Analysis of Randomized Householder-Cholesky QR Factorization with Multisketching

Andrew J. Higgins, Daniel B. Szyld,
Erik G. Boman, and Ichitaro Yamazaki

Report 23-09-11
September 2023. Revised August 2024.

Department of Mathematics
Temple University
Philadelphia, PA 19122

This report is available in the World Wide Web at
`http://www.math.temple.edu/~szyld`
It is also available at the Arxiv at `https://arxiv.org/abs/2309.05868`

Analysis of Randomized Householder-Cholesky QR Factorization with Multisketching

Andrew J. Higgins* Daniel B. Szyld† Erik G. Boman*
Ichitaro Yamazaki*

August 28, 2024

Abstract

CholeskyQR2 and shifted CholeskyQR3 are two state-of-the-art algorithms for computing tall-and-skinny QR factorizations since they attain high performance on current computer architectures. However, to guarantee stability, for some applications, CholeskyQR2 faces a prohibitive restriction on the condition number of the underlying matrix to factorize. Shifted CholeskyQR3 is stable but has 50% more computational and communication costs than CholeskyQR2. In this paper, a randomized QR algorithm called Randomized Householder-Cholesky (`rand_cholQR`) is proposed and analyzed. Using one or two random sketch matrices, it is proved that with high probability, its orthogonality error is bounded by a constant of the order of unit roundoff for any numerically full-rank matrix, and hence it is as stable as shifted CholeskyQR3. An evaluation of the performance of `rand_cholQR` on an NVIDIA A100 GPU demonstrates that for tall-and-skinny matrices, `rand_cholQR` with multiple sketch matrices is nearly as fast as, or in some cases faster than, CholeskyQR2. Hence, compared to CholeskyQR2, `rand_cholQR` is more stable with almost no extra computational or memory cost, and therefore a superior algorithm both in theory and practice.

Keywords: Randomized Linear Algebra, QR Factorization, Communication-Avoiding Algorithms, Error Analysis, Numerical Stability, GPUs

MSC Classification: 65F05, 65F20, 65F25, 65G50, 15B52

1 Introduction

Computing the QR factorization of tall-and-skinny matrices is a critical component of many scientific and engineering applications, including the solution

*Center for Computing Research, Sandia National Laboratories, Albuquerque, New Mexico, USA (`ajhiggi@sandia.gov`, `egboman@sandia.gov`, `iyamaza@sandia.gov`).
†Temple University, Philadelphia, Pennsylvania, USA (`szyld@temple.edu`).

of least squares problems, block orthogonalization kernels for solving linear systems and eigenvalue problems within block or s-step Krylov methods, dimensionality reduction methods for data analysis like Principal Component Analysis, and many others. The current state-of-the-art QR algorithms for tall-and-skinny matrices are the CholeskyQR2 and shifted CholeskyQR3 algorithms [12, 13], thanks to their communication-avoiding properties along with their exploitation of vendor provided highly-optimized dense linear algebra subroutines [2, 22, 23]. However, CholeskyQR2 may fail to accurately factorize a matrix V when its condition number $\kappa(V) \gtrapprox \mathbf{u}^{-1/2}$, where $\mathbf{u}$ is unit roundoff [36]. Shifted CholeskyQR3 is numerically stable as long as $\kappa(V) \lessapprox \mathbf{u}^{-1}$, but it requires over 50% more computational and communication cost than CholeskyQR2 [12]. Although more stable communication-avoiding algorithms exist, such as TSQR [8], they rely on Householder QR factorizations of potentially large matrices, and are often significantly slower than CholeskyQR2 in practice [13].

In this paper, we first present and analyze a randomized algorithm called `randQR` for orthogonalizing the columns of a tall-and-skinny matrix with respect to a specific inner product. In order to reduce the cost of the computations, we propose to use "multisketching," i.e., the use of two consecutive sketch matrices, within `randQR`. Using `randQR` with multisketching as a preconditioner for less stable QR factorizations can be an efficient strategy for computing the true QR factorization of a matrix, which leads to the primary focus of this paper, which is an algorithm called `rand_cholQR` for computing the true QR factorization of a tall-and-skinny matrix V. Our approach is general in the sense that our analysis applies to any two ϵ-subspace embedding sketching matrices (see Section 3 for definitions), but is specifically motivated by the use of a large sparse sketch followed by a smaller dense sketch, such as a Gaussian or Radamacher sketch [1]. Our analysis applies in particular to Count-Gauss (one application of CountSketch followed by a Gaussian sketch), as described in [18, 29, 30].

We prove that with high probability, the orthogonality error of `rand_cholQR` is on the order of unit roundoff for any numerically full-rank matrix V (i.e., $\kappa(V) \lessapprox \mathbf{u}^{-1}$) and hence it is as stable as shifted CholeskyQR3 and it is significantly more numerically stable than CholeskyQR2. Our numerical experiments ilustrate the theoretical results. In addition, the `rand_cholQR` algorithm may be implemented using the same basic linear algebra kernels as CholeskyQR2. Therefore, it is simple to implement and has the same communication-avoiding properties. We perform a computational study on a state-of-the-art GPU to demonstrate that `rand_cholQR` can perform up to 4% faster than CholeskyQR2 and 56.6% faster than shifted CholeskyQR3, while significantly improving the robustness of CholeskyQR2.

In summary, our primary contribution consists of a new error analysis of a multisketched randomized QR algorithm, proving it can be safely used for matrices of larger condition number than CholeskyQR2 can handle. This analysis also applies to the case of one sketch, improving upon the existing results. Our implementation confirms and illustrates the theory developed in this paper. Our secondary contribution is a computational study that tangibly demonstrates

that the multisketched algorithm has superior performance over the existing single sketch algorithm and similar performance to the high-performance but less stable CholeskyQR2 algorithm.

In Section 2, we begin by discussing prior work on similar topics. Then, in Section 3, we present some preliminary definitions and known results from randomized linear algebra relevant to this work. We follow with Section 4, where we present multisketching on a conceptual level, and how to incorporate it into a randomized QR factorization (`rand_cholQR`). We also discuss performance considerations for `rand_cholQR` compared to other high performance tall-skinny QR algorithms, leading to the motivation as to why multisketching is recommended. In Section 5, we present rigorous error bounds and their proofs for the proposed multisketched `rand_cholQR`. These bounds can also be applied to the case of a single sketch matrix, and we compare the new results to those available in the literature. Numerical experiments are presented in Section 6, followed by our conclusions. For completeness, all detailed run times of our experiments are reported in an Appendix.

2 Related Work

In the case of a single sketch matrix, the concept of sketching a tall-and-skinny matrix, computing its QR factorization, and then preconditioning the matrix with the resulting triangular factor like `randQR` (Algorithm 1) is not new. The earliest appearance of such an algorithm was by Rokhlin and Tygert in 2008 [27] for solving overdetermined least squares problems, where they proposed a version of `randQR` with a column-pivoted QR factorization and a single subsampled randomized Hadamard transform sketch. Fan et al. [10] proposed a similar algorithm called rQR-CholQR, but they only used a very simple sketch based on sampling rows of A. They did not consider subspace embeddings or multisketching.

Prior to this paper, Balabanov and Grigori proposed the "RCholeskyQR" method in an unpublished manuscript [5], which is identical to what we refer to as `randQR`, in the case of a single (ε, d, m) oblivious ℓ_2-subspace embedding. While this paper was being written, Balabanov gave stability results similar to Corollary 5.2 in an additional unpublished manuscript [3]. However, our results differ from Balabanov's, as ours impose no assumptions on the level of accuracy performed by subroutines within the algorithm, and we meticulously derive all bounds from existing roundoff error analysis of each subroutine. Additionally, Balabanov's work imposes a far stricter limit on the subspace embedding parameter $\epsilon \leq \frac{1}{2}$, while ours provides analysis up to $\epsilon < \frac{616}{634}$ for a (ε, d, m) oblivious ℓ_2-subspace embedding, which is nearly the theoretical upper limit of $\epsilon < 1$ imposed by the theory in Section 3. This is significant, because stability guarantees for larger values of ϵ ensure high accuracy with smaller sketch matrices, resulting in a more computationally efficient algorithm, as demonstrated in Section 6.4.

Our results extend beyond a single (ε, d, m) oblivious ℓ_2-subspace embed-

ding, and cover the more generalized case of two subspace embeddings (i.e., multisketch). Also, our work includes explicit analysis of the S_2S_1-orthogonality error of `randQR`, which is a specific notion of orthogonality with respect to a sketched inner product, and the loss of orthogonality error in the standard Euclidean inner product of `rand_cholQR`.

Our work is novel in several ways. To our knowledge, this work is the first to propose and analyze a randomized QR algorithm with multiple sketches. The stability results in this paper improve upon and expand the existing stability analysis of `randQR` and `rand_cholQR`, and considers the multisketch case for the first time. Additionally, our experimental results are the first to demonstrate the performance of `rand_cholQR` in a parallel heterogeneous computing environment under any sketching framework, particularly in the multisketch case which allows the algorithm to sometimes run faster than the widely used high-performance `cholQR2` algorithm. This tangibly demonstrates the potential of the multisketch `rand_cholQR` in exascale applications.

3 Preliminaries on Random Sketching

Suppose one would like to compress $V \in \mathbb{R}^{n\times m}$ into a matrix with fewer rows with nearly the same norm. We denote the *sketch matrix* by $S \in \mathbb{R}^{p\times n}$ for $p \ll n$. The sketch matrix is typically chosen to be a *ϵ-subspace embedding* (defined below), or a linear map to a lower dimensional space that preserves ℓ_2-inner products and norms of all vectors within the subspace up to a factor of $\sqrt{1 \pm \varepsilon}$ for $\varepsilon \in [0, 1)$ [4, 20, 28].

Definition 3.1 (ε-subspace embedding)**.** *Given $\varepsilon \in [0, 1)$, the sketch matrix $S \in \mathbb{R}^{p\times n}$ is an* ε-subspace embedding *for the subspace $\mathcal{V} \subset \mathbb{R}^n$ if $\forall x, y \in \mathcal{V}$,*

$$|\langle x, y\rangle - \langle Sx, Sy\rangle| \leq \varepsilon \|x\|_2 \|y\|_2,$$

where $\langle \cdot, \cdot \rangle$ is the Euclidean inner product.

Proposition 3.1. [4] *If the sketch matrix $S \in \mathbb{R}^{p\times n}$ is an* ε-subspace embedding *for the subspace $\mathcal{V} \subset \mathbb{R}^n$, then $\forall x \in \mathcal{V}$, then,*

$$\sqrt{1-\varepsilon}\ \|x\|_2 \leq \|Sx\|_2 \leq \sqrt{1+\varepsilon}\ \|x\|_2. \tag{1}$$

Corollary 3.1. *If the sketch matrix $S \in \mathbb{R}^{p\times n}$ is an* ε-subspace embedding *for the subspace $\mathcal{V} \subset \mathbb{R}^n$, and V is a matrix whose columns form a basis of $\mathcal{V}$, then*

$$\sqrt{1-\varepsilon}\|V\|_2 \leq \|SV\|_2 \leq \sqrt{1+\varepsilon}\|V\|_2, \tag{2}$$

$$\sqrt{1-\varepsilon}\|V\|_F \leq \|SV\|_F \leq \sqrt{1+\varepsilon}\|V\|_F, \tag{3}$$

$$\sqrt{1-\varepsilon}\ \sigma_{min}(V) \leq \sigma_{min}(SV). \tag{4}$$

Proposition 3.1 follows simply by substituting $y = x$ in Definition 3.1, and Corollary 3.1 is a simple consequence of Proposition 3.1 using the definition

of the ℓ_2 matrix norm, the Frobenius norm, and the minimum singular value. Furthermore, Proposition 3.1 can be used to relate the singular values of SV to those of V in a different way to bound the condition number of V by that of SV.

Proposition 3.2. [4] *If the sketch matrix $S \in \mathbb{R}^{p\times n}$ is an ε-*subspace embedding *for the subspace $\mathcal{V} \subset \mathbb{R}^n$, and V is a matrix whose columns form a basis of $\mathcal{V}$, then*

$$(1+\varepsilon)^{-1/2}\ \sigma_{min}(SV) \leq \sigma_{min}(V) \leq \sigma_{max}(V) \leq (1-\varepsilon)^{-1/2}\ \sigma_{max}(SV). \quad (5)$$

Thus,

$$\kappa(V) \leq \sqrt{\frac{1-\varepsilon}{1+\varepsilon}}\ \kappa(SV). \quad (6)$$

Proposition 3.2 implies that if SV is well conditioned, then so is V.

While ε-subspace embeddings require knowledge of the subspace $\mathcal{V} \subset \mathbb{R}^n$ a priori, (ε, d, m) *oblivious ℓ_2-subspace embeddings* do not.

Definition 3.2 ((ε, d, m) oblivious ℓ_2-subspace embedding)**.** [4] *$S \in \mathbb{R}^{p\times n}$ is an* (ε, d, m) oblivious ℓ_2-subspace embedding *if it is an ε-subspace embedding for any fixed m-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$ with probability at least $1-d$.*

An example of a (ε, d, m) oblivious ℓ_2-subspace embedding is $S = \frac{1}{\sqrt{p}}G$ for a fully dense Gaussian matrix $G \in \mathbb{R}^{p\times n}$ and

$$p = \Omega(\varepsilon^{-2}\log m \log(1/d));$$

see, e.g., [28]. Sparse (ε, d, m) oblivious ℓ_2-subspace embeddings exist, including CountSketch, which consists of a single ± 1 per column, where the row storing the entry and its sign are chosen uniformly at random [6, 35]. In order to be a (ε, d, m) oblivious ℓ_2-subspace embedding, the number of columns of the CountSketch matrix must satisfy

$$p \geq \frac{m^2+m}{\varepsilon^2 d}\ ; \quad (7)$$

see [19]. Other popular (ε, d, m) oblivious ℓ_2-subspace embeddings include subsampled randomized Hadamard and Fourier transforms, and "sparse dimension reduction maps" [4, 20], though obtaining high performance with these is difficult, and the complexity of applying them is higher than CountSketch. We do not consider such embeddings in this paper.

4 Multisketching

Next, we consider the case of applying two sketch matrices one after the other, which is what we refer to as "multisketching" in this paper, generalizing the approach of [18, 30], where one application of a large sparse CountSketch is

followed by a smaller dense Gaussian sketch. In these references though, there is no analysis of stability, as we do here. The main motivation for this approach is to be able to apply the dense Gaussian sketch to a smaller matrix, obtained after the application of a sparse sketch, thus significantly reducing the size of the matrix with little computational cost; see more details on this motivation in Section 4.2.

We first present the algorithm `randQR` using this multisketching approach, and then prove bounds similar to those in Proposition 3.2 for the case of two sketches. We emphasize that Algorithm 1 will be used as a pre-processing procedure for the final algorithm of interest, Algorithm 3, which will form the true QR factorization of a tall-and-skinny matrix.

Let $V \in \mathbb{R}^{n\times m}$, and suppose $S_1 \in \mathbb{R}^{p_1\times n}$ and $S_2 \in \mathbb{R}^{p_2\times p_1}$ are (ε_1, d_1, m) and (ε_2, d_2, m) oblivious ℓ_2-subspace embeddings, respectively. Let $d = d_1 + d_2 - d_1 d_2$, so that $1 - d = (1 - d_1)(1 - d_2)$. We define the Randomized Householder QR algorithm (`randQR`) in Algorithm 1, where we use MATLAB function call notation.

Algorithm 1 Randomized Householder QR: $[Q, R] = \texttt{randQR}(V, S_1, S_2)$

Input: Matrix $V \in \mathbb{R}^{n\times m}$, sketch matrices $S_1 \in \mathbb{R}^{p_1\times n}$, $S_2 \in \mathbb{R}^{p_2\times p_1}$
Output: S_2S_1-Orthogonal factor $Q \in \mathbb{R}^{n\times m}$, Triangular factor $R \in \mathbb{R}^{m\times m}$ such that $QR = V$.

1: Apply sketches $W = S_2S_1V$
2: Perform Householder QR: $[Q_{tmp}, R] = \texttt{hhqr}(W)$
3: Recover S_2S_1-orthogonal matrix: $Q = VR^{-1}$

Remark 4.1. In exact arithmetic, provided that $V \in \mathbb{R}^{n\times m}$ is full rank, then `randQR` produces a matrix Q that is S_2S_1-orthogonal[1] with probability at least $1 - d$; i.e., it satisfies $(S_2S_1Q)^T(S_2S_1Q) = I$, because

$$S_2S_1Q = S_2S_1VR^{-1} = WR^{-1} = Q_{tmp},$$

where Q_{tmp} is the orthogonal factor produced by the Householder QR factorization of $W = S_2S_1V$. Observe that Q being S_2S_1-orthogonal is equivalent to being an orthonormal matrix with respect to the inner product[2] $\langle S_2S_1\cdot, S_2S_1\cdot\rangle$. Unlike traditional Householder QR, even in exact arithmetic V must have full rank, since step 3 of Algorithm 1 requires $\text{rank}(V) = \text{rank}(R) = m$. In finite precision, intuition suggests that an inevitable requirement of `randQR` is that V must be numerically full rank (i.e., $\kappa(V) \lessapprox \mathbf{u}^{-1}$).

Next, we introduce some convenient norm, singular value, and condition number inequalities when one uses the multisketching approach with two oblivious ℓ_2-subspace embeddings.

[1] In exact arithmetic, Q will only fail to be S_2S_1-orthogonal if $V \in \text{null}(S_2S_1)$, which by Proposition 3.2, occurs with probability at most d.

[2] Although $\langle S_2S_1\cdot, S_2S_1\cdot\rangle$ is not an inner product over the traditional vector space $\mathbb{R}^{n\times m}$, it is an inner product over the complement of $\text{null}(S_2S_1)$.

Proposition 4.1. *Let $S_1 \in \mathbb{R}^{p_1 \times n}$ be a (ε_1, d_1, m) oblivious ℓ_2-subspace embedding in $\mathbb{R}^n$, $S_2 \in \mathbb{R}^{p_2 \times p_1}$ be a (ε_2, d_2, m) oblivious ℓ_2-subspace embedding in $\mathbb{R}^{p_1}$, generated independently. Let $\varepsilon_L = \varepsilon_1 + \varepsilon_2 - \varepsilon_1\varepsilon_2$, $\varepsilon_H = \varepsilon_1 + \varepsilon_2 + \varepsilon_1\varepsilon_2$, and $d = d_1 + d_2 - d_1 d_2$. Then for any m-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$ and $\forall x \in \mathcal{V}$,*

$$\sqrt{1-\varepsilon_L}\ \|x\|_2 \leq \|S_2 S_1 x\|_2 \leq \sqrt{1+\varepsilon_H}\ \|x\|_2, \tag{8}$$

with probability at least $1 - d$.

Proof. Let $x \in \mathcal{V}$. Then, $S_1\mathcal{V} \subset \mathbb{R}^{p_1}$. By assumption, S_2 is a (ε_2, d_2, m) oblivious ℓ_2-subspace embedding, and thus it is an ε_2-subspace embedding of $S_1\mathcal{V}$ with probability at least $1 - d_2$. Observe that $S_1 x \in S_1\mathcal{V}$. Therefore, by (1),

$$\sqrt{1-\varepsilon_2}\|S_1 x\|_2 \leq \|S_2 S_1 x\|_2 \leq \sqrt{1+\varepsilon_2}\|S_1 x\|_2,$$

with probability at least $1 - d_2$, because this is the probability at which (1) holds.

Again, by assumption, S_1 is a (ε_1, d_1, m) oblivious ℓ_2-subspace embedding, and thus it is an ε_1-subspace embedding of $\mathcal{V} \subset \mathbb{R}^n$ with probability at least $1 - d_1$. Now, using (1) again for S_1 and ϵ_1, we have

$$\sqrt{1-\varepsilon_1}\|x\|_2 \leq \|S_1 x\|_2 \leq \sqrt{1+\varepsilon_1}\|x\|_2,$$

with probability at least $1 - d_1$.

Combining these results, we find that

$$\begin{aligned}
\sqrt{1-(\varepsilon_1+\varepsilon_2-\varepsilon_1\varepsilon_2)}\|x\|_2 &= \sqrt{(1-\varepsilon_2)(1-\varepsilon_1)}\|x\|_2 \leq \sqrt{1-\varepsilon_2}\|S_1 x\|_2 \\
&\leq \|S_2 S_1 x\|_2 \leq \sqrt{1+\varepsilon_2}\|S_1 x\|_2 \\
&\leq \sqrt{(1+\varepsilon_2)(1+\varepsilon_1)}\|x\|_2 \\
&= \sqrt{1+(\varepsilon_1+\varepsilon_2+\varepsilon_1\varepsilon_2)}\|x\|_2
\end{aligned}$$

with probability at least $(1 - d_1)(1 - d_2) = 1 - (d_1 + d_2 - d_1 d_2)$.

Proving d and consequently $1 - d$ are between $[0, 1]$ is equivalent to showing $p(d_1, d_2) = d_1 + d_2 - d_1 d_2 \in [0, 1]$ for any $(d_1, d_2) \in [0, 1]^2$. This is straightforward, as on the boundaries, $p(0, d_2) = d_2 \in [0, 1]$, $p(d_1, 0) = d_1 \in [0, 1]$, $p(1, d_2) = p(d_1, 1) = 1 \in [0, 1]$, and $\nabla p \geq 0$ on $[0, 1]^2$, and therefore $p(d_1, d_2)$ cannot go below 0 or above 1. □

If S_1, S_2 are $\varepsilon_1, \varepsilon_2$ embeddings respectively, then by Corollary 3.1 along with Propositions 3.2 and 4.1,

$$\sqrt{1-\varepsilon_L}\|V\|_2 \leq \|S_2 S_1 V\|_2 \leq \sqrt{1+\varepsilon_H}\|V\|_2, \tag{9}$$

$$\sqrt{1-\varepsilon_L}\|V\|_F \leq \|S_2 S_1 V\|_F \leq \sqrt{1+\varepsilon_H}\|V\|_F, \tag{10}$$

$$\sqrt{1-\varepsilon_L}\ \sigma_{min}(V) \leq \sigma_{min}(S_2 S_1 V), \tag{11}$$

$$\begin{aligned}
(1+\varepsilon_H)^{-1/2}\ \sigma_{min}(S_2 S_1 V) \leq \sigma_{min}(V) \leq \sigma_{max}(V) \\
\leq (1-\varepsilon_L)^{-1/2}\ \sigma_{max}(S_2 S_1 V),
\end{aligned} \tag{12}$$

and so,

$$\kappa(V) \leq \sqrt{\frac{1-\varepsilon_L}{1+\varepsilon_H}}\, \kappa(S_2 S_1 V). \tag{13}$$

Remark 4.2. By Remark 4.1, in exact arithmetic, the Q factor computed by `randQR` is S_2S_1-orthogonal, and therefore, by (13),

$$\kappa(Q) \leq \sqrt{\frac{1-\varepsilon_L}{1+\varepsilon_H}}\, \kappa(S_2 S_1 Q) = \sqrt{\frac{1-\varepsilon_L}{1+\varepsilon_H}} = O(1)\ . \tag{14}$$

Thus, `randQR` serves well for applications where a well-conditioned set of vectors is sufficient, or as a pre-processing algorithm for less stable orthogonalization schemes.

4.1 Algorithms and Performance Considerations

We introduce the main algorithm of interest for this paper, `rand_cholQR` (Algorithm 3) and compare its performance to `cholQR2` [13] (Algorithm 4 below). In this paper, `randQR` is strictly used to precondition the tall-and-skinny matrix V as a pre-processing step for `rand_cholQR`, which is a true orthogonalization scheme.

As a proof of concept for why `rand_cholQR` (Algorithm 3) should be expected to form a reasonable QR factorization, in step 1, the algorithm computes a S_2S_1-orthogonal factor Q_0 from `randQR` (Algorithm 1). By Remark 4.2, in exact arithmetic, $\kappa(Q_0) = O(1)$. In step 2 of `rand_cholQR`, Q is computed by re-orthogonalizing Q_0 using Cholesky QR (`cholQR`, Algorithm 2). Since $\kappa(Q_0) = O(1)$, one can expect the resulting Q satisfies $\|Q^TQ - I\|_2 = O(\mathbf{u})$ using the roundoff error analysis of `cholQR` [36].

Algorithm 2 Cholesky QR: $[Q, R] = \texttt{cholQR}(V)$

Input: Matrix $V \in \mathbb{R}^{n\times m}$
Output: Orthogonal factor $Q \in \mathbb{R}^{n\times m}$, Triangular factor $R \in \mathbb{R}^{m\times m}$ such that $QR = V$.

1: Compute Gram matrix $G = V^TV$
2: Perform Cholesky on G: $R = \texttt{chol}(G)$, where $G = R^TR$
3: Recover orthogonal matrix: $Q = VR^{-1}$

Algorithm 3 Rand. Householder-Cholesky: $[Q, R] = \texttt{rand\_cholQR}(V, S_1, S_2)$

Input: Matrix $V \in \mathbb{R}^{n\times m}$, sketch matrices $S_1 \in \mathbb{R}^{p_1\times n}$, $S_2 \in \mathbb{R}^{p_2\times p_1}$
Output: Orthogonal factor $Q \in \mathbb{R}^{n\times m}$, Triangular factor $R \in \mathbb{R}^{m\times m}$ such that $QR = V$.

1: Recover S_2S_1-orthogonal matrix Q_0: $[Q_0, R_0] = \texttt{randQR}(V, S_1, S_2)$
2: Perform Cholesky QR on Q_0: $[Q, R_1] = \texttt{cholQR}(Q_0)$
3: Return R: $R = R_1R_0$

Next, we examine the expected performance of `randQR` and `rand_cholQR` by first discussing their communication costs compared to `cholQR` and `cholQR2` (Algorithm 4) respectively, and then analyze their arithmetic costs. The most computationally intensive parts of `randQR` (steps 1 and 3) are nearly identical to those of `cholQR`, in the sense that both perform a product of tall and skinny matrices, followed by a triangular solve of a tall and skinny matrix. Similar to the way `cholQR` requires only one processor synchronization total to compute the Gram matrix in step 1 of Algorithm 2, `randQR` only requires one synchronization total to compute W in step 1 of Algorithm 1 provided $m \leq p_2 \leq p_1 \ll n$ and therefore the algorithms incur the same number of processor synchronizations[3]. Moreover, `rand_cholQR` and `cholQR2` simply build on these algorithms, adding passes of `cholQR` to matrices of the same size for both algorithms. Thus, like `cholQR2`, `rand_cholQR` only requires two synchronizations total.

Algorithm 4 CholeskyQR2: $[Q, R] = \texttt{cholQR2}(V)$

Input: Full rank matrix $V \in \mathbb{R}^{n\times m}$
Output: Orthogonal factor $Q \in \mathbb{R}^{n\times m}$, Triangular factor $R \in \mathbb{R}^{m\times m}$
1: Perform Cholesky QR on W: $[Q_0, R_0] = \texttt{cholQR}(V)$
2: Perform Cholesky QR on Q_0: $[Q, R_1] = \texttt{cholQR}(Q_0)$
3: Return R: $R = R_1 R_0$

Next, we consider the computational costs of the algorithms. The computational cost of step 2 of `randQR` (Algorithm 1) is negligible compared to steps 1 and 3, since $W \in \mathbb{R}^{p_2\times m}$ with $p_2 \ll n$. The arithmetic cost of step 1 is dependent on the type of sketch matrices used. Suppose one replaces $S_2 S_1$ with a single dense Gaussian sketch matrix $S \in \mathbb{R}^{p\times n}$, which is conceptually simple, very efficient in parallel, but computationally expensive since it is very dense. Then the arithmetic cost of `randQR` and `rand_cholQR` (in FLOPs) are:

$$\texttt{randQR FLOPs: } \underbrace{pm(2n-1)}_{\text{Sketching}} + \underbrace{2pm^2 - \frac{2}{3}m^3}_{\text{Householder QR}} + \underbrace{nm^2}_{\text{Tri. solve}} \approx 2nmp + nm^2.$$

$$\texttt{rand\_cholQR FLOPs: } \underbrace{2nmp + nm^2}_{\texttt{randQR}} + \underbrace{2nm^2}_{\texttt{cholQR}} + \underbrace{m^2(2m-1)}_{\text{Matrix mult.}} \approx 2nmp + 3nm^2.$$

Provided that $p = O(m)$, e.g., $p \approx 2m$, then `rand_cholQR` FLOPs $\approx 7nm^2$.

In contrast, CholeskyQR2 (`cholQR2`, Algorithm 4) incurs a cost of

$$\texttt{cholQR2 FLOPs: } \underbrace{2nm^2}_{\texttt{cholQR}} + \underbrace{2nm^2}_{\texttt{cholQR}} + \underbrace{m^2(2m-1)}_{\text{Matrix mult.}} \approx 4nm^2.$$

[3]Specifically, suppose one has p parallel processes and $m \leq p_2 \leq p_1 \ll n$ so that $S_2 \in \mathbb{R}^{p_2\times p_1}$ can be stored locally on each process. One can distribute block row partitions of $V = [V_1^T, \ldots, V_p^T]^T$ and block column partitions of the larger sketch $S_1 = [(S_1)_1, \ldots, (S_1)_p]$ to each of the processes, along with the entire small sketch S_2 to each process. Then on process k, one computes $W_k = S_2(S_1)_k V_k$, and then one synchronizes the processes to compute $W = S_2 S_1 V = \sum_{k=1}^{p} W_k$ in a single reduction.

Thus, `randQR` (using a dense Gaussian sketch) and `cholQR` have about the same asymptotic arithmetic costs. Because the two algorithms have the same communication costs and `rand_cholQR` has a slightly higher arithmetic cost, in a large scale parallel setting, one can expect `rand_cholQR` to run slightly slower but on the same order of runtime as `cholQR2` and scale in the same way. However, as we show in Section 5.3, `rand_cholQR` is significantly more stable with high probability.

4.2 Motivation for Multisketching

To motivate the use of multisketching, we first discuss a few straightforward options using a single sketch matrix. Using a single Gaussian sketch requires a dense matrix-matrix multiply with a sketch matrix S of dimension $p \times n$. In addition to performing $O(nmp)$ FLOPs to apply this sketch, we need to store and load this completely dense $p \times n$ sketch matrix. As shown in Section 4.1, the time to sketch the matrix with the dense Gaussian can dominate the total factorization time for `randQR` and consequently `rand_cholQR`.

One can reduce the sketching cost using a sparse sketch such as a CountSketch matrix [6]. Since the CountSketch matrix has only one non-zero per column, the cost of applying the CountSketch matrix to $V \in \mathbb{R}^{n \times m}$ is only $O(nm)$, and it requires the storage of only $O(n)$ numerical values. Additionally, CountSketch can be implemented using the sparse-matrix multiple-vector multiply (SpMM), whose optimized implementation is often available on specific architectures. A clever implementation can exploit the fact that applying the CountSketch matrix is equivalent to adding/subtracting subsets of rows of V, and can therefore be parallelized well using batched BLAS-1 kernels or a highly-optimized sparse linear algebra library. Hence, high performance implementations of CountSketch can be achieved using only readily available linear algebra libraries. However, a CountSketch matrix requires a sketch size of $p = O(m^2)$ to maintain the ε-embedding properties, so `randQR`/`rand_cholQR` requires one to factorize $W \in \mathbb{R}^{O(m^2) \times m}$ with Householder QR, which incurs $O(m^4)$ FLOPs. In contrast, a sketch size of $p = O(m)$ ensures the Gaussian sketch is an ε-subspace embedding, meaning the cost of the Householder QR factorization is only $O(m^3)$ FLOPs. Householder QR imposes high communication costs and does not parallelize well [8]. As a result, on current computers, it obtains much lower performance than the BLAS-3 operations like the dense matrix product (`gemm`), and these $O(m^4)$ FLOPs for Householder QR become a performance bottleneck for sufficiently large m.

Ideally, we want an embedding that offers low computational and storage costs like CountSketch, while returning a sketched matrix $W \in \mathbb{R}^{p \times m}$ with $p = O(m)$ like the Gaussian sketch does, to avoid a performance bottleneck from Householder QR. This is possible by using the multisketching framework with first a sparse CountSketch and then a Gaussian sketch. To see this, suppose $S_1 \in \mathbb{R}^{p_1 \times n}$ is a CountSketch matrix with $p_1 = \frac{m^2+m}{\varepsilon_1^2 d_1}$, cf. (7), and suppose $S_2 \in \mathbb{R}^{p_2 \times p_1}$ is a Gaussian sketch where $p_2 = 2m$.

We split the computation of $W = S_2 S_1 V$ into two steps: first computing $W_1 = S_1 V$, then $W = S_2 W_1$. Storing S_1 only requires $O(n)$ bytes of memory, and the sparse matrix product $W_1 = S_1 V$ costs $O(nm)$ FLOPs. The cost to compute $W = S_2 W_1$ costs $O(m^4)$ FLOPs, but since the dense matrix product (`gemm`) obtains much higher performance than the Householder QR, this cost became negligible in our performance studies with a GPU. The storage of S_2 only requires $O(m^3)$ bytes of memory, and the Householder QR factorization of the $O(m) \times m$ matrix W incurs negligble computational cost as well.

Moreover, the $O(nm + m^4)$ total FLOPs incurred by `randQR` using the multisketch framework can actually be lower than the $O(nm^2)$ FLOPs required to perform `cholQR`, making `rand_cholQR` sometimes cheaper than `cholQR2` under the multisketch framework while incurring the same number of communications (as discussed in Section 4.1). Thus, the multisketch framework provides an avenue for an extremely efficient, stable QR factorization that can potentially outperform `cholQR2` in terms of both stability and practical speed on modern parallel machines.

5 Error Analysis of `randQR` and `rand_cholQR`

In this section we present the main results of this work on theoretical properties (with high probability) of $\hat{Q}$ and $\hat{R}$ computed by `randQR` and `rand_cholQR`. The structure of the section is as follows. First, we highlight the sources of floating point error of the `randQR` algorithm in Section 5.1. Then, in Section 5.2, we introduce our assumptions for the proofs and some preparatory results for the error analysis.

We identify which results are probabilistic, and explicitly state the necessary assumptions and some useful initial consequences in Sections 5.2.1–5.2.2. In Sections 5.2.3–5.2.7, we analyze how errors propagate through each step of `rand_cholQR`. Finally, in Section 5.3, we provide the key theorems on the stability and accuracy of our `randQR` and `rand_cholQR` algorithms, and prove them primarily through the preparatory results from Section 5.2. Some readers may want to go directly to Section 5.3 for our main results.

5.1 Sources of Floating Point Error in `randQR`

We use a hat to denote a computed version of each of the matrix in all algorithms. First, errors are incurred when performing the matrix products $W = S_2 S_1 V$ in step 1 of Algorithm 1. Specifically, there exist error terms $\Delta\hat{W}_1$, $\Delta\hat{W}$ such that

$$\begin{aligned} \hat{W}_1 &= S_1 V + \Delta\hat{W}_1, \\ \hat{W} &= S_2 \hat{W}_1 + \Delta\hat{W} \cdot \end{aligned} \tag{15}$$

We can group these error terms together so that the computed $\hat{W}$ satisfies

$$\hat{W} = S_2 S_1 V + E_1, \tag{16}$$

where $E_1 = S_2\Delta\hat{W}_1 + \Delta\hat{W}$. The error term E_1 is analyzed in Section 5.2.3.

Applying Householder QR to $\hat{W}$ in step 2 incurs error E_2. Only the triangular factor $\hat{R}$ is needed, so some (exactly) orthogonal Q_{tmp} exists such that

$$Q_{tmp}\hat{R} = \hat{W} + E_2 = S_2S_1V + E_1 + E_2. \tag{17}$$

Analysis of E_2 is provided in Section 5.2.4.

In step 3, solving the triangular system $Q\hat{R} = V$ also creates errors. These are analyzed in a row-wise fashion in Section 5.2.5, taking the form

$$\hat{Q}_{i,:} = V_{i,:}(\hat{R} + \Delta\hat{R}_i)^{-1} \quad (i = 1, 2, \dots m), \tag{18}$$

where $\hat{Q}_{i,:}$ and $V_{i,:}$ denote the i^{th} rows of $\hat{Q}$ and V, respectively, and $\Delta\hat{R}_i$ is an error term incurred during the solution of the triangular systems. Finally, we recast the errors incurred in step 3 as $\hat{Q} = (V + \Delta\tilde{V})\hat{R}^{-1}$ in Sections 5.2.6–5.2.7, which simplifies the analysis of the orthogonality of $\hat{Q}$ in Section 5.3.

5.2 Assumptions and Preparatory Results for our Proofs

Let $V \in \mathbb{R}^{n\times m}$, $n \gg m$, and suppose $S_1 \in \mathbb{R}^{p_1\times m}$ and $S_2 \in \mathbb{R}^{p_2\times p_1}$ are (ε_1, d_1, m) and (ε_2, d_2, m) oblivious ℓ_2-subspace embeddings, respectively, generated independently. Define $d = d_1 + d_2 - d_1d_2$, $\varepsilon_L = \varepsilon_1 + \varepsilon_2 - \varepsilon_1\varepsilon_2$, $\varepsilon_H = \varepsilon_1 + \varepsilon_2 + \varepsilon_1\varepsilon_2$.

5.2.1 Assumptions

For the sake of organization, we define a set of assumptions stating that V is sufficiently numerically full rank (i.e., $\kappa(V) \lessapprox \mathbf{u}^{-1}$), $n \gg m$, and that the sketch matrices S_1, S_2 simultaneously satisfy the subspace embedding properties, ensuring equations (1)–(6), (8)–(13) hold with probability at least $1 - d$. We also impose an assumption that ϵ_L is sufficiently–but need not be too far–below 1, to obtain a positive lower bound on $\sigma_m(\hat{Q})$ while maintaining as general of a result as possible.

Assumptions 5.1. *Suppose $S_1 \in \mathbb{R}^{p_1\times m}$ and $S_2 \in \mathbb{R}^{p_2\times p_1}$ are (ε_1, d_1, m) and (ε_2, d_2, m) oblivious ℓ_2-subspace embeddings respectively, generated independently. Define $d = d_1 + d_2 - d_1d_2$, $\varepsilon_L = \varepsilon_1 + \varepsilon_2 - \varepsilon_1\varepsilon_2$, $\varepsilon_H = \varepsilon_1 + \varepsilon_2 + \varepsilon_1\varepsilon_2$, where*

$$\varepsilon_L \in \left[0, \frac{616}{625} - \frac{9}{625}\varepsilon_H\right).$$

Further, suppose $V \in \mathbb{R}^{n\times m}$ has full rank and $1 < m \le p_2 \le p_1 \le n$ where $nm\boldsymbol{u} \le \frac{1}{12}$, $p_1\sqrt{p_2}\boldsymbol{u} \le \frac{1}{12}$, and

$$\delta = \frac{383\left(\sqrt{1+\varepsilon_H}\ p_2m^{3/2} + \sqrt{m}\|S_2\|_2(p_1\sqrt{p_2}\sqrt{1+\varepsilon_1} + n\|S_1\|_F)\right)}{\sqrt{1-\varepsilon_L}}\boldsymbol{u}\ \kappa(V) \le 1\ . \tag{19}$$

The assumption that the integers $m, p_2, p_1,$ and n have the ordering

$$1 < m \leq p_2 \leq p_1 \leq n \tag{20}$$

is logical, otherwise the embeddings $S_2 \in \mathbb{R}^{p_2 \times p_1}$ and $S_1 \in \mathbb{R}^{p_1 \times n}$ project V into a larger space, defeating the purpose of sketching. Further, we assume

$$nm\mathbf{u} \leq \frac{1}{12} \quad \text{and} \quad p_1\sqrt{p_2}\mathbf{u} \leq \frac{1}{12}\ , \tag{21}$$

which are not directly implied by (19), but, depending of the values of p_1, p_2, m, $\|S_1\|_F$, and $\|S_2\|_2$, typically are consequences of it. Note that for the usual case of $\mathbf{u} = 2^{-52} \approx 10^{-16}$, the assumption (21) is easily satisfied, e.g,. with $n = 10^{12}$, $m = 10^2$.

As is customary in error analysis calculations, e.g., as in [11, 16, 34], we define for any positive integer k

$$\gamma_k := \frac{k\mathbf{u}}{1 - k\mathbf{u}}\ . \tag{22}$$

Provided $k\mathbf{u} < \frac{1}{11}$, it follows that $\gamma_k \leq 1.1k\mathbf{u}$. In particular, since $\kappa(V) \geq 1$, we can deduce from (19) that the constants that we use in our analysis are bounded as follows,

$$p_2 m^{3/2}\mathbf{u} \leq \frac{1}{383} < \frac{1}{11}\ , \tag{23}$$

so (20)–(23) imply

$$\begin{gathered}\gamma_n \leq 1.1n\mathbf{u},\ \gamma_m \leq 1.1m\mathbf{u},\ \gamma_{p_1} \leq 1.1p_1\mathbf{u},\ \gamma_{p_2 m} \leq 1.1p_2 m\mathbf{u}, \\ \gamma_{29p_2 m} \leq 31.9p_2 m\mathbf{u} < \frac{383}{12}p_2 m\mathbf{u}\ ,\end{gathered} \tag{24}$$

and

$$1 + \gamma_n < 1.1.$$

Observe that by (19), it follows that $383p_2 m^{3/2}\mathbf{u} \leq 1$. This implies that

$$\gamma_{29p_2 m} \leq 31.9p_2 m^{3/2}\mathbf{u} < \frac{383}{12}p_2 m^{3/2}\mathbf{u} \leq \frac{1}{12}, \tag{25}$$

and so

$$1 + 1.1p_2 m^{3/2}\mathbf{u} \leq 1 + 31.9p_2 m^{3/2}\mathbf{u} \leq 1 + \frac{1}{12} < 1.1\ . \tag{26}$$

Finally, we will repeatedly use well-known bounds relating the ℓ_2 and Frobenius norms,

$$\|X\|_2 \leq \|X\|_F \leq \sqrt{m}\|X\|_2, \tag{27}$$

$$\|XY\|_F \leq \|X\|_2\|Y\|_F, \quad \text{for any } X \in \mathbb{R}^{n\times m}, Y \in \mathbb{R}^{m\times k}. \tag{28}$$

Remark 5.1. Note that instances of $\|S_1\|_F$ and $\|S_2\|_2$ in (19) do not dominate the bound on $\kappa(V)$ imposed by (19). If $S_1 \in \mathbb{R}^{p_1 \times n}$ is an unscaled CountSketch and $S_2 \in \mathbb{R}^{p_2 \times p_1}$ is a scaled Gaussian sketch, there is a deterministic bound

$$\|S_1\|_2 \leq \|S_1\|_F \leq \sqrt{n},$$

and there is a probabilistic bound that there is some constant C such that

$$\|S_2\|_2 \leq 1 + C\left(\sqrt{\frac{p_1}{p_2}} + \frac{3}{\sqrt{p_2}}\right),$$

with probability at least $1-2e^{-9} \approx 0.9998$ [31]. Thus, in the case of $p_1 = O(m^2)$ and $p_2 = O(m)$, it follows that $\|S_1\|_2 = O(\sqrt{n})$ and $\|S_2\|_2 = O(\sqrt{m})$ with very high probability. Therefore, condition (19) ultimately requires

$$\delta \;\leq\; g(n, m, p_1, p_2)\mathbf{u}\,\kappa(V) \leq 1,$$

where g is some low-degree polynomial for reasonable choices of sketches S_1, S_2.

5.2.2 Notes on Probabilistic Results

While some bounds constructed throughout the proofs of our results are deterministic, several are probabilistic. Here, we address specifically which equations are not deterministic, their prerequisite assumptions, and the probabilities with which they hold.

Throughout the proofs, it is assumed that S_1 embeds the column space of V and S_2 embeds the column space of S_1V simultaneously, which happens with probability at least $1 - d = (1 - d_1)(1 - d_2)$ because S_1 and S_2 are independently generated (ε_1, d_1, m) and (ε_2, d_2, m) oblivious ℓ_2-subspace embeddings respectively. Therefore,

$$\begin{aligned}
\sqrt{1-\varepsilon_1}\|V\|_2 \leq \|S_1V\|_2 &\leq \sqrt{1+\varepsilon_1}\|V\|_2 \\
\sqrt{1-\varepsilon_L}\|V\|_2 \leq \|S_2S_1V\|_2 &\leq \sqrt{1+\varepsilon_H}\|V\|_2 \\
\sqrt{1-\varepsilon_1}\|V\|_F \leq \|S_1V\|_F &\leq \sqrt{1+\varepsilon_1}\|V\|_F \qquad (29)\\
\sqrt{1-\varepsilon_L}\|V\|_F \leq \|S_2S_1V\|_F &\leq \sqrt{1+\varepsilon_H}\|V\|_F, \qquad (30)\\
\sqrt{1-\varepsilon_L}\;\sigma_{min}(V) &\leq \sigma_{min}(S_2S_1V), \qquad (31)
\end{aligned}$$

and

$$\begin{aligned}
(1+\varepsilon_H)^{-1/2}\;\sigma_{min}(S_2S_1V) \leq \sigma_{min}(V) \leq \sigma_{max}(V) \qquad (32)\\
\leq (1-\varepsilon_L)^{-1/2}\;\sigma_{max}(S_2S_1V),
\end{aligned}$$

along with the analogous statements for matrices whose column spaces are identical to V, will simultaneously hold with probability at least $1 - d$. Specifically, this implies equations (36), (39), (45), (46), (53), and (60) simultaneously hold with probability at least $1 - d$, which are used to build all of the results from (36)–(65) that are prerequisite to prove Theorems 5.1–5.4, all of which hold with high probability.

5.2.3 Forward Error in matrix-matrix multiplication S_2S_1V

By [16, Section 3.5], for $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times k}$, $C = AB$ executed in floating point satisfies

$$\hat{C} = AB + \Delta C, \quad |\Delta C| < \gamma_n |A||B| \ ,$$

where γ_n is defined in (22). Thus, in floating point, step 1 of Algorithm 1 becomes:

$$\hat{W}_1 = S_1V + \Delta\hat{W}_1, \quad |\Delta\hat{W}_1| < \gamma_n|S_1||V| \ , \tag{33}$$

$$\hat{W} = S_2\hat{W}_1 + \Delta\hat{W}, \quad |\Delta\hat{W}| < \gamma_{p_1}|S_2||\hat{W}_1| = \gamma_{p_1}|S_2||S_1V + \Delta\hat{W}_1| \cdot \tag{34}$$

In other words,

$$\hat{W} = S_2S_1V + E_1, \tag{35}$$

where the forward error of these matrix-matrix products $E_!$ is defined as:

$$E_1 = \hat{W} - W = \Delta\hat{W} + S_2\Delta\hat{W}_1 \cdot$$

By (27), (28), (29), (33), and (34),

$$\begin{aligned}
\|E_1\|_2 &\le \|\Delta\hat{W}\|_2 + \|S_2\|_2\|\Delta\hat{W}_1\|_2 \\
&\le \|\Delta\hat{W}\|_F + \|S_2\|_2\|\Delta\hat{W}_1\|_F \\
&\le \gamma_{p_1}\|S_2\|_F\|S_1V + \Delta\hat{W}_1\|_F + \|S_2\|_2\|\Delta\hat{W}_1\|_F \\
&\le \gamma_{p_1}\|S_2\|_F(\|S_1V\|_F + \|\Delta\hat{W}_1\|_F) + \|S_2\|_2\|\Delta\hat{W}_1\|_F \\
&\le \gamma_{p_1}\|S_2\|_F(\sqrt{1+\varepsilon_1}\|V\|_F + \|\Delta\hat{W}_1\|_F) + \|S_2\|_2\|\Delta\hat{W}_1\|_F \\
&\le \sqrt{p_2}\gamma_{p_1}\|S_2\|_2(\sqrt{1+\varepsilon_1}\|V\|_F + \|\Delta\hat{W}_1\|_F) + \|S_2\|_2\|\Delta\hat{W}_1\|_F \\
&= \sqrt{p_2}\gamma_{p_1}\|S_2\|_2\sqrt{1+\varepsilon_1}\|V\|_F + \|S_2\|_2(1+\sqrt{p_2}\gamma_{p_1})\|\Delta\hat{W}_1\|_F \\
&\le \sqrt{p_2}\gamma_{p_1}\|S_2\|_2\sqrt{1+\varepsilon_1}\|V\|_F + \|S_2\|_2(1+\sqrt{p_2}\gamma_{p_1})\gamma_n\|S_1\|_F\|V\|_F \\
&= \|S_2\|_2\left(\sqrt{p_2}\gamma_{p_1}\sqrt{1+\varepsilon_1} + \gamma_n(1+\sqrt{p_2}\gamma_{p_1})\|S_1\|_F\right)\|V\|_F \\
&\le \sqrt{m}\|S_2\|_2\left(\sqrt{p_2}\gamma_{p_1}\sqrt{1+\varepsilon_1} + \gamma_n(1+\sqrt{p_2}\gamma_{p_1})\|S_1\|_F\right)\|V\|_2.
\end{aligned}$$

Notice (21) and (24) imply $\gamma_n(1+\sqrt{p_2}\gamma_{p_1}) < 1.21n\mathbf{u}$. Hence,

$$\|E_1\|_2 \le \sqrt{m}\mathbf{u}\|S_2\|_2(1.1p_1\sqrt{p_2}\sqrt{1+\varepsilon_1} + 1.21n\|S_1\|_F)\|V\|_2. \tag{36}$$

Lemma 5.1. *If S_1 is a ε_1 embedding of the column space of V and S_2 is a ε_2 embedding of the column space of S_1V, then*

$$\frac{12}{\sqrt{1-\varepsilon_L}}\left(31.9\sqrt{1+\varepsilon_H}\ p_2m^{3/2}\boldsymbol{u}\ \kappa(V) + 1.1\|E_1\|_2\ \sigma_m(V)^{-1}\right) \le \delta \le 1 \ . \tag{37}$$

Proof. Follows directly from using (36) and the definition of δ in (19). □

5.2.4 Backward Error of Householder QR of $\hat{W}$

By [16, Theorem 19.4], Householder QR of $\hat{W} \in \mathbb{R}^{p_2 \times m}$ returns a triangular $\hat{R} \in \mathbb{R}^{m \times m}$ so that some orthogonal $Q_{tmp} \in \mathbb{R}^{p_2 \times m}$ satisfies,

$$\hat{W} + E_2 = Q_{tmp}\hat{R}, \quad \|(E_2)_j\|_2 \leq \gamma_{29p_2m}\|\hat{w}_j\|_2, \text{ for } j = 1, \ldots, m. \tag{38}$$

We mention that in [16], the bound in (38) has γ_{cp_2m}, for some small integer constant c. It is mentioned there that the exact value of c is "unimportant" for the general analysis. A careful look at the proof of [16, Theorem 19.4] indicates that one can take $c = 29$, and this is what we have done.

By (27), (35), and the embedding properties of S_2S_1 on V given in (30),

$$\begin{aligned}\|\hat{W}\|_F &\leq \|S_2S_1V\|_F + \|E_1\|_F \leq \sqrt{1+\epsilon_H}\|V\|_F + \|E_1\|_F \\ &\leq \sqrt{m}\left(\sqrt{1+\epsilon_H}\|V\|_2 + \|E_1\|_2\right),\end{aligned} \tag{39}$$

and therefore by (38),

$$\|E_2\|_F \leq \gamma_{29p_2m}\|\hat{W}\|_F \leq \gamma_{29p_2m}\sqrt{m}\left(\sqrt{1+\epsilon_H}\|V\|_2 + \|E_1\|_2\right) \ , \tag{40}$$

with probability at least $1 - d$. Finally, by (24) and (25),

$$\begin{aligned}\|E_2\|_2 \leq \|E_2\|_F &\leq 31.9p_2m\mathbf{u}(\sqrt{1+\epsilon_H}\|V\|_F + \|E_1\|_F) \\ &\leq 31.9p_2m^{3/2}\mathbf{u}(\sqrt{1+\epsilon_H}\|V\|_2 + \|E_1\|_2) \\ &= 31.9p_2m^{3/2}\mathbf{u}\sqrt{1+\epsilon_H}\|V\|_2 + 31.9p_2m^{3/2}\mathbf{u}\|E_1\|_2 \\ &\leq 31.9p_2m^{3/2}\mathbf{u}\sqrt{1+\epsilon_H}\|V\|_2 + 0.1\|E_1\|_2 \ ,\end{aligned} \tag{41}$$

with probability at least $1 - d$.

5.2.5 Backward Error of the Forward Substitution

In Step 3 of `randQR`, we solve for Q via the triangular system $Q\hat{R} = V$. By [16, Theorem 8.5], in floating point, $\hat{Q}_{i,:}$ satisfies

$$\hat{Q}_{i,:}(\hat{R} + \Delta R_i) = V_{i,:}, \quad |\Delta R_i| < \gamma_m|\hat{R}| \text{ for } i = 1, \ldots n. \tag{42}$$

While it would be convenient to simply write $\hat{Q}(R + \Delta R) = V$ for some ΔR, each ΔR_i error incurred depends on each right hand side of (42), and therefore each row must be accounted for separately. For each $i = 1, \ldots, n$,

$$\|\Delta\hat{R}_i\|_2 \leq \|\Delta\hat{R}_i\|_F = \||\Delta\hat{R}_i|\|_F < \gamma_m\||\hat{R}|\|_F = \gamma_m\|\hat{R}\|_F \ . \tag{43}$$

By (38), (40), and the orthogonality of Q_{tmp}, it follows that

$$\|\hat{R}\|_F = \|Q_{tmp}\hat{R}\|_F = \|\hat{W} + E_2\|_F \leq (1 + \gamma_{29p_2m})\|\hat{W}\|_F, \tag{44}$$

$$\begin{aligned}\|\hat{R}\|_2 &= \|\hat{W} + E_2\|_2 = \|S_2S_1V + E_1 + E_2\|_2 \\ &\leq \sqrt{1+\varepsilon_H}\|V\|_2 + \|E_1\|_2 + \|E_2\|_2.\end{aligned} \tag{45}$$

Therefore, by (23)–(26), (39), (43), and (44),

$$\begin{aligned}\|\Delta\hat{R}_i\|_2 &\leq 1.1m^{3/2}\mathbf{u}(1 + 31.9p_2m\mathbf{u})\left(\sqrt{1+\epsilon_H}\|V\|_2 + \|E_1\|_2\right) \\ &\leq 1.21m^{3/2}\mathbf{u}\left(\sqrt{1+\epsilon_H}\|V\|_2 + \|E_1\|_2\right).\end{aligned}$$

5.2.6 Bounding the 2-norm of $\hat{R}^{-1}$ and $V\hat{R}^{-1}$

By (31), (16), and Weyl's inequality [33], with probability at least $1-d$,

$$\begin{aligned}\sigma_m(\hat{W}+E_2) &\geq \sigma_m(\hat{W}) - \|E_2\|_2 \geq \sigma_m(S_2S_1V) - (\|E_1\|_2 + \|E_2\|_2) \\ &\geq \sqrt{1-\epsilon_L}\ \sigma_m(V) - (\|E_1\|_2 + \|E_2\|_2). \end{aligned} \tag{46}$$

By Lemma 5.1, and the fact that the fact that $\|V\|_2 = \kappa(V)\ \sigma_m(V)$,

$$31.9p_2m^{3/2}\mathbf{u}\sqrt{1+\varepsilon_H}\|V\|_2 + 1.1\|E_1\|_2 \leq \frac{\sqrt{1-\epsilon_L}}{12}\sigma_m(V)\ \delta. \tag{47}$$

Combining (41) and (47) and the assumption that $\delta \leq 1$, results in:

$$\begin{aligned}\|E_1\|_2 + \|E_2\|_2 &\leq 31.9p_2m^{3/2}\mathbf{u}\sqrt{1+\varepsilon_H}\|V\|_2 + 1.1\|E_1\|_2 \\ &\leq \frac{\sqrt{1-\epsilon_L}}{12}\sigma_m(V)\delta \leq \frac{\sqrt{1-\epsilon_L}}{12}\sigma_m(V), \end{aligned} \tag{48}$$

so by (46) and (48),

$$\sigma_m(\hat{R}) = \sigma_m(Q_{tmp}\hat{R}) = \sigma_m(\hat{W}+E_2) \geq \frac{11\sqrt{1-\epsilon_L}}{12}\ \sigma_m(V). \tag{49}$$

Therefore, by (49)

$$\|\hat{R}^{-1}\|_2 \leq \frac{12}{11\sqrt{1-\epsilon_L}}\left(\sigma_m(V)\right)^{-1}. \tag{50}$$

By (17), we have that Step 2 of `randQR` satisfies

$$S_2S_1V\hat{R}^{-1} = Q_{tmp} - (E_1+E_2)\hat{R}^{-1}. \tag{51}$$

Thus, by (48), (50), (51), the fact that Q_{tmp} is orthogonal,

$$\|S_2S_1V\hat{R}^{-1}\|_2 \leq \|Q_{tmp}\|_2 + (\|E_1\|_2 + \|E_2\|_2)\|\hat{R}^{-1}\|_2 \leq \frac{12}{11}\ . \tag{52}$$

Observe that V and $V\hat{R}^{-1}$ have the same column space; therefore if S_1, S_2 embed the column space of V, they will also embed the column space of $V\hat{R}^{-1}$. Therefore, by (8),

$$\|V\hat{R}^{-1}\|_2 \leq \frac{1}{\sqrt{1-\epsilon_L}}\|S_2S_1V\hat{R}^{-1}\|_2 \leq \frac{12}{11\sqrt{1-\epsilon_L}}\ , \tag{53}$$

with probability at least $1-d$.

5.2.7 Evaluation of the Backward Error $\Delta\tilde{V} = \hat{Q}\hat{R} - V$

Instead of using backward errors ΔR_i for each triangular solve in equation (18), we capture the errors of each triangular solve in a matrix $\Delta\tilde{V}$, where

$$\hat{Q} = (V + \Delta\tilde{V})\hat{R}^{-1} \iff \hat{Q}\hat{R} = V + \Delta\tilde{V}. \tag{54}$$

From (18), we have $\hat{Q}_{i,:}(R + \Delta R_i) = V_{i,:}$. Then $\Delta\tilde{V}$ can be defined row-wise,

$$\Delta\tilde{V}_{i,:} = -\hat{Q}_{i,:}\Delta R_i. \tag{55}$$

Thus, by (42), $|\Delta\tilde{V}_{i,:}| \leq 1.1m\mathbf{u}|\hat{Q}_{i,:}||\hat{R}|$, and so $|\Delta\tilde{V}| \leq 1.1m\mathbf{u}|\hat{Q}||\hat{R}|$, hence,

$$|\Delta\tilde{V}_{:,i}| \leq 1.1m\mathbf{u}|\hat{Q}||\hat{R}_{:,i}|.$$

From this, it follows that for each column $i = 1, \ldots, m$,

$$\begin{aligned}\|\Delta\tilde{V}_{:,i}\|_2 &\leq 1.1m\mathbf{u}\||\hat{Q}|\|_2\||\hat{R}_{:,i}|\|_2 \leq 1.1m\mathbf{u}\||\hat{Q}|\|_F\||\hat{R}_{:,i}|\|_2 \\ &= 1.1m\mathbf{u}\|\hat{Q}\|_F\|\hat{R}_{:,i}\|_2 \leq 1.1m^{3/2}\mathbf{u}\|\hat{Q}\|_2\|\hat{R}_{:,i}\|_2,\end{aligned}$$

and therefore by (25), (39), and (44),

$$\begin{aligned}\|\Delta\tilde{V}\|_2 \leq \|\Delta\tilde{V}\|_F &\leq 1.1m^{3/2}\mathbf{u}\|\hat{Q}\|_2\|\hat{R}\|_F \\ &\leq 1.1m^{3/2}\mathbf{u}\|\hat{Q}\|_2(1+\gamma_{29p_2m})\|\hat{W}\|_F \\ &\leq 1.1m^{3/2}\mathbf{u}\|\hat{Q}\|_2(1+\gamma_{29p_2m})\sqrt{m}\left(\sqrt{1+\varepsilon_H}\|V\|_2 + \|E_1\|_2\right) \\ &\leq 1.21m^2\mathbf{u}\|\hat{Q}\|_2\left(\sqrt{1+\varepsilon_H}\|V\|_2 + \|E_1\|_2\right)\ .\end{aligned} \tag{56}$$

By (20) it follows that $1.1\sqrt{m} \leq p_2$, and so by (26), (48), and (56),

$$\begin{aligned}\|\Delta\tilde{V}\|_F &\leq 1.21m^2\mathbf{u}\|\hat{Q}\|_2\left(\sqrt{1+\varepsilon_H}\|V\|_2 + \|E_1\|_2\right) \\ &= \|\hat{Q}\|_2\left(1.21m^2\mathbf{u}\sqrt{1+\varepsilon_H}\|V\|_2 + 1.21m^2\mathbf{u}\|E_1\|_2\right) \\ &\leq \|\hat{Q}\|_2\left(1.1p_2m^{3/2}\mathbf{u}\sqrt{1+\varepsilon_H}\|V\|_2 + 1.1p_2m^{3/2}\mathbf{u}\|E_1\|_2\right) \\ &\leq \|\hat{Q}\|_2\left(1.1p_2m^{3/2}\mathbf{u}\sqrt{1+\varepsilon_H}\|V\|_2 + (1+1.1p_2m^{3/2}\mathbf{u})\|E_1\|_2\right) \\ &\leq \|\hat{Q}\|_2\left(1.1p_2m^{3/2}\mathbf{u}\sqrt{1+\varepsilon_H}\|V\|_2 + 1.1\|E_1\|_2\right) \\ &\leq \|\hat{Q}\|_2\left(31.9p_2m^{3/2}\mathbf{u}\sqrt{1+\varepsilon_H}\|V\|_2 + 1.1\|E_1\|_2\right) \\ &\leq \|\hat{Q}\|_2\frac{\sqrt{1-\epsilon_L}}{12}\sigma_m(V)\ \delta\ .\end{aligned} \tag{57}$$

The remaining issue to resolve is that the bound on $\|\Delta\tilde{V}\|_F$ in (57) requires knowledge of $\|\hat{Q}\|_2$, which we have not yet found. Combining (19), (50), and (57) gives,

$$\begin{aligned}\|\hat{Q} - V\hat{R}^{-1}\|_F = \|\Delta\tilde{V}\hat{R}^{-1}\|_F &\leq \|\Delta\tilde{V}\|_F\|\hat{R}^{-1}\|_2 \\ &\leq \|\hat{Q}\|_2\frac{\sqrt{1-\epsilon_L}}{12}\sigma_m(V)\delta\frac{12}{11\sqrt{1-\epsilon_L}}\left(\sigma_m(V)\right)^{-1} \\ &= \frac{\delta}{11}\|\hat{Q}\|_2 \leq \frac{1}{11}\|\hat{Q}\|_2.\end{aligned} \tag{58}$$

Now, by (48), (50), and (51),

$$\|S_2 S_1 V \hat{R}^{-1} - Q_{tmp}\|_2 \leq (\|E_1\|_2 + \|E_2\|_2)\, \|\hat{R}^{-1}\|_2 \leq \frac{\delta}{11} \cdot \tag{59}$$

Applying Weyl's inequality to (59) and the fact that Q_{tmp} is orthogonal yields,

$$1 - \frac{\delta}{11} \leq \sigma_m(S_2 S_1 V \hat{R}^{-1}) \leq \sigma_1(S_2 S_1 V \hat{R}^{-1}) \leq 1 + \frac{\delta}{11} \cdot$$

Since V and $V\hat{R}^{-1}$ have identical column spaces and S_1,S_2 embed the column space of V, the embedding properties in (32) also apply to $V\hat{R}^{-1}$, and so

$$\frac{1 - \frac{\delta}{11}}{\sqrt{1+\epsilon_H}} \leq \sigma_m(V\hat{R}^{-1}) \leq \sigma_1(V\hat{R}^{-1}) \leq \frac{1 + \frac{\delta}{11}}{\sqrt{1-\epsilon_L}} \leq \frac{12}{11\sqrt{1-\epsilon_L}} \cdot \tag{60}$$

Then, we can use Weyl's inequality again on $\hat{Q} - V\hat{R}^{-1}$. In particular,

$$\sigma_m(V\hat{R}^{-1}) - \|\hat{Q} - V\hat{R}^{-1}\|_2 \leq \sigma_m(\hat{Q}) \leq \sigma_1(\hat{Q}) \leq \sigma_1(V\hat{R}^{-1}) + \|\hat{Q} - V\hat{R}^{-1}\|_2 \cdot \tag{61}$$

Then, by (58), (60), and (61),

$$\|\hat{Q}\|_2 = \sigma_1(\hat{Q}) \leq \sigma_1(V\hat{R}^{-1}) + \|\hat{Q} - V\hat{R}^{-1}\|_2 \leq \frac{12}{11\sqrt{1-\epsilon_L}} + \frac{1}{11}\|\hat{Q}\|_2, \tag{62}$$

and thus,

$$\|\hat{Q}\|_2 \leq \frac{6}{5\sqrt{1-\epsilon_L}} \cdot \tag{63}$$

Then, we obtain from (58),

$$\|\hat{Q} - V\hat{R}^{-1}\|_2 = \|\Delta\tilde{V}\hat{R}^{-1}\|_2 \leq \|\Delta\tilde{V}\hat{R}^{-1}\|_F \leq \frac{\delta}{11}\|\hat{Q}\|_2 \leq \frac{6\delta}{55\sqrt{1-\epsilon_L}}. \tag{64}$$

5.2.8 Bounding $\|S_2 S_1 \Delta\tilde{V}\hat{R}^{-1}\|_2$

If no additional assumptions on the embedding of S_1, S_2 are made, clearly it follows that

$$\|S_2 S_1 \Delta\tilde{V}\hat{R}^{-1}\|_2 \leq \|S_2\|_2 \|S_1\|_2 \|\Delta\tilde{V}\hat{R}^{-1}\|_2 \leq \frac{6\|S_2\|_2\|S_1\|_2}{55\sqrt{1-\epsilon_L}}\, \delta. \tag{65}$$

Alternatively, if we assume S_1, S_2 embed $\Delta\tilde{V}\hat{R}^{-1}$, by (8),

$$\|S_2 S_1 \Delta\tilde{V}\hat{R}^{-1}\|_2 \leq \sqrt{1+\epsilon_H}\|\Delta\tilde{V}\hat{R}^{-1}\|_2 \leq \frac{6\sqrt{1+\epsilon_H}}{55\sqrt{1-\epsilon_L}}\, \delta. \tag{66}$$

5.3 Key Theoretical Results

We begin by re-stating the assumptions of the theoretical results for readability.

Assumptions 5.1. *Suppose* $S_1 \in \mathbb{R}^{p_1 \times m}$ *and* $S_2 \in \mathbb{R}^{p_2 \times p_1}$ *are* (ε_1, d_1, m) *and* (ε_2, d_2, m) *oblivious* ℓ_2*-subspace embeddings respectively, generated independently. Define* $d = d_1 + d_2 - d_1 d_2$, $\varepsilon_L = \varepsilon_1 + \varepsilon_2 - \varepsilon_1\varepsilon_2$, $\varepsilon_H = \varepsilon_1 + \varepsilon_2 + \varepsilon_1\varepsilon_2$, *where*

$$\varepsilon_L \in \left[0, \frac{616}{625} - \frac{9}{625}\varepsilon_H\right).$$

Further, suppose $V \in \mathbb{R}^{n\times m}$ *has full rank and* $1 < m \le p_2 \le p_1 \le n$ *where* $nm\boldsymbol{u} \le \frac{1}{12}$, $p_1\sqrt{p_2}\boldsymbol{u} \le \frac{1}{12}$, *and*

$$\delta = \frac{383\left(\sqrt{1+\varepsilon_H}\ p_2 m^{3/2} + \sqrt{m}\|S_2\|_2(p_1\sqrt{p_2}\sqrt{1+\varepsilon_1} + n\|S_1\|_F)\right)}{\sqrt{1-\varepsilon_L}}\boldsymbol{u}\ \kappa(V) \le 1\ .$$

Remark 4.1 indicates that in exact arithmetic, `randQR` yields a matrix Q that is orthogonal with respect to $\langle S_2S_1\cdot, S_2S_1\cdot\rangle$. We show next that provided V has full numerical rank, then in floating point arithmetic, the orthogonality error of the matrix $\hat{Q}$ generated by `randQR` measured in $\langle S_2S_1\cdot, S_2S_1\cdot\rangle$ is $O(\mathbf{u})\kappa(V)$, and the factorization error is $O(\mathbf{u})\|V\|_2$ with high probability.

Theorem 5.1 (`randQR` Errors)**.** *Suppose Assumptions 5.1 are satisfied. Then the* $\hat{Q}, \hat{R}$ *factors obtained with Algorithm 1 (`randQR`) satisfy*

$$\|V - \hat{Q}\hat{R}\|_2 \le \frac{\delta}{10}\sigma_m(V), \tag{67}$$

and

$$\begin{aligned}&\|(S_2S_1\hat{Q})^T(S_2S_1\hat{Q}) - I\|_2\\ &\quad\le 2\frac{\delta}{11} + \left(\frac{\delta}{11}\right)^2 + \frac{24}{11}\frac{6\|S_2\|_2\|S_1\|_2}{55\sqrt{1-\epsilon_L}}\ \delta + \left(\frac{6\|S_2\|_2\|S_1\|_2}{55\sqrt{1-\epsilon_L}}\ \delta\right)^2.\end{aligned} \tag{68}$$

with probability at least $1 - d$*, where* δ *is defined in* (19)*. Furthermore,*

$$\|(S_2S_1\hat{Q})^T(S_2S_1\hat{Q}) - I\|_2 \le 3\delta \tag{69}$$

with probability at least $(1-d)^2$*.*

Proof. Equation (67) follows by combining (57) and (63), since $\Delta\tilde{V} = \hat{Q}\hat{R} - V$, and this holds with probability at least $1 - d$ because (57) and (63) hold with this probability, as discussed in Section 5.2.2.

Observe that by (17), we have $S_2S_1V = Q_{tmp}\hat{R} - (E_1 + E_2)$, and thus

$$\begin{aligned}&(S_2S_1V)^T(S_2S_1V) =\\ &\hat{R}^T\hat{R} - (E_1+E_2)^TQ_{tmp}\hat{R} - \hat{R}^TQ_{tmp}^T(E_1+E_2) + (E_1+E_2)^T(E_1+E_2).\end{aligned}$$

Using (54) to expand $S_2S_1\hat{Q} = (S_2S_1V + S_2S_1\Delta\tilde{V})\hat{R}^{-1}$, we obtain

$$\begin{aligned}(S_2S_1\hat{Q})^T(S_2S_1\hat{Q}) &= I - \hat{R}^{-T}(E_1+E_2)^TQ_{tmp} - Q_{tmp}^T(E_1+E_2)\hat{R}^{-1} \\ &+ \hat{R}^{-T}(E_1+E_2)^T(E_1+E_2)\hat{R}^{-1} + (S_2S_1\Delta\tilde{V}\hat{R}^{-1})^T(S_2S_1V\hat{R}^{-1}) \\ &+ (S_2S_1V\hat{R}^{-1})^TS_2S_1\Delta\tilde{V}\hat{R}^{-1} + (S_2S_1\Delta\tilde{V}\hat{R}^{-1})^T(S_2S_1\Delta\tilde{V}\hat{R}^{-1}).\end{aligned}$$

Therefore, by (48), (50), and (52),

$$\begin{aligned}&\|(S_2S_1\hat{Q})^T(S_2S_1\hat{Q}) - I\|_2 \\ &\qquad\leq 2(\|E_1\|_2 + \|E_2\|_2)\|\hat{R}^{-1}\|_2 + (\|E_1\|_2 + \|E_2\|_2)^2\|\hat{R}^{-1}\|_2^2 \\ &\qquad+ 2\|S_2S_1\Delta\tilde{V}\hat{R}^{-1}\|_2\|S_2S_1V\hat{R}^{-1}\|_2 + \|S_2S_1\Delta\tilde{V}\hat{R}^{-1}\|_2^2 \\ &\qquad\leq 2\frac{\delta}{11} + \left(\frac{\delta}{11}\right)^2 + \frac{24}{11}\|S_2S_1\Delta\tilde{V}\hat{R}^{-1}\|_2 + \|S_2S_1\Delta\tilde{V}\hat{R}^{-1}\|_2^2. \quad (70)\end{aligned}$$

Observe that (48), (50), and (52), simultaneously hold with probability at least $1-d$, because they rely on V and S_1V being simultaneously embedded by S_1 and S_2 respectively, which with this probability occurs, as discussed in Section 5.2.2. Thus, (70) holds with probability at least $1-d$. Observe that (65) requires no further assumptions on the embedding properties of S_1, S_2 and so applying (65) to (70) gives

$$\begin{aligned}&\|(S_2S_1\hat{Q})^T(S_2S_1\hat{Q}) - I\|_2 \\ &\qquad\leq 2\frac{\delta}{11} + \left(\frac{\delta}{11}\right)^2 + \frac{24}{11}\|S_2S_1\Delta\tilde{V}\hat{R}^{-1}\|_2 + \|S_2S_1\Delta\tilde{V}\hat{R}^{-1}\|_2^2 \\ &\qquad\leq 2\frac{\delta}{11} + \left(\frac{\delta}{11}\right)^2 + \frac{24}{11}\frac{6\|S_2\|_2\|S_1\|_2}{55\sqrt{1-\epsilon_L}}\,\delta + \left(\frac{6\|S_2\|_2\|S_1\|_2}{55\sqrt{1-\epsilon_L}}\,\delta\right)^2,\end{aligned}$$

with probability at least $1-d$, producing result (68).

On the other hand, observe that (66) requires not only the assumption that S_1, S_2 simultaneously embed V and S_1V respectively, but also that the sketch matrices embed $\Delta\tilde{V}\hat{R}^{-1}$ and $S_1\Delta\tilde{V}\hat{R}^{-1}$ respectively. Thus, (66) and (70) simultaneously hold with probability at least $(1-d)^2$, and the result of applying both of these results together yields,

$$\begin{aligned}&\|(S_2S_1\hat{Q})^T(S_2S_1\hat{Q}) - I\|_2 \\ &\qquad\leq 2\frac{\delta}{11} + \left(\frac{\delta}{11}\right)^2 + \frac{24}{11}\|S_2S_1\Delta\tilde{V}\hat{R}^{-1}\|_2 + \|S_2S_1\Delta\tilde{V}\hat{R}^{-1}\|_2^2 \\ &\qquad\leq 2\frac{\delta}{11} + \left(\frac{\delta}{11}\right)^2 + \frac{24}{11}\frac{6\sqrt{1+\varepsilon_H}}{55\sqrt{1-\epsilon_L}}\,\delta + \left(\frac{6\sqrt{1+\varepsilon_H}}{55\sqrt{1-\epsilon_L}}\,\delta\right)^2, \quad (71)\end{aligned}$$

with probability at least $(1-d)^2$. A useful consequence of Assumptions 5.1 is that

$$1 - \varepsilon_L > \frac{9}{625}(1+\varepsilon_H),$$

and thus

$$\frac{1+\varepsilon_H}{1-\varepsilon_L} < \frac{625}{9} \Rightarrow \sqrt{\frac{1+\varepsilon_H}{1-\varepsilon_L}} < \frac{25}{3} \ . \tag{72}$$

Applying (72) to (71) (which holds with probability at least $(1-d)^2$) along with the fact that $\delta \leq 1$ from (19) implies $\delta^2 \leq \delta$, and therefore,

$$\begin{aligned}
&\|(S_2 S_1 \hat{Q})^T (S_2 S_1 \hat{Q}) - I\|_2 \\
&\qquad \leq 2\frac{\delta}{11} + \left(\frac{\delta}{11}\right)^2 + \frac{24}{11}\frac{6\sqrt{1+\varepsilon_H}}{55\sqrt{1-\epsilon_L}}\ \delta + \left(\frac{6\sqrt{1+\varepsilon_H}}{55\sqrt{1-\epsilon_L}}\ \delta\right)^2 \\
&\qquad \leq \frac{2}{11}\delta + \left(\frac{1}{11}\right)^2 \delta + \frac{24}{11}\frac{6\cdot 25}{55\cdot 3}\ \delta + \left(\frac{6\cdot 25}{55\cdot 3}\right)^2 \delta \\
&\qquad = \frac{2\cdot 11\cdot 55^2\cdot 3^2 + 55^2\cdot 3^2 + 24\cdot 6\cdot 25\cdot 11\cdot 55\cdot 3 + 6^2\cdot 25^2\cdot 11^2}{11^2\cdot 55^2\cdot 3^2}\delta \\
&\qquad = 3\delta \ ,
\end{aligned}$$

with probability at least $(1-d)^2$, and thus result (69) follows. □

Similar to the analysis of the condition number of Q generated by `randQR` in exact arithmetic in Section 4, we show next that provided that V has full numerical rank, then $\hat{Q}$ generated by `randQR` in floating point arithmetic also satisfies $\kappa(\hat{Q}) = O(1)$.

Theorem 5.2 (Conditioning of `randQR`)**.** *Suppose Assumptions 5.1 are satisfied. Then with probability at least $1-d$, the $\hat{Q}$ matrix obtained with Algorithm 1 (`randQR`) has condition number $\kappa(\hat{Q}) = O(1)$. In fact,*

$$\kappa(\hat{Q}) \leq \frac{33}{25\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 3}. \tag{73}$$

Proof. As a direct consequence of (60), (61), (64), and the fact that $\delta \leq 1$,

$$\begin{aligned}
\sigma_m(\hat{Q}) &\geq \sigma_m(V\hat{R}^{-1}) - \|\hat{Q} - V\hat{R}^{-1}\|_2 \geq \frac{1-\frac{\delta}{11}}{\sqrt{1+\epsilon_H}} - \frac{6\delta}{55\sqrt{1-\epsilon_L}} \\
&\geq \frac{10}{11\sqrt{1+\epsilon_H}} - \frac{6}{55\sqrt{1-\epsilon_L}} \ .
\end{aligned}$$

Additionally, we found in (63) that

$$\sigma_1(\hat{Q}) = \|\hat{Q}\|_2 \leq \frac{6}{5\sqrt{1-\epsilon_L}} \ .$$

Thus,

$$\kappa(\hat{Q}) = \frac{\sigma_1(\hat{Q})}{\sigma_m(\hat{Q})} \leq \frac{33}{25\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 3} \ ,$$

which is the desired result. Since the intermediate results (60), (61), (63), and (64) simultaneously hold with probability at least $1-d$, as discussed in Section 5.2.2, the final result (73) holds with this probability as well. □

In the following result we show that `rand_cholQR`(V) (Algorithm 3) produces a factor $\hat{Q}$ that is orthogonal in the Euclidean inner product up to a factor of $O(\mathbf{u})$ and has a factorization error of $O(\mathbf{u})\|V\|_2$ for any numerically full rank V.

Theorem 5.3 (`rand_cholQR` Errors)**.** *Suppose Assumptions 5.1 are satisfied. Then with probability at least $1-d$, the $\hat{Q},\hat{R}$ factors obtained with Algorithm 3 (`rand_cholQR`) has $O(\boldsymbol{u})$ orthogonality error and $O(\boldsymbol{u})\|V\|_2$ factorization error. More specifically,*

$$\|\hat{Q}^T\hat{Q}-I\|_2 \le \frac{5445}{\left(25\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}}-3\right)^2}\left(nm+m(m+1)\right)\boldsymbol{u}, \tag{74}$$

$$\begin{aligned}\|V-\hat{Q}\hat{R}\|_2 \le &\left(\frac{56}{25\frac{1-\varepsilon_L}{\sqrt{1+\epsilon_H}}-3\sqrt{1-\varepsilon_L}}+\frac{1.5}{\sqrt{1-\varepsilon_L}}\sqrt{1+\frac{5445(nm+m(m+1))\boldsymbol{u}}{\left(25\sqrt{\frac{1-\varepsilon_L}{1+\varepsilon_H}}-3\right)^2}}\right)\\ &\left(\sqrt{1+\varepsilon_H}\|V\|_2+\frac{\sqrt{1-\varepsilon_L}}{12}\sigma_m(V)\delta\right)m^2\boldsymbol{u}+\frac{\delta}{10}\sigma_m(V),\end{aligned} \tag{75}$$

where δ is bounded as in (19)*.*

Proof. In Algorithm 3, we obtain $\hat{Q}_0,\hat{R}_0$ from `randQR` (so that the results in Section 5.2 apply to $\hat{Q}_0,\hat{R}_0$), and then obtain $\hat{Q},\hat{R}$ where $\hat{R}=\mathrm{fl}(\hat{R}_1\hat{R}_0)$ and $\hat{Q},\hat{R}_1$ are the outputs of Cholesky QR applied to $\hat{Q}_0$. As a direct consequence of Theorem 5.2, $\hat{Q}_0$ arising from Step 1 of Algorithm 3 satisfies

$$\kappa(\hat{Q}_0) \le \frac{33}{25\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}}-3}\ .$$

By [36, Lemma 3.1], it follows that step 2 of Algorithm 3 gives $\hat{Q}$ satisfying

$$\begin{aligned}\|\hat{Q}^T\hat{Q}-I\|_2 &\le \frac{5}{64}64\kappa(\hat{Q}_0)^2\left(nm+m(m+1)\right)\mathbf{u}\\ &\le \frac{5445}{\left(25\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}}-3\right)^2}\left(nm+m(m+1)\right)\mathbf{u},\end{aligned}$$

and thus (74) follows.

Now, notice that by (21)–(24),

$$\sqrt{\frac{1+\gamma_n m}{1-\gamma_{m+1}m}} \le \sqrt{\frac{1+1.1nm\mathbf{u}}{1-1.1(m+1)m\mathbf{u}}} \le \sqrt{\frac{1.1}{0.9}} \le 1.11. \tag{76}$$

Observe that $\hat{R} = \hat{R}_1\hat{R}_0 + \Delta\hat{R}$ where $\hat{R}_1$ is the Cholesky factor of $\hat{Q}_0^T\hat{Q}_0$, where $\hat{Q}_0$ results from `randQR`, and $|\Delta\hat{R}| < \gamma_m|\hat{R}_1||\hat{R}_0|$ [16, Eq. (3.13)]. Then, it follows by [36, Eq. (3.16)], (24), (45), (48), (63), and (76), that

$$\begin{aligned}\|\Delta\hat{R}\|_2 &\leq \|\Delta\hat{R}\|_F \leq \gamma_m\|\hat{R}_1\|_F\|\hat{R}_0\|_F \leq m\gamma_m\|\hat{R}_1\|_2\|\hat{R}_0\|_2 \\ &\leq m\gamma_m\sqrt{\frac{1+\gamma_n m}{1-\gamma_{m+1}m}}\|\hat{Q}_0\|_2\|\hat{R}_0\|_2 \\ &\leq 1.23m^2\mathbf{u}\frac{6}{5\sqrt{1-\varepsilon_L}}(\sqrt{1+\varepsilon_H}\|V\|_2 + \frac{\sqrt{1-\varepsilon_L}}{12}\sigma_m(V)\delta) \\ &\leq m^2\mathbf{u}\frac{1.5}{\sqrt{1-\varepsilon_L}}(\sqrt{1+\varepsilon_H}\|V\|_2 + \frac{\sqrt{1-\varepsilon_L}}{12}\sigma_m(V)\delta)\ . \end{aligned} \tag{77}$$

Next, observe that by (54) we have that $\hat{Q}_0\hat{R}_0 = V + \Delta\tilde{V}$ from `randQR`. Using this and [36, Eq. (3.24)] to bound $\|\hat{Q}_0 - \hat{Q}\hat{R}_1\|_2$,

$$\begin{aligned}-\|\Delta\tilde{V}\|_2 + \|V - \hat{Q}\hat{R}\|_2 \leq \|V + \Delta\tilde{V} - \hat{Q}\hat{R}\|_2 &= \|\hat{Q}_0\hat{R}_0 - \hat{Q}\hat{R}_1\hat{R}_0 - \hat{Q}\Delta\hat{R}\|_2 \\ &\leq \|\hat{R}_0\|\|\hat{Q}_0 - \hat{Q}\hat{R}_1\|_2 + \|\hat{Q}\|_2\|\Delta\hat{R}\|_2 \\ &\leq 1.4\|\hat{R}_0\|_2\kappa(\hat{Q}_0)\|\hat{Q}_0\|_2 m^2\mathbf{u} + \|\hat{Q}\|_2\|\Delta\hat{R}\|_2. \end{aligned} \tag{78}$$

Note that (74) implies

$$\|\hat{Q}\|_2 \leq \sqrt{1 + \frac{5445}{\left(25\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 3\right)^2}\left(nm + m(m+1)\right)\mathbf{u}}\ .$$

Additionally, by (67) in Theorem 5.1,

$$\|\Delta\tilde{V}\|_2 = \|V - \hat{Q}_0\hat{R}_0\|_2 \leq \frac{\delta}{10}\sigma_m(V). \tag{79}$$

Now, starting from (45), we can use (48) to obtain

$$\begin{aligned}\|\hat{R}_0\|_2 &\leq \sqrt{1+\varepsilon_H}\|V\|_2 + \|E_1\|_2 + \|E_2\|_2 \\ &\leq \sqrt{1+\varepsilon_H}\|V\|_2 + \frac{\sqrt{1-\varepsilon_L}}{12}\sigma_m(V)\delta. \end{aligned} \tag{80}$$

Using (73) and (63) to bound $\kappa(\hat{Q}_0)$ and $\|\hat{Q}_0\|_2$, (77) to bound $\|\Delta\hat{R}\|_2$, (80) to bound $\|\hat{R}_0\|_2$, adding $\|\Delta\tilde{V}\|_2$ to both sides of (78) and then bounding $\|\Delta\tilde{V}\|_2$ using (79), we finally obtain

$$\begin{aligned}\|V - \hat{Q}\hat{R}\|_2 \leq &\left(\frac{56}{25\frac{1-\varepsilon_L}{\sqrt{1+\epsilon_H}} - 3\sqrt{1-\varepsilon_L}} + \frac{1.5}{\sqrt{1-\varepsilon_L}}\sqrt{1 + \frac{5445(nm+m(m+1))\mathbf{u}}{\left(25\sqrt{\frac{1-\varepsilon_L}{1+\varepsilon_H}} - 3\right)^2}}\right) \\ &\left(\sqrt{1+\varepsilon_H}\|V\|_2 + \frac{\sqrt{1-\varepsilon_L}}{12}\sigma_m(V)\delta\right)m^2\mathbf{u} + \frac{\delta}{10}\sigma_m(V),\end{aligned}$$

which does indeed satisfy $\|V - \hat{Q}\hat{R}\|_2 = O(\mathbf{u})\|V\|_2$, since $\sigma_m(V)\delta = O(\mathbf{u})\|V\|_2$.

Finally, observe that the probabilistic results used in this proof, namely (36)–(65) and Theorems 5.1–5.2, simultaneously hold with probability at least $1 - d$ (see Section 5.2.2 for details), and hence (74) and (75) hold with this probability as well. □

Theorem 5.2 guarantees `randQR`(V) produces a well-conditioned $\hat{Q}$. We show next that `rand_cholQR`(V) produces a factor $\hat{Q}$ with $\kappa(\hat{Q}) \approx 1$ (up to unit roundoff) for any numerically full rank V.

Theorem 5.4 (Conditioning of `rand_cholQR`)**.** *Suppose Assumptions 5.1 are satisfied. Then with probability at least $1 - d$, the matrix $\hat{Q}$ obtained with Algorithm 1 satisfies $\kappa(\hat{Q}) \approx 1$. More specifically,*

$$\kappa(\hat{Q}) < \sqrt{\frac{1 + \frac{5445}{\left(25\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 3\right)^2}\left(nm + m(m+1)\right)\boldsymbol{u}}{1 - \frac{5445}{\left(25\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 3\right)^2}\left(nm + m(m+1)\right)\boldsymbol{u}}}. \tag{81}$$

Furthermore, if $\frac{5445}{\left(25\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 3\right)^2}\left(nm + m(m+1)\right)\boldsymbol{u} < \frac{1}{2}$*, then*

$$\kappa(\hat{Q}) < 1 + \frac{10890}{\left(25\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 3\right)^2}\left(nm + m(m+1)\right)\boldsymbol{u}. \tag{82}$$

Proof. It follows from (74) that the i^{th} eigenvalue of $\hat{Q}^T Q$ satisfies

$$\lambda_i(\hat{Q}^T\hat{Q}) \geq 1 - \frac{5445}{\left(25\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 3\right)^2}\left(nm + m(m+1)\right)\mathbf{u},$$
$$\lambda_i(\hat{Q}^T\hat{Q}) \leq 1 + \frac{5445}{\left(25\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 3\right)^2}\left(nm + m(m+1)\right)\mathbf{u}.$$

Thus, the i^{th} singular value of $\hat{Q}$ satisfies

$$\sigma_i(\hat{Q}) \geq \sqrt{1 - \frac{5445}{\left(25\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 3\right)^2}\left(nm + m(m+1)\right)\mathbf{u}},$$
$$\sigma_i(\hat{Q}) \leq \sqrt{1 + \frac{5445}{\left(25\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 3\right)^2}\left(nm + m(m+1)\right)\mathbf{u}},$$

which gives (81). Further, for any $x < \frac{1}{2}$, $\sqrt{\frac{1+x}{1-x}} < 1 + 2x$, which gives (82). Since (74) holds with probability at least $1 - d$, (81) and (82) hold with this probability as well. □

Theorems 5.1–5.4 correspond to multisketchings, that is, to the application of one sketch matrix after another. In the rest of the section, we recast our error bounds for a single sketch matrix in Corollaries 5.1–5.4. The results apply for a single (ε, d, m) oblivious ℓ_2-subspace embedding for any $\varepsilon \in [0, \frac{616}{634})$, covering nearly the entire range of possible $\varepsilon \in [0, 1)$ for such embeddings.

We prove all the Corollaries simultaneously, as they are direct consequences of Theorems 5.1–5.4 by exploiting the fact that a single sketch can be recast as a product of two sketches, one of which is the identity, which is by definition a $(0, 0, m)$ oblivious ℓ_2-subspace embedding.

Assumptions 5.2. *Suppose $\varepsilon \in [0, \frac{616}{634})$ and $S \in \mathbb{R}^{p\times m}$ is a (ε, d, m) oblivious ℓ_2-subspace embedding. Further, suppose $V \in \mathbb{R}^{n\times m}$ has full rank and $1 < m \le s \le n$ where $nm\boldsymbol{u} \le \frac{1}{12}$, $p^{3/2}\boldsymbol{u} \le \frac{1}{12}$, and*

$$\delta = \frac{383\left(pm^{3/2} + \sqrt{m}(p^{3/2}\sqrt{1+\varepsilon} + n\|S\|_F)\right)}{\sqrt{1-\varepsilon}}\boldsymbol{u}\,\kappa(V) \le 1. \tag{83}$$

Corollary 5.1 (`randQR` Errors)**.** *Suppose Assumptions 5.2 are satisfied. Then the $\hat{Q}, \hat{R}$ factors obtained with Algorithm 1 (`randQR`) satisfy*

$$\|V - \hat{Q}\hat{R}\|_2 \le \frac{\delta}{10}\sigma_m(V),$$

and

$$\|(S\hat{Q})^T(S\hat{Q}) - I\|_2 \le \frac{2\delta}{11} + \left(\frac{\delta}{11}\right)^2 + \frac{24}{11}\frac{6\|S\|_2}{55\sqrt{1-\epsilon}}\delta + \left(\frac{6\|S\|_2}{55\sqrt{1-\epsilon}}\delta\right)^2$$

with probability at least $1 - d$, where δ is defined as in (83). Furthermore,

$$\|(S\hat{Q})^T(S\hat{Q}) - I\|_2 \le 3\delta$$

with probability at least $(1 - d)^2$.

Corollary 5.2 (Conditioning of `randQR`)**.** *Suppose Assumptions 5.2 are satisfied. Then with probability at least $1-d$, the $\hat{Q}$ matrix obtained with Algorithm 1 (`randQR`) has condition number $\kappa(\hat{Q}) = O(1)$. In fact,*

$$\kappa(\hat{Q}) \le \frac{33}{25\sqrt{\frac{1-\epsilon}{1+\epsilon}} - 3}.$$

Therefore, if $\varepsilon \le 0.9$,

$$\kappa(\hat{Q}) \le 12.07.$$

Corollary 5.3 (`rand_cholQR` Errors)**.** *Suppose Assumptions 5.2 are satisfied. Then with probability at least* $1-d$*, the* $\hat{Q}, \hat{R}$ *factors obtained with Algorithm 3 (`rand_cholQR`) has* $O(\boldsymbol{u})$ *orthogonality error and* $O(\boldsymbol{u})\|V\|_2$ *factorization error. In fact,*

$$\|\hat{Q}^T\hat{Q} - I\|_2 \leq \frac{5445}{\left(25\sqrt{\frac{1-\epsilon}{1+\epsilon}} - 3\right)^2}\left(nm + m(m+1)\right)\boldsymbol{u},$$

$$\|V - \hat{Q}\hat{R}\|_2 \leq \left(\frac{56}{25\frac{1-\varepsilon}{\sqrt{1+\epsilon}} - 3\sqrt{1-\varepsilon}} + \frac{1.5}{\sqrt{1-\varepsilon}}\sqrt{1 + \frac{5445(nm+m(m+1))\boldsymbol{u}}{\left(25\sqrt{\frac{1-\varepsilon}{1+\varepsilon}} - 3\right)^2}}\right)$$
$$\left(\sqrt{1+\varepsilon}\|V\|_2 + \frac{\sqrt{1-\varepsilon}}{12}\sigma_m(V)\delta\right)m^2\boldsymbol{u} + \frac{\delta}{10}\sigma_m(V),$$

where δ *is bounded as in* (83)*.*

Corollary 5.4 (Conditioning of `rand_cholQR`)**.** *Suppose Assumptions 5.2 are satisfied. Then with probability at least* $1-d$*, the matrix* $\hat{Q}$ *obtained with Algorithm 1 satisfies* $\kappa(\hat{Q}) \approx 1$*. In fact,*

$$\kappa(\hat{Q}) < \sqrt{\frac{1 + \frac{5445}{\left(25\sqrt{\frac{1-\epsilon}{1+\epsilon}}-3\right)^2}\left(nm + m(m+1)\right)\boldsymbol{u}}{1 - \frac{5445}{\left(25\sqrt{\frac{1-\epsilon}{1+\epsilon}}-3\right)^2}\left(nm + m(m+1)\right)\boldsymbol{u}}}.$$

Furthermore, if $\frac{5445}{\left(25\sqrt{\frac{1-\epsilon}{1+\epsilon}}-3\right)^2}\left(nm + m(m+1)\right)\boldsymbol{u} < \frac{1}{2}$*, then*

$$\kappa(\hat{Q}) < 1 + \frac{10890}{\left(25\sqrt{\frac{1-\epsilon}{1+\epsilon}} - 3\right)^2}\left(nm + m(m+1)\right)\boldsymbol{u}.$$

Proof. We prove Corollaries 5.1–5.4 simultaneously by considering one subspace embedding $S = S_1$ is equivalent to two subspace embeddings S_2S_1 simply by interpreting $S_2 = I_{p,p}$ as the $p \times p$ identity, which is by definition a $(0, 0, m)$ oblivious ℓ_2-subspace embedding, therefore giving $\varepsilon_1 = \varepsilon_H = \varepsilon_L = \varepsilon$, $d = d_1$, $p = p_2 = p_1$, $\varepsilon_2 = 0$ and $d_2 = 0$.

We show next that if $\varepsilon = \varepsilon_L = \varepsilon_H \in [0, \frac{616}{634})$, then $\varepsilon_L \in [0, \frac{616}{625} - \frac{9}{625}\varepsilon_H)$. Indeed, $\varepsilon_L \in [0, \frac{616}{625} - \frac{9}{625}\varepsilon_H)$ is equivalent in this case to $0 \leq \varepsilon < \frac{616}{625} - \frac{9}{625}\varepsilon$, or $0 \leq \frac{634}{625}\varepsilon < \frac{616}{625}$, or what is the same, $\varepsilon \in [0, \frac{616}{634})$. This means that when $\varepsilon_H = \varepsilon_L = \varepsilon \in [0, \frac{616}{634})$, Assumptions 5.1 imply Assumptions 5.2. Thus, Assumptions 5.1 are satisfied and Corollaries 5.1–5.4 are direct consequences of Theorems 5.1– 5.4. □

Remark 5.2. Observe that (83) in Assumptions 5.2 is identical to (19) in Assumptions 5.1 using $S_1 = S$ and $S_2 = I_{p,p}$, where $p_1 = p_2 = p$. However, the analysis in Section 5.2.3 of $\|E_1\|_2$ takes into account roundoff errors for

two matrix multiplications for two sketches to compute $\hat{W}$, while in the case of Corollaries 5.1–5.4, only one sketch and therefore one matrix multiplication to compute $\hat{W}$ is necessary. Therefore, we are over-estimating the error $\|E_1\|_2$ in the single sketch case, and if the analysis in Sections 5.2 were carefully performed again, we could tighten the bound on δ in (83), thereby loosening the requirements on $\kappa(V)$ in Assumptions 5.2. However, asymptotically, the requirement on $\kappa(V)$ would ultimately still be that $\delta \leq g(n, m, p_1, p_2)\mathbf{u}\,\kappa(V) \leq 1$ for some low-degree polynomial g.

6 Numerical Experiments

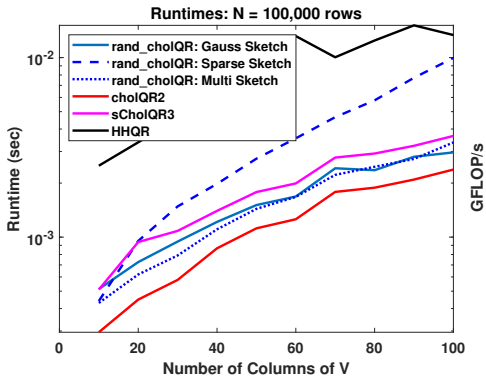
We conducted numerical experiments with two goals in mind. First, we compare the performance of `rand_cholQR` with the performance of `cholQR2`, `sCholQR3`, and Householder QR on a high-performance GPU leveraging vendor-optimized libraries. Second, we empirically validate the bounds given in Section 5.3, and more generally, compare the stability of `rand_cholQR` to the stability of `cholQR2`, `sCholQR3`, and Householder QR.

6.1 Implementation Details

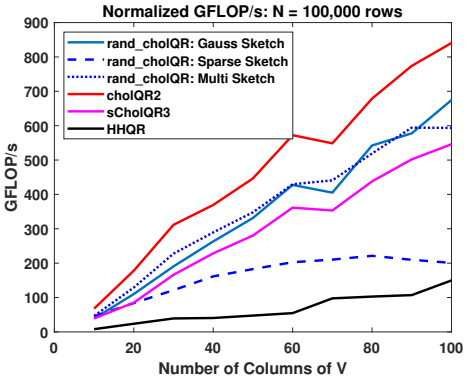
We implemented `rand_cholQR`, `cholQR2`, `sCholQR3`, and Householder QR in C++. To be portable to a GPU, we used the Kokkos Performance Portability Library [9] and Kokkos Kernels [26]. For our experiments on an NVIDIA GPU, we configured and built our code such that Kokkos Kernels calls NVIDIA's cuBLAS and cuSPARSE linear algebra libraries for optimized dense and sparse basic linear algebra routines [21, 24]. To perform LAPACK routines that are not currently available natively within Kokkos Kernels (i.e., `dgeqrf` and `dorgqr` for computing the Householder QR factorization, and `dpotrf` for the Cholesky factorization), we directly called NVIDIA's cuSOLVER linear algebra library [2, 22, 23]. Test results were obtained using Kokkos 3.7.01, Cuda 11.7.99, and GCC 7.2.0 on an AMD EPYC 7742 64-Core 2.25GHz CPU with an NVIDIA A100-SXM4 40GB GPU. All computations were done in double precision, so $\mathbf{u} = 2^{-52} \approx 10^{-16}$.

6.2 Performance Results

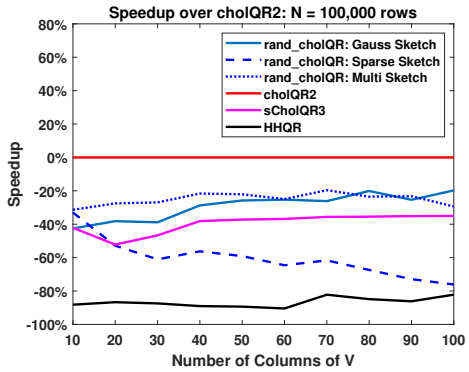
We tested `rand_cholQR` with a variety of sketching strategies. The simplest was the case of a Gaussian sketch $S = \frac{1}{\sqrt{p}}G \in \mathbb{R}^{p\times n}$, which was chosen to embed $V \in \mathbb{R}^{n\times m}$ with distortion $\epsilon = 0.99$, using a sketch size of $p = \lceil 36.01 \log(m) \rceil$ to produce a $(0.99, 1/m, m)$ oblivious ℓ_2-subspace embedding [1, Lemma 4.1]. We tested the CountSketch by explicit construction of the sparse matrix and applied it using a sparse-matrix vector product. The sketch size used with a $S \in \mathbb{R}^{p\times n}$ CountSketch matrix was $p = \lceil 6.8(m^2 + m) \rceil$, which can be shown to be a $(0.99, 0.15, m)$ oblivious ℓ_2-subspace embedding [19, Theorem 1].



(a) Raw runtimes



(b) Normalized GFLOP/s

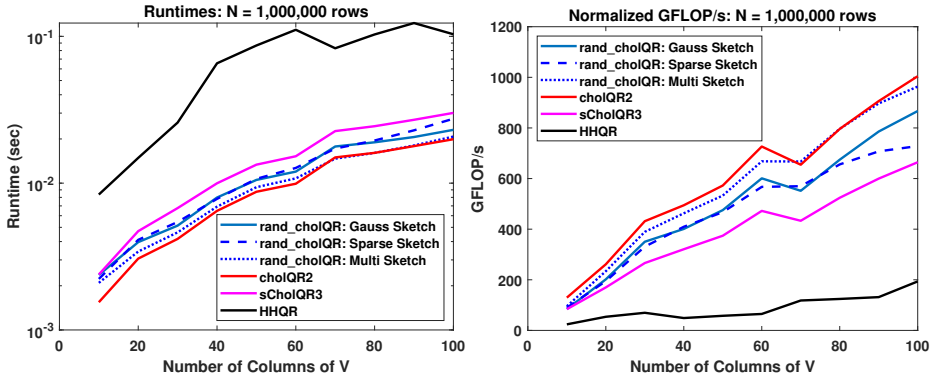


(c) Speedup over `cholQR2`

Figure 1: Performance of QR algorithms on matrices with $n = 100,000$ rows.

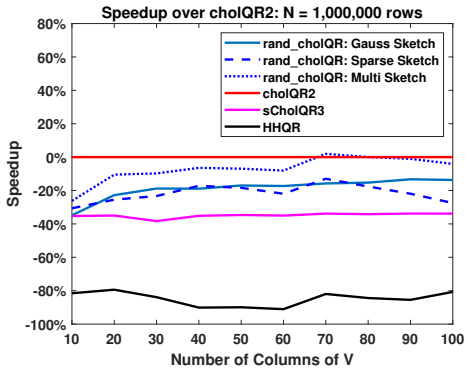
In our implementation of multisketching, we chose $S_1 \in \mathbb{R}^{p_1 \times n}$ as a CountSketch with $\varepsilon_1 = 0.9$ requiring sketch size $p_1 = \lceil 8.24(m^2 + m) \rceil$ to produce a $(0.9, 0.15, m)$ oblivious ℓ_2-subspace embedding, and $S_2 \in \mathbb{R}^{p_2 \times p_1}$ a Gaussian sketch with $p_2 = \lceil 74.3 \log(p_1) \rceil$ giving $\varepsilon_2 = 0.49$ to give a $(0.49, 1/m, m)$ oblivious ℓ_2-subspace embedding. Overall, $S_2 S_1$ produced an embedding with $\varepsilon_L \approx 0.9490$, $\varepsilon_H \approx 1.8310$, and $d \approx 0.15$. It is easily verified that $S_2 S_1$ is in line with Assumptions 5.1, and that both of S_1 and S_2 satisfy Assumptions 5.2, ensuring the analysis in Section 5.3 is relevant to the experiments. Runtimes of `rand_cholQR` did not include the time to generate the sketch, as this was assumed to be a fixed overhead time.

Figures 1–3 show the performance of each QR method for test problems with $n = 10^5 - 10^7$ rows and $m = 10$–100 columns. Within each figure, subfigure (a) shows the runtimes of each algorithm, and subfigure (b) shows the "normalized



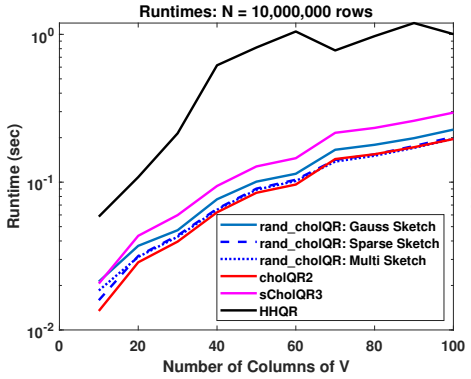
(a) Raw runtimes

(b) Normalized GFLOP/s

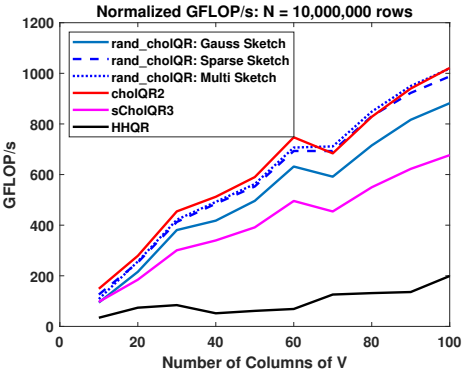


(c) Speedup over `cholQR2`

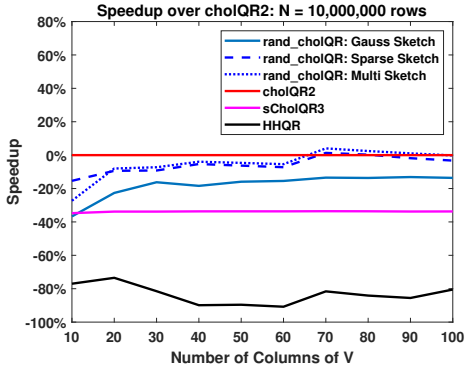
Figure 2: Performance of QR algorithms on matrices with $n = 1,000,000$ rows.



(a) Raw runtimes



(b) Normalized GFLOP/s



(c) Speedup over `cholQR2`

Figure 3: Performance of QR algorithms on matrices with $n = 10,000,000$ rows.

GFLOP/s" of each algorithm, which measured by

$$\text{Householder QR FLOPs/runtime} \times 10^{-9}.$$

The purpose of the normalization is to more fairly reflect the performance of each algorithm, as algorithms like `cholQR2` and `rand_cholQR` do more arithmetic operations than Householder QR, while they accomplish the same task, thereby skewing true GFLOP/s their favor. Since `cholQR2` is typically expected to be the fastest algorithm, subfigure (c) shows the relative speedup of each QR method compared to `cholQR2`.

Figures 1–3 indicate that `cholQR2` is indeed the fastest method in general, while multisketch `rand_cholQR` performs the closest to `cholQR2`. Additionally, Figures 2 (c) and 3 (c) show that in some cases, multisketch `rand_cholQR` actually outperforms `cholQR2`, specifically for $n = 10^6$ rows and $m = 70$ columns, and for $n = 10^7$ rows and $m = 70$–90 columns. The most notable result is that for $n = 10^7$ rows and $m = 70$ columns, multisketch `rand_cholQR` is 4% faster than `cholQR2`. Multisketch `rand_cholQR` is significantly faster than `sCholQR3`, as evidenced by Figures 1–3, and both algorithms have the same $O(\mathbf{u})\kappa(V) < 1$ stability requirement.

Subfigure (b) of Figures 1–3 demonstrate that the implementations and algorithms are indeed high-performance, with `cholQR2` and the multisketched `rand_cholQR` achieving up to 1,000 (normalized) GFLOP/s on a single GPU. It is important to note that for these algorithms, the normalized GFLOP/s are actually substantially lower than their true GFLOP/s, as both algorithms perform significantly more arithmetic operations than HouseholderQR.

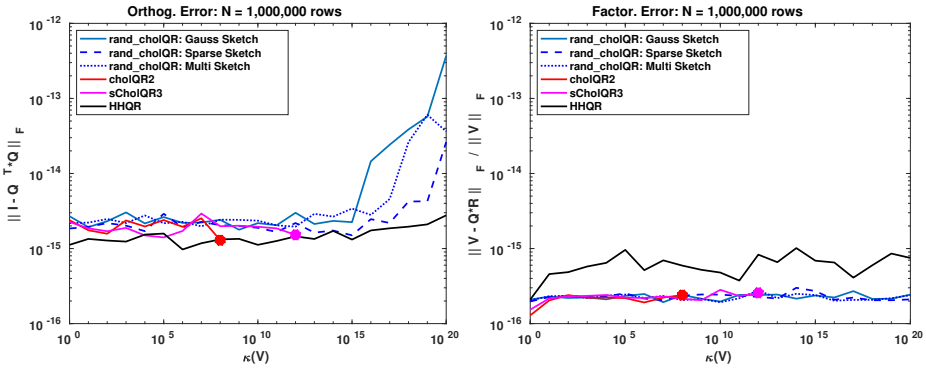
6.3 Numerical Results

Figure 4 shows the orthogonalization error $\|I - \hat{Q}^T\hat{Q}\|_F$ and the relative factorization error $\|V - \hat{Q}\hat{R}\|_F/\|V\|_F$ for condition number $\kappa(V) \in [1, 10^{16}]$. The results demonstrate that `rand_cholQR` maintains $O(\mathbf{u})$ orthogonality error and $O(\mathbf{u})\|V\|_2$ factorization error[4] while $\kappa(V) < O(\mathbf{u}^{-1})$, as predicted by Theorem 5.3, and is more robust than `cholQR2` and `sCholQR3`. In practice, it appears that `rand_cholQR` is stable even when V is numerically rank-deficient. In summary, Figures 1–4 demonstrate that multisketch `rand_cholQR` significantly improves the robustness of `cholQR2` and `sCholQR3` at little to no cost, therefore making `rand_cholQR` a superior high-performance QR algorithm. Additionally, observe that, as indicated by a large dot, lines for `cholQR2` and `sCholQR3` end at $\kappa(V) = 10^8$ and $\kappa(V) = 10^{12}$ respectively, as the methods fail beyond these points.

6.4 Effect of Larger Distortion Factor ϵ in `rand_cholQR`

In Section 2, we claimed that the stability guarantees for $\epsilon \approx 1$ is an improvement over existing results only providing stability guarantees up to $\epsilon \approx 0.5$

[4]This follows because $\|V - \hat{Q}\hat{R}\|_F/\|V\|_F = O(\mathbf{u})$.



(a) Orthogonality Error (b) Relative Factorization Error

Figure 4: Orthogonality (left) and relative factorization error (right) of the QR factorization of a matrix V with varying condition number. To explicitly control $\kappa(V)$, $V := L\Sigma R^T \in \mathbb{R}^{n\times m}$ using random orthogonal matrices L, R, and a diagonal Σ with log-equispaced entries in the range $[\kappa^{-\frac{1}{2}}(V), \kappa^{\frac{1}{2}}(V)]$. Indicated by a large dot, lines for `cholQR2` and `sCholQR3` end at $\kappa(V) = 10^8$ and $\kappa(V) = 10^{12}$ respectively, as the methods fail beyond these points.

as the larger values of ϵ allow for better performance. In this subsection, we include experiments that justify this and highlight why theoretical results for larger values of ϵ should be studied further in the context of randomized QR factorizations.

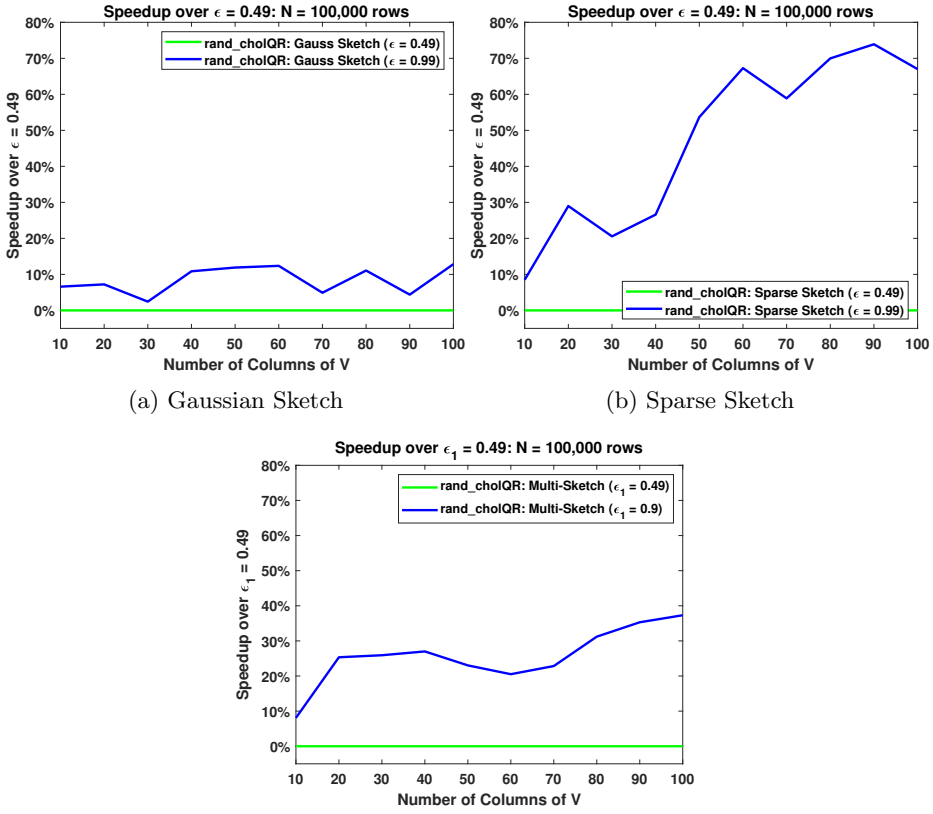
In Figures 5–7, we begin by demonstrating the speedup attained by using a single sketch with $\epsilon = 0.99$ over a sketch with $\epsilon = 0.49$ where subfigures (a) and (b) correspond to the Gaussian sketch and CountSketch experiments, respectively. Thereafter, we demonstrate the speedup in attained in the multisketch case using a larger $\epsilon_1 = 0.9$ compared to $\epsilon_1 = 0.49$ in subfigure (c). In the multisketch case, we keep $\epsilon_2 = 0.49$ fixed, and did not push ϵ_1 over 0.9 to ensure Assumptions 5.1 and therefore the theoretical results in Section 5.3 hold.

Specifically, in the case of the Gaussian experiments (Figures 5–7 (a)), we used $p = \lceil 36.01 \log(m) \rceil$ to attain distortion factor $\epsilon = 0.99$ and $p = \lceil 74.3 \log(m) \rceil$ to attain distortion factor $\epsilon = 0.49$[5]. In the case of the CountSketch (Figures 5–7 (b)), we used sketch size $p = \lceil 6.8(m^2+m) \rceil$ to attain $\epsilon = 0.99$ and sketch size $p = \lceil 27.8(m^2+m) \rceil$ to attain $\epsilon = 0.49$ [6]. Finally, in the multisketch case (Figures 5–7 (c)), the first sketch was a CountSketch with embedding dimension $p_1 = \lceil 8.24(m^2+m) \rceil$ to attain $\varepsilon_1 = 0.9$ and embedding dimension $p_1 = \lceil 27.8(m^2+m) \rceil$ to attain $\epsilon = 0.49$[7].

[5]These sketch sizes can be shown to produce $(0.99, 1/m, m)$ and $(0.49, 1/m, m)$ oblivious ℓ_2-subspace embeddings, respectively

[6]This can be shown to attain $(0.99, 0.15, m)$ and $(0.49, 0.15, m)$ oblivious ℓ_2-subspace embeddings, respectively.

[7]These choices can be shown to produce $(0.9, 0.15, m)$ and $(0.49, 0.15, m)$ oblivious ℓ_2-subspace embeddings, respectively

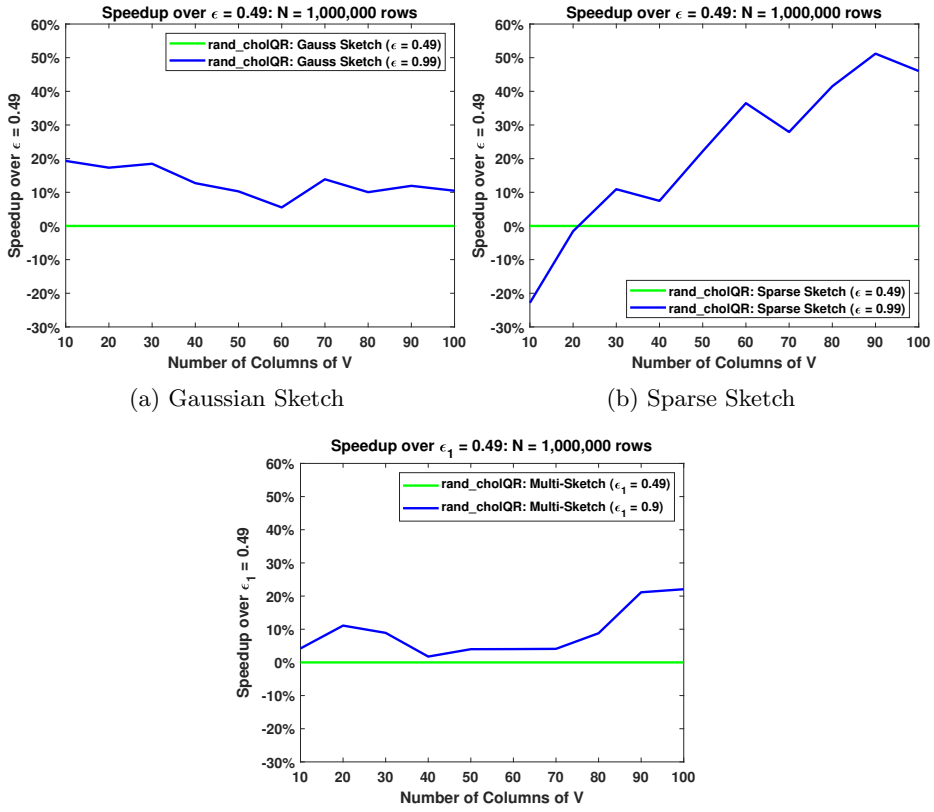


(a) Gaussian Sketch

(b) Sparse Sketch

(c) Multi-Sketch

Figure 5: Speedup attained in `rand_cholQR` using different values of ϵ, $n = 100,000$.

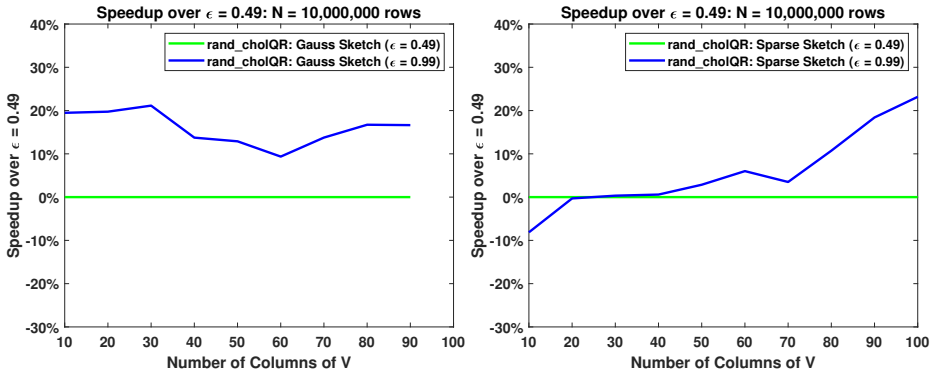


(a) Gaussian Sketch

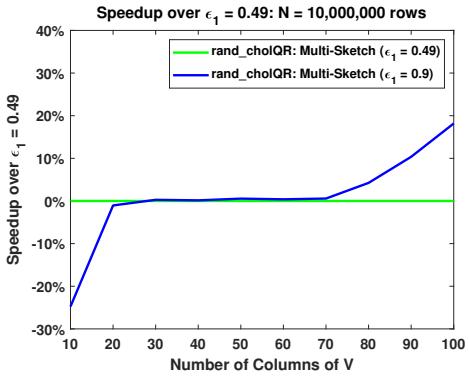
(b) Sparse Sketch

(c) Multi-Sketch

Figure 6: Speedup attained in `rand_cholQR` using different values of ϵ, $n = 1,000,000$.



(a) Gaussian Sketch (b) Sparse Sketch



(c) Multi-Sketch

Figure 7: Speedup attained in `rand_cholQR` using different values of ϵ, $n = 10,000,000$.

As predicted, the results indicate that the in most cases, the impact of choosing a larger ϵ allows for significant runtime improvements in the randomized QR routine tested. Presumably, these results would translate to other randomized methods where a large distortion is acceptable, such as in preprocessing/preconditioning procedures or inexact orthogonalization schemes, where only linear independence is strictly necessary.

7 Conclusions and Future Work

The results in Section 5.3 indicate that `rand_cholQR` using one or two sketch matrices orthogonalizes any numerically full-rank matrix V up to $O(\mathbf{u})$ error. This is a significant improvement over CholeskyQR2, which requires $\kappa(V) \lessapprox \mathbf{u}^{-1/2}$ to ensure a stable factorization. Our results for a single sketch apply for any ε-embedding with $\varepsilon \in [0, \frac{616}{634})$, covering nearly the entire possible range for ε-embeddings.

Our performance results in Section 6 indicate that the significantly better stability properties of `rand_cholQR` over `cholQR2` come at virtually no increase in the factorization time on a modern GPU. Additionally, `rand_cholQR` is theoretically just as stable and in practice more stable than `sCholQR3`, while being substantially faster. This is due to the fact that `rand_cholQR` and `cholQR2` incur the same number of processor synchronizations, while leveraging mostly BLAS-3 or optimized sparse matrix-vector routines for most of the required computation. In fact, `rand_cholQR` can perform better than `cholQR2` when using the multisketch framework. Of the sketching strategies considered, the multisketch framework is the most advantageous, likely because it requires little additional storage compared to `cholQR2`, and applying the sketches in this framework is extremely cheap.

Future work includes applying `rand_cholQR` to Krylov subspace methods that require tall-and-skinny QR factorizations, particularly block [15, 25], s-step [7, 17, 32], and enlarged Krylov methods [14], and further investigations into efficient multisketching implementations on a GPU, as our analysis is amenable to any multisketching strategy (not just a CountSketch followed by a dense Gaussian). In particular, applying the CountSketch matrix could potentially be optimized better than using a sparse-matrix vector multiplication by using a custom routine to add/subtract subsets of randomly selected rows in parallel using batched BLAS-1 routines, which should be investigated. Additionally, the performance of `randQR` and `rand_cholQR` using dense Rademacher sketch matrices in place of dense Gaussian sketches as in [1] should be investigated, as Rademacher sketches impose far lower storage requirements than a Gaussian sketch and can be generated much more efficiently.

Acknowledgements

The authors appreciate very much the comments and questions by the handling editor Ilse Ipsen, and by the three anonymous referees, which helped improved our presentation.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. This work was in part supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

Statements and Declarations

Competing interests. The authors declare no competing interests.

References

[1] Dimitris Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66:671–687, 2003. Special Issue on PODS 2001.

[2] Eric Anderson, Zhaojun Bai, Christopher Bischof, S. Blackford, James W. Demmel, Jack J. Dongarra, Jeremy Du Croz, Aanne Greenbaum, Sven Hammarling, A. McKenney, and Dannny C. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, Third edition, 1999.

[3] Oleg Balabanov. Randomized Cholesky QR factorizations, 2022. arXiv:2210.09953.

[4] Oleg Balabanov and Laura Grigori. Randomized Gram–Schmidt process with application to GMRES. *SIAM Journal on Scientific Computing*, 44:A1450–A1474, 2022.

[5] Oleg Balabanov and Laura Grigori. Randomized block Gram–Schmidt process for solution of linear systems and eigenvalue problems, 2023. arXiv:2111.14641.

[6] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In P. Widmayer, S. Eidenbenz, F. Triguero, R. Morales, R. Conejo, and M. Hennessy, editors, *Automata, Languages and Programming*, volume 2380 of *Lecture Notes in Computer Science*, pages 693–703, Berlin, Heidelberg, 2002. Springer. ICALP 2002.

[7] Anthony T. Chronopoulos and C. William Gear. s-step iterative methods for symmetric linear systems. *Journal of Computational and Applied Mathematics*, 25:153–168, 1989.

[8] James W. Demmel, Laura Grigori, Maek Hoemmen, and Julien Langou. Communication-optimal parallel and sequential QR and LU factorizations. *SIAM Journal on Scientific Computing*, 34:A206–A239, 2012.

[9] H. Carter Edwards, Christian R. Trott, and Daniel Sunderland. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. *Journal of Parallel and Distributed Computing*, 74:3202–3216, 2014.

[10] Yuwei Fan, Yixiao Guo, and Ting Lin. A novel randomized XR-based preconditioned CholeskyQR algorithm, 2021. arXiv:2111.11148.

[11] George E. Forsythe and Cleve B. Moler. *Computer Solution of Linear Algebraic Systems*. Prentice-Hall, Englewood Cliffs, N.J., 1967.

[12] Takeshi Fukaya, Ramaseshan Kannan, Yuji Nakatsukasa, Yusaku Yamamoto, and Yuka Yanagisawa. Shifted Cholesky QR for computing the QR factorization of ill-conditioned matrices. *SIAM Journal on Scientific Computing*, 42(1):A477–A503, 2020.

[13] Takeshi Fukaya, Yuji Nakatsukasa, Yuka Yanagisawa, and Yusaku Yamamoto. CholeskyQR2: A simple and communication-avoiding algorithm for computing a tall-skinny QR factorization on a large-scale parallel system. In *2014 5th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, pages 31–38, Los Alamitos, CA, 2014. IEEE Computer Society.

[14] Laura Grigori, Sophie Moufawad, and Frederic Nataf. Enlarged Krylov subspace conjugate gradient methods for reducing communication. *SIAM Journal on Matrix Analysis and Applications*, 37:744–773, 2016.

[15] Martin H. Gutknecht. Block Krylov subspace methods for linear systems with multiple right-hand sides: An introduction. In Abul Hasan Siddiqi, Iain S. Duff, and Ole Christensen, editors, *Modern Mathematical Models, Methods and Algorithms for Real World Systems*, chapter 10, pages 420–447. Anamaya Publishers, New Dehli, 2006.

[16] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, second edition, 2002.

[17] Mark Hoemmen. *Communication-avoiding Krylov subspace methods*. PhD thesis, EECS Department, University of California, Berkeley, 2010.

[18] Michael Kapralov, Vamsi Potluru, and David Woodruff. How to fake multiply by a Gaussian matrix. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 2101–2110. Proceedings of Machine Learning Research, 2016.

[19] Xiangrui Meng and Michael W. Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, page 91–100, New York, 2013. Association for Computing Machinery.

[20] Yuji Nakatsukasa and Joel A. Tropp. Fast & accurate randomized algorithms for linear systems and eigenvalue problems, 2021. arXiv:2111.00113.

[21] NVIDIA. cuBLAS documentation. `https://docs.nvidia.com/cuda/cublas/index.html`. Accessed: 2023-06-21.

[22] NVIDIA. CUDA toolkit documentation. `https://docs.nvidia.com/cuda/`. Accessed: 2023-06-21.

[23] NVIDIA. cuSOLVER documentation. `https://docs.nvidia.com/cuda/cusolver/index.html`. Accessed: 2023-06-21.

[24] NVIDIA. cuSPARSE documentation. `https://docs.nvidia.com/cuda/cusparse/index.html`. Accessed: 2023-06-21.

[25] Dianne P. O'Leary. The block conjugate gradient algorithm and related methods. *Linear Algebra and its Applications*, 29:293–322, 1980.

[26] Sivasankaran Rajamanickam, Seher Acer, Luc Berger-Vergiat, Vinh. Q. Dang, Nathan D. Ellingwood, Evan Harvey, Brian Kelley, Christian R. Trott, Jeremy J. Wilke, and Ichitaro Yamazaki. Kokkos kernels: Performance portable sparse/dense linear algebra and graph kernels, 2021. arxiv:2103.11991.

[27] Vladimir Rokhlin and Mark Tygert. A fast randomized algorithm for overdetermined linear least-squares regression. *Proceedings of the National Academy of Sciences of the United States of America*, 105:13212–13217, 2008.

[28] Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 143–152, Los Alamitos, CA, 2006. IEEE Computer Society.

[29] Aleksandros Sobczyk and Efstratios Gallopoulos. Estimating leverage scores via rank revealing methods and randomization. *SIAM Journal on Matrix Analysis and Applications*, 42:199–1228, 2021.

[30] Aleksandros Sobczyk and Efstratios Gallopoulos. pylspack: Parallel algorithms and data structures for sketching, column subset selection, regression, and leverage scores. *ACM Transactions on Mathematical Software*, 48:1–27, 2022.

[31] Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2018.

[32] Homer F. Walker. Implementation of the GMRES method using Householder transformations. *SIAM Journal on Scientific and Statistical Computing*, 9:152–163, 1988.

[33] Hermann Weyl. Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differentialgleichungen (mit einer Anwendung auf die Theorie der Hohlraumstrahlung). *Mathematische Annalen*, 71:441–479, 1912.

[34] James H. Wilkinson. *The Algebraic Eigenvalue Problem*. Oxford University Press, Oxford, UK, 1965.

[35] David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10:1–157, 2014.

[36] Yusaku Yamamoto, Yuji Nakatsukasa, Yuka Yanagisawa, and Takeshi Fukaya. Roundoff error analysis of the Cholesky QR2 algorithm. *Electronic Transactions on Numerical Analysis*, 44:306–326, 2015.

A Raw Runtimes From Experiments

Algorithm	Columns 10	20	30	40	50	60	70	80	90	100
`rand_cholQR`: Gauss ($\epsilon = 0.99$)	0.0005	0.0007	0.0009	0.0012	0.0015	0.0017	0.0024	0.0024	0.0028	0.0030
`rand_cholQR`: Gauss ($\epsilon = 0.49$)	0.0005	0.0008	0.0010	0.0014	0.0017	0.0019	0.0025	0.0027	0.0029	0.0034
`rand_cholQR`: Sparse ($\epsilon = 0.99$)	0.0004	0.0010	0.0015	0.0020	0.0027	0.0036	0.0047	0.0058	0.0077	0.0100
`rand_cholQR`: Sparse ($\epsilon = 0.49$)	0.0005	0.0013	0.0019	0.0027	0.0059	0.0108	0.0113	0.0192	0.0296	0.0302
`rand_cholQR`: Multi ($\epsilon_1 = 0.9, \epsilon_2 = 0.49$)	0.0004	0.0006	0.0008	0.0011	0.0014	0.0017	0.0022	0.0025	0.0027	0.0034
`rand_cholQR`: Multi ($\epsilon_1 = 0.49, \epsilon_2 = 0.49$)	0.0005	0.0008	0.0011	0.0015	0.0019	0.0021	0.0029	0.0036	0.0042	0.0054
`cholQR2`	0.0003	0.0004	0.0006	0.0009	0.0011	0.0013	0.0018	0.0019	0.0021	0.0024
`sCholQR3`	0.0005	0.0009	0.0011	0.0014	0.0018	0.0020	0.0028	0.0029	0.0032	0.0037
Householder	0.0005	0.0009	0.0011	0.0014	0.0018	0.0020	0.0028	0.0029	0.0032	0.0037

Table 1: Raw runtimes (in seconds) of all experiments with $n = 100,000$ rows.

Algorithm	Columns 10	20	30	40	50	60	70	80	90	100
`rand_cholQR`: Gauss ($\epsilon = 0.99$)	0.0024	0.0040	0.0051	0.0080	0.0105	0.0120	0.0178	0.0190	0.0206	0.0231
`rand_cholQR`: Gauss ($\epsilon = 0.49$)	0.0029	0.0048	0.0063	0.0091	0.0117	0.0127	0.0206	0.0211	0.0234	0.0258
`rand_cholQR`: Sparse ($\epsilon = 0.99$)	0.0022	0.0041	0.0054	0.0078	0.0107	0.0127	0.0172	0.0195	0.0229	0.0275
`rand_cholQR`: Sparse ($\epsilon = 0.49$)	0.0018	0.0040	0.0061	0.0084	0.0138	0.0200	0.0239	0.0334	0.0469	0.0509
`rand_cholQR`: Multi ($\epsilon_1 = 0.9, \epsilon_2 = 0.49$)	0.0021	0.0034	0.0046	0.0069	0.0094	0.0108	0.0147	0.0161	0.0181	0.0208
`rand_cholQR`: Multi ($\epsilon_1 = 0.49, \epsilon_2 = 0.49$)	0.0022	0.0038	0.0051	0.0070	0.0098	0.0112	0.0153	0.0176	0.0229	0.0266
`cholQR2`	0.0015	0.0031	0.0042	0.0065	0.0087	0.0099	0.0150	0.0161	0.0179	0.0199
`sCholQR3`	0.0024	0.0047	0.0068	0.0100	0.0134	0.0152	0.0226	0.0244	0.0270	0.0301
Householder	0.0024	0.0047	0.0068	0.0100	0.0134	0.0152	0.0226	0.0244	0.0270	0.0301

Table 2: Raw runtimes (in seconds) of all experiments with $n = 1,000,000$ rows.

Algorithm	Columns 10	20	30	40	50	60	70	80	90	100
`rand_cholQR`: Gauss ($\epsilon = 0.99$)	0.0212	0.0371	0.0473	0.0766	0.1008	0.1140	0.1657	0.1790	0.1984	0.2268
`rand_cholQR`: Gauss ($\epsilon = 0.49$)	0.0264	0.0462	0.0600	0.0888	0.1157	0.1258	0.1921	0.2150	0.2380	
`rand_cholQR`: Sparse ($\epsilon = 0.99$)	0.0159	0.0317	0.0436	0.0660	0.0904	0.1039	0.1414	0.1541	0.1756	0.2025
`rand_cholQR`: Sparse ($\epsilon = 0.49$)	0.0147	0.0316	0.0438	0.0664	0.0931	0.1105	0.1466	0.1726	0.2152	0.2635
`rand_cholQR`: Multi ($\epsilon_1 = 0.9, \epsilon_2 = 0.49$)	0.0185	0.0312	0.0427	0.0650	0.0889	0.1019	0.1378	0.1508	0.1706	0.1962
`rand_cholQR`: Multi ($\epsilon_1 = 0.49, \epsilon_2 = 0.49$)	0.0148	0.0309	0.0428	0.0651	0.0894	0.1023	0.1386	0.1575	0.1903	0.2398
`cholQR2`	0.0134	0.0287	0.0396	0.0625	0.0848	0.0964	0.1433	0.1545	0.1724	0.1958
`sCholQR3`	0.0206	0.0433	0.0598	0.0942	0.1278	0.1453	0.2158	0.2328	0.2603	0.2955
Householder	0.0206	0.0433	0.0598	0.0942	0.1278	0.1453	0.2158	0.2328	0.2603	0.2955

Table 3: Raw runtimes (in seconds) of all experiments with $n = 10,000,000$ rows. Blank entry indicates the experiment could not fit in GPU memory.