ACCURACY AND STABILITY OF CUR DECOMPOSITIONS WITH OVERSAMPLING *

TAEJUN PARK† AND YUJI NAKATSUKASA†

Abstract. This work investigates the accuracy and numerical stability of CUR decompositions with oversampling. The CUR decomposition approximates a matrix using a subset of columns and rows of the matrix. When the number of columns and the rows are the same, the CUR decomposition can become unstable and less accurate due to the presence of the matrix inverse in the core matrix. Nevertheless, we demonstrate that the CUR decomposition can be implemented in a numerical stable manner and illustrate that oversampling, which increases either the number of columns or rows in the CUR decomposition, can enhance its accuracy and stability. Additionally, this work devises an algorithm for oversampling motivated by the theory of the CUR decomposition and the cosine-sine decomposition, whose competitiveness is illustrated through experiments.

Key words. Low-rank approximation, CUR decomposition, stability analysis, oversampling

MSC codes. 15A23, 65F55, 65G50

1. Introduction.

The computation of a low-rank approximation to a matrix is omnipresent in the computational sciences [53]. It has seen increasing popularity due to its importance in tackling large scale problems. An important aspect of low-rank approximation is the selection of bases that approximate the span of a matrix's column and/or row spaces. In this work, we consider a natural choice of using a subset of rows and columns of the original matrix as low-rank bases, namely the CUR decomposition. The CUR decomposition [7, 38, 49], also known as a "matrix skeleton" approximation [28], is a low-rank approximation that approximates a matrix A as a product[1]

$$\underset{m\times n}{A} \approx \underset{m\times k}{C}\ \underset{k\times k}{Z}\ \underset{k\times n}{R} \tag{1.1}$$

where in MATLAB notation, $C = A(:, J)$ is a k-subset of columns of A with $J \subseteq \{1, \dots, m\}$ being the column indices and $R = A(I, :)$ is a k-subset of the rows of A with $I \subseteq \{1, \dots, n\}$ being the row indices. The factors C and R are subsets of the original matrix A that inherit certain properties of the original matrix, such as sparsity or nonnegativity, a property that is absent in the truncated SVD. They also assist with data interpretation by revealing the important rows and columns of A. The CUR decomposition is also memory efficient when the entries of A can be computed or extracted quickly because we can just store the column and row indices without explicitly storing C and R.

There are two common choices for Z, which we refer to as the core matrix: $C^\dagger A R^\dagger$ or $A(I, J)^{-1}$ (or more generally $A(I, J)^\dagger$) [31]. The choice $Z = C^\dagger A R^\dagger$ minimizes the Frobenius norm error $\|A - CZR\|_F$ given the choice of C and R. Hence, we refer to the CUR decomposition with this choice as CURBA (CUR with Best Approximation) for shorthand in this work. While this choice is more robust, it involves the full matrix A to form $C^\dagger A R^\dagger$, which can be costly, requiring $O(mnk)$ operations for a dense

*Date: November 20, 2024

Funding: TP was supported by the Heilbronn Institute for Mathematical Research. YN is supported by EPSRC grants EP/Y010086/1 and EP/Y030990/1.

†Mathematical Institute, University of Oxford, Oxford, OX2 6GG, UK, (park@maths.ox.ac.uk, nakatsukasa@maths.ox.ac.uk).

[1]The notation Z is temporarily used to denote the core matrix instead of the traditional notation U as we use U to denote the intersection of C and R, i.e., $U = A(I, J)$ later.

matrix A. On the other hand, the choice $Z = A(I,J)^{-1}$, which is often called the cross approximation [13], is more efficient as we only require the overlapping entries of C and R, and need not even read the whole matrix A. We refer to this version of the CUR decomposition as CURCA (CUR with Cross Approximation) for shorthand. However, this choice has the drawback that $A(I,J)$ can be (nearly) singular, which can lead to poor approximation error. Indeed, the ill-conditioning of $A(I,J)$ for CURCA becomes alarming when computing its matrix (pseudo)inverse, a concern that has been raised in various papers [20, 40, 49]. For this reason, the related interpolative decomposition [39] is often advocated to avoid numerical instability with CUR, especially CURCA. The aim of this paper is to address the instability of the CURCA by demonstrating that it can be implemented in a numerically stable manner using the ϵ-pseudoinverse in the presence of roundoff errors. The ϵ-pseudoinverse variant takes the ϵ-pseudoinverse of $A(I,J)$ by first truncating the singular values of $A(I,J)$ less than ϵ before computing its pseudoinverse, which has been explored in a related context in [8, 12, 42, 44]. We abbreviate the CURCA with ϵ-pseudoinverse as the stabilized CURCA (SCURCA for short). Furthermore, we illustrate that oversampling, which increases either the number of columns or rows in the CURCA, can be incorporated to enhance its accuracy and stability. Specifically, we recommend oversampling in proportion to the target rank; see Section 5.3. Throughout this work, we concentrate on CURCA and take $U = A(I,J)$ unless otherwise stated, such that CURCA can be written as $CU^\dagger R$, but the analysis and the relevant counterparts regarding CURBA can be found in Appendix A.[2]

Overview. In this work, we focus on the CURCA. Section 2 provides background information on the CUR decomposition, discusses related and existing works and outlines our contributions. In Section 3, we prove a theoretical bound for the CURCA and its ϵ-pseudoinverse variant, the SCURCA with oversampling and show that the SCURCA can be computed in a numerically stable way in the presence of roundoff errors.[3] In Section 4, we study oversampling of indices for the CURCA with the purpose of improving its accuracy and numerical stability based on the theory developed in Section 3. The setting we are interested in is when we have row indices I and column indices J with $|I| = |J| = k$, how should we oversample one of I or J. We address this question by proposing a deterministic way of oversampling for which the rationale for our idea can be explained by the cosine-sine (CS) decomposition. The construction could be of use in other contexts where (over)sampling from a subspace is desired; a common task that finds use, for example, in model reduction [11] and active learning [48]. We conclude with numerical illustrations in Section 5, demonstrating the stability of our method in computing the CURCA and the strength of oversampling.

Notation. Throughout, we use $\|\cdot\|_2$ for the spectral norm or the vector-ℓ_2 norm, $\|\cdot\|_F$ for the Frobenius norm and $\|\cdot\|$ for any unitarily invariant norms. We use dagger † to denote the pseudoinverse of a matrix and $[\![A]\!]_k$ to denote the best rank-k approximation to A in any unitarily invariant norm, i.e., the approximation derived from truncated SVD [35]. Unless specified otherwise, $\sigma_i(A)$ denotes the ith largest singular

[2] For the CURBA, there already exists a numerically stable way of computing the CURBA given by the StableCUR algorithm in [1]. Furthermore, the effect of oversampling is less effective as the CURBA, $A \approx C(C^\dagger A R^\dagger)R$ is a more robust and stable (but expensive) approximation than the CURCA, $A \approx CU^\dagger R$; see Appendix A.

[3] The stability analysis of the plain CURCA, $CU^\dagger R$ is not covered in this work; in our analysis we require using the ϵ-pseudoinverse. However, we observe its numerical stability in practice; see Section 5.1.

value of the matrix A. We use MATLAB style notation for matrices and vectors. For example, for the kth to $(k+j)$th columns of a matrix A we write $A(:, k:k+j)$. We use I and J for the row and the column indices respectively and set $\Pi_I = I_m(:, I)$ and $\Pi_J = I_n(:, J)$ so that $A(:, J) = A\Pi_J$ and $A(I, :) = \Pi_I^T A$ for $A \in \mathbb{R}^{m\times n}$. Here, I_m denotes the $m \times m$ identity matrix. Lastly, we use $|I|$ to denote the cardinality of the index set I and define $[n] := \{1, 2, ..., n\}$.

2. Background, Related Work and Contributions. In the CUR decomposition, it is important to get a good set of row and column indices as they dictate the quality of the low-rank approximation. There are many practical methods, but they largely fall into two different categories: pivoting or sampling. For pivoting based methods, we use pivoting schemes such as column pivoted QR (CPQR) [27] or LU with complete pivoting [51] on A or the singular vectors of A to obtain the pivots which we then use as row or column indices. For example, these can be used on the dominant singular vectors of A to obtain the pivots and the discrete empirical interpolation method (DEIM) is a popular example [11, 24, 49]. There are also pivoting schemes that provide a strong theoretical guarantee [29]. Applying pivoting schemes directly on A can be prohibitive for large matrices, and for this reason a number of randomized algorithms based on *sketching* have been proposed [20, 25, 54]. These methods "sketch" the original matrix down to a smaller-sized matrix using randomization and perform the pivoting schemes there. See [20] for a comparative study of randomized pivoting algorithms. On the other hand, for sampling based methods, we sample the column or row indices from some probability distribution obtained from certain information about A. For example, a popular choice is the row norms of the dominant singular vectors of A for leverage scores [23, 38]. There are other sampling strategies such as uniform sampling [12], volume sampling [13, 17, 18, 28], DPP sampling [16], and BSS sampling [5, 6]. In particular, volume sampling leads to a CUR approximation that have close-to-optimal error guarantees [13, 56]. Sampling based methods can also be prohibitive for large matrices, so a smaller-sized proxy of A via sketching have been proposed [22]. There is also a deterministic sampling method for leverage scores [46] and hybrid methods such as L-DEIM [26].

The vast majority of CUR decompositions of A comes with a theoretical guarantee that involves the term

$$\left\|V(J,:)^{-1}\right\|_2 = \frac{1}{\sigma_{\min}(V(J,:))}, \tag{2.1}$$

where $V \in \mathbb{R}^{n\times k}$ is the k (approximate) dominant right singular vectors of $A \in \mathbb{R}^{m\times n}$ and $J \subset [n]$ with $|J| = k$ is the set of column indices; see Theorems 3.3 and A.3. A similar term involving the left singular vectors and a row index set is also present. The term (2.1) is usually the deciding factor for the accuracy of the CUR decomposition and therefore, a majority of the algorithms aim at choosing a set of indices J that would diminish the effect of (2.1). A natural way of improving (2.1) is to *oversample*, that is, obtain extra indices $J_0 \in \mathbb{R}^p$ distinct from J so that $V(J \cup J_0, :) \in \mathbb{R}^{(k+p)\times k}$ becomes a rectangular matrix. By appending more rows to $V(J, :)$, $V(J \cup J_0, :)$ has a larger minimum singular value by the Courant-Fischer min-max theorem, which improves the accuracy of the CUR decomposition, as we will see in the later sections. The topic of oversampling is not new, in particular, it is shown in [1] that oversampling improves the accuracy of CURBA when the singular values decay rapidly. In the context of sampling based methods, oversampling has been suggested for theoretical guarantees; see e.g. [7, 12, 38]. It is easy to oversample for sampling based methods as

we can simply sample more than k indices from the given probability distribution. In contrast, it is often difficult to oversample for pivoting based methods. This is because we typically perform pivoting schemes on a smaller-sized surrogate $X \in \mathbb{R}^{n\times k}$, for example, the sketch of A and the pivots beyond the first k indices carry little to no information. Therefore, for pivoting based methods, usually a different strategy is used for oversampling. In the context of DEIM, various ways of oversampling have been suggested [9, 26, 47, 57]. Specifically, Zimmermann and Willcox [57] show that oversampling can improve the condition number of oblique projections, and Donello et al. [19] proves a bound for DEIM projectors[4] with oversampling, which extends the proof without oversampling from [49]. A majority of the aforementioned literature focuses on the DEIM projector or the CURBA. However, oversampling can often be more effective for the CURCA as we not only improve the bound involving (2.1) (see Theorem 3.3), we also make $A(I,J)$ a rectangular matrix when we oversample either I or J (but not both), which improves the condition number of $A(I,J)$ making the CURCA more accurate when computing the pseudoinverse of $A(I,J)$. Therefore, we advocate oversampling when possible over the standard choice $|I| = |J|$.

Existing work. The numerical stability of the stabilized CURCA, $A \approx CU_\epsilon^\dagger R$, involving the ϵ-pseudoinverse has not been studied previously to our knowledge. However, there are some related works exploring the idea of ϵ-pseudoinverse in the core matrix. The ϵ-pseudoinverse takes the core matrix $U = A(I,J)$, truncates its singular values that are less than ϵ, and computes the pseudoinverse of the resulting matrix. Firstly, Chiu and Demanet [12] study the ϵ-pseudoinverse in the context of the CURCA with uniform sampling and show that when the matrix is incoherent, the algorithm succeeds. However, this paper does not contain stability analysis, and has the condition that the matrix needs to be incoherent. The authors in [8, 44] explore the ϵ-pseudoinverse in the context of the symmetric Nyström method, $A \approx CU_\epsilon^\dagger C^T$, applied to symmetric indefinite matrices. However, they show that the ϵ-pseudoinverse can deteriorate the accuracy of the symmetric Nyström method when applied to symmetric indefinite matrices. Lastly, Nakatsukasa [42] studies the generalized Nyström algorithm[5] with ϵ-pseudoinverse and oversampling, which provides a numerically stable way of computing the generalized Nyström method and proves its stability. The paper also demonstrates that oversampling is necessary for a stable approximation in the generalized Nyström method.

In a related work, Hamm and Huang study the stability of sampling for CUR decompositions in [32] and the perturbation bounds of CUR decompositions in [33]. In [32], they study the problem of determining when a column submatrix of a rank k matrix A also has rank k. This is important as if the chosen columns or rows of a matrix is (numerically) rank-deficient then U is also (numerically) rank-deficient, which may cause numerical issues when computing the pseudoinverse of U. In [33], they derive perturbation bounds for CUR decompositions under the influence of a noise matrix. They investigate several variants of the CUR decomposition including the ϵ-pseudoinverse variant, SCURCA, and provide perturbation bounds in terms of the noise matrix. This work is related, but different from our work as they investigate the accuracy of the CUR decomposition for the perturbed matrix $\tilde{A} = A+E$, while we

[4]DEIM projector is given by $\mathcal{P}_U = U(\Pi^T U)^\dagger \Pi^T$ where $U \in \mathbb{R}^{n\times k}$ is an orthonormal matrix and $\Pi \in \mathbb{R}^{n\times k}$ is a submatrix of the identity matrix that picks k chosen rows.

[5]The generalized Nyström method is a variant of the CURCA where instead of subsets of rows and columns of A that approximate the row and column space of A, we have the sketches, $Y^T A$ and AX that approximate the row and column space of A where X and Y are random embeddings. See [42, 52] for more details.

investigate the accuracy and stability for a numerical implementation of the CURCA in the presence of rounding errors.

We review two existing algorithms for oversampling. First, Gidisu and Hochstenbach [26] oversample p extra indices by choosing the largest p leverage scores (row norms of (approximate) dominant singular vectors) out of the unchosen indices. The complexity is $\mathcal{O}(nk)$ for computing the leverage scores of an orthonormal matrix $V \in \mathbb{R}^{n\times k}$. If we are only given an approximator that is not orthonormal then (approximate) orthonormalization needs to be done, usually at a cost of $\mathcal{O}(nk^2)$. Second, Peherstorfer, Drmač and Gugercin [47], iteratively select p extra indices for oversampling to maximize the minimum singular value of $V(J,:)$ in a greedy fashion. This approach, which is called the `GappyPOD+E` algorithm, uses perturbation bounds on the eigenvalues given in [36] to find the next index that maximizes the lower bound for the minimum singular value of $V(J,:)$. This approach is also a special case of [57]. The algorithm runs with complexity $\mathcal{O}((k+p)^2k^2 + nk^2p)$ where p is the number of indices we oversample by. Again, if V is not orthonormal to begin with, then (approximate) orthonormalization needs to be done usually at a cost of $\mathcal{O}(nk^2)$. For a treatment of approximate orthonormalization, see, for example [3, 4].

Contributions. Our contribution is twofold: (1) presenting a method for computing the CURCA in a numerically stable manner with a theoretical guarantee, and (2) advocating the use of oversampling to improve the accuracy and stability of the CURCA.

Our first and main contribution lies in presenting a method for computing the CURCA in a numerically stable manner, accompanied by an analysis that guarantees its stability. We show that with the ϵ-pseudoinverse in the core matrix, the SCURCA, $A \approx CU_\epsilon^\dagger R$, can be computed in a numerically stable manner by taking the following steps. First, we compute each row of $CU_\epsilon^\dagger$ using a backward stable underdetermined linear solver. Then we compute the SCURCA by multiplying $CU_\epsilon^\dagger$ by R. See Section 3.2 for details. In addition to the stability analysis, we also analyze the CURCA and its ϵ-pseudoinverse variant, SCURCA in exact arithmetic by deriving a relative norm bound. While our analysis does not cover the stability of plain CURCA, $CU^\dagger R$, we observe its stability in practice without the ϵ-truncation; see Sections 3.2 and 5.1.

Our secondary contribution involves advocating the use of oversampling for the CURCA and for providing a deterministic algorithm to oversample row or column indices. We show that oversampling improves the accuracy and stability of the CURCA by providing a theoretical analysis that demonstrates the benefits of oversampling. We show that oversampling should be done such that it increases the minimum singular value(s) of $V(J,:)$ where $V \in \mathbb{R}^{n\times k}$ is the k (approximate) dominant right singular vectors of A and J is a set of indices with $|J| = k$. Our algorithm is motivated by the cosine-sine (CS) decomposition and runs with complexity $\mathcal{O}(nk^2 + nkp)$ where k is the target rank and p is the oversampling parameter. Note that this complexity only refers to the cost of the oversampling process, not the whole CUR process; the complexity of the initial CUR process is separate and depends on the method used to get the initial set of indices. We show that our algorithm is competitive with existing algorithms and in particular, performs similarly to the `GappyPOD+E` algorithm in [47], in which the oversampling process runs with complexity $\mathcal{O}((k+p)^2k^2 + nk^2p)$. The numerical experiments illustrate that oversampling is recommended.

3. Accuracy and stability of the stabilized CURCA.

In this section, we study two topics related to the CURCA, $A \approx CU^\dagger R$. We first analyze the accuracy

of the CURCA,

$$A_{IJ} = A(:,J)A(I,J)^\dagger A(I,:) = A\Pi_J(\Pi_I^T A\Pi_J)^\dagger \Pi_I^T A =: CU^\dagger R$$

and its ϵ-pseudoinverse variant, the SCURCA,

$$A_{IJ}^\epsilon = A(:,J)A(I,J)_\epsilon^\dagger A(I,:) = A\Pi_J(\Pi_I^T A\Pi_J)_\epsilon^\dagger \Pi_I^T A = CU_\epsilon^\dagger R,$$

and show that the ϵ-truncation in the core matrix compromises the accuracy of the CURCA only by ϵ times the condition number of the CURCA; see Remark 3.5. We then analyze the numerical stability of SCURCA in the presence of roundoff errors and show that SCURCA satisfies a similar bound under roundoff errors, making the SCURCA numerically stable as long as the selected rows and columns well-approximate the dominant row and column spaces of A; see Section 3.2.

We begin with some preliminaries: oblique projectors and standard assumptions. In the proofs below, we frequently use oblique projectors and their properties. We use $\mathcal{P}_{X,Y} := X(Y^TX)^\dagger Y^T$ where $X \in \mathbb{R}^{n\times k}$ and $Y \in \mathbb{R}^{n\times \ell}$ to denote an oblique projection onto the column space of X if $k \le \ell$ and Y^TX has full column rank or onto the row space of Y^T if $\ell \le k$ and Y^TX has full row rank. For example, the CUR decomposition A_{IJ} can be written as

$$A_{IJ} = A\Pi_J(\Pi_I^T A\Pi_J)^\dagger \Pi_I^T A = \mathcal{P}_{A\Pi_J,\Pi_I} A = A\mathcal{P}_{\Pi_J,A^T\Pi_I}.$$

Some of the important properties of projectors [50] are

1. $\mathcal{P}_{X,Y}\mathcal{P}_{X,Y} = \mathcal{P}_{X,Y}$,
2. $\mathcal{P}_{X,Y}X = X$ if Y^TX has full column rank,
3. $Y^T\mathcal{P}_{X,Y} = Y^T$ if Y^TX has full row rank,
4. $\|\mathcal{P}_{X,Y}\|_2 = \|I - \mathcal{P}_{X,Y}\|_2$ if $\mathcal{P}_{X,Y} \neq 0, I$.

Lastly, sometimes we can simplify the norm of oblique projectors, which is given by the lemma below.

LEMMA 3.1. *Let $\mathcal{P}_{X,Y} \in \mathbb{R}^{n\times n}$ be a projector where $X \in \mathbb{R}^{n\times k}$, $Y \in \mathbb{R}^{n\times \ell}$ and $Y^TX \in \mathbb{R}^{\ell\times k}$ all have full column rank (so $k \le \ell$). Then*

$$\|\mathcal{P}_{X,Y}\| = \left\|(Y^TQ_X)^\dagger Y^T\right\| \tag{3.1}$$

for any unitarily invariant norm $\|\cdot\|$ where Q_X is an orthonormal matrix spanning the columns of X.

Proof. Let $X = Q_XR_X$ be the thin QR decomposition of X. Then

$$\|\mathcal{P}_{X,Y}\| = \left\|X(Y^TX)^\dagger Y^T\right\| = \left\|Q_XR_X(Y^TQ_XR_X)^\dagger Y^T\right\| = \left\|(Y^TQ_X)^\dagger Y^T\right\|,$$

since $Y^TQ_X \in \mathbb{R}^{\ell\times k}$ has full column rank and $R_X \in \mathbb{R}^{k\times k}$ is nonsingular as Y^TX has full column rank. ∎

Now, we lay out some generic assumptions that hold in our theorems below. The assumptions are

Assumption 3.2.

1. $|I| = |J| = k \le \operatorname{rank}(A)$ where k is the target rank,
2. $A(I,J) \in \mathbb{R}^{k\times k}$ is a non-singular matrix,

3. $X(:, J) \in \mathbb{R}^{k\times k}$ has full row rank, where X is (any) row space approximator of A.[6]

When $\text{rank}(A) \le k$ and $\text{rank}(A(I, J)) = \text{rank}(A)$, we have $A = A_{IJ}$ [31]. Since $A(I, J)$ is assumed to be non-singular, $A(:, J)$ and $A(I, :)$ have full column and row rank, respectively. Under Assumption 3.2, by Lemma 3.1,

$$\left\|CU^\dagger\right\| = \left\|Q_C(I,:)^\dagger\right\|, \left\|U^\dagger R\right\| = \left\|Q_R(J,:)^\dagger\right\|, \left\|X(J,:)^\dagger X\right\| = \left\|Q_X(J,:)^\dagger\right\|$$

where Q_C, Q_R and Q_X are the orthonormal matrices spanning the columns of C, R^T and X^T, respectively. We now prove the accuracy of the CURCA and its ϵ-pseudoinverse variant, the SCURCA.

3.1. Accuracy of CUR and its ϵ-pseudoinverse variant. We prove the accuracy of the SCURCA, A^ϵ_{IJ} first. The accuracy for the CURCA, A_{IJ} follows by setting $\epsilon = 0$. The analysis presented in this section is a key contribution and plays an essential role for the stability analysis as we show that the SCURCA satisfies a similar bound under roundoff errors, establishing its numerical stability. The bound in Theorem 3.3 below somewhat resembles that in [21].

THEOREM 3.3. *Let $A \in \mathbb{R}^{m\times n}$ be a matrix, I and J be a set of row and column indices, respectively, with $|I| = |J| = k$, $\epsilon > 0$ and $X \in \mathbb{R}^{k\times n}$ be any row space approximator of A. Then under Assumption 3.2,*

$$\left\|A - A^\epsilon_{I\cup I_0,J}\right\| \le \left\|Q_C(I \cup I_0,:)^\dagger\right\|_2 \left\|Q_X(J,:)^{-1}\right\|_2 \left(\left\|A - AX^\dagger X\right\| + \|E\|\right) \tag{3.2}$$

for any unitarily invariant norm $\|\cdot\|$ where I_0 is a set of extra row indices distinct from I with $|I_0| = p$, and $E \in \mathbb{R}^{k_\epsilon\times k_\epsilon}$ is a matrix satisfying $\|E\|_2 \le \epsilon$ where $k_\epsilon \le k$ is the number of singular values of $A(I \cup I_0, J)$ smaller than ϵ.

Proof. For shorthand, let $I_* := I \cup I_0$. Let the thin SVD of $A(I_*, J) \in \mathbb{R}^{(k+p)\times k}$ be $W\Sigma V^T = [W_1, W_2]\,\text{diag}(\Sigma_1, \Sigma_2)[V_1, V_2]^T$ where Σ_2 contains the singular values of $A(I_*, J)$ smaller than ϵ. Then

$$\begin{aligned} A^\epsilon_{I_*J}\Pi_J &= A\Pi_J(\Pi^T_{I_*}A\Pi_J)^\dagger_\epsilon\Pi^T_{I_*}A\Pi_J = A\Pi_J V_1\Sigma_1^{-1}W_1^T W\Sigma V^T \\ &= A\Pi_J V_1V_1^T = A\Pi_J - A\Pi_J V_2V_2^T. \end{aligned}$$

Therefore,

$$\begin{aligned} A - A^\epsilon_{I_*J} &= \left(I - A\Pi_J\left(\Pi^T_{I_*}A\Pi_J\right)^\dagger_\epsilon\Pi^T_{I_*}\right)A \\ &= (I - A\Pi_J(\Pi^T_{I_*}A\Pi_J)^\dagger_\epsilon\Pi^T_{I_*})A(I - \Pi_J(X\Pi_J)^\dagger X) + A\Pi_J V_2V_2^T(X\Pi_J)^\dagger X. \end{aligned} \tag{3.3}$$

Note that $\mathcal{P}^\epsilon_{A\Pi_J,\Pi_{I_*}} := A\Pi_J(\Pi^T_{I_*}A\Pi_J)^\dagger_\epsilon\Pi^T_{I_*}$ is an oblique projector since

$$\begin{aligned} (\mathcal{P}^\epsilon_{A\Pi_J,\Pi_{I_*}})^2 &= A\Pi_J V_1\Sigma_1^{-1}W_1^T W\Sigma V^T V_1\Sigma_1^{-1}W_1^T\Pi^T_{I_*} \\ &= A\Pi_J V_1\Sigma_1^{-1}W_1^T\Pi^T_{I_*} \\ &= \mathcal{P}^\epsilon_{A\Pi_J,\Pi_{I_*}} \end{aligned}$$

[6] The assumptions and the theorems in this section are stated in terms of row space approximators, but similar assumptions and theorems for column space approximators can be obtained, for example, by considering A^T instead.

and similarly, $\mathcal{P}_{\Pi_J,X^T} = \Pi_J(X\Pi_J)^\dagger X$ is an oblique projector. Now bounding the first term of (3.3) gives

$$\begin{aligned}\left\|(I-\mathcal{P}^\epsilon_{A\Pi_J,\Pi_{I_*}})A(I-\mathcal{P}_{\Pi_J,X^T})\right\| &\le \left\|I-\mathcal{P}^\epsilon_{A\Pi_J,\Pi_{I_*}}\right\|_2 \left\|A(I-\mathcal{P}_{\Pi_J,X^T})\right\| \\ &= \left\|\mathcal{P}^\epsilon_{A\Pi_J,\Pi_{I_*}}\right\|_2 \left\|A(I-X^\dagger X)(I-\mathcal{P}_{\Pi_J,X^T})\right\| \\ &\le \left\|\mathcal{P}^\epsilon_{A\Pi_J,\Pi_{I_*}}\right\|_2 \left\|A(I-X^\dagger X)\right\| \left\|I-\mathcal{P}_{\Pi_J,X^T}\right\|_2 \\ &= \left\|\mathcal{P}^\epsilon_{A\Pi_J,\Pi_{I_*}}\right\|_2 \left\|\mathcal{P}_{\Pi_J,X^T}\right\|_2 \left\|A(I-X^\dagger X)\right\|\end{aligned}$$

where in the first equality we used $X\mathcal{P}_{\Pi_J,X^T} = X\Pi_J(X\Pi_J)^\dagger = I$, as $X\Pi_J = X(:,J)$ has full row rank by Assumption 3.2. The first term $\left\|\mathcal{P}^\epsilon_{A\Pi_J,\Pi_{I_*}}\right\|_2$ in the final expression can be bounded by letting $A\Pi_J = Q_C R_C$ be the thin QR decomposition and noting that $\Pi_{I_*}^T Q_C$ has full column rank, as

$$\begin{aligned}\left\|\mathcal{P}^\epsilon_{A\Pi_J,\Pi_{I_*}}\right\|_2 &= \left\|Q_C R_C(\Pi_{I_*}^T A\Pi_J)^\dagger_\epsilon\right\|_2 = \left\|R_C(\Pi_{I_*}^T A\Pi_J)^\dagger_\epsilon\right\|_2 \\ &= \left\|(\Pi_{I_*}^T Q_C)^\dagger \Pi_{I_*}^T Q_C R_C(\Pi_{I_*}^T A\Pi_J)^\dagger_\epsilon\right\|_2 \\ &\le \left\|(\Pi_{I_*}^T Q_C)^\dagger\right\|_2 \left\|\Pi_{I_*}^T A\Pi_J(\Pi_{I_*}^T A\Pi_J)^\dagger_\epsilon\right\|_2 \\ &\le \left\|(\Pi_{I_*}^T Q_C)^\dagger\right\|_2.\end{aligned}$$

Therefore, using Lemma 3.1 on $\mathcal{P}_{\Pi_J,X^T}$, the first term in (3.3) can be bounded as

$$\left\|(I-\mathcal{P}^\epsilon_{A\Pi_J,\Pi_{I_*}})A(I-\mathcal{P}_{\Pi_J,X^T})\right\| \le \left\|Q_C(I_*,:)^\dagger\right\|_2 \left\|Q_X(J,:)^{-1}\right\|_2 \left\|A(I-X^\dagger X)\right\|.$$

The second term in (3.3) can be bounded using a similar argument as

$$\begin{aligned}\left\|A\Pi_J V_2 V_2^T(X\Pi_J)^\dagger X\right\| &= \left\|Q_C R_C V_2 V_2^T(X\Pi_J)^\dagger X\right\| = \left\|R_C V_2 V_2^T(X\Pi_J)^\dagger X\right\| \\ &= \left\|(\Pi_{I_*}^T Q_C)^\dagger \Pi_{I_*}^T Q_C R_C V_2 V_2^T(X\Pi_J)^\dagger X\right\| \\ &= \left\|(\Pi_{I_*}^T Q_C)^\dagger W_2\Sigma_2 V_2^T(X\Pi_J)^\dagger X\right\| \\ &\le \left\|(\Pi_{I_*}^T Q_C)^\dagger\right\|_2 \left\|W_2\Sigma_2 V_2^T\right\| \left\|(X\Pi_J)^\dagger X\right\|_2 \\ &= \left\|Q_C(I_*,:)^\dagger\right\|_2 \left\|Q_X(J,:)^{-1}\right\|_2 \left\|\Sigma_2\right\|.\end{aligned}$$

Putting everything together and letting $E = \Sigma_2$, we get the desired result. ∎

COROLLARY 3.4. *Under the same assumptions as in Theorem 3.3,*

$$\|A - A_{I\cup I_0,J}\| \le \left\|Q_C(I\cup I_0,:)^\dagger\right\|_2 \left\|Q_X(J,:)^{-1}\right\|_2 \left\|A - AX^\dagger X\right\| \tag{3.4}$$

for any unitarily invariant norm $\|\cdot\|$.

Proof. Set $\epsilon = 0$ in Theorem 3.3. ∎

Remark 3.5.

1. The condition number of the CURCA is $\kappa = \left\|Q_C(I\cup I_0,:)^\dagger\right\|_2 \left\|Q_X(J,:)^{-1}\right\|_2$, as indicated by (3.2). Theorem 3.3 tells us that the SCURCA, $A^\epsilon_{I_*J}$ is worse than the CURCA by at most a factor $\kappa\sqrt{k}\epsilon$ in the Frobenius norm.
2. The bound in Theorem 3.3 and Corollary 3.4 has two factors involving the (pseudo)inverse, which is in contrast to the CURBA, $C(C^\dagger A R^\dagger)R$ (see Appendix A) having only one factor. This makes the CURCA, A_{IJ} usually worse

than the CURBA, which is expected as the CURCA is cheaper to compute. However, the CURCA can still be very accurate; see for example [13, 56], which establishes the existence of a rank-r CURCA that has error within a factor $r+1$ of the best rank-r approximation via the truncated SVD. The second multiplicative factor in Theorem 3.3 and Corollary 3.4 comes from the fact that A_{IJ} is associated with the oblique projector $\mathcal{P}_{A\Pi_J,\Pi_I}$ rather than the orthogonal projectors $CC^\dagger$ and $R^\dagger R$ for the CURBA. Nonetheless, the CURCA and the CURBA have comparable accuracy when employed with good row and column indices and oversampling, with the CURCA being much more computationally efficient.

3. The row space approximator $X \in \mathbb{R}^{k\times n}$ can be chosen in various ways, for example, the k-dominant right singular vectors of A or the row sketch of A. The bounds in Theorem 3.3 and Corollary 3.4 can both be computed a posteriori when X can be computed easily. The k-dominant right singular vectors would make the right-most term, $\left\|A - AX^\dagger X\right\|$ in the bound of Theorem 3.3 and Corollary 3.4, optimal. However, the singular vectors are often too expensive to compute.

Theorem 3.3 and Corollary 3.4 provide a bound for the SCURCA and the CURCA, respectively. It also demonstrates the benefit of oversampling through the extra set of row indices I_0. We obtain a pseudoinverse for $\left\|Q_C(I\cup I_0,:)^\dagger\right\|_2$ instead of the matrix inverse $\left\|Q_C(I,:)^{-1}\right\|_2$ with the former always being smaller. Additionally, since $\left\|Q_C(I\cup I_0,:)^\dagger\right\|_2 = \left\|A(:,J)A(I_*,J)^\dagger\right\|_2$ and we want to minimize this quantity, the row indices I and I_0 should be chosen in terms of the already-chosen columns of A. This has been suggested and employed in other works such as [14, 20, 54, 55]. This comes with the benefit that the resulting core matrix $A(I_*,J)$ will generally be better-conditioned, improving the accuracy of the CURCA. For example, consider the following 2×2 matrix,

$$A = \begin{bmatrix} \epsilon & 1 \\ 1 & 0 \end{bmatrix}$$

where $0<\epsilon<1$. For a rank-1 approximation of A, we need to choose a column and a row for the CURCA. If we choose a column and a row separately in the best possible way, we would choose the first row and the first column, giving us

$$A_{1,1} = \begin{bmatrix} \epsilon \\ 1 \end{bmatrix} \epsilon^{-1} \begin{bmatrix} \epsilon & 1 \end{bmatrix} = \begin{bmatrix} \epsilon & 1 \\ 1 & 1/\epsilon \end{bmatrix},$$

which is a poor approximation as $\|A - A_{1,1}\|_F = 1/\epsilon$ can be arbitrary large[7] as $\epsilon \to 0$. On the other hand, if we choose a column first and then a row, we choose the first column $[\epsilon, 1]^T$ and the second row as $\epsilon < 1$, giving us

$$A_{2,1} = \begin{bmatrix} \epsilon \\ 1 \end{bmatrix} 1^{-1} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} \epsilon & 0 \\ 1 & 0 \end{bmatrix},$$

which is a reasonable approximation as $\|A - A_{2,1}\|_F = 1 \approx \sigma_2(A) \approx 1 - \epsilon/2$.

The significance of controlling the $\left\|Q_C(I,:)^{-1}\right\|_2$ term will also be highlighted when we analyze the numerical stability of the CUR decomposition in the subsequent section.

[7] The accuracy of CURBA, by contrast, is good as long as the chosen columns and rows are good approximators for the range and co-range of A [20, Remark 1].

3.2. Numerical Stability of the CUR decomposition. In the absence of rounding errors, the error for the CURCA can be bounded by Theorem 3.3 and Corollary 3.4. In this section, we derive an error bound that accounts for rounding errors.

We use the standard model of floating-point arithmetic as in [34, Section 2.2]:

$$fl(x \text{ op } y) = (x \text{ op } y)(1+\delta),\ |\delta| \le u$$

where $\text{op} \in \{+,-,*,/\}$ is the basic arithmetic operations and $u \ll 1$, the unit roundoff, is the precision at which the computations are performed. We use $fl(\cdot)$ and $\widehat{\cdot}$ to denote the computed value of the expression. We define $\gamma := p(m,n,k)u$ where $p(m,n,k)$ is a low-degree polynomial in m,n and k, and use γ to suppress any constant factors and terms related to the size of the matrix and the target rank, e.g., $\sqrt{m}, \sqrt{n}$ and k, but not $\sigma_i(A)$ or $1/\epsilon$. While this may appear as an oversimplication, this approach is standard practice in stability analysis; see e.g. [42, 43].

In this section, we denote the ith row of a matrix B as $[B]_i$ and we use $C = A(:,J)$, $U = A(I_*,J)$ and $R = A(I,:)$ for shorthand. We assume that the rows are oversampled, i.e., $I_* = I \cup I_0$ is such that the number of the oversampling indices I_0 is bounded by a constant times k where $k = |I| = |J|$ is the target rank, and the truncation parameter ϵ satisfies $\|A\|_2 \gg \epsilon > \gamma \|A\|_2$. We begin by stating two lemmas that will be used in the CURCA stability analysis. Lemma 3.6 proves the perturbation bound for the projector $CU_\epsilon^\dagger \Pi_J$ when only C and U get perturbed and in Lemma 3.7, we prove that under perturbation on C and U, $\tilde{C}\tilde{U}_\epsilon^\dagger \Pi_J$ approximately projects C onto itself. The proofs for the two lemmas can be found in Appendix C.

LEMMA 3.6. *Under Assumption 3.2, for any ΔC and ΔU,*

$$\left\|(C+\Delta C)(U+\Delta U)_\epsilon^\dagger\right\|_2 \le \left\|Q_C(I_*,:)^\dagger\right\|_2 \left(1 + \frac{1}{\epsilon}\|\Delta U\|_2\right) + \frac{1}{\epsilon}\|\Delta C\|_2 . \tag{3.5}$$

LEMMA 3.7. *Under Assumption 3.2, for any ΔC and ΔU,*

$$(C+\Delta C)(U+\Delta U)_\epsilon^\dagger R \Pi_J = C + E_* \tag{3.6}$$

where

$$\|E_*\|_2 \le \left\|(Q_C(I_*,:)^\dagger\right\|_2 \left(\epsilon + 2\|\Delta U\|_2 + \frac{1}{\epsilon}\|\Delta U\|_2^2\right) + \|\Delta C\|_2 \left(1 + \frac{\|\Delta U\|_2}{\epsilon}\right).$$

We now begin with the stability analysis. Note that the stability depends on the specific implementation used. In the forthcoming analysis, we assume the SCURCA $A^\epsilon_{I_* J}$ is computed as follows.[8]

1. Compute the factors $\hat{C} = fl(A\Pi_J), \hat{U} = fl(\Pi_{I_*}^T A \Pi_J)$ and $\hat{R} = fl(\Pi_{I_*}^T A)$.
2. Solve the (rank-deficient) underdetermined linear systems,

$$\hat{s}_i^{(1)} = fl\left((\hat{U}^T)_\epsilon^\dagger [\hat{C}]_i^T\right) \in \mathbb{R}^{k+p}$$

for all $i \in [m]$, where $[\hat{C}]_i$ is the ith row of $\hat{C}$.

[8] Note that the stability crucially depends on the implementation. Other implementations are possible, an obvious one being one that computes $C(U^\dagger R)$ rather than $(CU^\dagger)R$ as done here. This is seen to work well too, although we do not have a proof.

3. Compute matrix-vector multiply $\hat{s}_i^{(2)} = fl\left(\hat{R}^T \hat{s}_i^{(1)}\right) \in \mathbb{R}^n$.

4. Let $\left(\hat{s}_i^{(2)}\right)^T$ be the ith row of the computed CUR decomposition $fl\left(A^{\epsilon}_{I_* J}\right)$.

We now analyze each step. The first step is matrix-matrix multiplications with orthonormal matrices. Using the forward error bound[9] for matrix-matrix multiplication [34, Section 3.5], we obtain the following:

- $\hat{C} = C + E_C$ where $\|E_C\|_2 \leq \gamma \|A\|_2$,
- $\hat{U} = U + E_U$ where $\|E_U\|_2 \leq \gamma \|A\|_2$,
- $\hat{R} = R + E_R$ where $\|E_R\|_2 \leq \gamma \|A\|_2$.

In many cases, these error matrices E_C, E_U and E_R are the zeros matrix as C, U and R are simply submatrices of the original matrix A.

In the second step, we solve the (rank-deficient) underdetermined linear systems row by row. The error analysis for the (rank-deficient) underdetermined linear systems can be summarized in the following theorem. The proof of Theorem 3.8 can be found in Appendix B.

THEOREM 3.8. *Consider the (rank-deficient) underdetermined linear system,*

$$\min_{x'} \|B_\epsilon x' - b\|_2 \tag{3.7}$$

where $B_\epsilon \in \mathbb{R}^{m\times n}$ $(m \leq n)$ is (possibly) rank-deficient $(\mathrm{rank}(B_\epsilon) \leq m)$ *with singular vales larger than ϵ and $b \in \mathbb{R}^m$. Then assuming $\epsilon > \gamma \|B_\epsilon\|_2$, the minimum norm solution to (3.7) can be computed in a backward stable manner, i.e., the computed solution $\hat{s}$ satisfies*

$$\hat{s} = (B_\epsilon + E_1)^\dagger (b + E_2) \tag{3.8}$$

where $\|E_1\|_2 \leq \gamma \|B_\epsilon\|_2$ and $\|E_2\|_2 \leq \gamma \|b\|_2$.

Theorem 3.8 tells us that the computed solution to a (rank-deficient) underdetermined linear system is the exact solution to a slightly perturbed problem. Now, using Theorem 3.8, for each $i \in [m]$ we obtain

$$\hat{s}_i^{(1)} = \left((\hat{U}^T)_\epsilon + E_i^{(U)}\right)^\dagger \left([\hat{C}]_i^T + E_i^{(C)}\right), \tag{3.9}$$

where $\left\|E_i^{(U)}\right\|_2 \leq \gamma \|A\|_2$ and $\left\|E_i^{(C)}\right\|_2 \leq \gamma \|A\|_2$. It is worth emphasizing that the backward errors $E_i^{(C)}$, $E_i^{(U)}$ depend on i. Now since $\epsilon > \gamma \|A\|_2$ by assumption and $\sigma_{\min}\left(\hat{U}_\epsilon^T + E_i^{(U)}\right) \geq \epsilon - \gamma \|A\|_2$ by Weyl's inequality, there exists a perturbation $E_i \in \mathbb{R}^{k\times(k+p)}$ with $\|E_i\|_2 \leq \epsilon + \gamma \|A\|_2$[10] such that

$$\hat{s}_i^{(1)} = \left(\hat{U}^T + E_i\right)^\dagger_{\epsilon - \gamma\|A\|_2} \left([\hat{C}]_i^T + E_i^{(C)}\right). \tag{3.10}$$

For shorthand, let $\hat{S}$ be a matrix with its ith row equal to $\left(\hat{s}_i^{(1)}\right)^T$.

[9] The error is termed the forward error as it indicates how close the computed version $\hat{C}$ is to the exact version C [34, Section 1.5].

[10] If $\hat{U}^T = W_1 \Sigma_1 V_1^T + W_2 \Sigma_2 V_2^T$ is the SVD of $\hat{U}^T$ where Σ_2 contains the singular values of $\hat{U}^T$ smaller than ϵ, then we can take $E_i = -W_2 \Sigma_2 V_2^T + E_i^{(U)}$ for each i.

In the third step, we compute a matrix-vector product [34, Section 3.5] with $\hat{R}^T$, which gives us $\hat{s}_i^{(2)} = fl\left(\hat{R}^T \hat{s}_i^{(1)}\right) = \hat{R}^T \hat{s}_i^{(1)} + E_{s_i}$ with

$$\begin{aligned}
\|E_{s_i}\|_2 &\le \gamma \left\|\hat{R}^T\right\|_2 \left\|\hat{s}_i^{(1)}\right\|_2 \\
&\le \gamma \|A\|_2 \left(\left\|Q_C(I_*,:)^\dagger\right\|_2 \left(1 + \frac{\|E_i\|_2 + \|E_U\|_2}{\epsilon - \gamma\|A\|_2}\right) + \frac{\left\|E_i^{(C)}\right\|_2 + \|E_C\|_2}{\epsilon - \gamma\|A\|_2}\right) \\
&\le \gamma \|A\|_2 \left(\left\|Q_C(I_*,:)^\dagger\right\|_2 \left(1 + \frac{(\epsilon + \gamma\|A\|_2) + \gamma\|A\|_2}{\epsilon - \gamma\|A\|_2}\right) + \frac{\gamma\|A\|_2 + \gamma\|A\|_2}{\epsilon - \gamma\|A\|_2}\right) \\
&\le \gamma \|A\|_2 \left\|Q_C(I_*,:)^\dagger\right\|_2, \qquad (3.11)
\end{aligned}$$

where Lemma 3.6 was used in the penultimate line with $\Delta U = E_U + E_i^T$ and $\Delta C = E_C + e_i E_i^{(C)}$ where $e_i \in \mathbb{R}^m$ is the ith canonical basis vector. In the last line of (3.11), we used the fact that γ suppresses any low-degree polynomial in m, n and k, and $\epsilon > \gamma\|A\|_2$.

The following shows the expression for $\hat{s}_i^{(2)}$,

$$\begin{aligned}
\hat{s}_i^{(2)} &= \hat{R}^T \hat{s}_i^{(1)} + E_{s_i} = \hat{R}^T \left(\hat{U}^T + E_i\right)^\dagger_{\epsilon - \gamma\|A\|_2} \left([\hat{C}]_i^T + E_i^{(C)}\right) + E_{s_i} \\
&= (R + E_R)^T \left(U^T + E_U^T + E_i\right)^\dagger_{\epsilon - \gamma\|A\|_2} \left([C + E_C]_i^T + E_i^{(C)}\right) + E_{s_i}.
\end{aligned}$$

Finally, in the fourth step, we combine $\hat{s}_i^{(2)} \in \mathbb{R}^n$ into a matrix to form $\widehat{A^\epsilon_{I_* J}} = fl\left(A^\epsilon_{I_* J}\right) \in \mathbb{R}^{m \times n}$ by setting the ith row of $\widehat{A^\epsilon_{I_* J}}$ to be $\left(\hat{s}_i^{(2)}\right)^T$, giving us

$$fl\left(A^\epsilon_{I_* J}\right) = \hat{S}\hat{R} + E \qquad (3.12)$$

where $E \in \mathbb{R}^{m \times n}$ is a matrix with its ith row equal to $E_{s_i}^T$ and satisfies $\|E\|_2 \le \gamma\|A\|_2 \left\|Q_C(I_*,:)^\dagger\right\|_2$, since γ suppresses any low-degree polynomial in m.

We now state the main stability result of the CURCA with the ϵ-pseudoinverse, $A^\epsilon_{I_* J}$.

THEOREM 3.9. *Let $0 < \epsilon \ll 1$ be a truncation parameter for the pseudoinverse such that $\epsilon > \gamma\|A\|_2$. Suppose that $A^\epsilon_{I_* J} = A(:,J)A(I_*,J)^\dagger_\epsilon A(I_*,:)$ is computed in the following order:*

1. *Compute $C = A(:,J)$, $U = A(I_*,J)$ and $R = A(I_*,:)$,*
2. *Compute each row of $CU^\dagger_\epsilon$ using a backward stable (rank-deficient) underdetermined linear solver,*
3. *Compute $CU^\dagger_\epsilon$ times R. Let $fl\left(A^\epsilon_{I_* J}\right)$ denote the output.*

Then under Assumption 3.2,

$$\begin{aligned}
\left\|A - fl\left(A^\epsilon_{I_* J}\right)\right\|_F \le\ & 4\sqrt{m}\left\|Q_C(I_*,:)^\dagger\right\|_2 \left\|Q_X(J,:)^\dagger\right\|_2 \left(\left\|A(I - X^\dagger X)\right\|_F + 2\epsilon\right) \\
& + \gamma\|A\|_2 \left\|Q_C(I_*,:)^\dagger\right\|_2. \qquad (3.13)
\end{aligned}$$

Theorem 3.9 tells us that the bound for the computed version of the stabilized CURCA $\widehat{A^\epsilon_{IJ}}$ is at most a factor $\mathcal{O}(\sqrt{m})$ worse than its exact arithmetic counterpart A^ϵ_{IJ} plus an error of $\gamma\|A\|_2 \left\|Q_C(I_*,:)^\dagger\right\|_2$. More specifically, we have

$$\left\|A - fl(A^\epsilon_{I_* J})\right\|_F \le 4\sqrt{m}\,(\text{Bound (3.2)}) + \gamma\|A\|_2 \left\|Q_C(I_*,:)^\dagger\right\|_2.$$

Roughly, this shows that the computed error is in the same order as the error in exact arithmetic, up to a factor $\mathcal{O}(\sqrt{m})$. The factor $\mathcal{O}(\sqrt{m})$ is likely an artifact of the proof given below; in stability analysis it is common to see bounds with such overestimates, and also standard practice to expect to observe much better performance in practice. Throughout various parts of the proof, we loosely bound the 2-norm by the Frobenius norm, typically because we only have information about the norms of rows or columns of a matrix. For example in the proof of Theorem 3.9, we bound $\left\|\hat{S}\right\|_2 \leq \left\|\hat{S}\right\|_F$ in (3.15), which is likely a pessimistic overestimate.

Proof of Theorem 3.9. From the above analysis, we have

$$fl\left(A^{\epsilon}_{I_* J}\right) = \hat{S}\hat{R} + E = \hat{S}R + \hat{S}E_R + E, \tag{3.12}$$

where $\|E\|_2 \leq \gamma \|A\|_2 \left\|Q_C(I_*,:)^\dagger\right\|_2$. Let us apply Lemma 3.7 to each row of $\hat{S}R$ to obtain

$$\hat{S}_i R\Pi_J = \left([C+E_C]_i + \left(E_i^{(C)}\right)^T\right)(U + E_U + E_i^T)^\dagger_{\epsilon-\gamma\|A\|_2} R\Pi_J = [C]_i + E_i^{(P)},$$

where $\left\|E_i^{(P)}\right\|_2 \leq 8\epsilon \left\|Q_C(I_*,:)^\dagger\right\|_2$. Therefore, letting $E^{(P)}$ be a matrix with $E_i^{(P)}$ as its ith row, we get

$$\hat{S}R\Pi_J = C + E^{(P)},$$

where $\left\|E^{(P)}\right\|_F \leq 8\epsilon\sqrt{m}\left\|Q_C(I_*,:)^\dagger\right\|_2$. Now proceeding similarly to the proof of Theorem 3.3, we obtain

$$\begin{aligned} A - \hat{S}R &= \left(I - \hat{S}\Pi_{I_*}^T\right)A = \left(I - \hat{S}\Pi_{I_*}^T\right)A(I - \Pi_J(X\Pi_J)^\dagger X) - E^{(P)}(X\Pi_J)^\dagger X \\ &= \left(I - \hat{S}\Pi_{I_*}^T\right)A(I - X^\dagger X)(I - \Pi_J(X\Pi_J)^\dagger X) - E^{(P)}(X\Pi_J)^\dagger X, \end{aligned} \tag{3.14}$$

where $E^{(P)}(X\Pi_J)^\dagger X$ satisfies

$$\left\|E^{(P)}(X\Pi_J)^\dagger X\right\|_F \leq \left\|E^{(P)}\right\|_F \left\|(X\Pi_J)^\dagger X\right\|_2 \leq 8\epsilon\sqrt{m}\left\|Q_C(I_*,:)^\dagger\right\|_2 \left\|Q_X(J,:)^\dagger\right\|_2.$$

The first term in (3.14) can be bound by

$$\begin{aligned} &\left\|I - \hat{S}\Pi_{I_*}^T\right\|_2 \left\|A(I - X^\dagger X)\right\|_F \left\|I - \Pi_J(X\Pi_J)^\dagger X\right\|_2 \\ &\qquad \leq \left(1 + \left\|\hat{S}\right\|_2\right)\left\|A(I - X^\dagger X)\right\|_F \left\|\Pi_J(X\Pi_J)^\dagger X\right\|_2 \\ &\qquad \leq 4\sqrt{m}\left\|Q_C(I_*,:)^\dagger\right\|_2 \left\|Q_X(J,:)^\dagger\right\|_2 \left\|A(I - X^\dagger X)\right\|_F \end{aligned}$$

where in the final line we used (3.11), noting that

$$\left\|\hat{S}\right\|_2 \leq \left\|\hat{S}\right\|_F \leq \sqrt{\sum_{i=1}^m \left\|\hat{s}_i^{(1)}\right\|_2^2} \leq 3\sqrt{m}\left\|Q_C(I_*,:)^\dagger\right\|_2. \tag{3.15}$$

Using (3.15), we can also bound

$$\left\|\hat{S}E_R\right\|_F \leq \left\|\hat{S}\right\|_F \|E_R\|_2 \leq \gamma\|A\|_2\left\|Q_C(I_*,:)^\dagger\right\|_2.$$

Finally putting everything together, we obtain

$$\begin{aligned}\left\|A - fl(A^{\epsilon}_{I_* J})\right\|_F &= \left\|A - \hat{S}R - \hat{S}E_R - E\right\|_F \\ &\leq 4\sqrt{m}\left\|Q_C(I_*,:)^\dagger\right\|_2 \left\|Q_X(J,:)^\dagger\right\|_2 \left(\left\|A(I - X^\dagger X)\right\|_F + 2\epsilon\right) \\ &\quad + \gamma \left\|A\right\|_2 \left\|Q_C(I_*,:)^\dagger\right\|_2 . \end{aligned}$$ ∎

It is natural to wonder what could go wrong without the ϵ-pseudoinverse. Two problems may arise without the ϵ-pseudoinverse. First, the matrix U may not be numerically full rank, so Theorem 3.8 cannot be used as $\epsilon > \gamma \left\|A\right\|_2$ may no longer hold. In addition, the two lemmas, Lemma 3.6 and Lemma 3.7 become meaningless as both require division by ϵ. Without the ϵ-pseudoinverse, the error bound can become uncontrollably large, causing issues in several places in the proof of Theorem 3.9. For example, the bound for the error $\|E_{s_i}\|_2$ in (3.11) for computing the matrix-vector product and the bound for $\left\|E_i^{(P)}\right\|_2$ in the proof of Theorem 3.9 may no longer hold as they can become arbitrarily large without the ϵ-pseudoinverse. Nevertheless, in practice we observe stability without the ϵ-truncation, so we recommend a careful implementation of the pseudoinverse (without the ϵ-truncation) for practical purposes.[11] A similar observation, commenting on the role of the ϵ-pseudoinverse, has been mentioned in [42, Section 4.2]. See Section 5.1 for numerical experiments.

In Theorem 3.9, one might question how large $\left\|Q_C(I_*,:)^\dagger\right\|_2$ can be, given that it is part of the added term, $\gamma \left\|A\right\|_2 \left\|Q_C(I_*,:)^\dagger\right\|_2$ in Theorem 3.9. This could pose a problem if $\left\|Q_C(I_*,:)^\dagger\right\|_2$ grows exponentially in m or n. However, Theorem 3.9 is enough to conclude that A^{ϵ}_{IJ} is numerically stable when the indices are chosen reasonably. This means that we first choose the column indices J, and sensibly select the row indices I_* from the the selected columns $A(:,J)$ such that $\left\|A(:,J)A(I_*,J)^\dagger\right\|_2 = \left\|Q_C(I_*,:)^\dagger\right\|_2$ is bounded by a low-degree polynomial involving m, n and k; see Section 5.2. A similar approach is also employed in other works such as [14, 20, 54, 55]. For example, Gu-Eisenstat's strong rank-revealing QR factorization [29] can be used on the chosen columns $A(:,J)$ to obtain $\left\|Q_C(I,:)^{-1}\right\|_2 \leq \sqrt{mk}$, which can be reduced further with oversampling. This also highlights the importance of oversampling. In the absence of oversampling, poorly selected indices can make $\left\|Q_C(I,:)^{-1}\right\|_2$ exponentially large. Therefore, oversampling can be employed to stabilize the CURCA, which we discuss further in the following section.

4. Oversampling for the CURCA. In the previous section, we proved theoretical results involving the CURCA (Corollary 3.4), the stabilized CURCA (Theorem 3.3) and the stabilized CURCA in the presence of rounding errors (Theorem 3.9). All of the results involved an oversampling parameter p and an extra set of row indices I_0 and bounding $\left\|Q_C(I \cup I_0,:)^\dagger\right\|_2$ was important for the accuracy and stability of the CURCA. In this section, we discuss oversampling in the context of the CURCA and devise an algorithm that naturally arises from the discussion.

We first describe the setting. Suppose we have obtained the row indices I and the column indices J with $|I| = |J| = k$ by applying some algorithm, for example, the ones discussed in the introduction (Section 1), on a row space approximator $X \in \mathbb{R}^{k\times n}$ of

[11] To be clear, implementing the ϵ-truncation does not increase the complexity and can be recommended for guaranteed stability. We have simply not observed instability without the ϵ-truncation.

$A \in \mathbb{R}^{m\times n}$.[12] To have a concrete algorithm in mind for getting the set of indices I and J, we present pivoting on a random sketch [20, 25, 54] below in Algorithm 4.1.

Algorithm 4.1 Pivoting on a random sketch ([20, Algorithm 1])

Require: $A \in \mathbb{R}^{m\times n}$ of rank r, target rank $k \le r$ (typically $k \ll \min\{m,n\}$)
Ensure: Column indices J and row indices I with $|I| = |J| = k$

`function` $[I, J] =$ `Rand_Pivot`(A, k)
1: Draw a random embedding $\Omega \in \mathbb{R}^{k\times m}$.
2: Set $X = \Omega A \in \mathbb{R}^{k\times n}$, a row sketch of A
3: Apply CPQR on X. Let J be the k column pivots.
4: Apply CPQR on $A(:, J)^T$. Let I be the k row pivots.

Algorithm 4.1 is a version of Algorithm 1 from [20], which selects the column indices first by applying column pivoted QR (CPQR)[13] on the row sketch $X = \Omega A$ and then selects the row indices by applying CPQR on the chosen columns $A(:, J)^T$. Algorithm 4.1 is an example where we obtain the row indices from the already-chosen column indices, which was recommended in the previous section. Here, the row space approximator is the row sketch $X = \Omega A$ and the column space approximator is the columns $A(:, J)$. A bound for the CURCA (see Corollary 3.4) without oversampling is given by

$$\|A - A_{IJ}\|_F \le \left\|Q_C(I,:)^{-1}\right\|_2 \left\|Q_X(J,:)^{-1}\right\|_2 \left\|A - AX^\dagger X\right\|_F \tag{4.1}$$

where $A_{IJ} = A(:, J)A(I, J)^\dagger A(I,:)$ is the CURCA and Q_C and Q_X are orthonormal matrices spanning the columns of C and X^T respectively.

Many existing algorithms focus on minimizing the first two terms on the right-hand side of (4.1) as they control the accuracy of the CURCA. In the CURCA and the other CUR decompositions such as the CURBA, $C(C^\dagger AR^\dagger)R$, we often take $|I| = |J|$, however, without increasing the overall rank of the approximation, we can oversample either I or J.[14] Suppose we oversample the rows to $I_* := I \cup I_0$ where $|I_0| = p$ is an extra set of row indices for oversampling. Then the first term of the bound changes from $\left\|Q_C(I,:)^{-1}\right\|_2$ to $\left\|Q_C(I_*,:)^\dagger\right\|_2$ (see Corollary 3.3). Now $\sigma_{\min}(Q_C(I_*,:)) \ge \sigma_{\min}(Q_C(I,:))$ by the Courant-Fischer min-max theorem, which improves the bound in (4.1) as $\left\|Q_C(I_*,:)^\dagger\right\|_2 \le \left\|Q_C(I,:)^{-1}\right\|_2$. Now, to maximize the effect of oversampling, we ought to find unchosen indices that enrich the trailing singular subspace of $Q_C(I,:)$, which in turn increases the minimum singular value(s) of $Q_C(I,:)$. It turns out that we can achieve this by projecting $Q_C([m] - I,:)$ onto the trailing singular subspace of $Q_C(I,:)$ and use a good row selection algorithm to choose p extra rows. Before stating the algorithm, we first motivate our rationale behind our approach using the cosine-sine (CS) decomposition.

Let $Q \in \mathbb{R}^{n\times k}$ be any orthonormal matrix with partition

$$Q = \begin{matrix} k \\ \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \end{matrix} \begin{matrix} \\ n_1 \\ n_2 \end{matrix} \tag{4.2}$$

[12] A similar version for column space approximator can also be devised by considering A^T instead. In the case when X and Y are not available, for example, when the initial set of indices were obtained using uniform sampling, $R = A(I,:)$ and $C = A(:, J)$ can be used as the row space approximator and the column space approximator, respectively.

[13] LU with partial pivoting is also effective in practice [20].

[14] Oversampling both the row indices I and the column indices J independently is not recommended. See Section 5.3 for a further discussion.

where $n_1, n_2 \geq k$ with $n_1 + n_2 = n$. Then the CS decomposition [45] gives us

$$Q_1 = U_1 C V^T \text{ and } Q_2 = U_2 S V^T, \tag{4.3}$$

where $U_1 \in \mathbb{R}^{n_1 \times k}, U_2 \in \mathbb{R}^{n_2 \times k}, V \in \mathbb{R}^{k \times k}$ are matrices with orthonormal columns and $C = \text{diag}(c_1, c_2, ..., c_k), S = \text{diag}(s_1, s_2, ..., s_k)$ are diagonal matrices satisfying $c_i^2 + s_i^2 = 1$ for all i. In our context, we can view $Q_C(I, :)$ as Q_1 and $Q_C([m] - I, :)$ as Q_2. Now assume, without loss of generality, that the c_i's are in non-increasing order so the s_i's are in non-decreasing order. Then in order to increase the minimum singular value of Q_1, we could add the rows of Q_2 that contribute the most to the trailing right singular subspace of Q_1, i.e., $V_{-p} = V(:, k-p+1:k) \in \mathbb{R}^{k \times p}$. By (4.3), V_{-p} is also the dominant right singular subspace of Q_2. When we add the rows of Q_2 that lies in the subspace spanned by V_{-p} to Q_1, we can increase the minimum singular value(s) of Q_1, i.e., increase c_k or the last few c_i's. Therefore we can apply any algorithm (such as those discussed in Section 1) that finds good row indices on $Q_2 V_{-p} \in \mathbb{R}^{n_2 \times p}$ and append them to Q_1 to increase the minimum singular value of Q_1.

If the algorithm requires the dominant left singular vectors of $Q_2 V_{-p}$ such as in DEIM or leverage scores sampling, we can simply scale the columns of $Q_2 V_{-p}$ using the singular values of Q_1, $C = \text{diag}(c_1, ..., c_k)$ because

$$Q_2 V_{-p} = U_{2,-p} \,\text{diag}(s_{k-p+1}, ..., s_k) = U_{2,-p} \sqrt{1 - \text{diag}(c_{k-p+1}^2, ..., c_k^2)}$$

where $U_{2,-p} = U_2(:, k-p+1:k)$.

In light of the observation made above, we propose the following algorithm (Algorithm 4.2) for oversampling indices. For simplicity, we use column pivoted QR (CPQR) on $Q_2 V_{-p}$ in Algorithm 4.2 to obtain a good set of oversampling indices. However, any algorithm that finds a good set of rows can replace line 4 of Algorithm 4.2.

Algorithm 4.2 Oversampling indices

Require: A full column rank matrix $B \in \mathbb{R}^{n \times k}$ with $n \geq k$, an index set I with $|I| = k$ and an oversampling parameter $p \leq k$

Ensure: Extra indices I_0 with $|I_0| = p$

`function` $I_0 = \texttt{OS}(B, I, p)$

1: $[Q_B, \sim] = \texttt{qr}(B, 0)$, ▷ Skip this step if B is already orthonormal.

2: $[\sim, \sim, V] = \texttt{svd}(Q_B(I, :))$,

3: Set $V_{-p} = V(:, k-p+1:k)$, the trailing p right singular vectors of $Q_B(J, :)$.

4: Apply CPQR on $(Q_B([m] - I, :)V_{-p})^T$. Let I_0 be the extra p indices for oversampling.

Given a full column rank matrix B, an index set I and an oversampling parameter $p(\leq k)$, Algorithm 4.2 finds the extra indices for oversampling by projecting Q_B onto the unchosen indices $[m] - I$ from the left and the trailing p right singular vectors of $Q_B(I, :)$ from the right and performing CPQR to obtain the extra indices I_0 with $|I_0| = p$. When $p > k$, we can iterate Algorithm 4.2 to obtain at most k oversampling indices at each iteration. We can also devise a version of Algorithm 4.2 with a tolerance parameter $0 < \epsilon < 1$ rather than taking p as input; for example $\epsilon = \sqrt{k/n}$ may be a reasonable choice (given that singular values of a $O(k) \times k$ submatrix of an $n \times n$ Haar distributed orthogonal matrix are of this order [41]). This version uses projection onto the trailing right singular subspace of $Q_B(I, :)$ corresponding to the singular values that are less than ϵ. While this version does not guarantee $\sigma_{\min}(Q_B(I \cup I_0, :)) \geq \epsilon$,

it can be applied iteratively to achieve this. To guarantee $\sigma_{\min}\left(Q_B(I \cup I_0, :)\right) \geq \epsilon$, one can alternatively use the `GappyPOD+E` oversampling algorithm [47]. However, this approach comes with a higher computational cost. The complexity of Algorithm 4.2 is $\mathcal{O}(nk^2)$ where the dominant cost comes from taking the QR decomposition of B (line 1). If B were orthonormal to begin with, then the dominant cost comes from forming $Q_B([m] - I, :)V_{-p}$, which costs $\mathcal{O}(nkp)$.

In the analysis for the CURCA in Section 3, oversampling played two roles: (i) improve the accuracy of the CURCA (see Theorem 3.3, Corollary 3.4, Theorem 3.9), and (ii) improve the stability of the CURCA in the presence of roundoff errors (see Theorem 3.9). Therefore, it is important to oversample, especially if the original set of indices are not good. For example, if the core matrix $A(I, J)$ is (nearly) singular. Oversampling should be done in such a way that the term $\left\|Q_C(I, :)^{-1}\right\|_2$ is reduced further by adding an extra set of indices I_0 that lie in the trailing singular subspace of $Q_C(I, :)$. Algorithm 4.2 achieves this by picking good unchosen indices from the trailing singular subspace of Q_C. We further highlight the significance of oversampling through numerical illustrations in Section 5.

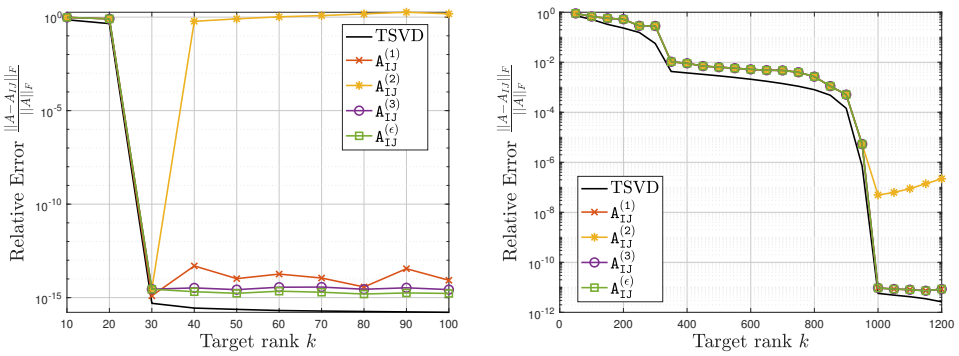
5. Numerical Illustration. In this section, we illustrate the concepts discussed in the previous sections through numerical experiments. We first discuss implementation details of (S)CURCA in MATLAB. Then we show that, without loss of generality, after selecting the rows first, the columns should be chosen based on those rows, i.e., the rows and the columns for the CURCA should not be chosen independently. We show that oversampling can improve the quality when the indices are poorly selected. Additionally, we also illustrate the effectiveness of the oversampling algorithm, Algorithm 4.2, for the CURCA and show its competitiveness against some existing methods. In all the experiments, the best rank-k approximation error using the truncated SVD (TSVD) is used as reference. The experiments were conducted in MATLAB version 2021a using double precision arithmetic.

5.1. Implementation of (S)CURCA. The main concern in the computation of the CURCA is that (i) the representation of the CURCA typically involves three (highly) ill-conditioned matrix and (ii) we take the pseudoinverse of a (highly) ill-conditioned matrix. When the original matrix A is low-rank and we have a good low-rank approximation of A, then we expect C, U and R to be highly ill-conditioned. We test four possible implementations in MATLAB,

1. $\mathtt{A}_{\mathtt{IJ}}^{(1)} = (\mathtt{A}(:, \mathtt{J})/\mathtt{A}(\mathtt{I}, \mathtt{J})) * \mathtt{A}(\mathtt{I}, :)$,
2. $\mathtt{A}_{\mathtt{IJ}}^{(2)} = \mathtt{A}(:, \mathtt{J}) * (\mathtt{V}/\mathtt{S} * \mathtt{W}') * \mathtt{A}(\mathtt{I}, :)$ where $[\mathtt{W}, \mathtt{S}, \mathtt{V}] = \mathtt{svd}(\mathtt{U}, \text{`}\mathtt{econ}')$,
3. $\mathtt{A}_{\mathtt{IJ}}^{(3)} = (\mathtt{A}(:, \mathtt{J}) * \mathtt{V}/\mathtt{S}) * (\mathtt{W}' * \mathtt{A}(\mathtt{I}, :))$ where $[\mathtt{W}, \mathtt{S}, \mathtt{V}] = \mathtt{svd}(\mathtt{U}, \text{`}\mathtt{econ}')$,
4. $\mathtt{A}_{\mathtt{IJ}}^{(\epsilon)} = (\mathtt{A}(:, \mathtt{J}) * \mathtt{V}(:, \mathtt{II})/\mathtt{S}(\mathtt{II}, \mathtt{II})) * (\mathtt{W}(:, \mathtt{II})' * \mathtt{A}(\mathtt{I}, :))$
   where $[\mathtt{W}, \mathtt{S}, \mathtt{V}] = \mathtt{svd}(\mathtt{U}, \text{`}\mathtt{econ}')$ and $\mathtt{II} = (\mathtt{diag}(\mathtt{S}) > \epsilon)$ with $\epsilon = 10^{-15}$.

The third implementation, $\mathtt{A}_{\mathtt{IJ}}^{(3)}$ is the suggested implementation of the CURCA. For guaranteed stability, we suggest the fourth implementation, $\mathtt{A}_{\mathtt{IJ}}^{(\epsilon)}$ with the ϵ-pseudoinverse. However we have not observed instability without the ϵ-pseudoinverse using the third implementation. We use two test matrices: (1) 1000×1000 test matrix generated using the MATLAB command, $\mathtt{randn}(1000, 30) * \mathtt{randn}(30, 1000)$ and (2) 1374×1374 matrix named `nnc1374` from the SuiteSparse Matrix Collection [15]. We choose the rows and columns by first choosing the column indices J using column pivoted QR (CPQR) on A and then using CPQR on the chosen columns $A(:, J)$ to get the row indices I.

The results are depicted in Figure 1. First, we notice that the third implemen-



(a) $\mathtt{randn}(1000, 30) * \mathtt{randn}(30, 1000)$ (b) `nnc1374`

FIG. 1. *Tests for different implementations of the CURCA. The third and fourth implementation performs stably. We recommend the third implementation.*

tation, $\mathtt{A}_{\mathtt{IJ}}^{(3)}$, which is the one we suggest, yields stable approximation throughout the experiment. In particular, the third implementation performs similarly to the fourth implementation with the ϵ-pseudoinverse. On the other hand, the second implementation, $\mathtt{A}_{\mathtt{IJ}}^{(2)}$ suffers from numerical errors as we lose accuracy when the singular values decay extremely rapidly. The reason is because we explicitly form the pseudoinverse of the core matrix $A(I, J)$, which is highly ill-conditioned, an observation also noted in [42]. Therefore, it is important to not form the pseudoinverse of $U = A(I, J)$ explicitly in the CURCA. Note also that the order in which the factors are multiplied is essential for numerical stability, as the second and third implementations only differ by the order in which the factors are computed. The first implementation, $\mathtt{A}_{\mathtt{IJ}}^{(1)}$ may lose 1 or 2 orders of magnitude in Figure 1a once the target rank becomes larger than the rank of the test matrix and appear to be less stable than the third implementation. The slash commands $(/, \backslash)$ in MATLAB should be used with caution for underdetermined problems, as its backslash command applied to numerically rank-deficient underdetermined problems output a sparse solution based on a pivoting strategy [2, § 2.4], which can differ significantly from the minimum-norm solution and may not satisfy the assumptions in our analysis. Note that using the `pinv` command with tolerance parameter 0 for the CURCA in MATLAB, i.e., $\mathtt{A}(:, \mathtt{J}) * \mathtt{pinv}(\mathtt{A}(\mathtt{I}, \mathtt{J}), 0), *\mathtt{A}(\mathtt{I}, :)$, is equivalent to the second implementation, which can be unstable. For the rest of the numerical experiments, we use the third implementation

$$(\mathtt{A}(:, \mathtt{J}) * \mathtt{V}/\mathtt{S}) * (\mathtt{W}' * \mathtt{A}(\mathtt{I}, :)) \text{ where } [\mathtt{W}, \mathtt{S}, \mathtt{V}] = \mathtt{svd}(\mathtt{U}, \text{`}\mathtt{econ}'). \tag{5.1}$$

The fourth implementation computes the stabilized CURCA and is implemented in such a way that the analysis in Section 3.2 hold. A less expensive alternative is to use a rank-revealing QR factorization and truncate the bottom-right corner of the upper triangular factor and the relevant columns of the orthonormal factor corresponding to the diagonal elements less than ϵ. In the rank-revealing QR, the diagonal elements give a good approximation to the singular values; see for example [10, 29]. A possible workaround without the ϵ-pseudoinverse is also discussed in [42] where we perturb the core matrix $A(I, J)$ by a small noise matrix such that the singular values of $A(I, J)$ are all larger than the unit roundoff.

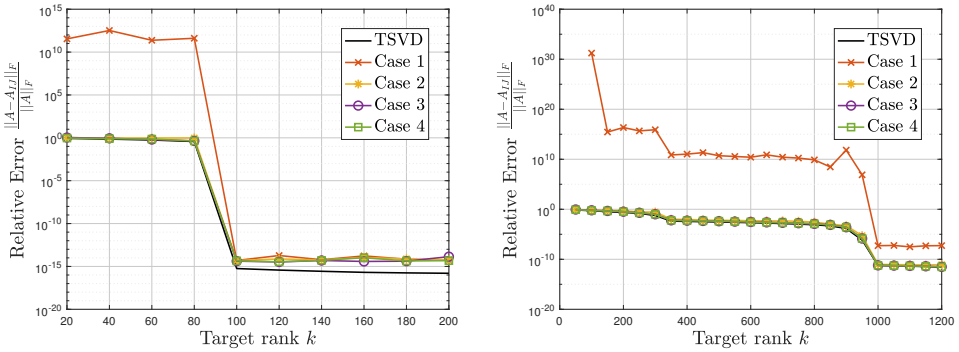
5.2. Importance of not choosing rows and columns independently. In this section, we demonstrate the importance of choosing the rows and columns that are dependent on one another in the CURCA. If the rows and columns are chosen independently of each other, the matrix $U = A(I, J)$ can be (nearly) singular. In such a scenario, we show that a sufficient amount of oversampling can remedy this problem. We use two test matrices: (1) synthetic matrix generated using

$$A = \begin{bmatrix} 10^{-10} \cdot \texttt{randn}(50, 50) & \texttt{randn}(50, 950) \\ \texttt{randn}(950, 50) & 0 \end{bmatrix} \in \mathbb{R}^{1000 \times 1000}, \tag{5.2}$$

and (2) `nnc1374` matrix from the SuiteSparse Matrix Collection used in the previous section. We test the following four cases:

1. Choose columns and rows independently by applying CPQR on A and A^T respectively,
2. Choose columns and rows dependently by applying CPQR on A and then applying CPQR on the chosen columns $A(:, J)$ to obtain the rows,
3. Apply Case 1 and additionally do row oversampling using Algorithm 4.2 with $p = k$,
4. Apply Case 2 and additionally do row oversampling using Algorithm 4.2 with $p = k$,

where k is the target rank.



(a) Block random matrix (5.2) (b) `nnc1374`

FIG. 2. *Relationship between the rows and columns in the CURCA. In Case* 1*, the rows and columns are chosen independently from one another and in Case* 2*, we select the columns first and then the rows were computed from the selected columns. Cases* 3 *and* 4 *correspond to Cases* 1 *and* 2 *with row oversampling ($p = k$), respectively. When the rows and columns are chosen independently (Case* 1*), the resulting approximation can be catastrophic.*

In Figure 2, we show the importance of choosing the columns and rows in the CURCA. When the columns and rows are chosen independently (Case 1), the resulting CURCA can be catastrophically poor. In Figure 2a, poor approximation happens because when we choose the columns and rows independently, we choose the first 50 rows and columns, as they are the most important, implying that the core matrix is the small $(1, 1)$-block of A. Since the chosen rows and columns are $\mathcal{O}(1)$ and their

intersection is $\mathcal{O}(10^{-10})$, the CURCA becomes inaccurate.[15] A similar issue also arises in Figure 2b where the approximation becomes noticeably unreliable throughout; see also the 2×2 example at the end of Section 3.1. However, when we oversample sufficiently in Case 3, we obtain a good CUR approximation. When we choose the rows and columns dependently in Cases 2 and 4, we have good stable approximations. Therefore, it is highly recommended to choose the rows and columns dependently. However, if that is not possible, a sufficient amount of oversampling is recommended.

5.3. Oversampling algorithm comparison. In this section, we illustrate advantages of oversampling through numerical experiments. In all experiments, unless stated otherwise, we use pivoting on a random sketch (Algorithm 4.1) with column pivoted QR [20, 54] and the Gaussian sketch to get the initial set of k row and column indices, where k is the target rank. We then obtain the oversampling indices using the following algorithms:

1. `OS + P`: Algorithm 4.2,
2. `OS + L`: Choose p extra indices corresponding to the largest p leverage scores out of the unchosen indices as in [26],
3. `OS + E`: `GappyPOD + E` algorithm in [47].

We set the number of oversampling indices to be $p = 0$, $p = 10$ and $p = 0.5k$.

We consider three different classes of test matrices, which are summarized below:

1. `CIFAR10`: The CIFAR-10 training set [37] consists of 60000 images of size $32 \times 32 \times 3$. We choose 10000 random images, each flattened to a vector, and treat it as a 10000×3072 data matrix.
2. `YaleFace64x64`: Yale face is a full-rank dense matrix of size 165×4096 consisting of 165 face images each of size 64×64. The flattened image vectors are centered and normalized such that the mean is zero and the entries lie within $[-1, 1]$.
3. `SNN`: Random sparse non-negative matrices are test matrices used in [49, 54] that is given by,

$$\texttt{SNN} = \sum_{j=1}^{r} s_j x_j y_j^T,$$

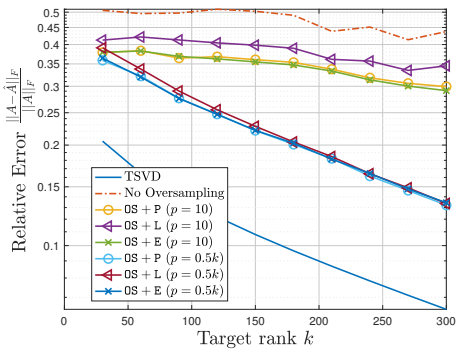
where $s_1 \geq \cdots \geq s_r > 0$ and $x_j \in \mathbb{R}^m, y_j \in \mathbb{R}^n$ are random sparse vectors with non-negative entries. We take $m = 100000, n = 300$ with

$$\sum_{j=1}^{50} \frac{2}{j} x_j y_j^T + \sum_{j=51}^{300} \frac{1}{j} x_j y_j^T,$$

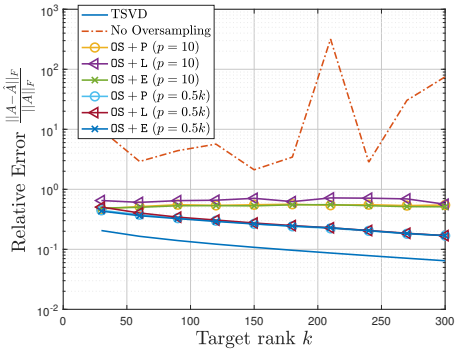
where the sparse vectors x_j's and y_j's are computed in MATLAB using the command $\texttt{sprand}(m, 1, 0.025)$ and $\texttt{sprand}(n, 1, 0.025)$, respectively.

The results are depicted in Figure 3. We compare different oversampling algorithms for the three test matrices. We use pivoting on a random sketch (Algorithm 4.1) to obtain the initial set of indices except for Figure 3b where we use uniform

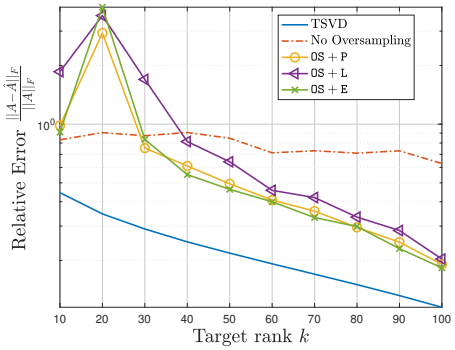
[15] A similar problem may also arise in random sampling where sampling rows and columns independently may lead to a poor CUR decomposition with high probability. For example, when $A = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$ and the target rank is 1, the probability of sampling zero as the core matrix using leverage scores or column norms is 0.5 when the rows and columns are sampled independently. Independent sampling leads to a poor CUR approximation 50% of the time. However, if we sample them dependently, we choose 1 as the core matrix with probability 1, leading to a sensible CUR approximation.



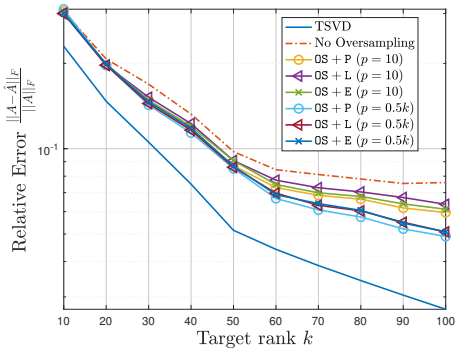
(a) `CIFAR10` dataset with pivoting on a random sketch with row oversampling.



(b) `CIFAR10` dataset with uniform sampling with row oversampling.



(c) `YaleFace64x64` dataset with pivoting on a random sketch with row ($p = 0.5k$) and column ($p = 10$) oversampling.



(d) `SNN` dataset with pivoting on a random sketch with row oversampling

FIG. 3. *Comparison of different oversampling methods for various test matrices. We use pivoting on a random sketch to obtain the initial set of indices except for Figure 3b where we use uniform sampling. Then we oversample the row indices using the three oversampling algorithms,* `OS+P`*,* `OS+L` *and* `OS+E`*. We also oversample column indices in Figure 3c to demonstrate that oversampling both row and column indices can be harmful.*

sampling; pivoting on a random sketch usually results in a good set of indices, while uniform sampling can give a poor set of indices. Then we oversample the row indices using the three oversampling algorithms, `OS+P`, `OS+L` and `OS+E`. We also oversample column indices in Figure 3c. For Figure 3c, $p = 0.5k$ for row oversampling and $p = 10$ for column oversampling.

We begin with the top two plots involving the `CIFAR10` dataset, Figures 3a and 3b. We observe that as the oversampling parameter increases, the accuracy improves. The different oversampling techniques yield a similar result with `OS+L` being usually slightly worse. In Figures 3a and 3b, we observe that oversampling plays a big role for the accuracy of the CURCA. In Figure 3a, $p = 10$ improves the accuracy slightly, but when the oversampling parameter p becomes proportional to the target rank, we make further progress and we approximately capture the singular value decay rate. In Figure 3b, when the initial set of indices are worse, as it can be when we perform uniform sampling, we observe that oversampling makes the approximation

more robust even when the CURCA without oversampling yields unstable results. Therefore, oversampling helps the CURCA yield more accurate and stable results.

In Figure 3c, using the `YaleFace64x64` dataset, we show what happens when we oversample both row and column indices. In this experiment, we oversample the row indices by $p = 0.5k$ and the column indices by $p = 10$. In Figure 3c, the CURCA can get worse with oversampling and the approximation may become unstable when both column and row indices are oversampled. This is caused by the core matrix $A(I \cup I_0, J \cup J_0)$ underestimating the singular values of A as oversampling in only one of row or column indices improve the condition number of the core matrix, but when we oversample both row and column indices, condition number of the core matrix can become worse. This is similar to the phenomenon happening in Section 5.2, where the CURCA can yield poor accuracy when we choose the rows and columns independently. Therefore, we advocate oversampling in only one of row or column indices for the CURCA. Note that if we do not oversample column indices in Figure 3c, so that we only oversample rows for the `YaleFace64x64` dataset, the relative error decreases steadily, as observed in the other figures in Figure 3. In Figure 3d, we use the `SNN` dataset. The effect of oversampling is less immediate in this example, but the accuracy still improves and the decay rate is more in line with the spectral decay.

Lastly, in all the experiments in Figure 3, we observe that our algorithm for oversampling $\texttt{OS}+\texttt{P}$ (Algorithm 4.2) is competitive with $\texttt{OS}+\texttt{E}$ with $\texttt{OS}+\texttt{L}$ usually being slightly worse. Therefore, Algorithm 4.2, which runs with complexity $\mathcal{O}(nk^2 + nkp)$ is competitive with `OS`+`E`, which run with complexity $\mathcal{O}(nk^2p + k^4)$.

6. Conclusion. In this work, we study the accuracy and stability of the CUR decomposition with oversampling. We prove the relative norm bounds for the CUR decomposition, $A \approx CU^\dagger R$, and its stabilized version, $A \approx CU_\epsilon^\dagger R$. We further show that the stabilized version satisfies a similar relative norm bound in the presence of roundoff errors under the assumption that the rows and columns for the CUR decomposition are chosen reasonably. This means that the rows should be selected based on the chosen columns or vice versa; see Section 5.2. This aims to reduce the quantity $\left\|Q_C(I,:)^{-1}\right\|_2$ (see Theorem 3.9), which can be further reduced by oversampling the row indices. For a stable implementation of the CURCA, $A \approx CU^\dagger R$, we recommend the MATLAB implementation

$$\texttt{A}_{\texttt{IJ}}^{(3)} = (\texttt{A}(:,\texttt{J}) * \texttt{V}/\texttt{S}) * (\texttt{W}' * \texttt{A}(\texttt{I},:)) \text{ where } [\texttt{W},\texttt{S},\texttt{V}] = \texttt{svd}(\texttt{U},\text{`}\texttt{econ}\text{'})$$

or the corresponding version where QR factorization is used instead of the SVD.

We also proposed how oversampling should be done. Oversampling should be done such that it increase the minimum singular value of a certain square matrix that is a submatrix of an orthonormal matrix; see Section 4. We suggest doing so through projecting the unchosen rows of an orthonormal matrix onto the trailing singular subspace of the square matrix and finding the important unchosen indices that will enrich the trailing singular subspace; see Algorithm 4.2. Oversampling improves the stability as the core matrix becomes rectangular and rectangular matrices are more well-conditioned than square matrices. We recommend oversampling in one only one of row or column indices, but not both (see Figure 3c) and choose the oversampling parameter to be proportional to the target rank when possible. Experiments show that the algorithm is competitive with other existing methods. Therefore, we advocate oversampling for the accuracy and stability of the CUR decomposition whenever possible.

REFERENCES

[1] D. Anderson, S. Du, M. Mahoney, C. Melgaard, K. Wu, and M. Gu, *Spectral Gap Error Bounds for Improving CUR Matrix Decomposition and the Nyström Method*, in Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, G. Lebanon and S. V. N. Vishwanathan, eds., vol. 38 of Proceedings of Machine Learning Research, San Diego, California, USA, 09–12 May 2015, PMLR, pp. 19–27.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, et al., *LAPACK Users' guide*, SIAM, 1995.

[3] H. Avron, P. Maymounkov, and S. Toledo, *Blendenpik: Supercharging LAPACK's least-squares solver*, SIAM J. Sci. Comput., 32 (2010), pp. 1217–1236, https://doi.org/10.1137/090767911.

[4] O. Balabanov and L. Grigori, *Randomized Gram–Schmidt process with application to GMRES*, SIAM J. Sci. Comput., 44 (2022), pp. A1450–A1474, https://doi.org/10.1137/20M138870X.

[5] J. Batson, D. A. Spielman, and N. Srivastava, *Twice-ramanujan sparsifiers*, SIAM J. Comput., 41 (2012), pp. 1704–1721, https://doi.org/10.1137/090772873.

[6] C. Boutsidis, P. Drineas, and M. Magdon-Ismail, *Near-optimal column-based matrix reconstruction*, SIAM J. Comput., 43 (2014), pp. 687–717, https://doi.org/10.1137/12086755X.

[7] C. Boutsidis and D. P. Woodruff, *Optimal CUR matrix decompositions*, SIAM J. Comput., 46 (2017), pp. 543–589, https://doi.org/10.1137/140977898.

[8] D. Cai, J. Nagy, and Y. Xi, *Fast deterministic approximation of symmetric indefinite kernel matrices with high dimensional datasets*, SIAM J. Matrix Anal. Appl., 43 (2022), pp. 1003–1028, https://doi.org/10.1137/21M1424627.

[9] K. Carlberg, C. Farhat, J. Cortial, and D. Amsallem, *The GNAT method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows*, J. Comput. Phys., 242 (2013), pp. 623–647, https://doi.org/https://doi.org/10.1016/j.jcp.2013.02.028.

[10] T. F. Chan, *Rank revealing QR factorizations*, Linear Algebra Appl., 88-89 (1987), pp. 67–82, https://doi.org/https://doi.org/10.1016/0024-3795(87)90103-0.

[11] S. Chaturantabut and D. C. Sorensen, *Nonlinear model reduction via discrete empirical interpolation*, SIAM J. Sci. Comput., 32 (2010), pp. 2737–2764, https://doi.org/10.1137/090766498.

[12] J. Chiu and L. Demanet, *Sublinear randomized algorithms for skeleton decompositions*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1361–1383, https://doi.org/10.1137/110852310.

[13] A. Cortinovis and D. Kressner, *Low-rank approximation in the Frobenius norm by column and row subset selection*, SIAM J. Matrix Anal. Appl., 41 (2020), pp. 1651–1673.

[14] A. Cortinovis and L. Ying, *A sublinear-time randomized algorithm for column and row subset selection based on strong rank-revealing QR factorizations*, arXiv preprint arXiv:2402.13975, (2024).

[15] T. A. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Trans. Math. Softw., 38 (2011), https://doi.org/10.1145/2049662.2049663, https://doi.org/10.1145/2049662.2049663.

[16] M. Dereziński and M. Mahoney, *Determinantal point processes in randomized numerical linear algebra*, Notices Amer. Math. Soc., 60 (2021), p. 1, https://doi.org/10.1090/noti2202.

[17] A. Deshpande, L. Rademacher, S. S. Vempala, and G. Wang, *Matrix approximation and projective clustering via volume sampling*, Theory Comput., 2 (2006), pp. 225–247, https://doi.org/10.4086/toc.2006.v002a012.

[18] A. Deshpande and S. Vempala, *Adaptive sampling and fast low-rank matrix approximation*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, J. Díaz, K. Jansen, J. D. P. Rolim, and U. Zwick, eds., Berlin, Heidelberg, 2006, Springer Berlin Heidelberg, pp. 292–303.

[19] M. Donello, G. Palkar, M. H. Naderi, D. C. Del Rey Fernández, and H. Babaee, *Oblique projection for scalable rank-adaptive reduced-order modelling of nonlinear stochastic partial differential equations with time-dependent bases*, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 479 (2023), p. 20230320, https://doi.org/10.1098/rspa.2023.0320.

[20] Y. Dong and P.-G. Martinsson, *Simpler is better: a comparative study of randomized pivoting algorithms for CUR and interpolative decompositions*, Adv. Comput. Math., 49 (2023), https://doi.org/10.1007/s10444-023-10061-z.

[21] P. Drineas and I. C. F. Ipsen, *Low-rank matrix approximations do not need a singular*

value gap, SIAM J. Matrix Anal. Appl., 40 (2019), pp. 299–319, https://doi.org/10.1137/18M1163658.

[22] P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff, *Fast approximation of matrix coherence and statistical leverage*, J. Mach. Learn. Res., 13 (2012), pp. 3475–3506, http://jmlr.org/papers/v13/drineas12a.html.

[23] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, *Relative-error CUR matrix decompositions*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 844–881, https://doi.org/10.1137/07070471X.

[24] Z. Drmač and S. Gugercin, *A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions*, SIAM J. Sci. Comput., 38 (2016), pp. A631–A648, https://doi.org/10.1137/15M1019271.

[25] J. A. Duersch and M. Gu, *Randomized projection for rank-revealing matrix factorizations and low-rank approximations*, SIAM Rev., 62 (2020), pp. 661–682, https://doi.org/10.1137/20M1335571.

[26] P. Y. Gidisu and M. E. Hochstenbach, *A hybrid DEIM and leverage scores based method for CUR index selection*, in Progress in Industrial Mathematics at ECMI 2021, M. Ehrhardt and M. Günther, eds., Cham, 2022, Springer International Publishing, pp. 147–153.

[27] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, 4 ed., 2013.

[28] S. Goreinov, E. Tyrtyshnikov, and N. Zamarashkin, *A theory of pseudoskeleton approximations*, Linear Algebra Appl., 261 (1997), pp. 1–21, https://doi.org/https://doi.org/10.1016/S0024-3795(96)00301-1.

[29] M. Gu and S. C. Eisenstat, *Efficient algorithms for computing a strong rank-revealing QR factorization*, SIAM J. Sci. Comput., 17 (1996), pp. 848–869, https://doi.org/10.1137/0917055.

[30] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), p. 217–288, https://doi.org/10.1137/090771806.

[31] K. Hamm and L. Huang, *Perspectives on CUR decompositions*, Appl. Comput. Harmon. Anal., 48 (2020), pp. 1088–1099, https://doi.org/https://doi.org/10.1016/j.acha.2019.08.006.

[32] K. Hamm and L. Huang, *Stability of sampling for CUR decompositions*, Foundations of Data Science, 2 (2020), pp. 83–99, https://doi.org/10.3934/fods.2020006.

[33] K. Hamm and L. Huang, *Perturbations of CUR decompositions*, SIAM J. Matrix Anal. Appl., 42 (2021), pp. 351–375, https://doi.org/10.1137/19M128394X.

[34] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, second ed., 2002, https://doi.org/10.1137/1.9780898718027.

[35] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 2 ed., 2012, https://doi.org/10.1017/9781139020411.

[36] I. C. F. Ipsen and B. Nadler, *Refined perturbation bounds for eigenvalues of Hermitian and non-Hermitian matrices*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 40–53, https://doi.org/10.1137/070682745.

[37] A. Krizhevsky, *Learning multiple layers of features from tiny images*, 2009.

[38] M. W. Mahoney and P. Drineas, *CUR matrix decompositions for improved data analysis*, Proc. Natl. Acad. Sci., 106 (2009), pp. 697–702, https://doi.org/10.1073/pnas.0803205106.

[39] P.-G. Martinsson, *Randomized methods for matrix computations*, The Mathematics of Data, 25 (2019), pp. 187–231.

[40] P.-G. Martinsson and J. A. Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numer., 29 (2020), p. 403–572, https://doi.org/10.1017/s0962492920000021.

[41] E. S. Meckes, *The random matrix theory of the classical compact groups*, vol. 218, Cambridge University Press, 2019.

[42] Y. Nakatsukasa, *Fast and stable randomized low-rank matrix approximation*, arXiv preprint arXiv:2009.11392, (2020), https://arxiv.org/abs/2009.11392.

[43] Y. Nakatsukasa and N. J. Higham, *Backward stability of iterations for computing the polar decomposition*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 460–479, https://doi.org/10.1137/110857544.

[44] Y. Nakatsukasa and T. Park, *Randomized low-rank approximation for symmetric indefinite matrices*, SIAM J. Matrix Anal. Appl., 44 (2023), pp. 1370–1392, https://doi.org/10.1137/22M1538648.

[45] C. Paige and M. Wei, *History and generality of the CS decomposition*, Linear Algebra Appl., 208-209 (1994), pp. 303–326, https://doi.org/https://doi.org/10.1016/0024-3795(94)90446-4.

[46] D. Papailiopoulos, A. Kyrillidis, and C. Boutsidis, *Provable deterministic leverage score sampling*, in Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA, 2014, Association for Computing Machinery, p. 997–1006, https://doi.org/10.1145/2623330.2623698, https://doi.org/10.1145/2623330.2623698.
[47] B. Peherstorfer, Z. Drmač, and S. Gugercin, *Stability of discrete empirical interpolation and gappy proper orthogonal decomposition with randomized and deterministic sampling points*, SIAM J. Sci. Comput., 42 (2020), pp. A2837–A2864, https://doi.org/10.1137/19M1307391.
[48] A. Shimizu, X. Cheng, C. Musco, and J. Weare, *Improved active learning via dependent leverage score sampling*, arXiv preprint arXiv:2310.04966, (2023).
[49] D. C. Sorensen and M. Embree, *A DEIM induced CUR factorization*, SIAM J. Sci. Comp., 38 (2016), pp. A1454–A1482, https://doi.org/10.1137/140978430.
[50] D. B. Szyld, *The many proofs of an identity on the norm of oblique projections*, Numer. Algorithms, 42 (2006), pp. 309–323, https://doi.org/10.1007/s11075-006-9046-2.
[51] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, 1997, https://doi.org/10.1137/1.9780898719574.
[52] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Practical sketching algorithms for low-rank matrix approximation*, SIAM J. Matrix Anal. Appl., 38 (2017), p. 1454–1485, https://doi.org/10.1137/17m1111590.
[53] M. Udell and A. Townsend, *Why are big data matrices approximately low rank?*, SIAM J. Math. Data Sci., 1 (2019), pp. 144–160, https://doi.org/10.1137/18M1183480.
[54] S. Voronin and P.-G. Martinsson, *Efficient algorithms for CUR and interpolative matrix decompositions*, Adv. Comput. Math., 43 (2017), pp. 495–516, https://doi.org/10.1007/s10444-016-9494-8.
[55] J. Xia, *Making the Nyström method highly accurate for low-rank approximations*, SIAM J. Sci. Comput., 46 (2024), pp. A1076–A1101, https://doi.org/10.1137/23M1585039.
[56] N. L. Zamarashkin and A. I. Osinsky, *On the existence of a nearly optimal skeleton approximation of a matrix in the Frobenius norm*, Dokl. Math., 97 (2018), pp. 164–166, https://doi.org/10.1134/S1064562418020205.
[57] R. Zimmermann and K. Willcox, *An accelerated greedy missing point estimation procedure*, SIAM J. Sci. Comput., 38 (2016), pp. A2827–A2850, https://doi.org/10.1137/15M1042899.

Appendix A. Analysis of the CURBA. In this section, we analyze the CURBA,

$$A_{IJ}^{(BA)} = A(:,J)\left(A(:,J)^{\dagger}AA(I,:)^{\dagger}\right)A(I,:) = C(C^{\dagger}AR^{\dagger})R \tag{A.1}$$

with oversampling. For the CURBA, there exists a numerically stable algorthm given by the StableCUR algorithm in [1]. The algorithm computes the CURBA in the following way in MATLAB,

$$[\mathtt{Q_C},\sim] = \mathtt{qr}(\mathtt{A}(:,\mathtt{J}),0),\ [\mathtt{Q_R},\sim] = \mathtt{qr}(\mathtt{A}(\mathtt{I},:)',0),\ A_{IJ}^{(BA)} = \mathtt{Q_C} * (\mathtt{Q_C}' * \mathtt{A} * \mathtt{Q_R}) * \mathtt{Q_R}'. \tag{A.2}$$

We use this implementation of the CURBA in the experiments at the end of this section.

We first prove a relative norm bound for the CURBA with oversampling. We make the following standard assumptions, which is analogous to the CURCA counterpart in Section 3.

Assumption A.1.

1. $|I| = |J| = k$ where k is the target rank. The oversampling indices are I_0 with $|I_0| = p_1$ for the rows and J_0 with $|J_0| = p_2$ for the columns and they satisfy $k + \max\{p_1, p_2\} \leq \mathrm{rank}(A)$.
2. $A(:, J \cup J_0)$ has full column rank and $A(I \cup I_0, :)$ has full row rank.
3. $X(:,J) \in \mathbb{R}^{k\times(k+p_2)}$ has full row rank, where $X \in \mathbb{R}^{k\times n}$ is a row space approximator of A.
4. $Y(I,:) \in \mathbb{R}^{k\times(k+p_2)}$ has full row rank, where $Y \in \mathbb{R}^{m\times k}$ is a column space approximator of A.

The assumption that $X(:,J)$ and $A(I,:)$ having a full row rank and $Y(I,:)$ and $A(:,J)$ having a full column rank is generic, since most methods that pick good row and column indices satisfy this assumption. If one of X or Y are not available then $A(I,:)$ or $A(:,J)$ can be used instead, respectively.

We now prove the result for the CURBA with oversampling. The results shown below is a simple extension of [49, Lemma 4.2] and [20, Theorem 1]. Lemma A.2 considers one-sided projection with oversampling where we project A onto the chosen columns of A. In Theorem A.3, we consider the CURBA, $C(C^\dagger AR^\dagger)R$ where we project A onto the chosen rows and columns of A.

LEMMA A.2. *Under Assumption A.1,*

$$\left\|A - CC^\dagger A\right\| \le \left\|Q_X(J \cup J_0, :)^\dagger\right\|_2 \left\|A - AX^\dagger X\right\| \tag{A.3}$$

where $\|\cdot\|$ is any unitarily invariant norm and $Q_X \in \mathbb{R}^{n\times k}$ is an orthonormal matrix spanning the columns of X^T.

Proof. For shorthand let $J_* = J \cup J_0$ and let $\Pi_{J_*} = I_n(:, J_*) \in \mathbb{R}^{n\times(k+p)}$ such that $C = A(:,J_*) = A\Pi_{J_*}$. We first define two oblique projectors

$$\mathcal{P}_X := \Pi_{J_*}(X\Pi_{J_*})^\dagger X, \qquad \mathcal{P}_C := \Pi_{J_*} C^\dagger A \in \mathbb{R}^{n\times n}.$$

Note that since C has full column rank, $C^\dagger A\Pi_{J_*} = C^\dagger C = I_{k+p}$, and

$$\mathcal{P}_C\mathcal{P}_X = \Pi_{J_*} C^\dagger A\Pi_{J_*}(X\Pi_{J_*})^\dagger X = \mathcal{P}_X.$$

Therefore we get

$$A - CC^\dagger A = A(I - \mathcal{P}_C) = A(I - \mathcal{P}_C)(I - \mathcal{P}_X) = (I - CC^\dagger)A(I - \mathcal{P}_X).$$

Since $X\Pi_{J_*} = X(:, J_*)$ has full row rank,

$$X\mathcal{P}_X = X\Pi_{J_*}\left(X\Pi_{J_*}\right)^\dagger X = X$$

and we obtain

$$(I - \mathcal{P}_X) = (I - X^\dagger X)(I - \mathcal{P}_X).$$

Now putting these together, we obtain

$$\begin{aligned}\left\|A - CC^\dagger A\right\| &= \left\|(I - CC^\dagger)A(I - X^\dagger X)(I - \mathcal{P}_X)\right\| \\ &\le \left\|I - CC^\dagger\right\|_2 \left\|A(I - X^\dagger X)\right\| \left\|I - \mathcal{P}_X\right\|_2 \\ &= \left\|I - \mathcal{P}_X\right\|_2 \left\|A(I - X^\dagger X)\right\|,\end{aligned}$$

since $\left\|I - CC^\dagger\right\|_2 = 1$ as $CC^\dagger$ is an orthogonal projector. Now since $\mathcal{P}_X$ is an oblique projector [50], $\|I - \mathcal{P}_X\|_2 = \|\mathcal{P}_X\|_2$ so the result follows by noting that ∎

$$\|\mathcal{P}_X\|_2 = \left\|(X\Pi_J)^\dagger X\right\|_2 = \left\|Q_X(J,:)\right\|_2.$$

THEOREM A.3. *Under Assumption A.1,*

$$\left\|A - A^{(BA)}_{I\cup I_0, J\cup J_0}\right\| \le \left\|Q_X(J \cup J_0, :)^\dagger\right\|_2 \left\|A - AX^\dagger X\right\| \tag{A.4}$$

$$+ \left\|Q_Y(I \cup I_0, :)^\dagger\right\|_2 \left\|A - YY^\dagger A\right\|$$

where $\|\cdot\|$ is any unitarily invariant norm and $Q_X \in \mathbb{R}^{n\times k}$ and $Q_Y \in \mathbb{R}^{m\times k}$ are the orthonormal matrices spanning the columns of X^T and Y, respectively.

Proof. The proof follows by Lemma A.2 and noting that

$$\begin{aligned}
\left\|A - CC^\dagger AR^\dagger R\right\| &\le \left\|A - CC^\dagger A\right\| + \left\|CC^\dagger A - CC^\dagger AR^\dagger R\right\| \\
&\le \left\|A - CC^\dagger A\right\| + \left\|CC^\dagger\right\|_2 \left\|A - AR^\dagger R\right\| \\
&= \left\|A - CC^\dagger A\right\| + \left\|A - AR^\dagger R\right\| \\
&\le \left\|Q_X(J, :)^\dagger\right\|_2 \left\|A - AX^\dagger X\right\| + \left\|Q_Y(I, :)^\dagger\right\|_2 \left\|A - YY^\dagger A\right\|
\end{aligned}$$

where the inequality for the second term $\left\|A - AR^\dagger R\right\|$ in the final line can be shown by considering A^T in Lemma A.2. ∎

Remark A.4.

1. The difference between Theorem A.3 and its counterpart without oversampling, e.g. [20, Theorem 1], are the terms $\left\|Q_X(J, :)^\dagger\right\|_2$ and $\left\|Q_Y(I, :)^\dagger\right\|_2$ where instead of the matrix inverse, we have the pseudoinverse. This tightens the bound (A.4) as

$$\left\|Q_X(J \cup J_0, :)^\dagger\right\|_2 \le \left\|Q_X(J, :)^{-1}\right\|_2,$$

and

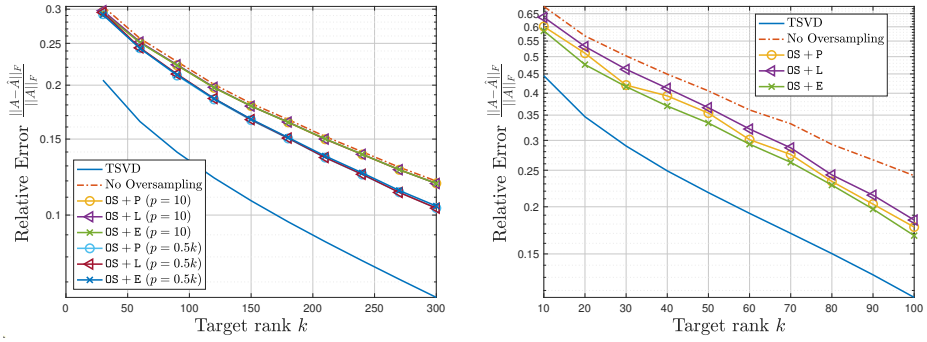
$$\left\|Q_Y(I \cup I_0, :)^\dagger\right\|_2 \le \left\|Q_Y(I, :)^{-1}\right\|_2,$$

where I_0 and J_0 are the extra indices for oversampling.
2. There are many possible choices for X and Y. For example, if we choose them to be the dominant right and left singular vectors of A, then we get the bound similar to the one in [49] involving the best rank-k approximation since $\left\|A - AX^\dagger X\right\| = \left\|A - YY^\dagger A\right\| = \|A - [\![A]\!]_k\|$ where $[\![A]\!]_k$ is the best rank-k approximation to A. If we choose X and Y to be the row sketch and the column sketch (see Algorithm 4.1) then we involve the randomized SVD error [30]. Choosing X and Y to be the dominant singular vectors in Theorem A.3 can be the optimal choice, but when X and Y are approximators, which can be computed easily, the bound (A.4) can be computed a posteriori.
3. When $k + \min\{p_1, p_2\} \ge \operatorname{rank}(A)$ and $\operatorname{rank}(C) = \operatorname{rank}(R) = \operatorname{rank}(A)$, we have $A = CC^\dagger AR^\dagger R$ [31].

Theorem A.3 suggests that oversampling helps to improve the accuracy of the CUR decomposition $A^{(BA)}_{IJ}$. However, the numerical simulations shown below illustrate that the improvement is not too significant.

The numerical results are depicted in Figure 4, illustrating the effect of oversampling for the CURBA. We use pivoting on a random sketch (Algorithm 4.1) to obtain the initial set of indices. Then we oversample the row indices using the three oversampling algorithms, `OS+P`, `OS+L` and `OS+E` as in Section 5.3. We also oversample column indices in Figure 4b.

We start with the `CIFAR10` dataset in Figure 4a. We first observe that as the oversampling parameter increases, the accuracy improves and the different oversampling techniques yield a similar result. However, there is no significant improvement when



(a) `CIFAR10` dataset with pivoting on a random sketch with row oversampling.

(b) `YaleFace64x64` dataset with pivoting on a random sketch with row ($p = 0.5k$) and column ($p = 10$) oversampling.

FIG. 4. *The effect of oversampling for the CURBA. We use pivoting on a random sketch (Algorithm 4.1) to obtain the initial set of indices. Then we oversample the row indices using the three oversampling algorithms,* `OS+P`, `OS+L` *and* `OS+E`. *We also oversample column indices in Figure 4b.*

oversampling is used for the CURBA. This shows that the CURBA gives a robust approximation and oversampling only plays a slight role in its accuracy. In Figure 4b, we used the `YaleFace64x64` dataset to illustrate the effect of oversampling both rows and columns. We observe that when we oversample both rows and columns, we obtain a higher accuracy for the CURBA. This is expected as we are enlarging the subspace that we project onto A, which increases the accuracy of the CURBA. This is contrary to the CURCA, as the CURCA can become more inaccurate and unstable when we oversample both rows and columns; see Section 5.3.

To conclude, while oversampling improves the accuracy of the CURBA, it has less significant effect on its accuracy and stability.

Appendix B. Stability of rank-deficient systems. In this section, we prove the stability result for solving rank-deficient underdetermined linear systems (Theorem 3.8). The result can also be extended to rank-deficient overdetermined problems; see Appendix B.1. Theorem 3.8 was used to derive an expression for the computed solution to

$$\min_x \left\| (\hat{U}^T)_\epsilon x - [\hat{C}]_i^T \right\|_2 \tag{B.1}$$

for each row of $\hat{C}$. The statement and the proof is shown below. The proof follows a similar method outlined in [42, Section 4.1].

THEOREM B.1 (Theorem 3.8). *Consider the (rank-deficient) underdetermined linear system,*

$$\min_{x'} \|B_\epsilon x' - b\|_2 \tag{3.7}$$

where $B_\epsilon \in \mathbb{R}^{m\times n}$ $(m \le n)$ is (possibly) rank-deficient $(\mathrm{rank}(B_\epsilon) \le m)$ *with singular vales larger than ϵ and $b \in \mathbb{R}^m$. Then assuming $\epsilon > \gamma \|B_\epsilon\|_2$, the minimum norm solution to (3.7) can be computed in a backward stable manner, i.e., the computed*

solution $\hat{s}$ satisfies

$$\hat{s} = (B_\epsilon + E_1)^\dagger (b + E_2)$$

where $\|E_1\|_2 \leq \gamma \|B_\epsilon\|_2$ *and* $\|E_2\|_2 \leq \gamma \|b\|_2$.

Proof. First, if B_ϵ has full row-rank then the statement follows by the stability result in [34, Theorem 21.4]. If B_ϵ is rank-deficient, the stability result in [34, Theorem 21.4] cannot be invoked as it is only applicable for underdetermined full-rank linear systems. So we project B_ϵ onto its column space Q_ϵ to make the problem an underdetermined full-rank linear system:

$$\min_{x'} \left\| Q_\epsilon^T B_\epsilon x' - Q_\epsilon^T b \right\|_2 .$$

Let $\hat{Q}_\epsilon$ be the computed column space of B_ϵ. Then $\hat{Q}_\epsilon$ is the exact column space of $B_\epsilon + \Delta B$ where $\|\Delta B\|_2 \leq \gamma \|B_\epsilon\|_2$, $\hat{Q}_\epsilon^T B_\epsilon$ is numerically full-rank with singular values larger than ϵ and $\hat{Q}_\epsilon \hat{Q}_\epsilon^T B_\epsilon = B_\epsilon + E$ where $\|E\|_2 \leq \gamma \|B_\epsilon\|_2$ [27, Chapter 5.4.1]. Note the following from matrix-matrix (or matrix-vector) multiplication [34, Section 3.5],

$$fl(\hat{Q}_\epsilon^T B_\epsilon) = \hat{Q}_\epsilon^T B_\epsilon + E^{(1)}$$

where $\left\|E^{(1)}\right\|_2 \leq \gamma \|B_\epsilon\|_2$ and

$$fl(\hat{Q}_\epsilon^T b) = \hat{Q}_\epsilon^T b + E^{(2)}$$

where $\left\|E^{(2)}\right\|_2 \leq \gamma \|b\|_2$. Since $Q_\epsilon^{\hat{T}} B_\epsilon \in \mathbb{R}^{\text{rank}(B_\epsilon) \times n}$ is a fat rectangular matrix, we solve the following underdetermined full-rank linear system,

$$\min_{x'} \left\| (\hat{Q}_\epsilon^T B_\epsilon) x' - \hat{Q}_\epsilon^T b \right\|_2 . \tag{B.2}$$

Assuming $\gamma \kappa_2(B_\epsilon) \leq \gamma \|B_\epsilon\|_2 / \epsilon < 1$ (where κ_2 denotes the 2-norm condition number), we obtain the computed solution of (B.2), $\hat{s}$, satisfying [34, Theorem 21.4],

$$\begin{aligned}
\hat{s} = fl\left(\left(\hat{Q}_\epsilon^T B_\epsilon \right)^\dagger \hat{Q}_\epsilon^T b \right) &= \left(fl(\hat{Q}_\epsilon^T B_\epsilon) + E^{(3)} \right)^\dagger fl(\hat{Q}_\epsilon^T b) \\
&= \left(\hat{Q}_\epsilon^T B_\epsilon + E^{(1)} + E^{(3)} \right)^\dagger \left(\hat{Q}_\epsilon^T b + E^{(2)} \right) \\
&= \left(\hat{Q}_\epsilon \hat{Q}_\epsilon^T B_\epsilon + \hat{Q}_\epsilon E^{(1)} + \hat{Q}_\epsilon E^{(3)} \right)^\dagger \left(b + \hat{Q}_\epsilon E^{(2)} \right) \\
&= \left(B_\epsilon + E + \hat{Q}_\epsilon E^{(1)} + \hat{Q}_\epsilon E^{(3)} \right)^\dagger \left(b + \hat{Q}_\epsilon E^{(2)} \right)
\end{aligned}$$

where $\left\|E^{(3)}\right\|_2 \leq \gamma \|B_\epsilon\|_2$ and we use $(QA)^\dagger b = A^\dagger Q^T b$ for a tall-skinny orthonormal matrix Q in line 3. Therefore, assuming that $\gamma \|B_\epsilon\|_2 < \epsilon$, the computed solution of (3.7), $\hat{s}$ satisfies

$$\hat{s} = fl(B_\epsilon^\dagger b) = (B_\epsilon + E_1)^\dagger (b + E_2) , \tag{B.3}$$

where $\|E_1\|_2 \leq \gamma \|B_\epsilon\|_2$ and $\|E_2\|_2 \leq \gamma \|b\|_2$. ∎

B.1. Extension to rank-deficient overdetermined problems. Theorem 3.8 can be extended to include rank-deficient overdetermined least-sqaures problems. Consider the rank-deficient overdetermined least-squares problem,

$$\min_{x'} \|C_\epsilon x' - b\|_2$$

where $C_\epsilon \in \mathbb{R}^{m\times n}$ $(m \geq n)$ is a rank-deficient ($\mathrm{rank}(C_\epsilon) < n$), tall-skinny matrix with singular vales larger than ϵ and $b \in \mathbb{R}^m$. Then under the same assumption as Theorem 3.8, i.e., $\epsilon > \gamma \|B_\epsilon\|_2$, the solution to the overdetermined least-squares problem can be computed in a backward stable manner.

The proof proceeds in the same way as Theorem 3.8, as once the tall-skinny matrix C_ϵ is projected onto its column space W_ϵ, $W_\epsilon^T C_\epsilon \in \mathbb{R}^{\mathrm{rank}(C_\epsilon)\times n}$ becomes a fat full-rank matrix, and we are now solving the underdetermined full-rank linear system: $\min_{x'} \left\|(\hat{W}_\epsilon^T C_\epsilon)x' - \hat{W}_\epsilon^T b\right\|_2$.

Appendix C. Two lemmas. We give the proofs for the two lemmas, Lemma 3.6 and Lemma 3.7 in this section.

LEMMA C.1 (Lemma 3.6). *Under Assumption 3.2, for any ΔC and ΔU,*

$$\left\|(C+\Delta C)(U+\Delta U)_\epsilon^\dagger\right\|_2 \leq \left\|Q_C(I_*,:)^\dagger\right\|_2 \left(1 + \frac{1}{\epsilon}\|\Delta U\|_2\right) + \frac{1}{\epsilon}\|\Delta C\|_2 . \tag{C.1}$$

Proof. Let $C = Q_C R_C$ be the thin QR factorization of C. Then

$$\begin{aligned}
\left\|(C+\Delta C)(U+\Delta U)_\epsilon^\dagger\right\|_2 &\leq \left\|Q_C R_C (U+\Delta U)_\epsilon^\dagger\right\|_2 + \left\|\Delta C (U+\Delta U)_\epsilon^\dagger\right\|_2 \\
&= \left\|R_C (U+\Delta U)_\epsilon^\dagger\right\|_2 + \left\|\Delta C (U+\Delta U)_\epsilon^\dagger\right\|_2 \\
&= \left\|(\Pi_{I_*}^T Q_C)^\dagger \Pi_{I_*}^T Q_C R_C (U+\Delta U)_\epsilon^\dagger\right\|_2 + \left\|\Delta C (U+\Delta U)_\epsilon^\dagger\right\|_2 \\
&\leq \left\|(\Pi_{I_*}^T Q_C)^\dagger\right\|_2 \left\|U(U+\Delta U)_\epsilon^\dagger\right\|_2 + \frac{1}{\epsilon}\|\Delta C\|_2
\end{aligned}$$

where we use $(\Pi_{I_*}^T Q_C)^\dagger \Pi_{I_*}^T Q_C = I$ for the penultimate line. The result follows by noting that $\left\|U(U+\Delta U)_\epsilon^\dagger\right\|_2$ simplifies to

$$\left\|U(U+\Delta U)_\epsilon^\dagger\right\|_2 = \left\|(U+\Delta U)(U+\Delta U)_\epsilon^\dagger - \Delta U (U+\Delta U)_\epsilon^\dagger\right\|_2 \leq 1 + \frac{1}{\epsilon}\|\Delta U\|_2 . \quad \square$$

LEMMA C.2 (Lemma 3.7). *Under Assumption 3.2, for any ΔC and ΔU,*

$$(C+\Delta C)(U+\Delta U)_\epsilon^\dagger R \Pi_J = C + E_* \tag{C.2}$$

where

$$\|E_*\|_2 \leq \left\|(Q_C(I_*,:)^\dagger\right\|_2 \left(\epsilon + 2\|\Delta U\|_2 + \frac{1}{\epsilon}\|\Delta U\|_2^2\right) + \|\Delta C\|_2 \left(1 + \frac{\|\Delta U\|_2}{\epsilon}\right).$$

Proof. We divide the expression into two pieces:

$$(C+\Delta C)(U+\Delta U)_\epsilon^\dagger R\Pi_J = \underbrace{C(U+\Delta U)_\epsilon^\dagger R\Pi_J}_{(i)} + \underbrace{\Delta C (U+\Delta U)_\epsilon^\dagger R\Pi_J}_{(ii)}$$

and treat them separately. Let the thin SVD of $U + \Delta U$ be

$$U + \Delta U = W\Sigma V^T = [W_1, W_2] \begin{bmatrix} \Sigma_1 & \\ & \Sigma_2 \end{bmatrix} [V_1, V_2]^T$$

where Σ_2 contains the singular values of $U + \Delta U$ smaller than ϵ and $C = Q_C R_C$ be the thin QR decomposition of C.

We begin by examining the matrix (i).

$$\begin{aligned} C(U + \Delta U)_\epsilon^\dagger R\Pi_J &= C(U + \Delta U)_\epsilon^\dagger U = C(U + \Delta U)_\epsilon^\dagger (U + \Delta U) - C(U + \Delta U)_\epsilon^\dagger \Delta U \\ &= CV_1V_1^T - C(U + \Delta U)_\epsilon^\dagger \Delta U = C - CV_2V_2^T - C(U + \Delta U)_\epsilon^\dagger \Delta U \\ &= C + E_1 \end{aligned}$$

where $E_1 = -CV_2V_2^T - C(U + \Delta U)_\epsilon^\dagger \Delta U$ satisfies

$$\begin{aligned} \|E_1\|_2 &\le \left\|CV_2V_2^T\right\|_2 + \left\|C(U + \Delta U)_\epsilon^\dagger \Delta U\right\|_2 \\ &\le \left\|CV_2V_2^T\right\|_2 + \left\|Q_C(I_*, :)^\dagger\right\|_2 \left(1 + \frac{1}{\epsilon} \|\Delta U\|\right) \|\Delta U\|_2 \end{aligned}$$

by Lemma 3.6. We bound $\left\|CV_2V_2^T\right\|_2$ as in the final part of the proof of Theorem 3.3.

$$\begin{aligned} \left\|CV_2V_2^T\right\|_2 &= \left\|R_C V_2 V_2^T\right\|_2 = \left\|(\Pi_{I_*}^T Q_C)^\dagger \Pi_{I_*}^T Q_C R_C V_2 V_2^T\right\|_2 = \left\|(\Pi_{I_*}^T Q_C)^\dagger U V_2 V_2^T\right\|_2 \\ &\le \left\|(\Pi_{I_*}^T Q_C)^\dagger\right\|_2 \left\|(U + \Delta U)V_2V_2^T - \Delta U V_2 V_2^T\right\|_2 \\ &\le \left\|(\Pi_{I_*}^T Q_C)^\dagger\right\|_2 (\|W_2\Sigma_2\|_2 + \|\Delta U\|_2) \\ &\le \left\|(\Pi_{I_*}^T Q_C)^\dagger\right\|_2 (\epsilon + \|\Delta U\|_2) . \end{aligned}$$

Therefore, (i) can be bounded as

$$C(U + \Delta U)_\epsilon^\dagger R\Pi_J = C + E_1$$

where $\|E_1\|_2 \le \left\|(Q_C(I_*, :)^\dagger\right\|_2 \left(\epsilon + 2\|\Delta U\|_2 + \frac{1}{\epsilon}\|\Delta U\|_2^2\right)$.

Next, we bound the matrix (ii). Let $E_2 = \Delta C(U + \Delta U)_\epsilon^\dagger R\Pi_J$, then

$$\begin{aligned} \|E_2\|_2 &= \left\|\Delta C(U + \Delta U)_\epsilon^\dagger R\Pi_J\right\|_2 \\ &\le \|\Delta C\|_2 \left\|(U + \Delta U)_\epsilon^\dagger U\right\|_2 \\ &\le \|\Delta C\|_2 \left(\left\|(U + \Delta U)_\epsilon^\dagger (U + \Delta U)\right\|_2 + \left\|(U + \Delta U)_\epsilon^\dagger \Delta U\right\|_2\right) \\ &\le \|\Delta C\|_2 \left(1 + \frac{\|\Delta U\|_2}{\epsilon}\right) . \end{aligned}$$

Putting everything together and setting $E_* = E_1 + E_2$, we get the desired result:

$$(C + \Delta C)(U + \Delta U)_\epsilon^\dagger R\Pi_J = C + E_*$$

where

$$\|E_*\|_2 \le \left\|(Q_C(I_*, :)^\dagger\right\|_2 \left(\epsilon + 2\|\Delta U\|_2 + \frac{1}{\epsilon}\|\Delta U\|_2^2\right) + \|\Delta C\|_2 \left(1 + \frac{\|\Delta U\|_2}{\epsilon}\right) . \quad \square$$