Programs and Algorithms of Numerical Mathematics 21
J. Chleboun, P. Kůs, J. Papež, M. Rozložník, K. Segeth, J. Šístek (Eds.)
Institute of Mathematics CAS, Prague 2023

MIXED PRECISION GMRES-BASED ITERATIVE REFINEMENT WITH RECYCLING

Eda Oktay, Erin Carson

Faculty of Mathematics and Physics, Charles University
Prague, Czech Republic
oktay@karlin.mff.cuni.cz, carson@karlin.mff.cuni.cz

Abstract: With the emergence of mixed precision hardware, mixed precision GMRES-based iterative refinement schemes for solving linear systems $Ax = b$ have recently been developed. However, in certain settings, GMRES may require too many iterations per refinement step, making it potentially more expensive than the alternative of recomputing the LU factors in a higher precision. In this work, we incorporate the idea of Krylov subspace recycling, a well-known technique for reusing information across sequential invocations, of a Krylov subspace method into a mixed precision GMRES-based iterative refinement solver. The insight is that in each refinement step, we call preconditioned GMRES on a linear system with the same coefficient matrix A. In this way, the GMRES solves in subsequent refinement steps can be accelerated by recycling information obtained from previous steps. We perform numerical experiments on various random dense problems, Toeplitz problems, and problems from real applications, which confirm the benefits of the recycling approach.

Keywords: GMRES, iterative refinement, mixed precision, recycling

MSC: 65F08, 65F10, 65G50, 65Y10

1. Introduction and background

There are various algorithms for solving linear systems of equations $Ax = b$, where $A \in \mathbb{R}^{n\times n}$ and $x, b \in \mathbb{R}^n$. One approach is iterative refinement (IR) which is based on improving the approximate solution in each refinement step [23]. Iterative refinement typically starts with using Gaussian elimination with partial pivoting (GEPP) to compute an initial approximate solution. Then using the L and U factors of A and the residual r_i, the system $Ad_{i+1} = r_i$ is solved for the correction term d_i to improve the approximate solution via $x_{i+1} = x_i + d_{i+1}$. A general IR scheme is shown in Algorithm 1.

Recently, mixed-precision capabilities have become available in hardware, which can have significant performance benefits [4, 1]. In Algorithm 1, the authors in [6]

DOI: 10.21136/panm.2022.14

used three hardware precisions: u_r for computing the residual, u_f for the LU factorization, and working precision u for the remaining calculations. There is also a fourth "effective solve precision" u_s, which depends on the particular solver and precisions used in line 5. The idea is that low precision can be used for the LU factorization, which is the most expensive part of the computation, and accuracy can be recovered through use of higher precisions in other parts of the computation. Indeed, it has been shown that on NVIDIA V100 GPUs, using half precision instead of double precision for the LU factorization can give over $4\times$ speedups; see [12, Figure 3(b)]. Throughout this work we assume that $u_f \geq u \geq u_r$ e.g., $(u_f, u, u_r)=$ (half, single, double).

Algorithm 1 General Iterative Refinement Scheme

```
Input: $n \times n$ matrix $A$; right-hand side $b$; max. number of refinement steps $i_{\max}$.
Output: Approximate solution $x_{i+1}$ to $Ax = b$.
1: Compute LU factorization $A \approx LU$ 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 $x_{i+1}$ in precision $u$. end if
```

For a given combination of precisions and choice of solver, it is well-understood under which conditions Algorithm 1 will converge and what the limiting accuracy will be. The constraint for convergence in line 7 is usually stated via a constraint on the infinity-norm condition number of the matrix A. Table 1 shows the 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 and solvers used in this study. For further information, see, e.g., [6, 3]. For a description of stopping criteria used for detecting convergence within iterative refinement in practice, see, e.g., [9], [19].

From Table 1, we see that if the computed LU factors are used to solve for the correction in line 5, often referred to as "standard IR" (SIR), then $\kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$ must be less than u_f^{-1} in order for convergence to be guaranteed. To relax this constraint on condition number, the authors of [5] and [6] devised a mixed precision GMRES-based iterative refinement scheme (GMRES-IR). In GMRES-IR, the correction equation in line 5 is solved via left-preconditioned GMRES, where the computed LU factors of A are used as preconditioners. This results in a looser constraint on condition number in order to guarantee the convergence of forward and backward errors; in the case that the preconditioned matrix is applied to a vector in each iteration of GMRES in double the working precision, we require $\kappa_\infty(A) \leq u^{-1/2} u_f^{-1}$, and in the case that a uniform precision is used within GMRES (a variant

u_f	u	u_r	SIR	GMRES-IR	SGMRES-IR
half	single	double	$2 \cdot 10^3$	$8 \cdot 10^6$	$4 \cdot 10^4$
half	double	quad	$2 \cdot 10^3$	$2 \cdot 10^{11}$	$3 \cdot 10^7$
single	double	quad	$2 \cdot 10^7$	$2 \cdot 10^{15}$	$1 \cdot 10^{10}$

Table 1: 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 SIR, GMRES-IR (which uses double the working precision in applying the preconditioned matrix to a vector) and SGMRES-IR (which uses the working precision throughout).

which we call SGMRES-IR), we require $\kappa_\infty(A) \leq u^{-1/3} u_f^{-2/3}$; see [3]. If these constraints are met, preconditioned GMRES is guaranteed to converge to a backward stable solution after n iterations and the iterative refinement scheme will converge to its limiting accuracy. We note that to guarantee backward stability, all existing analyses (e.g., [5, 6, 3]) assume that unrestarted GMRES is used within GMRES-IR.

We also note that existing analyses do not guarantee how fast GMRES will converge in each refinement step, only that it will do so within n iterations. However, if indeed n iterations are required to converge in each GMRES solve, this can make GMRES-IR more expensive than simply computing the LU factorization in higher precision and using SIR. Indeed, high-performance experiments show that slow GMRES convergence can negatively impact the achievable performance; see [12, Figure 7 (b)]. To make more precise the relative costs of each step of SIR and GMRES-IR, we list their costs in terms of asymptotic computational complexity in Table 2.

Unfortunately, GMRES convergence speed is difficult to predict. In fact, for any set of prescribed eigenvalues, one can construct a linear system for which GMRES will stagnate entirely until the nth iteration [11]. The situation is better understood at least in the case of normal matrices; see, e.g., [15]. The worst-case scenario in the case of normal A is when eigenvalues are clustered near the origin, which can cause complete stagnation of GMRES [15]. After the preconditioning step in GMRES-IR, all eigenvalues of the preconditioned matrix ($U^{-1}L^{-1}A$) ideally become 1 in the absence of finite precision error in computing LU and within GMRES. However, in practice, since we have inexact LU factors, if A has a cluster of eigenvalues near the origin, this imperfect preconditioner may fail to shift some of them away from the origin, which can cause GMRES to stagnate. For instance, when random dense matrices having geometrically distributed singular values are used in the multistage iterative refinement algorithm devised in [19], the authors showed that for relatively large condition numbers relative to precision u_f, GMRES tends to perform n iterations in each refinement step.

Figure 1 shows the eigenvalue distribution of a double-precision 100×100 random dense matrix having geometrically distributed singular values with condition number $\kappa_2(A) = 10^{12}$, generated via the command `gallery('randsvd',100,1e12,3)` in

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

Table 2: Asymptotic computational complexity of operations in each refinement step for SIR and GMRES-IR.

MATLAB. In the unpreconditioned case (left), the eigenvalues are clustered around the origin, a known difficult case for GMRES. When double-precision LU factors are used for preconditioning (middle), the eigenvalues of the preconditioned system are now clustered around 1. On the other hand, using half-precision LU factors as preconditioners (right) causes a cluster of eigenvalues to remain near the origin, indicating that GMRES convergence will likely be slow (we note that these are nonnormal matrices and so the theory of [15] does not apply, but our experimental evidence indicates that this is the case).

Thus, even when low-precision LU factors can theoretically be used in GMRES-IR, they may not be the best choice from a performance perspective. In this scenario, we are left with two options: either increase the precision in which the LU factors are computed, or seek to improve the convergence behavior of GMRES through other means. It is the latter approach that we take in this work. In particular, we investigate the use of Krylov subspace recycling.

In Section 2, we give a background on the use of recycling in Krylov subspace methods and describe our approach. Extensive numerical experiments that demonstrate the potential benefit of recycling within GMRES-based iterative refinement are presented in Section 3. We outline open problems in Section 4.

2. Iterative refinement with Krylov subspace recycling

One way to speed up the convergence of GMRES is using recycling [20, 21]. The idea of recycling is that if we have a sequence of linear systems ($Ax_1 = b_1, Ax_2 = b_2, \ldots$) to solve involving the same (or a similar) coefficient matrix A, then we can reuse the Krylov subspace information generated in solving $Ax_1 = b_1$ to speed up converge of the method in solving $Ax_2 = b_2$, etc. This is exactly the situation we have in GMRES-IR: within the iterative refinement loop, we call GMRES on the matrix A many times, and only the right-hand side changes between refinement steps. Thus we can use Krylov subspace recycling within GMRES across iterative refinement steps, and theoretically the convergence of GMRES should improve as the refinement proceeds.

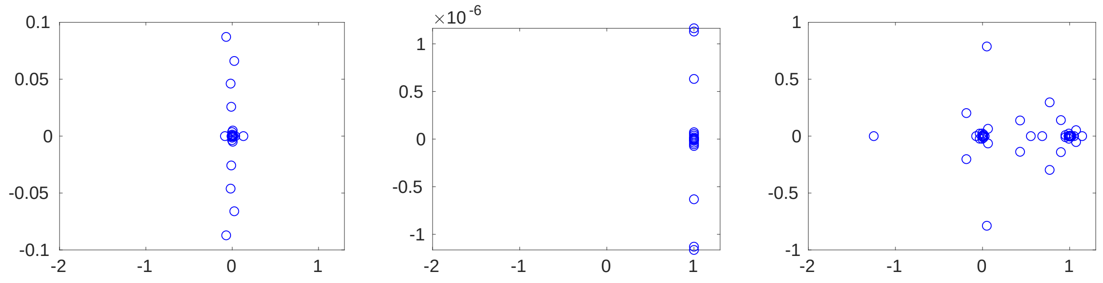

Figure 1: Eigenvalue distribution of a double-precision random dense matrix with $\kappa_2(A) = 10^{12}$ without preconditioner (left), with a double-precision LU preconditioner (middle), and with a half-precision LU preconditioner (right).

GMRES-DR [17] is a truncated and restarted solver developed for solving single nonsymmetric linear systems. The method deflates small eigenvalues for the new subspace to improve the convergence of restarted GMRES. Another truncated solver used for recycling is called GCROT [8]. The method recycles a subspace that minimizes the loss of orthogonality with Krylov subspace from the previous system.

By far the most popular Krylov subspace method implementing recycling is GCRO-DR [20]. GCRO-DR can be seen as a combination of GMRES-DR and a modified GCROT. In GCRO-DR, the residual minimization and orthogonalization are performed over the recycled subspace, leading to an adaptive truncated recycling method. GCRO-DR uses the deflated restarting idea in GMRES-DR in the same manner as GCROT. Let m denote the maximum size of the Krylov subspace and let k denote the number of vectors to recycle. In one cycle of GCRO-DR(m, k), first, using the k harmonic Ritz vectors corresponding to the k smallest harmonic Ritz values, the solution space is constructed. After finding the optimal solution over the solution space and computing the residual, GCRO-DR constructs the Arnoldi relation by generating a Krylov subspace of dimension $m - k + 1$. After completing the Arnoldi process, the algorithm solves a minimization problem at the end of each cycle, which reduces to an $(m + 1) \times m$ least-squares problem. After solving the least-squares problem and computing the residual, a generalized eigenvalue problem is solved, and harmonic Ritz vectors are recovered. Since harmonic Ritz vectors are constructed differently than in GMRES-DR, GCRO-DR is suitable for solving individual linear systems and sequences of them. See [20] for further details.

The use of recycling may also be favorable from a performance perspective. GCRO-DR performs only $m - k$ Arnoldi steps implying that it performs $m - k$ matrix-vector multiplications per cycle, whereas GMRES(m) performs m matrix-vector multiplications. It is also mentioned in [20] that since GCRO-DR stores U_k and C_k, it performs $2kn(1 + k)$ fewer operations during the Arnoldi process. On the other hand, since we are using k eigenvectors, GCRO-DR(m, k) requires storing k more vectors than GMRES(m).

In an effort to reduce the overall computational cost of the GMRES solves within GMRES-IR, we develop a recycled GMRES-based iterative refinement algorithm, called RGMRES-IR. In line 5 of Algorithm 1, RGMRES-IR uses preconditioned GCRO-DR(m, k) instead of preconditioned GMRES to solve the correction equation. As in GMRES-IR, RGMRES-IR will use three precisions: u_f, u, and u_r. Again we note that we assume $u_r \le u \le u_f$. Using different precision settings results in different constraints on the condition number to guarantee convergence of the forward and backward errors. Although our experiments here will use three different precisions, two precisions (only computing residuals in higher precision) or fixed (uniform) precision can also be used in the RGMRES-IR algorithm.

3. Numerical experiments

In this section, we compare GMRES-IR and RGMRES-IR for solving $Ax = b$. We adapted MATLAB implementations of the GMRES-IR method from [6], and the GCRO-DR method from [20]. To simulate half-precision, we use the `chop` library and associated functions from [13], available at `https://github.com/higham/chop` and `https://github.com/SrikaraPranesh/LowPrecision_Simulation`. For single and double precision, we use the MATLAB built-in data types and to simulate quadruple precision we use the Advanpix multiprecision computing toolbox, see [2]. We restrict ourselves to IEEE precisions, although we note that one could also use formats like bfloat16 [14]. The experiments are performed on a computer with Intel Core i7-9750H having 12 CPUs and 16 GB RAM with OS system Ubuntu 20.04.1. Our RGMRES-IR algorithm and associated functions are available at `https://github.com/edoktay/rgmresir`, which includes scripts for generating the data and plots in this work.

The GMRES convergence tolerance τ dictates the stopping criterion for the inner GMRES iterations. GMRES is considered converged if the relative (preconditioned) residual norm drops below τ. In tests here with single working precision, we use $\tau = 10^{-4}$. For double working precision, we use $\tau = 10^{-8}$. Note that for the outer iterative refinement scheme, we explicitly compute the true solution x and stop the iterations if the forward and backward errors are less than u. For practical stopping criteria relevant to GMRES-IR schemes, see [19]. To ensure that we fully exhibit the behavior of the methods, we set the maximum number of refinement steps to $i_{\max} = 10000$, which is large enough to allow all approaches that eventually converge sufficient time to do so.

The results are compared in two different metrics: the number of GMRES iterations per refinement step and the total number of GMRES iterations. For simplicity, b is chosen to be the vector of ones for all matrices, and the precisions are chosen such that $u \le u_f^2$, and $u_r \le u^2$. We compare GMRES-IR and RGMRES-IR in the setting where precision u^2 is used for preconditioning, except for the experiments in Section 3.3.1 where we investigate the use of uniform precision within the solver.

For a fair comparison between GMRES-IR and RGMRES-IR, GMRES-IR is used with restart value m, which is the maximum size of the subspace used in RGMRES-IR. Since the first refinement step of RGMRES-IR does not have a recycled subspace, it is the same as the first step of GMRES-IR. We thus expect a decrease in the number of GMRES iterations per refinement step starting from the second refinement step. For RGMRES-IR, the optimal number $k < m$ of harmonic Ritz vectors is chosen for each group of matrices with the desired precision settings after several experiments on various (m, k) scenarios. The optimum k differs for each matrix. The least total number of GMRES iterations is obtained for $k =$ (# of GMRES iterations in the first refinement step) -1 since, in this case, we are recycling the whole generated subspace, which is expensive. Thus one should choose a k value as small as possible to reduce computational cost while benefiting from recycling. Figure 2 shows the change in the total GMRES iterations according to the given k values for two matrices. From the plots, one can easily *find the knee*, i.e., find the smallest k value that gives a reasonably small number of GMRES iterations.

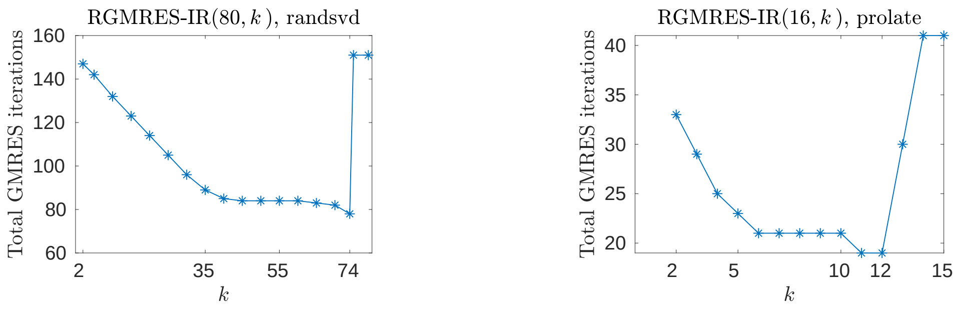

Figure 2: Total GMRES iterations for various k for a randsvd matrix with $\kappa_2(A) = 10^{13}$ (left) and a prolate matrix with $\alpha = 0.434$ (right) for $(u_f, u, u_r) =$ (single, double, quad).

In the tables we will present, the first number shows the total number of GMRES iterations. The numbers in the parentheses indicate the number of GMRES iterations performed in each refinement step. For instance, 5(2,3) implies that there are 2 refinement steps, the first of which performs 2 GMRES iterations and the second of which performs 3, giving a total of 5 GMRES iterations. We note that we use the GMRES iteration count as a proxy for performance, although this is not a perfect measure; for recent results on mixed precision GMRES-IR with restarting see [16]. We also note that additional numerical experiments for RGMRES-IR can be found in the associated technical report [18].

3.1. Prolate matrices

We first test our algorithm on prolate (symmetric, ill-conditioned Toeplitz matrices whose eigenvalues are distinct, lie in the interval $(0, 1)$, and tend to cluster around 0 and 1) matrices [22] of dimension $n = 100$, generated using the MATLAB command `gallery('prolate',n,alpha)`, where `alpha` is the array of the desired parameters α ={0.475, 0.47, 0.467, 0.455, 0.45, 0.4468, 0.44, 0.434}. When $\alpha < 0.5$, it becomes difficult for GMRES-IR to solve the system since the eigenvalues skew more towards zero. Table 3 shows the number of GMRES iterations performed by GMRES-IR and RGMRES-IR for the setting (u_f, u, u_r) = (half, single, double).

From Table 3, we can see that GMRES-IR diverges for $\alpha < 0.45$ with and without recycling. However, when $\alpha = 0.45$, we see that RGMRES-IR diverges although GMRES-IR converges. This is because of the multiple periods of stagnation in the second refinement step due to recycling. GCRO-DR cannot converge in the first $16 - 5 = 11$ iterations in the second step, causing an infinite restart which results in divergence. However, for cases where both GMRES-IR and RGMRES-IR converge, RGMRES-IR always requires fewer GMRES iterations. For $\alpha = 0.455$, GMRES restarts in the second refinement step of GMRES-IR, while recycling allows RGMRES-IR to converges without restarting, decreasing the cost.

α	$\kappa_\infty(A)$	$\kappa_2(A)$	GMRES-IR (16)	RGMRES-IR (16,5)
0.475	$1 \cdot 10^6$	$4 \cdot 10^5$	12 (6,6)	8 (6,2)
0.47	$3 \cdot 10^7$	$8 \cdot 10^6$	16 (8,8)	10 (8,2)
0.467	$2 \cdot 10^8$	$5 \cdot 10^7$	19 (9,10)	11 (9,2)
0.455	$3 \cdot 10^{11}$	$8 \cdot 10^{10}$	50 (15,25,10)	19 (15,4)
0.45	$7 \cdot 10^{12}$	$2 \cdot 10^{12}$	89 (14,43,32)	-
0.4468	$5 \cdot 10^{13}$	$1 \cdot 10^{13}$	-	-
0.44	$3 \cdot 10^{15}$	$9 \cdot 10^{14}$	-	-
0.434	$3 \cdot 10^{16}$	$9 \cdot 10^{15}$	-	-

Table 3: Number of GMRES-IR and RGMRES-IR refinement steps/GMRES iterations for prolate matrices with various α values, using precisions (u_f, u, u_r) = (half, single, double) and (m, k) = (16,5).

3.2. SuiteSparse matrices

We now test our algorithm on three real matrices taken from the SuiteSparse Collection [7]. Table 4 compares the performance of GMRES-IR and RGMRES-IR for precisions (u_f, u, u_r) = (half, double, quad). It is seen that RGMRES-IR successfully reduces the total number of GMRES iterations in all cases.

3.3. Random dense matrices

Finally, we test our algorithm on random dense matrices of dimension $n = 100$ having geometrically distributed singular values. We generated the matrices using the MATLAB command `gallery('randsvd',n,kappa(i),3)`, where `kappa` is the array of the desired 2-norm condition numbers $\kappa_2(A)$ ={10^4, 10^5, 10^6, 10^7, 10^8, 10^9, 10^{10}, 10^{11}, 10^{12}, 10^{13}}, and mode `3` corresponds to the matrix having geometrically

Matrix	n	nnz	$\kappa_\infty(A)$	GMRES-IR (40)	RGMRES-IR(40,10)
orsirr_1	1030	6858	$1 \cdot 10^5$	22 (11,11)	20 (11,9)
comsol	1500	97645	$3 \cdot 10^6$	52 (25,27)	34 (25,9)
circuit204	1020	5883	$9 \cdot 10^9$	59 (18,20,21)	47 (18,14,15)

Table 4: Number of GMRES-IR and RGMRES-IR refinement steps/GMRES iterations for real matrices, using precisions (u_f, u, u_r) = (half, double, quad) and (m, k) = (40,10).

distributed singular values. For reproducibility, we use the MATLAB command `rng(1)` each time we run the algorithm. We compare methods using precisions (u_f, u, u_r) = (single, double, quad) and (u_f, u, u_r) = (half, double, quad). As shown in Figure 1, these matrices have eigenvalues clustered around the origin, which can be a difficult case for GMRES convergence. This class of problems thus represents a good use case for the RGMRES-IR algorithm.

3.3.1. SGMRES-IR versus RSGMRES-IR

In practice, implementations often use a uniform precision within GMRES (i.e., applying the preconditioned matrix to a vector in precision u rather than u^2). This is beneficial from a performance perspective (in particular if precision u^2 must be implemented in software). The cost is that the constraint on condition numbers for which the refinement scheme is guaranteed to converge becomes tighter. To illustrate the benefit of recycling in this scenario, we first compare what we call SGMRES-IR (GMRES-IR but with a uniform precision within GMRES) to the recycled version, RSGMRES-IR. For a fair comparison, restarted SGMRES-IR (SGMRES-IR(m)) is compared with recycled SGMRES-IR (RSGMRES-IR(m, k)).

Table 5 shows the number of GMRES iterations performed by SGMRES-IR and RSGMRES-IR in the (u_f, u, u_r) = (single, double, quad) setting. We observe that recycling reduces the number of GMRES iterations in this case as well. The reason why SGMRES-IR does not converge for $\kappa_2(A) \geq 10^{14}$ is that in the first refinement step, restarted SGMRES does not converge (restarting an infinite number of times). For RSGMRES-IR, in the first GCRO-DR call, recycling after the first restart cycle helps, allowing GCRO-DR to converge. We note that this is another benefit of the recycling approach, as it can improve the reliability of restarted GMRES, which is almost always used in practice.

3.3.2. GMRES-IR versus RGMRES-IR

We now return to our usual setting and compare GMRES-IR and RGMRES-IR for random dense matrices with condition numbers $\kappa_2(A) = \{10^4, 10^5, 10^6, 10^7, 10^8, 10^9, 10^{10}, 10^{11}, 10^{12}, 10^{13}, 10^{14}, 10^{14}\}$. Results using precisions (u_f, u, u_r) = (half, double, quad) and two different choices of (m, k) are displayed in Table 6.

$\kappa_\infty(A)$	$\kappa_2(A)$	SGMRES-IR (80)	RSGMRES-IR (80,18)
$6 \cdot 10^9$	10^9	64 (19,23,22)	34 (19,8,7)
$6 \cdot 10^{10}$	10^{10}	120 (39,40,41)	65 (39,13,13)
$6 \cdot 10^{11}$	10^{11}	160 (52,54,54)	94 (52,21,21)
$6 \cdot 10^{12}$	10^{12}	196 (65,65,66)	163 (65,32,32,34)
$5 \cdot 10^{13}$	10^{13}	301 (75,75,75,76)	199 (75,41,41,42)
$5 \cdot 10^{14}$	10^{14}	-	493 (131,51,51,52,52,52,52,52)
$5 \cdot 10^{15}$	10^{15}	-	2093*

Table 5: Number of SGMRES-IR and RSGMRES-IR refinement steps/GMRES iterations for precisions (u_f, u, u_r) = (single, double, quad) and (m, k) = (80,18). For $\kappa_2(A) = 10^{15}$, RSGMRES-IR required 2093 total GMRES iterations over 37 refinement steps.

$\kappa_\infty(A)$	$\kappa_2(A)$	GMRES-IR (100)	RGMRES-IR (100,30)	GMRES-IR (90)	RGMRES-IR (90,40)
$9 \cdot 10^4$	10^4	33 (16,17)	33 (16,17)	33 (16,17)	33 (16,17)
$8 \cdot 10^5$	10^5	85 (41,44)	71 (41,15,15)	85 (41,44)	50 (41,9)
$7 \cdot 10^6$	10^6	134 (66,68)	85 (66,19)	134 (66,68)	81 (66,15)
$7 \cdot 10^7$	10^7	167 (83,84)	113 (83,30)	167 (83,84)	100 (83,17)
$7 \cdot 10^8$	10^8	193 (96,97)	138 (96,42)	-	149 (119,30)
$6 \cdot 10^9$	10^9	200 (100,100)	151 (100,51)	-	179 (134,45)
$6 \cdot 10^{10}$	10^{10}	200 (100,100)	158 (100,58)	-	470 (388,41,41)
$6 \cdot 10^{11}$	10^{11}	200 (100,100)	165 (100,65)	-	-
$6 \cdot 10^{12}$	10^{12}	200 (100,100)	170 (100,70)	-	-
$5 \cdot 10^{13}$	10^{13}	3954*	241 (171,70)	-	-

Table 6: Number of GMRES-IR and RGMRES-IR refinement steps/GMRES iterations for random dense matrices having geometrically distributed singular values (mode 3) with various condition numbers, using precisions (u_f, u, u_r) = (half, double, quad) and settings (m, k) = (100,30) and (m, k)=(90,40). For $m = 100$ and 10^{13}, GMRES-IR required 3954 total GMRES iterations over 45 refinement steps.

For both choices of (m, k), when $\kappa_\infty(A) > 10^5$, recycling reduces the total number of GMRES iterations. This class of matrices with a low-precision LU preconditioner is a known difficult case for GMRES, and thus we can clearly see the benefits of recycling. We see the most significant improvement for the matrix with $\kappa_2(A) = 10^{13}$, in which RGMRES-IR requires over $16\times$ fewer GMRES iterations than GMRES-IR when $m = 100$. We note that GMRES-IR is only guaranteed to converge up to $\kappa_2(A) < 10^{12}$ for this combination of precisions; for detailed information, see [6].

The reason that RGMRES-IR outperforms GMRES-IR for $m = 100$ and $\kappa_2(A) = 10^{13}$ is different than in the previous cases (caused by stagnation due to restarting), and is almost accidental in this case. We investigate this more closely in Figure 3. In the left plot, we see the convergence trajectory of GMRES(100). In the first restart cycle, the residual decreases from 10^6 to 10^3 after 100 GMRES iterations. GMRES restarts and performs two more iterations, at which point it converges to a relative residual of 10^{-8} (absolute residual of around 10^{-2}). Hence, the first refinement step of GMRES-IR does $100 + 2 = 102$ iterations. The right plot shows the residual trajectory for GCRO-DR. The first restart cycle is the same as in GMRES; however, once the method restarts, the residual stagnates just above the level required to declare convergence. After $m - k = 70$ more iterations, GCRO-DR restarts again, and this time, the residual drops significantly lower. So while GCRO-DR requires more iterations (171) to converge to the specified tolerance, when it does converge, it converges to a solution with a smaller residual. This phenomenon can in turn reduce the total number of refinement steps required. It is possible that we could reduce the overall number of GMRES iterations within GMRES-IR (and also RGMRES-IR) by making the GMRES convergence tolerance τ smaller. We did not experiment with changing the GMRES tolerance within GMRES-IR or RGMRES-IR, but this trade-off would be interesting to explore in the future.

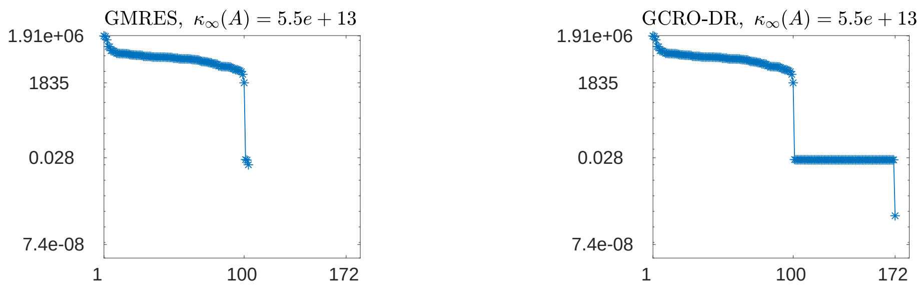

Figure 3: Residual trajectory of GMRES (left) and GCRO-DR (right), used within GMRES-IR and RGMRES-IR, respectively, for a randsvd matrix with $\kappa_2(A) = 10^{13}$ and precisions (u_f, u, u_r) = (half, double, quad).

We stress that the convergence guarantees for GMRES-IR for various precisions stated in [5, 6, 3] hold only for the case of unrestarted GMRES, i.e., $m = n$. When $m < n$, there is no guarantee that GMRES converges to a backward stable solution and thus no guarantee that GMRES-IR will converge. Choosing a restart parameter m that allows for convergence is a difficult problem, and a full theory regarding the behavior of restarted GMRES is lacking. The behavior of restarted GMRES is often unintuitive; whereas one would think that a larger restart parameter is likely to be better than a smaller one as it is closer to unrestarted GMRES, this is not always the case. In [10], the author gives examples where a larger restart parameter causes complete stagnation, whereas a smaller one results in fast convergence.

In Table 6 for the case $m = 90$, we can see that both methods converge for $\kappa_\infty(A) < 10^8$. After this point, GMRES-IR does not converge, whereas RGMRES-IR does. This serves as an example where the convergence guarantees given in [5, 6] do not hold for GMRES-IR with restarted GMRES; for unrestarted GMRES, convergence is guaranteed up to $\kappa_\infty(A) \leq 10^{12}$ for this precision setting. Here, GMRES-IR does not converge because of the stagnation caused by restarting in the first refinement step. Aided by the recycling between restart cycles, RGMRES-IR does converge up to $\kappa_2(A) = 10^{10}$, although the large number of GMRES iterations required in the first refinement step makes this approach impractical.

4. Conclusion and future work

In this work, we have incorporated Krylov subspace recycling into mixed precision GMRES-based iterative refinement in order to reduce the total number of GMRES iterations required. We call our algorithm RGMRES-IR. Instead of preconditioned GMRES, RGMRES-IR uses a preconditioned GCRO-DR algorithm to solve for the approximate solution update in each refinement step. Our numerical experiments on random dense matrices, prolate matrices, and matrices from SuiteSparse [7] show the potential benefit of the recycling approach. Even in cases where the number of GMRES iterations does not preclude the use of GMRES-based iterative refinement, recycling can have a benefit. In particular, it can improve the reliability of restarted GMRES, which is used in most practical scenarios.

One major caveat for GMRES-based iterative refinement schemes is that the analysis and convergence criteria discussed in the literature all rely on the use of unrestarted GMRES. When restarted GMRES is used, we cannot give such concrete guarantees, as restarted GMRES may not converge even in infinite precision. A greater understanding of the theoretical behavior of restarted GMRES (and GCRO-DR) both in infinite and finite precision would be of great interest. Another potential future direction is the exploration of the potential for the use of mixed precision within GCRO-DR. We expect that it may be possible to use low precision within GCRO-DR, for example, in the computation of harmonic Ritz pairs.

Acknowledgements

We acknowledge funding from Charles University project 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.

References

[1] Abdelfattah, A. et al.: A survey of numerical linear algebra methods utilizing mixed-precision arithmetic. The International Journal of High Performance Computing Applications **35** (2021), 344–369.

[2] Advanpix LLC. Multiprecision Computing Toolbox for MATLAB. URL http://www.advanpix.com/.

[3] Amestoy, P. et al.: Five-precision GMRES-based iterative refinement. MIMS EPrint 2021.5, Manchester Institute for Mathematical Sciences, The University of Manchester, Manchester, UK, 2021. URL http://eprints.maths.manchester.ac.uk/2807/.

[4] Baboulin, M. et al.: Accelerating scientific computations with mixed precision algorithms. Computer Physics Communications **180** (2009), 2526–2533.

[5] Carson, E. and Higham, N.J.: A new analysis of iterative refinement and its application to accurate solution of ill-conditioned sparse linear systems. SIAM Journal on Scientific Computing **39** (2017), A2834–A2856.

[6] Carson, E. and Higham, N.J.: Accelerating the solution of linear systems by iterative refinement in three precisions. SIAM Journal on Scientific Computing **40** (2018), A817–A847.

[7] Davis, T.A. and Hu, Y.: The University of Florida Sparse Matrix Collection. ACM Transactions on Mathematical Software **38** (2011).

[8] De Sturler, E.: Truncation strategies for optimal Krylov subspace methods. SIAM Journal on Numerical Analysis **36** (1999), 864–889.

[9] Demmel, J. et al.: Error bounds from extra-precise iterative refinement. ACM Trans. Math. Softw. **32** (2006), 325–351.

[10] Embree, M.: The tortoise and the hare restart GMRES. SIAM Review **45** (2003), 259–266.

[11] Greenbaum, A., Pták, V., and Strakoš, Z.: Any nonincreasing convergence curve is possible for GMRES. SIAM Journal on Matrix Analysis and Applications **17** (1996), 465–469.

[12] Haidar, A., Tomov, S., Dongarra, J., and Higham, N.J.: Harnessing gpu tensor cores for fast fp16 arithmetic to speed up mixed-precision iterative refinement solvers. In: *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. 2018 pp. 603–613.

[13] Higham, N.J. and Pranesh, S.: Simulating low precision floating-point arithmetic. SIAM Journal on Scientific Computing **41** (2019), C585–C602.

[14] Intel Corporation: Bfloat16 – hardware numerics definition. Tech. Rep. 338302-001US, Revision 1.0, Intel, 2018.

[15] Liesen, J. and Tichý, P.: The worst-case GMRES for normal matrices. BIT Numerical mathematics **44** (2004), 79–98.

[16] Lindquist, N., Luszczek, P., and Dongarra, J.: Accelerating restarted GMRES with mixed precision arithmetic. IEEE Transactions on Parallel and Distributed Systems **33** (2022), 1027–1037.

[17] Morgan, R.B.: GMRES with deflated restarting. SIAM Journal on Scientific Computing **24** (2002), 20–37.

[18] Oktay, E. and Carson, E.: Mixed precision GMRES-based iterative refinement with recycling. arXiv preprint arXiv:2201.09827 (2022).

[19] Oktay, E. and Carson, E.: Multistage mixed precision iterative refinement. Numerical Linear Algebra with Applications (2022), e2434.

[20] Parks, M.L., de Sturler, E., Mackey, G., Johnson, D.D., and Maiti, S.: Recycling Krylov subspaces for sequences of linear systems. SIAM Journal on Scientific Computing **28** (2006), 1651–1674.

[21] Soodhalter, K.M., de Sturler, E., and Kilmer, M.: A survey of subspace recycling iterative methods. arXiv preprint arXiv:2001.10347 (2020).

[22] Varah, J.: The prolate matrix. Linear Algebra and its Applications **187** (1993), 269–278.

[23] Wilkinson, J.H.: *Rounding errors in algebraic processes.* Prentice-Hall, 1963.