except for SIR, which again fails to converge once $\kappa_\infty(A) > 2 \cdot 10^5$, all algorithms converge for $\kappa_\infty(A) < 10^{16}$. Even in cases where SIR converges, the SIR convergence is slow enough that MSIR always switches to SGMRES-IR. MSIR only performs the second switch to GMRES-IR

TABLE 7 Number of standard iterative refinement (SIR), SGMRES-IR, GMRES-IR, and multistage iterative refinement (MSIR) steps with the number of GMRES iterations for each SGMRES-IR and GMRES-IR step for random dense matrices (mode 2) with various condition numbers $\kappa_\infty(A)$, $\kappa_2(A)$, using initial precisions (u_f, u, u_r) = (half, double, quad)

$\kappa_\infty(A)$	$\kappa_2(A)$	SIR	SGMRES-IR	GMRES-IR	MSIR
$2 \cdot 10^2$	10^1	7	(5,5)	(5,5)	2, (5)
$2 \cdot 10^3$	10^2	9	(5,5)	(5,5)	2, (5)
$2 \cdot 10^5$	10^4	47	(5,6)	(5,6)	2, (5,6)
$2 \cdot 10^6$	10^5	—	(5,6)	(5,6)	2, (5,6)
$2 \cdot 10^8$	10^7	—	(6,7)	(6,7)	2, (6,7)
$2 \cdot 10^{10}$	10^9	—	(7,8)	(7,8)	2, (7,8)
$2 \cdot 10^{12}$	10^{11}	—	(8,8,9)	(8,9)	2, (8,8), (9)
$2 \cdot 10^{15}$	10^{14}	—	(100,100,100,25,10,10)	(100,32,10)	2, (10), (10); 3, (4,4,4)

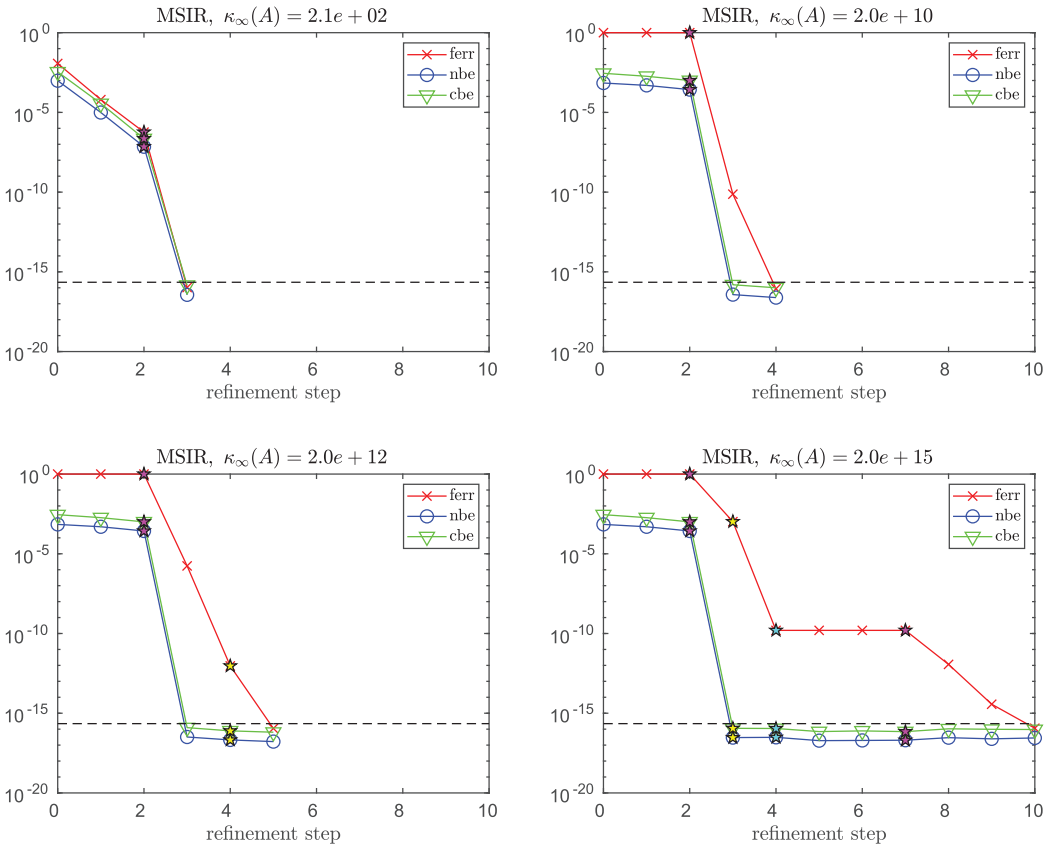

FIGURE 6 Convergence of errors in multistage iterative refinement for random dense matrices (mode 2) with $\kappa_2(A) = 10^1$ (top left), $\kappa_2(A) = 10^9$ (top right), $\kappa_2(A) = 10^{11}$ (bottom left), and $\kappa_2(A) = 10^{14}$ (bottom right) for initial precisions (u_f, u, u_r) = (half, double, quad); see also Table 7

for the two most ill-conditioned problems (although this switch is actually unnecessary at least for $\kappa_2(A) = 10^{11}$; see also the convergence trajectory in the bottom left plot in Figure 6). Again, this emphasizes the need for a multistage, adaptive approach; if we were to go by the theoretical bounds, we would not expect SGMRES-IR to work beyond $\kappa_\infty(A) = 3 \cdot 10^7$, but here it works well even for a matrix with a condition number five orders of magnitude greater. For the most ill-conditioned case, MSIR also performed a precision switch from (u_f, u, u_r) = (half, double, quad) to (u_f, u, u_r) = (single, double, quad) since GMRES-IR required too many GMRES iterations. We plot convergence trajectories for MSIR for the problems with $\kappa_2(A) \in \{10^1, 10^9, 10^{11}, 10^{14}\}$ in Figure 6.

TABLE 8 Number of standard iterative refinement (SIR), SGMRES-IR, GMRES-IR, and multistage iterative refinement (MSIR) steps with the number of GMRES iterations for each SGMRES-IR and GMRES-IR step for random dense matrices having geometrically distributed singular values (mode 3) with various condition numbers $\kappa_\infty(A)$, $\kappa_2(A)$, using initial precisions (u_f, u, u_r) = (single, double, quad)

$\kappa_\infty(A)$	$\kappa_2(A)$	SIR	SGMRES-IR	GMRES-IR	MSIR
$2 \cdot 10^2$	10^1	2	(2)	(2)	2
10^3	10^2	2	(2)	(2)	2
$9 \cdot 10^4$	10^4	3	(3)	(3)	2, (3)
$8 \cdot 10^5$	10^5	4	(4)	(4)	2, (4)
$7 \cdot 10^7$	10^7	13	(7,7)	(7,7)	2, (6,7)
$6 \cdot 10^9$	10^9	—	(22,23,25)	(22,26)	2, (10), (10); 2
$6 \cdot 10^{11}$	10^{11}	—	(53,54,55)	(53,55)	2, (10), (10); 2, (2)
$5 \cdot 10^{14}$	10^{14}	—	(84,84,84,85,85)	(84,88)	2, (10), (10); 2, (3,3,3)

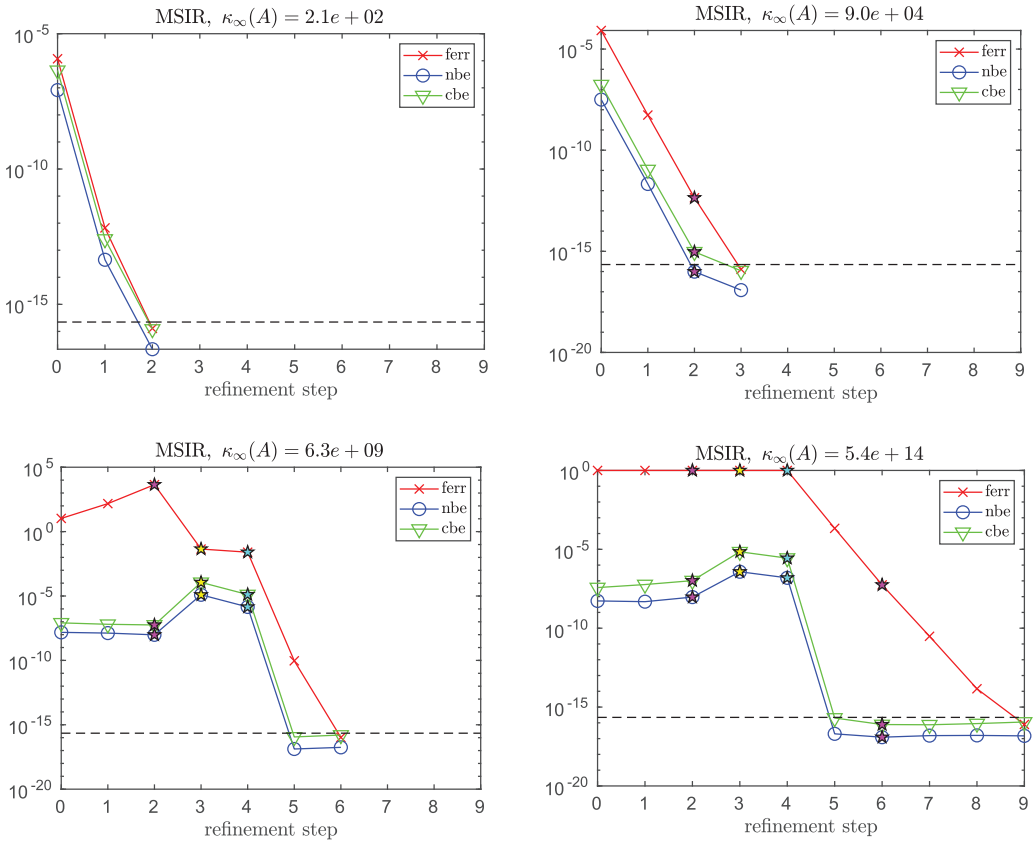

FIGURE 7 Convergence of errors in multistage iterative refinement for random dense matrices having geometrically distributed singular values (mode 3) with $\kappa_2(A) = 10^1$ (top left), $\kappa_2(A) = 10^4$ (top right), $\kappa_2(A) = 10^9$ (bottom left), and $\kappa_2(A) = 10^{14}$ (bottom right) for initial precisions (u_f, u, u_r) = (single, double, quad); see also Table 8

3.1.2 | Random dense matrices with geometrically distributed singular values

We now test the convergence behavior of the IR variants for matrices generated via the MATLAB command `gallery('randsvd',n,kappa(i),3)`. As mentioned, we expect that these are more challenging problems for the GMRES-based IR schemes since they contain clusters of eigenvalues close to the origin which may fail to be effectively shifted away from the origin by a low precision preconditioner.

TABLE 9 Number of standard iterative refinement (SIR), SGMRES-IR, GMRES-IR, and multistage iterative refinement (MSIR) steps with the number of GMRES iterations for each SGMRES-IR and GMRES-IR step for random dense matrices having geometrically distributed singular values (mode 3) with various condition numbers $\kappa_\infty(A)$, $\kappa_2(A)$, using initial precisions (u_f, u, u_r) = (half, single, double). For $\kappa_2(A) = 10^{11}$, GMRES-IR used (100,100,100,100,100,100,100,100,100,100,99,95,100,97) iterations

$\kappa_\infty(A)$	$\kappa_2(A)$	SIR	SGMRES-IR	GMRES-IR	MSIR
$2 \cdot 10^2$	10^1	3	(3)	(3)	2, (3)
10^3	10^2	4	(4)	(4)	2, (4)
$9 \cdot 10^4$	10^4	856	(13,14)	(13,14)	2, (10), (10)
$8 \cdot 10^5$	10^5	—	(38,40)	(38,41)	0, (10), (10); 3
$7 \cdot 10^7$	10^7	—	(100,100,100,100,100,83)	(100,100)	0, (10), (10); 2, (5,5,5)
$6 \cdot 10^9$	10^9	—	—	(100,100,100,100)	0, (10), (10);1, (10), (10); 1
$6 \cdot 10^{11}$	10^{11}	—	—	(100, …) see caption.	0, (10), (10); 1, (10), (10); 1
$5 \cdot 10^{14}$	10^{14}	—	—	—	0, (10), (10); 1, (10), (10); 2

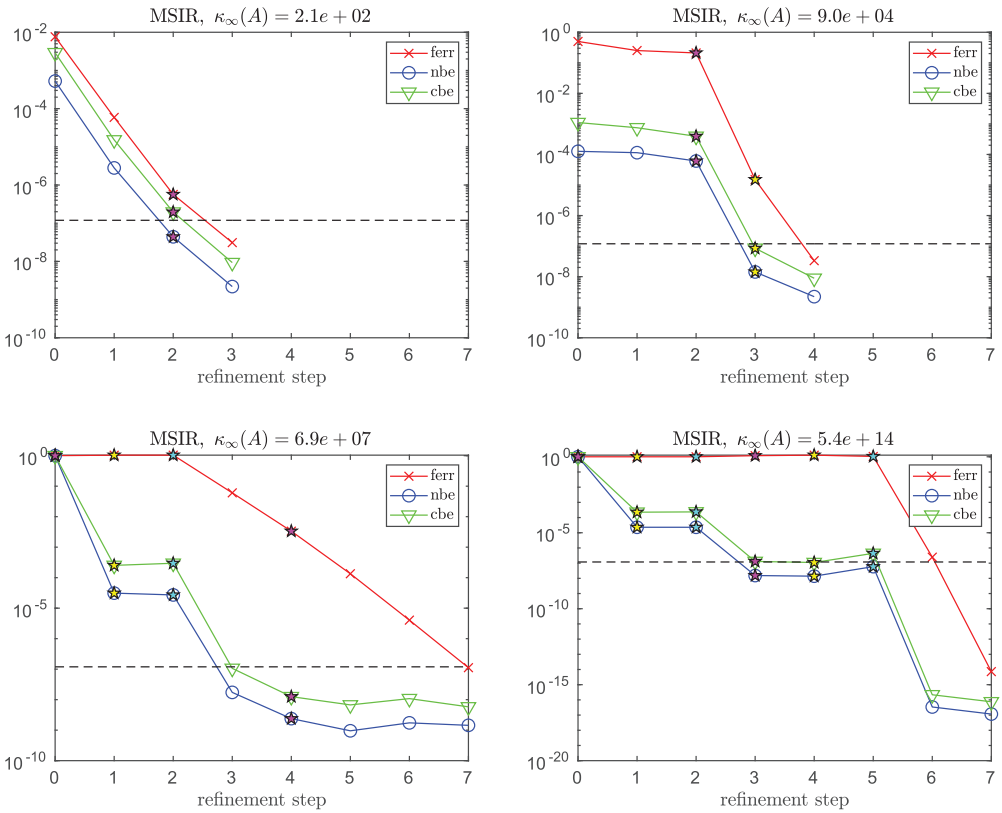

FIGURE 8 Convergence of errors in multistage iterative refinement for random dense matrices having geometrically distributed singular values (mode 3) with $\kappa_2(A) = 10^1$ (top left), $\kappa_2(A) = 10^4$ (top right), $\kappa_2(A) = 10^7$ (bottom left), and $\kappa_2(A) = 10^{14}$ (bottom right) for initial precisions (u_f, u, u_r) = (half, single, double); see also Table 9

In Table 8 we test initial precisions (u_f, u, u_r) = (single, double, quad). Here, SIR fails to converge once $\kappa_\infty(A)$ exceeds 10^9. Once $\kappa_\infty(A)$ exceeds 10^9, the number of GMRES iterations required per SGMRES-IR and GMRES-IR step begins to increase dramatically. This is expected since there is a cluster of eigenvalues remaining close to the origin even after low precision preconditioning. Notice that in these cases, MSIR performs the second switch to GMRES-IR after just one SGMRES-IR step and then performs a precision switch from (u_f, u, u_r) = (single, double, quad) to (u_f, u, u_r) = (double, double, quad) after just one GMRES-IR step, since the number of GMRES iterations per refinement step for both SGMRES-IR and GMRES-IR exceeds our specified value of $k_{\max} = 0.1n$. We show plots for $\kappa_2(A) \in \{10^1, 10^4, 10^9, 10^{14}\}$ in Figure 7.

TABLE 10 Number of standard iterative refinement (SIR), SGMRES-IR, GMRES-IR, and multistage iterative refinement (MSIR) steps with the number of GMRES iterations for each SGMRES-IR and GMRES-IR step for random dense matrices having geometrically distributed singular values (mode 3) with various condition numbers $\kappa_\infty(A)$, $\kappa_2(A)$, using initial precisions (u_f, u, u_r) = (half, double, quad). For $\kappa_2(A) = 10^{14}$, SGMRES-IR used (100,100,100,100,100,85,96,93,90,84,94,92,90,81,95,84,89,82,90,96) iterations

$\kappa_\infty(A)$	$\kappa_2(A)$	SIR	SGMRES-IR	GMRES-IR	MSIR
$2 \cdot 10^2$	10^1	7	(5,5)	(5,5)	2, (5)
10^3	10^2	9	(6,6)	(6,6)	2, (6,6)
$9 \cdot 10^4$	10^4	—	(19,20)	(19,20)	2, (10), (10); 2
$8 \cdot 10^5$	10^5	—	(43,46)	(43,46)	0, (10), (10); 3, (3)
$7 \cdot 10^7$	10^7	—	(83,85)	(83,85)	0, (10), (10); 2, (6,7)
$6 \cdot 10^9$	10^9	—	(100,100)	(100,100)	0, (10), (10); 2, (10), (10); 2
$6 \cdot 10^{11}$	10^{11}	—	(100,100,100)	(100,100)	0, (10), (10); 3, (10), (10); 2, (2)
$5 \cdot 10^{14}$	10^{14}	—	(100, …) see caption.	(100,100,100,91,84,88,83,96,85,100)	0, (10), (10); 2, (10), (10); 2, (3,3,3)

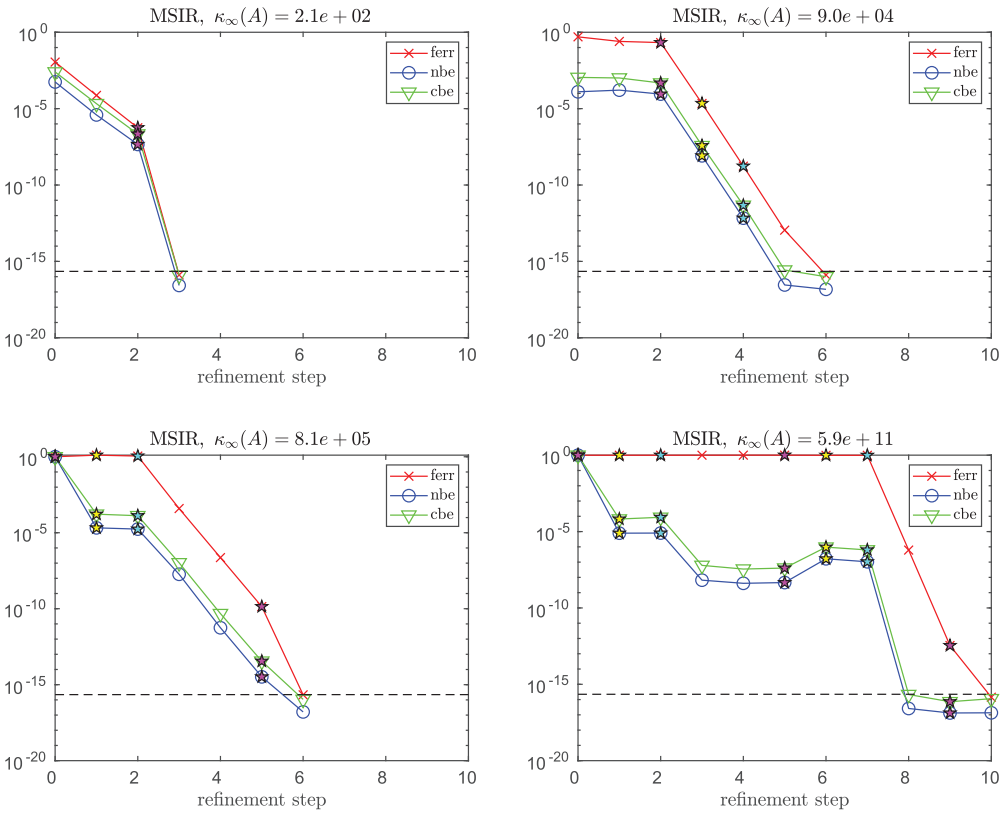

FIGURE 9 Convergence of errors in multistage iterative refinement (MSIR) for random dense matrices having geometrically distributed singular values (mode 3) with $\kappa_2(A) = 10^1$ (top left), $\kappa_2(A) = 10^4$ (top right), $\kappa_2(A) = 10^5$ (bottom left), and $\kappa_2(A) = 10^{11}$ (bottom right) for initial precisions (u_f, u, u_r) = (half, double, quad)

The story for initial precisions (u_f, u, u_r) = (half, single, double), shown in Table 9, is similar. Once the condition number exceeds around $\sqrt{u}$, the number of GMRES iterations required per refinement step increases significantly for both SGMRES-IR and GMRES-IR. Again, this means that MSIR switches from SGMRES-IR to GMRES-IR and then performs a precision switch (to $[u_f, u, u_r]$ = [single, single, double]) since $k_{\max}$ is exceeded. When κ_∞ exceeds 10^9, however, using single precision for the factorization is not enough to combat the slow GMRES convergence, and thus a second precision switch occurs, switching to (u_f, u, u_r) = (double, double, quad). For the most ill-conditioned problem for which SGMRES-IR converges, MSIR does significantly fewer GMRES iterations in total than SGMRES-IR (although we use

TABLE 11 Matrices from Reference 16 used for numerical experiments and their properties

Name	Size	Nnz	κ_∞	Group	Kind
cage6	93	785	2.34E+01	vanHeukelum	Directed Weighted Graph
tols90	90	1746	3.14E+04	Bai	Computational Fluid Dynamics Problem
bfwa62	62	450	1.54E+03	Bai	Electromagnetics Problem
cage5	37	233	2.91E+01	vanHeukelum	Directed Weighted Graph
d_dyn	87	230	8.71E+06	Grund	Chemical Process Simulation Problem
d_ss	53	144	6.02E+08	Grund	Chemical Process Simulation Problem
Hamrle1	32	98	5.51E+05	Hamrle	Circuit Simulation Problem
ww_36_pmec_36	66	1194	4.283E+11	Rommes	Eigenvalue/Model Reduction Problem
steam3	80	314	7.64E+10	HB	Computational Fluid Dynamics Problem

TABLE 12 Number of standard iterative refinement (SIR), SGMRES-IR, GMRES-IR, and multistage (MSIR) steps with the number of GMRES iterations for each SGMRES-IR and GMRES-IR step for real matrices with initial precisions (u_f, u, u_r) = (single, double, quad)

Matrix	SIR	SGMRES-IR	GMRES-IR	MSIR
cage6	2	(2)	(2)	2
tols90	2	(2)	(2)	2
bfwa62	2	(2)	(2)	2
cage5	2	(2)	(2)	2
Hamrle1	2	(2)	(2)	2
d_ss	2	(2)	(2)	2
d_dyn	2	(2)	(2)	2
ww_36_pmec_36	-	(2,3,3)	(2,3,3)	2, (2,3,3)
steam3	2	(2)	(2)	2

higher precision for u_f (single), the computational cost will still less considering the total number of GMRES steps performed in SGMRES-IR.). Notice also that for $\kappa_2(A) \geq 10^5$, MSIR performs no SIR steps at the initial precision setting (u_f, u, u_r) = (half, single, double) since an Inf or NaN is detected in the first correction term. We plot MSIR convergence for $\kappa_2(A) \in \{10^1, 10^4, 10^7, 10^{14}\}$ in Figure 8.

In Table 10 we show results for initial precisions (u_f, u, u_r) = (half, double, quad). Here the increase in the number of GMRES iterations per GMRES-based refinement step as $\kappa(A)$ increases is also dramatic. One interesting thing to notice here and even more so in the previous examples with mode 3 matrices is that, in cases where both SGMRES-IR and GMRES-IR converge, the extra precision used in GMRES-IR does not help improve the convergence rate of GMRES (nearly the same number of GMRES iterations per refinement step are required). The aspect that improves is the number of refinement steps. This is likely related to the GMRES convergence trajectories for this class of problems; the residual will nearly stagnate (or at least not make much progress) for a number of iterations and then convergence will happen suddenly. The difference is likely that when this convergence does happen, the extra precision in GMRES-IR results in the approximate solution being found to greater accuracy than in the uniform precision iterations in SGMRES-IR.

As a result of this slow convergence of GMRES, MSIR does a precision switch from (u_f, u, u_r) = (half, double, quad) to (u_f, u, u_r) = (single, double, quad) for matrices with condition number $\kappa_2(A) \geq 10^4$. For matrices with condition number $\kappa_2(A) \geq 10^9$, MSIR does a precision switch a second time, from (u_f, u, u_r) = (single, double, quad) to (u_f, u, u_r) = (double, double, quad). Even though this means that MSIR computes three LU factorizations, one in half, one in single, and one in double, this is still likely to be much faster than SGMRES-IR, which does a total of 1841 $O(n^2)$ triangular solves in double precision, or GMRES-IR, which does a total of 927 $O(n^2)$ triangular solves in quadruple precision. We show MSIR convergence for $\kappa_2(A) \in \{10^1, 10^4, 10^5, 10^{11}\}$ in Figure 9.

TABLE 13 Number of standard iterative refinement (SIR), SGMRES-IR, GMRES-IR, and multistage iterative refinement (MSIR) steps with the number of GMRES iterations for each SGMRES-IR and GMRES-IR step for real matrices with initial precisions (u_f, u, u_r) = (half, single, double). Cases where scaling was required prior to LU factorization are marked with a *

Matrix	SIR	SGMRES-IR	GMRES-IR	MSIR
cage6	2	(3)	(3)	2
tols90	2	(2)	(2)	2
bfwa62	4	(3)	(3)	2, (3)
cage5	2	(3)	(3)	2
Hamrle1	2	(2)	(2)	2
d_ss	—	(3,3)	(3,3)	0, (3,3)
d_dyn	—	(2,2)	(2,2)	0, (2,2)
ww_36_pmec_36	—	—	(66,66,7)	0, (7), (7); 2, (2,2,3,3), (2)
steam3*	—	(2)	(2)	2, (2,2), (2)

TABLE 14 Number of standard iterative refinement (SIR), SGMRES-IR, GMRES-IR, and multistage iterative refinement (MSIR) steps with the number of GMRES iterations for each SGMRES-IR and GMRES-IR step for real matrices with initial precisions (u_f, u, u_r) = (half, double, quad). Cases where scaling was required prior to LU factorization are marked with a *

Matrix	SIR	SGMRES-IR	GMRES-IR	MSIR
cage6	5	(4,4)	(4,4)	2, (4)
tols90	5	(3,3)	(3,3)	2, (3)
bfwa62	9	(4,5)	(4,5)	3, (4)
cage5	5	(4,4)	(4,4)	2, (3)
Hamrle1	5	(3,3)	(3,3)	2, (3)
d_ss	—	(4,5)	(4,5)	0, (4,5)
d_dyn	—	(4,4)	(4,4)	0, (4,4)
ww_36_pmec_36	—	(8,8)	(8,9)	0, (7), (7); 2, (3)
steam3*	—	(3,3)	(3,3)	2, (3,3,3)

3.2 | SuiteSparse matrices

We now test a subset of matrices taken from the SuiteSparse Collection[16] whose properties are listed in Table 11. As before, we test SIR, SGMRES-IR, GMRES-IR, and MSIR with initial precisions (u_f, u, u_r) = (single, double, quad), (u_f, u, u_r) = (half, single, double), and (u_f, u, u_r) = (half, double, quad). For these problems, we always set the right-hand side b to be the vector of ones. Tables 12–14 show the convergence behavior of the different algorithmic variants. For each precision combination, we show a few interesting convergence trajectories in Figures 10–12.

In Table 12 for initial precisions (u_f, u, u_r) = (single, double, quad), it is observed that SIR converges for all matrices except ww_36_pmec_36 (the most ill-conditioned one). Since SIR converges in few steps, MSIR does not switch to SGMRES-IR. For ww_36_pmec_36, however, MSIR detects nonconvergence of SIR and switches to SGMRES-IR, which then converges in 3 steps.

In Table 13 for initial precisions (u_f, u, u_r) = (half, single, double), it is seen that SIR converges for the matrices with $\kappa_\infty(A) < 10^6$. It is also observed that SGMRES-IR fails to converge for the extremely ill-conditioned matrix ww_36_pmec_36. MSIR switches to GMRES-IR for the two most ill-conditioned matrices, although for the steam3 case, the second switch is not technically necessary. It is not clear why for the steam3 matrix, MSIR performs two SGMRES-IR steps and one GMRES-IR step when SGMRES-IR alone only requires one step. For ww_36_pmec_36, MSIR also switches from (u_f, u, u_r) = (half, single, double) to (u_f, u, u_r) = (single, single, double) since GMRES-IR required too many GMRES iterations. It is lastly seen that for steam3, scaling is performed due to the resulting L and U factors containing Inf or NaN when the factorization is performed in half precision, as explained in Section 2.1.2.

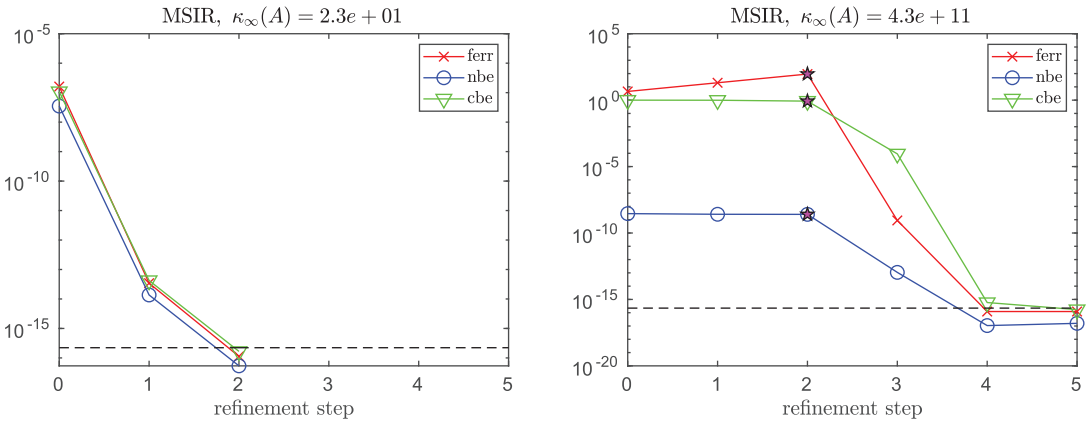

FIGURE 10 Convergence of errors in multistage iterative refinement for *cage6* (left) and *ww_36_pmec_36* (right) using initial precisions (u_f, u, u_r) = (single, double, quad)

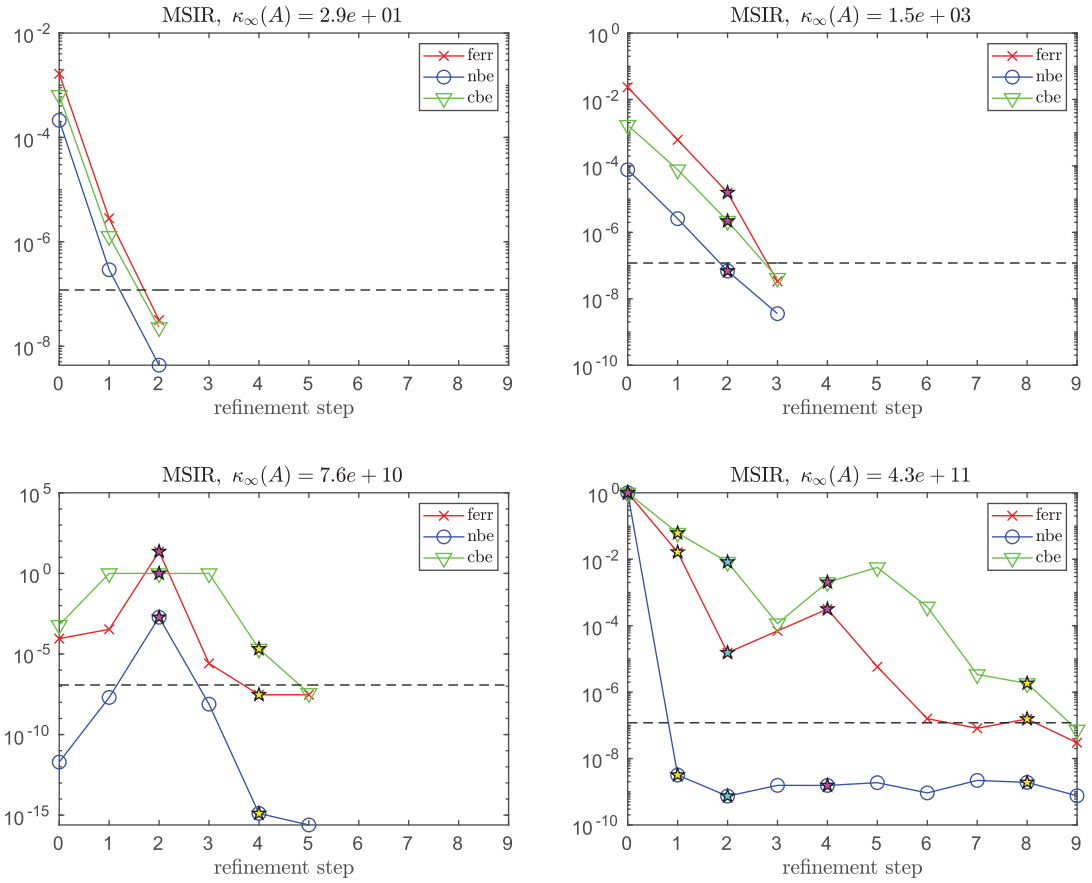

FIGURE 11 Convergence of errors in multistage iterative refinement for *cage5* (top left), *bwfa62* (top right), *steam3* (bottom left), and *ww_36_pmec_36* (bottom right) using initial precisions (u_f, u, u_r) = (half, single, double)

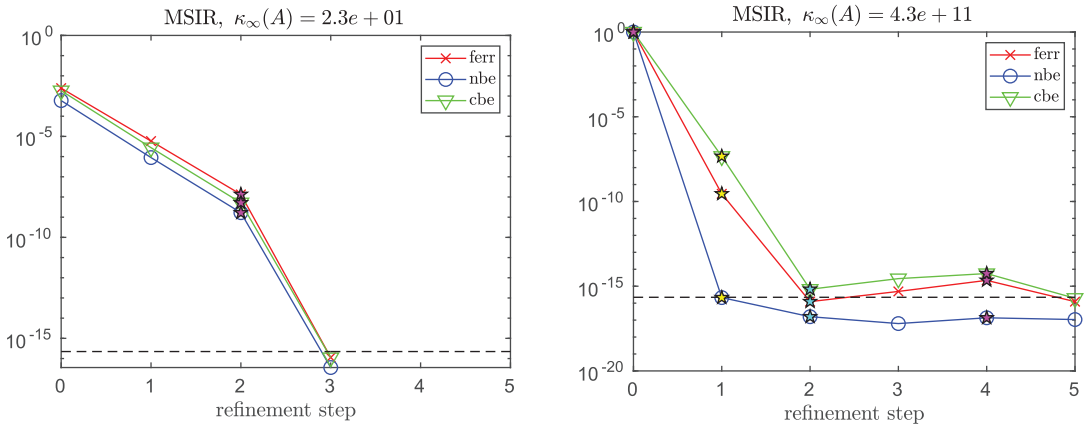

FIGURE 12 Convergence of errors in multistage iterative refinement for *cage6* (left) and *ww_36_pmec_36* (right) using initial precisions (u_f, u, u_r) = (half, double, quad)

In Table 14, using initial precisions (u_f, u, u_r) = (half, double, quad), we see that MSIR is able to converge for all test problems and switches at least to SGMRES-IR in every case due to slow convergence of SIR. For the matrix ww_36_pmec_36, MSIR makes the second switch to GMRES-IR and then does a precision switch from (u_f, u, u_r) = (half, double, quad) to (u_f, u, u_r) = (single, double, quad) after one step due to the constraint on the number of GMRES iterations per step. As in the (u_f, u, u_r) = (half, single, double) case, due to the use of half precision in the factorization process, scaling is required for steam3.

Overall, we note that for these real-world sparse problems, to an even greater extent than in the dense test cases, SGMRES-IR and GMRES-IR often still perform well for matrices with condition numbers greatly exceeding constraints given in Table 4. We again stress that this motivates our multistage approach, since the theoretical analysis cannot necessarily give indication of which algorithm variant will be the best choice.

4 | CONCLUSIONS AND FUTURE WORK

Mixed precision IR, and in particular, GMRES-based IR, has seen great use recently due to the emergence of mixed precision hardware. The freedom to choose different precision combinations as well as a particular solver leads to an explosion of potential IR variants, each of which has different performance costs and different condition number constraints under which convergence is guaranteed. In practice, particular IR variants often work well for problems beyond what is indicated by the analysis, making it difficult for a user to select a priori the least expensive variant that will converge. Motivated by improving usability and reliability, we have developed a multistage mixed precision IR approach, called MSIR. For a given combination of precisions, MSIR switches between three different IR variants distinguished by the way in which they solve for the correction to the approximate solution in order of increasing cost, according to stopping criteria adapted from Reference 14. Then, only if necessary, it increases the precision(s), refactorizes the matrix in a higher precision, and begins again. We discuss details of the algorithm and perform extensive numerical experiments on both random dense matrices and matrices from SuiteSparse[16] using a variety of initial precision combinations. Our experiments confirm that the algorithmic variants often outperform what is dictated by the theoretical condition number constraints and demonstrate the benefit of the multistage approach; in particular, our experiments show that there can be an advantage to first trying other solvers before resorting to increasing the precision and refactorizing.

There are a number of potential extensions to the work presented here. First, while we have only tested IEEE precisions, it is also worthwhile to perform experiments using non-IEEE floating point formats such as bfloat16.[19] We also plan to extend the multistage approach to least squares problems, which can be accomplished by combining the error bounds derived in Reference 22 and the three-precision IR schemes for least squares problems in Reference 23. Finally, while we have roughly used the number of triangular solves in a given precision as a metric for discussing probable performance characteristics, this may not be a good indication of performance in practice. Thorough high-performance experiments

on modern GPUs are needed to appropriately gauge the overhead of the MSIR approach and the algorithm's suitability in practice.

ACKNOWLEDGMENTS

The authors thank the anonymous referees, whose comments greatly improved the content and presentation of the paper.

CONFLICT OF INTEREST

This study does not have any conflicts to disclose.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in MSIR at https://github.com/edoktay/msir.

ORCID

Eda Oktay https://orcid.org/0000-0003-0761-2184
Erin Carson https://orcid.org/0000-0001-9469-7467

REFERENCES

1. HPL-AI mixed-precision benchmark; 2019. https://icl.bitbucket.io/hpl-ai/
2. Kudo S, Nitadori K, Ina T & Imamura T Prompt report on exa-scale HPL-AI benchmark. Proceedings of the 2020 IEEE International Conference on Cluster Computing (CLUSTER), Kobe, Japan; 2020;418–9.
3. TOP500; 2021 [Online].
4. Carson E, Higham NJ. Accelerating the solution of linear systems by iterative refinement in three precisions. SIAM J Sci Comput. 2018;40(2):A817–47.
5. Wilkinson JH. Progress report on the automatic computing engine. Teddington, UK: Mathematics Division, Department of Scientific and Industrial Research, National Physical Laboratory; 1948 MA/17/1024.
6. Wilkinson JH. Rounding errors in algebraic processes. National Physical Laboratory Notes on Applied Science. Vol 32. London, UK: Office of Public Sector Information; 1963.
7. Moler CB. Iterative refinement in floating point. J Assoc Comput Mac. 1967;14(2):316–21.
8. Jankowski M, Woźniakowski H. Iterative refinement implies numerical stability. BIT Numer Math. 1977;17(3):303–11.
9. Langou J, Langou J, Luszczek P, Kurzak J, Buttari A, Dongarra J. Exploiting the performance of 32 bit floating point arithmetic in obtaining 64 bit accuracy (revisiting iterative refinement for linear systems). Proceedings of the 2006 ACM/IEEE Conference on Supercomputing SC'06. IEEE; 2006. p. 50.
10. Carson E, Higham NJ. a new analysis of iterative refinement and its application to accurate solution of ill-conditioned sparse linear systems. SIAM J Sci Comput. 2017;39(6):A2834–56.
11. Haidar A, Tomov S, Dongarra J & Higham NJ Harnessing GPU tensor cores for fast FP16 arithmetic to speed up mixed-precision iterative refinement solvers. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis SC18, Dallas, Texas, USA; 2018;603–13.
12. Amestoy P, Buttari A, Higham NJ, L'Excellent JY, Mary T, Vieublé B. Five-precision GMRES-based iterative refinement. Manchester, UK: Manchester Institute for Mathematical Sciences, University of Manchester; 2021.p. 5.
13. Paige CC, Rozložník M, Strakoš Z. Modified Gram-Schmidt (MGS), least squares, and backward stability of MGS-GMRES. SIAM J Matrix Anal Appl. 2006;28(1):264–84.
14. Demmel J, Hida Y, Kahan W, Li XS, Mukherjee S, Riedy EJ. Error bounds from extra-precise iterative refinement. ACM Trans Math Softw. 2006;32(2):325–51.
15. Tomov S, Dongarra J, Baboulin M. Towards dense linear algebra for hybrid GPU accelerated manycore systems. Parallel Comput. 2010;36(5-6):232–40.
16. Davis TA, Hu Y. The University of Florida sparse matrix collection. ACM Trans Math Softw. 2011;38(1):1–25.
17. Bowdler HJ, Martin R, Peters G, Wilkinson J. Solution of real and complex systems of linear equations. Numer Math. 1966;8(3): 217–34.
18. Higham NJ, Pranesh S, Zounon M. Squeezing a matrix into half precision, with an application to solving linear systems. SIAM J Sci Comput. 2019;41(4):A2536–51.
19. Intel Corporation. BFLOAT16 – Hardware numerics definition. Intel. 338302-001US, revision 1.0; 2018.
20. Kalfa, M., Ertürk, V. B., Ergül, Ö. Error Analysis of MLFMA With Closed-Form Expression. IEEE Trans Antenn Propag. 2021;60(10):6618–6623.
21. Higham NJ, Pranesh S. Simulating low precision floating-point arithmetic. SIAM J Sci Comput. 2019;41(5):C585–602.
22. Demmel J, Hida Y, Riedy EJ, Li XS. Extra-precise iterative refinement for overdetermined least squares problems. ACM Trans Math Softw. 2009;35(4):1–32.

23. Carson E, Higham NJ, Pranesh S. Three-precision GMRES-based iterative refinement for least squares problems. SIAM J Sci Comput. 2020;42(6):A4063–83.

How to cite this article: Oktay E, Carson E. Multistage mixed precision iterative refinement. Numer Linear Algebra Appl. 2022;e2434. https://doi.org/10.1002/nla.2434