MANCHESTER
1824

The University of Manchester

Three-Precision GMRES-Based Iterative Refinement for Least Squares Problems

Carson, Erin and Higham, Nicholas J. and Pranesh, Srikara

2020

MIMS EPrint: **2020.5**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

THREE-PRECISION GMRES-BASED ITERATIVE REFINEMENT FOR LEAST SQUARES PROBLEMS*

ERIN CARSON†, NICHOLAS J. HIGHAM‡, AND SRIKARA PRANESH‡

Abstract. The standard iterative refinement procedure for improving an approximate solution to the least squares problem $\min_x \|b - Ax\|_2$, where $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ has full rank, is based on solving the $(m+n)\times(m+n)$ augmented system with the aid of a QR factorization. In order to exploit multiprecision arithmetic, iterative refinement can be formulated to use three precisions, but the resulting algorithm converges only for a limited range of problems. We build an iterative refinement algorithm called GMRES-LSIR, analogous to the GMRES-IR algorithm developed for linear systems [*SIAM J. Sci. Comput.*, 40 (2019), pp. A817-A847], that solves the augmented system using GMRES preconditioned by a matrix based on the computed QR factors. We explore two left preconditioners; the first has full off-diagonal blocks and the second is block diagonal and can be applied in either left-sided or split form. We prove that for a wide range of problems the first preconditioner yields backward and forward errors for the augmented system of order the working precision under suitable assumptions on the precisions and the problem conditioning. Our proof does not extend to the block diagonal preconditioner, but our numerical experiments show that with this preconditioner the algorithm performs about as well in practice.

Key words. least squares, iterative refinement, GMRES, preconditioning, mixed precision, half precision arithmetic

AMS subject classifications. 65F05, 65F08, 65F20

1. Introduction.

We consider the linear least squares problem $\min_x \|b - Ax\|_2$, where $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ has full rank. A common method of solution uses the QR factorization

$$A = Q\begin{bmatrix} R \\ 0 \end{bmatrix} \equiv QU,$$

where $Q = [Q_1, Q_2] \in \mathbb{R}^{m\times m}$ is an orthogonal matrix with $Q_1 \in \mathbb{R}^{m\times n}$ and $Q_2 \in \mathbb{R}^{m\times(m-n)}$, and $R \in \mathbb{R}^{n\times n}$ is upper triangular. The unique least squares solution is $x = R^{-1}Q_1^T b$ with residual $\|b - Ax\|_2 = \|Q_2^T b\|_2$. Least squares problems may be ill conditioned in practice, and so rounding errors may result in an insufficiently accurate solution. In this case, iterative refinement may be used to improve accuracy, and it also improves stability.

Two different approaches can be used for iterative refinement of least squares problems. When the overdetermined system is nearly compatible (i.e., there exists an x such that $Ax = b$ exactly or nearly exactly), an approach analogous to iterative refinement for square linear systems can be employed. After computing the initial approximate solution x_0, the solution is refined via the process whose $(i+1)$st step is

1. Compute $r_i = b - Ax_i$.
2. Solve the least squares problem $\min_{d_i} \|Ad_i - r_i\|_2$.
3. Update $x_{i+1} = x_i + d_i$.

If a QR factorization was used to compute the initial approximate solution x_0 then the

*Funding: The first author was supported by Charles University Primus program project PRIMUS/19/SCI/11. The second author was supported by the Royal Society. The second and third authors were supported by Engineering and Physical Sciences Research Council grant EP/P020720/1.

†Faculty of Mathematics and Physics, Charles University, 186 75, Praha 8, Czech Republic. (carson@karlin.mff.cuni.cz).

‡Department of Mathematics, University of Manchester, Manchester, M13 9PL, UK (nick.higham@manchester.ac.uk, srikara.pranesh@manchester.ac.uk).

QR factors can be reused to solve for the correction d_i in each step 2. This approach was used by Golub [10] and analyzed by Golub and Wilkinson [11].

A generalization of this approach that works even when $Ax = b$ is inconsistent was suggested by Björck [2]. Refinement is performed on the augmented system

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}, \tag{1.1}$$

which is equivalent to the normal equations. Given x_0 and $r_0 = b - Ax_0$, the $(i+1)$st refinement step is as follows.

1. Compute the residual vector for the augmented system:

$$\begin{bmatrix} f_i \\ g_i \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r_i \\ x_i \end{bmatrix} = \begin{bmatrix} b - r_i - Ax_i \\ -A^T r_i \end{bmatrix}. \tag{1.2}$$

2. Solve

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \delta r_i \\ \delta x_i \end{bmatrix} = \begin{bmatrix} f_i \\ g_i \end{bmatrix}. \tag{1.3}$$

3. Update the solution to the augmented system:

$$\begin{bmatrix} r_{i+1} \\ x_{i+1} \end{bmatrix} = \begin{bmatrix} r_i \\ x_i \end{bmatrix} + \begin{bmatrix} \delta r_i \\ \delta x_i \end{bmatrix}. \tag{1.4}$$

In this way, the solution x_i and residual r_i for the least squares problem are simultaneously refined. Björck [2] shows that the linear system (1.3) can be solved by reusing the QR factors of A via the process

$$h = R^{-T} g_i, \tag{1.5}$$

$$\begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = [Q_1, Q_2]^T f_i, \tag{1.6}$$

$$\delta r_i = Q \begin{bmatrix} h \\ d_2 \end{bmatrix}, \tag{1.7}$$

$$\delta x_i = R^{-1}(d_1 - h). \tag{1.8}$$

Existing analyses of the convergence and accuracy of this approach in finite precision assume that at most two precisions are used; the working precision u is used to compute the QR factorization, solve the system (1.3), and compute the update (1.4). A second precision $u_r \leq u$ is used to compute the residuals in (1.2). Typically $u_r = u^2$, in which case it can be shown that as long as the condition number of the augmented system matrix is smaller than u^{-1}, the refinement process will converge with a limiting forward error on the order of u; see [3] and [15, sect. 20.5] and the references therein.

Motivated by the emergence of multi-precision capabilities in hardware, Carson and Higham [5] have recently analyzed iterative refinement for (square) linear systems in the case where three different precisions are used: u_f for the matrix factorization, u for the working precision, and u_r for the computation of residuals, where $u_f \geq u \geq u_r$. The analysis additionally uses a fourth "precision", denoted u_s, which is essentially a parameter that describes how accurately the correction equation is solved (and u_s takes the value u or u_f in the cases of interest).

As the factorization of the system matrix is often the most expensive part of the computation, it is desirable from a performance standpoint that low precision be used in the factorization, ideally without affecting convergence, numerical stability, or accuracy. The results in [5] show that this is possible under certain constraints on the condition number of the matrix, which depend on the particular method of solving the correction equations. For example, with single precision as the working precision, the matrix factorization computed in half precision, residuals computed in double precision, and the correction equation solved by GMRES preconditioned with the computed LU factors, it is possible to solve the square linear system $Ax = b$ to full single precision accuracy for condition numbers $\kappa_\infty(A) = \|A^{-1}\|_\infty \|A\|_\infty \leq 10^7$, with the $O(n^3)$ part of the computations carried out entirely in half precision. Therefore significant speedups can be obtained on hardware that supports half precision. In [12], [13], using the tensor cores on an NVIDIA V100 GPU, Haidar et al. obtain a speedup of 4 over the state of the art double precision solver. In this work we wish to extend the results of [5] to least squares problems. In [17] a mixed-precision least square solver based on the normal equations was proposed for the case where A is well conditioned. The present work differs in that it is not based on the normal equations and it is applicable to a wider range of problems. This work also differs from that in [9], which focuses on the use of higher precision arithmetic and does not use a lower precision than the working precision.

Define

$$\widetilde{A} = \begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix}, \quad \widetilde{b} = \begin{bmatrix} b \\ 0 \end{bmatrix}, \quad y_i = \begin{bmatrix} r_i \\ x_i \end{bmatrix}, \quad \delta y_i = \begin{bmatrix} \delta r_i \\ \delta x_i \end{bmatrix}, \quad s_i = \begin{bmatrix} f_i \\ g_i \end{bmatrix}. \tag{1.9}$$

Three-precision iterative refinement based on the augmented system is written in Algorithm 1.1 in an analogous way as for linear systems.

Algorithm 1.1 (LSIR) Iterative refinement in three precisions for the augmented system defined in (1.9).

1: Solve $\widetilde{A}y_0 = \widetilde{b}$ in precision u_f and store y_0 at precision u.
2: **for** $i = 0 : \infty$ **do**
3: Compute $s_i = \widetilde{b} - \widetilde{A}y_i$ at precision u_r and round s_i to precision u_s.
4: Solve $\widetilde{A}\delta y_i = s_i$ at precision u_s and store δy_i at precision u.
5: $y_{i+1} = y_i + \delta y_i$ at precision u.
6: **end for**

The theorems developed in [5] regarding the forward error and normwise and componentwise backward error for iterative refinement of linear systems are thus applicable. The only thing that will change is the analysis of the method for solving the correction equation in line 4, since we now have a QR factorization of A, which can be used in various ways, including the procedure (which we will refer to as the "standard" method) outlined in (1.5)–(1.8). For a given solver, in order to apply the analysis from [5] we need to show that

$$\widehat{\delta y_i} = (I + u_s E_i)\delta y_i, \quad u_s \|E_i\|_\infty < 1, \tag{1.10}$$

$$\|\widehat{s}_i - \widetilde{A}\,\widehat{\delta y_i}\|_\infty \leq u_s(c_1\|\widetilde{A}\|_\infty \|\widehat{\delta y_i}\|_\infty + c_2\|\widehat{s}_i\|_\infty), \tag{1.11}$$

$$|\widehat{s}_i - \widetilde{A}\widehat{\delta y_i}| \leq u_s G_i |\widehat{\delta y_i}|, \tag{1.12}$$

TABLE 1.1

For Algorithm 1.1, under assumptions (1.10)–(1.12), conditions for convergence and the limiting size of the forward error, normwise backward error, and componentwise backward error for the solution of the augmented system $\widetilde{A}y = \widetilde{b}$ (1.1). The quantity p denotes the maximum number of nonzeros per row of $[\widetilde{A}, y]$ and $\mu_i \equiv \|\widetilde{A}(y - \widehat{y}_i)\|_\infty / \|\widetilde{A}\|_\infty \|y - \widehat{y}_i\|_\infty$; for further explanation see [4, sect. 2.1].

Error	Convergence condition	Bound on limiting value
Forward	$2u_s \min(\mathrm{cond}(\widetilde{A}), \kappa_\infty(\widetilde{A})\mu_i)$ $+ u_s\|E_i\|_\infty < 1$	$4pu_r \,\mathrm{cond}(\widetilde{A}, x) + u$
Normwise backward	$(c_1\kappa_\infty(\widetilde{A}) + c_2)u_s < 1$	pu
Componentwise backward	$u_s\|\,G_i\|\widetilde{A}^{-1}|\,\|_\infty + (1 + u_s)$ $\times \gamma_p^r \,\mathrm{cond}(\widetilde{A}^{-1}) \le 1/2$	$\gamma_p^r(1 + u_s\|G_i\|\widetilde{A}^{-1}|\|_\infty + \gamma_p^r \,\mathrm{cond}(\widetilde{A}^{-1}))$ $\times \xi(\|\widetilde{b}\| + \|\widetilde{A}\|\|y\|) + u$

where hats denote quantities computed in finite precision and E_i, c_1, c_2, and G_i are functions of $m + n$, $\widetilde{A}$, $\widehat{s}_i$, and u_s, and have nonnegative entries. Here, and throughout, inequalities between vectors and matrices hold componentwise. Given these bounds for a particular solution method, the convergence conditions and bounds on limiting values of various errors for three-precision iterative refinement for linear systems proved in [5] yield the convergence conditions and bounds for Algorithm 1.1 shown in Table 1.1, which is based on [5, Table 5.1]. Within Table 1.1, $\mathrm{cond}(\widetilde{A}) = \||\widetilde{A}^{-1}||\widetilde{A}|\|_\infty$ denotes the Skeel condition number, and $\xi(x) = \max_j |x_j| / \min_j |x_j|$ where x_j is the jth component of the vector x. In Table 1.1 and in remainder of the paper we use the quantities

$$\gamma_k = \frac{ku}{1 - ku}, \quad \tilde{\gamma}_k = \frac{cku}{1 - cku},$$

where c is a small constant independent of m and n. A superscript on γ will denote that u carries that superscript as a subscript; thus, for example, $\gamma_k^r = ku_r/(1 - ku_r)$. The quantity $\mu_i \equiv \|\widetilde{A}(y - \widehat{y}_i)\|_\infty / (\|\widetilde{A}\|_\infty \|y - \widehat{y}_i\|_\infty)$ that appears in the forward error convergence criterion in Table 1.1 describes the tightness of the inequality $\|\widetilde{A}(y - \widehat{y}_i)\|_\infty \le \|\widetilde{A}\|_\infty \|y - \widehat{y}_i\|_\infty$. As explained in [4, sect. 2.1], we expect μ_i to be close to its lower bound of $\kappa_\infty(\widetilde{A})^{-1}$ near the beginning of the iterations and to only grow close to its upper bound of 1 once the forward error becomes small. We note that depending on the particular solver that is used for the correction equation, the term with μ_i or the term with $\|E_i\|_\infty$ can dominate.

Our contributions in this work are as follows.

- We show that (1.10)–(1.12) hold for Algorithm 1.1 and determine bounds for $\|E_i\|_\infty$, c_1, c_2, and $\|G_i\|_\infty$.
- We extend the GMRES-based refinement scheme of [4] to the least squares case and show that one can construct a left preconditioner using the existing QR factors of A such that GMRES provably converges to a backward stable solution of the preconditioned augmented system.
- We show that an existing preconditioner developed for saddle point systems can also work well in the GMRES-based approach in practice, even though our error analysis is not applicable.

In section 2, we analyze the normwise relative error, normwise relative backward error, and componentwise relative backward error for the QR-based correction solve for the augmented system and write these bounds in the forms given by (1.10)–(1.12). In section 3, we discuss a technique that uses the already computed QR factors in a

preconditioned Krylov subspace method for solving the correction equations, which extends the range of matrix condition numbers for which lower precision factorizations can be used. This is analogous to the GMRES-IR technique presented in [4]. In section 4 we present numerical experiments, and we give conclusions in section 5.

2. Least squares iterative refinement in three precisions. We will assume that the QR factorization is computed via the Householder method, carried out in precision u_f. By [7, Thm. 1.1], [15, Thm. 19.4] if $\widehat{U}$ is the computed upper trapezoidal factor of A obtained via Householder QR, then there exists an orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ such that

$$A + E = Q\widehat{U} = Q_1\widehat{R}, \quad \|E\|_F \le \tilde{\gamma}^f_{mn}\|A\|_F, \quad |E| \le m\tilde{\gamma}^f_{mn}C_1|A|, \tag{2.1}$$

with $C_1 \ge 0$ and $\|C_1\|_F = 1$, where $Q_1 = Q(:, 1\colon n)$ and $R = \widehat{U}(1\colon n, 1\colon n)$. For the *computed* factor $\widehat{Q}$, we have [15, p. 360]

$$\widehat{Q} = Q(I_m + \Delta_I) \equiv Q + \Delta_Q, \tag{2.2}$$

where $\Delta_Q = Q\Delta_I$, with $\|\Delta_I(:, j)\|_2 \le \tilde{\gamma}^f_{mn}$. Therefore $\|\Delta_Q(:, j)\|_2 = \|Q\Delta_I(:, j)\|_2 \le \tilde{\gamma}^f_{mn}$, and thus by [15, Lem. 6.6],

$$\|\Delta_Q\|_F \le \sqrt{m}\tilde{\gamma}^f_{mn}, \quad |\Delta_Q| \le m\tilde{\gamma}^f_{mn}C_2, \tag{2.3}$$

where $C_2 \ge 0$ and $\|C_2\|_F = 1$.

We first state a result showing that if the system (1.3) is solved via equations (1.5)–(1.8) in precision $u_f \ge u$ using a QR factorization computed in precision u_f then the effective solve precision is $u_s = u_f$.

THEOREM 2.1. *Let the system (1.3) be solved via the process in (1.5)–(1.8) carried out in working precision u, with the QR factorization of A computed in precision $u_f \ge u$ such that (2.1)–(2.3) hold. Then*

$$\begin{bmatrix} I & A + \Delta A_1 \\ (A + \Delta A_2)^T & 0 \end{bmatrix} \begin{bmatrix} \widehat{\delta r}_i \\ \widehat{\delta x}_i \end{bmatrix} = \begin{bmatrix} f_i + \Delta f_i \\ g_i + \Delta g_i \end{bmatrix}, \tag{2.4}$$

with

$$\begin{aligned} |\Delta f_i| &\le \sqrt{mn}(\tilde{\gamma}^f_{m^2} + \tilde{\gamma}_{m^2})(H_1|f_i| + H_2|\widehat{\delta r}_i|), \\ |\Delta g_i| &\le \sqrt{mn}(\tilde{\gamma}^f_{m^2} + \tilde{\gamma}_{m^2})(|A^T|H_3|\widehat{\delta r}_i|), \\ |\Delta A_j| &\le mn(\tilde{\gamma}^f_m H_4 + \tilde{\gamma}_m H_5)|A|, \quad j = 1, 2, \end{aligned}$$

where $\|H_j\|_F = 1$, $H_j \ge 0$ for $j = 1 : 5$.

Proof. The proof follows closely the analysis in [2] and [14] and is omitted. ∎

This theorem essentially says that the backward error in the correction solve using the standard approach is limited by the precision in which the QR factorization is computed. Following the argument in [15, sect. 20.5] and using the assumption that $u_f \ge u$, we can obtain an expression in which only the matrix is perturbed, giving

$$\left(\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} + \Delta\widetilde{A} \right) \begin{bmatrix} \widehat{\delta r}_i \\ \widehat{\delta x}_i \end{bmatrix} = \begin{bmatrix} f_i \\ g_i \end{bmatrix}, \quad \|\Delta\widetilde{A}\|_2 \le c_{m,n}u_f \left\| \begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \right\|_2,$$

where $c_{m,n}$ is a constant that depends on m and n. Changing notation as described in (1.9), we can write the error in the correction solve in each step of least squares iterative refinement as

$$(\widetilde{A} + \Delta\widetilde{A})\widehat{\delta y}_i = s_i, \quad \|\Delta\widetilde{A}\|_\infty \le \widetilde{c}_{m,n} u_f \|\widetilde{A}\|_\infty.$$

Thus

$$\widehat{\delta y}_i = (I - \widetilde{A}^{-1}\Delta\widetilde{A})\delta y_i \equiv (I + u_s E_i)\delta y_i,$$

so in (1.10), we can take

$$u_s \|E_i\|_\infty = \widetilde{c}_{m,n} u_f \kappa_\infty(\widetilde{A}).$$

Thus for the forward error, the $u_s\|E_i\|_\infty$ term will dominate the criterion for convergence (see the convergence condition in Table 1.1), and we can say that as long as $\kappa_\infty(\widetilde{A}) \lesssim u_f^{-1}$, we expect the method to converge to achieve the limiting relative forward error

$$\frac{\|y - \widehat{y}\|_\infty}{\|y\|_\infty} \lesssim 4pu_r \operatorname{cond}(\widetilde{A}, y) + u,$$

where p is the maximum number of nonzeros per row of $[\widetilde{A}, y]$. It is also clear that we can take $u_s \max(c_1, c_2) = O(u_f)$ in (1.11) and $u_s\|G_i\|_\infty = O(u_f\|\widetilde{A}\|_\infty)$ in (1.12), so again, we expect the normwise and componentwise backward errors for the augmented system to be of order u as long as $\kappa_\infty(\widetilde{A}) \lesssim u_f^{-1}$. (The backward error for the least squares problem is not bounded by our analysis; see [15, secs 20.5, 20.8] for details of this backward error.)

It is important to comment on the condition number of the augmented system $\widetilde{A}$ and how it is related to the condition number of A. Björck [2] shows that the matrix that results from scaling by a parameter α,

$$\widetilde{A}_\alpha \equiv \begin{bmatrix} \alpha I & A \\ A^T & 0 \end{bmatrix},$$

has condition number bounded by

$$\sqrt{2}\kappa_2(A) \le \min_\alpha \kappa_2(\widetilde{A}_\alpha) \le 2\kappa_2(A), \quad \max_\alpha \kappa_2(\widetilde{A}_\alpha) > \kappa_2(A)^2, \tag{2.5}$$

where $\min_\alpha \kappa_2(\widetilde{A}_\alpha)$ is attained for the choice

$$\alpha = 2^{-1/2}\sigma_{\min}(A). \tag{2.6}$$

This scaling is equivalent to $b - Ax \leftarrow (b - Ax)/\alpha$, which does not change the solution to the problem, so we can assume in the analysis that $\kappa_2(\widetilde{A})$ is the same order of magnitude as $\kappa_2(A)$. We can further make the simplifying assumption that α is a power of the machine base and thus does not affect the rounding errors. With this scaling, the correction equation to solve in each step of iterative refinement becomes

$$\begin{bmatrix} \alpha I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \widehat{\delta r}_i \\ \alpha\widehat{\delta x}_i \end{bmatrix} = \begin{bmatrix} \alpha f_i \\ g_i \end{bmatrix}. \tag{2.7}$$

In the remainder of the paper we will let $\widetilde{A}$ denote the scaled system (i.e., $\widetilde{A} \leftarrow \widetilde{A}_\alpha$), with α as in (2.6), to simplify the notation.

TABLE 2.1

Summary of results for various choices of IEEE standard precisions for standard least squares iterative refinement. The value of $\kappa_\infty(A)$ shown is the maximum for which the indicated limiting forward errors and backward errors (for the augmented system) are guaranteed to hold.

				Backward error		
u_f	u	u_r	$\kappa_\infty(A)$	Normwise	Componentwise	Forward error
half	single	single	$2 \cdot 10^3$	single	single	$\mathrm{cond}(\widetilde{A}, y) \cdot 10^{-7}$
half	single	double	$2 \cdot 10^3$	single	single	single
half	double	double	$2 \cdot 10^3$	double	double	$\mathrm{cond}(\widetilde{A}, y) \cdot 10^{-16}$
half	double	quad	$2 \cdot 10^3$	double	double	double
single	single	single	10^7	single	single	$\mathrm{cond}(\widetilde{A}, y) \cdot 10^{-7}$
single	single	double	10^7	single	single	single
single	double	double	10^7	double	double	$\mathrm{cond}(\widetilde{A}, y) \cdot 10^{-16}$
single	double	quad	10^7	double	double	double

TABLE 2.2

Parameters for several floating-point arithmetics to three significant figures: unit roundoff u, smallest positive (subnormal) number $x^s_{\min}$, smallest normalized positive number $x_{\min}$, and largest finite number $x_{\max}$. In Intel's bfloat16 specification subnormal numbers are not supported [21].

	u	$x^s_{\min}$	$x_{\min}$	$x_{\max}$
bfloat16	3.91×10^{-3}	9.18×10^{-41}	1.18×10^{-38}	3.39×10^{38}
fp16	4.88×10^{-4}	5.96×10^{-8}	6.10×10^{-5}	6.55×10^{4}
fp32	5.96×10^{-8}	1.40×10^{-45}	1.18×10^{-38}	3.40×10^{38}
fp64	1.11×10^{-16}	4.94×10^{-324}	2.22×10^{-308}	1.80×10^{308}
fp128	9.63×10^{-35}	6.48×10^{-4966}	3.36×10^{-4932}	1.19×10^{4932}

We summarize the implications of the analysis in Table 2.1, which gives the limiting forward error, normwise, and componentwise backward error, and the maximum $\kappa_\infty(A)$ (with which we are approximating $\kappa_\infty(\widetilde{A})$, for the reasons explained in the previous paragraph) for which these bounds apply for various combinations of precisions (u_f, u, u_r). For example, we expect that "standard" iterative refinement for least squares problems using half precision for the QR factorization, single precision as the working precision, and double precision for the residual computation, will obtain forward and backward errors to single precision accuracy for matrices A with condition number less than u_f^{-1}. The value of the unit roundoff parameter for various precisions can be found in Table 2.2.

3. GMRES-based least squares iterative refinement. The analysis in [4] (and in the subsequent work [5]) shows that iterative refinement for linear systems can converge even in cases where the condition number of the matrix exceeds u_f^{-1}, as long as the corrections are computed with some degree of relative accuracy. The standard approach based on LU factorization, in which the computed LU factors are reused to solve the correction equation, will not provide any relative accuracy for matrix condition numbers larger than u_f^{-1}. Thus for linear systems that are very ill conditioned relative to the factorization precision, a different solver is needed.

In [4], [5] it was shown that the GMRES method with the computed LU factors (potentially computed in lower precision) used as left preconditioners can provide the required relative accuracy in the computed solution of the correction equation. This idea was motivated by the observation that even for ill-conditioned matrices the computed LU factors contain useful information and thus can still be effective

preconditioners in terms of reducing the condition number.

One option is to apply this GMRES-based iterative refinement—referred to as GMRES-IR—to the augmented system. However, computing the LU factorization of $\widetilde{A}$ incurs $O((m+n)^3)$ operations as opposed to $O(mn^2)$ operations for the QR factorization of A, with potentially also greater interprocessor communication requirements. At scale, this is an unattractive approach, especially if we already have an existing QR factorization of A.

In the next section we show that it is possible to instead construct an effective left preconditioner M for $\widetilde{A}$ from the already computed QR factors of A. As a side-effect, our analysis demonstrates how the analysis of GMRES-IR in [4] and [5] can be extended to allow for a general choice of preconditioner (i.e., one not necessarily based on an LU factorization).

3.1. A preconditioner from computed QR factors. We wish to construct a preconditioner for which the resulting preconditioned matrix has a sufficiently small condition number. This will allow us to prove that preconditioned GMRES will find a sufficiently accurate solution. We also require, for analysis purposes, that left preconditioning is used. This is because right-preconditioned (and thus split-preconditioned) GMRES is not backward stable, although we note that split preconditioning may nevertheless work well in practice. We elaborate on this in section 3.2.

For a given preconditioner M, we write $\widetilde{E} = \widetilde{A} - M$. We have

$$M^{-1}\widetilde{A} = M^{-1}(M + \widetilde{E}) = I + M^{-1}\widetilde{E},$$
$$\widetilde{A}^{-1}M = (M + \widetilde{E})^{-1}M \approx I - M^{-1}\widetilde{E},$$

and therefore

$$\kappa_\infty(M^{-1}\widetilde{A}) = \|M^{-1}\widetilde{A}\|_\infty \|\widetilde{A}^{-1}M\|_\infty \lesssim (1 + \|M^{-1}\widetilde{E}\|_\infty)^2. \tag{3.1}$$

We now give results for the particular preconditioner

$$M = \begin{bmatrix} \alpha I & Q_1\widehat{R} \\ \widehat{R}^T Q_1^T & 0 \end{bmatrix}, \tag{3.2}$$

which will meet our stated requirements. In order to simplify the analysis, we have used Q_1 rather than $\widehat{Q}_1$ in the definition above; i.e., we assume that $\widehat{Q}_1$ has orthonormal columns. This is equivalent to ignoring terms in u_f^2 and will not affect the conclusions of the analysis. We have

$$M^{-1} = \begin{bmatrix} \frac{1}{\alpha}(I - Q_1Q_1^T) & Q_1\widehat{R}^{-T} \\ \widehat{R}^{-1}Q_1^T & -\alpha\widehat{R}^{-1}\widehat{R}^{-T} \end{bmatrix} \tag{3.3}$$

and by (2.1),

$$\widetilde{E} = \widetilde{A} - M = \begin{bmatrix} 0 & -E \\ -E^T & 0 \end{bmatrix}. \tag{3.4}$$

Then

$$M^{-1}\widetilde{E} = \begin{bmatrix} -Q_1\widehat{R}^{-T}E^T & -\frac{1}{\alpha}(I - Q_1Q_1^T)E \\ \alpha\widehat{R}^{-1}\widehat{R}^{-T}E^T & -\widehat{R}^{-1}Q_1^TE \end{bmatrix}. \tag{3.5}$$

Now we seek to bound $\|M^{-1}\widetilde{E}\|_\infty$. We have

$$\begin{aligned}\|M^{-1}\widetilde{E}\|_\infty &\le \max\Big(\|M^{-1}\widetilde{E}(1,1)\|_\infty + \|M^{-1}\widetilde{E}(1,2)\|_\infty, \|M^{-1}\widetilde{E}(2,1)\|_\infty \\ &\qquad + \|M^{-1}\widetilde{E}(2,2)\|_\infty\Big) \\ &\le \sqrt{m}\max\left(\|M^{-1}\widetilde{E}(1,1)\|_F, \|M^{-1}\widetilde{E}(2,1)\|_F\right) \\ &\quad + \sqrt{n}\max\left(\|M^{-1}\widetilde{E}(1,2)\|_F, \|M^{-1}\widetilde{E}(2,2)\|_F\right), \end{aligned} \tag{3.6}$$

where $M^{-1}\widetilde{E}(i,j)$ denotes the (i,j) block of $M^{-1}\widetilde{E}$ as in (3.5). Bounding the individual blocks of $M^{-1}\widetilde{E}$, using $|\alpha| \approx \|A^+\|_2^{-1}$, and again ignoring terms of order u_f^2, we have

$$\begin{aligned}\|\frac{1}{\alpha}(I - Q_1Q_1^T)E\|_F &\le \frac{1}{|\alpha|}\|I - Q_1Q_1^T\|_2\|E\|_F \lesssim \tilde{\gamma}_{mn}^f\|A^+\|_F\|A\|_F, \\ \|\widehat{R}^{-1}Q_1^TE\|_F &\le \|\widehat{R}^{-1}Q_1^T\|_F\|E\|_F \lesssim \tilde{\gamma}_{mn}^f\|A^+\|_F\|A\|_F, \\ \|Q_1\widehat{R}^{-T}E^T\|_F &\le \|Q_1\widehat{R}^{-T}\|_F\|E^T\|_F \lesssim \tilde{\gamma}_{mn}^f\|A^+\|_F\|A\|_F, \\ \|\alpha\widehat{R}^{-1}\widehat{R}^{-T}E^T\|_2 &\le |\alpha|\|\widehat{R}^{-1}\widehat{R}^{-T}\|_F\|E^T\|_F \lesssim \tilde{\gamma}_{mn}^f\|A^+\|_F\|A\|_F.\end{aligned}$$

Therefore, using (3.6),

$$\begin{aligned}\|M^{-1}\widetilde{E}\|_\infty &\lesssim (\sqrt{m}+\sqrt{n})\tilde{\gamma}_{mn}^f\|A^+\|_2\|A\|_F \le \sqrt{m}\sqrt{n}(\sqrt{m}+\sqrt{n})\tilde{\gamma}_{mn}^f\|A^+\|_\infty\|A\|_\infty \\ &\le 2m\sqrt{n}\tilde{\gamma}_{mn}^f\kappa_\infty(A),\end{aligned}$$

and hence from (3.1) we have

$$\kappa_\infty(M^{-1}\widetilde{A}) \lesssim (1 + 2m\sqrt{n}\tilde{\gamma}_{mn}^f\kappa_\infty(A))^2, \tag{3.7}$$

where we note that this inequality is pessimistic. Thus even in cases where $\kappa_\infty(A) \gg u_f^{-1}$, we still expect the preconditioner M composed from the QR factors computed in precision u_f to reduce the condition number of $\widetilde{A}$. By the result proved in [4, sect. 3], if $u_r = u^2$ then GMRES run on $\widetilde{A}$ with left preconditioner M will result in

$$u_s\|E_i\|_\infty \equiv uf(m+n)\kappa_\infty(M^{-1}\widetilde{A}), \tag{3.8}$$

where $f(m+n)$ is a quadratic polynomial. Thus for the GMRES-based solver, we achieve an error of order $u_s = u$ in (1.10). Combining the above with (3.7), we expect to have $u_s\|E_i\|_\infty < 1$ (and thus convergence of the forward error in GMRES-based iterative refinement) when $\kappa_\infty(A) < u^{-1/2}u_f^{-1}$. The case where $u_r = u$ is treated in [16, sect. 3.1]. In this case, the condition for convergence of the backward error becomes $\kappa_\infty(A) < u^{-1/2}$ and the condition for the convergence of the forward error becomes $\kappa_\infty(A) < u^{-1/3}u_f^{-2/3}$. We note, as is stated in [16], that these results are pessimistic. These results are summarized for various combinations of IEEE standard precisions in Table 3.1; note that the table exploits the fact that $\kappa_\infty(\widetilde{A})$ can be replaced by $\kappa_\infty(A)$ (up to a constant that includes the change of norm) in view of (2.5). Note also that this table is very similar to that for iterative refinement of square linear systems in [16, Table 3.1]. Again, the value of the unit roundoff parameter for various precisions can be found in Table 2.2.

TABLE 3.1

Summary of results for various choices of IEEE standard precisions for GMRES-based least squares iterative refinement. The value of $\kappa_\infty(A)$ shown in the fourth column is the maximum value for which the indicated limiting backward errors (for the augmented system) and forward errors, respectively, are guaranteed to hold.

				Backward error			
u_f	u	u_r	$\kappa_\infty(A)$	Norm.	Comp.	$\kappa_\infty(A)$	Forward error
half	half	single	$2 \cdot 10^3$	half	half	10^5	half
half	single	single	10^3	single	single	10^4	$\mathrm{cond}(\widetilde{A}, y) \cdot 10^{-7}$
half	single	double	10^7	single	single	10^7	single
half	double	double	10^6	double	double	10^7	$\mathrm{cond}(\widetilde{A}, y) \cdot 10^{-16}$
half	double	quad	10^{16}	double	double	10^{11}	double
single	single	double	10^7	single	single	10^{11}	single
single	double	double	10^7	double	double	10^{10}	$\mathrm{cond}(\widetilde{A}, y) \cdot 10^{-16}$
single	double	quad	10^{16}	double	double	10^{15}	double

In Figure 3.1 we plot the infinity norm condition numbers of $\widetilde{A}$ and $M^{-1}\widetilde{A}$, for M defined in (3.2) with the QR factorization performed in half precision and α set to the optimal value $2^{-1/2}\sigma_{\min}(A)$. We also plot an approximation of the bound (3.7) with the dimensional constants ignored, as well as the value u^{-1} for single precision. By (3.8), when $\kappa_\infty(M^{-1}\widetilde{A})$ is approximately below this level, we expect convergence of the forward error. The computations (and all computations in the remainder of the paper) were performed in MATLAB R2019a. To compute a QR factorization of A in half precision we use a MATLAB implementation of Householder QR factorization that calls the `chop` function of Higham and Pranesh [18]. The preconditioned system $M^{-1}\widetilde{A}$ was computed in quadruple precision arithmetic using the Advanpix Multiprecision Computing Toolbox [23] and then rounded to the working precision. The matrices were generated as `gallery('randsvd',[100,10],kappa(i),3)`, which constructs a random 100×10 matrix with specified 2-norm condition number `kappa(i)` and geometrically distributed singular values. We tested `kappa` values 10^j, $j = 0\colon 10$.

From Figure 3.1, we can see that indeed, the condition number of $M^{-1}\widetilde{A}$ grows as expected with the bound (3.7), and that the preconditioned matrix has a smaller condition number than $\widetilde{A}$ until around $\kappa_\infty(A) \approx 10^7$. The horizontal line plotting the quantity u^{-1} indicates the maximum condition number allowable such that $u_s\|E_i\|_\infty$ in (3.8) is less than 1, meaning that the iterative refinement process is guaranteed to converge. As mentioned, the bound (3.7), which intersects the horizontal line at $u^{-1/2}u_f^{-1}$, is a slight overestimate; the value of $u_s\|E_i\|_\infty$ can actually remain below 1 for even higher values of $\kappa_\infty(A)$.

Again, for the backward errors the results in [4] show that when the matrix–vector products with the preconditioned matrix are computed at precision $u_r = u^2$, the original correction equation $\widetilde{A}\delta y_i = s_i$ is solved with backward error of order u, and thus c_1 and c_2 in (1.11) can be taken to be of order 1 and G_i in (1.12) can be taken to have norm of order $\|\widetilde{A}\|_\infty$.

3.2. Practical considerations and other preconditioners. Table 2.2 displays the properties of some floating-point formats currently available in hardware, where IEEE formats are denoted "fp" [20]. Owing to the narrow range of the half precision fp16 format, underflow and overflow can readily arise in practice. To address this issue in the context of solving linear systems, a two-sided diagonal scaling-based algorithm was proposed in [19]. In our context of QR factorization we propose one-

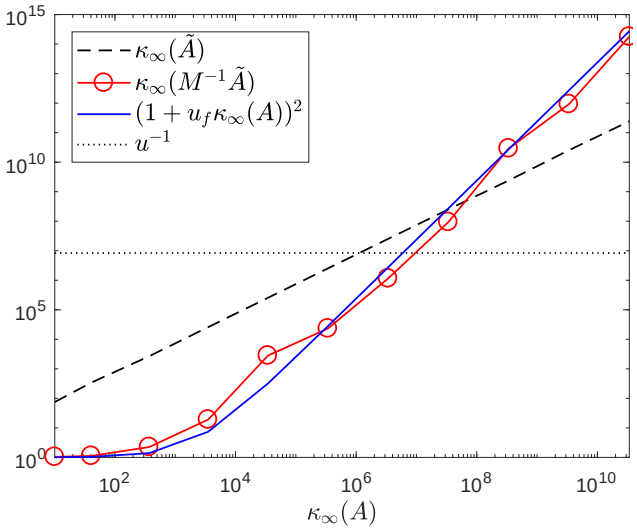

FIG. 3.1. *Condition numbers for augmented matrix $\widetilde{A}$ and preconditioned augmented matrix $M^{-1}\widetilde{A}$ for M in (3.2), for 100×10 matrices A described in the text. The solid blue line shows the approximate bound on $\kappa_\infty(M^{-1}\widetilde{A})$ given in (3.7) without constants.*

sided diagonal scaling on the right, which is summarized in Algorithm 3.1, in which $\mathrm{fl_h}$ denotes rounding to fp16. Note that the scaled matrix AS has columns of unit 2-norm. The purpose of the scaling by μ is to make full use of the dynamic range while allowing some headroom for subsequent computations. Note that this column scaling strategy is not required for bfloat16 because of its much larger range. We are not concerned here with the implementation details of QR factorization in fp16; see [25] for an implementation targeted at GPU tensor cores.

Algorithm 3.1 (fp16 QR factorization). This algorithm rounds $A \in \mathbb{R}^{m\times n}$ given in precision u to the fp16 matrix $A^{(h)}$, scaling all elements to avoid overflow, and then performs the QR factorization in fp16. $\theta_{\max} \in (0,1]$ is a parameter.

1: $S = \mathrm{diag}\big(\|A(:,j)\|_\infty^{-1}\big)$
2: $\mu = \theta_{\max} x_{\max}$
3: $A^{(h)} = \mathrm{fl_h}(\mu(AS))$
4: Compute the QR factorization $A^{(h)} = \widehat{Q}\widehat{R}$.

There is a wealth of work towards developing preconditioners for saddle-point systems. Common preconditioners that can be constructed using the QR factors of A include block diagonal preconditioners, triangular preconditioners, and constraint-based preconditioners. For example, we could construct the block diagonal preconditioner

$$\begin{bmatrix} \alpha I & 0 \\ 0 & \frac{1}{\alpha}\widehat{R}^T\widehat{R} \end{bmatrix} = \begin{bmatrix} \sqrt{\alpha} I & 0 \\ 0 & \frac{1}{\sqrt{\alpha}}\widehat{R}^T \end{bmatrix} \begin{bmatrix} \sqrt{\alpha} I & 0 \\ 0 & \frac{1}{\sqrt{\alpha}}\widehat{R} \end{bmatrix} \equiv M_1 M_2. \tag{3.9}$$

This block diagonal preconditioner may have an advantage over the preconditioner (3.2) in terms of the communication cost of applying it to a vector; this will depend on the relative sizes of m and n and the particular parallel distribution of the QR factors.

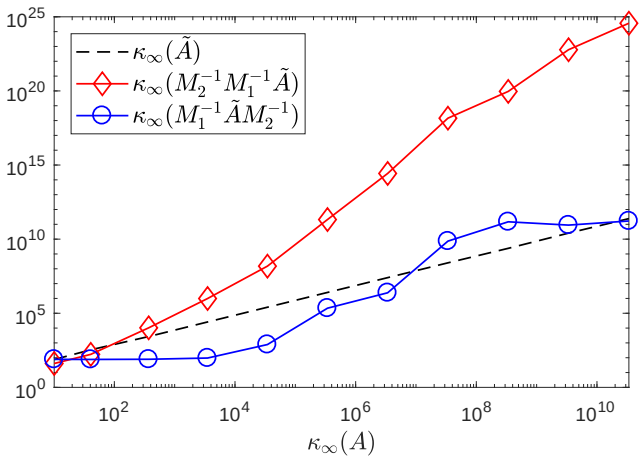

FIG. 3.2. *Condition numbers for augmented matrix $\widetilde{A}$ and left, split, preconditioned matrices $M_2^{-1}M_1^{-1}\widetilde{A}$, $M_1^{-1}\widetilde{A}M_2^{-1}$ respectively, where M_1 and M_2 are defined in (3.9).*

It follows from the results in [22], [24] that if the QR factorization is computed exactly then the left-preconditioned system matrix

$$M_2^{-1}M_1^{-1}\widetilde{A} = \begin{bmatrix} I & A/\alpha \\ \alpha\widehat{R}^{-1}\widehat{R}^{-T}A^T & 0 \end{bmatrix}$$

will be nonsymmetric and diagonalizable with three distinct nonzero eigenvalues, $\{1, \frac{1}{2}(1\pm\sqrt{5})\}$. However, as the resulting left-preconditioned matrix is nonsymmetric, this tells us nothing about the singular values. We have observed experimentally that the condition number of $M_2^{-1}M_1^{-1}\widetilde{A}$ can be very large when A is ill-conditioned (often larger than the condition number of $\widetilde{A}$ itself), making this preconditioner unsuitable for our purpose of proving that GMRES provides a backward stable solution to the left-preconditioned system.

The preconditioner in (3.9) can also be used as a two-sided preconditioner. Accordingly we obtain

$$M_1^{-1}\widetilde{A}M_2^{-1} = \begin{bmatrix} I & A\widehat{R} \\ \widehat{R}^{-T}A^T & 0 \end{bmatrix}.$$

In this case, the preconditioned system is symmetric, so the absolute values of the eigenvalues (which, again, are $\{1, \frac{1}{2}(1\pm\sqrt{5})\}$) coincide with the singular values, and in contrast to the left-preconditioned case we are guaranteed a well-conditioned system, at least in exact arithmetic. We confirm this behavior numerically in Figure 3.2; the computational details are as for Figure 3.1, which plots $\kappa_\infty(M_2^{-1}M_1^{-1}\widetilde{A})$ and $\kappa_\infty(M_1^{-1}\widetilde{A}M_2^{-1})$, for M_1 and M_2 defined in (3.9).

It is clear from the plot that the preconditioned matrix has a much lower condition number for split preconditioning than for left preconditioning. In fact, the condition number of the left preconditioned matrix quickly becomes much larger than $\kappa_\infty(\widetilde{A})$! Thus the left preconditioned system is too ill conditioned to ensure that $u_s\|E_i\|_\infty < 1$ in (1.10). Despite the improved condition number of the split preconditioned system, we cannot use it for the theoretical purpose of proving the backward stability of

GMRES; in contrast to the left-preconditioned case (see [4]), right-preconditioned (and thus split-preconditioned) GMRES is not backward stable [1, pp. 2035]. We note that our desire for a well-conditioned preconditioned system comes from meeting the constraint $u_s\|E_i\|_\infty < 1$ in (1.10); the condition number alone does not dictate the convergence behavior of Krylov subspace methods even in the case of symmetric positive definite linear systems; see, e.g., [6].

We emphasize that the GMRES method and the preconditioner (3.2) were chosen so that we could prove that GMRES-based iterative refinement converges. In practical applications one might use any preconditioning technique or any other Krylov subspace method. As a demonstration, in section 4, we show convergence of GMRES-based iterative refinement with the use of both the left preconditioner defined in (3.2) and the split block diagonal preconditioner (3.9). We note that since the matrix $\widetilde{A}$ is symmetric, it makes sense to use a Krylov subspace method designed for symmetric linear systems (such as MINRES) in practice. Higham and Pranesh [17] show that such other techniques can work for symmetric positive definite linear systems, although we cannot say that Krylov subspace methods for symmetric linear systems such as conjugate gradient and MINRES are backward stable in the traditional sense.

Based on these practical considerations GMRES-based least squares iterative refinement method is summarized in Algorithm 3.2.

Algorithm 3.2 (GMRES-LSIR) Let $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ and $b \in \mathbb{R}^m$ be given in precision u. This algorithm solves the least squares problem $\min_x \|b - Ax\|_2$ using GMRES-based iterative refinement with precisions u_f, u, and u_r. The quantities $\widetilde{A}$, $\widetilde{b}$, s_i, and y_i are defined in (1.9).

if u_f is fp16 **then**
  Compute the QR factors, diagonal matrix S and scalar μ using Algorithm 3.1, such that $A \approx (1/\mu)\widehat{Q}_1\widehat{R}S^{-1}$.
else
  Compute the QR factorization $A = \widehat{Q}_1\widehat{R}$ in precision u_f.
end if
Solve $\min_x \|b - Ax\|_2$ in precision u_f using the computed QR factors and store in precision u.
for $i = 0\colon i_{\max} - 1$ **do**
  Compute $s_i = \widetilde{b} - \widetilde{A}y_i$ at precision u_r and round r_i to precision u.
  Solve $M^{-1}\widetilde{A}d_i = M^{-1}r_i$ by GMRES at precision u, with matrix–vector products with $M\widetilde{A}$ computed at precision u_r, exploiting the block structure, and store d_i at precision u. M is given by either (3.3) or (3.9).
  $y_{i+1} = y_i + d_i$ at precision u.
  if converged **then**
    return y_{i+1}, **quit**
  end if
end for

4. Numerical experiments. We present numerical experiments to illustrate the results of our analysis for three-precision least squares iterative refinement using the combinations of precisions, (u_f, u, u_r) = (half, single, double), (half, double, quad), and (single, double, quad). Only u_f = half is considered. All the experiments are performed in MATLAB 2019a, on a Lenovo ThinkStation with Intel Xeon

TABLE 4.1
Two norm condition number and number of iterative refinement steps for (u_f, u, u_r) = (half, single, double). Numbers in parentheses denote the total number of GMRES iterations in Algorithm 3.2. Failure to converge is denoted by "–".

$\kappa_2(A)$	LSIR	GMRES-LSIR	GMRES-LSIR-BD
1.00e+03	13	2 (12)	2 (28)
1.00e+04	–	2 (22)	2 (39)
1.00e+05	–	2 (50)	2 (41)
1.00e+06	–	3 (168)	2 (68)
1.00e+07	–	20 (1822)	3 (94)
1.00e+08	–	– (–)	10 (244)

TABLE 4.2
Various condition numbers corresponding to (u_f, u, u_r) = (half, single, double).

$\kappa_2(A)$	$\kappa_\infty(\widetilde{A})$	$\kappa_\infty(M\widetilde{A})$	$\kappa_\infty(M_1\widetilde{A}M_2)$
1.00e+03	3.60e+04	2.30e+01	1.27e+03
1.00e+04	3.89e+05	1.57e+03	3.00e+03
1.00e+05	6.49e+06	9.00e+04	5.76e+03
1.00e+06	3.72e+08	5.12e+07	2.81e+04
1.00e+07	2.16e+11	9.92e+09	1.91e+06
1.00e+08	3.05e+09	1.09e+13	3.86e+08

W-2123 CPU and 32 Gb RAM. Mixed precision computations are carried out both in the native single and double precisions and using `chop` to simulate half precision and the Advanpix Multiprecision Computing Toolbox [23] for quadruple precision. Our codes are available at https://github.com/SrikaraPranesh/LSIR3. We test both the standard least squares iterative refinement described in Section 2 as well as our GMRES-based approach described in Algorithm 3.2. For u_f = half, $\theta_{\max} = 0.1$ is used. In the GMRES-based approach, we present results for the left and split preconditioners (3.2), and (3.9) respectively. All the condition numbers reported are computed in high precision using `mp.Digits(64)` option of Advanpix toolbox.

As before, in all tests we generate A via `gallery('randsvd',[m,n], kappa,3)` with $m = 100$ and $n = 10$ with various specified 2-norm condition numbers `kappa`. The same right-hand side b is used in all tests, generated as a MATLAB `randn` vector with unit 2-norm. For reproducibility we seed the random number generator by calling the MATLAB function `rng(1)`. For the GMRES-based approach, to minimize the condition number we scale the (1,1) block of the augmented matrix (1.9) by α in (2.6), which is estimated at the working precision using the R factor of the QR factorization computed in precision u_f using the `svd` command.

For the iterative refinement process, we use the convergence condition that the relative error in both the solution $\widehat{x}_i$ and the residual $\widehat{r}_i$ is less than u. We set the GMRES convergence tolerance to 10^{-6} for u = single and 10^{-12} for u = double.

First we consider (u_f, u, u_r) = (half, single, double). In Table 4.1 we display the number of refinement steps for standard least squares iterative refinement in Algorithm 1.1 (LSIR), for GMRES-based iterative refinement with left preconditioner M in Algorithm 3.2 (GMRES-LSIR), and GMRES-based iterative refinement with split block diagonal preconditioners M_1 and M_2 (GMRES-LSIR-BD). For GMRES-LSIR and GMRES-LSIR-BD the numbers in parentheses denotes the total number of GMRES iterations. Various infinity norm condition numbers are displayed in Table 4.2.

From Tables 4.1 and 4.2 we can see that indeed, standard LSIR converges to

TABLE 4.3

Two norm condition number and number of iterative refinement steps for (u_f, u, u_r) = (half, double, quad). Numbers in parentheses denote the total number of GMRES iterations in Algorithm 3.2. Failure to converge is denoted by "–".

$\kappa_2(A)$	LSIR	GMRES-LSIR	GMRES-LSIR-BD
1.00e+02	13	2 (16)	2 (31)
1.00e+04	–	2 (24)	2 (41)
1.00e+07	–	2 (70)	2 (41)
1.00e+09	–	4 (212)	2 (41)
1.00e+10	–	5 (375)	3 (77)
1.00e+11	–	13 (992)	2 (41)
1.00e+12	–	– (–)	3 (151)

TABLE 4.4

Two norm condition number and number of iterative refinement steps for (u_f, u, u_r) = (single, double, quad). Numbers in parentheses denote the total number of GMRES iterations in Algorithm 3.2. Failure to converge is denoted by "–".

$\kappa_2(A)$	LSIR	GMRES-LSIR	GMRES-LSIR-BD
1.00e+03	3	1 (3)	1 (7)
1.00e+05	6	2 (8)	2 (19)
1.00e+07	19	2 (13)	2 (27)
1.00e+09	–	2 (21)	2 (39)
1.00e+11	–	2 (29)	2 (41)
1.00e+13	–	4 (254)	2 (41)
1.00e+15	–	8 (652)	17 (342)
1.00e+16	–	– (–)	– (–)

single precision if $\kappa_\infty(\widetilde{A}) \lesssim u_f^{-1}$, and fails to converge for $\kappa_\infty(\widetilde{A}) \gg u_f^{-1}$, as predicted by the analysis. From the analysis in section 3, we expect the GMRES-based approach to work for higher $\kappa_\infty(\widetilde{A})$ than the standard approach, for $\kappa_\infty(\widetilde{A}) \lesssim u^{-1/2}u_f^{-1}$. When u is single precision and u_f is half precision, this condition becomes roughly $\kappa_\infty(\widetilde{A}) \lesssim 10^7$. Indeed, GMRES-based iterative refinement (with both preconditioners) converges for condition numbers $\kappa_\infty(\widetilde{A})$ up to the order of 10^7. Note that since α in (2.6) is estimated using the u_f precision R factor, $\kappa_\infty(\widetilde{A})$ is somewhat larger than might be expected in view of (2.5). The GMRES-based algorithm converges even for $\kappa_\infty(\widetilde{A}) > 10^9$, but the left preconditioner requires more GMRES iterations than the split preconditioner. Also from columns 3 and 4 of Table 4.2 we can observe that for ill-conditioned matrices, split preconditioning gives a lower condition number. Comparing GMRES-LSIR and GMRES-LSIR BD in Table 4.1, we see that GMRES-LSIR BD takes slightly fewer GMRES iterations to converge than GMRES-LSIR.

In Tables 4.3 and 4.4 we display the results for LSIR, GMRES-LSIR, and GMRES-LSIR-BD for (u_f, u, u_r) = (half, double, quad) and (single, double, quad), respectively. In the Tables 4.5 and 4.6 relevant infinity norm condition numbers are presented. The results are as expected by the theory; LSIR converges for $\kappa_\infty(\widetilde{A})$ up to around u^{-1}, although as $\kappa_\infty(\widetilde{A})$ approaches u^{-1}, a larger number of refinement steps are needed. The GMRES-LSIR approach is able to converge for $\kappa_\infty(\widetilde{A})$ up to $u^{-1/2}u_f^{-1}$ as predicted by the theory, where $u^{-1/2}u_f^{-1} \approx 10^{16}$ for (single, double, quad) and $u^{-1/2}u_f^{-1} \approx 10^{12}$ for (half, double, quad). However, the number of GMRES iterations required does grow large as $\kappa_\infty(\widetilde{A})$ approaches $u^{-1/2}u_f^{-1}$. The split preconditioner tends to performs better than the left preconditioner on the more ill-conditioned problems. From these experiments we can conclude that even though the convergence of iterative

TABLE 4.5
Various condition numbers corresponding to $(u_f, u, u_r) = (half, double, quad)$.

$\kappa_2(A)$	$\kappa_\infty(\widetilde{A})$	$\kappa_\infty(M\widetilde{A})$	$\kappa_\infty(M_1\widetilde{A}M_2)$
1.00e+02	3.14e+03	2.93e+00	4.37e+02
1.00e+04	3.48e+05	3.48e+03	4.16e+03
1.00e+07	2.55e+10	2.69e+11	1.77e+07
1.00e+09	1.45e+14	1.38e+15	3.69e+10
1.00e+10	2.91e+16	1.62e+15	5.29e+12
1.00e+11	9.17e+17	1.17e+17	8.65e+13
1.00e+12	6.25e+19	2.79e+17	8.32e+15

TABLE 4.6
Various condition numbers corresponding to $(u_f, u, u_r) = (single, double, quad)$.

$\kappa_2(A)$	$\kappa_\infty(\widetilde{A})$	$\kappa_\infty(M\widetilde{A})$	$\kappa_\infty(M_1\widetilde{A}M_2)$
1.00e+03	3.44e+04	1.00e+00	1.30e+03
1.00e+05	3.62e+06	1.07e+00	1.35e+04
1.00e+07	2.65e+08	2.00e+01	1.25e+05
1.00e+09	4.81e+10	9.41e+04	4.14e+05
1.00e+11	2.33e+14	9.89e+08	7.95e+05
1.00e+13	2.03e+18	1.08e+14	1.32e+08
1.00e+15	3.78e+22	3.74e+17	3.00e+12
1.00e+16	1.02e+24	1.22e+17	4.29e+13

refinement with two-sided preconditioning cannot be theoretically guaranteed, this approach works well in practice.

For the final experiment we consider matrices from the SuiteSparse matrix collection [8]. We consider all rectangular full rank matrices with $20 \le m \le 2000$, $n \le 400$, and $n < m$. Properties of the matrices are displayed in Table 4.7, and we observe that entries of matrix 5 and 6 will overflow upon conversion to fp16. A random right-hand side vector is generated using `randn`, seeded for reproducibility, and both (half, single, double) and (half, double, quad) precision combinations are used. Again left (GMRES-LSIR) and two-sided (GMRES-LSIR-BD) preconditioners are considered. In Table 4.8 we display the results for (half, single, double), and as predicted by the analysis, LSIR fails to converge for $\kappa_\infty(\widetilde{A}) \gtrsim 10^4$. GMRES-LSIR and GMRES-LSIR-BD converge for all the matrices except 6 and 11, for which $\widetilde{A}$ and the preconditioned matrices are extremely ill conditioned as can be observed from Table 4.9. Results for (half, double, quad) are displayed in Tables 4.10 and 4.11. For both preconditioners, Algorithm 3.2 converges for all matrices, but GMRES-LSIR preforms slightly better than GMRES-LSIR-BD. However, for the ill-conditioned matrices 6 and 11, many GMRES iterations are required.

5. Conclusion. We have extended iterative refinement in three precisions for square linear systems, as studied in [5], to least squares problems. Our approach is to work with the augmented system and a QR factorization of A computed at the lowest of the three precisions. Our GMRES-IR algorithm uses GMRES to solve the correction equation with a choice of two different preconditioners, (3.2) and (3.9).

Our rounding error analysis, combined with the results in [4], [5], shows that GMRES-IR with (3.2) yields a forward error, and a backward error for the augmented system, of order the working precision under reasonable assumptions. Our error analysis is not applicable to the preconditioner (3.9). Numerical experiments show

TABLE 4.7

Rectangular test matrices chosen from the SuiteSparse Matrix Collection. m, n denote the number rows and columns respectively, $\max_{i,j}|a_{ij}|$ and the last column denotes the absolute value of the maximum entry and minimum nonzero entry, respectively.

Index	Matrix	(m,n)	$\kappa_2(A)$	$\max_{i,j}\|a_{ij}\|$	$\min_{i,j}\{\|a_{ij}\| : a_{ij} \neq 0\}$
1	divorce	(50,9)	1.94e+01	1.00e+00	$1.00e+00$
2	Cities	(55,46)	2.07e+02	7.10e+01	$1.00e+00$
3	ash219	(219,85)	3.02e+00	1.00e+00	$1.00e+00$
4	WorldCities	(315,100)	6.60e+01	1.00e+00	$1.00e+00$
5	ash331	(331,104)	3.10e+00	2.37e+06	$1.85e+04$
6	robot24c1_mat5	(404,302)	3.33e+11	2.46e+07	$4.00e+00$
7	ash608	(608,188)	3.37e+00	1.00e+00	$1.00e+00$
8	ash958	(958,292)	3.20e+00	1.57e+02	$3.04e-05$
9	illc1033	(1033,320)	1.89e+04	1.95e+00	$1.24e-02$
10	well1033	(1033,320)	1.66e+02	1.00e+00	$1.00e+00$
11	photogrammetry	(1388,390)	4.35e+08	1.00e+00	$1.00e+00$

TABLE 4.8

For matrices in Table 4.7, number of iterative refinement steps in standard least squares iterative refinementand Algorithm 3.2, for $(u_f, u, u_r) = (half, single, double)$. Numbers in parentheses denote the total number of GMRES iterations in Algorithm 3.2, for left (GMRES-LSIR), and two-sided (GMRES-LSIR-BD) preconditioning. Failure to converge is denoted by "–".

Index	LSIR	GMRES-LSIR	GMRES-LSIR-BD
1	3	1 (3)	1 (7)
2	4	1 (4)	1 (11)
3	2	1 (3)	1 (7)
4	4	1 (5)	1 (11)
5	2	1 (3)	1 (7)
6	–	– (–)	– (–)
7	2	1 (3)	1 (7)
8	2	1 (3)	1 (7)
9	–	2 (79)	2 (169)
10	–	2 (16)	2 (39)
11	–	– (–)	– (–)

that GMRES-IR behaves as predicted by the theory and that it can solve a much wider range of problems than three-precision iterative refinement without the use of GMRES; it also works about as well with the preconditioner (3.9).

Future aspects to explore include the extension of the results to the minimum 2-norm solution of underdetermined systems, refining the implementation details such as the choice of GMRES convergence tolerance, and implementation and performance studies on available low-precision hardware.

TABLE 4.9
For matrices in Table 4.7, various condition numbers corresponding to $(u_f, u, u_r) = (half, single, double)$.

Index	$\kappa_\infty(\widetilde{A})$	$\kappa_\infty(M\widetilde{A})$	$\kappa_\infty(M_1\widetilde{A}M_2)$
1	2.86e+02	1.26e+00	4.76e+01
2	2.69e+03	7.21e+00	8.04e+01
3	4.32e+01	1.05e+00	2.30e+01
4	3.37e+03	4.04e+01	2.30e+02
5	5.27e+01	1.05e+00	2.78e+01
6	3.66e+12	2.94e+20	1.76e+14
7	5.99e+01	1.06e+00	2.75e+01
8	5.86e+01	1.06e+00	2.83e+01
9	1.54e+06	1.21e+06	1.22e+04
10	1.44e+04	2.32e+02	1.34e+03
11	1.10e+10	4.24e+14	3.95e+10

TABLE 4.10
For matrices in Table 4.7, number of iterative refinement steps in standard least squares iterative refinementand Algorithm 3.2, for $(u_f, u, u_r) = (half, double, quad)$. *Numbers in parentheses denote the total number of GMRES iterations in Algorithm 3.2, for left (GMRES-LSIR), and two-sided (GMRES-LSIR-BD) preconditioning. Failure to converge is denoted by "–".*

Index	LSIR	GMRES-LSIR	GMRES-LSIR-BD
1	7	1 (6)	1 (12)
2	9	2 (14)	2 (33)
3	5	1 (5)	1 (10)
4	11	2 (16)	2 (33)
5	5	1 (5)	1 (10)
6	–	4 (1616)	3 (1127)
7	5	1 (5)	1 (10)
8	5	1 (5)	1 (10)
9	–	2 (89)	2 (191)
10	–	2 (28)	2 (63)
11	–	4 (5428)	2 (1950)

TABLE 4.11
For matrices in Table 4.7, various condition numbers corresponding to $(u_f, u, u_r) = (half, double, quad)$.

Index	$\kappa_\infty(\widetilde{A})$	$\kappa_\infty(M\widetilde{A})$	$\kappa_\infty(M_1\widetilde{A}M_2)$
1	2.86e+02	1.26e+00	4.76e+01
2	2.69e+03	7.21e+00	8.04e+01
3	4.32e+01	1.05e+00	2.30e+01
4	3.37e+03	4.13e+01	2.30e+02
5	5.27e+01	1.05e+00	2.78e+01
6	1.36e+16	3.35e+20	1.75e+14
7	5.99e+01	1.06e+00	2.75e+01
8	5.86e+01	1.06e+00	2.83e+01
9	1.54e+06	1.21e+06	1.22e+04
10	1.44e+04	2.32e+02	1.34e+03
11	1.66e+15	2.88e+15	1.69e+11

REFERENCES

[1] M. Arioli, I. S. Duff, S. Gratton, and S. Pralet. A note on GMRES preconditioned by a perturbed LDL^T decomposition with static pivoting. *SIAM J. Sci. Comput.*, 29(5):2024–2044, 2007.

[2] Åke Björck. Iterative refinement of linear least squares solutions I. *BIT Numerical Mathematics*, 7(4):257–278, 1967.

[3] Åke Björck. Iterative refinement and reliable computing. In *Reliable Numerical Computation*, M. G. Cox and S. J. Hammarling, editors, Oxford University Press, 1990, pages 249–266.

[4] Erin Carson and Nicholas J. Higham. A new analysis of iterative refinement and its application to accurate solution of ill-conditioned sparse linear systems. *SIAM J. Sci. Comput.*, 39(6): A2834–A2856, 2017.

[5] Erin Carson and Nicholas J. Higham. Accelerating the solution of linear systems by iterative refinement in three precisions. *SIAM J. Sci. Comput.*, 40(2):A817–A847, 2018.

[6] Erin Carson and Zdeněk Strakoš. On the cost of iterative computations. *Phil. Trans. R. Soc. A*, 378(2166):1–22, 2020.

[7] Anthony J. Cox and Nicholas J. Higham. Stability of Householder QR factorization for weighted least squares problems. In *Numerical Analysis* 1997*, Proceedings of the* 17*th Dundee Biennial Conference*, D. F. Griffiths, D. J. Higham, and G. A. Watson, editors, volume 380 of *Pitman Research Notes in Mathematics*, Addison Wesley Longman, Harlow, Essex, UK, 1998, pages 57–73.

[8] Timothy A. Davis and Yifan Hu. The University of Florida Sparse Matrix Collection. *ACM Trans. Math. Software*, 38(1):1:1–1:25, 2011.

[9] James Demmel, Yozo Hida, E. Jason Riedy, and Xiaoye S. Li. Extra-precise iterative refinement for overdetermined least squares problems. *ACM Trans. Math. Software*, 35(4):28:1–28:32, 2009.

[10] Gene Golub. Numerical methods for solving linear least squares problems. *Numerische Mathematik*, 7(3):206–216, 1965.

[11] Gene H. Golub and James H. Wilkinson. Note on the iterative refinement of least squares solution. *Numerische Mathematik*, 9(2):139–148, 1966.

[12] Azzam Haidar, Ahmad Abdelfattah, Mawussi Zounon, Panruo Wu, Srikara Pranesh, Stanimire Tomov, and Jack Dongarra. The design of fast and energy-efficient linear solvers: On the potential of half-precision arithmetic and iterative refinement techniques. In *Computational Science—ICCS 2018*, Yong Shi, Haohuan Fu, Yingjie Tian, Valeria V. Krzhizhanovskaya, Michael Harold Lees, Jack Dongarra, and Peter M. A. Sloot, editors, Springer International Publishing, Cham, 2018, pages 586–600.

[13] Azzam Haidar, Stanimire Tomov, Jack Dongarra, and Nicholas J. Higham. Harnessing GPU tensor cores for fast FP16 arithmetic to speed up mixed-precision iterative refinement solvers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, SC '18 (Dallas, TX), Piscataway, NJ, USA, 2018, pages 47:1–47:11. IEEE Press.

[14] Nicholas J. Higham. Iterative refinement enhances the stability of QR factorization methods for solving linear equations. *BIT*, 31:447–468, 1991.

[15] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Second edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002. xxx+680 pp. ISBN 0-89871-521-0.

[16] Nicholas J. Higham. Error analysis for standard and GMRES-based iterative refinement in two and three-precisions. MIMS EPrint 2019.19, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, November 2019. 8 pp.

[17] Nicholas J. Higham and Srikara Pranesh. Exploiting lower precision arithmetic in solving symmetric positive definite linear systems and least squares problems. MIMS EPrint 2019.20, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, November 2019. 19 pp.

[18] Nicholas J. Higham and Srikara Pranesh. Simulating low precision floating-point arithmetic. *SIAM J. Sci. Comput.*, 41(5):C585–C602, 2019.

[19] Nicholas J. Higham, Srikara Pranesh, and Mawussi Zounon. Squeezing a matrix into half precision, with an application to solving linear systems. *SIAM J. Sci. Comput.*, 41(4): A2536–A2551, 2019.

[20] *IEEE Standard for Floating-Point Arithmetic, IEEE Std 754-2008 (revision of IEEE Std 754-1985)*. IEEE Computer Society, New York, 2008. 58 pp. ISBN 978-0-7381-5752-8.

[21] Intel Corporation. BFLOAT16—hardware numerics definition, November 2018. White paper. Document number 338302-001US.

[22] Ilse C.F. Ipsen. A note on preconditioning nonsymmetric matrices. *SIAM J. Sci. Comput.*, 23 (3):1050–1051, 2001.
[23] Multiprecision Computing Toolbox. Advanpix, Tokyo. http://www.advanpix.com.
[24] Malcom F. Murphy, Gene H. Golub, and Andrew J. Wathen. A note on preconditioning for indefinite linear systems. *SIAM J. Sci. Comput.*, 21(6):1969–1972, 2000.
[25] Shaoshuai Zhang and Panruo Wu. High accuracy low precision QR factorization and least square solver on GPU with tensorcore. ArXiv preprint 1912.05508, 2019.