SIAM J. SCI. COMPUT.
Vol. 42, No. 1, pp. A477–A503

SHIFTED CHOLESKY QR FOR COMPUTING THE QR FACTORIZATION OF ILL-CONDITIONED MATRICES*

TAKESHI FUKAYA†, RAMASESHAN KANNAN‡, YUJI NAKATSUKASA§, YUSAKU YAMAMOTO¶, AND YUKA YANAGISAWA‖

Abstract. The Cholesky QR algorithm is an efficient communication-minimizing algorithm for computing the QR factorization of a tall-skinny matrix $X \in \mathbb{R}^{m\times n}$, where $m \gg n$. Unfortunately it is inherently unstable and often breaks down when the matrix is ill-conditioned. A recent work [Yamamoto et al., *ETNA*, 44, pp. 306–326 (2015)] establishes that the instability can be cured by repeating the algorithm twice (called CholeskyQR2). However, the applicability of CholeskyQR2 is still limited by the requirement that the Cholesky factorization of the Gram matrix $X^\top X$ runs to completion, which means that it does not always work for matrices X with the 2-norm condition number $\kappa_2(X)$ roughly greater than $\mathbf{u}^{-1/2}$, where $\mathbf{u}$ is the unit roundoff. In this work we extend the applicability to $\kappa_2(X) = \mathcal{O}(\mathbf{u}^{-1})$ by introducing a shift to the computed Gram matrix so as to guarantee the Cholesky factorization $R^\top R = A^\top A + sI$ succeeds numerically. We show that the computed AR^{-1} has reduced condition number that is roughly bounded by $\mathbf{u}^{-1/2}$, for which CholeskyQR2 safely computes the QR factorization, yielding a computed Q of orthogonality $\|Q^\top Q - I\|_2$ and residual $\|A - QR\|_F/\|A\|_F$ both of the order of $\mathbf{u}$. Thus we obtain the required QR factorization by essentially running Cholesky QR thrice. We extensively analyze the resulting algorithm shifted-CholeskyQR3 to reveal its excellent numerical stability. The shiftedCholeskyQR3 algorithm is also highly parallelizable, and applicable and effective also when working with an oblique inner product. We illustrate our findings through experiments, in which we achieve significant speedup over alternative methods.

Key words. QR factorization, Cholesky QR factorization, oblique inner product, roundoff error analysis, communication-avoiding algorithms

AMS subject classifications. 65F30, 15A23, 65F15, 15A18, 65G50

DOI. 10.1137/18M1218212

1. Introduction.

Computing the QR factorization $X = QR$ is required in various applications in scientific computing. Here, $X \in \mathbb{R}^{m\times n}$ with $m \geq n$, $Q \in \mathbb{R}^{m\times n}$ is a matrix with orthonormal columns and $R \in \mathbb{R}^{n\times n}$ is an upper triangular matrix with positive diagonal elements. Such a problem arises, for example, in the linear least squares [10], s-step Krylov subspace methods [16], and electronic structure calculations [2, 21]. In the following, we assume that X is of full rank. The Cholesky QR algorithm computes the factorization by:

*Submitted to the journal's Methods and Algorithms for Scientific Computing section October 1, 2018; accepted for publication (in revised form) November 26, 2019; published electronically February 20, 2020.

https://doi.org/10.1137/18M1218212

Funding: The first author's work was supported by JSPS KAKENHI Grant JP15K16000 and JP18K18058. The fourth author's work was supported by JSPS KAKENHI Grant 17H02828, 17K19966, and 19KK02555. The financial support for open access fee was provided by Information Initiative Center, Hokkaido University.

†Hokkaido University, Hokkaido, Japan (fukaya@iic.hokudai.ac.jp).

‡Arup, 3 Piccadilly Place, Manchester M1 3BN, United Kingdom (Ramaseshan.Kannan@arup.com).

§Mathematical Institute, University of Oxford, Oxford, OX2 6GG, UK, and National Institute of Informatics (nakatsukasa@maths.ox.ac.uk).

¶The University of Electro-Communications, Tokyo, Japan (yusaku.yamamoto@uec.ac.jp).

‖Waseda University, Waseda Research Institute for Science and Engineering, Tokyo, Japan (yuuka@aoni.waseda.jp).

$$A = X^\top X, \tag{1.1}$$

$$R = \mathrm{chol}(A), \tag{1.2}$$

$$Q = XR^{-1}, \tag{1.3}$$

where $\mathrm{chol}(A)$ denotes the upper triangular Cholesky factor of A. Cholesky QR is a communication-avoiding algorithm whose communication cost is equivalent to that of the tall-skinny QR (TSQR) algorithm, which has been devised specifically to minimize communication for the QR factorization of tall-skinny matrices [6].

Cholesky QR has the advantage over TSQR that its arithmetic cost is about half and that its reduction operator is addition, while that of TSQR is a QR factorization of a small matrix [8]. As a result, Cholesky QR usually runs faster than TSQR. However, Cholesky QR is rarely used in practice because of its instability: the distance from orthogonality of its computed Q grows rapidly with the condition number of the input matrix. By contrast, TSQR is unconditionally stable.

A recent work [23] shows that the instability can be cured significantly by repeating the algorithm twice (called CholeskyQR2). However, the applicability of CholeskyQR2 is still limited by the requirement that the Cholesky factorization of the Gram matrix $A = X^\top X$ runs to completion, which means that it does not always work for matrices X with $\kappa_2(X) = \mathcal{O}(\mathbf{u}^{-1/2})$ or larger, where $\mathbf{u}$ is the unit roundoff and $\kappa_2(X) = \|X\|_2\|X^{-1}\|_2$ is the condition number, where $\|X\|_2 = \sigma_{\max}(X)$ denotes the 2-norm. In this work we extend the applicability of Cholesky QR-based algorithms to matrices with $\kappa_2(X)$ up to $\mathcal{O}(\mathbf{u}^{-1})$.

The idea is to execute a preconditioning step so that the condition number is reduced to a point where CholeskyQR2 is applicable. How do we find an effective preconditioner? An inspiration to answer this is the fact that Cholesky QR and CholeskyQR2 belong to the category of *triangular orthogonalization* type algorithms [22, Lecture 10]—along with Gram–Schmidt processes—which is based on multiplying an upper triangular matrix to orthogonalize X. This is in contrast to Householder type algorithms, which follows the principle of *orthogonal triangularization*, wherein a sequence of orthogonal matrices are left-multiplied to obtain an upper triangular matrix. We summarize the classification of algorithms in terms of their principle, communication cost, and stability in Figure 1.1, which also clarifies where our contribution (shifted CholeskyQR3) stands.

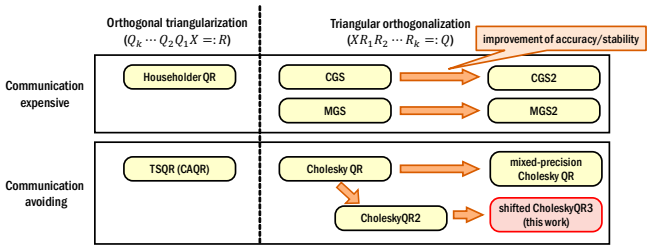

FIG. 1.1. *Classification of QR factorization algorithms. CGS stands for classical Gram–Schmidt, MGS modified Gram–Schmidt, and CGS2, MGS2 are their repeated variants. Mixed-precision Cholesky QR is the algorithm in* [24].

In triangular orthogonalization, one right-multiplies an appropriate upper triangular matrix R so that $\kappa_2(XR) = 1$. Now, what if our goal is merely $\kappa_2(XR) = O(\mathbf{u}^{-1/2})$? Once we have this, we can safely compute the QR factorization $XR = QR_1$ using CholeskyQR2, to arrive at the overall QR factorization $X = Q(R_1R^{-1})$.

Clearly, such R is not unique, and we propose one way of finding such R. Namely, as in Cholesky QR we compute the Gram matrix, but add a small shift sI so as to

guarantee the Cholesky factorization $R^\top R = X^\top X + sI$ does not break down numerically. We show that under the mild assumption $\kappa_2(X) \le \mathbf{u}^{-1}$, the resulting XR^{-1} has reduced condition number that is smaller than $\mathbf{u}^{-1/2}$, for which CholeskyQR2 safely computes the QR factorization, yielding computed Q, R with excellent orthogonality $\|Q^\top Q - I\|_F = \mathcal{O}(\mathbf{u})$ and residual $\|X - QR\|_F/\|X\|_F = \mathcal{O}(\mathbf{u})$, where $\|\cdot\|_F$ denotes the Frobenius norm, overall a backward stable QR factorization. The algorithm is deceptively simple (essentially the only new ingredient being the introduction of a shift); the analysis is however not trivial. We give a detailed analysis that gives the constants hidden in the $\mathcal{O}(\mathbf{u})$ notation.

The main message of this paper is that for any matrix with condition number well above $\mathbf{u}^{-1/2}$ (but bounded by $\mathbf{u}^{-1}$), the QR factorization can be computed in a backward stable manner by essentially running Cholesky QR thrice, namely, running the first with a shift and the second and the third without a shift. We refer to this overall algorithm as shiftedCholeskyQR3; Figure 1.2 shows its flow and how $\kappa_2(X)$ is reduced eventually to 1 through repeated multiplication by triangular matrices.

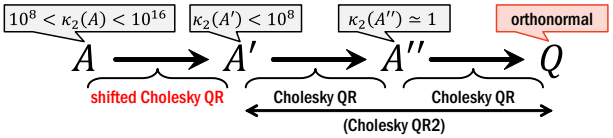

FIG. 1.2. *Diagram illustrating shiftedCholeskyQR3.*

Let us comment on related studies in the literature. A recent work of Yamazaki et al. [24] uses doubled precision arithmetic (e.g., quadruple precision when working in double precision) for the first two steps (1.1), (1.2), and shows that the condition number gets reduced by about $O(\mathbf{u}^{-1})$, and thus the QR factorization will be obtained by repeating the process. This results in about 8.5 times as many arithmetic operations (per iteration) as does the standard Cholesky QR without doubled precision. Moreover, this approach clearly requires that higher-precision arithmetic is available, which may not always be the case (and even when it is, it often comes with a significant overhead in terms of speed). The shiftedCholeskyQR3 algorithm developed in this paper does not require a change of arithmetic precision (in our experiments we use only IEEE 754 double precision in which $\mathbf{u} \approx 1.1 \times 10^{-16}$), and requires just one more Cholesky QR iteration than [24], thus it is usually much faster.

As a bonus, our development is straightforward to apply to computing the QR factorization in a nonstandard inner product space induced by a positive definite matrix B, in which the inner product is defined by $(x, y)_B := x^\top By$. Such oblique QR factorization arises for example in the solution of generalized eigenvalue problems [1, Chapter 5], and weighted least-squares problems arising, e.g., in compressed sensing [3]. Available algorithms for this task include (modified) Gram–Schmidt [18] and Cholesky QR [14, 15, 20]. A recent work by Lowery and Langou [15] studies the numerical stability, with no method apparently being (near) optimal for both orthogonality and backward error. We analyze the stability of shiftedCholeskyQR3 in this case to show it has favorable stability properties. Moreover, shiftedCholeskyQR3 is significantly faster than Gram–Schmidt type algorithms.

This paper is organized as follows. In section 2 we describe the shiftedCholeskyQR algorithm. Section 3 analyzes shiftedCholeskyQR in detail and shows that it can be used to reduce $\kappa_2(X)$. In section 4 we combine shiftedCholeskyQR and CholeskyQR2 to derive shiftedCholeskyQR3 for ill-conditioned matrices with $\kappa_2(X) = \mathcal{O}(\mathbf{u}^{-1})$ and prove its backward stability. Section 5 discusses an extension to the nonstandard

inner product space. Numerical experiments are shown to illustrate the results in section 6, and section 7 summarizes the performance of our implementation.

Notation. Throughout, we denote the kth largest singular value of a matrix A by $\sigma_k(A)$. The smallest singular value of A is written as $\sigma_{\min}(A)$. For a symmetric positive definite matrix A, $\lambda_k(A)$ denotes the kth largest eigenvalue of A. The symbol $fl(\cdot)$ denotes the value of an expression in the parentheses evaluated in floating-point arithmetic and $\gamma_n := n\mathbf{u}/(1-n\mathbf{u})$. We primarily focus on real matrices $X \in \mathbb{R}^{m\times n}$, but everything carries over to complex matrices $X \in \mathbb{C}^{m\times n}$. We assume $m \geq n$, and the algorithms developed here are particularly useful in the tall-skinny case $m \gg n$.

2. Shifted Cholesky QR. To overcome a possible numerical breakdown in the Cholesky factorization (1.2), we propose a simple remedy: to introduce a small shift s in order to force the computed Gram matrix $X^\top X$ to be numerically positive definite, so that its Cholesky factorization runs without breakdown. The rest of the algorithm is the same as Cholesky QR, and the algorithm shiftedCholeskyQR can be summarized in pseudocode in Algorithm 2.1.

Algorithm 2.1. shiftedCholeskyQR for $X = QR$.

1: $A = X^\top X$,
2: choose $s > 0$
3: $R = \mathrm{chol}(A + sI)$
4: $Q = XR^{-1}$

Introducing shifts in Cholesky QR was briefly mentioned in [20], also as a remedy for the breakdown. However, the focus there was on the algorithm called singular vector QB factorization (SVQB), which computes the SVD of the Gram matrix instead of the Cholesky factorization. SVQB computes a factorization of the form $X = QB$, where B is a full $n \times n$ matrix. While Q is still a basis for the column space of X, an additional QR factorization of B is needed to obtain a complete QR factorization of X. Furthermore, experiments suggest that there are benefits in working with a triangular matrix as in Cholesky QR as opposed to full matrices as in SVQB, with respect to the row-wise stability of the computed decomposition.

We shall discuss an appropriate choice of the shift s, which will be $O(\mathbf{u})$, and prove that applying shiftedCholeskyQR to a matrix X with $\mathbf{u}^{-1/2} < \kappa_2(X) < \mathbf{u}^{-1}$ results in the computed Q, which we denote by $\hat{Q}$, with sufficiently reduced condition number $\kappa_2(\hat{Q}) < \mathbf{u}^{-1/2}$, for which CholeskyQR2 suffices to compute the QR factorization.

3. Convergence and stability analysis of shiftedCholeskyQR. In this section we present the main technical analysis of shiftedCholeskyQR applied to $X \in \mathbb{R}^{m\times n}$. We assume that $m \geq n$ and X is of full rank. The goal is to show that the algorithm improves the conditioning significantly, that is, $\kappa_2(\hat{Q}) \leq \mathcal{O}(\mathbf{u}^{1/2})\kappa_2(X) \ll \kappa_2(X)$.

A few assumptions are needed on the matrix size and condition number. The constants below are not of significant importance but chosen so that the forthcoming analysis runs smoothly. We shall assume the following hold:

$$6n^2\mathbf{u}\kappa_2(X) < 1, \tag{3.1}$$

$$mn\mathbf{u} \leq \frac{1}{64}, \tag{3.2}$$

$$n(n+1)\mathbf{u} \leq \frac{1}{64}, \tag{3.3}$$

$$11(mn + n(n+1))\mathbf{u}\|X\|_2^2 \leq s \leq \frac{1}{100}\|X\|_2^2. \tag{3.4}$$

Roughly speaking, the first three assumptions require that the condition number $\kappa_2(X)$ is safely bounded above by $\mathbf{u}^{-1}$ and the matrix dimensions m, n are small compared with the precision $\mathbf{u}^{-1}$. We reiterate that $\kappa_2(X) > \mathbf{u}^{-1/2}$ is allowed, a crucial difference from CholeskyQR2 treated in [23]. The assumption (3.4) imposes that s is large enough for Cholesky to work [19, Theorem 2.3] but small compared with $\|X\|_2^2$.

Note that by (3.2), (3.3) we have

$$\gamma_m := \frac{m\mathbf{u}}{1-m\mathbf{u}} \le 1.02m\mathbf{u}, \quad \gamma_{n+1} := \frac{(n+1)\mathbf{u}}{1-(n+1)\mathbf{u}} \le 1.02(n+1)\mathbf{u}. \tag{3.5}$$

3.1. Preparations. We denote the computed quantities in shiftedCholeskyQR, accounting for the numerical errors, by

$$\hat{A} = X^\top X + E_1, \tag{3.6}$$

$$\hat{R}^\top \hat{R} = \hat{A} + sI + E_2 \;=\; X^\top X + sI + E_1 + E_2, \tag{3.7}$$

$$\hat{q}_i^\top = x_i^\top (\hat{R} + \Delta\hat{R}_i)^{-1} \quad (i = 1, 2, \dots m). \tag{3.8}$$

Here, $\hat{q}_i^\top$, $x_i^\top$ are the ith rows of X and $\hat{Q}$, respectively. The matrix E_1 represents the matrix-matrix multiplication error in the computation of the Gram matrix $X^\top X$, and E_2 is the $\hat{A}$ backward error incurred when computing the Cholesky factorization. The matrix $\Delta\hat{R}_i$ represents the backward error involved in the solution of the linear system $q_i^\top \hat{R} = x_i^\top$.

Let us write

$$s := \alpha\|X\|_2^2, \quad 0 < \alpha < \frac{1}{100}. \tag{3.9}$$

On the other hand, (3.8) gives

$$\hat{q}_i^\top = x_i^\top (\hat{R} + \Delta\hat{R}_i)^{-1} = x_i^\top (I + \hat{R}^{-1}\Delta\hat{R}_i)^{-1}\hat{R}^{-1}.$$

Hence, letting

$$(I + \hat{R}^{-1}\Delta\hat{R}_i)^{-1} = I + \breve{R}_i \tag{3.10}$$

and defining

$$\Delta x_i^\top := x_i^\top \breve{R}_i, \tag{3.11}$$

we have

$$\hat{q}_i^\top = (x_i^\top + \Delta x_i^\top)\hat{R}^{-1} \quad (i = 1, 2, \dots m). \tag{3.12}$$

Let

$$\Delta X = \begin{bmatrix} \Delta x_1^\top \\ \vdots \\ \Delta x_m^\top \end{bmatrix}$$

be the matrix obtained by stacking up the row vectors $\Delta x_i^\top$. Then

$$\hat{Q} = (X + \Delta X)\hat{R}^{-1}. \tag{3.13}$$

For later use, here we examine the relation between $\Delta\hat{R}_i$ and $\Delta x_i^\top$. From (3.8) we have $\hat{q}_i^\top \hat{R} + \hat{q}_i^\top \Delta\hat{R}_i = x_i^\top$. We also have from (3.12) $\hat{q}_i^\top \hat{R} = x_i^\top + \Delta x_i^\top$. Combining these we obtain

$$\Delta x_i^\top = -\hat{q}_i^\top \Delta\hat{R}_i. \tag{3.14}$$

3.1.1. Error in computing $X^\top X$. For general matrices $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times m}$, the error in computing the matrix product $C = AB$ can be bounded by [11, Chapter 3]

$$|AB - fl(AB)| \le \gamma_n |A||B|. \tag{3.15}$$

Here $|A|$ is the matrix whose (i,j) element is $|a_{ij}|$. In [23] it is shown that E_1 is bounded by

$$\|E_1\|_2 \le \| \, |E_1| \, \|_F \le \gamma_m n \|X\|_2^2. \tag{3.16}$$

Simplifying (3.16) using (3.5) yields $\|E_1\|_2 \le 1.1mn\mathbf{u}\|X\|_2^2$.

3.1.2. Backward error in the Cholesky factorization. Suppose that $A \in \mathbb{R}^{n\times n}$ is a positive definite matrix and its Cholesky factorization computed in floating-point arithmetic runs to completion and outputs $\hat{R}$. Then there exists $\Delta A \in \mathbb{R}^{n\times n}$ such that [11, Theorem 10.3]

$$\hat{R}^\top \hat{R} = A + \Delta A, \quad |\Delta A| \le \gamma_{n+1} |\hat{R}^\top||\hat{R}|. \tag{3.17}$$

Applying this to our situation ($\Delta A := E_2$) gives

$$\|E_2\|_2 \le \| \, |E_2| \, \|_F \le \gamma_{n+1} \| \, |\hat{R}| \, \|_F^2 \le \gamma_{n+1} n \|\hat{R}\|_2^2 \le \gamma_{n+1} n (\|\hat{A}\|_2 + \|E_2\|_2), \tag{3.18}$$

where we used $\| \, |\hat{R}| \, \|_F^2 = \|\hat{R}\|_F^2 \le n\|\hat{R}\|_2^2$. Using [23, Equation 3.16] in the right-hand side gives

$$\begin{aligned}
\|E_2\|_2 &\le \frac{\gamma_{n+1} n(1+\gamma_m n + \alpha)}{1 - \gamma_{n+1} n} \|X\|_2^2 \\
&\le \frac{1.02(n+1)\mathbf{u} \cdot n(1 + 1.02m\mathbf{u} \cdot n + 0.01)}{1 - 1.02(n+1)\mathbf{u} \cdot n} \|X\|_2^2 \\
&\le \frac{1.02 \cdot n(n+1)\mathbf{u} \cdot (1 + 1.02 \cdot \frac{1}{64} + 0.01)}{1 - \frac{1.02}{64}} \|X\|_2^2 \le 1.1n(n+1)\mathbf{u}\|X\|_2^2, \qquad (3.19)
\end{aligned}$$

where we have used $\alpha = s/\|X\|_2^2 \le 1/100$ and (3.2), (3.3).

3.1.3. Bounding $\|\hat{R}^{-1}\|_2$ and $\|X\hat{R}^{-1}\|_2$. Using Weyl's theorem [10, Section 8.6.2] in (3.7) gives

$$\sigma_n(\hat{R})^2 \ge (\sigma_n(X))^2 + s - (\|E_1\|_2 + \|E_2\|_2). \tag{3.20}$$

By $\|E_1\|_2 \le 1.1mn\mathbf{u}\|X\|_2^2$ and (3.4) and (3.19) we obtain

$$\|E_1\|_2 + \|E_2\|_2 \le 1.1(mn + n(n+1))\mathbf{u}\|X\|_2^2 \le 0.1s. \tag{3.21}$$

Substituting this into (3.20) we obtain

$$(\sigma_n(\hat{R}))^2 \ge (\sigma_n(X))^2 + 0.9s, \tag{3.22}$$

therefore

$$\|\hat{R}^{-1}\|_2 = \sigma_n(\hat{R})^{-1} \le \frac{1}{\sqrt{(\sigma_n(X))^2 + 0.9s}}. \tag{3.23}$$

Bound $\|X\hat{R}^{-1}\|_2$ can be done using (3.7), (3.21), and (3.23) as

$$\begin{aligned}
\|X\hat{R}^{-1}\|_2 &\le \sqrt{1 + \|\hat{R}^{-1}\|_2^2 (s + \|E_1\|_2 + \|E_2\|_2)} \\
&\le \sqrt{1 + \frac{1.1s}{(\sigma_n(X))^2 + 0.9s}} \le \sqrt{1 + \frac{1.1}{0.9}} \le 1.5. \qquad (3.24)
\end{aligned}$$

3.1.4. Bounding $\|\Delta\hat{R}_i\|_2$. Let $R \in \mathbb{R}^{n\times n}$ be a nonsingular upper triangular matrix. The computed solution $\hat{x}$ obtained by solving an upper triangular linear system $Rx = b$ by back substitution in floating-point arithmetic satisfies [11, Theorem 8.5]

$$(R + \Delta R)\hat{x} = b, \quad |\Delta R| \le \gamma_n |R|. \tag{3.25}$$

We have for $1 \le i \le m$

$$\|\Delta\hat{R}_i\|_2 \le \||\Delta\hat{R}_i|\|_F \le \gamma_n\sqrt{n}\|\hat{R}\|_2. \tag{3.26}$$

From (3.7), (3.4), and (3.21) we obtain

$$\|\hat{R}\|_2^2 \le \|X\|_2^2 + s + \|E_1\|_2 + \|E_2\|_2 \le \|X\|_2^2 + 1.1s \le 1.1\|X\|_2^2. \tag{3.27}$$

Substituting this into (3.26) gives

$$\|\Delta\hat{R}_i\|_2 \le 1.02n\mathbf{u}\cdot\sqrt{n}\cdot\sqrt{1.1}\|X\|_2 \le 1.1n\sqrt{n}\mathbf{u}\|X\|_2. \tag{3.28}$$

3.1.5. Bounding $\|\Delta X\|_2$ roughly. Here we give a rough bound for $\|\Delta X\|_F$ and prove that $\|\Delta X\|_F = O(\mathbf{u}\|X\|_2^2/\sqrt{s})$. This will be insufficient for proving that $\kappa_2(\hat{Q}) = O(1/\sqrt{\mathbf{u}})$, for which we will need $\|\Delta X\|_F = O(\mathbf{u}\|X\|_2)$. We will prove this later after having obtained a bound for $\hat{Q}$. We shall proceed as follows.

1. Derive the "rough" bound $\|\Delta X\|_F = O(\mathbf{u}\|X\|_2^2/\sqrt{s})$.
2. Use the result of step 1 to show $\|\hat{Q}\|_2 = O(1)$.
3. Use the result of step 2 and (3.14) to prove the "tight" bound $\|\Delta X\|_F = O(\mathbf{u}\|X\|_2)$.
4. Use abovethe result of step 3 to prove $\kappa_2(\hat{Q}) = O(\mathbf{u}^{-\frac{1}{2}})$.

To establish the first statement, we begin with deriving a bound on $\|\hat{R}^{-1}\Delta\hat{R}_i\|_2$. Using (3.23), (3.28), and (3.4), we have

$$\begin{aligned}\|\hat{R}^{-1}\Delta\hat{R}_i\|_2 \le \|\hat{R}^{-1}\|_2\|\Delta\hat{R}_i\|_2 &\le \frac{1.1n\sqrt{n}\mathbf{u}\|X\|_2}{\sqrt{(\sigma_n(X))^2 + 0.9s}} \\ &\le \frac{1.1n\sqrt{n}\mathbf{u}\|X\|_2}{\sqrt{0.9s}} \\ &\le \frac{1.1n\sqrt{n}\mathbf{u}\|X\|_2}{\sqrt{0.9\cdot 11mn\mathbf{u}\|X\|_2^2}} \le \sqrt{\frac{1.21}{9.9}\cdot\frac{n^2\mathbf{u}}{m}} \le 0.05.\end{aligned} \tag{3.29}$$

Hence, the matrix $\breve{R}_i$ in (3.10) can be expanded as $\breve{R}_i = (I + \hat{R}^{-1}\Delta\hat{R}_i)^{-1} - I = \sum_{k=1}^{\infty}(-\hat{R}^{-1}\Delta\hat{R}_i)^k$ and its norm bounded as

$$\|\breve{R}_i\|_2 \le \sum_{k=1}^{\infty}(\|\hat{R}^{-1}\|_2\|\Delta\hat{R}_i\|_2)^k = \frac{\|\hat{R}^{-1}\|_2\|\Delta\hat{R}_i\|_2}{1 - \|\hat{R}^{-1}\|_2\|\Delta\hat{R}_i\|_2}. \tag{3.30}$$

Using (3.29) both in the denominator and the numerator yields

$$\|\breve{R}_i\|_2 \le \frac{1}{0.95}\cdot\frac{1.1n\sqrt{n}\mathbf{u}\|X\|_2}{\sqrt{(\sigma_n(X))^2 + 0.9s}} \le \frac{1.2n\sqrt{n}\mathbf{u}\|X\|_2}{\sqrt{(\sigma_n(X))^2 + 0.9s}}. \tag{3.31}$$

Together with the fact $\|\Delta x_i^\top\| \le \|x_i^\top\|\,\|\breve{R}_i\|_2$, we can bound $\|\Delta X\|_F$ as

(3.32)
$$\|\Delta X\|_F = \sqrt{\sum_{i=1}^{m} \|\Delta x_i^\top\|^2} \le \sqrt{\sum_{i=1}^{m} \|x_i^\top\|^2} \cdot \max_{1\le i\le m} \|\breve{R}_i\|_2 \le \frac{1.2n^2\mathbf{u}\|X\|_2^2}{\sqrt{(\sigma_n(X))^2 + 0.9s}},$$

where we used $\sqrt{\sum_{i=1}^{m} \|x_i^\top\|^2} = \|X\|_F \le \sqrt{n}\|X\|_2$ for the last inequality.

3.1.6. Bounding $\|\hat{Q}\|_2$. We now proceed to bound $\|\hat{Q}\|_2$.

LEMMA 3.1. *Suppose that $X \in \mathbb{R}^{m\times n}$ with $m \ge n$ satisfies* (3.2) *and* (3.3). *Then the matrix $\hat{Q}$ obtained by applying the shiftedCholeskyQR algorithm in floating-point arithmetic to X satisfies*

$$\|\hat{Q}^\top\hat{Q} - I\|_2 < 2,$$

and hence

$$\|\hat{Q}\|_2 < \sqrt{3}. \tag{3.33}$$

Proof. We have

$$\begin{aligned}\hat{Q}^\top\hat{Q} &= \hat{R}^{-\top}(X+\Delta X)^\top(X+\Delta X)\hat{R}^{-1}\\ &= I - \hat{R}^{-\top}(s+E_1+E_2)\hat{R}^{-1} + (X\hat{R}^{-1})^\top\Delta X\hat{R}^{-1} + \hat{R}^{-\top}\Delta X^\top(X\hat{R}^{-1})\\ &\quad + \hat{R}^{-\top}\Delta X^\top\Delta X\hat{R}^{-1}.\end{aligned}$$

Thus we can bound $\|\hat{Q}^\top\hat{Q} - I\|_2$ as

$$\begin{aligned}\|\hat{Q}^\top\hat{Q} - I\|_2 \le{}& \|\hat{R}^{-1}\|_2^2(s + \|E_1\|_2 + \|E_2\|_2) + 2\|\hat{R}^{-1}\|_2\|X\hat{R}^{-1}\|_2\|\Delta X\|_F\\ &+ \|\hat{R}^{-1}\|_2^2\|\Delta X\|_F^2.\end{aligned} \tag{3.34}$$

The first term of (3.34) can be bounded as

$$\|\hat{R}^{-1}\|_2^2(s + \|E_1\|_2 + \|E_2\|_2) \le \frac{1.1s}{(\sigma_n(X))^2 + 0.9s} \le \frac{1.1}{0.9}, \tag{3.35}$$

where we used (3.21). For the second term in (3.34), using (3.23), (3.24), and (3.32) we obtain

$$\begin{aligned}2\|\hat{R}^{-1}\|_2\|X\hat{R}^{-1}\|_2\|\Delta X\|_F &\le 2\cdot\frac{1}{\sqrt{(\sigma_n(X))^2 + 0.9s}}\cdot 1.5\cdot\frac{1.2n^2\mathbf{u}\|X\|_2^2}{\sqrt{(\sigma_n(X))^2 + 0.9s}}\\ &\le \frac{2\cdot 1.5\cdot 1.2\cdot\frac{1}{11}s}{0.9s} = \frac{4}{11}.\end{aligned} \tag{3.36}$$

For the third term in (3.34), from (3.23) and (3.32)

$$\begin{aligned}\|\hat{R}^{-1}\|_2^2\|\Delta X\|_F^2 &\le \frac{1}{(\sigma_n(X))^2 + 0.9s}\cdot\frac{(1.2n^2\mathbf{u}\|X\|_2^2)^2}{(\sigma_n(X))^2 + 0.9s}\\ &\le \frac{(1.2\cdot\frac{1}{11}s)^2}{(0.9s)^2} = \frac{16}{1089}.\end{aligned} \tag{3.37}$$

Summarizing, we can bound the right-hand side of (3.34) as $\|\hat{Q}^\top\hat{Q} - I\|_2 < 2$, as required. Hence, the eigenvalues of $\hat{Q}^\top\hat{Q} - I$ lie in $(-2, 2)$ and those of $\hat{Q}^\top\hat{Q}$ lie in $[0, 3)$ (since $\hat{Q}^\top\hat{Q}$ is positive semidefinite), which means that $\|\hat{Q}\|_2 < \sqrt{3}$. ∎

3.1.7. Bounding the residual in shiftedCholeskyQR. We now bound the residual.

LEMMA 3.2. *Under the assumptions in Lemma* 3.1*,* $\hat{Q}\hat{R}$ *computed by shiftedCholeskyQR satisfies*

$$\frac{\|\hat{Q}\hat{R} - X\|_F}{\|X\|_2} \le 2n^2\mathbf{u}.$$

Proof. First note that

$$\|\hat{q}_i^\top \hat{R} - x_i^\top\| = \|\hat{q}_i^\top \hat{R} - \hat{q}_i^\top(\hat{R} + \Delta\hat{R}_i)\| \le \|\hat{q}_i^\top \Delta\hat{R}_i\| \le \|\hat{q}_i^\top\|\|\Delta\hat{R}_i\|.$$

Substituting (3.28) into this gives

$$\|\hat{q}_i^\top \hat{R} - x_i^\top\| \le \|\hat{q}_i^\top\| \cdot 1.1n\sqrt{n}\mathbf{u}\|X\|_2. \tag{3.38}$$

On the other hand, from (3.33) we have

$$\|\hat{Q}\|_F < \sqrt{3n}. \tag{3.39}$$

Hence it follows that

$$\begin{aligned}\|\hat{Q}\hat{R} - X\|_F &= \sqrt{\sum_{i=1}^m \|\hat{q}_i^\top \hat{R} - x_i^\top\|^2} \le \sqrt{\sum_{i=1}^m \|\hat{q}_i^\top\|^2} \cdot 1.1n\sqrt{n}\mathbf{u}\|X\|_2 \\ &= \|\hat{Q}\|_F \cdot 1.1n\sqrt{n}\mathbf{u}\|X\|_2 \le 2n^2\mathbf{u}\|X\|_2. \end{aligned} \tag{3.40}$$

∎

Lemma 3.2 shows that shiftedCholeskyQR gives optimal residual up to a factor involving a low-degree polynomial of m, n (recall that $\|X\|_F \le \sqrt{n}\|X\|_2$) since even for $\hat{Q}$ and $\hat{R}$ obtained by rounding the true Q and R to floating-point matrices, $\|\hat{Q}\hat{R} - X\|_F = O(\mathbf{u})\|X\|_2$.

By (3.13) we have $\Delta X = \hat{Q}\hat{R} - X$, so (3.40) in fact provides a bound for $\|\Delta X\|_F$ that is tighter than the previous bound (3.32):

$$\|\Delta X\|_F \le 2n^2\mathbf{u}\|X\|_2. \tag{3.41}$$

3.2. Main result. We are now ready to state the main result of the section, which bounds the condition number of $\hat{Q}$.

THEOREM 3.3. *With shiftedCholeskyQR in double precision arithmetic applied to* X *satisfying* (3.1)–(3.3) *with shift* s *satisfying* (3.4)*, we obtain* $\hat{Q}$ *with*

$$\kappa_2(\hat{Q}) \le 2\sqrt{1 + \alpha(\kappa_2(X))^2} \cdot \sqrt{3}, \qquad \text{where} \quad \sqrt{\alpha} = \frac{\sqrt{s}}{\|X\|_2}. \tag{3.42}$$

Proof. Recall from (3.33) that $\sigma_1(\hat{Q}) < \sqrt{3}$. The remaining task is to bound $\sigma_n(\hat{Q})$ from below. Using Weyl's theorem in (3.13) gives

$$\sigma_n(\hat{Q}) \ge \sigma_n(X\hat{R}^{-1}) - \|\Delta X\hat{R}^{-1}\|_2. \tag{3.43}$$

Using (3.23) and (3.41) we obtain

$$\|\Delta X\hat{R}^{-1}\|_2 \le \|\Delta X\|_F\|\hat{R}^{-1}\|_2 \le \frac{2n^2\mathbf{u}\|X\|_2}{\sqrt{(\sigma_n(X))^2 + 0.9s}}. \tag{3.44}$$

Note that this is $O(\mathbf{u}^{\frac{1}{2}})$ when we take $s = O(\mathbf{u})$ and regard low-degree polynomials in m, n as constants.

We next bound $\sigma_n(X\hat{R}^{-1})$ from below. We proceed by examining the equation

$$\hat{R}^{-\top}\left(X^\top X + sI\right)\hat{R}^{-1} = I - \hat{R}^{-\top}(E_1 + E_2)\hat{R}^{-1}. \tag{3.45}$$

Let $X = U\Sigma V^\top$ be the SVD of X. Then, there exists a diagonal matrix G such that

$$X^\top X + sI = \left(U(\Sigma + G)V^\top\right)^\top \left(U(\Sigma + G)V^\top\right). \tag{3.46}$$

Indeed the left-hand side is $V(\Sigma^2 + sI)V^\top$ and the right-hand side $V(\Sigma + G)^2V^\top$, so we can take

$$G = (\Sigma^2 + sI)^{\frac{1}{2}} - \Sigma = \operatorname{diag}\left(\sqrt{(\sigma_i(X))^2 + s} - \sigma_i(X)\right). \tag{3.47}$$

Now setting $T = U(\Sigma + G)V^\top\hat{R}^{-1}$ we have

$$T^\top T = I - \hat{R}^{-\top}(E_1 + E_2)\hat{R}. \tag{3.48}$$

We next bound the singular values of T. By (3.21) and (3.23) we have

$$\|\hat{R}^{-\top}(E_1 + E_2)\hat{R}^{-1}\|_2 \le \|\hat{R}^{-1}\|_2^2(\|E_1\|_2 + \|E_2\|_2) \le \frac{0.1s}{(\sigma_n(X))^2 + 0.9s} \le \frac{1}{9}, \tag{3.49}$$

so $\sigma_i(T) \in [\sqrt{1 - \frac{1}{9}}, \sqrt{1 + \frac{1}{9}}] \subset [0.9, 1.1]$. Let $T = U'\Sigma'V'^\top$ be the SVD of T and write $\Sigma' = I + E'$. Then we have

$$T = U'(I + E')V'^\top = U'V'^\top\left(I + V'E'V'^\top\right) = Q(I + E), \tag{3.50}$$

where $Q = U'V'^\top$ has orthonormal columns, and $\|E\|_2 = \|V'E'V'^\top\|_2 = \|E'\|_2 \le 0.1$.

Now plugging into (3.50) the definition of T gives

$$Q(I + E) = U(\Sigma + G)V^\top\hat{R}^{-1}. \tag{3.51}$$

Recalling that $X = U\Sigma V^\top$, we have $\sigma_i(X\hat{R}^{-1}) = \sigma_i(\Sigma V^\top\hat{R}^{-1})$ and

$$\begin{aligned}\Sigma V^\top\hat{R}^{-1} &= \Sigma(\Sigma + G)^{-1}U^\top Q(I + E)\\ &= \operatorname{diag}\left(\frac{\sigma_i(X)}{\sqrt{(\sigma_i(X))^2 + s}}\right)U^\top Q(I + E).\end{aligned} \tag{3.52}$$

Using the general inequality for singular values of matrix products $\sigma_{\min}(AB) \ge \sigma_{\min}(A)\sigma_{\min}(B)$ (which holds when A or B is square) along with (3.22), we obtain

$$\sigma_n(X\hat{R}^{-1}) \ge \frac{\sigma_n(X)}{\sqrt{(\sigma_n(X))^2 + s}} \cdot 0.9. \tag{3.53}$$

Using this and (3.44), from (3.43) we obtain

$$\begin{aligned}\sigma_n(\hat{Q}) &\geq \frac{0.9\sigma_n(X)}{\sqrt{(\sigma_n(X))^2+s}} - \frac{2n^2\mathbf{u}\|X\|_2}{\sqrt{(\sigma_n(X))^2+0.9s}} \\ &\geq \frac{0.9\sigma_n(X)}{\sqrt{(\sigma_n(X))^2+s}} - \frac{2n^2\mathbf{u}\|X\|_2}{\sqrt{0.9}\sqrt{(\sigma_n(X))^2+s}} \\ &\geq \frac{0.9}{\sqrt{(\sigma_n(X))^2+s}}\left(\sigma_n(X) - \frac{2}{0.9\sqrt{0.9}}\cdot n^2\mathbf{u}\|X\|_2\right) \\ &\geq \frac{0.9}{\sqrt{(\sigma_n(X))^2+s}}(\sigma_n(X) - 0.4\sigma_n(X)) \\ &\geq \frac{\sigma_n(X)}{2\sqrt{(\sigma_n(X))^2+s}} = \frac{1}{2\sqrt{1+\alpha(\kappa_2(X))^2}}. \end{aligned} \tag{3.54}$$

We have used the assumption (3.1) for the fourth inequality. Together with (3.33) we obtain

$$\kappa_2(\hat{Q}) = \frac{\|\hat{Q}\|_2}{\sigma_n(\hat{Q})} \leq 2\sqrt{1+\alpha(\kappa_2(X))^2}\cdot\sqrt{3}. \tag{3.55}$$ ∎

Theorem 3.3 implies that, provided that $\alpha(\kappa_2(X))^2 \gg 1$,

$$\kappa_2(\hat{Q}) \lesssim 2\sqrt{3}\cdot\sqrt{\alpha}\kappa_2(X). \tag{3.56}$$

Thus applying one step of shiftedCholeskyQR results in the condition number being reduced by about a factor $\sqrt{\alpha} = \frac{\sqrt{s}}{\|X\|_2}$.

4. shiftedCholeskyQR3: shiftedCholeskyQR + CholeskyQR2. We now discuss an algorithm for the QR factorization of an ill-conditioned matrix that first uses shiftedCholeskyQR, then runs CholeskyQR2. We refer to this algorithm as shiftedCholeskyQR3, since it runs Cholesky QR (or its shifted variant) three times. The initial shiftedCholeskyQR can be regarded as a preconditioning step that reduces the condition number so that CholeskyQR2 becomes applicable. We continue to assume (3.1)–(3.3); a particular case of interest is $\mathbf{u}^{-1/2} < \kappa_2(X) < \mathbf{u}^{-1}$.

4.1. Choice of shift *s*. We first discuss the choice of the shift s for shiftedCholeskyQR, which balances two requirements:

- The shift s should be as small as possible to maximize the condition number improvement (3.42).
- The shift s should be large enough so that the Cholesky factorization chol$(A+sI)$ runs to completion without numerically breaking down.

In addition to these, s must satisfy (3.4) for the error analysis in the previous section to be valid.

To address the issue of breakdown we review Rump and Ogita's [19] error analysis for Cholesky factorization, which builds upon Demmel's early work [5].

Let A be a symmetric positive definite matrix. It is known [19, Theorem 2.3] that chol(A) succeeds numerically if the following holds:

$$\lambda_n(A) \geq \sum_{i=1}^{n} \frac{\gamma_{i+1}}{1-\gamma_{i+1}}\|A\|_2. \tag{4.1}$$

It has been shown in [25] that a sufficient condition for (4.1) to hold is to shift A by a positive quantity $\tilde{s}$ satisfying

$$\tilde{s} \geq c_{n+2}\mathbf{u}\mathrm{tr}(A), \quad c_{n+2} := \frac{(n+2)}{1-(n+1)(n+3)\mathbf{u}} < 2.2(n+1). \tag{4.2}$$

In our context of the Cholesky factorization (3.7), we need to take into account the error term E_1 in computing the matrix multiplication $A = X^\top X$. By (3.6) and Weyl's theorem we obtain a lower bound

$$\lambda_n(\hat{A}) \geq \lambda_n(X^\top X) - \|E_1\|_2. \tag{4.3}$$

This means that the positive definiteness of $\hat{A}$ cannot be guaranteed if $\lambda_n(X^\top X) \leq \gamma_m n\|X\|_2^2$. Accordingly, to apply formula (4.2), we must first shift $\hat{A}$ by $\gamma_m n\|X\|_2^2$. Thus we conclude that a safe choice of s to avoid numerical breakdown is

$$\tilde{s} := \gamma_{m+1} n\|X\|_2^2 + c_{n+2}\mathbf{u}\mathrm{tr}(\hat{A} + \gamma_{m+1} n\|X\|_2^2 I). \tag{4.4}$$

By further taking into account (3.4), we have

$$s := \max\left(11(mn + n(n+1))\mathbf{u}\|X\|_2^2, \tilde{s}\right). \tag{4.5}$$

This expression can be simplified by evaluating $\tilde{s}$ using the results in the previous section. First, we note that $\hat{A} = X^\top X + E_1$ from (3.6). Denoting the jth column vector of X by $\tilde{\mathbf{x}}_j$, we can evaluate $\mathrm{tr}(E_1)$ as

$$\mathrm{tr}(E_1) \leq \sum_{i=1}^{n} |E_1|_{ii} \leq \sum_{i=1}^{n} \gamma_m |\tilde{\mathbf{x}}_i|^\top |\tilde{\mathbf{x}}_i| = \gamma_m \sum_{i=1}^{n} \|\tilde{\mathbf{x}}_i\|^2 = \gamma_m \|X\|_F^2 \leq \gamma_m n\|X\|_2^2. \tag{4.6}$$

On the other hand,

$$\mathrm{tr}(X^\top X) = \|X\|_F^2 \leq n\|X\|_2^2. \tag{4.7}$$

Plugging these into the right-hand side of (4.4), we can bound $\tilde{s}$ as

$$\begin{aligned} \tilde{s} &\leq \gamma_{m+1} n\|X\|_2^2 + 2.2(n+1)\mathbf{u}(n\|X\|_2^2 + \gamma_m n\|X\|_2^2 + \gamma_{m+1} n\|X\|_2^2) \\ &\leq 2.4(mn + n(n+1))\mathbf{u}\|X\|_2^2, \end{aligned} \tag{4.8}$$

which shows that the maximum in (4.5) is attained by the first argument. Hence, in what follows we shall assume

$$s := 11(mn + n(n+1))\mathbf{u}\|X\|_2^2. \tag{4.9}$$

In practice, since $\|X\|_2$ is expensive to compute we can estimate it reliably using a norm estimator, e.g., MATLAB's function `normest`, or alternatively just replace it with $\|X\|_F$, which results in a larger (more conservative) shift.

We now summarize our algorithm in pseudocode. Algorithm 4.1 blends shiftedCholeskyQR and CholeskyQR2 in an adaptive manner, initially attempting to reduce the condition number using shiftedCholeskyQR as in (3.56) so that CholeskyQR2 becomes applicable. The shifts are introduced only when necessary, judged by whether or not the Cholesky factorization $\mathrm{chol}(A^{(k)})$ breaks down. Our experiments suggest that Algorithm 4.1 may well be applicable even to extremely ill-conditioned matrices with possibly $\kappa_2(X) > \mathbf{u}^{-1}$.

Algorithm 4.1. Iterated Cholesky QR for $X = QR$ with shifts when necessary.

Let $Q := X, R := I$
repeat
  $A := Q^\top Q$
  $\tilde{R} := \mathrm{chol}(A)$ //chol: Cholesky factorization
  if chol(A) breaks down **then**
    $s := 11(mn + n(n+1))\mathbf{u}\|X\|_2^2$ //introduce shift
    $\tilde{R} := \mathrm{chol}(A + sI)$
  end if
  $Q := Q\tilde{R}^{-1}$, $R := \tilde{R}R$
until $\|Q^TQ - I\|_F \le \sqrt{n}\mathbf{u}$

Algorithm 4.2. shiftedCholeskyQR3 for $X = QR$.

Let $Q := X$
$A := Q^\top Q$
$s := 11(mn + n(n+1))\mathbf{u}\|X\|_2^2$ //introduce shift
$R := \mathrm{chol}(A + sI)$ // shiftedCholeskyQR
$Q := Q\tilde{R}^{-1}$
$\tilde{R} := \mathrm{chol}(Q^\top Q)$, $Q := Q\tilde{R}^{-1}$, $R := \tilde{R}R$ // Cholesky QR
$\tilde{R} := \mathrm{chol}(Q^\top Q)$, $Q := Q\tilde{R}^{-1}$, $R := \tilde{R}R$ // CholeskyQR2

In the analysis below, we focus on the case $\kappa_2(X) < \mathbf{u}^{-1}$; to be precise, when $\mathbf{u}^{-1/2} < \kappa_2(X)$ (so that CholeskyQR2 is inapplicable) and $\kappa_2(X)$ is bounded from above by (4.12) given below. For such matrices, the algorithm provenly runs one shiftedCholeskyQR, then CholeskyQR2 (thus executing three Cholesky factorizations). This computes a stable QR factorization, and we refer to this algorithm as shiftedCholeskyQR3. For completeness we present its pseudocode in Algorithm 4.2. The last two lines represent CholeskyQR2 for the Q obtained by the first shiftedCholeskyQR.

4.2. When is thrice enough? Here we derive a condition on $\kappa_2(X)$ that guarantees that shiftedCholeskyQR3 gives a numerically stable QR factorization of X.

Recall that (3.55) gives a bound for the condition number of $\hat{Q}$ obtained by shiftedCholeskyQR: $\kappa_2(\hat{Q}) \le 2\sqrt{3(1+\alpha(\kappa_2(X))^2)}$. As in (4.9), to guarantee avoidance of breakdown we take $\alpha = 11(mn+n(n+1))\mathbf{u}$, so the condition number of $\hat{Q}$ is bounded as

$$\kappa_2(\hat{Q}) \le 2\sqrt{3}\sqrt{1 + 11(mn + n(n+1))\mathbf{u}(\kappa_2(X))^2}. \tag{4.10}$$

On the other hand, as shown in [23], a sufficient condition for CholeskyQR2 to compute a stable QR factorization of $\hat{Q}$ is

$$\kappa_2(\hat{Q}) \le \frac{1}{8\sqrt{(mn + n(n+1))\mathbf{u}}}. \tag{4.11}$$

Combining these facts, we obtain the following condition under which shiftedCholeskyQR3 is guaranteed to compute a numerically stable QR factorization:

$$2\sqrt{3}\sqrt{1 + 11(mn + n(n+1))\mathbf{u}(\kappa_2(X))^2} \le \frac{1}{8\sqrt{(mn + n(n+1))\mathbf{u}}}.$$

If $\kappa_2(X)$ is greater than $\mathbf{u}^{-\frac{1}{2}}$, we have $1 + 11(mn + n(n+1))\mathbf{u}(\kappa_2(X))^2 \simeq 11(mn + n(n+1))\mathbf{u}(\kappa_2(X))^2$, and the condition can be simplified as

$$\kappa(X) \leq \frac{\mathbf{u}^{-1}}{96(mn + n(n+1))}. \tag{4.12}$$

Note that this ensures that the condition (3.1) for the error analysis in section 3 is automatically satisfied.

In practice, it often happens that shiftedCholeskyQR3 (or more often the iterated Algorithm 4.1) computes the QR factorization for matrices with even larger condition numbers than indicated by (4.12). One explanation is that in the absence of roundoff errors, one iteration of shiftedCholeskyQR reduces the condition number by a factor $\approx \mathbf{u}^{\frac{1}{2}}$. However, we think that a rigorous convergence analysis in finite precision arithmetic would be possible only under some assumption on $\kappa_2(X)$ and that (4.12) provides a sharp bound up to (at most) a low-degree polynomial in m, n.

4.3. Numerical stability of shiftedCholeskyQR3. We now examine the numerical stability of shiftedCholeskyQR3 and show that it enjoys excellent stability both in orthogonality and backward error. Roughly, the result follows by combining the facts that (i) shiftedCholeskyQR gives a $\hat{Q}$ with $\kappa_2(\hat{Q}) < \mathbf{u}^{-1/2}$ with small backward error, and (ii) for matrices with condition number $< \mathbf{u}^{-1/2}$, CholeskyQR2 computes a stable QR factorization of $\hat{Q}$ as shown in [23]. Below we make this statement precise.

THEOREM 4.1. *Let $X \in \mathbb{R}^{m\times n}$ be a matrix satisfying* (3.1)–(3.3) *and* (4.12). *Then shiftedCholeskyQR3 computes a QR factorization $X \approx \hat{Q}\hat{R}$ satisfying the orthogonality measure*

$$\|\hat{Q}^\top \hat{Q} - I\|_F \leq 6(mn + n(n+1))\mathbf{u} \tag{4.13}$$

and backward error

$$\frac{\|\hat{Q}\hat{R} - X\|_F}{\|X\|_2} \leq 15n^2\mathbf{u}. \tag{4.14}$$

Proof. The orthogonality measure of the output $\hat{Q}$ of shiftedCholeskyQR3 is essentially exactly the same as that of CholeskyQR2, which is analyzed in detail in [23]. This is because the bound there applies to any matrix with condition number $\lesssim \mathbf{u}^{-\frac{1}{2}}$.

We next establish (4.14). By (3.13), with the first shiftedCholeskyQR executed in finite precision arithmetic we have

$$X + \Delta X = \hat{Q}\hat{R}, \tag{4.15}$$

where $\|\Delta X\|_F$ is bounded as in (3.41). We then apply CholeskyQR2 to $\hat{Q}$ to obtain the QR factorization $\hat{Q} = ZU$. In finite precision we have

$$\hat{Q} + \Delta\hat{Q} = \hat{Z}\hat{U}, \tag{4.16}$$

where (see Appendix in the supplementary materials; this bound slightly improves [23])

$$\|\Delta\hat{Q}\|_F \leq 5n^2\mathbf{u}\|\hat{Q}\|_2. \tag{4.17}$$

Let S be the upper triangular factor in the QR factorization of the original matrix X and $\hat{S}$ be its computed counterpart. Then,

$$\hat{S} = fl(\hat{U}\hat{R}) = \hat{U}\hat{R} + \Delta S. \tag{4.18}$$

Here ΔS represents the forward error incurred in the matrix multiplication.

Summarizing, we can bound the overall backward error as

$$\begin{aligned} \|\hat{Z}\hat{S} - X\|_F &= \|\hat{Z}(\hat{U}\hat{R} + \Delta S) - \hat{Q}\hat{R} + \Delta X\|_F \\ &= \|\Delta\hat{Q}\hat{R} + \hat{Z}\Delta S + \Delta X\|_F \\ &\le \|\Delta\hat{Q}\|_F\|\hat{R}\|_2 + \|\hat{Z}\|_2\|\Delta S\|_F + \|\Delta X\|_F. \end{aligned} \tag{4.19}$$

We now bound the terms in the right-hand side. For the first term, using [23, Theorem 3.5] and (3.33) we can bound $\|\Delta\hat{Q}\|_F$ as

$$\|\Delta\hat{Q}\|_F \le 5n^2\mathbf{u}\|\hat{Q}\|_2 \le 5\sqrt{3}n^2\mathbf{u}. \tag{4.20}$$

To bound $\|\hat{R}\|_2$, we use (3.27) to obtain

$$\|\hat{R}\|_2 \le \sqrt{1.1}\|X\|_2. \tag{4.21}$$

We next bound the second term in (4.19). By [23, Theorem. 3.3], $\|\hat{Z}\|_2$ can be bounded as

$$\|\hat{Z}\|_2 \le \sqrt{1 + 6(mn\mathbf{u} + n(n+1)\mathbf{u})} \le \sqrt{1 + 6\left(\frac{1}{64} + \frac{1}{64}\right)} = \frac{\sqrt{76}}{8}. \tag{4.22}$$

Regarding $\|\Delta S\|_F$, using the general error bound for matrix multiplications $|\Delta S| \le \gamma_n|\hat{U}|\,|\hat{R}|$ we obtain

$$\|\Delta S\|_F \le \gamma_n\||\hat{U}|\,|\hat{R}|\| \le \gamma_n\|\hat{U}\|_F\|\hat{R}\|_F \le n\gamma_n\|\hat{U}\|_2\|\hat{R}\|_2. \tag{4.23}$$

Here $\|\hat{R}\|_2$ can be bounded as in (4.21). To bound $\|\hat{U}\|_2$, we recall (4.16) and left-multiply $\hat{Z}^\top$ to obtain

$$\hat{Z}^\top(\hat{Q} + \Delta\hat{Q}) = \hat{Z}^\top\hat{Z}\hat{U}. \tag{4.24}$$

Now, by [23, Theorem 3.3], the eigenvalues of $\hat{Z}^\top\hat{Z}$ lie in the interval $[1 - 6(mn\mathbf{u} + n(n+1)\mathbf{u}), 1 + 6(mn\mathbf{u} + n(n+1)\mathbf{u})]$, so it follows that

$$\begin{aligned} \|\hat{U}\|_2 &\le \|(\hat{Z}^\top\hat{Z})^{-1}\|_2\|\hat{Z}^\top\|_2(\|\hat{Q}\|_2 + \|\Delta\hat{Q}\|_2) \\ &\le \frac{1}{1 - 6(mn\mathbf{u} + n(n+1)\mathbf{u})} \cdot \frac{\sqrt{76}}{8}\left(\sqrt{3} + 5\sqrt{3}n^2\mathbf{u}\right) \\ &\le \frac{1}{1 - 6\left(\frac{1}{64} + \frac{1}{64}\right)} \cdot \frac{\sqrt{76}}{8}\left(\sqrt{3} + 5\sqrt{3} \cdot \frac{1}{64}\right) \le 2.6. \end{aligned} \tag{4.25}$$

Finally, we can bound $\|\Delta X\|_F$ as in (3.41).

Combining the above bounds and substituting into (4.19) yields

$$\begin{aligned} \|\hat{Z}\hat{S} - X\|_F &\le 5\sqrt{3}n^2\mathbf{u} \cdot \sqrt{1.1}\|X\|_2 + \frac{\sqrt{76}}{8} \cdot n \cdot 1.02n\mathbf{u} \cdot 2.6 \cdot \sqrt{1.1}\|X\|_2 + 2n^2\mathbf{u}\|X\|_2 \\ &\le 15n^2\mathbf{u}\|X\|_2, \end{aligned} \tag{4.26}$$

as required. ∎

Comparison with classical Gram–Schmidt twice (CGS2). As we saw above, (4.12) is a sufficient condition for shiftedCholeskyQR3 to work in finite precision arithmetic. This condition roughly requires that $\kappa_2(X)(mn+n^2)\mathbf{u} = O(1)$. Let us compare this with the analysis in [9] for the CGS2 algorithm, which shows that

$$\kappa_2(X)m^2n^3\mathbf{u} = O(1) \tag{4.27}$$

is a sufficient condition for CGS2 to compute the QR factorization in a stable manner.

Observe that (4.27) is much more stringent than (4.12); indeed in large-scale computing in which $m \geq 10^6$ and $n \geq 100$, with double precision (4.27) is unlikely to be satisfied even with well-conditioned X.

This difference might appear to suggest shiftedCholeskyQR3 is superior to CGS2 in terms of robustness, but we have not observed this in practice. We suspect that the difference is an artifact of the analysis, and the practical robustness of CGS2 and shiftedCholeskyQR3 seem comparable. An advantage of shiftedCholeskyQR3 is that it is rich in BLAS-3 operations and offers ample opportunity for parallelization.

5. Oblique inner product. The (shifted) Cholesky QR algorithm is readily applicable to the QR decomposition in a nonstandard inner product space $(x, y)_B = x^\top By$ defined via a symmetric positive definite matrix $B \in \mathbb{R}^{m\times m}$. The resulting algorithm is almost identical to Algorithm 2.1, except that A is computed as $A = X^\top BX$ and the shift s is chosen in a manner to be described below. Based on this, we can also formulate the shiftedCholeskyQR3 algorithm for the nonstandard inner product, as shown Algorithm 5.1.

Algorithm 5.1. shiftedCholeskyQR3 for $X = QR$, $Q^\top BQ = I_n$.

1: Let $Q := X$
2: $A := Q^\top BQ$
3: choose $s > 0$
4: $R := \mathrm{chol}(A + sI)$
5: $Q := QR^{-1}$
6: $\tilde{R} := \mathrm{chol}(Q^\top BQ)$, $Q := Q\tilde{R}^{-1}$, $R := \tilde{R}R$
7: $\tilde{R} := \mathrm{chol}(Q^\top BQ)$, $Q := Q\tilde{R}^{-1}$, $R := \tilde{R}R$

The stability of Algorithm 5.1 can be analyzed in a similar way as in the standard inner product case presented in the previous section, but new features arise that affect the bounds, in particular involving $\sqrt{\kappa_2(B)}$. Here, due to space limitation, we present only the final results. Details of the analysis can be found in [7] or the supplementary materials.

We make the following assumptions on m, n, X, and B. As in the case of standard inner product, the constants below are not of significant importance but chosen so that the analysis goes through.

$$\frac{\|X\|_2\sqrt{\|B\|_2}}{\sqrt{\sigma_n(X^\top BX)}} \cdot \sqrt{\kappa_2(B)} \leq \frac{\mathbf{u}^{-1}}{96(2m\sqrt{mn}+n(n+1))}, \tag{5.1}$$

$$\kappa_2(B) \leq \frac{\mathbf{u}^{-1}}{80(m\sqrt{mn}+n(n+1))}. \tag{5.2}$$

The assumption (5.1) roughly demands that X is not too ill-conditioned relative to $\mathbf{u}$. Instead of (5.1) and (5.2), we can use a simpler assumption

$$\kappa_2(X)\kappa_2(B) \le \frac{\mathbf{u}^{-1}}{96(2m\sqrt{mn}+n(n+1))}, \tag{5.3}$$

but the combination of (5.1) and (5.2) is less stringent. We note that (5.1) imposes that X has full column rank.

We now state our main results, which show that shiftedCholeskyQR3 in the oblique inner product enjoys excellent stability both in orthogonality and backward error.[1]

THEOREM 5.1. *Let $B \in \mathbb{R}^{m\times m}$ be positive definite and $X \in \mathbb{R}^{m\times n}$ be a matrix satisfying* (5.1) *and* (5.2). *Moreover, let the shift s be chosen as*

$$s := 11\left(2m\sqrt{mn}+n(n+1)\right)\mathbf{u}\|X\|_2^2\|B\|_2. \tag{5.4}$$

Then shiftedCholeskyQR followed by CholeskyQR2 computes a QR factorization $X = QR$ satisfying the B-orthogonality measure

$$\|\hat{Q}^\top B\hat{Q} - I\|_F \le 8[m\sqrt{mn}\mathbf{u} + n(n+1)\mathbf{u}]\kappa_2(B), \tag{5.5}$$

and the backward error

$$\frac{\|\hat{Q}\hat{R} - X\|_F}{\|\hat{X}\|_2} \le 16n^2\mathbf{u}(\kappa_2(B))^{3/2}. \tag{5.6}$$

Experiments indicate that the bounds in Theorem 5.1 are overestimates, in particular the dependence on $\kappa_2(B)$ appears to be much weaker.

6. Numerical experiments. In this section we present some numerical experiments to illustrate our results. All computations were carried out on MATLAB 2017b and IEEE standard 754 binary64 (double precision) in Mac OS X version 10.13 with 2 GHz Intel Core i7 Duo processor, so that $\mathbf{u} = 2^{-53} \approx 1.11 \times 10^{-16}$.

6.1. Convergence with CholeskyQR iterates. First, we take $B = I$ and examine how $\kappa_2(\hat{Q}^{(k)})$ and $\|\hat{Q}^{(k)\top}\hat{Q}^{(k)} - I\|_2$ are reduced after k (shifted)Cholesky QR steps. We also compare shiftedCholeskyQR3 with the mixed-precision Cholesky QR (mixedCholQR) [24], which uses doubled precision for the first two steps (1.1), (1.2) and repeats the process in double precision. To run mixedCholQR we used the Multiprecision Computing Toolbox [17], which enables computation in MATLAB with arbitrary precision. We generate test matrices with a specified condition number by forming

$$X := U\Sigma V^T \in \mathbb{R}^{m\times n}, \tag{6.1}$$

where U is an $m \times n$ random orthogonal matrix obtained by taking the QR factorization of a random matrix, V is an $n \times n$ random orthogonal matrix, and

$$\Sigma = \operatorname{diag}\left(1, \sigma^{\frac{1}{n-1}}, \ldots, \sigma^{\frac{n-2}{n-1}}, \sigma\right).$$

[1] The following theorem corresponds to Theorem 5.4 in the supplementary materials. Note that the assumptions (5.1), (5.2), and (5.3) in the supplementary materials are automatically satisfied if the assumptions (5.1) and (5.2) above hold.

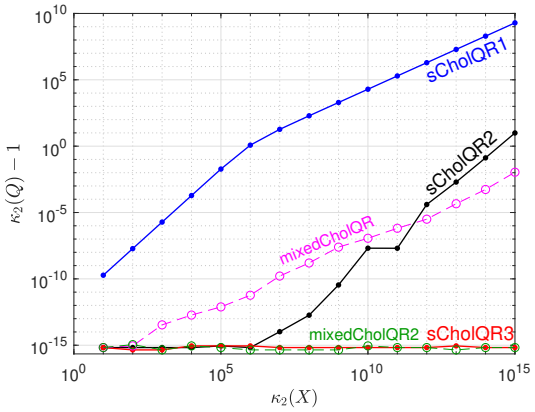

FIG. 6.1. $\kappa_2(X)$ *vs.* $\kappa_2(Q)-1$ *for* Q *computed with various algorithms. Here we use* $\kappa_2(Q)-1$ *to monitor convergence instead of* $\|Q^TQ-I\|_2$*, because we have* $\|Q^TQ-I\|_2 \approx 1$ *whenever* $\kappa_2(Q) \gg 1$*, making it difficult to understand the behavior of sCholQR1 with* $\|Q^TQ-I\|_2$ *when* $\kappa_2(X) \geq 10^6$*.*

Here, $0 < \sigma < 1$ is some constant. This is essentially MATLAB's `randsvd` construction. Thus $\|X\|_2 = 1$ and the 2-norm condition number of X is $\kappa_2(X) = 1/\sigma$. Let k denote the number of iterations. Figure 6.1 shows the results of computing the QR factorization using Algorithm 4.1 (shiftedCholeskyQR3) and mixedCholQR, tracing the convergence of $\kappa_2(\hat{Q}^{(k)})$ as we take more iterations. We took $m = 100, n = 30$, and varied $\kappa_2(X)$.

The figure illustrates that the conditioning $\kappa_2(\hat{Q}^{(1)})$ is improved by shiftedCholeskyQR to approximately $\mathcal{O}(\sqrt{\alpha})\kappa_2(X) \lesssim \mathbf{u}^{-1/2}$ (the line denoted sCholQR1). This is consistent with Theorem 3.3 (see (3.56)). Then, CholeskyQR2 can safely compute the QR factorization of $\hat{Q}^{(1)}$. We also see that $\kappa_2(\hat{Q}^{(1)}) \approx \mathbf{u}\kappa_2(X) = \mathcal{O}(1)$ in mixedCholQR, which is consistent with Theorem 3.4 in [24]. Here shiftedCholeskyQR3 requires one more iteration than mixedCholQR, which is a typical behavior and reflects the theory for $\kappa_2(X) \in \left[\mathbf{u}^{-1/2}, \mathbf{u}^{-1}\right]$. shiftedCholeskyQR3 has the advantage over mixedCholQR that no high-precision arithmetic is needed, thereby being much faster in practice; indeed here it was faster by orders of magnitude.

We next turn to the case where $B \neq I$ and B is an SPD matrix, and we examine the convergence of B-orthogonality of Q via $\kappa_2(B^{\frac{1}{2}}Q)$. We set X to be a random matrix with a specified condition number as in (6.1) and form a positive definite matrix B as in (6.1), now taking $V = U$. Figure 6.2 is analogous to Figure 6.1 but with $\kappa_2(B) = 10$ (left) and $\kappa_2(B) = 10^8$ (right). For the $\kappa_2(B) = 10^8$ case, both shiftedCholeskyQR3 and mixedCholQR broke down for $\kappa_2(B^{\frac{1}{2}}X) > 10^8$ (respectively, in the second (unshifted) and first Cholesky factorization). The situation is similar to the $B = I$ case, especially when B is not ill-conditioned. When $\kappa_2(B) \gg 1$, both the final orthogonality and robustness appear to deteriorate accordingly, as suggested by Theorem 5.1. In any case, if the first shiftedCholeskyQR reduces the condition number sufficiently, CholeskyQR2 then safely completes the QR factorization.

The choice of s in (4.9) (and (5.4)) tends to be a conservative overestimate, and in most cases, a successful Cholesky factorization can be computed with a smaller shift, such as $s = \mathbf{u}\|X\|_2^2\|B\|_2$. It can be seen (if Cholesky still does not break down) that the reduction factor of $\kappa_2(\hat{Q}^{(k)})$ improves to about $(\sqrt{\mathbf{u}})^k$ after k shiftedCholeskyQR steps. To illustrate this, in Figure 6.3 we show the values of $\kappa_2(\hat{Q}^{(1)})$ as we vary the shift in shiftedCholeskyQR taking $B = I, m = 1000$, $n = 50$, $\kappa_2(X) = 10^{15}$.

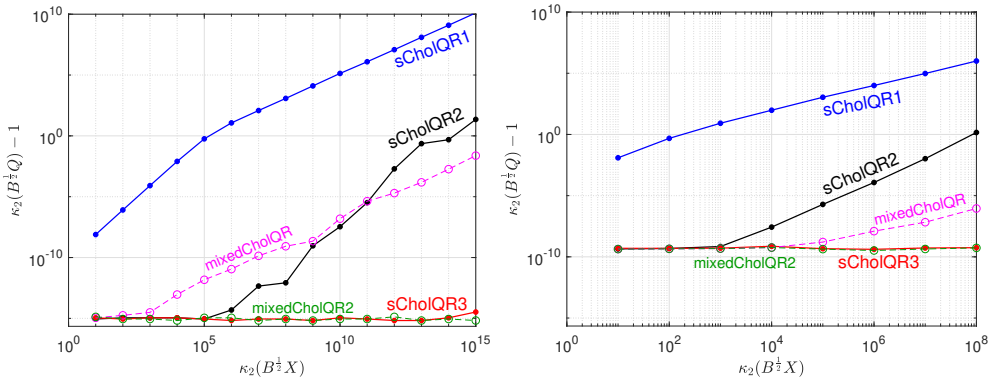

FIG. 6.2. *Convergence of* $\kappa_2(B^{\frac{1}{2}}Q)$ *as the iterates proceed for* $\kappa_2(B) = 10$ *(left) and* $\kappa_2(B) = 10^8$ *(right).*

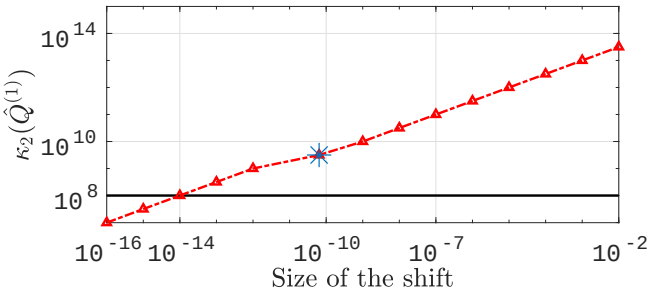

FIG. 6.3. $\kappa_2(\hat{Q}^{(1)})$ *for test matrix with* $\kappa_2(X) = 10^{15}$, $m = 1000$, $n = 50$, *varying the shift of shiftedCholeskyQR.*

Our "safe" choice $s := 11(mn + n(n+1))\mathbf{u}\|X\|_2^2 \approx 6.1 \cdot 10^{-11}$ is shown in Figure 6.3 by an asterisk. In this case, $\kappa_2(\hat{Q}^{(1)})$ is larger than required by (4.11), and more shiftedCholeskyQR iterations would be needed before CholeskyQR2 can be applied. This is because here $\kappa_2(X)$ is larger than the bound (4.12), so shiftedCholeskyQR3 with our default shift choice is not guaranteed to succeed. On the other hand, if we set $s := \mathbf{u}\|X\|_2^2$, then $\hat{Q}^{(1)}$ will satisfy the sufficient condition $\kappa_2(\hat{Q}^{(1)}) \le \mathbf{u}^{-\frac{1}{2}}$ for CholeskyQR2 to work. Note however that there is no guarantee that the initial Cholesky factorization $\mathrm{chol}(A + \mathbf{u}\|X\|_2^2 I)$ does not break down.

6.2. Orthogonality and residual. Next we examine the numerical stability of shiftedCholeskyQR3 (shown in the figures as sCholQR3) and compare it with other popular QR decomposition algorithms, namely, Householder QR, CGS, and MGS (we also run them twice, CGS2 and MGS2). We first take $B = I$ and vary $\kappa_2(X)$, m, and n and investigate the dependence of the orthogonality and residual on them. We set X as in (6.1). We examine the orthogonality and residual measured by the Frobenius norm in Figure 6.4. Figure 6.4 shows the orthogonality $\|\hat{Q}^T\hat{Q} - I\|_F$ and residual $\|\hat{Q}\hat{R} - X\|_F$, where we take $m = 300$, $n = 10$ and $\kappa_2(X)$ varies from 10^8 to 10^{15}.

We see in Figure 6.4 that with shiftedCholeskyQR3, the orthogonality and the residual are independent of $\kappa_2(X)$ and are of $O(\mathbf{u})$, as long as $\kappa_2(X)$ is at most $O(\mathbf{u}^{-1})$. This is in good agreement with Theorem 4.1. Our experiments (not shown) indicate

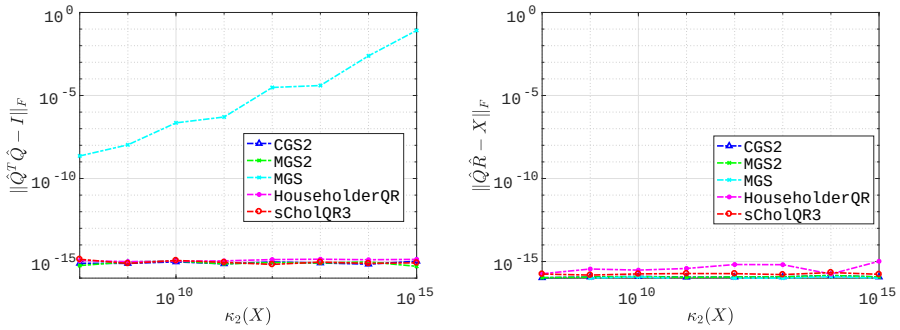

FIG. 6.4. *Orthogonality* $\|\hat{Q}^\top\hat{Q}-I\|_F$ *and residual* $\|\hat{Q}\hat{R}-X\|_F$ *for test matrices with* $m = 300$, $n = 10$, *varying* $\kappa_2(X)$.

that the orthogonality and residual increase only mildly with m and n. Although they are inevitably overestimates, these also reflect our results (4.13) and (4.14). Compared with Householder QR, we observe that shiftedCholeskyQR3 usually produces slightly better orthogonality and residual. With MGS, the deviation from orthogonality increases proportionally to $\kappa_2(X)$. It is well known that Gram–Schmidt type algorithms perform well when repeated twice, and we can verify this here. As mentioned in section 4.3, an advantage of shiftedCholeskyQR3 is that it is rich in BLAS-3 operations and easily parallelized. Overall, we see that shiftedCholeskyQR3 is a reliable method for matrices with condition number at most $O(\mathbf{u}^{-1})$.

Next, we again take $B \neq I$ and test Algorithm 5.1, comparing it with the stability of other popular QR decomposition algorithms, namely, MGS, CGS2, and MGS2. We varied $\kappa_2(B)$, m, and n and investigated the orthogonality $\|\hat{Q}^T B\hat{Q} - I\|_F$ and residual $\|\hat{Q}\hat{R} - X\|_F$. Figure 6.5 shows the results for the case $m = 500$, $n = 20$, $\kappa_2(B) = 10^{10}$ and $\kappa_2(X)$ was varied from 10^1 to 10^{10}. Figure 6.6 takes $m = 300$, $n = 50$, $\kappa_2(X) = 10^{10}$, and $\kappa_2(B)$ was varied from 10^8 to 10^{15}. Figure 6.7 takes $\kappa_2(X) = 10^8$, $\kappa_2(B) = 10^{10}$, $m = 1000$, and n was varied from 50 to 500.

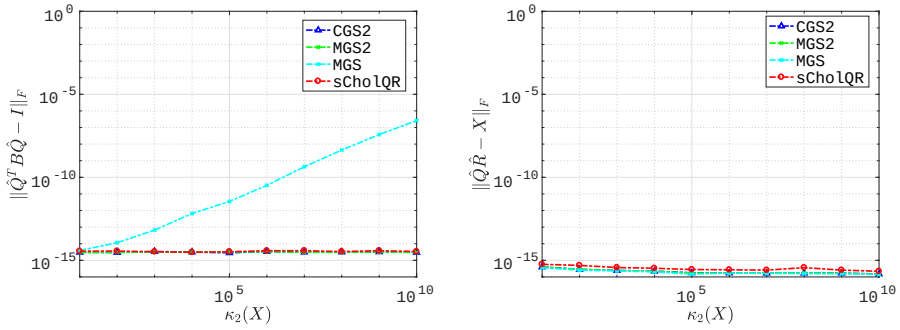

FIG. 6.5. *Orthogonality* $\|\hat{Q}^\top B\hat{Q}-I\|_F$ *and Residual* $\|\hat{Q}\hat{R}-X\|_F$ *for test matrices with* $m = 500$, $n = 20$, $\kappa_2(B) = 10^{10}$, *varying* $\kappa_2(X)$.

From Figure 6.5 we see that the orthogonality and residual of shiftedCholeskyQR3 are independent of $\kappa_2(X)$ and are of $O(\mathbf{u})$, as long as $\|X\|_2\sqrt{\|B\|_2}/\sqrt{\sigma_n(X^\top BX)} \lesssim O(\mathbf{u}^{-1})$, reflecting Theorem 5.1. Figure 6.6 shows that the orthogonality increase rather mildly with $\kappa_2(B)$, indicating the dependence on $\kappa_2(B)$ suggested by (5.5) is

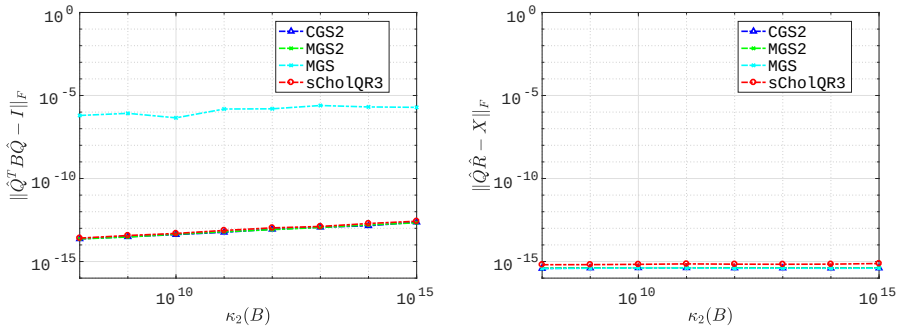

FIG. 6.6. *Orthogonality* $\|\hat{Q}^\top B\hat{Q} - I\|_F$ *and Residual* $\|\hat{Q}\hat{R} - X\|_F$ *for test matrices with* $m = 300$, $n = 50$, $\kappa_2(X) = 10^{10}$, *varying* $\kappa_2(B)$.

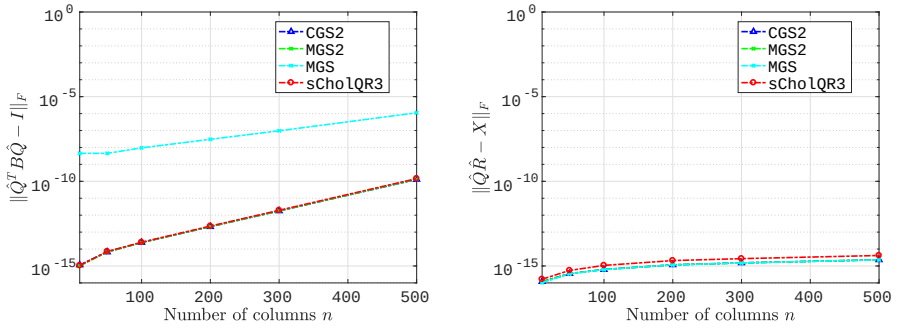

FIG. 6.7. *Orthogonality* $\|\hat{Q}^\top B\hat{Q} - I\|_F$ *and Residual* $\|\hat{Q}\hat{R} - X\|_F$ *for test matrices with* $\kappa_2(X) = 10^8$, $\kappa_2(B) = 10^{10}$, $m = 1000$, *varying* n.

perhaps improvable; we leave this for future work. Figure 6.7 illustrates the orthogonality and residual increase with n only mildly (the influence of m is even weaker). These are also in agreement with (5.5) and (5.6) in Theorem 5.1; again, its m, n-dependence may be improvable. Compared with CGS2 and MGS2, the orthogonality and residual of CGS2 and MGS2 and that of shiftedCholeskyQR3 are of the same magnitude. Again, shiftedCholeskyQR3 has the advantage of being parallelization-friendly. All our experiments corroborate that shiftedCholeskyQR3 is a reliable method whether $B = I$ or $B \neq I$, for matrices with $\|X\|_2\sqrt{\|B\|_2}/\sqrt{\sigma_n(X^\top BX)} < O(\mathbf{u}^{-1})$.

7. Runtime Performance. We next evaluate the runtime performance of shiftedCholeskyQR3 in multicore CPU environments for both the standard and the oblique case involving a sparse symmetric, positive definite matrix B. Throughout this section, unless otherwise specified we used a compute node of the Laurel 2 supercomputer system installed at the Academic Center for Computing and Media Studies, Kyoto University, whose specifications are listed in Table 7.1. We note that only shared memory parallelism is employed here. The code is written in Fortran90 and uses LAPACK and BLAS routines. We used the Intel ifort compiler version 17.0.6 and BLAS and LAPACK libraries implemented in Intel MKL version 2017.0.6.

7.1. Standard inner product. Here we focus on the standard $B = I$ case, to facilitate comparison with available algorithms. Test matrices are generated as in the

TABLE 7.1
Specifications of the Laurel 2 system.

Item	Specification
CPU	Intel Xeon E5-2695 v4 (Broadwell, 2.1 GHz, 18 cores)
Number of CPUs / node	2
Number of threads	36
Memory size / node	128 GB
Peak FLOPS / node	1.21 TFLOPS (in double precision)
Compiler	Intel `ifort` ver. 17.0.6
Compile options	`-mcmodel=medium`, `-shared-intel`, `-qopenmp` `-O3`, `-ipo`, `-xHost`
BLAS, LAPACK	Intel MKL ver. 2017.0.6 (`-mkl=parallel`)

previous experiments, using (6.1). Here, random orthogonal matrices are obtained by applying the LAPACK Householder QR routines (`dgeqrf` and `dorgqr`) to a random matrix.

Table 7.2 presents the computational time of several methods, where the QR factorization of matrices whose condition number is 10^{11} is computed. Here, we compare shiftedCholeskyQR3 (`dgemm` and `dsyrk` versions), Householder QR, and CGS2 (including its blocked version). It is worth noting that `dgeqr` is a novel LAPACK routine that appropriately uses the TSQR algorithm. The block width in Block CGS2 was empirically tuned. Table 7.2 clearly shows that shiftedCholeskyQR3 (both `dgemm` and `dsyrk` versions) outperforms other methods for all cases. Among methods besides shiftedCholeskyQR3, `dgeqr` is fastest, probably due to employing TSQR, but shifted-CholeskyQR3 (`dsyrk`) is more than 1.7 times faster in every case. These results also indicate that, even if four iterations is required for ill-conditioned problems, iterated Cholesky QR (shown in Algorithm 4.1) would still be faster than other methods.

TABLE 7.2
Computational time on Laurel 2: $\kappa_2(X) = 10^{11}$, $m = 100{,}000$ *and the number of threads is* 36.

	Time (sec.)			
Method	$n = 32$	$n = 64$	$n = 128$	$n = 256$
sCholQR3 (`dgemm` ver.)	2.62×10^{-3}	9.16×10^{-3}	3.32×10^{-2}	1.16×10^{-1}
sCholQR3 (`dsyrk` ver.)	2.39×10^{-3}	7.44×10^{-3}	2.45×10^{-2}	8.20×10^{-2}
`dgeqrf` + `dorgqr`	6.31×10^{-2}	8.94×10^{-2}	1.19×10^{-1}	1.96×10^{-1}
`dgeqr` + `dgemqr`	4.39×10^{-3}	1.30×10^{-2}	4.39×10^{-2}	1.68×10^{-1}
CGS2	9.60×10^{-3}	2.31×10^{-2}	1.40×10^{-1}	1.24
Block CGS2	9.62×10^{-3}	2.49×10^{-2}	7.88×10^{-2}	2.05×10^{-1}

It is of interest to compare shiftedCholeskyQR3 with mixedCholQR [24] from the viewpoint of computational time, but implementing and highly tuning double-double `gemm` or `syrk` routines (whose input matrices are in double precision) is generally difficult and requires significant effort. We thus estimate the computational cost of mixedCholQR routine; we only discuss the case where `gemm` is used, but almost the same discussion is applicable when we use `syrk`. Table 7.3 presents the results of the benchmark for `dgemm`, which computes $X^\top X$, where X is an $m \times n$ matrix. From this table, we can assume that 700 GFLOPS is a rough upper bound of the achieved performance. According to the paper on mixed precision CholeskyQR [24], the number of double precision operations required in `ddgemm` (a double-double precision `gemm` routine) is 8 times (Cray-style) or 12.5 times (IEEE-style) that required in `dgemm`. Therefore, assuming that double precision operations in `ddgemm` are performed at 700 GFLOPS, we can estimate the time (sec.) of `ddgemm` as $8 \cdot 2/7 \cdot mn^2 \cdot 10^{-11}$ (Cray-style) or $12.5 \cdot 2/7 \cdot mn^2 \cdot 10^{-11}$ (IEEE-style).

TABLE 7.3
Achieved performance of `dgemm`*:* $m = 100{,}000$ *and the number of threads is* 36.

n	32	64	128	256	512	1,024	4,096	16,384
GFLOPS	476	559	539	589	600	624	654	628

Based on the above estimation and the breakdown of timing results of shiftedCholeskyQR3, we compare shiftedCholeskyQR3 and mixed precision CholeskyQR in Table 7.4. Here, we ignore the increasing cost for `ddpotrf` because it is small relative to the total time. From the table, we can expect that shiftedCholeskyQR3 is faster than mixed precision CholeskyQR in this computational environment. Considering this estimation and the fact that a well-tuned `ddgemm` is currently rarely available, shiftedCholeskyQR3 seems to be more practical than mixed precision CholeskyQR.

TABLE 7.4
Comparison of shiftedCholeskyQR3 with Mixed Precision CholeskyQR based on the performance estimation for `ddgemm`*:* $m = 100{,}000$ *and* $n = 64$.

	shiftedCholeskyQR3		Mixed Precision CholeskyQR		
	Routine	Time (sec.)	Routine	Cray-style Time (sec.)	IEEE-style Time (sec.)
sCholQR	`dgemm`	1.90×10^{-3}	–	–	–
	`dpotrf`	3.00×10^{-5}	–	–	–
	`dtrsm`	1.11×10^{-3}	–	–	–
CholQR	`dgemm`	1.68×10^{-3}	`ddgemm`	$\geq 9.36 \times 10^{-3}$	$\geq 1.46 \times 10^{-2}$
	`dpotrf`	2.91×10^{-5}	`ddpotrf`	$\geq 2.91 \times 10^{-5}$	$\geq 2.91 \times 10^{-5}$
	`dtrsm`	1.31×10^{-3}	`dtrsm`	1.31×10^{-3}	1.31×10^{-3}
	`dtrmm`	3.10×10^{-5}	–	–	–
CholQR	`dgemm`	1.81×10^{-3}	`dgemm`	1.81×10^{-3}	1.81×10^{-3}
	`dpotrf`	3.79×10^{-5}	`dpotrf`	3.79×10^{-5}	3.79×10^{-5}
	`dtrsm`	1.24×10^{-3}	`dtrsm`	1.24×10^{-3}	1.24×10^{-3}
	`dtrmm`	2.91×10^{-5}	`dtrmm`	2.91×10^{-5}	2.91×10^{-5}
Misc.		1.36×10^{-4}		≥ 0	≥ 0
Total		9.34×10^{-3}		$\geq 1.38 \times 10^{-2}$	$\geq 1.91 \times 10^{-2}$

Finally we briefly mention the performance of shiftedCholeskyQR3 on large-scale distributed parallel systems. Based on our previous performance evaluation of CholeskyQR2 on the K computer (for details, see [8]), we give a rough estimation of the computational time of shiftedCholeskyQR3 in Table 7.5, where we estimate the computational time of shiftedCholeskyQR3 as 1.5 times that of CholeskyQR2, simply based on the number of iterations. As the input matrix becomes taller and skinnier, the cost of computing the shift becomes negligible as we only need to access the diagonal elements of the Gram matrix. As such our estimation is reasonable as there is no additional communication incurred. This table suggests that shiftedCholeskyQR3 is expected to be still significantly faster than Householder QR methods (both TSQR and ScaLAPACK routines) in large-scale parallel computation for matrices $\kappa_2(X) < \mathbf{u}^{-1}$. A thorough investigation of the shiftedCholeskyQR3 performance is left for future work, including comparisons with other methods, optimization in large-scale computation, and scaling studies.

7.2. Oblique inner product. In the case of the inner product defined by a positive definite matrix B, we focus on large and sparse $B \in \mathbb{R}^{m \times m}$, which arises commonly in applications. We compare the performance of shiftedCholeskyQR3 with that of CGS2 and block CGS2.

TABLE 7.5
Estimation of the computational time of shiftedCholeskyQR3 on the K computer: the number of nodes (= MPI processes) is 16,384.

		Time (sec.)			
		Measured			Estimated
m	n	TSQR	`pdgeqrf` + `pdorgqr`	CholQR2	sCholQR3
4,194,304	16	1.64×10^{-3}	1.04×10^{-2}	8.02×10^{-4}	1.20×10^{-3}
	64	7.42×10^{-3}	4.14×10^{-2}	2.52×10^{-3}	3.79×10^{-3}
	256	2.32×10^{-1}	1.84×10^{-1}	3.05×10^{-2}	4.57×10^{-2}
16,777,216	16	1.84×10^{-3}	1.13×10^{-2}	9.06×10^{-4}	1.36×10^{-3}
	64	8.82×10^{-3}	5.65×10^{-2}	3.13×10^{-3}	4.70×10^{-3}
	256	2.40×10^{-1}	3.92×10^{-1}	3.38×10^{-2}	5.07×10^{-2}

Let us describe the implementation details of shiftedCholeskyQR3, CGS2, and block CGS2. In all implementations of shiftedCholeskyQR3, CGS2, and block CGS2, the matrix B is stored using the Compressed Sparse Row (CSR) format, where the symmetric property is not exploited in order to avoid the exclusion control in the thread parallelization of the Sparse Matrix Vector multiplication (SpMV); both the upper and lower triangular parts are explicitly stored. When computing an oblique inner product $y^\top Bx$, where x and y are vectors, we first compute $z := Bx$, then $y^\top z$. In the first step, we employ a standard implementation of SpMV using the CSR format, where we apply a straightforward thread parallelization using OpenMP. In shiftedCholeskyQR3 and block CGS2, a block version of the oblique inner product appears, and we compute them as $Y^\top(BX)$, where X and Y are matrices. For computing BX, we do not repeat SpMV but employ an kernel of Sparse Matrix Multiple Vectors multiplication (SpMMV), which can reuse the data of the sparse matrix; owing to this advantage, SpMMV can generally achieve higher effective performance than SpMV. It is worth noting that the SpMMV kernel can be partially employed in the MGS described in [12], and we used a similar technique in the CGS2 implementation.

The test matrices, i.e., symmetric positive definite matrices, that define oblique inner product were taken from the SuiteSparse Matrix Collection (formerly the University of Florida Sparse Matrix Collection) [4], as listed in Table 7.6. The matrix $X \in \mathbb{R}^{m\times n}$ was generated by $X := U\Sigma V^\top$, just as explained in section 6.1, except that U was computed by applying CGS2 in the oblique inner product defined by B to a random matrix. The matrix X thus generated has the property that $\sqrt{\kappa_2(X^\top BX)} = \kappa_2(\Sigma)$, and we set $\kappa_2(\Sigma) = 10^{11}$ in the experiment. In this situation, CGS and MGS (not iterated twice) cannot compute an accurate Q factor and both Cholesky QR and CholeskyQR2 break down.

TABLE 7.6
Test matrices used for benchmarking `chol_borth`.

Name	Kind	Size (m)	Nonzeros	Nonzeros/row
apache2	structural problem	715,176	2,766,523	3.86
bairport	structural problem	67,537	774,378	11.46
bone010	model reduction	986,703	47,851,783	48.50
G3_circuit	circuit simulation	1,585,478	7,660,826	4.83
Geo_1438	structural problem	1,437,960	60,236,322	41.89
parabolic_fem	CFD	525,825	2,100,225	3.99
serena	structural problem	1,391,349	64,131,971	46.09
shipsec8	structural problem	114,919	3,303,553	28.74
watercube	structural problem	68,598	1,439,940	20.99

As for the shift, we used the following formula that proved to be practical based on preliminary experiments (which is smaller than (5.4)):

$$s = \sqrt{m}\|X\|_F^2 \tilde{\lambda}_{\max}(B) \times 10^{-16}, \tag{7.1}$$

where $\tilde{\lambda}_{\max}(B)$ is an upper bound on the largest eigenvalue of B computed by the Gershgorin circle theorem. Both $\|X\|_F$ and $\tilde{\lambda}_{\max}(B)$ are computed during the first computation of the Gram matrix, that is, while computing $X' := BX$. The costs of computing $\|X\|_F$ and $\tilde{\lambda}_{\max}(B)$ are $O(mn)$ and $O(\mathrm{nnz}(B))$, respectively, where $\mathrm{nnz}(B)$ is the number of nonzeros in B. In all our experiments the shift (7.1) resulted in negligible computational time and orthogonality comparable to the best competing methods.

We present the computational time of shiftedCholeskyQR3, CGS2, and block CGS2 in Table 7.7, where the block size in block CGS2 was empirically tuned. The shiftedCholeskyQR3 algorithm was consistently faster than block CGS2 and offered speedup by a factor of up to 2.6. It was also faster than CGS2, except for the shipsec8 and bone010 matrices with $n = 32$, and the performance difference increased with n. This is reasonable because the performance difference between SpMMV and SpMV usually becomes larger as n grows. This suggests that shiftedCholeskyQR3 is the algorithm of choice when n is not too small.

TABLE 7.7
Computational time of oblique QR factorization on Laurel 2: $\sqrt{\kappa_2(X^\top BX)} = 10^{11}$, *and the number of threads* = 36.

		Time (sec.)			
Matrix	Method	$n = 32$	$n = 64$	$n = 128$	$n = 256$
apache2	sCholQR3	8.14×10^{-2}	1.73×10^{-1}	4.00×10^{-1}	1.16
	CGS2	2.14×10^{-1}	5.87×10^{-1}	2.22	9.23
	Block CGS2	2.16×10^{-1}	4.59×10^{-1}	1.05	2.61
bairport	sCholQR3	1.28×10^{-2}	3.00×10^{-2}	7.14×10^{-2}	1.93×10^{-1}
	CGS2	1.57×10^{-2}	3.51×10^{-2}	1.46×10^{-1}	6.99×10^{-1}
	Block CGS2	2.48×10^{-2}	4.83×10^{-2}	1.12×10^{-1}	2.71×10^{-1}
bone010	sCholQR3	5.16×10^{-1}	9.72×10^{-1}	1.94	5.67
	CGS2	4.84×10^{-1}	1.20	3.90	14.1
	Block CGS2	6.54×10^{-1}	1.44	3.23	7.08
G3_circuit	sCholQR3	2.23×10^{-1}	4.71×10^{-1}	1.17	3.02
	CGS2	4.78×10^{-1}	1.27	5.01	19.8
	Block CGS2	5.89×10^{-1}	1.22	2.83	6.49
Geo_1438	sCholQR3	4.59×10^{-1}	8.99×10^{-1}	1.81	4.84
	CGS2	5.98×10^{-1}	1.51	4.89	19.0
	Block CGS2	7.63×10^{-1}	1.71	3.77	9.46
parabolic_fem	sCholQR3	1.39×10^{-1}	3.04×10^{-1}	6.21×10^{-1}	1.47
	CGS2	1.81×10^{-1}	5.54×10^{-1}	2.08	8.57
	Block CGS2	1.99×10^{-1}	4.26×10^{-1}	9.89×10^{-1}	2.41
serena	sCholQR3	5.34×10^{-1}	1.11	2.28	5.90
	CGS2	5.65×10^{-1}	1.50	4.88	19.3
	Block CGS2	7.61×10^{-1}	1.74	3.81	8.60
shipsec8	sCholQR3	4.34×10^{-2}	9.76×10^{-2}	2.07×10^{-1}	5.61×10^{-1}
	CGS2	3.42×10^{-2}	1.11×10^{-1}	4.84×10^{-1}	2.02
	Block CGS2	4.86×10^{-2}	1.04×10^{-1}	2.48×10^{-1}	5.70×10^{-1}
watercube	sCholQR3	1.22×10^{-2}	2.85×10^{-2}	6.92×10^{-2}	1.72×10^{-1}
	CGS2	1.57×10^{-2}	3.53×10^{-2}	1.48×10^{-1}	7.87×10^{-1}
	Block CGS2	2.45×10^{-2}	4.86×10^{-2}	1.12×10^{-1}	2.71×10^{-1}

Finally, we mention a direction for further performance improvement of shiftedCholeskyQR3. In this paper, we employed a simple implementation of SpMMV, that is, the 2-step computation $X^\top(BX)$, however a fused implementation is possible. In the fused implementation, where $X^\top BX$ is regarded as a unit, several blocking techniques that improve the use of cache memory can be applied. We expect these sophisticated implementation techniques would strengthen the advantage of shiftedCholeskyQR3. For a more detailed discussion and performance analysis of such implementations, we refer the reader to [13] and [14, section 6.4].

Our experiments used in-house kernels for SpMV and SpMMV operations for ease of development and integration. It might be possible to use alternative kernels such as those from Intel MKL. Since SpMV is memory bounded, MKL SpMV (or other SpMV kernel implemented in a standard way) usually provides almost the same performance as our in-house SpMV. On the other hand, any SpMMV kernel with better performance than our in-house ones, probably including MKL SpMMV, will further strengthen the advantage of shiftedCholeskyQR3 over its alternatives.

8. Conclusion and discussion. Our algorithm shiftedCholeskyQR3 combines speed, stability, and versatility (applicable to $B \neq I$). We believe it offers an attractive alternative in high-performance computing (HPC) to the conventional Householder-based QR factorization algorithms for both the $B = I$ and $B \neq I$ cases.

The shiftedCholeskyQR3 algorithm as presented could benefit from further tuning, and this work suggests a few future directions. First, the choice of shift s introduced in this paper is conservative, and severely so when m, n are large. Our experiments suggest that a much smaller shift, such as $s = O(\mathbf{u}\|A\|_2)$, is usually sufficient to avoid breakdown in $\mathrm{chol}(A + sI)$, and as illustrated in Figure 6.3, a smaller shift results in improved conditioning, and hence smaller number of shiftedCholeskyQR iterations. Introduction of a shift strategy that is both stable and efficient is an important remaining task.

Our performance results hold promise for the competitiveness of shiftedCholeskyQR3, and further work will focus on comparing it with other state-of-the-art implementations for QR factorizations such as TSQR in an HPC setting. In the case where $B \neq I$ for a sparse B, the performance benefits over CGS2 are considerable when n is not too small and our algorithm can be the clear choice for applications.

Finally, for rank-deficient matrices, shiftedCholeskyQR3 is inapplicable, because AR is rank-deficient for any R.[2] This issue is not present in Householder-type methods, and a workaround for shiftedCholeskyQR3 is much desired.

REFERENCES

[1] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000.

[2] R. Car and M. Parrinello, *Unified approach for molecular dynamics and density-functional theory*, Phys. Rev. Lett., 55 (1985), pp. 2471–2474, https://doi.org/10.1103/PhysRevLett.55.2471.

[3] I. Daubechies, R. DeVore, M. Fornasier, and C. S. Güntürk, *Iteratively reweighted least squares minimization for sparse recovery*, Comm. Pure Appl. Math., 63 (2010), pp. 1–38.

[4] T. A. Davis and Y. Hu, *The University of Florida Sparse Matrix Collection*, ACM Trans. Math. Softw., 38 (2011), pp. 1:1–1:25, https://doi.org/10.1145/2049662.2049663.

[2] However, roundoff errors often map the zero singular values to $O(\mathbf{u})$, and so a few iterations of shiftedCholeskyQR usually result in $\kappa_2(Q) \leq \mathbf{u}^{-1/2}$.

[5] J. Demmel, *On floating point errors in Cholesky*, Tech. Report 14, LAPACK Working Note, 1989.
[6] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou, *Communication-optimal parallel and sequential QR and LU factorizations*, SIAM J. Sci. Comp, 34 (2012), pp. A206–A239, https://doi.org/10.1137/080731992.
[7] T. Fukaya, R. Kannan, Y. Nakatsukasa, Y. Yamamoto, and Y. Yanagisawa, *Shifted CholeskyQR for computing the QR factorization of ill-conditioned matrices*, Preprint, arXiv:1809.11085, 2018, https://arxiv.org/abs/1809.11085.
[8] T. Fukaya, Y. Nakatsukasa, Y. Yanagisawa, and Y. Yamamoto, *CholeskyQR2: A simple and communication-avoiding algorithm for computing a tall-skinny QR factorization on a large-scale parallel system*, in Proceedings of the 5th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems, IEEE Press, 2014, pp. 31–38.
[9] L. Giraud, J. Langou, M. Rozložník, and J. Eshof, *Rounding error analysis of the classical gram-schmidt orthogonalization process*, Numer. Math., 101 (2005), pp. 87–100.
[10] G. H. Golub, V. Loan, and C. F., *Matrix Computations*, 4th ed., The Johns Hopkins University Press, Baltimore, 2013.
[11] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.
[12] A. Imakura and Y. Yamamoto, *Efficient implementations of the modified Gram-Schmidt orthogonalization with a non-standard inner product*, Japan Journal of Industrial and Applied Mathematics, 2019, https://doi.org/10.1007/s13160-019-00356-4.
[13] R. Kannan, *Efficient sparse matrix multiple-vector multiplication using a bitmapped format*, in 20th IEEE International Conference on High Performance Computing (HiPC'13), December 2013, pp. 286–294, https://doi.org/10.1109/HiPC.2013.6799135.
[14] R. Kannan, *Numerical Linear Algebra problems in Structural Analysis*, PhD thesis, School of Mathematics, The University of Manchester, 2014.
[15] B. R. Lowery and J. Langou, *Stability analysis of QR factorization in an oblique inner product*, Preprint, arXiv:1401.5171, 2014, https://arxiv.org/abs/1401.5171.
[16] M. Mohiyuddin, M. Hoemmen, J. Demmel, and K. Yelick, *Minimizing communication in sparse matrix solvers*, in Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC '09, New York, NY, USA, 2009, ACM, pp. 36:1–36:12, https://doi.org/10.1145/1654059.1654096.
[17] *Multiprecision Computing Toolbox*. Advanpix, Tokyo, http://www.advanpix.com.
[18] M. Rozložník, M. Tůma, A. Smoktunowicz, and J. Kopal, *Numerical stability of orthogonalization methods with a non-standard inner product*, BIT, (2012), pp. 1–24.
[19] S. M. Rump and T. Ogita, *Super-fast validated solution of linear systems*, J. Comput. Appl. Math., 199 (2007), pp. 199–206.
[20] A. Stathopoulos and K. Wu, *A block orthogonalization procedure with constant synchronization requirements*, SIAM J. Sci. Comp, 23 (2002), pp. 2165–2182.
[21] S. Toledo and E. Rabani, *Very large electronic structure calculations using an out-of-core filter diagonalization method*, J. Comput. Phys., 180 (2002), pp. 256–269.
[22] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.
[23] Y. Yamamoto, Y. Nakatsukasa, Y. Yanagisawa, and T. Fukaya, *Roundoff error analysis of the CholeskyQR2 algorihm*, Electron. Trans. Numer. Anal, 44 (2015), pp. 306–326.
[24] I. Yamazaki, S. Tomov, and J. Dongarra, *Mixed-precision Cholesky QR factorization and its case studies on Multicore CPU with Multiple GPUs*, SIAM J. Sci. Comput., 37 (2015), pp. C307–C330.
[25] Y. Yanagisawa, T. Ogita, and S. Oishi, *A modified algorithm for accurate inverse Cholesky factorization*, Nonlinear Theory and Its Applications, IEICE, 5 (2014), pp. 35–46.