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

Department of Mathematics
Temple University
Philadelphia, PA 19122

This report is available in the World Wide Web at
`http://www.math.temple.edu/~szyld`

Analysis of Randomized Householder-Cholesky QR Factorization with Multisketching

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

September 11, 2023

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 evluation of the performance of `rand_cholQR` on a 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

*Temple University, Philadelphia, Pennsylvania, USA (`andrew.higgins@temple.edu`, `szyld@temple.edu`).

†Center for Computing Research, Sandia National Laboratories, Albuquerque, New Mexico, USA (`egboman@sandia.gov`, `iyamaza@sandia.gov`).

1 Introduction

Computing the QR factorization of tall-and-skinny matrices is a critical component of many scientific and engineering applications, including the solution 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. Modern high-performance computers enable extremely fast floating point operations (FLOPs), but are limited by relatively slow communication (data transfers) between processors and through the memory hierarchy. Current state-of-the-art QR algorithms for tall-and-skinny matrices are the CholeskyQR2 and shifted CholeskyQR3 algorithms [9, 10], thanks to their communication-avoiding properties along with their exploitation of vendor-provided highly-optimized dense linear algebra subroutines [2, 19, 20]. However, CholeskyQR2 may fail to accurately factorize a matrix V when its condition number $\kappa(V) > O(\mathbf{u}^{-1/2})$, where $\mathbf{u}$ is unit roundoff [31]. Shifted CholeskyQR3 is numerically stable as long as $\kappa(V) < O(\mathbf{u}^{-1})$, but it requires over 50% more computational and communication cost than CholeskyQR2 [9]. Although more stable communication-avoiding algorithms exist, such as TSQR [7], they rely on Householder QR factorizations, and are often significantly slower than CholeskyQR2 in practice [10].

In this paper, we present and analyze a randomized algorithm called `randQR` for orthogonalizing the columns of a tall-and-skinny matrix with respect to a specific bilinear form. In order to reduce the cost of the computations, we propose to use "multisketching," i.e., the use of two consecutive sketch matrices, obtaining another algorithm called `rand_cholQR` for computing the 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 2 for definitions), but what we have in mind is one sparse sketch and one 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 [15, 26, 27].

We prove that with high probability, the orthogonality error of `rand_cholQR` is bounded by a constant of the order of unit roundoff for any numerically full-rank matrix V, 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 heterogeneous computer 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 contribution consist of a new error analysis of multisketching, expanding the applicability of the method, as it can be safely used for matrices of larger condition number. This analysis applies in particular to the

case of one sketch, improving upon the existing results. Our implementation confirms the theory developed in this paper.

In the next section we present some preliminary definitions and known results. We follow with Section 3, where we give the multisketching algorithm, together with a result on the use of two sketch matrices. We also present FLOP counts for different algorithms, leading to the motivation as to why multisketching is recommended. In Section 4 we present rigorous error bounds for the proposed multisketching algorithm (the proofs are given in an Appendix). 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.

2 Preliminaries on Random Sketching

Suppose one would like to compress $V \in \mathbb{R}^{n,m}$ into a matrix with fewer columns with nearly the same norm. We denote the *sketch matrix* by $S \in \mathbb{R}^{s,n}$ for $s \ll n$. The sketch matrix is typically chosen to be a *ϵ-subspace embedding*, 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, 17, 25].

Definition 2.1 (ε-subspace embedding)**.** *Given $\varepsilon \in [0, 1)$, the sketch matrix $S \in \mathbb{R}^{s,n}$ is an* ε-subspace embedding *for the subspace $\mathcal{V} \subset \mathbb{R}^n$ if $\forall x, y \in \mathcal{V}$,*

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

Corollary 2.1. *If the sketch matrix $S \in \mathbb{R}^{s,n}$ is an* ε-subspace embedding *for the subspace $\mathcal{V} \subset \mathbb{R}^n$, then $\forall x \in \mathcal{V}$,*

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

Corollary 2.2. *If the sketch matrix $S \in \mathbb{R}^{s,n}$ 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*

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

Thus,

$$\kappa(V) \leq \sqrt{\frac{1-\varepsilon}{1+\varepsilon}}\ \kappa(SV). \tag{3}$$

As already indicated in [4], Corollary 2.2 implies that the singular values of V are bounded by those of SV. Hence 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 [4].

Definition 2.2 ((ε, d, m) oblivious ℓ_2-subspace embedding)**.** *$S \in \mathbb{R}^{s,n}$ 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 $S = \frac{1}{\sqrt{s}}G$ for a Gaussian matrix $G \in \mathbb{R}^{s,n}$ and

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

this is a full matrix; see, e.g., [25]. 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 [5, 30]. In order to be a (ε, d, m) oblivious ℓ_2-subspace embedding, the number of columns of the CountSketch matrix must satisfy

$$s \geq \frac{m^2 + m}{\varepsilon^2 d} \ ; \tag{4}$$

see [16]. Other popular (ε, d, m) oblivious ℓ_2-subspace embeddings include subsampled randomized Hadamard and Fourier transforms, and "sparse dimension reduction maps" [4, 17], 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.

3 Multisketching

Next, we consider the case of applying two sketch matrices one after the other; which is what we propose in this paper, generalizing the approach of [15, 27], where one application of a (sparse) CountSketch is followed by a 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 obtaining similar good results at a fraction of the cost; see more details on this motivation in Section 3.2.

We first present the algorithm `randQR` with two sketches, and then prove bounds similar to those in Corollary 2.2 for the case of two sketches.

Let $V \in \mathbb{R}^{n,m}$, and suppose $S_1 \in \mathbb{R}^{s_1,n}$ and $S_2 \in \mathbb{R}^{s_2,s_1}$ are (ε_1, d_1, m) and (ε_2, d_2, m) oblivious ℓ_2-subspace embeddings, respectively. We define the Randomized Householder QR algorithm (`randQR`) in Algorithm 1, where we have used a Matlab function call.

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

Input: Matrix $V \in \mathbb{R}^{n,m}$, sketch matrices $S_1 \in \mathbb{R}^{s_1,n}$, $S_2 \in \mathbb{R}^{s_2,s_1}$
Output: S_2S_1-Orthogonal factor $Q \in \mathbb{R}^{n,m}$, Triangular factor $R \in \mathbb{R}^{m,m}$ 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 3.1. In exact arithmetic, provided that $V \in \mathbb{R}^{n,m}$ is full rank, then `randQR` produces a matrix Q that is S_2S_1-orthogonal; i.e., it is orthogonal with respect to $(S_2S_1\cdot, S_2S_1\cdot)$, or $(S_2S_1Q)^T(S_2S_1Q) = I$, as $S_2S_1Q = S_2S_1VR^{-1} = WR^{-1} = Q_{tmp}$, where Q_{tmp} is the orthogonal factor produced by the Householder QR factorization of $W = S_2S_1V$. 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.

Proposition 3.1. *Let $S_1 \in \mathbb{R}^{s_1,n}$ be a (ε_1, d_1, m) oblivious ℓ_2-subspace embedding in $\mathbb{R}^n$, $S_2 \in \mathbb{R}^{s_2,s_1}$ be a (ε_2, d_2, m) oblivious ℓ_2-subspace embedding in $\mathbb{R}^{s_1}$, generated independently. Let $\varepsilon_L = \varepsilon_1 + \varepsilon_2 - \varepsilon_1\varepsilon_2$, $\varepsilon_H = \varepsilon_1 + \varepsilon_2 + \varepsilon_1\varepsilon_2$, and $d = d_1 + d_2 - d_1d_2$. Then for any m-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$ and $\forall x \in \mathcal{V}$,*

$$\sqrt{1-\varepsilon_L}\ \|x\|_2 \le \|S_2S_1x\|_2 \le \sqrt{1+\varepsilon_H}\ \|x\|_2, \tag{5}$$

with probability at least $1 - d$.

Proof. S_2 is a (ε_2, d_2, m) oblivious ℓ_2-subspace embedding of $S_1\mathcal{V} \in \mathbb{R}^{s_1}$, and that if $x \in \mathcal{V}$, then $S_1x \in S_1\mathcal{V}$. Therefore, for any $x \in \mathcal{V}$,

$$\sqrt{1-\varepsilon_2}\|S_1x\|_2 \le \|S_2S_1x\|_2 \le \sqrt{1+\varepsilon_2}\|S_1x\|_2.$$

Now, using (1), for S_1 and ϵ_1, we have

$$\begin{aligned}\sqrt{1-(\varepsilon_1+\varepsilon_2-\varepsilon_1\varepsilon_2)}\|x\|_2 &= \sqrt{(1-\varepsilon_2)(1-\varepsilon_1)}\|x\|_2 \le \sqrt{1-\varepsilon_2}\|S_1x\|_2 \\ &\le \|S_2S_1x\|_2 \le \sqrt{1+\varepsilon_2}\|S_1x\|_2 \\ &\le \sqrt{(1+\varepsilon_2)(1+\varepsilon_1)}\|x\|_2 = \sqrt{1+(\varepsilon_1+\varepsilon_2+\varepsilon_1\varepsilon_2)}\|x\|_2\end{aligned}$$

with probability at least $(1-d_1)(1-d_2) = 1-(d_1+d_2-d_1d_2)$. □

If S_1, S_2 are $\varepsilon_1, \varepsilon_2$ embeddings respectively, then by Corollary 2.2 and Proposition 3.1,

$$\begin{aligned}(1+\varepsilon_H)^{-1/2}\ \sigma_{min}(S_2S_1V) \le \sigma_{min}(V) \le \sigma_{max}(V) \\ \le (1-\varepsilon_L)^{-1/2}\ \sigma_{max}(S_2S_1V),\end{aligned} \tag{6}$$

and so,

$$\kappa(V) \le \sqrt{\frac{1-\varepsilon_L}{1+\varepsilon_H}}\ \kappa(S_2S_1V). \tag{7}$$

3.1 Algorithms and FLOP count

Let Q be the S_2S_1-orthogonal factor from `randQR`. By Corollary 2.2 and the fact that S_2S_1Q is orthogonal, $\kappa(Q) \le \sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}}\kappa(S_2S_1Q) = \sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} = O(1)$. Thus, if one re-orthogonalizes Q using Cholesky QR (`cholQR`, Algorithm 2), it can be shown that the result has a loss of orthogonality on the order of machine

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

Input: Matrix $V \in \mathbb{R}^{n,m}$

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

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

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

Input: Matrix $V \in \mathbb{R}^{n,m}$, sketch matrices $S_1 \in \mathbb{R}^{s_1,n}$, $S_2 \in \mathbb{R}^{s_2,s_1}$

Output: Orthogonal factor $Q \in \mathbb{R}^{n,m}$, Triangular factor $R \in \mathbb{R}^{m,m}$ 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: $R = R_1R_0$

precision [31]. We take this approach, resulting in Algorithm 3, which we call `rand_cholQR`.

The computational cost of Step 2 of `randQR` (Algorithm 1) is negligible compared to steps 1 and 3, since $W \in \mathbb{R}^{s_2,m}$ with $s_2 \ll n$. The cost of Step 1 is dependent on the type of sketch matrices used. Suppose one replaces S_2S_1 with a single dense Gaussian sketch matrix $S \in \mathbb{R}^{s,n}$, which is conceptually simple, very efficient in parallel, but computationally expensive since it is fully dense. Then the computational cost of `randQR` and `rand_cholQR` (in FLOPs) are:

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

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

Provided that $s = O(m)$, e.g., $s \approx 2m$, then `rand_cholQR` FLOPs $\approx 7nm^2$. What we propose instead is to perform `cholQR` twice, and this is rendered in Algorithm 4, which we call `CholQR2`, where in Step 3, we use the dense matrix-matrix product subroutine `gemm`. `CholQR2` incurs a cost of

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

Comparing `randQR` (using a dense Gaussian sketch) to `cholQR`, their computational costs about the same asymptotically (as we discuss in the next section, the size of s_1 and s_2 depend of m, but are independent of n). In addition, the dominant costs incurred in steps 1 and 3 of `randQR` are nearly identical to `cholQR`, in the sense that both perform a product of tall and skinny matrices,

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

Input: Full rank matrix $V \in \mathbb{R}^{n,m}$

Output: Orthogonal factor $Q \in \mathbb{R}^{n,m}$, Triangular factor $R \in \mathbb{R}^{m,m}$

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: $R = R_1 R_0$ // `gemm`

followed by a triangular solve of a tall and skinny matrix, and therefore the algorithms should incur the same number of processor synchronizations. Moreover, `rand_cholQR` and `cholQR2` simply build on these algorithms, adding passes of `cholQR` to matrices of the same size for both algorithms. Thus, 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 4, `rand_cholQR` is significantly more stable.

3.2 Motivation for Multisketching

The Gaussian sketch performs the dense matrix-matrix multiply with the sketch matrix S of dimension $s \times n$. Hence, we need to store and load the $s \times n$ sketch matrix. As shown in Section 3.1, the time to sketch the matrix becomes dominant in the total `randQR` factorization time (the memory access is expensive on a current computer architecture like a GPU).

One can reduce the sketching cost using a sparse sketch such as a CountSketch matrix [5]. Since the CountSketch matrix has only one non-zero per column, the cost of applying the CountSketch matrix to $V \in \mathbb{R}^{n,m}$ is only $O(nm)$, and it only requires to store $O(n)$ 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, CountSketch could obtain high performance using only readily available linear algebra libraries. However, a CountSketch matrix requires $s = O(m^2)$ to maintain the ε-embedding properties, so one is left to factorize $W \in \mathbb{R}^{s,m}$ with Householder QR, which incurs $O(m^4)$ FLOPs. In contrast, the Gaussian sketch ensures that S is an ε-subspace embedding with $s = O(m)$, 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 [7]. 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 $W \in \mathbb{R}^{s,m}$ with $s =$

$O(m)$ like the Gaussian sketch does to avoid a performance bottleneck from Householder QR. This is possible by using a "Multisketch" framework with first a sparse CountSketch and then a Gaussian sketch. To see this, suppose $S_1 \in \mathbb{R}^{s_1,n}$ is a CountSketch matrix with $s_1 = \frac{m^2+m}{\varepsilon_1^2 d_1}$, cf.(4), and suppose $S_2 \in \mathbb{R}^{s_2,s_1}$ is a Gaussian sketch where $s_2 = 2m$. By Proposition 3.1, for any $V \in \mathbb{R}^{n,m}$, with probability at least $1 - d$

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

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 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. 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.

4 Error Bounds for Floating Point Arithmetic

Here, 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`. For the sake of organization, we define a set of assumptions stating V is numerically full rank (i.e., $\kappa(V) \leq O(\mathbf{u}^{-1})$), $n \gg m$, and that the sketch matrices S_1, S_2 simultaneously satisfy the subspace embedding properties, ensuring equations (1)–(3), (5)–(7) 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 4.1. *Suppose $S_1 \in \mathbb{R}^{s_1,m}$ and $S_2 \in \mathbb{R}^{s_2,s_1}$ 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*

$$\varepsilon_L \in \left[0, \frac{221}{225} - \frac{4}{225}\varepsilon_H\right).$$

Further, suppose $V \in \mathbb{R}^{n,m}$ with $m^{3/2} \leq n$ and $m \leq s_2 \leq s_1 \leq n$ satisfies $cnm\boldsymbol{u} \leq \frac{1}{12}$ and

$$\delta := \frac{12\sqrt{1+\epsilon_H}}{\sqrt{1-\epsilon_L}} \cdot 1.1c\boldsymbol{u}\sqrt{m}\,(1.1n(1 + 1.1cs_2 m\boldsymbol{u}) + s_2 m)\,\kappa(V) \leq 1.$$

Remark 3.1 indicates that in exact arithmetic, `randQR` yields a matrix Q that is orthogonal with respect to $(S_2S_1\cdot, S_2S_1\cdot)$. 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 $(S_2S_1\cdot, S_2S_1\cdot)$ is $O(\mathbf{u})\kappa(V)$, and the factorization error is $O(\mathbf{u})\|V\|_2$. To maintain the flow of the exposition all proofs of the results in this section are postponed until the Appendices.

Theorem 4.1 (`randQR` Errors)**.** *Suppose Assumptions 4.1 are satisfied. With probability at least $1-d$, the $\hat{Q}, \hat{R}$ factors obtained with Algorithm 1 (`randQR`) satisfy*

$$\|(S_2S_1\hat{Q})^T(S_2S_1\hat{Q}) - I\|_2 \leq 3\delta,$$
$$\|V - \hat{Q}\hat{R}\|_2 \leq \frac{\delta}{9}\sigma_m(V). \tag{8}$$

Similar to the analysis of the condition number of Q generated by `randQR` in exact arithmetic in Section 3, we show next that provided that V has full numerical rank, then $\hat{Q}$ generated by `randQR` in floating point arithmetic also has $\kappa(\hat{Q}) = O(1)$.

Theorem 4.2 (Conditioning of `randQR`)**.** *Suppose Assumptions 4.1 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,*

$$\kappa(\hat{Q}) \leq \frac{20}{15\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 2}\cdot \tag{9}$$

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 4.3 (`rand_cholQR` Errors)**.** *Suppose Assumptions 4.1 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. In fact,*

$$\|\hat{Q}^T\hat{Q} - I\|_2 \leq \frac{2000}{\left(15\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 2\right)^2}\left(nm + m(m+1)\right)\boldsymbol{u}, \tag{10}$$

$$\|V - \hat{Q}\hat{R}\|_2 \leq \left(\frac{17m^{3/2}}{\frac{7.5}{\sqrt{1+\epsilon_H}} - 1} + \frac{1.5cm}{\sqrt{1-\varepsilon_L}}\sqrt{1 + \frac{2000(nm + m(m+1))\boldsymbol{u}}{\left(15\sqrt{\frac{1-\varepsilon_L}{1+\varepsilon_H}} - 2\right)^2}}\right) \times$$
$$\left(\sqrt{m}\sqrt{1+\varepsilon_H}\|V\|_2 + \frac{\sqrt{1-\varepsilon_L}}{12}\sigma_m(V)\delta\right)\sqrt{m}\boldsymbol{u} + \frac{\delta}{9}\sigma_m(V).$$

Theorem 4.2 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 4.4 (Conditioning of `rand_cholQR`). *Suppose Assumptions 4.1 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,*

$$\kappa(\hat{Q}) < \sqrt{\frac{1 + \frac{2000}{\left(15\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}}-2\right)^2}(nm+m(m+1))\,\boldsymbol{u}}{1 - \frac{2000}{\left(15\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}}-2\right)^2}(nm+m(m+1))\,\boldsymbol{u}}}. \tag{11}$$

Furthermore, if $\frac{2000}{\left(15\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}}-2\right)^2}(nm+m(m+1))\,\boldsymbol{u} < \frac{1}{2}$, *then*

$$\kappa(\hat{Q}) < 1 + \frac{4000}{\left(15\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}}-2\right)^2}(nm+m(m+1))\,\boldsymbol{u}. \tag{12}$$

Theorems 4.1–4.4 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, that is, for a single (ε, d, m) oblivious ℓ_2-subspace embedding for any $\varepsilon \in [0, \frac{221}{229})$, covering nearly the entire range of possible $\varepsilon \in [0, 1)$ for such embeddings.

Assumptions 4.2. *Suppose $V \in \mathbb{R}^{n,m}$, $\varepsilon \in [0, \frac{221}{229})$, $S \in \mathbb{R}^{s,m}$ is a (ε, d, m) oblivious ℓ_2-subspace embedding, $m^{3/2} \le n$, $m \le s \le n$, $cnm\boldsymbol{u} \le \frac{1}{12}$, and*

$$\delta := \frac{12\sqrt{1+\epsilon}}{\sqrt{1-\epsilon}} \cdot 1.1c\boldsymbol{u}\sqrt{m}\,(1.1n(1+1.1csm\boldsymbol{u}) + sm)\,\kappa(V) \le 1.$$

Corollary 4.1 (`randQR` Errors). *Suppose Assumptions 4.2 are satisfied. Then with probability at least $1-d$, the $\hat{Q}, \hat{R}$ factors obtained with Algorithm 1 (`randQR`) satisfy*

$$\|(S\hat{Q})^T(S\hat{Q}) - I\|_2 \le 3\delta,$$
$$\|V - \hat{Q}\hat{R}\|_2 \le \frac{\delta}{9}\sigma_m(V).$$

Corollary 4.2 (Conditioning of `randQR`). *Suppose Assumptions 4.2 are satisfied. Then with probability at least $1-d$, the matrix $\hat{Q}$ obtained with Algorithm 1 (`randQR`) satisfies*

$$\kappa(\hat{Q}) \le \frac{20}{15\sqrt{\frac{1-\epsilon}{1+\epsilon}} - 2}\ .$$

Therefore, if $\varepsilon \le 0.9$,

$$\kappa(\hat{Q}) \le 13.88.$$

Corollary 4.3 (`rand_cholQR` Errors)**.** *Suppose Assumptions 4.2 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,*

$$\|\hat{Q}^T\hat{Q} - I\|_2 \leq \frac{2000}{\left(15\sqrt{\frac{1-\epsilon}{1+\epsilon}} - 2\right)^2}\left(nm + m(m+1)\right)\boldsymbol{u},$$

$$\|V - \hat{Q}\hat{R}\|_2 \leq \left(\frac{17m^{3/2}}{\frac{7.5}{\sqrt{1+\epsilon_H}} - 1} + \frac{1.5cm}{\sqrt{1-\varepsilon_L}}\sqrt{1 + \frac{2000(nm + m(m+1))\boldsymbol{u}}{\left(15\sqrt{\frac{1-\varepsilon_L}{1+\varepsilon_H}} - 2\right)^2}}\right)$$
$$\left(\sqrt{m}\sqrt{1+\varepsilon_H}\|V\|_2 + \frac{\sqrt{1-\varepsilon_L}}{12}\sigma_m(V)\delta\right)\sqrt{m}\boldsymbol{u} + \frac{\delta}{9}\sigma_m(V).$$

Corollary 4.4 (Conditioning of `rand_cholQR`)**.** *Suppose Assumptions 4.2 are satisfied. Then with probability at least* $1-d$*, the matrix* $\hat{Q}$ *obtained with Algorithm 3 (`rand_cholQR`) satisfies* $\kappa(\hat{Q}) \approx 1$*. In fact,*

$$\kappa(\hat{Q}) < \sqrt{\frac{1 + \frac{2000}{\left(15\sqrt{\frac{1-\epsilon}{1+\epsilon}}-2\right)^2}\left(nm + m(m+1)\right)\boldsymbol{u}}{1 - \frac{2000}{\left(15\sqrt{\frac{1-\epsilon}{1+\epsilon}}-2\right)^2}\left(nm + m(m+1)\right)\boldsymbol{u}}}.$$

Furthermore, if $\frac{2000}{\left(15\sqrt{\frac{1-\epsilon}{1+\epsilon}}-2\right)^2}\left(nm + m(m+1)\right)\boldsymbol{u} < \frac{1}{2}$*, then*

$$\kappa(\hat{Q}) < 1 + \frac{4000}{\left(15\sqrt{\frac{1-\epsilon}{1+\epsilon}} - 2\right)^2}\left(nm + m(m+1)\right)\boldsymbol{u}.$$

5 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` is not new. The earliest appearance of such an algorithm was by Rokhlin and Tygert in 2008 [24] 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.

While this paper was being written, Balabanov proposed the "RCholeskyQR" and "RCholeskyQR2" methods in an unpublished manuscript [3], which are identical to what we refer to as `randQR` and `rand_cholQR`, respectively, in the case of a single (ε, d, m) oblivious ℓ_2-subspace embedding, and gave stability results similar to Corollary 4.2 of this paper. However, our results differ from Balabanov's, as ours impose no assumptions on the level of accuracy performed

by subroutines within the algorithm, meticulously deriving 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 $\epsilon < \frac{221}{229}$ 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 2. 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.

Our results extend beyond a single (ε, d, m) oblivious ℓ_2-subspace embedding, 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`, and the loss of orthogonality error in the standard Euclidean inner product of `rand_cholQR`.

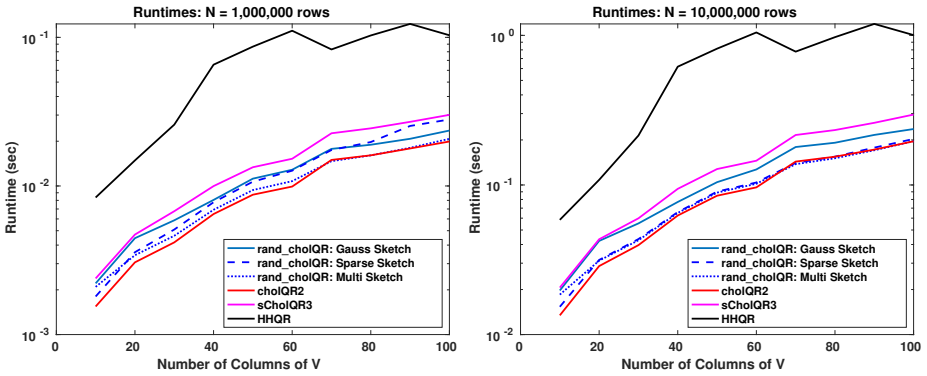
Our work is novel in several ways. To our knowledge, this work is the first to propose a randomized QR algorithm with multiple sketches. The stability results in this paper improve upon and expand the existing stability analysis of `randQR` and `rand_cholQR`, and considers the multisketch case for the 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.

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 latest GPU leveraging vendor-optimized libraries. Second, we empirically validate the bounds given in Section 4, 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 [8] and Kokkos Kernels [23]. 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 [18, 21]. 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, 19, 20]. Test results were obtained using Kokkos 3.7.01, Cuda 11.7.99, and



(a) $n = 1,000,000$ rows (b) $n = 10,000,000$ rows

Figure 1: Runtimes (in seconds) of QR factorizations of V with $\kappa(V) = 10^6$ for a fixed number of rows as the number of columns vary.

GCC 7.2.0 on an AMD EPYC 7742 64-Core 2.25GHz CPU with a NVIDIA A100-SXM4 40GB GPU. All computations were done in double precision, so $\mathbf{u} = 2^{-52} \approx 10^{-16}$.

We tested a variety of sketching strategies. The simplest was the case of a Gaussian sketch $S = \frac{1}{\sqrt{s}}G \in \mathbb{R}^{s,n}$, which were generated within a parallel for loop. The sketch size chosen for a Gaussian to embed $V \in \mathbb{R}^{n,m}$ was $s = \lceil 74.3 \log(m) \rceil$, which can be shown to produce a $(0.49, 1/m, m)$ oblivious ℓ_2-subspace embedding [1, Lemma 4.1]. To test using CountSketch, we explicitly constructed a sparse matrix and applied the sketch using a sparse-matrix vector product. The sketch size used to embed $V \in \mathbb{R}^{n,m}$ with a $S \in \mathbb{R}^{s,n}$ CountSketch matrix was $s = \lceil 8.24(m^2 + m) \rceil$, which can be shown to be a $(0.9, 0.15, m)$ oblivious ℓ_2-subspace embedding [16, Theorem 1].

In our implementation of multisketching, we chose $S_1 \in \mathbb{R}^{s_1,m}$ as a CountSketch described above with $\varepsilon_1 = 0.9$, and $S_2 \in \mathbb{R}^{s_2,s_1}$ a Gaussian sketch with $s_2 = \lceil 74.3 \log(s_1) \rceil$ giving $\varepsilon_2 = 0.49$. Thus, $S_2 S_1$ 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 4.1, and that both of S_1 and S_2 satisfy Assumptions 4.2, ensuring the analysis in Section 4 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.

6.2 Performance and Numerical Results

Figure 1 shows the runtimes of each QR method for test problems with $n = 10^6$ and $n = 10^7$ rows, and $m = 10$–100 columns. Since `cholQR2` is typically expected to be the fastest algorithm, Table 1 shows the relative slowdown of each QR method compared to `cholQR2` averaged across each data point from Figure 1. Table 1 and Figure 1 indicate that `cholQR2` is indeed the fastest method in

Average Slowdown compared to `cholQR2`

	1,000,000 rows	10,000,000 rows
`rand_cholQR`: Gauss Sketch	21.5%	23.1%
`rand_cholQR`: CountSketch	19.5%	5.3%
`rand_cholQR`: multisketch	7.1%	4.9%
`sCholQR3`	35.9%	33.8%
Householder	84.8%	83.4%

Table 1: Average slowdowns of each QR algorithm compared to `cholQR2`, taken from experiments shown in Figure 1. Smaller values indicate faster runtimes. Slowdowns for each QR algorithm are measured as (QR algorithm runtime − `cholQR2` runtime)/(`cholQR2` runtime) × 100%

general, while multisketch `rand_cholQR` performs the closest to `cholQR2`, averaging only a 4.9–7.1% slowdown. Additionally, Figure 1 shows 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 $m = 70$–80 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 Figure 1, and both algorithms have the same $O(\mathbf{u})\kappa(V) < 1$ stability requirement.

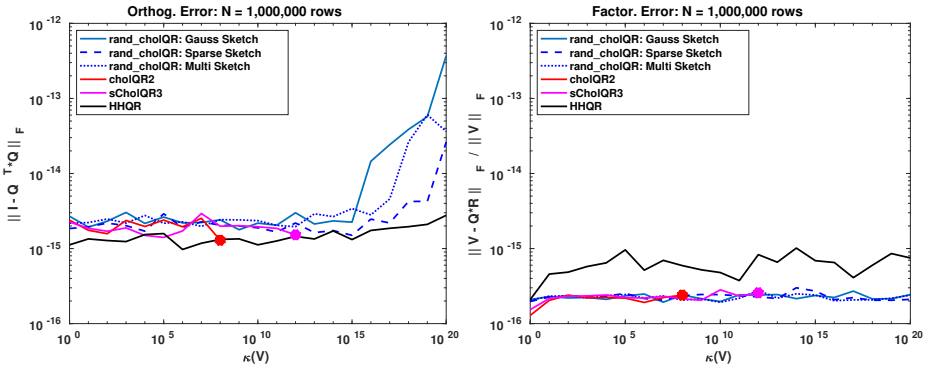
Figure 2 shows the orthogonalization error $\|I - \hat{Q}^T\hat{Q}\|_F$ and the relative factorization error $\|V - \hat{Q}\hat{R}\|_F/\|V\|_F$ for condition number $\kappa(V) \in [1, 10^{16}]$. The results demonstrate that `rand_cholQR` maintains $O(\mathbf{u})$ orthogonality error and $O(\mathbf{u})\|V\|_2$ factorization error[1] while $\kappa(V) < O(\mathbf{u})$, as predicted by Theorem 4.3, 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 and 2 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.

7 Conclusions

The results in Section 4 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 $\kappa(V) < O(\mathbf{u}^{-1/2})$ to ensure a stable factorization. Our results for a single sketch apply for any ε-embedding with $\varepsilon \in [0, \frac{221}{229})$, covering nearly the entire possible range for ε-embeddings.

Our performance results in Section 6.2 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

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



(a) Orthogonality Error (b) Relative Factorization Error

Figure 2: Orthogonality (left) and relative factorization error (right) of the QR factorization of a matrix V with varying condition number. To explicitly control $\kappa(V)$, $V := L\Sigma R^T \in \mathbb{R}^{n,m}$ 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.

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 [12, 22], s-step [6, 14, 28], and enlarged Krylov methods [11], 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.

8 Acknowledgements

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] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

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

[15] 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.

[16] 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.

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

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

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

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

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

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

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

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

[31] 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 Preparatory Results for our Proofs

Let $V \in \mathbb{R}^{n,m}$, $n \gg m$, and suppose $S_1 \in \mathbb{R}^{s_1,m}$ and $S_2 \in \mathbb{R}^{s_2,s_1}$ are (ε_1, d_1, m) and (ε_2, d_2, m) oblivious ℓ_2-subspace embeddings, respectively, generated independently. Define $d = d_1 + d_2 - d_1 d_2$, $\varepsilon_L = \varepsilon_1 + \varepsilon_2 - \varepsilon_1\varepsilon_2$, $\varepsilon_H = \varepsilon_1 + \varepsilon_2 + \varepsilon_1\varepsilon_2$. Define δ in the following way, and assume it is bounded by 1:

$$\delta := \frac{12\sqrt{1+\epsilon_H}}{\sqrt{1-\epsilon_L}} \cdot 1.1c\mathbf{u}\sqrt{m}\,(1.1n(1+1.1cs_2m\mathbf{u}) + s_2m)\,\kappa(V) \leq 1\ . \tag{13}$$

Assume that

$$m \leq s_2 \leq s_1 \leq n \tag{14}$$

which is logical, otherwise the embeddings $S_2 \in \mathbb{R}^{s_2,s_1}$ and $S_1 \in \mathbb{R}^{s_1,n}$ project V into a larger space, defeating the purpose of sketching.

For some small integer constant c, define

$$\gamma_k := \frac{ck\mathbf{u}}{1-ck\mathbf{u}}\ .$$

Provided $ck\mathbf{u} < \frac{1}{11}$, it follows that $\gamma_k \leq 1.1ck\mathbf{u}$. In particular, since $\kappa(V) \geq 1$, we can deduce from (13) that

$$\max\{cs_2m\mathbf{u}, cn\mathbf{u}\} \leq cn\mathbf{u} + cs_2m\mathbf{u} \leq \frac{1}{12}\ , \tag{15}$$

implying that

$$\gamma_n \leq 1.1cn\mathbf{u},\ \gamma_m \leq 1.1cm\mathbf{u},\ \gamma_{s_1} \leq 1.1cs_1\mathbf{u} \text{ and } \gamma_{s_2m} \leq 1.1cs_2m\mathbf{u}\ , \tag{16}$$

and

$$1 + \gamma_n \leq 1.1\ . \tag{17}$$

Finally, assume that

$$cnm\mathbf{u} \leq \frac{1}{12}\ , \tag{18}$$

which is not directly implied by (13), but, depending of the values of s_2 and m, often is a consequence of it.

A.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, error E_1 is incurred when performing the matrix products $W = S_2S_1V$ in Step 1 of Algorithm 1, i.e., the computed $\hat{W}$ satisfies

$$\hat{W} = S_2S_1V + E_1. \tag{19}$$

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

$$Q_{tmp}\hat{R} = \hat{W} + E_2 = S_2 S_1 V + E_1 + E_2. \tag{20}$$

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

$$\hat{Q}_{i,:} = V_{i,:}(\hat{R} + \Delta\hat{R}_i)^{-1} \quad (i = 1, 2, \ldots m), \tag{21}$$

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 can recast the errors incurred in Step 3 as $\hat{Q} = (V + \Delta\tilde{V})\hat{R}^{-1}$, which simplifies the analysis of the orthogonality of $\hat{Q}$.

A.2 Forward Error in matrix-matrix multiplication S_2S_1V

For $x, y \in \mathbb{R}^n$, $\mathrm{fl}(x^T y) = x^T(y + \Delta y)$, where $|\Delta y| < \gamma_n |y|$ [13]. Then for $A \in \mathbb{R}^{m,n}$, $x \in \mathbb{R}^n$, the matrix-vector product $y = Ax$ in floating point satisfies, $\hat{y}_i = A_{i,:}(x + \Delta x)$, where $|\Delta x| < \gamma_n |x|$ and $A_{i,:}$ is the i^{th} row of A. Thus, $\hat{y} = A(x + \Delta x)$, where $|\Delta x| < \gamma_n |x|$.

Therefore, for $A \in \mathbb{R}^{m,n}$, $B \in \mathbb{R}^{n,k}$, $C = AB$ executed in floating point,

$$\hat{c}_j = A(b_j + \Delta b_j), \quad |\Delta b_j| < \gamma_n |b_j| \text{ for each } j = 1, \ldots, k.$$

Equivalently,

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

Thus, in floating point, Step 1 of Algorithm 1 becomes:

$$\hat{W}_1 = S_1(V + \Delta V), \quad |\Delta V| < \gamma_n |V| \ , \tag{22}$$

$$\hat{W} = S_2(\hat{W}_1 + \Delta\hat{W}_1), \quad |\Delta\hat{W}_1| < \gamma_{s_1} |\hat{W}_1| = \gamma_{s_1} |S_1 V + S_1 \Delta V| \cdot \tag{23}$$

Then the forward error of this matrix-matrix product is:

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

By the embedding properties of S_2, with probability at least $1 - d_2$,

$$\|E_1\|_2 \le \|E_1\|_F \le \sqrt{1 + \varepsilon_2}\|\Delta\hat{W}_1\|_F \cdot \tag{24}$$

By (22) and (23), with probability at least $1 - d_1$,

$$\|\Delta\hat{W}_1\|_F < \gamma_{s_1}(1 + \gamma_n)\sqrt{1 + \varepsilon_1}\|V\|_F \le \gamma_{s_1}(1 + \gamma_n)\sqrt{1 + \varepsilon_1}\sqrt{m}\|V\|_2 \cdot \tag{25}$$

Notice (14), (16), and (17) imply $\gamma_{s_1}(1 + \gamma_n) < 1.21cn\mathbf{u}$. This, combined with (24) and (25), ensures with probability at least $1 - d = (1 - d_1)(1 - d_2)$ that

$$\|E_1\|_F \le \gamma_{s_1}(1 + \gamma_n)\sqrt{1 + \epsilon_H}\sqrt{m}\|V\|_2 \le 1.21cn\mathbf{u}\sqrt{1 + \epsilon_H}\sqrt{m}\|V\|_2 \cdot \tag{26}$$

A.3 Backward Error of Householder QR of $\hat{W}$

By [13, Theorem 19.4], Householder QR of $\hat{W} \in \mathbb{R}^{s_2,m}$ returns a triangular $\hat{R} \in \mathbb{R}^{m,m}$ so that some orthogonal $Q_{tmp} \in \mathbb{R}^{s_2,m}$ satisfies,

$$\hat{W} + E_2 = Q_{tmp}\hat{R}, \quad \|(E_2)_j\|_2 \leq \gamma_{s_2 m}\|\hat{w}_j\|_2, \text{ for } j = 1, \ldots, m. \tag{27}$$

By (26), the relation $\hat{W} = S_2 S_1 V + E_1$, and the embedding properties of $S_2 S_1$,

$$\|E_2\|_F \leq \gamma_{s_2 m}\|\hat{W}\|_F \leq \gamma_{s_2 m}\sqrt{1+\epsilon_H}(1 + 1.21cn\mathbf{u})\|V\|_F \ , \tag{28}$$

with probability at least $1 - d$. Finally, since $\|V\|_F \leq \sqrt{m}\|V\|_2$ and by (16),

$$\|E_2\|_2 \leq \|E_2\|_F \leq 1.1cs_2 m\mathbf{u}\sqrt{1+\epsilon_H}(1 + 1.21cn\mathbf{u})\sqrt{m}\|V\|_2 \cdot \tag{29}$$

A.4 Backward Error of the Forward Substitution

In Step 3 of `randQR`, we solve for Q via the triangular system $Q\hat{R} = V$. By [13, Theorem 8.5], in floating point, $\hat{Q}_{i,:}$ satisfies

$$\hat{Q}_{i,:}(\hat{R} + \Delta R_i) = V_{i,:}, \quad |\Delta R_i| < \gamma_m|\hat{R}| \text{ for } i = 1, \ldots n. \tag{30}$$

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 (30), and therefore each row must be accounted for separately. For each $i = 1, \ldots, n$,

$$\|\Delta\hat{R}_i\|_2 \leq \|\Delta\hat{R}_i\|_F = \||\Delta\hat{R}_i|\|_F < \gamma_m\||\hat{R}|\|_F = \gamma_m\|\hat{R}\|_F \leq \gamma_m\sqrt{m}\|\hat{R}\|_2 \cdot \tag{31}$$

By (27), (28), and the orthogonality of Q_{tmp}, it follows that

$$\|\hat{R}\|_2 \leq \|\hat{R}\|_F = \|Q_{tmp}\hat{R}\|_F = \|\hat{W} + E_2\|_F \leq (1 + \gamma_{s_2 m})\|\hat{W}\|_F \cdot \tag{32}$$

Therefore, by (16), (28), (31), (32), and since $\|V\|_F \leq \sqrt{m}\|V\|_2$,

$$\|\Delta\hat{R}_i\|_2 \leq 1.1cm^2\mathbf{u}(1 + 1.1cs_2 m\mathbf{u})\sqrt{1+\epsilon_H}(1 + 1.21cn\mathbf{u})\|V\|_2 \cdot$$

A.5 Bounding the 2-norm of $\hat{R}^{-1}$ and $V\hat{R}^{-1}$

By (6), (19), and Weyl's inequality [29], with probability at least $1 - d$,

$$\begin{aligned}\sigma_m(\hat{W} + E_2) &\geq \sigma_m(\hat{W}) - \|E_2\|_2 \geq \sigma_m(S_2 S_1 V) - (\|E_1\|_2 + \|E_2\|_2) \\ &\geq \sqrt{1-\epsilon_L}\ \sigma_m(V) - (\|E_1\|_2 + \|E_2\|_2).\end{aligned} \tag{33}$$

By (13), and the fact that the fact that $\|V\|_2 = \kappa(V)\sigma_m(V)$,

$$\sqrt{1+\epsilon_H}1.1c\mathbf{u}\sqrt{m}\,(1.1n(1 + 1.1cs_2 m\mathbf{u}) + s_2 m)\,\|V\|_2 \leq \frac{\sqrt{1-\epsilon_L}}{12}\sigma_m(V). \tag{34}$$

Combining (26), (29), and (34) and the assumption that $\delta \leq 1$, results in:

$$\|E_1\|_2 + \|E_2\|_2 \leq \|E_1\|_F + \|E_2\|_F \leq \frac{\sqrt{1-\epsilon_L}}{12}\sigma_m(V)\delta \leq \frac{\sqrt{1-\epsilon_L}}{12}\sigma_m(V), \quad (35)$$

so by (33) and (35),

$$\sigma_m(\hat{R}) = \sigma_m(Q_{tmp}\hat{R}) = \sigma_m(\hat{W} + E_2) \geq \frac{11\sqrt{1-\epsilon_L}}{12}\ \sigma_m(V). \quad (36)$$

Therefore, by (36)

$$\|\hat{R}^{-1}\|_2 \leq \frac{12}{11\sqrt{1-\epsilon_L}}\left(\sigma_m(V)\right)^{-1}. \quad (37)$$

By (20), we have that Step 2 of `randQR` satisfies

$$S_2S_1V\hat{R}^{-1} = Q_{tmp} - (E_1 + E_2)\hat{R}^{-1}. \quad (38)$$

Thus, by (35), (37), (38), the fact that Q_{tmp} is orthogonal,

$$\|S_2S_1V\hat{R}^{-1}\|_2 \leq \|Q_{tmp}\|_2 + (\|E_1\|_2 + \|E_2\|_2)\|\hat{R}^{-1}\|_2 \leq \frac{12}{11}\ , \quad (39)$$

with probability at least $1 - d$. Therefore, by (5),

$$\|V\hat{R}^{-1}\|_2 \leq \frac{1}{\sqrt{1-\epsilon_L}}\|SV\hat{R}^{-1}\|_2 \leq \frac{12}{11\sqrt{1-\epsilon_L}}\ .$$

A.6 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 (21), we capture the errors of each triangular solve in a matrix $\Delta\tilde{V}$, where

$$\hat{Q} = (V + \Delta\tilde{V})\hat{R}^{-1} \iff \hat{Q}\hat{R} = V + \Delta\tilde{V}. \quad (40)$$

We note that we use the notation $\Delta\tilde{V}$ to diferenciate it from ΔV from (22). From (21), we have $\hat{Q}_{i,:}(R + \Delta R_i) = V_{i,:}$. Then $\Delta\tilde{V}$ can be defined row-wise,

$$\Delta\tilde{V}_{i,:} = -\hat{Q}_{i,:}\Delta R_i. \quad (41)$$

Thus, by (30), $|\Delta\tilde{V}_{i,:}| \leq 1.1cm\mathbf{u}|\hat{Q}_{i,:}||\hat{R}|$, and so $|\Delta\tilde{V}| \leq 1.1cm\mathbf{u}|\hat{Q}||\hat{R}|$, hence,

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

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

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

and therefore by (16), (28), (32), and the fact that $\|V\|_F \leq \sqrt{m}\|V\|_2$,

$$\begin{aligned}\|\Delta\tilde{V}\|_2 \leq \|\Delta\tilde{V}\|_F &\leq 1.1cm^{3/2}\mathbf{u}\|\hat{Q}\|_2\|\hat{R}\|_F \\ &\leq 1.1cm^2\mathbf{u}(1+1.1cs_2m\mathbf{u})(1+1.21cn\mathbf{u})\sqrt{1+\epsilon_H}\|V\|_2\|\hat{Q}\|_2. \end{aligned} \tag{42}$$

Further, using (15) to ensure $1 + 1.21cn\mathbf{u} < 1.21$, the fact that $s_2m > 0$, and provided that $m^{3/2} \leq n$ (implying $1.1m^2 \leq 1.1n\sqrt{m}$), and by (42),

$$\begin{aligned}\|\Delta\tilde{V}\|_F &\leq c\sqrt{m}\mathbf{u}(1.1n(1+1.1cs_2m\mathbf{u}))(1+1.21cn\mathbf{u})\sqrt{1+\epsilon_H}\|V\|_2\|\hat{Q}\|_2 \\ &\leq 1.21c\sqrt{m}\mathbf{u}(1.1n(1+1.1cs_2m\mathbf{u}))\sqrt{1+\epsilon_H}\|V\|_2\|\hat{Q}\|_2 \\ &\leq 1.1\cdot 1.1c\sqrt{m}\mathbf{u}(1.1n(1+1.1cs_2m\mathbf{u})+s_2m)\sqrt{1+\epsilon_H}\|V\|_2\|\hat{Q}\|_2. \end{aligned} \tag{43}$$

The remaining issue to resolve is that the bound on $\|\Delta\tilde{V}\|_F$ in (43) requires knowledge of $\|\hat{Q}\|_2$, which we have not yet found. Combining (13), (37), and (43) gives,

$$\begin{aligned}\|\hat{Q} - V\hat{R}^{-1}\|_F = \|\Delta\tilde{V}\hat{R}^{-1}\|_F &\leq \|\Delta\tilde{V}\|_F\|\hat{R}^{-1}\|_2 \\ &\leq 1.1\cdot\frac{12\sqrt{1+\epsilon_H}}{11\sqrt{1-\epsilon_L}}1.1c\sqrt{m}\mathbf{u}\left(1.1n(1+1.1cs_2m\mathbf{u})+s_2m\right)\kappa(V)\|\hat{Q}\|_2 \\ &= \frac{\delta}{10}\|\hat{Q}\|_2 \leq \frac{1}{10}\|\hat{Q}\|_2. \end{aligned} \tag{44}$$

Now, by (35), (37), and (38),

$$\begin{aligned}\|S_2S_1V\hat{R}^{-1} - Q_{tmp}\|_2 \leq \|S_2S_1V\hat{R}^{-1} - Q_{tmp}\|_F &= \|(E_1+E_2)\hat{R}^{-1}\|_F \\ &\leq \left(\|E_1\|_F + \|E_2\|_F\right)\|\hat{R}^{-1}\|_2 \leq \frac{\delta}{11}\cdot \end{aligned} \tag{45}$$

Applying Weyl's inequality to (45) and the fact that Q_{tmp} is orthogonal yields,

$$1 - \frac{\delta}{11} \leq \sigma_m(S_2S_1V\hat{R}^{-1}) \leq \sigma_1(S_2S_1V\hat{R}^{-1}) \leq 1 + \frac{\delta}{11}\cdot$$

It follows from (6) that

$$\frac{1-\frac{\delta}{11}}{\sqrt{1+\epsilon_H}} \leq \sigma_m(V\hat{R}^{-1}) \leq \sigma_1(V\hat{R}^{-1}) \leq \frac{1+\frac{\delta}{11}}{\sqrt{1-\epsilon_L}} \leq \frac{12}{11\sqrt{1-\epsilon_L}}\cdot \tag{46}$$

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

$$\sigma_m(V\hat{R}^{-1}) - \|\hat{Q} - V\hat{R}^{-1}\|_2 \leq \sigma_m(\hat{Q}) \leq \sigma_1(\hat{Q}) \leq \sigma_1(V\hat{R}^{-1}) + \|\hat{Q} - V\hat{R}^{-1}\|_2\cdot \tag{47}$$

Then, by (44), (46), and (47),

$$\|\hat{Q}\|_2 = \sigma_1(\hat{Q}) \leq \sigma_1(V\hat{R}^{-1}) + \|\hat{Q} - V\hat{R}^{-1}\|_2 \leq \frac{12}{11\sqrt{1-\epsilon_L}} + \frac{1}{10}\|\hat{Q}\|_2, \tag{48}$$

and so,

$$\|\hat{Q}\|_2 \leq \frac{120}{99\sqrt{1-\epsilon_L}} \cdot \tag{49}$$

Then, we obtain from (44),

$$\|\hat{Q} - V\hat{R}^{-1}\|_2 = \|\Delta\tilde{V}\hat{R}^{-1}\|_2 \leq \|\Delta\tilde{V}\hat{R}^{-1}\|_F \leq \frac{\delta}{10}\|\hat{Q}\|_2 \leq \frac{12\delta}{99\sqrt{1-\epsilon_L}}. \tag{50}$$

Furthermore, by (5),

$$\|S_2S_1\Delta\tilde{V}\hat{R}^{-1}\|_2 \leq \sqrt{1+\epsilon_H}\|\Delta\tilde{V}\hat{R}^{-1}\|_2 \leq \frac{12\sqrt{1+\epsilon_H}}{99\sqrt{1-\epsilon_L}}\ \delta. \tag{51}$$

B Proofs of Results Given in Section 4

A useful consequence of Assumptions 4.1 is that

$$1 - \varepsilon_L > \frac{4}{225}(1+\varepsilon_H),$$

and thus

$$\frac{1+\varepsilon_H}{1-\varepsilon_L} < \frac{225}{4} \Rightarrow \sqrt{\frac{1+\varepsilon_H}{1-\varepsilon_L}} < \frac{15}{2}\ . \tag{52}$$

B.1 Proof of Theorem 4.1 (`randQR` Errors)

Proof. Observe that by (20), we have $S_2S_1V = Q_{tmp}\hat{R} - (E_1+E_2)$, and so

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

Using (40) to expand $S_2S_1\hat{Q} = (S_2S_1V + S_2S_1\Delta\tilde{V})\hat{R}^{-1}$, we get

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

Therefore, by (35), (37), (39), (51), and (52), and the fact that $\delta^2 \leq \delta \leq 1$,

$$\begin{aligned}&\|(S_2S_1\hat{Q})^T(S_2S_1\hat{Q}) - I\|_2 \\ &\qquad\leq 2(\|E_1\|_2 + \|E_2\|_2)\|\hat{R}^{-1}\|_2 + (\|E_1\|_2 + \|E_2\|_2)^2\|\hat{R}^{-1}\|_2^2 \\ &\qquad + 2\|S_2S_1\Delta\tilde{V}\hat{R}^{-1}\|_2\|S_2S_1V\hat{R}^{-1}\|_2 + \|S_2S_1\Delta\tilde{V}\hat{R}^{-1}\|_2^2 \\ &\qquad\leq 2\frac{\delta}{11} + \left(\frac{\delta}{11}\right)^2 + 2\frac{12\sqrt{1+\epsilon_H}}{99\sqrt{1-\epsilon_L}}\ \delta\frac{12}{11} + \left(\frac{12\sqrt{1+\epsilon_H}}{99\sqrt{1-\epsilon_L}}\ \delta\right)^2 \\ &\qquad\leq \frac{1}{1089}\left(207 + \frac{16(1+\epsilon_H)}{(1-\epsilon_L)} + \frac{288\sqrt{1+\epsilon_H}}{\sqrt{1-\epsilon_L}}\right)\delta \leq 3\delta.\end{aligned}$$

Equation (8) follows by combining (43) and (49), since $\Delta\tilde{V} = \hat{Q}\hat{R} - V$. □

B.2 Proof of Theorem 4.2 (Conditioning of `randQR`)

Proof. As a direct consequence of (46), (47), (50), and the fact that $\delta \leq 1$,

$$\begin{aligned}\sigma_m(\hat{Q}) &\geq \sigma_m(V\hat{R}^{-1}) - \|\hat{Q} - V\hat{R}^{-1}\|_2 \geq \frac{1-\frac{\delta}{11}}{\sqrt{1+\epsilon_H}} - \frac{12\delta}{99\sqrt{1-\epsilon_L}} \\ &\geq \frac{10}{11\sqrt{1+\epsilon_H}} - \frac{12}{99\sqrt{1-\epsilon_L}} \cdot\end{aligned}$$

Additionally, we found in (49) that

$$\sigma_1(\hat{Q}) = \|\hat{Q}\|_2 \leq \frac{120}{99\sqrt{1-\epsilon_L}} \cdot$$

Thus,

$$\kappa(\hat{Q}) = \frac{\sigma_1(\hat{Q})}{\sigma_m(\hat{Q})} \leq \frac{20}{15\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 2} \cdot \qquad \square$$

B.3 Proof of Theorem 4.3 (`rand_cholQR` Errors)

Proof. We next turn to Algorithm 3. First, we obtain $\hat{Q}_0, \hat{R}_0$ from `randQR` (so that the results in Appendix A 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 4.2, $\hat{Q}_0$ arising from Step 1 of Algorithm 3 satisfies

$$\kappa(\hat{Q}_0) \leq \frac{20}{15\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 2} \cdot$$

By [31, Lemma 3.1], it follows that Step 2 of Algorithm 3 gives $\hat{Q}$ satisfying

$$\begin{aligned}\|\hat{Q}^T\hat{Q} - I\|_2 &\leq \frac{5}{64} 64\kappa(\hat{Q}_0)^2 \left(nm + m(m+1)\right)\mathbf{u} \\ &\leq \frac{2000}{\left(15\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 2\right)^2}\left(nm + m(m+1)\right)\mathbf{u}.\end{aligned}$$

Notice that by (16)–(18),

$$\sqrt{\frac{1+\gamma_n m}{1-\gamma_{m+1}m}} \leq \sqrt{\frac{1+1.1cnm\mathbf{u}}{1-1.1c(m+1)m\mathbf{u}}} \leq \sqrt{\frac{1.1}{0.9}} \leq 1.11. \tag{53}$$

Observe $\hat{R} = (\hat{R}_1 + \Delta\hat{R})R_0$ where $\hat{R}_1$ is the Cholesky factor of $\hat{Q}_0^T\hat{Q}_0$ and $\hat{Q}_0$ results from `randQR`. By [13, Eq. (3.11)], [31, Eq. (3.16)], (16), (49), and (53),

$$\begin{aligned}\|\Delta\hat{R}\|_2 \leq \|\Delta\hat{R}\|_F &\leq \sqrt{m}\gamma_m\|\hat{R}_1\|_2 \leq \sqrt{m}\gamma_m\sqrt{\frac{1+\gamma_n m}{1-\gamma_{m+1}m}}\|\hat{Q}_0\|_2 \\ &\leq 1.23cm^{3/2}\mathbf{u}\frac{120}{99\sqrt{1-\varepsilon_L}} \leq \sqrt{m}\mathbf{u}\frac{1.5cm}{\sqrt{1-\varepsilon_L}} \cdot\end{aligned} \tag{54}$$

Next, observe that by (40) we have that $\hat{Q}_0\hat{R}_0 = V + \Delta\tilde{V}$ from `randQR`. Using this and [31, Eq. (3.24)] to bound $\|\hat{Q}_0 - \hat{Q}\hat{R}_1\|_2$,

$$\begin{aligned}-\|\Delta\tilde{V}\|_2 + \|V - \hat{Q}\hat{R}\|_2 \leq \|V + \Delta\tilde{V} - \hat{Q}\hat{R}\|_2 &= \|\hat{Q}_0\hat{R}_0 - \hat{Q}\hat{R}_1\hat{R}_0 - \hat{Q}\Delta\hat{R}\hat{R}_0\|_2 \\ &\leq \|\hat{R}_0\| \left(\|\hat{Q}_0 - \hat{Q}\hat{R}_1\|_2 + \|\hat{Q}\|_2\|\Delta\hat{R}\|_2\right) \\ &\leq \|\hat{R}_0\|_2 \left(1.4\kappa(\hat{Q}_0)\|\hat{Q}_0\|_2 m^2\mathbf{u} + \|\hat{Q}\|_2\|\Delta\hat{R}\|_2\right). \quad (55)\end{aligned}$$

Observe that (10) implies

$$\|\hat{Q}\|_2 \leq \sqrt{1 + \frac{2000}{\left(15\sqrt{\frac{1-\varepsilon_L}{1+\varepsilon_H}} - 2\right)^2}(nm + m(m+1))\mathbf{u}}\,.$$

Additionally, by (8),

$$\|\Delta\tilde{V}\|_2 = \|V - \hat{Q}_0\hat{R}_0\|_2 \leq \frac{\delta}{9}\sigma_m(V). \quad (56)$$

Now, starting from (32), replace the factors of the bound using (16), (28), and (13) to obtain

$$\begin{aligned}\|\hat{R}_0\|_2 &\leq (1 + 1.1cs_2m\mathbf{u})\sqrt{1+\epsilon_H}(1 + 1.21cn\mathbf{u})\sqrt{m}\|V\|_2 \\ &= \sqrt{m}\sqrt{1+\varepsilon_H}\|V\|_2 + \frac{\sqrt{1-\varepsilon_L}}{12}\sigma_m(V)\delta. \quad (57)\end{aligned}$$

Using (9) and (49) to bound $\kappa(\hat{Q}_0)$ and $\|\hat{Q}_0\|_2$, (54) to bound $\|\Delta\hat{R}\|_2$, (57) to bound $\|\hat{R}_0\|_2$, adding $\|\Delta\tilde{V}\|_2$ to both sides of (55) and then bounding $\|\Delta\tilde{V}\|_2$ using (56), we finally obtain

$$\begin{aligned}\|V - \hat{Q}\hat{R}\|_2 \leq &\left(\frac{17m^{3/2}}{\frac{7.5}{\sqrt{1+\epsilon_H}} - 1} + \frac{1.5cm}{\sqrt{1-\varepsilon_L}}\sqrt{1 + \frac{2000(nm + m(m+1))\mathbf{u}}{\left(15\sqrt{\frac{1-\varepsilon_L}{1+\varepsilon_H}} - 2\right)^2}}\right) \\ &\left(\sqrt{m}\sqrt{1+\varepsilon_H}\|V\|_2 + \frac{\sqrt{1-\varepsilon_L}}{12}\sigma_m(V)\delta\right)\sqrt{m}\mathbf{u} + \frac{\delta}{9}\sigma_m(V),\end{aligned}$$

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

B.4 Proof of Theorem 4.4 (Conditioning of `rand_cholQR`)

Proof. It follows from (10) that the i^{th} eigenvalue of $\hat{Q}^T Q$ satisfies

$$\begin{aligned}\lambda_i(\hat{Q}^T\hat{Q}) &\geq 1 - \frac{2000}{\left(15\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 2\right)^2}(nm + m(m+1))\,\mathbf{u}, \\ \lambda_i(\hat{Q}^T\hat{Q}) &\leq 1 + \frac{2000}{\left(15\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 2\right)^2}(nm + m(m+1))\,\mathbf{u}.\end{aligned}$$

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

$$\sigma_i(\hat{Q}) \geq \sqrt{1 - \frac{2000}{\left(15\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 2\right)^2}\left(nm + m(m+1)\right)\mathbf{u}},$$
$$\sigma_i(\hat{Q}) \leq \sqrt{1 + \frac{2000}{\left(15\sqrt{\frac{1-\epsilon_L}{1+\epsilon_H}} - 2\right)^2}\left(nm + m(m+1)\right)\mathbf{u}},$$

which gives (11). Further, for any $x < \frac{1}{2}$, $\sqrt{\frac{1+x}{1-x}} < 1 + 2x$, which gives (12). □

B.5 Proofs of Corollaries 4.1–4.4 (One Sketch Matrix Theory)

Proof. Considering one subspace embedding $S = S_2$ is equivalent to two subspace embeddings $S_2 S_1$ simply by interpreting $S_1 = I_{m,m}$ as the $m \times m$ identity, which is by definition a $(0, 0, m)$ oblivious ℓ_2-subspace embedding, therefore giving $\varepsilon_1 = 0$ and $d_1 = 0$, thereby allowing $\varepsilon_H = \varepsilon_L = \varepsilon$, $d = d_2$, and $s = s_2$.

We show next that if $\varepsilon = \varepsilon_L = \varepsilon_H \in [0, \frac{221}{229})$, then $\varepsilon_L \in [0, \frac{221}{225} - \frac{4}{225}\varepsilon_H)$. Indeed, $\varepsilon_L \in [0, \frac{221}{225} - \frac{4}{225}\varepsilon_H)$ is equivalent in this case to $0 \leq \varepsilon < \frac{221}{225} - \frac{4}{225}\varepsilon$, or $0 \leq \frac{229}{225}\varepsilon < \frac{221}{225}$, or what is the same, $\varepsilon \in [0, \frac{221}{229})$. This means that when $\varepsilon_H = \varepsilon_L = \varepsilon$, Assumptions 4.2 imply Assumptions 4.1. Thus, Assumptions 4.1 are satisfied and Corollaries 4.1–4.4 are direct consequences of Theorems 4.1–4.4. □