CholeskyQR with Randomization and Pivoting for Tall Matrices (CQRRPT)

Maksim Melnichenko* Oleg Balabanov†‡ Riley Murray§
James Demmel† Michael W. Mahoney‡¶ † Piotr Luszczek‖*

March 18, 2025

Abstract

This paper develops and analyzes a new algorithm for QR decomposition with column pivoting (QRCP) of rectangular matrices with many more rows than columns. The algorithm carefully combines methods from randomized numerical linear algebra to accelerate pivot decisions for the input matrix *and* the process of decomposing the pivoted matrix into the QR form. The source of the latter improvement is CholeskyQR with randomized preconditioning. Comprehensive analysis is provided in both exact and finite-precision arithmetic to characterize the algorithm's rank-revealing properties and its numerical stability granted probabilistic assumptions of the sketching operator. An implementation of the proposed algorithm is described and made available inside the open-source RandLAPACK library, which itself relies on RandBLAS. Experiments with this implementation on an Intel Xeon Gold 6248R CPU demonstrate order-of-magnitude speedups over LAPACK's standard function for QRCP, and comparable performance to a specialized algorithm for unpivoted QR of tall matrices, which lacks the strong rank-revealing properties of the proposed method.

1 Introduction

The QR factorization, considered one of "The Big Six Matrix Factorizations" [Hig22], is fundamental to the field of numerical linear algebra. Its applications range from linear least squares problems [Bjö96] and block orthogonalization [SW02] to contemporary randomized low-rank approximation algorithms [TW23]. Beyond the basic QR decomposition, QR with column pivoting (QRCP) has additional benefits for numerically challenging problems, as it can help with rank-revealing tasks. To describe this problem concretely, let $\mathbf{M}$ be a matrix of size $m \times n$ with $m \geq n$. QRCP is concerned with finding a column permutation matrix $\mathbf{\Pi}$ and a QR decomposition of the product $\mathbf{M\Pi}$, i.e.,

$$\mathbf{M\Pi} = \mathbf{QR}$$

such that the information on the leading and trailing singular values of $\mathbf{M}$ can be inferred from the spectra of leading and trailing blocks in 2×2 partitions of $\mathbf{R}$.

*Innovative Computing Laboratory, University of Tennessee, Knoxville
†University of California, Berkeley
‡International Computer Science Institute (ICSI)
§Sandia National Laboratories
¶Lawrence Berkeley National Laboratory
‖MIT Lincoln Laboratory
Funding acknowledgments and additional affiliations appear at the end of the paper.
Send correspondence to mmelnic1@vols.utk.edu.

Remark 1. We use the term *spectrum* in reference to singular values, not eigenvalues. Eigenvalues are of interest to us only insofar as they coincide with singular values for positive semidefinite matrices.

QRCP is considerably more expensive than unpivoted QR from a communication standpoint, even with straightforward pivoting strategies. For example, Householder QR with Businger and Golub's *max-norm pivoting* requires updating column norms of every partial decomposition of $\mathbf{M}$ as the algorithm progresses from left to right across the columns [BG65]. These column-norm updates entail matrix-vector (BLAS 2) operations, which are far less suitable for modern hardware than the matrix-matrix (BLAS 3) operations abundant in classic algorithms for unpivoted QR. The significance of this problem has been known for decades [QOSB98], and has compounded over time.

We use methods common in Randomized Numerical Linear Algebra (RandNLA) to develop a fast and reliable QRCP algorithm we call *CQRRPT*,[1] which stands for "CholeskyQR with Randomization and Pivoting for Tall matrices." As we show below, CQRRPT can outperform not only other QRCP algorithms but also specialized communication-avoiding algorithms for unpivoted QR, such as shifted CholeskyQR3 [FKN+20] and TSQR [DGHL12]. The dominant cost of our algorithm is attributed exclusively to BLAS 3 operations and a random projection, rendering it highly efficient from a communication standpoint. On distributed memory systems, it can be implemented with just two sum-reduce exchanges. Compared to shifted CholeskyQR3, CQRRPT requires $\frac{4}{3}\times$ fewer data passes and $\frac{3}{2}\times$ fewer inter-processor messages. Furthermore, compared to TSQR, CQRRPT lends itself more easily to performance optimization on massively parallel architectures, thanks to its simple reduction operator. The highly efficient computational profile, together with its numerical stability, suggests that CQRRPT essentially "solves" the algorithm design problem of QRCP for tall matrices.

1.1 Where is the randomness?

CQRRPT uses randomness only once: as its very first step, it samples a linear dimension reduction map from a prescribed distribution. The dimension reduction map is called a *sketching operator* and is said to be sampled from a *sketching distribution*. Its next step is to apply the sketching operator to the input matrix to produce a *sketch*. Computing a sketch can be as simple as selecting rows from a matrix and as complicated as fast Fourier transforms. Our implementation of CQRRPT uses structured sparse sketching operators, since these can provide exceptional speed without sacrificing reliability of the algorithm.

Analysis of sketching-based algorithms often comes in two parts. In the first part, the algorithm is analyzed as a purely deterministic method, conditional on an event that the sketch retains suitable geometric information from the input matrix. The second part uses tools from random matrix theory to bound the probability with which the desirable event happens, where the bound is *unconditional on numerical properties of the input matrix*. Our analysis of CQRRPT follows this tradition. Our presentation strongly emphasizes the first phase of analysis, since the probabilistic event that the first phase conditions on is extremely well understood.

We refer the reader to the recent monograph [MDM+23] for a broad overview of how randomness can be leveraged in numerical linear algebra.

[1] pronounced "see-crypt"

1.2 The CholeskyQR method and its variants

The idea behind CholeskyQR is simple. Given the $m \times n$ matrix $\mathbf{M}$ with $m \geq n$, compute the Gram matrix $\mathbf{G} = \mathbf{M}^*\mathbf{M}$. If we can factor $\mathbf{G} = \mathbf{R}^*\mathbf{R}$ where $\mathbf{R}$ is an invertible upper-triangular matrix, then we can obtain the QR decomposition's orthogonal factor – hereafter, the "Q-factor" – as $\mathbf{Q} = \mathbf{M}\mathbf{R}^{-1}$. Note that this procedure only works if $\mathbf{M}$ has rank n.

Computational profile. Computing $\mathbf{G}$ and $\mathbf{R}$ in CholeskyQR can be carried out by the BLAS function `SYRK` and the LAPACK function `POTRF`, respectively. The Q-factor can be formed explicitly with the Level 3 BLAS function `TRSM`. The *flop count* for CholeskyQR using these three functions is roughly $2mn^2 + n^3/3$ [ADO94, Page 120], which is close to the $2mn^2 - (2/3)n^3$ flop count of Householder QR computed by LAPACK's `GEQRF` function [ADO94, Page 122] when $m \geq n$.

Notably, CholeskyQR returns an explicit Q-factor, while `GEQRF` outputs an implicit representation of its Q-factor. The implicit representation is preferable for manipulating the full $m \times m$ orthogonal operator in a "non-economic" QR decomposition. Nevertheless, many common numerical methods, such as subspace projection methods or singular value decomposition of tall matrices, require computing the Q-factor explicitly. Translating `GEQRF`'s implicit factor into an explicit form (using LAPACK's `ORGQR`) takes an additional $2mn^2 - (2/3)n^3$ flops [ADO94, Page 122]. Consequently, when $r := m/n > 1$, CholeskyQR requires a fraction $(6r + 1)/(12r + 4)$ of the flops of the traditional method of obtaining a fully explicit QR factorization.

Limitations. Despite its simplicity and speed, CholeskyQR is rarely used in practice. One clear shortcoming is that it fails if the computed Gram matrix is numerically rank-deficient, which can happen even if the input matrix is numerically full-rank. More generally, if $\kappa(\mathbf{M})$ denotes the condition number of $\mathbf{M}$ in the spectral norm and u denotes the unit roundoff in working precision, then rounding errors can lead to significant *orthogonality loss*, in the sense that $\|\mathbf{Q}^*\mathbf{Q} - \mathbf{I}\|_2$ can be as large as $\mathcal{O}(\mathsf{u}\,\kappa(\mathbf{M})^2)$.

The orthogonality loss of CholeskyQR can be mitigated by a variety of methods.[2] For example, *Jacobi preconditioning* normalizes all columns of $\mathbf{M}$ to have unit norm and brings the condition number of $\mathbf{M}$ to within a factor $\sqrt{n}$ of the best-possible diagonal preconditioner [Slu69, Theorem 3.5]; similar results are available for normalizing column blocks via a block-diagonal preconditioner [Dem23]. Recently, it has been suggested to obtain a preconditioner from an LU decomposition of $\mathbf{M}$ [TOO20] or from CholeskyQR on a regularized version of $\mathbf{M}$ [FKN+20]. Another approach to mitigating orthogonality loss involves reorthogonalization (essentially running CholeskyQR twice) resulting in "CholeskyQR2" [FNYY14, YNYF15]. However, this method can still fail when $\kappa(\mathbf{M}) > \mathsf{u}^{-1/2}$. Finally, a mixed precision approach computes the Gram matrix in higher precision whereby the problematic conditioning can be better controlled [YTD15].

1.3 Randomized preconditioning for CholeskyQR

The following algorithm serves as the starting point of our work. Given the matrix $\mathbf{M}$, it sketches $\mathbf{M}$ and then uses the triangular factor of the sketch's QR decomposition as a preconditioner in CholeskyQR. Like standard (unpreconditioned) CholeskyQR, it has a hard requirement that $\mathbf{M}$ is full rank in exact arithmetic.

[2] Indeed, this paper adds CQRRPT to the list of orthogonality-loss mitigation methods, even though CQRRPT is concerned with pivoted rather than unpivoted QR.

1. Compute $\mathbf{M}^{\text{sk}} = \mathbf{S}\mathbf{M}$ using a sketching operator $\mathbf{S} \in \mathbb{R}^{d \times m}$ with $n \leq d \leq m$.
2. Compute $[\mathbf{Q}^{\text{sk}}, \mathbf{R}^{\text{sk}}] = \texttt{qr}(\mathbf{M}^{\text{sk}})$ and discard $\mathbf{Q}^{\text{sk}}$.
3. Form $\mathbf{M}^{\text{pre}} = \mathbf{M}(\mathbf{R}^{\text{sk}})^{-1}$ explicitly, using back substitution.
4. Compute $\mathbf{G} = (\mathbf{M}^{\text{sk}})^*(\mathbf{M}^{\text{sk}})$ and $\mathbf{R}^{\text{pre}} = \texttt{chol}(\mathbf{G}, \text{“upper”})$.
5. Set $\mathbf{Q} = \mathbf{M}^{\text{pre}}(\mathbf{R}^{\text{pre}})^{-1}$ and $\mathbf{R} = \mathbf{R}^{\text{pre}}\mathbf{R}^{\text{sk}}$, then return $\mathbf{Q}$ and $\mathbf{R}$.

This method was first introduced [FGL21] without the context of the broader RandNLA literature; it was first studied in detail with the RandNLA literature in mind in [Bal22] and subsequently in [HSBY23]. Notably, the possibility of first computing a sketch-orthonormal Q-factor $\mathbf{M}^{\text{pre}}$, and subsequently retrieving the ℓ_2-orthonormal Q-factor via a CholeskyQR on $\mathbf{M}^{\text{pre}}$ was originally proposed elsewhere [BG22, BG21]. Furthermore, the idea of using $\mathbf{R}^{\text{sk}}$ as the preconditioner traces back to the *sketch-and-precondition* paradigm for overdetermined least squares [RT08, AMT10, MSM14].[3] If a well-studied condition called a *subspace embedding property* holds between $\mathbf{S}$ and range($\mathbf{M}$), then $\mathbf{M}^{\text{pre}}$ will be nearly-orthonormal [DMM06, Sar06, DMMS11, Mah11].

For any reasonable sketching operator, $\mathbf{S}$, the flop count of the algorithm outlined above will be $\mathcal{O}(mn^2)$. Whether or not practical speedup is observed depends greatly on the sketching distribution and on the method used to compute the product $\mathbf{S}\mathbf{M}$. If a fast sketching operator is used with $n \leq d \leq m$, then the leading term in the resulting algorithm's flop count will be $3mn^2$, which is only mn^2 more than the standard CholeskyQR.

1.4 Randomization in QR with column pivoting

Substantial efforts have been devoted to the design of randomized algorithms for QRCP of general matrices, i.e., rectangular matrices of any aspect ratio of numbers of rows and columns. These efforts originate with independent contributions by Martinsson [Mar15] as well as Duersch and Gu [DG17], with subsequent extensions by Martinsson *et al.* [MQOHvdG17] and also by Xiao, Gu, and Langou [XGL17]. While there are many variations on these methods, they share a common structure that we outline here.

These methods take in integers (b, s) where $b > 0$, $s \geq 0$, and $n \geq b + s$. They form a sketch $\mathbf{Y} = \mathbf{S}\mathbf{M}$ with $b + s$ rows, and then proceed with a three-step iterative loop.

1. Use any QRCP method to find P_{block}, a length-b vector containing column indices for the first b pivots for the wide matrix $\mathbf{Y}$.
2. Process the tall matrix $\mathbf{M}[:, P_{\text{block}}]$ by QRCP or unpivoted QR.
3. Suitably update $\mathbf{M}$ and $\mathbf{Y}$, then return to Step 1.

The update to $\mathbf{M}$ at Step 3 can be handled by standard methods, such as those in blocked unpivoted Householder QR. The update to $\mathbf{Y}$ is more subtle. If done appropriately, then the leading term in the algorithm's flop count can match that of Householder QR [DG17].

A core limitation of these methods is that their best performance is attained with small block sizes. For example, $b = 64$ and $s = 10$ is used in both shared- and distributed-memory settings [MQOHvdG17, XGL17]. Much larger block sizes would be needed for the updating operations to be performed near the limit of machines' peak performance rates. One of our motivations for developing CQRRPT has been to provide an avenue to extend earlier randomized QRCP algorithms to block sizes on the order of thousands.

[3] In this context, $\mathbf{M}^{\text{pre}}$ was only formed implicitly and was accessed as a linear operator.

1.5 Our contributions and outline

CQRRPT takes the method from Section 1.3 and replaces the call to a QR function on $\mathbf{M}^{\text{sk}}$ with a call to a QRCP function on $\mathbf{M}^{\text{sk}}$. It uses the results from this call to QRCP to provide: the pivot indices, an estimate of $\mathbf{M}$'s numerical rank, and the information for our preconditioner in the form of an upper-triangular matrix $\mathbf{R}^{\text{sk}}$. The numerical rank estimate is needed to deal with the possibility that $\mathbf{M}$ might be rank-deficient.

CQRRPT's discovery was reported in two independent works [Bal22] and [MDM+23]. This paper is a joint effort by the authors of these works to (1) conduct thorough theoretical analysis of this algorithm, and (2) demonstrate experimental results of a high-performance implementation of the algorithm using RandBLAS and RandLAPACK.

Section 2 formally introduces CQRRPT as Algorithm 1. Our formalism emphasizes how the effect of randomness on CQRRPT's behavior can be isolated in the choice of distribution used for the sketching operator. The core operations of CQRRPT are presented in Algorithm 2, which can be analyzed as a purely deterministic algorithm. Theorem 2 establishes the correctness of CQRRPT's output, and Theorem 3 characterizes the spectrum of the preconditioned input matrix. The remaining results in the section, Theorems 4 and 5, concern the quality of the pivots obtained for the input matrix.

Section 3 concerns CQRRPT's numerical stability. Its main result, Theorem 10, says that CQRRPT can be implemented to provide numerical stability that is *unconditional* on numerical properties of the input matrix. Specifically, with an appropriate criterion for determining numerical rank, CQRRPT can produce a decomposition with relative error and orthogonality loss on the order of machine precision, as long as the unit roundoff and problem dimensions satisfy $\mathsf{u} \leq m^{-1}F(n)^{-1}$ for some low-degree polynomial $F(n)$. After outlining the proof of this result we explain how numerical rank selection can be performed in practice. Appendix B states and proves a more technical version of Theorem 10 in the form of Theorem 16.

Section 4 empirically investigates pivot quality. It shows how easy-to-compute metrics of pivot quality compare when running the LAPACK default function (`GEQP3`) versus when running CQRRPT (based on applying `GEQP3` to $\mathbf{M}^{\text{sk}}$). The results show the interaction between *coherence* – which essentially represents a condition number for the act of sampling – and parameter choices for sparse sketching operators.

Section 5 provides performance experiments with a RandLAPACK implementation of CQRRPT. The experiments use a machine with dual 24-core Xeon Gold 6248R CPUs, where Intel MKL and RandBLAS are the underlying linear algebra libraries. We first compare CQRRPT with alternative methods for both unpivoted and pivoted QR factorizations. Our algorithm handily beats competing methods for large matrices, and it has the advantage of a simpler representation of $\mathbf{Q}$. We then investigate how CQRRPT's performance might be improved. Runtime profiling results which show that computing QRCP of the sketch via `GEQP3` can exceed the cost of CholeskyQR several times over. This motivates an experiment where QRCP on the sketch is handled by HQRRP [MQOHvdG17] (one of the randomized algorithms for QRCP of general matrices described in Section 1.4) instead of `GEQP3`.

Concluding remarks are given in Section 6.

1.6 Definitions and notation

1.6.1 The mundane

Matrices appear in boldface sans-serif capital letters. The transpose of a matrix $\mathbf{X}$ is given by $\mathbf{X}^*$, its ℓ_2 condition number is $\kappa(\mathbf{X})$, and its elementwise absolute value is $|\mathbf{X}|$. Numerical

vectors appear as boldface lowercase letters, while index vectors appear as uppercase letters. The $k \times k$ identity matrix is denoted by $\mathbf{I}_k$. We enumerate components of matrices and vectors with indices starting from one, rather than starting from zero. We extract the leading k columns of $\mathbf{X}$ by writing $\mathbf{X}[:, 1{:}k]$, while its trailing $n - k$ columns are extracted by writing $\mathbf{X}[:, k+1{:}n]$. The $(i, j)^{\text{th}}$ entry of $\mathbf{X}$ is $\mathbf{X}[i, j]$. Similar conventions apply to extracting the rows of a matrix or components of a vector.

The matrix we ultimately aim to decompose is denoted by $\mathbf{M}$ and has dimensions $m \times n$. We use the letters k and ℓ as integer indices between 1 and n. Their precise meaning is determined by context; we are free to redefine them at any time.

1.6.2 Some necessary evils

Virtuous notation is readable and intuitive. Despite our best efforts, we have been unable to devise wholly virtuous notation for this manuscript. We have settled instead for notation that is readable and *internally consistent* for matrices with similar structures.

Leading columns of matrices subject to pivoting and QR. Suppose $\mathbf{X}$ is a matrix with n columns that we aim to decompose via QR after column pivoting by J (a permutation vector of $\{1, \ldots, n\}$). For each $k \in \{1, \ldots, n\}$, we define the truncated, pivoted matrix

$$\mathbf{X}_k := \mathbf{X}[:, J[1{:}k]].$$

Taking $k = n$ lets us refer to a pivoted matrix without any truncation.

We use this notation from the outset with the $m \times n$ matrix $\mathbf{X} = \mathbf{M}$ and the $d \times n$ matrix $\mathbf{X} = \mathbf{M}^{\text{sk}}$. Section 3 extends this convention to a matrix denoted by "$\mathbf{M}^{\text{pre}}$," which is decomposed via *unpivoted* QR.

Column-pivoted QR decompositions, and their factors. A QR decomposition consists of two conformable matrices: an orthonormal matrix called the *Q-factor* and an upper-trapezoidal matrix called the *R-factor*. We equip these factors with special indexing rules. Specifically, the first k columns of a Q-factor "$\mathbf{Q}$" are denoted by $\mathbf{Q}_k$, and the first k rows of an R-factor "$\mathbf{R}$" are denoted by $\mathbf{R}_k$.

This manuscript features three matrices with important QR factors: a $d \times n$ pivoted matrix $\mathbf{M}^{\text{sk}}_n$, a matrix $\mathbf{M}^{\text{pre}}$ of dimensions $m \times k$ ($k \leq n$), and an $m \times n$ pivoted matrix $\mathbf{M}_n$. The QR factors of these matrices are distinguished from one another with superscripts, or the absence thereof (e.g., $\mathbf{R}^{\text{sk}}$, $\mathbf{R}^{\text{pre}}$, and $\mathbf{R}$).

Definition 1. *Let $\mathbf{X}$ denote a rank-k matrix with n columns. A column-pivoted QR decomposition of $\mathbf{X}$ consists of QR factors $(\mathbf{Q}, \mathbf{R})$ and a length-n permutation vector J such that the pivoted matrix $\mathbf{X}_n = \mathbf{X}[:, J]$ satisfies $\mathbf{X}_n = \mathbf{Q}_k \mathbf{R}_k$.*

Partitions of R-factors. We partition R-factors with notation that is similar to the celebrated paper of Gu and Eisenstat [GE96]. Specifically, to any $k \times n$ upper-trapezoidal matrix $\mathbf{R}$ and any $\ell \leq k$, we associate the distinguished submatrices

$$\mathbf{A}_\ell = \mathbf{R}[1{:}\ell,\ 1{:}\ell], \quad \mathbf{B}_\ell = \mathbf{R}[1{:}\ell,\ \ell+1{:}n], \quad \text{and} \quad \mathbf{C}_\ell = \mathbf{R}[\ell+1{:}k,\ \ell+1{:}n].$$

For fixed ℓ, these matrices let us cleanly express $\mathbf{R}$ as a 2×2 block matrix

$$\mathbf{R} = \begin{bmatrix} \mathbf{A}_\ell & \mathbf{B}_\ell \\ & \mathbf{C}_\ell \end{bmatrix}.$$

This notation helps in analyzing our algorithm's rank-revealing properties and numerical stability. We use it with R-factors from the QR decompositions of $\mathbf{M}^{\text{sk}}_n$, $\mathbf{M}^{\text{pre}}$, and $\mathbf{M}_n$.

2 Getting to know our algorithm

Section 1.1 mentioned a two-phase approach to analysis of randomized algorithms. Here we describe CQRRPT in a way that puts this approach front and center. We do this by delineating between the full algorithm (which includes a random sampling step) and the algorithm's core (which is purely deterministic).

The full algorithm appears below. Its speed and reliability are affected decisively by the sketching distribution, and it has two parameters that control this distribution. The higher-level parameter is a *distribution family*; this is an association of matrix dimensions to probability distributions over matrices with those dimensions. The lower-level parameter, called the *sampling factor*, sets the size of the sketch in proportion to n.

Algorithm 1 CholeskyQR with randomization and pivoting for tall matrices

Required inputs. An $m \times n$ matrix $\mathbf{M}$.

Optional inputs. A distribution family $\mathscr{F}$ and a sampling factor $\gamma \geq 1$.

Outputs. QR factors of dimensions $m \times k$ and $k \times n$, and a length-n permutation vector. These comprise a column-pivoted QR decomposition of $\mathbf{M}$ in the sense of Definition 1 if and only if $\text{rank}(\mathbf{SM}) = \text{rank}(\mathbf{M})$; see Theorem 2.

1: **function** $[\mathbf{Q}, \mathbf{R}, J] = \texttt{cqrrpt}(\mathbf{M}, \gamma, \mathscr{F})$
2: If $\mathscr{F}$ is not provided, set it to a default family of sparse sketching distributions.
3: If γ is not provided, set $\gamma = 1.25$.
4: Set $d = \lceil \gamma n \rceil$, and randomly sample a $d \times m$ matrix $\mathbf{S}$ from $\mathscr{F}_{d,m}$.
5: Compute $[\mathbf{Q}, \mathbf{R}, J] = \texttt{cqrrpt\_core}(\mathbf{M}, \mathbf{S})$
6: **return**

CQRRPT's core appears in Algorithm 2. We characterize its behavior in Sections 2.1 and 2.2 using the concept of *restricted singular values*. The restricted singular values of a $d \times m$ matrix $\mathbf{S}$ on a subspace $L \subset \mathbb{R}^m$ are the singular values of $\mathbf{SU}$ where $\mathbf{U}$ is any orthonormal matrix with range L.[4] The *restricted condition number* of $\mathbf{S}$ on L, denoted $\kappa(\mathbf{S}|L)$, is the ratio of its largest to smallest restricted singular values.

Section 2.3 explains how the distribution family and sampling factor affect the probabilistic behavior of Algorithm 1. By the end of this section, it will be clear that standard results from random matrix theory can be used to probabilistically yet rigorously bound the restricted singular values of $\mathbf{S}$ on range($\mathbf{M}$). This will show that there are many ways of using CQRRPT to achieve different trade-offs between speed and reliability.

Remark 2. The analysis in this section is conducted in exact arithmetic. Such analysis may seem strange in the context of a CholeskyQR algorithm, but it plays a valuable role in plotting the course for our finite-precision analysis.

2.1 CQRRPT's deterministic core

The algorithm below relies on functions called "`qrcp`" and "`rank`." For now, we only assume that `qrcp` always returns column-pivoted QR decompositions in the sense of Definition 1. The details of `rank` become important when we consider finite-precision computations in Section 3. For now, it is just a black-box that computes the exact rank of its input.

[4] Note that these singular values do not depend on the choice of orthonormal basis.

Algorithm 2 : `cqrrpt_core`

Input: A matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$, and a sketching operator $\mathbf{S} \in \mathbb{R}^{d \times m}$ where $n \leq d \leq m$.

1: **function** `cqrrpt_core`$(\mathbf{M}, \mathbf{S})$
2: Sketch $\mathbf{M}^{\text{sk}} = \mathbf{SM}$
3: Decompose $[\mathbf{Q}^{\text{sk}}, \mathbf{R}^{\text{sk}}, J] =$ `qrcp`$(\mathbf{M}^{\text{sk}})$
// ^ we return this n-vector J in full, regardless of subsequent steps.
4: Determine $k =$ `rank`$(\mathbf{R}^{\text{sk}})$
5: Precondition $\mathbf{M}^{\text{pre}} = \mathbf{M}_k(\mathbf{A}_k^{\text{sk}})^{-1}$
// ^ recall our notation that $\mathbf{M}_k := \mathbf{M}[:, J[1{:}k]]$ and $\mathbf{A}_k^{\text{sk}} := \mathbf{R}^{\text{sk}}[1{:}k, 1{:}k]$
6: Compute $\mathbf{G} = (\mathbf{M}^{\text{pre}})^*(\mathbf{M}^{\text{pre}})$
7: Decompose $\mathbf{R}^{\text{pre}} =$ `chol`$(\mathbf{G})$
8: Set $\mathbf{Q}_k = (\mathbf{M}^{\text{pre}})(\mathbf{R}^{\text{pre}})^{-1}$
9: Undo preconditioning $\mathbf{R}_k = \mathbf{R}^{\text{pre}}\mathbf{R}_k^{\text{sk}}$
// ^ recall our notation that $\mathbf{R}_k^{\text{sk}} := \mathbf{R}^{\text{sk}}[1{:}k, :]$.
10: **return** $\mathbf{Q}_k$, $\mathbf{R}_k$, J

There are two pressing questions for Algorithm 2.

1. Under what conditions does it actually obtain a decomposition of $\mathbf{M}$?
2. Just how "safe" is its use of CholeskyQR on Lines 6 through 8?

Section 3 answers these questions in finite-precision arithmetic. Here are simpler answers under the assumption of computation in exact arithmetic. When interpreting them, it can be informative to use the fact that $\kappa(\mathbf{S}|\operatorname{range}(\mathbf{M}))$ is finite if and only if $\operatorname{rank}(\mathbf{SM}) = \operatorname{rank}(\mathbf{M})$.

Theorem 2. *If* $\kappa(\mathbf{S}|\operatorname{range}(\mathbf{M}))$ *is finite, then* $[\mathbf{Q}_k, \mathbf{R}_k, J] =$ `cqrrpt_core`$(\mathbf{M}, \mathbf{S})$ *define a column-pivoted QR decomposition of* $\mathbf{M}$ *in the sense of Definition 1.*

Proof. It is clear that $\mathbf{Q}_k$ is orthonormal and $\mathbf{R}_k$ is upper-trapezoidal. We need to show that if $\operatorname{rank}(\mathbf{SM}) = \operatorname{rank}(\mathbf{M})$ then $\mathbf{\Delta} = \mathbf{M}_n - \mathbf{Q}_k\mathbf{R}_k$ is zero. As a step towards this, note the identity $\mathbf{Q}_k\mathbf{R}_k = \mathbf{M}^{\text{pre}}\mathbf{R}_k^{\text{sk}}$, which implies $\operatorname{range}(\mathbf{\Delta}) \subset \operatorname{range}(\mathbf{M})$. Next, use the assumption that `qrcp` produces column-pivoted QR decompositions in sense of Definition 1 to find

$$\begin{aligned}\mathbf{S\Delta} &= \mathbf{SM}_n - \mathbf{SM}^{\text{pre}}\mathbf{R}_k^{\text{sk}} \\ &= \mathbf{M}_n^{\text{sk}} - \mathbf{Q}_k^{\text{sk}}\mathbf{R}_k^{\text{sk}} = \mathbf{0}.\end{aligned}$$

Since $\operatorname{rank}(\mathbf{SM}) = \operatorname{rank}(\mathbf{M})$ implies $\ker(\mathbf{S}) \cap \operatorname{range}(\mathbf{M})$ is trivial, we have $\mathbf{\Delta} = \mathbf{0}$. □

Theorem 3. *Let* $\mathbf{M}^{\text{pre}}$ *be as on Line 5 of* Algorithm 2 *for inputs* $\mathbf{M}$ *and* $\mathbf{S}$*. If* $\kappa(\mathbf{S}|\operatorname{range}(\mathbf{M}))$ *is finite, then the singular values of* $\mathbf{M}^{\text{pre}}$ *are the inverses of the restricted singular values of* $\mathbf{S}$ *on* $\operatorname{range}(\mathbf{M})$.

Proof. Let $\mathbf{U}$ be an $m \times k$ orthonormal matrix with the same range as $\mathbf{M}$. Define the set $\mathcal{U}_i = \{L \subset \mathbb{R}^k : L \text{ is a linear subspace and } \dim(L) = i\}$.

Since the singular values of a matrix are the square roots of the eigenvalues of its Gram matrix, and since the square root is monotonic, the classic min-max principle for eigenvalues of Hermitian matrices directly translates to singular values of general matrices. This

formulation of the min-max principle tells us that

$$\sigma_i\left(\mathbf{M}^{\text{pre}}\right) = \min_{X \in \mathcal{U}_i} \max_{\boldsymbol{x} \in X} \frac{\|\mathbf{M}^{\text{pre}}\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} \quad \text{and} \quad \sigma_{k-i+1}\left(\mathbf{SU}\right) = \max_{Y \in \mathcal{U}_i} \min_{\boldsymbol{y} \in Y} \frac{\|\mathbf{SU}\boldsymbol{y}\|_2}{\|\boldsymbol{y}\|_2}.$$

Our goal is to show that $\sigma_i\left(\mathbf{M}^{\text{pre}}\right) = \left(\sigma_{k-i+1}\left(\mathbf{SU}\right)\right)^{-1}$. To do this, we assume the restricted singular values of $\mathbf{S}$ on range($\mathbf{M}$) are all nonzero. This ensures that range($\mathbf{M}^{\text{pre}}$) = range($\mathbf{M}$) and subsequently that there is an invertible matrix $\mathbf{T}$ where $\mathbf{M}^{\text{pre}} = \mathbf{UT}$. We also rely on the fact that the $d \times k$ matrix $\mathbf{SM}^{\text{pre}}$ is orthonormal. Combining these observations gives a chain of identities

$$\sigma_i\left(\mathbf{M}^{\text{pre}}\right) = \min_{X \in \mathcal{U}_i} \max_{\boldsymbol{x} \in X} \frac{\|\mathbf{M}^{\text{pre}}\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} = \min_{X \in \mathcal{U}_i} \max_{\boldsymbol{x} \in X} \frac{\|\mathbf{M}^{\text{pre}}\boldsymbol{x}\|_2}{\|\mathbf{SM}^{\text{pre}}\boldsymbol{x}\|_2} = \min_{X \in \mathcal{U}_i} \max_{\boldsymbol{x} \in X} \frac{\|\mathbf{UT}\boldsymbol{x}\|_2}{\|\mathbf{SUT}\boldsymbol{x}\|_2}.$$

Then we apply a change of variables $\boldsymbol{y} = \mathbf{T}\boldsymbol{x}$ to get

$$\sigma_i\left(\mathbf{M}^{\text{pre}}\right) = \min_{X \in \mathcal{U}_i} \max_{\boldsymbol{x} \in X} \frac{\|\mathbf{T}\boldsymbol{x}\|_2}{\|\mathbf{SUT}\boldsymbol{x}\|_2} = \min_{Y \in \mathcal{U}_i} \max_{\boldsymbol{y} \in Y} \frac{\|\boldsymbol{y}\|_2}{\|\mathbf{SU}\boldsymbol{y}\|_2} = \left(\max_{Y \in \mathcal{U}_i} \min_{\boldsymbol{y} \in Y} \frac{\|\mathbf{SU}\boldsymbol{y}\|_2}{\|\boldsymbol{y}\|_2}\right)^{-1}$$

which completes the proof. □

Arithmetic complexity. The arithmetic complexity of Algorithm 2 depends on how we compute and then decompose $\mathbf{M}^{\text{sk}}$. Since there are many practical ways to handle the first of these operations, we shall simply say its cost in flops is C_{sk}. There are also many ways one might perform the second operation, but the most practical is to use the LAPACK function `GEQP3`. With this choice, the algorithm's flop count is

$$2mk^2 + mk(k+1) + 4dnk - 2k^2(d+n) + 5k^3/3 + C_{\text{sk}} \tag{1}$$

plus lower-order terms; see Appendix A.1 for a derivation of this fact.

If we plug $k = n$ into (1) and assume $d, n \in o(m)$, we see that the leading-order term in CQRRPT's flop count is $3mn^2 + C_{\text{sk}}$. This compares favorably to the $4mn^2$ flops required to compute a Householder QR decomposition and then explicitly restore the Q-factor with LAPACK's `ORGQR`. What's more, in certain applications it can suffice to represent $\mathbf{Q}$ as a composition of two elementary operators, $\mathbf{M}^{\text{pre}}$ and $(\mathbf{R}^{\text{pre}})^{-1}$. This comes with no sacrifices to the numerical stability (as will be clear from our stability analysis) yet it decreases the leading term in CQRRPT's arithmetic complexity to $2mn^2 + C_{\text{sk}}$.

Communication cost. CQRRPT lends itself well to distributed computation. Here it is prudent to choose the sketching distribution that allows for evaluation of Line 2 with the smallest possible value of d while retaining good statistical properties. This can be achieved in theory and practice with Gaussian sketching (see Section 2.3.2) and $\gamma = d/n \in [1.25, 5]$; see [MDM+23, §A.1.1].

Consider a popular setting where $\mathbf{M}$ is distributed block row-wise across $p \leq m/d$ processors, each of which has at least $2dn + n^2$ words of memory. Here we implement Line 2 with local multiplications of blocks of $\mathbf{M}$ with the corresponding blocks of columns of $\mathbf{S}$ (using dn words of memory) and summing the contributions with an all-reduce operation (using another dn words of memory). From there, each processor performs QRCP on $\mathbf{M}^{\text{sk}}$, recovers $\mathbf{A}_k^{\text{sk}}$ from $\mathbf{R}^{\text{sk}}$, and applies its inverse to the corresponding local block of $\mathbf{M}$. When using the classical binomial tree version of all-reduce, the overall computation of $\mathbf{M}^{\text{pre}}$ requires one global synchronization, $2\log_2 p$ messages, $2dn\log_2 p$ communication volume, and two data

passes. Adding the cost of the CholeskyQR step [ND15], we get the total cost of CQRRPT: two global synchronizations, $4\log_2 p$ messages, $(2dn+n^2)\log_2 p$ communication volume, and three data passes. Notably, if we used the recursive-halving version of all-reduce [TRG05], the communication volume can be improved to a mere $2dn + n^2$.

2.2 How `cqrrpt_core` inherits rank-revealing properties

Here we explain how CQRRPT inherits pivot quality properties from its underlying `qrcp` function. To describe these properties, we speak in terms of a matrix $\mathbf{X}$ consisting of n columns and at least as many rows,[5] along with its decomposition $[\mathbf{Q}, \mathbf{R}, J] = \texttt{qrcp}(\mathbf{X})$. We set $k := \text{rank}(\mathbf{X})$, and for any $\ell \leq k$ we use $(\mathbf{A}_\ell, \mathbf{B}_\ell, \mathbf{C}_\ell)$ to denote submatrices of $\mathbf{R}$ using the conventions established in Section 1.6.

We first consider the *rank revealing QR* (RRQR) property. This concerns how well the spectrum of $\mathbf{A}_\ell$ approximates the leading singular values of $\mathbf{X}$, and how well the spectrum of $\mathbf{C}_\ell$ approximates the trailing singular values of $\mathbf{X}$. When $\mathbf{X}$ is fixed, we can describe the approximation quality by a sequence of coefficients $f_1, \ldots, f_k$, all at least unity. Formally, `qrcp` has the RRQR property for $\mathbf{X}$ with coefficients $(f_\ell)_{\ell=1}^k$ if, for all $\ell \leq k$, we have

$$\sigma_j(\mathbf{A}_\ell) \geq \frac{\sigma_j(\mathbf{X})}{f_\ell} \quad \text{for all} \quad j \leq \ell, \tag{2a}$$

and

$$\sigma_j(\mathbf{C}_\ell) \leq f_\ell \, \sigma_{\ell+j}(\mathbf{X}) \quad \text{for all} \quad j \leq k - \ell. \tag{2b}$$

Theorem 4. *Let $c = \kappa(\mathbf{S}|\,\text{range}(\mathbf{M}))$. If* `qrcp` *satisfies the RRQR property for $\mathbf{SM}$ with coefficients $(f_\ell)_{\ell=1}^k$, then* `cqrrpt_core`$(\cdot, \mathbf{S})$ *satisfies the RRQR property for $\mathbf{M}$ with coefficients $(cf_\ell)_{\ell=1}^k$.*

It is common to ask that a QRCP algorithm admit a *function* f where it ensures the RRQR property with coefficients $(f(\ell, n))_{\ell=1}^k$ for any rank-k matrix with n columns. This is a significant request. Indeed, QRCP with the max-norm pivot rule does not satisfy the resulting requirements for *any* function f, as can be seen by taking a limit of Kahan matrices of fixed dimension [GE96, Example 1]. Still, there are several QRCP algorithms that can ensure the RRQR property, particularly where $f(\ell, n)$ is bounded by a low-degree polynomial in ℓ and n. Theorem 4 shows that if CQRRPT uses such an algorithm, then it will satisfy a nearly identical RRQR property.

Next, we consider a *strong RRQR* property. This concerns our ability to use $\mathbf{R}$ to find a well-conditioned basis for an approximate null space of $\mathbf{X}_n$. The particular basis is the columns of the block matrix $\mathbf{Y} = [\mathbf{A}_\ell^{-1}\mathbf{B}_\ell; -\mathbf{I}]$, which satisfies $\|\mathbf{X}_n\mathbf{Y}\| = \|\mathbf{C}_\ell\|$ in every unitarily invariant norm. For our purposes, we say that `qrcp` satisfies the strong RRQR property for $\mathbf{X}$ with coefficients $(f_\ell)_{\ell=1}^k$ and $(g_\ell)_{\ell=1}^k$ when the former coefficients satisfy (2a)-(2b) and the latter coefficients satisfy

$$\|\mathbf{A}_\ell^{-1}\mathbf{B}_\ell\|_2 \leq g_\ell \tag{3}$$

for all $\ell \leq k$.

Theorem 5. *Let $c = \kappa(\mathbf{S}|\,\text{range}(\mathbf{M}))$. If* `qrcp` *satisfies the strong RRQR property for $\mathbf{SM}$ with coefficients $(f_\ell)_{\ell=1}^k$ and $(g_\ell)_{\ell=1}^k$, then* `cqrrpt_core`$(\cdot, \mathbf{S})$ *satisfies the strong RRQR property for $\mathbf{M}$ with coefficients $(cf_\ell)_{\ell=1}^k$ and $(g_\ell + cf_\ell^2)_{\ell=1}^k$.*

[5] Which we might take as $\mathbf{X} = \mathbf{M}$ or $\mathbf{X} = \mathbf{M}^{\text{sk}}$, depending on context.

As with RRQR, we can ask that a QRCP algorithm be associated with *functions* f and g for which $(f(\ell,n))_{\ell=1}^k$ and $(g(\ell,n))_{\ell=1}^k$ provide strong RRQR coefficients for any rank-k matrix with n columns. The first algorithm that could ensure this was introduced by Gu and Eisenstat [GE96].[6] Setting the tuning parameter of their algorithm to two leads to strong RRQR coefficients

$$f(\ell,n) = \sqrt{1+4\ell(n-\ell)} \quad \text{and} \quad g(\ell,n) = 2\sqrt{\ell(n-\ell)},$$

with a runtime of $\mathcal{O}(dn^2 \log n)$ for a $d \times n$ matrix with $d \geq n$. Theorem 5 shows that if CQRRPT uses this algorithm for QRCP on SM, then it will enjoy an analogous strong RRQR property with some increase to g.

2.3 Probabilistic aspects of CQRRPT

From the perspective of Algorithm 1, the sketching operator S is sampled at random from a probability distribution, and so anything that S affects in Algorithm 2 becomes a random variable with some induced distribution. In particular, the probabilistic behavior of Algorithm 1 is determined by the induced distribution for the condition number of M^{pre}. Therefore to understand CQRRPT's behavior, we must explore the following question.

> How can we bound the probability that $\kappa(\mathsf{M}^{\text{pre}})$ stays within a prescribed limit, *no matter the matrix* M?

The answer to this question depends greatly on the distribution family $\mathscr{F}$ and the sampling factor γ used in Algorithm 1. Here we give a range of answers based on different choices for these values.

2.3.1 Sketching distribution families: examples and intuition

For a fixed distribution family $\mathscr{F}$, we use "$\mathscr{F}_{d,m}$" for the distribution in $\mathscr{F}$ over $d \times m$ matrices. Here are the structures of samples from $\mathscr{F}_{d,m}$ for three prominent distribution families.

- *Gaussian matrices.* The entries of S are iid Gaussian random variables with mean zero and variance $1/d$.
- *SASOs* (short-axis-sparse operators). The columns of S are independent. Each column has exactly s nonzeros (for a tuning parameter s) whose locations are chosen uniformly at random and whose values are chosen to be $\pm 1/\sqrt{s}$ with equal probability.[7] In practice, it is common to keep ℓ between one and eight, even when d is on the order of tens of thousands.
- *SRHTs* (subsampled randomized Hadamard transforms). These are ordinarily only defined when $w = \log_2 m$ is an integer and are extended to general m by zero-padding the input. To absorb the zero-padding into the SRHT, define a $2^{\lceil w \rceil} \times m$ matrix D whose upper $m \times m$ block is a diagonal matrix populated by independent Rademacher random variables and whose lower block is all zeros. An SRHT is then a composition of three operators: $\mathsf{S} = \sqrt{m/d} \cdot \mathsf{\Pi}[1{:}d, :]\mathsf{HD}$, where H is a Hadamard transform of order $2^{\lceil w \rceil}$ and $\mathsf{\Pi}$ is a random permutation matrix.

[6] The Gu-Eisenstat algorithm actually bounds $(\mathsf{A}_\ell)^{-1}\mathsf{B}_\ell$ *elementwise*. The elementwise bounds readily imply the spectral-norm bounds that we require.

[7] For the etymology of these sketching operators, see [MDM+23].

These families share key properties. First, if $\mathbf{S}$ is a random matrix sampled from $\mathscr{F}_{d,m}$, then its expected covariance matrix satisfies $\mathbb{E}[\mathbf{S}^*\mathbf{S}] = \mathbf{I}_m$. Second, if $\mathbf{U}$ is an $m \times k$ orthonormal matrix and d/k is sufficiently large, then the sketched Gram matrix $\mathbf{U}^*\mathbf{S}^*\mathbf{S}\mathbf{U}$ concentrates strongly around $\mathbb{E}[\mathbf{U}^*\mathbf{S}^*\mathbf{S}\mathbf{U}] = \mathbf{I}_k$. To rephrase this second property: $\mathbf{S}\mathbf{U}$ should concentrate strongly around the real Stiefel manifold $M_{d,k}$ of $d \times k$ orthonormal matrices.

To see the usefulness of these properties, consider Theorem 3. If $\mathrm{range}(\mathbf{U}) = \mathrm{range}(\mathbf{M})$ and $\mathbf{S}\mathbf{U}$ has rank-k, then the singular values of $\mathbf{M}^{\mathrm{pre}}$ will equal those of the Moore-Penrose pseudo-inverse of $\mathbf{S}\mathbf{U}$. Therefore the desirable property of $\mathbf{M}^{\mathrm{pre}}$ concentrating around $M_{m,k}$ is equivalent to $\mathbf{S}\mathbf{U}$ concentrating around $M_{d,k}$.

2.3.2 Oblivious subspace embeddings

Let $\mathbf{U}$ be a matrix whose columns are an orthonormal basis for a linear subspace $L \subset \mathbb{R}^m$, and let $\mathbf{S}$ be a $d \times m$ matrix.

Definition 6. *We call $\mathbf{S}$ a subspace embedding for L with distortion $\delta \in [0,1]$ if*

$$1 - \delta \leq \sigma_{\min}(\mathbf{S}\mathbf{U})^2 \quad \text{and} \quad \sigma_{\max}(\mathbf{S}\mathbf{U})^2 \leq 1 + \delta. \tag{4}$$

Such a matrix is also called a δ-embedding for L.

Note that we always have $\kappa(\mathbf{S}|L) \leq \sqrt{(1+\delta)/(1-\delta)}$ when $\mathbf{S}$ is a δ-embedding for L. There is a long history of using subspace embeddings in randomized algorithms for least squares problems [Sar06, DMM06, DMMS11], including as a tool for finding preconditioners to solve least squares problems to higher accuracy [RT08, AMT10, MSM14].

Theory is available on how to choose d so that a sample from $\mathscr{F}_{d,m}$ will be a δ-embedding for any fixed linear subspace of a given dimension with high probability. We can use this theory to select d in practice for very well-behaved distribution families. For example, here is a representative result for Gaussians.

Remark 3. We state the following results with linear subspaces of dimension "n," since the dimension of $\mathrm{range}(\mathbf{M})$ is never larger than n.

Theorem 7. *Fix an n-dimensional linear subspace $L \subset \mathbb{R}^m$, along with some $\delta \in (0,1)$ and $\tau > 0$. If $\mathbf{S}$ is a $d \times m$ Gaussian operator with*

$$\frac{d}{n} \geq \left(\frac{1+\tau}{(1+\delta)^{1/2} - 1} \right)^2,$$

then it will be a δ-embedding for L with probability at least $1 - 2\exp(-n\tau^2/2)$.

Proof. By rotational invariance of the Gaussian distribution, we can take $L = \mathrm{range}(\mathbf{U})$ for $\mathbf{U} = \mathbf{I}_m[:, 1{:}n]$. The claim then follows from [MT20, Theorem 8.4]. To see how, set $\theta = (1+\delta)^{1/2} - 1$. If d is chosen in the way indicated above, then setting $t := \tau\theta/(1+\tau)$ in the statement of [MT20, Theorem 8.4] ensures that $1 - \theta \leq \sigma_{\min}(\mathbf{S}\mathbf{U})$ and $\sigma_{\max}(\mathbf{S}\mathbf{U}) \leq 1 + \theta$ hold with probability at least $1 - 2\exp(-n\tau^2/2)$. From there, simply note that $1 - \delta \leq (1-\theta)^2$ and $(1+\theta)^2 = 1 + \delta$ to find that $1 - \delta \leq \sigma_i(\mathbf{S}\mathbf{U})^2 \leq 1 + \delta$ for all i. □

We give results for SASOs and SRHTs below. These are often of interest since they are algorithmically more attractive than Gaussian matrices. However, the available bounds are quite pessimistic (they suggest that one take γ proportional to $\log n$, but taking $\gamma = 1.25$ suffices for practical purposes when n is large).

Theorem 8. [Coh16, Theorem 4.2] *Fix an n-dimensional linear subspace $L \subset \mathbb{R}^m$, and any $B > 2$, $t \geq 1$, and $\delta < 1/2$. There are absolute constants c_1, c_2 where, upon taking*

$$\frac{d}{n} \geq c_1 t(B \log B)\frac{\log n}{\delta^2} \quad \text{and} \quad s \geq c_2 t(\log B)\frac{\log n}{\delta},$$

sampling a $d \times m$ SASO with s nonzeros per column provides a δ-embedding for L with probability at least $1 - B^{-t}$.

Theorem 9. [BN19, Proposition 3.9] *Let $L \subset \mathbb{R}^m$ be a linear subspace of dimension n. Let $\delta, p \in (0, 1)$. If $\mathbf{S}$ is a $d \times m$ SRHT with $d \leq m$ and*

$$\frac{d}{n} \geq 2(\delta^2 - \delta^3/3)^{-1}\left(1 + \sqrt{\frac{8\log(6m/p)}{n}}\right)^2 \log(3n/p),$$

then it is a δ-embedding for L with probability at least $1 - p$.

We note that the analysis in [BN19, Proposition 3.9], cited above, builds on more fundamental results from [Tro11, BG13].

3 Numerical stability

This section analyzes `cqrrpt_core` (Algorithm 2) under the assumption that all algebraic operations in it and in its subroutines are performed in finite precision arithmetic. Suppose for concreteness that we have

$$[\mathbf{Q}_k, \mathbf{R}_k, J] = \texttt{cqrrpt\_core}(\mathbf{M}, \mathbf{S}), \tag{5}$$

where $\mathbf{Q}_k$ is $m \times k$ and $\mathbf{R}_k$ is $k \times n$ and upper-triangular. In these terms, our analysis is concerned with bounding the *reconstruction error* $\|\mathbf{M}_n - \mathbf{Q}_k\mathbf{R}_k\|_{\mathrm{F}}/\|\mathbf{M}\|_{\mathrm{F}}$ and the *orthogonality loss* $\|\mathbf{Q}_k^*\mathbf{Q}_k - \mathbf{I}_k\|_2$.

Proving bounds on these quantities requires assumptions on the `qrcp` and `rank` subroutines in Algorithm 2. So far we have said very little about these points because they are ultimately design questions that have no single answer. In this section, our goal is to show that `qrcp` and `rank` can be implemented so that upon conditioning on $\mathbf{S}$ being a δ-embedding for range($\mathbf{M}$) (for some $\delta \leq 1/2$), the reconstruction error and orthogonality loss are bounded above by low-degree polynomials in (m, n). Such a result will directly imply that CQRRPT's properties introduced in Section 2 and proven in Appendix A are preserved under finite precision arithmetic.

New notation. For purposes of exposition in this section only, we adopt notation where scalars x and y are said to satisfy $x \lesssim y$ if $x \leq cy + G(n)\mathsf{u}$, where c is a constant close to 1, $G(n)$ is a low-degree in the small matrix dimension n, and u is a unit roundoff.[8]

Up until now the $m \times k$ and $k \times k$ matrices $\mathbf{M}^{\mathrm{pre}}$ and $\mathbf{R}^{\mathrm{pre}}$ have been referred to without notational dependence on k. Moving forward, we use $\mathbf{M}_k^{\mathrm{pre}}$ and $\mathbf{R}_k^{\mathrm{pre}}$ for these matrices. We can select submatrices from them using notation consistent with Section 1.6. For example, we can speak of a parameter $\ell < k$ and use $\mathbf{M}_\ell^{\mathrm{pre}}$ in reference to the first ℓ columns of $\mathbf{M}_k^{\mathrm{pre}}$. Similarly, we can use $\mathbf{A}_\ell^{\mathrm{pre}}$ to denote the leading $\ell \times \ell$ submatrix of $\mathbf{R}_k^{\mathrm{pre}}$.

[8] The polynomials we use will never exceed degree four, and are typically degree one or two.

3.1 Summary

3.1.1 The challenge

We begin by emphasizing that since we want to decide how `rank` should be implemented, the value $k = \operatorname{rank}(\mathbf{M}^{\mathrm{sk}})$ on Line 4 of `cqrrpt_core` is really something we *choose*.

That established, let us see how k affects reconstruction error and orthogonality loss. In the tradition of adding zero and applying the triangle inequality, one can obtain the following bound on reconstruction error committed in the preconditioning step:

$$\left\|\mathbf{M}_n - \mathbf{M}^{\mathrm{pre}}\mathbf{R}_k^{\mathrm{sk}}\right\|_{\mathrm{F}} \leq \underbrace{\left\|\mathbf{M}_n - \mathbf{M}_k(\mathbf{A}_k^{\mathrm{sk}})^{-1}\mathbf{R}_k^{\mathrm{sk}}\right\|_{\mathrm{F}}}_{\text{truncation error}} + \left\|(\mathbf{M}_k - \mathbf{M}^{\mathrm{pre}}\mathbf{A}_k^{\mathrm{sk}})(\mathbf{A}_k^{\mathrm{sk}})^{-1}\mathbf{R}_k^{\mathrm{sk}}\right\|_{\mathrm{F}}. \tag{6}$$

Of the two terms in this upper bound, the *truncation error* is more opaque. To better understand it, suppose δ is the distortion of $\mathbf{S}$ for the range of $\mathbf{M}$. We claim that under mild assumptions for the accuracy of operations on Lines 2 and 3, and an assumption that $\mathbf{A}_k^{\mathrm{sk}}$ is not too ill-conditioned, one can bound

$$\left\|\mathbf{M}_n - \mathbf{M}_k(\mathbf{A}_k^{\mathrm{sk}})^{-1}\mathbf{R}_k^{\mathrm{sk}}\right\|_{\mathrm{F}} / \|\mathbf{M}\|_{\mathrm{F}} \lesssim \sqrt{1+\delta}\left\|\mathbf{C}_k^{\mathrm{sk}}\right\|_{\mathrm{F}} / \left\|\mathbf{M}^{\mathrm{sk}}\right\|_{\mathrm{F}}. \tag{7}$$

Since $\|\mathbf{C}_k^{\mathrm{sk}}\|_{\mathrm{F}}$ is decreasing with k, this bound suggests that k should be large to keep reconstruction error under control.

Unfortunately, if $\mathbf{M}$ is ill-conditioned, then choosing k too large can pose severe problems. The basic reason for this stems from the fact that $\kappa(\mathbf{A}_n^{\mathrm{sk}})$ can be as large as $\kappa(\mathbf{SM})$, which in turn can be as large as $\sqrt{\frac{1+\delta}{1-\delta}}\,\kappa(\mathbf{M})$. This suggests that $\mathbf{A}_k^{\mathrm{sk}}$ can be ill-conditioned when k is large, even when δ is small. This matters since larger condition numbers for $\mathbf{A}_k^{\mathrm{sk}}$ risk larger rounding errors at the step when we form the $m \times k$ matrix $\mathbf{M}^{\mathrm{pre}}$. If this step is performed inaccurately, then $\kappa(\mathbf{M}^{\mathrm{pre}})$ might be far from one even if δ is small, which risks orthogonality loss in $\mathbf{Q}_k$ that cannot be controlled by our choice of sketching distribution for Algorithm 1.

3.1.2 A high-level result

The tension described above raises an important question. Can we be certain that there *exists* a truncation index k so that both reconstruction error and orthogonality loss are kept near machine precision? The following result says the answer is yes, provided the QRCP of $\mathbf{M}^{\mathrm{sk}}$ is (sufficiently) strongly rank-revealing.

Theorem 10 (Simplified version of Theorem 16 from Appendix B)**.** *Consider Algorithm 2 where Line 5 is executed with unit roundoff* u*, and other lines are executed with unit roundoff* $\tilde{\mathsf{u}}$*. Assume that the* `qrcp` *subroutine at Line 3 produces* $\mathbf{Q}^{\mathrm{sk}}$ *and* $\mathbf{R}^{\mathrm{sk}}$ *by a pivoted Householder QR process or pivoted Givens QR process. Additionally, assume its pivots satisfy the strong rank-revealing properties* (2a)*,* (2b)*, and* (3) *with coefficients* $f_\ell, g_\ell \leq 2\sqrt{n\ell}$ *when* $\mathbf{X} = \mathbf{M}^{\mathrm{sk}}$*. Finally, suppose* $\mathbf{S}$ *is a* δ*-embedding for* $\operatorname{range}(\mathbf{M})$ *with* $\delta \leq 1/2$*.*

There exist low-degree polynomials $G_1, \ldots, G_5$ *(that have no dependence on* $\mathbf{M}$ *or* $\mathbf{S}$*) and a truncation index* k *such that if* $\tilde{\mathsf{u}} \leq m^{-1}G_1(n,d)^{-1}\mathsf{u}$ *and* $\mathsf{u} \leq G_2(n)^{-1}$*, then*

$$\left\|\mathbf{M}_n - \mathbf{M}^{\mathrm{pre}}\mathbf{R}_k^{\mathrm{sk}}\right\|_{\mathrm{F}} \leq G_3(n)\mathsf{u}\,\|\mathbf{M}\|_{\mathrm{F}} \quad \textit{and} \quad \kappa\left(\mathbf{M}^{\mathrm{pre}}\right) \leq 1.8. \tag{8}$$

Furthermore, as a direct consequence of Eq. (8)*, we have*

$$\|\mathbf{M}_n - \mathbf{Q}_k\mathbf{R}_k\|_{\mathrm{F}} \leq G_4(n)\mathsf{u}\,\|\mathbf{M}\|_{\mathrm{F}} \quad \textit{and} \quad \left\|\mathbf{Q}_k^*\mathbf{Q}_k - \mathbf{I}\right\|_2 \leq G_5(n)\mathsf{u}. \tag{9}$$

We have chosen a computational model with two unit roundoffs to highlight an important property of the preconditioner – that the dominant Line 5 can be performed with a unit roundoff not constrained by m. This property may have significant implications for multi- and low-precision arithmetic architectures. Clearly, the result can also be used in the computational model with a single roundoff.

We note that Theorem 10 is an *existential* result for a suitable truncation rank k. Our proof of this result includes a method for how such k can be identified efficiently, under the stated assumptions on the unit roundoffs. However, in practice, the unit roundoff is usually a constant that does not change with the matrix dimensions. This creates a need for a separate and more practical way to choose the truncation rank, which is a topic we address in Section 3.3.

3.2 Proof sketch for Theorem 10

The following lemma, which we prove in Appendix B.1 by straightforward methods, shows that Eq. (9) in Theorem 10 directly follows from Eq. (8).

Lemma 11. *Consider Lines 5 to 7 of Algorithm 2 where the computations are performed with unit roundoff* $\tilde{\mathsf{u}} \leq 0.00022m^{-1}n^{-1}$. *If* $\mathbf{M}_k^{\text{pre}}$ *and* $\mathbf{R}_k^{\text{sk}}$ *satisfy*

$$\left\|\mathbf{M}_n - \mathbf{M}_k^{\text{pre}}\mathbf{R}_k^{\text{sk}}\right\|_2 \leq 0.01\,\|\mathbf{M}\|_2 \;\text{ and }\; \kappa(\mathbf{M}_k^{\text{pre}}) \leq 6, \tag{10}$$

then the output factors $\mathbf{Q}_k$ *and* $\mathbf{R}_k$ *satisfy*

$$\|\mathbf{M}_n - \mathbf{Q}_k\mathbf{R}_k\|_{\text{F}} \leq \left\|\mathbf{M}_n - \mathbf{M}_k^{\text{pre}}\mathbf{R}_k^{\text{sk}}\right\|_{\text{F}} + 60n^2\tilde{\mathsf{u}}\,\|\mathbf{M}\|_2\,, \tag{11}$$

$$\|\mathbf{Q}_k^*\mathbf{Q}_k - \mathbf{I}\|_2 \leq 180mn\tilde{\mathsf{u}}. \tag{12}$$

Taking Lemma 11 as given, the question becomes how to choose k so that Eq. (8) holds. We call this the *preconditioner stability question*. In brief, the main idea is to choose k so that

$$\|\mathbf{C}_k^{\text{sk}}\|_{\text{F}}/\|\mathbf{R}^{\text{sk}}\|_2 \leq G_0(n)\mathsf{u} \leq \|\mathbf{C}_{k-1}^{\text{sk}}\|_{\text{F}}/\|\mathbf{R}^{\text{sk}}\|_2, \tag{13}$$

for a low-degree polynomial $G_0(n)$ that does not depend on $\mathbf{M}$ or $\mathbf{S}$. The specific polynomial $G_0(n)$ sufficient to ensure Eq. (8) is given in Theorem 16, which is stated and proven in Appendix B.2.2. In what follows, we outline the key steps in the argument for proving that theorem, using the same notation as in our formal proofs.

Truncation ensures full numerical rank. Choosing k to satisfy (13) ensures that $\mathbf{A}_k^{\text{sk}}$ is numerically full-rank. Assuming $\mathbf{R}^{\text{sk}}$ comes from a column-pivoted QR decomposition of $\mathbf{M}^{\text{sk}}$ with the theorem's stated strong RRQR properties, such k provides for the existence of low-degree polynomials G_6 and G_7 where

$$\kappa(\mathbf{A}_k^{\text{sk}}) = \frac{\|\mathbf{A}_k^{\text{sk}}\|_2}{\sigma_{\min}\left(\mathbf{A}_k^{\text{sk}}\right)} \leq \frac{\|\mathbf{R}^{\text{sk}}\|_2}{\|\mathbf{C}_{k-1}^{\text{sk}}\|_{\text{F}}} G_6(n) \leq G_7(n)\mathsf{u}^{-1}.$$

This bound can be used to show that $\mathbf{M}_k$ is also numerically full-rank, in that

$$\kappa(\mathbf{M}_k) \leq \sqrt{\frac{1+\delta}{1-\delta}}\,\kappa(\mathbf{S}\mathbf{M}_k) \lesssim \kappa(\mathbf{A}_k^{\text{sk}}) \leq G_7(n)\mathsf{u}^{-1}. \tag{14}$$

The first inequality in (14) holds due to the δ-embedding property of $\mathbf{S}$. The second inequality can be shown through three steps: $\kappa(\mathbf{A}_k^{\text{sk}}) \gtrsim \kappa(\mathbf{Q}_k^{\text{sk}}\mathbf{A}_k^{\text{sk}}) \gtrsim \kappa(\mathbf{M}_k^{\text{sk}}) \gtrsim \kappa(\mathbf{S}\mathbf{M}_k)$, employing standard rounding bounds for matrix operations, with the fact that the left-hand-side matrix is numerically full rank (in steps two and three).

Preconditioning and reconstruction error for leading columns. The next step in the proof is to look at $\mathbf{M}_k^{\mathrm{pre}}\mathbf{A}_k^{\mathrm{sk}}$ as an unpivoted "sketched CholeskyQR" decomposition of $\mathbf{M}_k$, in the sense of [Bal22]. According to [Bal22, Theorem 5.2], if $\kappa(\mathbf{M}_k) \lesssim G_7(n)\mathsf{u}^{-1}$, then there is a low-degree polynomial G_8 for which

$$\|\mathbf{M}_k - \mathbf{M}_k^{\mathrm{pre}}\mathbf{A}_k^{\mathrm{sk}}\|_{\mathrm{F}} \leq G_8(n)\mathsf{u}\|\mathbf{M}_k\|_{\mathrm{F}} \quad \text{and} \quad \kappa(\mathbf{M}_k^{\mathrm{pre}}) \lesssim \sqrt{\frac{1+\delta}{1-\delta}}. \tag{15}$$

This establishes the condition number bound needed in (8). It also controls reconstruction error for the first k columns of $\mathbf{M}_n$.

Reconstruction error for trailing columns. To establish (8), it remains to bound the reconstruction error of the trailing $n-k$ columns of $\mathbf{M}$. Toward this end, we adopt the notation that

$$\overline{\mathbf{M}}_k := \mathbf{M}[:, J[k+1{:}n]] \quad \text{and} \quad \overline{\mathbf{M}}_k^{\mathrm{sk}} := \mathbf{M}^{\mathrm{sk}}[:, J[k+1{:}n]].$$

By using the reconstruction error bound for $\mathbf{M}_k$ from Eq. (15), the δ-embedding property, standard bounds for matrix operations, and the reconstruction error bound for $\mathbf{M}_k^{\mathrm{sk}}$, one finds that

$$\begin{aligned}\frac{\|\overline{\mathbf{M}}_k - \mathbf{M}_k^{\mathrm{pre}}\mathbf{B}_k^{\mathrm{sk}}\|_{\mathrm{F}}}{\|\mathbf{M}\|_{\mathrm{F}}} &\lesssim \frac{\|\overline{\mathbf{M}}_k - \mathbf{M}_k(\mathbf{A}_k^{\mathrm{sk}})^{-1}\mathbf{B}_k^{\mathrm{sk}}\|_{\mathrm{F}}}{\|\mathbf{M}\|_{\mathrm{F}}} \leq \sqrt{\frac{1+\delta}{1-\delta}}\frac{\|\mathbf{S}\overline{\mathbf{M}}_k - \mathbf{S}\mathbf{M}_k(\mathbf{A}_k^{\mathrm{sk}})^{-1}\mathbf{B}_k^{\mathrm{sk}}\|_{\mathrm{F}}}{\|\mathbf{S}\mathbf{M}\|_{\mathrm{F}}} \\ &\lesssim \sqrt{\frac{1+\delta}{1-\delta}}\frac{\|\overline{\mathbf{M}}_k^{\mathrm{sk}} - \mathbf{M}_k^{\mathrm{sk}}(\mathbf{A}_k^{\mathrm{sk}})^{-1}\mathbf{B}_k^{\mathrm{sk}}\|_{\mathrm{F}}}{\|\mathbf{M}^{\mathrm{sk}}\|_{\mathrm{F}}} \lesssim \sqrt{\frac{1+\delta}{1-\delta}}\frac{\|\overline{\mathbf{M}}_k^{\mathrm{sk}} - \mathbf{Q}_k^{\mathrm{sk}}\mathbf{B}_k^{\mathrm{sk}}\|_{\mathrm{F}}}{\|\mathbf{M}^{\mathrm{sk}}\|_{\mathrm{F}}}.\end{aligned} \tag{16}$$

Notably, the strong RRQR property of $\mathbf{R}_k$ plays a key role in Eq. (16). It ensures that the terms of the form $\|\mathbf{E}(\mathbf{A}_k^{\mathrm{sk}})^{-1}\mathbf{B}_k^{\mathrm{sk}}\|_{\mathrm{F}}$, for some error matrices such as $\mathbf{E} = \mathbf{M}_k - \mathbf{M}_k^{\mathrm{pre}}\mathbf{A}_k^{\mathrm{sk}}$, are bounded by $g_k\|\mathbf{E}\|_{\mathrm{F}}$, where $g_k \leq 2\sqrt{kn}$.

We combine Eq. (16) with the stability of the `qrcp` routine and the criterion Eq. (13) used for selecting the truncation index k to get

$$\frac{\|\overline{\mathbf{M}}_k - \mathbf{M}_k^{\mathrm{pre}}\mathbf{B}_k^{\mathrm{sk}}\|_{\mathrm{F}}}{\|\mathbf{M}\|_{\mathrm{F}}} \lesssim \sqrt{\frac{1+\delta}{1-\delta}}\frac{\|\overline{\mathbf{M}}_k^{\mathrm{sk}} - \mathbf{Q}_k^{\mathrm{sk}}\mathbf{B}_k^{\mathrm{sk}}\|_{\mathrm{F}}}{\|\mathbf{M}^{\mathrm{sk}}\|_{\mathrm{F}}} \lesssim \sqrt{\frac{1+\delta}{1-\delta}}\frac{\|\mathbf{C}_k^{\mathrm{sk}}\|_{\mathrm{F}}}{\|\mathbf{R}^{\mathrm{sk}}\|_{\mathrm{F}}} \leq G_9(n)\mathsf{u}. \tag{17}$$

This fulfills our need for a reconstruction error bound for the trailing columns of $\mathbf{M}$.

3.3 Determining numerical rank in practice

We suggest a flexible two-stage approach to numerical rank estimation. The idea is that since orthogonality loss increases with truncation rank and reconstruction error (typically) decreases with truncation rank, it is reasonable to choose the largest rank where some estimate for the orthogonality loss is below a specified tolerance (say, $\epsilon_{\mathrm{tol}} = 100\mathsf{u}$).

Our first stage begins by finding a crude upper bound on the numerical rank of $\mathbf{R}^{\mathrm{sk}}$. For example, one can set

$$k_o = \min\{\ell \,:\, \|\mathbf{C}_\ell^{\mathrm{sk}}\|_{\mathrm{F}} \leq \mathsf{u}s\}, \tag{18a}$$

where s is the maximum entry of $|\mathbf{R}^{\mathrm{sk}}|$. This bound is cheap to compute and ensures $\|\mathbf{C}_{k_o}^{\mathrm{sk}}\|_2 \leq \mathsf{u}\|\mathbf{R}^{\mathrm{sk}}\|_2$. The rest of the first stage consists of forming the preconditioned matrix $\mathbf{M}_{k_o}^{\mathrm{pre}} = (\mathbf{M}_{k_o})(\mathbf{A}_{k_o}^{\mathrm{sk}})^{-1}$ and computing the Cholesky decomposition of $(\mathbf{M}_{k_o}^{\mathrm{pre}})^*(\mathbf{M}_{k_o}^{\mathrm{pre}})$.

Our second stage uses a function for estimating condition numbers of triangular matrices. Given such a function, `cond`, we estimate the orthogonality loss of choosing truncation rank ℓ by $\mathsf{u}\cdot(\mathtt{cond}(\mathbf{A}_\ell^{\mathrm{pre}}))^2$. The motivation for this is that if `cond` bounds condition numbers

from above, then choosing ℓ to satisfy $\texttt{cond}(\mathbf{A}_\ell^{\text{pre}}) \leq \sqrt{\epsilon_{\text{tol}}/\mathsf{u}}$ bounds orthogonality loss by $\mathcal{O}(\epsilon_{\text{tol}})$. Therefore the formal goal of the second stage is to find

$$k = \max\{\ell \,:\, \texttt{cond}(\mathbf{A}_\ell^{\text{pre}}) \leq \sqrt{\epsilon_{\text{tol}}/\mathsf{u}}\}. \tag{18b}$$

When computing (18b) we can assume that $\texttt{cond}(\mathbf{A}_{\ell+1}^{\text{pre}}) \geq \texttt{cond}(\mathbf{A}_\ell^{\text{pre}})$. This assumption holds for the true condition number, as can be seen by applying the eigenvalue interlacing theorem to the Gram matrices of $\mathbf{A}_{\ell+1}^{\text{pre}}$ and $\mathbf{A}_\ell^{\text{pre}}$ (the latter being a submatrix of the former). This assumption is useful because it lets us compute (18b) by binary search over ℓ in $\{1, \ldots, k_o\}$. Even if this assumption does not hold, the only risk in deciding k by binary search is underestimating numerical rank.

One can bound the condition number of a triangular matrix $\mathbf{X}$ in $\mathcal{O}(n^2 \log n)$ time by applying Krylov subspace methods to $\|\mathbf{X}\|_2$ and $\|\mathbf{X}^{-1}\|_2$. A similar approach with half the complexity could be used to estimate $\tau \geq \|\mathbf{I} - \mathbf{X}\|_2$, which could be turned around to bound $\kappa(\mathbf{X}) \leq (1+\tau)/(1-\tau)$ if $\tau < 1$. If we relax the requirement that `cond` always upper-bounds condition numbers, then we can take $\texttt{cond}(\mathbf{X}) = \kappa(\text{diag}(\mathbf{X}))$ to estimate $\kappa(\mathbf{X})$ from below. The last of these three is actually our preferred method, since it works well in practice and is extremely simple to implement.

Remark 4 (What if Cholesky fails in stage one?). Set $\mathbf{G} = (\mathbf{M}_n^{\text{pre}})^*(\mathbf{M}_n^{\text{pre}})$. Consider an index k_o where $\mathbf{G}[1{:}k_o, 1{:}k_o]$ is positive definite, but $\mathbf{G}[1{:}(k_o+1), 1{:}(k_o+1)]$ is not. Running LAPACK's `POTRF` function on $\mathbf{G}$ will produce an error. However, the leading k_o-by-k_o submatrix of the output $\mathbf{R}$ factor will be well-formed and satisfy $(\mathbf{R}[1{:}k_o, 1{:}k_o])^*\mathbf{R}[1{:}k_o, 1{:}k_o] = \mathbf{G}[1{:}k_o, 1{:}k_o]$. Therefore if `POTRF` fails at Line 7 and returns an error code k, then we can take $k_o = k - 1$ as an initial estimate for numerical rank.

4 Pivot quality experiments

This section gives experimental comparisons of pivot quality using the LAPACK default (`GEQP3`) versus those produced by CQRRPT *based on* on that default. Of course, the results of such a comparison depend heavily on the distribution family $\mathscr{F}$ and the sampling factor γ that determines the distribution of our sketching operator. Therefore we take this as an opportunity to show how one might set $\mathscr{F}$ and γ in practice.

Background on leverage scores and coherence Section 2.3 presented results on how γ can be chosen to achieve oblivious subspace embedding properties for various sketching families. The relevant result for SASOs, Theorem 8, provides worst-case bounds that are valuable in theoretical analysis. However, some subspaces are "easier to sketch" with SASOs than this result would suggest, in the sense that the subspace embedding property can reliably be obtained with a far smaller sampling factor or sparsity parameter.

Leverage scores are a useful concept for understanding when a subspace might be easy or hard to accurately sketch with a given distribution. They quantify the extent to which a low-dimensional subspace aligns with coordinate subspaces [Mah11, DM16, DM21].

Definition 12. *Let* $\mathbf{U}$ *be an* $m \times n$ *orthonormal matrix. The* i^{th} *leverage score of* range($\mathbf{U}$) *is the squared row-norm* $\|\mathbf{U}[i, :]\|_2^2$.

When using sketching operators such as SASOs or SRHTs, it is standard to summarize leverage scores with a concept called *coherence*. This is essentially a condition number for random sampling algorithms [Mah11, DM16, DM21]. Formally, the coherence of an $m \times n$ matrix $\mathbf{M}$ is m times the largest leverage score of range($\mathbf{M}$). It is well-documented in the literature that matrices with higher coherence are harder to sketch.

Pivot quality metrics We use two pivot quality metrics. The first is the Frobenius norms of the matrices $\mathbf{C}_\ell$ in block 2-by-2 partitions of $\mathbf{R}$. This has the natural interpretation as the norm of a rank-ℓ approximation of $\mathbf{M}_n - \mathbf{Q}_\ell\mathbf{R}_\ell$; we plot this metric as ratios $\|\mathbf{C}_\ell^{\text{qp3}}\|_{\text{F}}/\|\mathbf{C}_\ell^{\text{ours}}\|_{\text{F}}$. Our second pivot quality metric is the ratios of $r_{ii} := |\mathbf{R}[i,i]|$ to the singular values of $\mathbf{M}$. If $\mathbf{R}$ comes from `GEQP3` then this ratio can be quite bad in the worst case. Letting σ_i denote the i^{th} singular value of $\mathbf{M}$, this only guarantees that $\phi_i := r_{ii}/\sigma_i$ is between $(n(n+1)/2)^{-1/2}$ and 2^{n-1} [Hig21]. Since there is a chance for large deviations, we plot r_{ii}/σ_i for our algorithm and `GEQP3` separately (rather than plotting the ratio $r_{ii}^{\text{qp3}}/r_{ii}^{\text{ours}}$).

4.1 Example low-coherence matrices

Here we consider tall $m \times n$ matrices with $m = 2^{17} = 131072$ and $n = 2000$ whose spectrum falls into one of the two following categories.

- Matrices for which the first ten percent of their singular values are all equal to one and the rest are decaying polynomially down to $1/\kappa(\mathbf{M}) = 10^{-10}$.
- Matrices with a four-step "staircase-shaped" spectrum. The first quarter of the singular values are 1, the next quarter are $8 \cdot 10^{-10}$, the quarter after that are $4 \cdot 10^{-10}$, and the final quarter are all 10^{-10}.

The singular vectors were generated by orthogonalizing the columns of matrices with iid Gaussian entries. Generating the left singular vectors in this way is a standard method for generating low-coherence matrices.

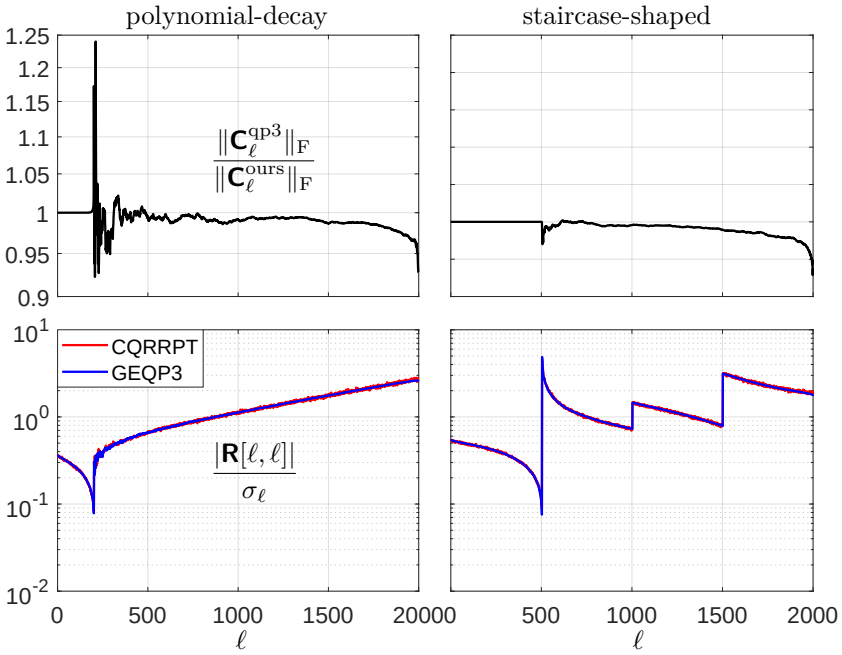

Figure 1: Pivot quality results for low-coherence matrices with two types of spectral decay.

Figure 1 visualizes metrics of CQRRPT's pivot quality for these matrices, when it is configured to use maximally aggressive dimension reduction via SASOs ($\gamma = 1$ and one nonzero per column). For both matrices the ratio of `GEQP3`'s residual-norm metric to that of CQRRPT is close to 1. Although, it is noteworthy that the deviations of this ratio from

1 are more pronounced at ranks which coincide with rapid drops in the singular values. For the second pivot quality metric, observe that the curves for CQRRPT and `GEQP3` are extremely similar (in fact, visually coincident) for the two types of matrices.

4.2 An example high-coherence matrix

Here we consider a matrix $\mathbf{M}$ of the same dimensions as before, $(m, n) = (131072, 2000)$, constructed in three steps. The first step is to vertically stack $c = \lfloor m/n \rfloor$ copies of the $n \times n$ identity matrix and one copy of the first $m - cn$ rows of the $n \times n$ identity. The second step is to select n rows of $\mathbf{M}$ at random and multiply them by 10^{10}. Finally, the third step is to multiply $\mathbf{M}$ on the right by a random $n \times n$ orthogonal matrix. The left singular vectors are not explicitly generated in this matrix to ensure its high coherence.

Column one of Figure 2 shows pivot quality results when applying CQRRPT to $\mathbf{M}$ with overly-aggressive dimension reduction; column two of Figure 2 shows the analogous data for only modestly more expensive dimension reduction. The figures' combined message is two-fold: that there is good reason to use $\gamma > 1$ and more than one nonzero per column in $\mathbf{S}$, and that extensive parameter tuning is not needed to achieve reliable results from CQRRPT at low computational cost.

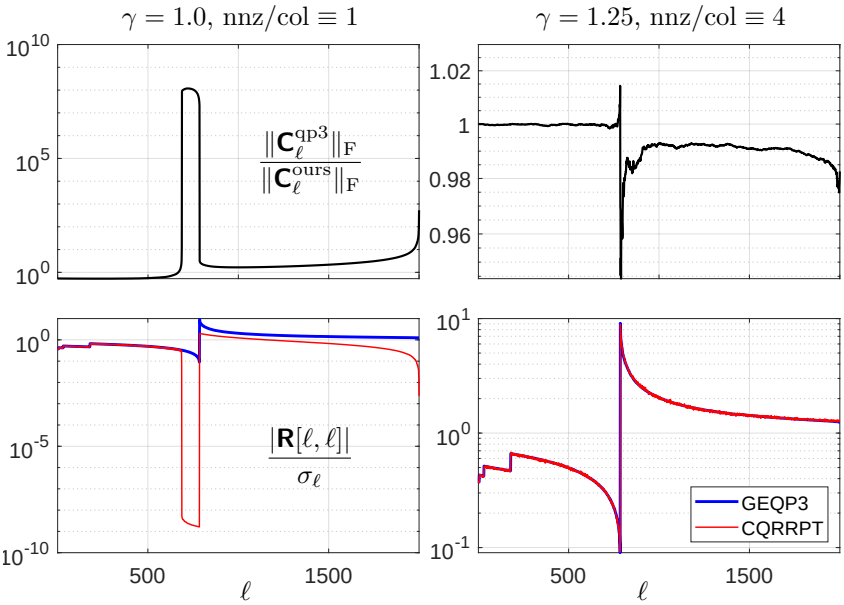

Figure 2: Pivot quality results for a high-coherence matrix under two choices of sketching distribution parameters.

5 Algorithm benchmarking

CQRRPT, together with its testing and benchmarking frameworks, have been implemented in the open-source C++ library called RandLAPACK. Together with its counterpart, RandBLAS, RandLAPACK provides a platform for developing, testing, and benchmarking of high-performance RandNLA algorithms. This software was developed as part of the BALLISTIC [DDL+20] project and is actively contributed to by the authors of this paper. All experiments in this section were run using the following version of our software:

https://github.com/BallisticLA/RandLAPACK/releases/tag/CQRRPT-benchmark-update.

The code for CQRRPT can be found in `/RandLAPACK/drivers/`. The code for constructing and dispatching the experiments can be found in `/benchmark/*`.

Experiments in this section are only concerned with algorithm speed. We measure speed in terms of either an algorithm's canonical flop rate or wall clock time. The former metric is obtained by dividing the `GEQRF` flop count for a given matrix size (see [ADO94]) by the wall clock time required by a particular run of the algorithm. This metric helps in understanding how well different algorithms utilize the capabilities of a given machine.

The test matrices were generated with RandBLAS, by sampling each entry iid from the standard normal distribution. While these test matrices would not be suitable for assessing pivot quality, they are perfectly appropriate for speed benchmarks. All of our tests used double-precision arithmetic, all code was compiled with the -O3 flag, and we use `OMP_NUM_THREADS` = 48 unless otherwise stated; MKL uses the same number of threads. We ran the experiments on a single node in UT Knoxville's ISAAC-NG cluster, equipped with two Intel Xeon Gold 6248R processors. Detailed machine specifications appear in Table 1.

		Intel Xeon Gold 6248R (2x)
Cores per socket		24
Clock Speed	**Base**	3.00 GHz
	Boost	4.00 GHz
Cache sizes per socket	**L1**	1536 KB
	L2	24 MB
	L3	35.75 MB
RAM	**Type**	DDR4-2933
	Size	192 GB
BLAS & LAPACK		MKL 2023.2
GFLOPS in GEMM		1570

Table 1: Key configurations on hardware where testing was performed.

5.1 CQRRPT versus other algorithms

Our first round of experiments compares CQRRPT to other algorithms for QR and QRCP. Specifically, we compare our algorithm to the following.

- `GEQRF` - standard unpivoted Householder QR.
- `GEQR` - unpivoted QR, which dispatches either a specialized algorithm for QR of tall and skinny matrices *or* a general-purpose algorithm according to implementation-specific logic. The selected algorithm is nominally based on a prediction of which algorithm will be more efficient for the given matrix. Intel MKL documentation does not promise that `GEQR` makes the optimal decision in this regard.
- `GEQP3` - standard pivoted QR.
- `GEQPT` - composed of performing `GEQR` on an input matrix to get the factors $\mathbf{Q}_1$ and $\mathbf{R}_1$ and then getting data $(\mathbf{Q}_2, \mathbf{R}, J)$ by running `GEQP3` on $\mathbf{R}_1$; the final matrix $\mathbf{Q}$ is defined implicitly as the product of $\mathbf{Q}_1$ and $\mathbf{Q}_2$.
- `sCholQR3` - shifted CholeskyQR3, our implementation of [FKN+20, Algorithm 4.2], using the Frobenius norm on Line 3 of said algorithm.

The performance data for each algorithm is taken as the best of its execution times over twenty consecutive runs.

Varying matrix sizes. Our first set of experiments was run on $m \times n$ matrices with $m = 65536$ (Figure 3, left) and $m = 131072$ (Figure 3, right), with n varying as the powers of two from 512 to 8192. The sampling factor, γ, is set to the default value of 1.25; the number of nonzeros per column in the sketching operator is set to the default value of 4. Note that all matrices in this experiment are out of cache.

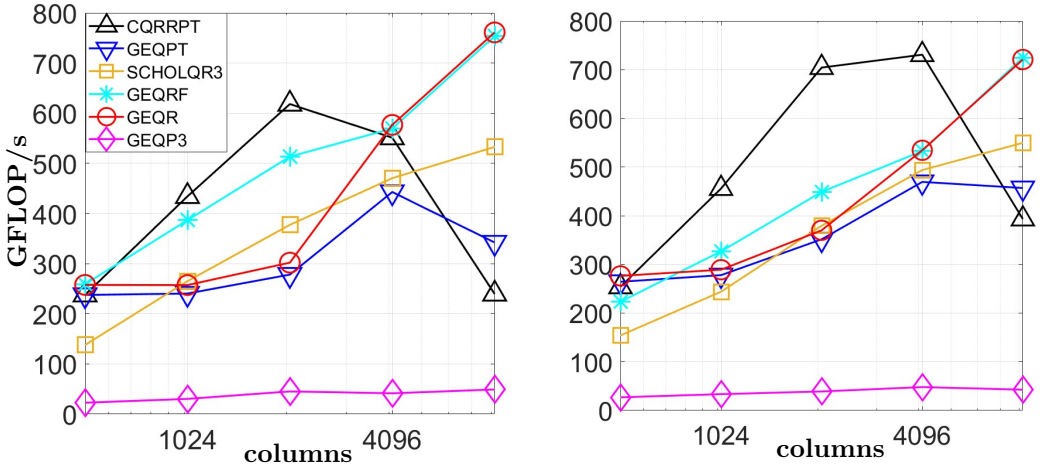

Figure 3: QR schemes performance comparisons for matrices with fixed numbers of rows ($2^{16} = 65536$, and $2^{17} = 131072$ respectively) and varying numbers of columns $(512, \ldots, 8192)$. In the case with 131072 rows, CQRRPT remains the fastest algorithm up until $n = 8192$; at that point, operations on submatrices in underlying LAPACK routines no longer fit in the cache.

The results show that with an increase in input matrix size, CQRRPT outperforms alternative pivoted schemes and has either comparable or superior performance to unpivoted schemes. We are seeing an order of magnitude acceleration over `GEQP3`, and accelerations of up to 2x over `GEQR` and `GEQPT`, and 1.6x over `GEQRF`, for matrices with 131072 rows. On a plot with 65536 rows, the performance of CQRRPT declines after 2048 columns due to the fact that the matrix is not tall enough.

Note that `GEQR` fails to match the performance of `GEQRF` and consequently causes poor performance in `GEQPT`. This is likely due to the fact that the matrix sizes that we use in our experiments are not considered "tall enough" by the internal metric of the MKL implementation of `GEQR`. For experiments with matrices where m is several orders of magnitude larger than n, we refer the readers to Appendix C.

Varying number of threads. Table 2 depicts the performance scaling of various algorithms with the change in number of threads used. For the dimensions of the test matrix used here, CQRRPT remains the fastest algorithm for any number of threads.

Threads	CQRRPT	GEQPT	sCholQR3	GEQRF	GEQR	GEQP3
8	250	159	147	121	166	21
16	490	286	276	274	300	25
24	577	351	316	343	369	29
36	677	372	388	411	393	50
48	704	351	379	448	370	39

Table 2: Canonical GFLOP/s scaling with the number of threads used, given $\mathbf{M} \in \mathbb{R}^{131072 \times 2048}$. At this particular input matrix size, CQRRPT outperforms all other listed algorithms.

It is worth noting that CQRRPT has another possible performance advantage over the alternative algorithms that come from the output format that CQRRPT uses. Namely, having an explicit representation of a **Q**-factor ensures that the factor can be applied via a fast `GEMM` function as opposed to a slower `ORMQR`, which is used to apply an implicitly-stored **Q**-factor. We acknowledge the existence of applications where having an implicitly stored **Q**-factor is a necessity (for example, when the full square **Q** is required). In the context of such applications, the output format of CQRRPT would be disadvantageous.

5.2 Profiling and optimizing CQRRPT

As seen from Algorithm 2, CQRRPT consists of a handful of subcomponents. Understanding how each such subcomponent affects overall runtime for varying input parameters is important for potential future optimizations of CQRRPT. To give perspective on this, we present some timing data in Figure 4. As before, we use an $m \times n$ Gaussian test matrix with $m = 131072$ and n as powers of two from 32 to 16384. We also set the sampling factor, γ, to the default value of 1.25.

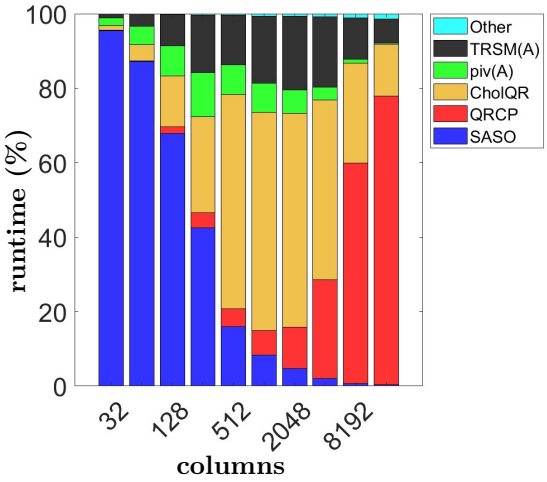

Figure 4: Percentages of CQRRPT's runtime, occupied by its respective subroutines. Note that the cost of sketching becomes negligible for larger matrices. Note also that when $d \geq n$, the cost of applying QRCP to the $d \times n$ sketch grows as $\Omega(n^3)$. By contrast, the cost of applying CholeskyQR to the $m \times n$ preconditioned matrix grows as $\mathcal{O}(mn^2)$. Therefore, it is reasonable that QRCP consumes a larger fraction of runtime as n increases.

An immediate observation to be made here is that QRCP and CholeskyQR become the most time-consuming subroutines as the matrix size increases. Additionally, QRCP clearly becomes the main computational bottleneck. As the embedding dimension parameter greatly affects the timing of QRCP, it is important to understand how the choice of this parameter affects CQRRPT's overall wall clock time.

5.2.1 Effect of varying the embedding dimension parameter

Let's take a look at the effect of varying the embedding dimension in practice. We use Gaussian random input matrices of sizes $131072 \times n$ for $n \in \{1024, 2048, 4096\}$ with the ratio $\gamma = d/n$ varying from 1 to 4 in steps of 0.5. Results are presented in Figure 5.

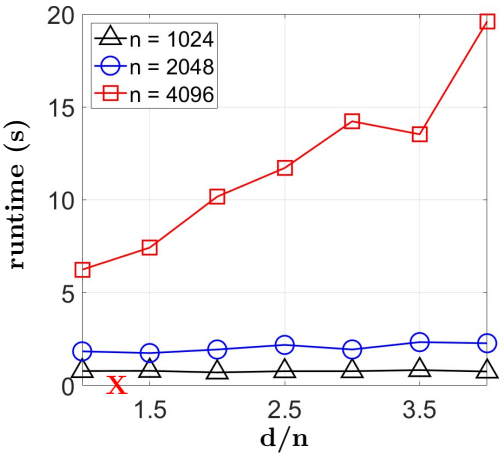

Figure 5: Effect of varying the sampling factor $\gamma \in \{1, 1.5, 2, ..., 4\}$ for matrices of sizes $131072 \times \{1024, 2048, 4096\}$. Runtime represents the wall clock time for the full execution of CQRRPT. An increase of the embedding dimension has a larger effect on wider matrices, as QRCP becomes more expensive with the increased number of columns, as shown in Figure 4. Note that our default value of $\gamma = 1.25$ is marked with **X**; performance in this case can be inferred by interpolating between the first two data points of each series.

Note that for a smaller column size n, increasing the parameter γ has little to no effect on CQRRPT's overall runtime, as the portion of runtime dedicated to performing the QRCP on a sketch is smaller. However, as the figure suggests, for a larger $n = 4096$, the effect of increasing γ is rather noticeable. Recalling our results from Section 4.1, using small values of γ can reliably produce accurate results for matrices of low coherence. Therefore, a good starting point for the value of this parameter would be somewhere between one and two in such cases.

5.2.2 Performing HQRRP on the sketch

By now we have repeatedly seen that CQRRPT's method for QRCP of $\mathbf{M}^{\text{sk}}$ can decisively impact its runtime when n is large. Here we explore the possibility of accelerating this operation by replacing CQRRPT's call to `GEQP3` with a call to HQRRP (Householder QR Factorization With Randomization for Pivoting, see [MQOHvdG17]). As outlined in Section 1.4, HQRRP uses low-dimensional random projections to make pivot decisions in small blocks.

We incorporated an HQRRP implementation into RandLAPACK and ran experiments comparing HQRRP to `GEQP3` directly. In these experiments, we used square matrices of sizes $n \times n$ for $n \in \{1000, 2000, \ldots, 10000\}$ with varying number of OpenMP threads used (which also sets the number of threads used by Intel MKL). Another important tuning parameter to consider in HQRRP benchmarking is the block size. Upon experimenting with different HQRRP block sizes, we have concluded that on our particular system, using the block size of 32 results in the best performance for HQRRP. Using other block sizes in $\{8, 16, 64, 128, 256\}$ resulted in worse performance than what we report in this paper. Results are presented in Figure 6.

As seen in Figure 6, HQRRP achieves its peak performance with 24 threads for large matrix sizes, undoubtedly beating the performance of `GEQP3`. However, for small input matrices, using a large number of threads results in HQRRP failing to outperform `GEQP3`. When 48 threads are used, the performance of HQRRP stagnates, similar to other QR factorization algorithms; see Appendix C for an explanation of why this happens.

Figure 6 allows us to conclude that it is preferred to use 8 threads when running HQRRP on the matrices with $n \leq 6000$; using 24 threads would be optimal for matrices of larger sizes. Additionally, using 24 threads should result in the best performance of `GEQP3`. Assuming CQRRPT used ideal parameter choices for HQRRP and `GEQP3` on our machine, we would expect the HQRRP version to outperform the `GEQP3` version when $n \geq 3000$.

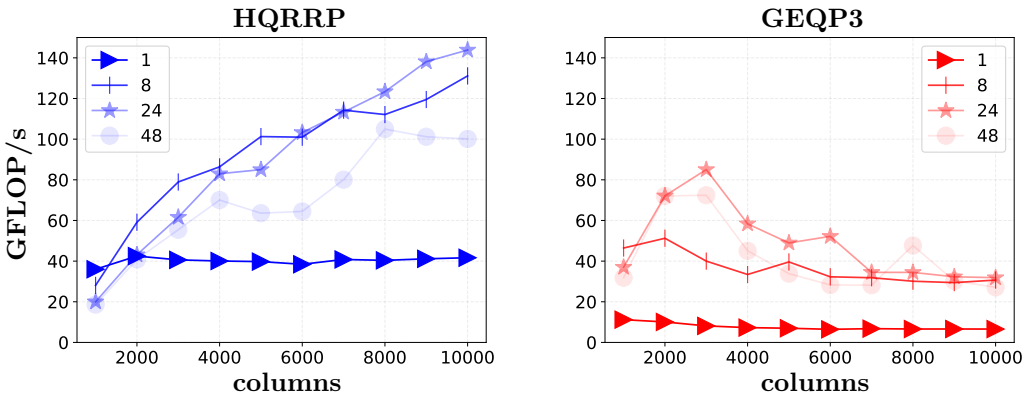

Figure 6: Canonical flop rates for HQRRP vs `GEQP3` with `OMP_THREADS` in $\{1, 8, 24, 48\}$. The experimental speedup of HQRRP over `GEQP3` for the larger matrix sizes is significant, but it does not match the speedup reported in [MQOHvdG17]; this may be due to a variety of factors (primarily, the machine used for benchmarking).

6 Conclusion

This paper introduces CQRRPT, a randomized algorithm for computing a QRCP factorization of a tall matrix. The algorithm consists of two ingredients: obtaining the pivot order and an approximate R-factor from a small random sketch; and retrieving the complete QRCP factorization by a preconditioned CholeskyQR. In comparison to standard QR algorithms, such as Householder QR, TSQR, and their pivoted (possibly randomized) derivatives, our CQRRPT requires $\frac{4}{3}$x less flops[9] and is better suited to modern computational architectures due to fewer global synchronization points, fewer data passes, and/or more straightforward reduction operator. Notably, CQRRPT provides numerical stability even for rank-deficient matrices, in contrast to existing sketched QR, and CholeskyQR methods. This makes the algorithm an appealing tool for orthogonalization.

We established rigorous conditions for the random sketch required by CQRRPT to attain the rank-revealing property and numerical stability. These conditions can be met by employing a sketching distribution that possesses oblivious subspace embedding property. The robustness and performance advantages of CQRRPT were confirmed through extensive numerical experiments of a RandLAPACK implementation based on RandBLAS. In particular, we observed flop rates improvements of up to 10x compared to the standard pivoted QR routine `GEQP3` and 1.4x compared to the unpivoted QR routine `GEQRF` from LAPACK.

This paper also highlights several promising research directions. First, the stability requirements and the efficiency of CQRRPT in certain scenarios can be improved by using more sophisticated sketching operators and schemes for computing the sketch-orthonormal preconditioned matrix. The performance of CQRRPT in modern computing environments such as GPUs, distributed systems, and multi-(or low-)precision arithmetic systems also remains to be explored. Second, combining our theoretical results from Appendix A with the analysis from [XGL17] may lead to a deeper understanding of the properties of HQRRP and other RandNLA methods existing or yet to emerge. Third, CQRRPT can be extended to be applicable to matrices with any aspect ratio by integrating ideas from this paper with ideas from [XGL17, DG17, Mar15, MQOHvdG17] for HQRRP and the related randomized and communication-avoiding QR techniques.

Acknowledgements

The authors would like to thank three anonymous referees for valuable feedback, which greatly improved the clarity of our results and the paper's presentation.

[9] to compute QRCP factorization in explicit form

This work was partially funded by an NSF Collaborative Research Framework: Basic ALgebra LIbraries for Sustainable Technology with Interdisciplinary Collaboration (BALLISTIC), a project of the International Computer Science Institute, the University of Tennessee's ICL, the University of California at Berkeley, and the University of Colorado at Denver (NSF Grant Nos. 2004235, 2004541, 2004763, 2004850, respectively). MWM would also like to acknowledge the NSF, DOE, and ONR Basic Research Challenge on RLA for providing partial support for this work.

RM was partially supported by Laboratory Directed Research and Development (LDRD) funding from Sandia National Laboratories; Sandia is a multimission laboratory managed and operated by National Technology & 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 DENA0003525.

This research was sponsored by the Department of the Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000.

The views and conclusions contained in this document are those of the authors and should not be interpreted as presenting the official policies, either expressed or implied, of the Department of the Air Force, the Department of Energy, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[ADO94] E. Anderson, J. Dongarra, and S. Ostrouchov, *LAPACK working note 41 installation guide for LAPACK*, Technical Report 37996-1301, 1994.

[AMT10] H. Avron, P. Maymounkov, and S. Toledo, *Blendenpik: Supercharging LAPACK's Least-Squares Solver*, SIAM Journal on Scientific Computing **32** (January 2010), no. 3, 1217–1236.

[Bal22] O. Balabanov, *Randomized Cholesky QR factorizations*, arXiv, 2022.

[BG13] Christos Boutsidis and Alex Gittens, *Improved matrix algorithms via the subsampled randomized Hadamard transform*, SIAM Journal on Matrix Analysis and Applications **34** (January 2013), no. 3, 1301–1340.

[BG21] O. Balabanov and L. Grigori, *Randomized block gram-schmidt process for solution of linear systems and eigenvalue problems*, arXiv, 2021.

[BG22] ———, *Randomized Gram–Schmidt process with application to GMRES*, SIAM Journal on Scientific Computing **44** (2022), no. 3, A1450–A1474.

[BG65] Peter Businger and Gene H. Golub, *Linear least squares solutions by householder transformations*, Numerische Mathematik **7** (June 1965), no. 3, 269–276.

[Bjö96] Åke Björk, *Numerical methods for least squares problems*, SIAM, 1996. ISBN 0-89871-360-9.

[BN19] Oleg Balabanov and Anthony Nouy, *Randomized linear algebra for model reduction. part i: Galerkin methods and error estimation*, Advances in Computational Mathematics **45** (2019), no. 5, 2969–3019.

[Coh16] M.B. Cohen, *Nearly tight oblivious subspace embeddings by trace inequalities*, Proceedings of the 27th annual ACM-SIAM Symposium on Discrete Algorithms, December 2016.

[DDL+20] J. Demmel, J. Dongarra, J. Langou, J. Langou, P. Luszczek, and M.W. Mahoney, *Prospectus for the next LAPACK and ScaLAPACK libraries: Basic ALgebra LIbraries for Sustainable Technology with Interdisciplinary Collaboration (BALLISTIC)*, 2020.

[Dem23] J. Demmel, *Nearly optimal block-Jacobi preconditioning*, SIAM Journal on Matrix Analysis and Applications **44** (March 2023), no. 1, 408–413.

[DG17] J.A. Duersch and M. Gu, *Randomized QR with column pivoting*, SIAM Journal on Scientific Computing **39** (January 2017), no. 4, C263–C291.

[DGHL12] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou, *Communication-optimal parallel and sequential QR and LU factorizations*, SIAM Journal on Scientific Computing **34** (2012), no. 1, A206–A239.

[DM16] P. Drineas and M. W. Mahoney, *RandNLA: Randomized numerical linear algebra*, Communications of the ACM **59** (2016), 80–90.

[DM21] M. Derezinski and M. W. Mahoney, *Determinantal point processes in randomized numerical linear algebra*, Notices of the AMS **68** (2021), no. 1, 34–45.

[DMM06] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, *Sampling algorithms for ℓ_2 regression and applications*, Proceedings of the 17th annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2006, pp. 1127–1136.

[DMMS11] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlós, *Faster least squares approximation*, Numerische Mathematik **117** (2011), no. 2, 219–249.

[FGL21] Y. Fan, Y. Guo, and T. Lin, *A novel randomized XR-based preconditioned CholeskyQR algorithm*, arXiv, 2021.

[FKN+20] T. Fukaya, R. Kannan, Y. Nakatsukasa, Y. Yamamoto, and Y. Yanagisawa, *Shifted Cholesky QR for computing the QR factorization of ill-conditioned matrices*, SIAM Journal on Scientific Computing **42** (2020), no. 1, A477–A503 (cit. on pp. 2, 19, 20).

[FNYY14] T. Fukaya, Y. Nakatsukasa, Y. Yanagisawa, and Y. Yamamoto, *Choleskyqr2: A simple and communication-avoiding algorithm for computing a tall-skinny QR factorization on a large-scale parallel system*, 2014 5th workshop on latest advances in scalable algorithms for large-scale systems, 2014, pp. 31–38.

[GE96] M. Gu and S.C. Eisenstat, *Efficient algorithms for computing a strong rank-revealing QR factorization*, SIAM Journal on Scientific Computing **17** (July 1996), no. 4, 848–869.

[GVL13] G.H. Golub and C.F. Van Loan, *Matrix computations*, 4th ed., Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 2013 (en).

[Hig02] N.J. Higham, *Accuracy and stability of numerical algorithms*, Second, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

[Hig21] ———, *What is a rank-revealing factorization?*, 2021. [Accessed 03-Apr-2023].

[Hig22] ———, *The big six matrix factorizations— nhigham.com*, 2022. [Accessed 15-Mar-2023].

[HSBY23] Andrew J. Higgins, Daniel B. Szyld, Erik G. Boman, and Ichitaro Yamazaki, *Analysis of randomized Householder-Cholesky QR factorization with multisketching*, 2023. arXiv:2309.05868.

[Mah11] M. W. Mahoney, *Randomized algorithms for matrices and data*, Foundations and Trends in Machine Learning, NOW Publishers, Boston, 2011.

[Mar15] P.-G. Martinsson, *Blocked rank-revealing QR factorizations: How randomized sampling can be used to avoid single-vector pivoting*, arXiv preprint arXiv:1505.08115 (2015).

[MDM+23] R. Murray, J. Demmel, M.W. Mahoney, N.B. Erichson, M. Melnichenko, O.A. Malik, L. Grigori, P. Luszczek, M. Derezinski, M.E. Lopes, T. Liang, H. Luo, and J. Dongarra, *Randomized numerical linear algebra: A perspective on the field with an eye to software*, 2023. arXiv:2302.11474:v2.

[MQOHvdG17] P.-G. Martinsson, G. Quintana-Ortí, N. Heavner, and R. van de Geijn, *Householder QR factorization with randomization for column pivoting (HQRRP)*, SIAM Journal on Scientific Computing **39** (January 2017), no. 2, C96–C115.

[MSM14] X. Meng, M.A. Saunders, and M.W. Mahoney, *LSRN: A parallel iterative solver for strongly over- or underdetermined systems*, SIAM Journal on Scientific Computing **36** (January 2014), no. 2, C95–C118.

[MT20] P.-G. Martinsson and J. A. Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numerica **29** (2020), 403–572.

[ND15] Hong Diep Nguyen and James Demmel, *Reproducible tall-skinny QR*, 2015 IEEE 22nd symposium on computer arithmetic, 2015, pp. 152–159.

[QOSB98] G. Quintana-Ortí, X. Sun, and C.H. Bischof, *A BLAS-3 version of the QR factorization with column pivoting*, SIAM Journal on Scientific Computing **19** (1998), no. 5, 1486–1494.

[RT08] V. Rokhlin and M. Tygert, *A fast randomized algorithm for overdetermined linear least-squares regression*, Proceedings of the National Academy of Sciences **105** (September 2008), no. 36, 13212–13217.

[Sar06] T. Sarlos, *Improved approximation algorithms for large matrices via random projections*, Proceedings of the 47th annual IEEE Symposium on Foundations of Computer Science (FOCS), 2006, pp. 143–152.

[Slu69] A. Van Der Sluis., *Condition numbers and equilibration of matrices.*, Numerische Mathematik **14** (1969), 14–23.

[SW02] A. Stathopoulos and K. Wu, *A block orthogonalization procedure with constant synchronization requirements*, SIAM J. Sci. Comput. **23** (2002), 2165–2182.

[TOO20] T. Terao, K. Ozaki, and T. Ogita, *LU-Cholesky QR algorithms for thin QR decomposition*, Parallel Computing **92** (2020), 102571.

[TRG05] Rajeev Thakur, Rolf Rabenseifner, and William Gropp, *Optimization of collective communication operations in MPICH*, The International Journal of High Performance Computing Applications **19** (2005), no. 1, 49–66.

[Tro11] J.A. Tropp, *Improved analysis of the subsampled randomized Hadamard transform*, Advances in Adaptive Data Analysis **03** (April 2011), no. 01n02, 115–126.

[TW23] Joel A. Tropp and Robert J. Webber, *Randomized algorithms for low-rank matrix approximation: Design, analysis, and applications*, 2023.

[XGL17] J. Xiao, M. Gu, and J. Langou, *Fast parallel randomized QR with column pivoting algorithms for reliable low-rank matrix approximations*, 2017 IEEE 24th international conference on high performance computing (HiPC), 2017, pp. 233–242.

[YNYF15] Y. Yamamoto, Y. Nakatsukasa, Y. Yanagisawa, and T. Fukaya, *Roundoff error analysis of the CholeskyQR2 algorithm*, Electron. Trans. Numer. Anal **44** (2015), no. 01, 306–326.

[YTD15] I. Yamazaki, S. Tomov, and J. Dongarra, *Mixed-precision Cholesky QR factorization and its case studies on multicore CPUs with multiple GPUs*, SIAM J. Scientific Computing **37** (2015), C307–C330.

A Analysis in exact arithmetic

A.1 Detailed arithmetic complexity analysis

In an effort to give the reader a perspective on an approximate flop count of Algorithm 2, let us take a closer look at what happens inside this algorithm after computing the sketch.

1. Factor $\mathbf{M}^{\text{sk}}$ by QRCP. The cost of this step depends on the algorithm used for QRCP. By default, we use the method from LAPACK's `GEQP3`. With LAPACK's `GEQP3`, this operation would cost $4dnk - 2k^2(d+n) + 4k^3/3$ flops [GVL13, Algorithm 5.4.1], where $k = \text{rank}(\mathbf{M}^{\text{sk}})$ and d is the user-defined embedding dimension of a sketch.

2. Apply the preconditioner to $\mathbf{M}$, to get $\mathbf{M}^{\text{pre}}$. This step is done via `TRSM`, so the cost is mk^2 [ADO94, Page 120].

3. Run CholeskyQR on $\mathbf{M}^{\text{pre}}$. This operation is composed of `SYRK`, `POTRF`, and `TRSM` functions. This operation would cost $mk(k+1) + k^3/3 + k^2/2 + k/6 + mk^2$ flops [ADO94, Page 120].

4. Combine the triangular factors from QRCP on $\mathbf{M}^{\text{sk}}$ and CholeskyQR on $\mathbf{M}^{\text{pre}}$. This cost is negligible; an $\mathcal{O}(n^3)$ `TRMM`.

This results in a cumulative flop count of $2mk^2 + mk(k+1) + 4dnk - 2k^2(d+n) + 5k^3/3 + k^2/2 + k/6$. Put simply, we have an algorithm with an asymptotic flop count with a leading term $3mn^2$ when $d \in o(n)$ and $k = n$.

A.2 Rank-revealing properties

Here we prove pivot quality results claimed in Theorems 4 and 5. Our arguments use the following concept to relate subspace embedding results (common in RandNLA theory) and restricted condition numbers.

Definition 13. *The effective distortion of a $d \times m$ matrix $\mathbf{S}$ for a linear subspace $L \subset \mathbb{R}^m$ is the smallest real number δ where there exists a $t \geq 0$ for which $t\mathbf{S}$ is a δ-embedding for L.*

The specific relationship is as follows [MDM+23, Proposition A.1.1].

Proposition 14. *Suppose the columns of $\mathbf{U}$ are an orthonormal basis for $L = \operatorname{range}(\mathbf{M})$, and let δ denote the effective distortion of $\mathbf{S}$ for L. If $\delta < 1$, then we have*

$$\kappa(\mathbf{SU}) = \sqrt{\frac{1+\delta}{1-\delta}}.$$

Furthermore, we have $\delta = 1$ if and only if $\operatorname{rank}(\mathbf{SM}) < \operatorname{rank}(\mathbf{M})$.

Now we fix an arbitrary permutation vector J of $\{1, \ldots, n\}$ and analyze the R-factors from QR decompositions of the pivoted matrices $\mathbf{M}_n = \mathbf{M}[:, J]$ and $\mathbf{M}_n^{\mathrm{sk}} = \mathbf{M}^{\mathrm{sk}}[:, J]$. Set $k = \operatorname{rank}(\mathbf{M})$. We assume $\operatorname{rank}(\mathbf{M}^{\mathrm{sk}}) = k$, since otherwise there is nothing to prove.

Proposition 15. *If $\mathbf{S}$ has effective distortion $\delta < 1$ for the range of $\mathbf{M}$, then we have*

$$\frac{\sigma_j(\mathbf{A}_\ell)}{\sigma_j(\mathbf{M})} \geq \left(\frac{1-\delta}{1+\delta}\right)^{1/2} \frac{\sigma_j\left(\mathbf{A}_\ell^{\mathrm{sk}}\right)}{\sigma_j\left(\mathbf{M}^{\mathrm{sk}}\right)} \quad \text{for} \quad j \leq \ell \tag{19a}$$

$$\frac{\sigma_j(\mathbf{C}_\ell)}{\sigma_{\ell+j}(\mathbf{M})} \leq \left(\frac{1+\delta}{1-\delta}\right)^{1/2} \frac{\sigma_j\left(\mathbf{C}_\ell^{\mathrm{sk}}\right)}{\sigma_{\ell+j}\left(\mathbf{M}^{\mathrm{sk}}\right)} \quad \text{for} \quad j \leq k-\ell, \tag{19b}$$

$$\text{and} \qquad \|(\mathbf{A}_\ell)^{-1}\mathbf{B}_\ell\|_2 \leq \|(\mathbf{A}_\ell^{\mathrm{sk}})^{-1}\mathbf{B}_\ell^{\mathrm{sk}}\|_2 + \left(\frac{1+\delta}{1-\delta}\right)^{1/2} \sigma_{\min}\left(\mathbf{A}_\ell^{\mathrm{sk}}\right)^{-1} \|\mathbf{C}_\ell^{\mathrm{sk}}\|_2. \tag{19c}$$

Proof. First, note that all of our claimed bounds are invariant under scaling of $\mathbf{S}$ by positive constants. This means that we can assume $\mathbf{S}$ is scaled ($\mathbf{S} \leftarrow t\mathbf{S}$ for some $t \neq 0$) so that its distortion and its effective distortion coincide. From here, it follows directly from the min-max principle and the subspace embedding property of $\mathbf{S}$ that

$$(1-\delta)^{1/2}\sigma_j(\mathbf{M}) \leq \sigma_j(\mathbf{SM}) \leq (1+\delta)^{1/2}\sigma_j(\mathbf{M}) \text{ for } 1 \leq j \leq n. \tag{20}$$

Consider the matrix $\mathbf{M}^{\mathrm{pre}}$ computed in Algorithm 2. We can interpret Theorem 3 with the definition of a subspace embedding to find that

$$(1+\delta)^{-1/2} \leq \sigma_j(\mathbf{M}^{\mathrm{pre}}) \leq (1-\delta)^{-1/2} \text{ for } 1 \leq j \leq k. \tag{21}$$

Next, partition the matrix $\mathbf{R}^{\mathrm{pre}}$ from Algorithm 2 into a 2×2 block matrix in the way analogous to the partitioning of $\mathbf{R}$ and $\mathbf{R}^{\mathrm{sk}}$, and recall the definition $\mathbf{R} = \mathbf{R}^{\mathrm{pre}}\mathbf{R}^{\mathrm{sk}}$.

The bounds in Eq. (21) ensure that $\sigma_{\min}(\mathbf{A}_\ell^{\mathrm{pre}}) \geq \sigma_{\min}(\mathbf{R}^{\mathrm{pre}}) = \sigma_{\min}(\mathbf{M}^{\mathrm{pre}}) \geq (1+\delta)^{-1/2}$ and $\|\mathbf{C}_\ell^{\mathrm{pre}}\| \leq \|\mathbf{R}^{\mathrm{pre}}\| = \|\mathbf{M}^{\mathrm{pre}}\| \leq (1-\delta)^{-1/2}$. Meanwhile, rote calculations let us express the blocks of $\mathbf{R}$ as $\mathbf{A}_\ell = \mathbf{A}_\ell^{\mathrm{pre}}\mathbf{A}_\ell^{\mathrm{sk}}$, $\mathbf{C}_\ell = \mathbf{C}_\ell^{\mathrm{pre}}\mathbf{C}_\ell^{\mathrm{sk}}$ and $\mathbf{B}_\ell = \mathbf{A}_\ell^{\mathrm{pre}}\mathbf{B}_\ell^{\mathrm{sk}} + \mathbf{B}_\ell^{\mathrm{pre}}\mathbf{C}_\ell^{\mathrm{sk}}$. Consequently,

$$\begin{aligned}
\sigma_j\left(\mathbf{A}_\ell^{\mathrm{sk}}\right) &\leq \sigma_{\min}(\mathbf{A}_\ell^{\mathrm{pre}})^{-1}\,\sigma_j\left(\mathbf{A}_\ell^{\mathrm{pre}}\mathbf{A}_\ell^{\mathrm{sk}}\right) = \sigma_{\min}(\mathbf{A}_\ell^{\mathrm{pre}})^{-1}\,\sigma_j(\mathbf{A}_\ell) \leq (1+\delta)^{1/2}\sigma_j(\mathbf{A}_\ell)\,, \\
\sigma_j\left(\mathbf{C}_\ell^{\mathrm{sk}}\right) &\geq \|\mathbf{C}_\ell^{\mathrm{pre}}\|_2^{-1}\sigma_j\left(\mathbf{C}_\ell^{\mathrm{pre}}\mathbf{C}_\ell^{\mathrm{sk}}\right) = \|\mathbf{C}_\ell^{\mathrm{pre}}\|_2^{-1}\sigma_j(\mathbf{C}_\ell) \geq (1-\delta)^{-1/2}\sigma_j(\mathbf{C}_\ell)\,.
\end{aligned}$$

We can combine these inequalities with Eq. (20) to easily get Eqs. (19a) and (19b).

To show Eq. (19c), we first notice that

$$\begin{aligned}\|(\mathbf{A}_\ell)^{-1}\mathbf{B}_\ell\|_2 &= \|(\mathbf{A}_\ell^{\mathrm{pre}}\mathbf{A}_\ell^{\mathrm{sk}})^{-1}(\mathbf{A}_\ell^{\mathrm{pre}}\mathbf{B}_\ell^{\mathrm{sk}} + \mathbf{B}_\ell^{\mathrm{pre}}\mathbf{C}_\ell^{\mathrm{sk}})\|_2 \\ &\leq \|(\mathbf{A}_\ell^{\mathrm{sk}})^{-1}\mathbf{B}_\ell^{\mathrm{sk}}\|_2 + \|(\mathbf{A}_\ell^{\mathrm{sk}})^{-1}(\mathbf{A}_\ell^{\mathrm{pre}})^{-1}\mathbf{B}_\ell^{\mathrm{pre}}\mathbf{C}_\ell^{\mathrm{sk}}\|_2.\end{aligned}$$

In turn, we have

$$\begin{aligned}\|(\mathbf{A}_\ell^{\mathrm{sk}})^{-1}(\mathbf{A}_\ell^{\mathrm{pre}})^{-1}\mathbf{B}_\ell^{\mathrm{pre}}\mathbf{C}_\ell^{\mathrm{sk}}\|_2 &\leq \sigma_{\min}\left(\mathbf{A}_\ell^{\mathrm{sk}}\right)^{-1}\sigma_{\min}\left(\mathbf{R}^{\mathrm{pre}}\right)^{-1}\|\mathbf{R}^{\mathrm{pre}}\|_2\|\mathbf{C}_\ell^{\mathrm{sk}}\|_2 \\ &\leq \left(\frac{1+\delta}{1-\delta}\right)^{1/2}\sigma_{\min}\left(\mathbf{A}_\ell^{\mathrm{sk}}\right)^{-1}\|\mathbf{C}_\ell^{\mathrm{sk}}\|_2,\end{aligned}$$

which finishes the proof. □

It is easy to prove Theorems 4 and 5 from here. Start by using Proposition 14 to express the bounds from Proposition 15 in terms of the restricted condition number for $\mathbf{S}$ on range($\mathbf{M}$). Straightforward algebra shows that if (2a) and (2b) hold for $\mathbf{X} = \mathbf{SM}$, then (19a) and (19b) imply the claim of Theorem 4. Proving Theorem 5 requires a slight detour to show that $\sigma_{\min}\left(\mathbf{A}_\ell^{\mathrm{sk}}\right)^{-1}\|\mathbf{C}_\ell^{\mathrm{sk}}\|_2 \leq f_\ell^2$ follows from (19a) and (19b). Combine this new bound with (19c) to see that if (3) holds for $\mathbf{X} = \mathbf{SM}$ then the claim of Theorem 5 follows.

B Technical numerical stability results

Here we prove results from Section 3, using notation first defined on page 13.

B.1 Proof of Lemma 11

The relation Eq. (12) follows directly from [YNYF15]. To show Eq. (11), notice that

$$\|\mathbf{M}_n - \mathbf{Q}_k\mathbf{R}_k\|_{\mathrm{F}} = \|\mathbf{M}_n - \mathbf{M}_k^{\mathrm{pre}}\mathbf{R}_k^{\mathrm{sk}} + \mathbf{E}_1\mathbf{R}_k^{\mathrm{sk}} + \mathbf{Q}_k\mathbf{E}_2\|_{\mathrm{F}}, \tag{23}$$

where $\mathbf{E}_1 = \mathbf{M}_k^{\mathrm{pre}} - \mathbf{Q}_k\mathbf{R}_k^{\mathrm{pre}}$ and $\mathbf{E}_2 = \mathbf{R}_k - \mathbf{R}_k^{\mathrm{pre}}\mathbf{R}_k^{\mathrm{sk}}$. According to [YNYF15], it holds that $\|\mathbf{E}_1\|_{\mathrm{F}} \leq 8.4\|\mathbf{M}_k^{\mathrm{pre}}\|_2 n^2\mathsf{u}$. By combining this relation with Eq. (12) we deduce that $\|\mathbf{R}_k^{\mathrm{pre}}\|_{\mathrm{F}} \leq 1.01\|\mathbf{M}_k^{\mathrm{pre}}\|_{\mathrm{F}}$. Furthermore, by standard rounding bounds for matrix multiplication [Hig02], we have $\|\mathbf{E}_2\|_{\mathrm{F}} \leq \frac{n\mathsf{u}}{1-n\mathsf{u}}\|\mathbf{R}_k^{\mathrm{pre}}\|_{\mathrm{F}}\|\mathbf{R}_k^{\mathrm{sk}}\|_{\mathrm{F}}$. Next, from Eq. (10) it can be deduced that $\|\mathbf{M}_k^{\mathrm{pre}}\mathbf{R}_k^{\mathrm{sk}}\|_2 \leq 1.01\|\mathbf{M}\|_2$, and therefore

$$\|\mathbf{M}_k^{\mathrm{pre}}\|_2\|\mathbf{R}_k^{\mathrm{sk}}\|_2 \leq 6\,\kappa(\mathbf{M}_k^{\mathrm{pre}})^{-1}\|\mathbf{M}_k^{\mathrm{pre}}\|_2\|\mathbf{R}_k^{\mathrm{sk}}\|_2 \leq 6\sigma_{\min}\left(\mathbf{M}_k^{\mathrm{pre}}\right)\|\mathbf{R}_k^{\mathrm{sk}}\|_2 \leq 6.06\|\mathbf{M}\|_2.$$

By plugging the obtained bounds for $\|\mathbf{E}_1\|_{\mathrm{F}}$, $\|\mathbf{R}_k^{\mathrm{pre}}\|_{\mathrm{F}}$, $\|\mathbf{E}_2\|_{\mathrm{F}}$ and $\|\mathbf{M}_k^{\mathrm{pre}}\|_2\|\mathbf{R}_k^{\mathrm{sk}}\|_2$ to Eq. (23), we obtain

$$\begin{aligned}\|\mathbf{M}_n - \mathbf{Q}_k\mathbf{R}_k\|_{\mathrm{F}} - \|\mathbf{M}_n - \mathbf{M}_k^{\mathrm{pre}}\mathbf{R}_k^{\mathrm{sk}}\|_{\mathrm{F}} &\leq \|\mathbf{E}_1\|_{\mathrm{F}}\|\mathbf{R}_k^{\mathrm{sk}}\|_2 + \|\mathbf{Q}_k\|_2\|\mathbf{E}_2\|_{\mathrm{F}} \\ &\leq 8.4\|\mathbf{M}_k^{\mathrm{pre}}\|_2 n^2\mathsf{u}\|\mathbf{R}_k^{\mathrm{sk}}\|_2 + 1.01n\mathsf{u}\|\mathbf{R}_k^{\mathrm{pre}}\|_{\mathrm{F}}\|\mathbf{R}_k^{\mathrm{sk}}\|_{\mathrm{F}} \\ &\leq 51n^2\mathsf{u}\|\mathbf{M}\|_2 + 1.03n\mathsf{u}\|\mathbf{M}_k^{\mathrm{pre}}\|_{\mathrm{F}}\|\mathbf{R}_k^{\mathrm{sk}}\|_{\mathrm{F}} \\ &\leq (50.5n^2\mathsf{u} + 6.3n^2\mathsf{u})\|\mathbf{M}\|_2,\end{aligned}$$

which completes the proof.

B.2 Preconditioner stability

B.2.1 Assumptions

To characterize the numerical stability of CQRRPT's preconditioner, we need assumptions on the accuracy of the operations at Lines 2 to 5 of Algorithm 2 under finite precision arithmetic. We consider a scenario where Line 5, which dominates the preconditioner's computational cost, is computed using unit roundoff u, whereas cheaper Lines 2 to 4 are computed at a higher precision, using a roundoff $\tilde{\mathsf{u}} = m^{-1}F(n)^{-1}\mathsf{u}$, where $F(n)$ is a low-degree polynomial. We use this computational model to illustrate a crucial property of the preconditioner: the ability to ensure numerical stability with a roundoff u for dominant operations not constrained by m. This property is of interest for integrating CQRRPT into multi- or low-precision arithmetic architectures. Clearly, the numerical stability of CQRRPT computed with two unit roundoffs also implies its stability when computed with a single roundoff. Define,

$$\begin{aligned} \mathbf{E}_1 &:= \mathbf{S}\mathbf{M}_k - \mathbf{M}_k^{\text{sk}}, \\ \mathbf{E}_2 &:= \mathbf{M}^{\text{sk}} - \mathbf{Q}_k^{\text{sk}}\mathbf{R}_k^{\text{sk}} \\ \mathbf{E}_3 &:= \mathbf{M}_k - \mathbf{M}_k^{\text{pre}}\mathbf{A}_k^{\text{sk}} \\ \mathbf{E}_4 &:= \mathbf{S}\overline{\mathbf{M}}_k - \overline{\mathbf{M}}_k^{\text{sk}}, \end{aligned} \tag{24}$$

where $\overline{\mathbf{M}}_k := \mathbf{M}[:, J[k+1{:}n]]$ and $\overline{\mathbf{M}}_k^{\text{sk}} := \mathbf{M}^{\text{sk}}[:, J[k+1{:}n]]$.

We require that several bounds hold on these matrices. The choice for these bounds begins with a base assumption that the unit roundoff u and the dimensions n and k satisfy

$$\mathsf{u} \leq 0.001 n^{-\frac{3}{2}} k^{-\frac{5}{2}}. \tag{25a}$$

With that, the first round of bounds that we discuss are

$$\|\mathbf{E}_1[:, j]\|_2 \leq 0.01 k^{-\frac{1}{2}} \mathsf{u} \|\mathbf{M}_k[:, j]\|_2 \text{ for } 1 \leq j \leq k, \tag{25b}$$

$$\|\mathbf{E}_2[:, j]\|_2 \leq 0.01 k^{-\frac{1}{2}} \mathsf{u} \|\mathbf{M}^{\text{sk}}[:, j]\|_2 \text{ for } 1 \leq j \leq n, \quad \text{and} \quad \|\mathbf{Q}_k^{\text{sk}*}\mathbf{Q}_k^{\text{sk}} - \mathbf{I}\|_{\text{F}} \leq 0.1\mathsf{u}. \tag{25c}$$

It is easy to ensure that these conditions hold. To begin, we note that the classical worst-case rounding analysis ensures $|\mathbf{E}_1| \leq \frac{m\tilde{\mathsf{u}}}{1-m\tilde{\mathsf{u}}}|\mathbf{S}||\mathbf{M}_k|$. This tells us that Eq. (25b) can be achieved if $\tilde{\mathsf{u}} = \mathcal{O}(m^{-1}n^{-1}\mathsf{u})$ and $\|\mathbf{S}\|_{\text{F}} = \mathcal{O}(\sqrt{m})$. Meanwhile, the condition Eq. (25c) can be achieved by using any stable QRCP factorization executed in sufficient precision. This for instance includes the Householder QR with unit roundoff $\tilde{\mathsf{u}} = \mathcal{O}(n^{-1}d^{-\frac{3}{2}}\mathsf{u})$ or Givens QR with unit roundoff $\tilde{\mathsf{u}} = \mathcal{O}(n^{-1}d^{-\frac{1}{2}}\mathsf{u})$ [Hig02].

Our next round of assumptions is more substantial. In particular, assume that the output of the black-box `qrcp` function satisfies the rank-revealing properties

$$\sigma_{\min}\left(\mathbf{A}_k^{\text{sk}}\right) \geq 0.5 n^{-\frac{1}{2}} k^{-\frac{1}{2}} \sigma_k\left(\mathbf{M}^{\text{sk}}\right), \qquad \|\mathbf{C}_{k-1}^{\text{sk}}\|_2 \leq 2 n^{\frac{1}{2}} k^{\frac{1}{2}} \sigma_k\left(\mathbf{M}^{\text{sk}}\right), \tag{25d}$$

$$\|(\mathbf{A}_k^{\text{sk}})^{-1}\mathbf{B}_k^{\text{sk}}\|_{\text{F}} \leq 2 n^{\frac{1}{2}} k^{\frac{1}{2}}. \tag{25e}$$

These conditions can be achieved with the strong rank-revealing QR method from [GE96] with unit roundoff $\tilde{\mathsf{u}} = \mathcal{O}(n^{-1}d^{-\frac{1}{2}}\mathsf{u})$. The method from [GE96] contains an extra parameter f that in our case should be taken as, say, 1.5. Then the `qrcp` subroutine on Line 3 will take a negligible amount $\mathcal{O}(dn^2 \log n)$ of flops and satisfy Eqs. (25d) and (25e). It is important to note that in practical applications, the condition Eqs. (25d) and (25e) is expected to

hold also for traditional QRCP with max-norm column pivoting. However, there are a few exceptional cases in which the traditional QRCP would not satisfy Eq. (25e).

Next, according to [Hig02, Theorem 8.5], we have $\mathbf{M}_k^{\text{pre}}[j,:](\mathbf{A}_k^{\text{sk}}+\Delta\mathbf{A}^{(j)}) = \mathbf{M}_k[j,:]$ with $|\Delta\mathbf{A}^{(j)}| \leq 1.1\mathsf{u}k|\mathbf{A}_k^{\text{sk}}|$, which implies that $|\mathbf{E}_3|$ is bounded by $1.1\mathsf{u}k|\mathbf{M}_k^{\text{pre}}||\mathbf{A}_k^{\text{sk}}|$ and leads to the following condition

$$|\mathbf{E}_3[:,j]| \leq 1.1k\mathsf{u}|\mathbf{M}_k^{\text{pre}}||\mathbf{A}^{\text{sk}}[:,j]| \text{ for } 1 \leq j \leq k. \tag{25f}$$

Furthermore, by the standard rounding analysis we have $|\mathbf{E}_4| \leq \frac{m\tilde{\mathsf{u}}}{1-m\tilde{\mathsf{u}}}|\mathbf{S}||\overline{\mathbf{M}}_k|$, which implies

$$\|\mathbf{E}_4[:,j]\|_2 \leq 0.1n^{-\frac{1}{2}}\mathsf{u}\|\overline{\mathbf{M}}_k[:,j]\|_2 \text{ for } 1 \leq j \leq n-k, \tag{25g}$$

if $\tilde{\mathsf{u}} = \mathcal{O}(m^{-1}n^{-1}\mathsf{u})$ and $\|\mathbf{S}\|_{\text{F}} = \mathcal{O}(\sqrt{m})$.

With the computational model based on the assumptions Eq. (25), we can now establish a numerical characterization of the CQRRPT preconditioner.

B.2.2 Main result

Theorem 16 is our main numerical stability result. It guarantees stability of CQRRPT's preconditioner under the condition that the truncation error associated with the preceding index $k-1$ is greater than the threshold value $G(n,k)\mathsf{u}$, where $G(n,k)$ is a low-degree polynomial. Consequently, by using the largest k that satisfies this condition, we will have a factorization $\mathbf{M}_k^{\text{pre}}\mathbf{R}_k^{\text{sk}}$, which is within a distance of $G(n,k)\mathsf{u}$ from $\mathbf{M}_n$ and for which $\mathbf{M}_k^{\text{pre}}$ is well-conditioned.

Theorem 16. *Consider Algorithm 2 where Line 5 is executed with unit roundoff* $\mathsf{u} \leq 0.001n^{-\frac{3}{2}}k^{-\frac{5}{2}}$. *Assume that the other lines are executed with unit roundoff* $\tilde{\mathsf{u}} = m^{-1}F(n)^{-1}\mathsf{u}$, *where* $F(n)$ *is some low-degree polynomial, and* $\|\mathbf{S}\|_{\text{F}} = \mathcal{O}(\sqrt{m})$, *so that the conditions Eq. (25) hold. Assume that* k *satisfies*

$$1000n^{\frac{3}{2}}k^{\frac{5}{2}}\mathsf{u} \leq \frac{\|\mathbf{C}_{k-1}^{\text{sk}}\|_{\text{F}}}{\|\mathbf{R}^{\text{sk}}\|_2}.$$

If $\mathbf{S}$ *is an* δ*-embedding for* $\text{range}(\mathbf{M})$ *with* $\delta \leq 1/2$, *then*

$$\frac{\|\mathbf{M}_n - \mathbf{M}_k^{\text{pre}}\mathbf{R}_k^{\text{sk}}\|_{\text{F}}}{\|\mathbf{M}\|_{\text{F}}} \leq 2\frac{\|\mathbf{C}_k^{\text{sk}}\|_{\text{F}}}{\|\mathbf{R}^{\text{sk}}\|_2} + 10nk\mathsf{u} \tag{26a}$$

$$0.8 \leq \sigma_{\min}(\mathbf{M}_k^{\text{pre}}) \leq \sigma_{\max}(\mathbf{M}_k^{\text{pre}}) \leq 1.44. \tag{26b}$$

Our proof of Theorem 16 uses two intermediate propositions that provide further insights into the stability of the CQRRPT preconditioner. First, Proposition 17 concerns how the permutation obtained on Line 3 of CQRRPT effectively bounds the condition number of the submatrix $\mathbf{M}_k$. Second, Proposition 18 involves examining the factors $\mathbf{M}_k^{\text{pre}}$ and $\mathbf{A}_k^{\text{sk}}$ as an unpivoted "sketched CholeskyQR" factorization of $\mathbf{M}_k$, which enables us to leverage a result from [Bal22].

Proposition 17. *Consider Algorithm 2 where Line 5 is executed with unit roundoff* $\mathsf{u} \leq 0.001n^{-\frac{3}{2}}k^{-\frac{5}{2}}$ *and other lines are executed with unit roundoff* $\tilde{\mathsf{u}} = m^{-1}F(n)^{-1}\mathsf{u}$, *where* $F(n)$ *is some low-degree polynomial, and* $\|\mathbf{S}\|_{\text{F}} = \mathcal{O}(\sqrt{m})$, *so that the conditions Eq. (25) hold. Assume that* k *is such that*

$$n^{\frac{3}{2}}k\mathsf{u} \leq \|\mathbf{C}_{k-1}^{\text{sk}}\|_{\text{F}}\|\mathbf{R}^{\text{sk}}\|_2^{-1}.$$

If $\mathbf{S}$ *is an* δ*-embedding for* $\mathbf{M}_k$ *with* $\delta \leq 1/2$, *then we have*

$$\kappa(\mathbf{M}_k) \leq 10n^{\frac{3}{2}}k\|\mathbf{C}_{k-1}^{\text{sk}}\|_{\text{F}}^{-1}\|\mathbf{R}^{\text{sk}}\|_2 \leq 2.5\mathsf{u}^{-1}.$$

Proof. Denote $\|\mathbf{C}_{k-1}^{\mathrm{sk}}\|_{\mathrm{F}}\|\mathbf{R}_k^{\mathrm{sk}}\|_2^{-1}$ by τ_{k-1}. Frame the assumptions Eq. (25). Notice that by Eq. (25d),

$$\sigma_{\min}\left(\mathbf{A}_k^{\mathrm{sk}}\right) \geq 4^{-1}n^{-1}k^{-1}\|\mathbf{C}_{k-1}^{\mathrm{sk}}\|_2 \geq 4^{-1}n^{-\frac{3}{2}}k^{-1}\tau_{k-1}\|\mathbf{A}_k^{\mathrm{sk}}\|_2. \tag{27}$$

Thus, it is deduced that

$$\kappa(\mathbf{A}_k^{\mathrm{sk}}) \leq 4n^{\frac{3}{2}}k\tau_{k-1}^{-1} \leq \mathsf{u}^{-1}. \tag{28}$$

Furthermore, by Eq. (25c) we have

$$\|\mathbf{M}_k^{\mathrm{sk}}\|_2 \leq \|\mathbf{Q}_k^{\mathrm{sk}}\|_2\|\mathbf{A}_k^{\mathrm{sk}}\|_2 + \|\mathbf{E}_2[:,1:k]\|_2 \leq 1.01\|\mathbf{A}_k^{\mathrm{sk}}\|_2, \tag{29}$$

and

$$\begin{aligned}\sigma_{\min}\left(\mathbf{M}_k^{\mathrm{sk}}\right) &\geq \sigma_{\min}\left(\mathbf{Q}_k^{\mathrm{sk}}\right)\sigma_{\min}\left(\mathbf{A}_k^{\mathrm{sk}}\right) - \|\mathbf{E}_2[:,1:k]\|_2 \\ &\geq 0.99\sigma_{\min}\left(\mathbf{A}_k^{\mathrm{sk}}\right) - 0.01u\|\mathbf{M}_k^{\mathrm{sk}}\|_2 \geq 0.99\sigma_{\min}\left(\mathbf{A}_k^{\mathrm{sk}}\right) - 0.011\mathsf{u}\|\mathbf{A}_k^{\mathrm{sk}}\|_2 \\ &\geq \sigma_{\min}\left(\mathbf{A}_k^{\mathrm{sk}}\right)\left(0.99 - 0.011\mathsf{u}\,\kappa(\mathbf{A}_k^{\mathrm{sk}})\right) \geq 0.97\sigma_{\min}\left(\mathbf{A}_k^{\mathrm{sk}}\right).\end{aligned} \tag{30}$$

Consequently,

$$\kappa(\mathbf{M}_k^{\mathrm{sk}}) \leq 4.2n^{\frac{3}{2}}k\tau_{k-1}^{-1} \leq 1.05\mathsf{u}^{-1}. \tag{31}$$

Next, by Eq. (25b) we get

$$\|\mathbf{S}\mathbf{M}_k\|_2 \leq \|\mathbf{M}_k^{\mathrm{sk}}\|_2 + \|\mathbf{E}_1\|_2 \leq 1.01\|\mathbf{M}_k^{\mathrm{sk}}\|_2, \tag{32}$$

which due to the δ-embedding property of $\mathbf{S}$ implies that

$$\|\mathbf{M}_k\|_2 \leq 1.5\|\mathbf{M}_k^{\mathrm{sk}}\|_2. \tag{33}$$

We also have by Eqs. (25b) and (31),

$$\begin{aligned}\sigma_{\min}\left(\mathbf{S}\mathbf{M}_k\right) &\geq \sigma_{\min}\left(\mathbf{M}_k^{\mathrm{sk}}\right) - \|\mathbf{E}_1\|_2 \geq \sigma_{\min}\left(\mathbf{M}_k^{\mathrm{sk}}\right) - 0.01u\|\mathbf{M}_k\|_2 \\ &\geq \sigma_{\min}\left(\mathbf{M}_k^{\mathrm{sk}}\right) - 0.015u\|\mathbf{M}_k^{\mathrm{sk}}\|_2 \geq 0.92\sigma_{\min}\left(\mathbf{M}_k^{\mathrm{sk}}\right).\end{aligned} \tag{34}$$

Consequently, by the δ-embedding property of $\mathbf{S}$ and Eqs. (33) and (34),

$$\kappa(\mathbf{M}_k) \leq \sqrt{\frac{1+\delta}{1-\delta}}\,\kappa(\mathbf{S}\mathbf{M}_k) \leq 1.91\,\kappa(\mathbf{M}_k^{\mathrm{sk}}) \leq 10n^{\frac{3}{2}}k\tau_{k-1}^{-1} \leq 2.5u^{-1},$$

which finishes the proof. □

Proposition 18 (Corollary of Theorem 5.2 from [Bal22])**.** *Consider Algorithm 2 where Line 5 is executed with unit roundoff* $\mathsf{u} \leq 0.01n^{-\frac{3}{2}}\,\kappa(\mathbf{M}_k)^{-1}$ *and other lines are executed with unit roundoff* $\tilde{\mathsf{u}} = m^{-1}F(k)^{-1}\mathsf{u}$*, where* $F(k)$ *is some low-degree polynomial, and* $\|\mathbf{S}\|_{\mathrm{F}} = \mathcal{O}(\sqrt{m})$*, so that the assumptions Eq. (25) hold. If* $\mathbf{S}$ *is an* δ*-embedding for* $\mathbf{M}_k$ *with* $\delta \leq 1/2$ *then*

$$\mathbf{M}_k + \Delta\mathbf{M} = \mathbf{M}_k^{\mathrm{pre}}\mathbf{A}_k^{\mathrm{sk}} \tag{35}$$

with $\|\Delta\mathbf{M}_k[:,j]\|_2 \leq 2.1k\mathsf{u}\|\mathbf{M}[:,j]\|_2$ *for* $1 \leq j \leq k$*. Furthermore, we have*

$$(1+\delta)^{-1/2} - 4k^{\frac{3}{2}}\mathsf{u}\,\kappa(\mathbf{M}_k) \leq \sigma_{\min}\left(\mathbf{M}_k^{\mathrm{pre}}\right) \leq \sigma_{\max}\left(\mathbf{M}_k^{\mathrm{pre}}\right) \leq (1-\delta)^{-1/2} + 4k^{\frac{3}{2}}\mathsf{u}\,\kappa(\mathbf{M}_k). \tag{36}$$

We are now ready to present the proof of Theorem 16.

Proof of Theorem 16. Denote $\|\mathbf{C}_k^{\text{sk}}\|_{\text{F}}\|\mathbf{R}_k^{\text{sk}}\|_2^{-1}$ by τ_k. By Eqs. (25b), (25c), (25e) and (25g) we have

$$\begin{aligned}\|\overline{\mathbf{M}}_k - \mathbf{M}_k^{\text{pre}}\mathbf{B}_k^{\text{sk}}\|_{\text{F}} &\le \|\overline{\mathbf{M}}_k - \mathbf{M}_k(\mathbf{A}_k^{\text{sk}})^{-1}\mathbf{B}_k^{\text{sk}}\|_{\text{F}} + \|\mathbf{E}_3(\mathbf{A}_k^{\text{sk}})^{-1}\mathbf{B}_k^{\text{sk}}\|_{\text{F}} \\ &\le (1-\delta)^{-1/2}\|\mathbf{S}(\overline{\mathbf{M}}_k - \mathbf{M}_k(\mathbf{A}_k^{\text{sk}})^{-1}\mathbf{B}_k^{\text{sk}})\|_{\text{F}} + \|\mathbf{E}_3\|_{\text{F}}\|(\mathbf{A}_k^{\text{sk}})^{-1}\mathbf{B}_k^{\text{sk}}\|_2 \\ &\le (1-\delta)^{-1/2}(\|\mathbf{E}_4 + \mathbf{E}_5 + \mathbf{E}_6 + \mathbf{E}_7\|_{\text{F}}) + 2n^{\frac{1}{2}}k^{\frac{1}{2}}\|\mathbf{E}_3\|_{\text{F}},\end{aligned} \tag{37}$$

where $\mathbf{E}_4 := \mathbf{S}\overline{\mathbf{M}}_k - \overline{\mathbf{M}}_k^{\text{sk}}, \mathbf{E}_5 := \overline{\mathbf{M}}_k^{\text{sk}} - \mathbf{Q}_k^{\text{sk}}\mathbf{B}_k^{\text{sk}}, \mathbf{E}_6 := (\mathbf{Q}_k^{\text{sk}}\mathbf{A}_k^{\text{sk}} - \mathbf{M}_k^{\text{sk}})(\mathbf{A}_k^{\text{sk}})^{-1}\mathbf{B}_k^{\text{sk}}$, and $\mathbf{E}_7 := (\mathbf{M}_k^{\text{sk}} - \mathbf{S}\mathbf{M}_k)(\mathbf{A}_k^{\text{sk}})^{-1}\mathbf{B}_k^{\text{sk}}$ satisfy the following bounds

$$\begin{aligned}\|\mathbf{E}_4\|_{\text{F}} &= \|\mathbf{S}\overline{\mathbf{M}}_k - \overline{\mathbf{M}}_k^{\text{sk}}\|_{\text{F}} \le 0.1\mathsf{u}\|\overline{\mathbf{M}}_k\|_2 \\ \|\mathbf{E}_5\|_{\text{F}} &= \|\overline{\mathbf{M}}_k^{\text{sk}} - \mathbf{Q}_k^{\text{sk}}\mathbf{B}_k^{\text{sk}}\|_{\text{F}} \le \|\mathbf{Q}_k^{\text{sk}}\|_2\|\mathbf{C}_k^{\text{sk}}\|_{\text{F}} + \|\mathbf{E}_2\|_{\text{F}} \\ &\le 1.01\tau_k\|\mathbf{R}_k^{\text{sk}}\|_2 + \|\mathbf{E}_2\|_{\text{F}} \le (1+\delta)^{1/2}(1.02\tau_k + 0.01\sqrt{n}u)\|\mathbf{M}\|_{\text{F}} \\ \|\mathbf{E}_6\|_{\text{F}} &= \|(\mathbf{Q}_k^{\text{sk}}\mathbf{A}_k^{\text{sk}} - \mathbf{M}_k^{\text{sk}})(\mathbf{A}_k^{\text{sk}})^{-1}\mathbf{B}_k^{\text{sk}}\|_{\text{F}} \\ &\le \|\mathbf{Q}_k^{\text{sk}}\mathbf{A}_k^{\text{sk}} - \mathbf{M}_k^{\text{sk}}\|_{\text{F}}\|(\mathbf{A}_k^{\text{sk}})^{-1}\mathbf{B}_k^{\text{sk}}\|_2 \\ &\le 2n^{\frac{1}{2}}k^{\frac{1}{2}}\|\mathbf{E}_2[:,1:k]\|_{\text{F}} \le 0.02n^{\frac{1}{2}}k^{\frac{1}{2}}\mathsf{u}\|\mathbf{M}_k^{\text{sk}}\|_2 \le 0.03n\mathsf{u}\|\mathbf{M}_k\|_2 \\ \|\mathbf{E}_7\|_{\text{F}} &= \|(\mathbf{M}_k^{\text{sk}} - \mathbf{S}\mathbf{M}_k)(\mathbf{A}_k^{\text{sk}})^{-1}\mathbf{B}_k^{\text{sk}}\|_2 \le \|\mathbf{M}_k^{\text{sk}} - \mathbf{S}\mathbf{M}_k\|_{\text{F}}\|(\mathbf{A}_k^{\text{sk}})^{-1}\mathbf{B}^{\text{sk}}\|_2 \\ &\le 2n^{\frac{1}{2}}k^{\frac{1}{2}}\|\mathbf{E}_1\|_{\text{F}} \le 0.02n\mathsf{u}\|\mathbf{M}_k\|_2.\end{aligned}$$

By using the triangle inequality and the fact that $\delta \le 1/2$, we obtain

$$\|\overline{\mathbf{M}}_k - \mathbf{M}_k^{\text{pre}}\mathbf{B}_k^{\text{sk}}\|_{\text{F}} \le \frac{4}{3}(0.2n\mathsf{u} + 1.02\tfrac{5}{4}\tau_k)\|\mathbf{M}\|_{\text{F}} + 2n^{\frac{1}{2}}k^{\frac{1}{2}}\|\mathbf{E}_3\|_{\text{F}}. \tag{38}$$

Furthermore, from Proposition 17 it follows that

$$\kappa(\mathbf{M}_k) \le 10n^{\frac{3}{2}}k{\tau_k}^{-1}. \tag{39}$$

By looking at $\mathbf{M}_k^{\text{pre}}$ and $\mathbf{A}_k^{\text{sk}}$ as a sketched CholeskyQR factorization of $\mathbf{M}_k$, according to Proposition 18, we have

$$\|\mathbf{E}_3\|_{\text{F}} = \|\mathbf{M}_k - \mathbf{M}_k^{\text{pre}}\mathbf{A}_k^{\text{sk}}\|_{\text{F}} \le 2.1k\mathsf{u}\|\mathbf{M}_k\|_{\text{F}} \tag{40a}$$

$$(1+\delta)^{-1/2} - 4k^{\frac{3}{2}}\mathsf{u}\,\kappa(\mathbf{M}_k) \le \sigma_{\min}(\mathbf{M}_k^{\text{pre}}) \le \sigma_{\max}(\mathbf{M}_k^{\text{pre}}) \le (1-\delta)^{-1/2} + 4k^{\frac{3}{2}}\mathsf{u}\,\kappa(\mathbf{M}_k) \tag{40b}$$

By combing Eq. (40a) with Eq. (38) and using the triangle inequality we obtain Eq. (26a). By combining Eq. (40b) with Eq. (39) we obtain Eq. (26b) and finish the proof. □

C More on numerical experiments

Extremely tall matrices. The matrices used in Section 5's experiments were tall, but not as tall as some readers might anticipate. Therefore here we consider matrices with one million rows and $32 \ldots 16384$ columns. Figure 7 presents performance results (in canonical GFLOPs/second) for the algorithms considered before. It shows that for very thin matrices ($n \le 256$), CQRRPT may be inefficient compared to alternative algorithms. This can be understood with performance profiling data in Figure 8. Specifically, for these extremely tall and thin matrices, the runtime is dominated by sketching, and the time to pivot $\mathbf{M}$ before preconditioning takes nearly as long as CholeskyQR. As before, our experiments were run on the machine described in Table 1 using 48 threads.

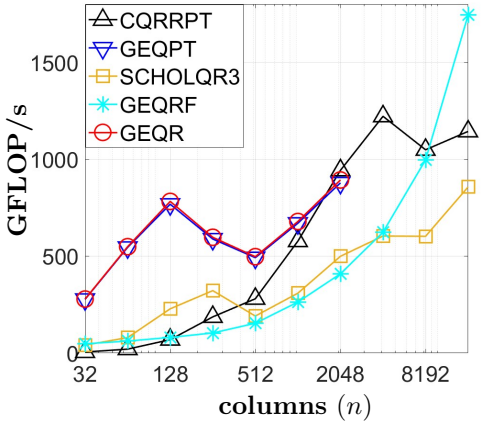

Figure 7: QR schemes performance comparisons for matrices with one million rows and varying numbers of columns $(32, \ldots, 16384)$. The plot does not depict `GEQR` and `GEQPT` results for the number of columns larger than 2048, because of an internal overflow of parameters related to the input matrix size and a consequent early termination.

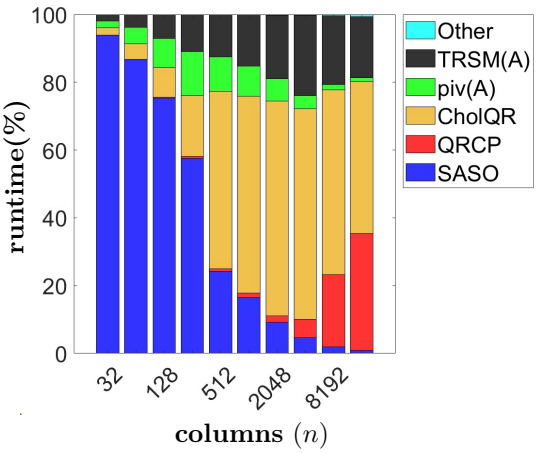

Figure 8: Percentages of runtime used by CQRRPT's subroutines, for matrices with one million rows and the indicated number of columns. Note that although CholQR occupies the dominant portion of runtime across the larger ($n \geq 512$) matrix sizes, its runtime is primarily comprised of vector-vector operations for the smaller matrix sizes. As the matrix size increases, `TRSM` routine gradually achieves a `GEMM`-like performance level.

HQRRP performance details. Figure 6 is our attempt to give a contemporary refresh of Figures 4 and 5 from the original HQRRP publication [MQOHvdG17]. It is important to note that the initial HQRRP thread scalability study was performed on Intel Xeon E5-2695v3 (the Haswell platform) processor with clocked capped at 2.3 GHz featuring "only" 14 cores in a single socket with 22nm node size. Our experiments used dual-socket Intel Xeon Gold 6248R (Cascade Lake platform) with 24 cores per socket and 48 cores total with 14nm feature size - a NUMA design. In essence, the new processor engages a much larger number of cores in the computation and they have to be coordinated by on-node interconnect, that is, the design keeps the content of Level 1 caches coherent through a memory controller state tracking each L1 cache line. For smaller matrix sizes, lower-level caches, especially Level 3, are very effective in alleviating the demand for cache lines from the main memory and HQRRP behaves as it did on the old Haswell system. For larger matrix sizes, HQRRP performance drops as its internal pivoted QR function, that is implemented using level 2 BLAS routines, becomes too costly.