Sublinear Randomized Algorithms for Skeleton Decompositions


Please share how this access benefits you. Your story matters.
SUBLINEAR RANDOMIZED ALGORITHMS FOR SKELETON DECOMPOSITIONS∗

JIawei CHiu† and Laurent Demanet‡

Abstract. A skeleton decomposition of a matrix $A$ is any factorization of the form $A \rightarrow C Z A_R$, where $A_C$ comprises columns of $A$, and $A_R$ comprises rows of $A$. In this paper, we investigate the conditions under which random sampling of $C$ and $R$ results in accurate skeleton decompositions. When the singular vectors (or more generally the generating vectors) are incoherent, we show that a simple algorithm returns an accurate skeleton in sublinear $O(\ell^3)$ time from $\ell \approx k \log n$ rows and columns drawn uniformly at random, with an approximation error of the form $O(n \ell \sigma_k)$ where $\sigma_k$ is the $k$th singular value of $A$. We discuss the crucial role that regularization plays in forming the middle matrix $U$ as a pseudoinverse of the restriction $A_{RC}$ of $A$ to rows in $R$ and columns in $C$. The proof methods enable the analysis of two alternative sublinear-time algorithms, based on the rank-revealing QR decomposition, which allows us to tighten the number of rows and/or columns to $k$ with error bound proportional to $\sigma_k$.

Key words. skeleton, column subset selection, low-rank approximations, CUR, interpolative decomposition, Nystrom method

AMS subject classifications. 68W20, 65F30

DOI. 10.1137/110852310

1. Introduction.

1.1. Skeleton decompositions. This paper is concerned with the decomposition known as the matrix skeleton, pseudoskeleton [22], or CUR factorization [28, 18]. Throughout this paper we adopt the following MATLAB-friendly notation. Let $R, C$ be index sets. Given $A \in \mathbb{C}^{m \times n}$, let $A_C$ denote the restriction of $A$ to columns indexed by $C$, and $A_R$ denote the restriction of $A$ to rows indexed by $R$. A skeleton decomposition of $A$ is any factorization of the form

$$A_C Z A_R$$

for some $Z \in \mathbb{C}^{k \times k}$.

In general, storing a rank-$k$ approximation of $A$ takes $O((m+n)k)$ space. For skeletons however, only the middle factor $Z$ and the two index sets $C$ and $R$ need to be stored, if we assume that $A$’s entries can be sampled on-demand by an external function. Hence specifying the skeleton decomposition of $A$ only requires $O(k^2)$ space. In addition, rows and columns from the original matrix may carry more physical significance than their linear combinations.

There are important examples where the full matrix itself is not low rank but can be partitioned into blocks each of which has low numerical rank. One example is the Green’s function of elliptic operators with mild regularity conditions [5]. Another example is the amplitude factor in certain Fourier integral operators and wave propagators [12, 17]. Algorithms that compute good skeleton representations can be used to manipulate such matrices.
Algorithm 1. \(\tilde{O}(k^3)\)-time algorithm where \(\tilde{O}\) is the \(O\) notation with log factors dropped

**Input:** A matrix \(A \in \mathbb{C}^{m \times n}\) that is approximately rank \(k\), and user-defined parameters \(\ell = \tilde{O}(k)\) and \(\delta\).

**Output:** Column index set \(C\) of size \(\ell\), row index set \(R\) of size \(\ell\), center matrix \(Z\) of a matrix skeleton. Implicitly, we have the matrix skeleton \(A_{CZ}A_{R}^{*}\).

**Steps:**

1. Let \(C\) be a random index set of size \(\ell\) chosen uniformly from \(\{1, \ldots, n\}\). Implicitly, we have \(A_{C}\).  
2. Let \(R\) be a random index set of size \(\ell\) chosen uniformly from \(\{1, \ldots, m\}\). Implicitly, we have \(A_{R}\).  
3. Form \(A_{RC}\), the intersection of \(A_{C}\) and \(A_{R}\).  
4. Compute the thin SVD of \(A_{RC}\) as \(U_{1}\Sigma_{1}V_{1}^{*} + U_{2}\Sigma_{2}V_{2}^{*}\) where \(\Sigma_{1}\), \(\Sigma_{2}\) are diagonal, \(\Sigma_{1}\) contains singular values \(\geq \delta\), and \(\Sigma_{2}\) contains singular values \(< \delta\).  
5. Compute \(Z = V_{1}\Sigma_{1}^{-1}U_{1}^{*}\).

**MATLAB code:**

```
function [C,Z,R]=skeleton1(A,l,delta)  
C=randperm(n,l); R=randperm(m,l); Z=pinv(A(R,C),delta);  
end
```

1.2. Overview. This paper mostly treats the case of skeleton decompositions with \(C\) and \(R\) drawn uniformly at random. Denote by \(A_{RC}\) the restriction of \(A\) to rows in \(R\) and columns in \(C\), we compute the middle matrix \(Z\) as the pseudoinverse of \(A_{RC}\) with some amount of regularization. Algorithm 1 details the form of this regularization.

Throughout the paper we use the letter \(k\) to denote the baseline small dimension of the factorization: it is either exactly the rank of \(A\), or, more generally, it is the index of the singular value \(\sigma_{k}\) that governs the approximation error of the skeleton decomposition. The small dimension of the skeleton decomposition may or may not be \(k\); for instance, Algorithm 1 requires a small oversampling since \(\ell = O(k \log \max(m, n))\). Later, we consider two algorithms where \(\ell\) is exactly \(k\).

The situation in which Algorithm 1 works is when \(A\) is a priori known to have a factorization of the form \(\approx X_{1}A_{11}Y_{1}^{*}\) where \(X_{1}, Y_{1}\) have \(k\) orthonormal columns, and these columns are *incoherent*, or spread, in the sense that their uniform norm is about as small as their normalization allows. In this scenario, our main result in Theorem 1.1 states that the output of Algorithm 1 obeys

\[
\|A - A_{CZ}A_{R}\| = O\left(\|A - X_{1}A_{11}Y_{1}^{*}\| \left(\frac{mn}{\ell}\right)^{1/2}\right)
\]

with high probability, for some adequate choice of the regularization parameter \(\delta\).

The drawback of Algorithm 1 is that it requires setting an appropriate regularization parameter in advance. Unfortunately, there is no known way of estimating it fast, and this regularization step cannot be skipped. In section 4.1, we illustrate with a numerical example that without the regularization, the error in the operator norm can blow up in a way predicted by our main result, Theorem 1.1.

Finally, we use our proof framework to establish error estimates for two other algorithms. The goal of these algorithms is to further reduce the small dimension
of the skeleton decomposition to exactly $k$ (instead of $\ell = O(k \log \max(m, n))$), with $\sigma_k$ still providing control over the approximation error. The proposed methods still run in $o(mn)$ time (i.e., sublinear); they use well-known strong rank-revealing QR (RRQR) factorizations applied after some amount of pruning via uniform random sampling of rows and columns. This combination of existing ideas is an important part of the discussion of how skeleton factorizations can be computed reliably without visiting all the elements of the original matrix.

1.3. Related work. The idea of uniformly sampling rows and columns to build a matrix skeleton is not new. In particular, for the case where $A$ is symmetric, this technique is known as the Nyström method. The skeleton used is $A_C A_C^T A_C$, which is symmetric, and the error in the operator norm was recently analyzed by Talwalkar [36] and Gittens [21]. Both papers make the assumption that $X_1, Y_1$ are incoherent. Gittens obtained relative error bounds that are similar to ours.

Nonetheless, our results are more general. They apply to nonsymmetric matrices that are low rank in a broader sense. Specifically, when we write $A \simeq X_1 A_{11} Y_1^*$, $A_{11}$ is not necessarily diagonal and $X_1, Y_1$ are not necessarily the singular vectors of $A$. This relaxes the incoherence requirement on $X_1, Y_1$. Furthermore, in the physical sciences, it is not uncommon to work with linear operators that are known a priori to be almost (but not fully) diagonalized by the Fourier basis or related bases in harmonic analysis. These bases are often incoherent. One example is an integral operator with a smooth kernel. See section 4 for more details.

A factorization that is closely related to matrix skeletons is the interpolative decomposition [15], also called the column subset selection problem [20] or a RRQR [13, 23]. An interpolative decomposition of $A$ is the factorization $A_C D$ for some $D$. It is relevant to this paper because algorithms that compute interpolative decompositions can be used to compute matrix skeletons [27]. Algorithms 2 and 3 require the computation of interpolative decompositions.

One of the earliest theoretical results concerning matrix skeletons is due to Gorkin, Tytunshnikov, and Zamarashkin [22]. In that paper, it is shown that for any $A \in \mathbb{C}^{m \times n}$, there exists a skeleton $A_C Z A_R$ such that in the operator norm, $\|A - A_C Z A_R\| = O(\sqrt{m} + \sqrt{n} \sigma_{k+1}(A))$. Although the proof is constructive, it requires computing the SVD of $A$, which takes much more time and space than the algorithms considered in this paper. A useful idea in [22] for selecting $C$ and $R$ is to maximize the volume or determinant of submatrices. This idea may date back to interpolating projections [14] and the proof of Auerbach’s theorem [33].

A popular method of computing matrix skeletons is cross-approximation. The idea is to iteratively select good rows and columns based on the residual matrix. As processing the entire residual matrix is not practical, there are faster variants that operate on only a small part of the residual, e.g., adaptive cross approximation [3, 4] and incomplete cross approximation [38]. The algorithms considered in this paper are noniterative, arguably easier to implement and analyze, yet possibly less efficient for some applications.

In this paper, we compute a matrix skeleton by randomly sampling rows and columns of $A$. This idea dates back at least to the work of Frieze, Kannan, and Vempala [20]. One way of sampling rows and columns of $A$ is called “subspace sampling” [28, 18] by Drineas, Mahoney, and Muthukrishnan. If we assume that the top $k$ singu-

---

1. In machine learning, the Nyström method can be used to approximate kernel matrices of support vector machines, or the Laplacian of affinity graphs, for instance.
lar vectors of $A$ are incoherent, then a result due to Rudelson [31] and Rudelson and Vershynin [32] implies that uniform sampling of rows and columns, a special case of "subspace sampling," will produce a good skeleton representation $A_C(A_C^+AA_R^+)A_R$. However, it is not clear how the middle matrix $A_C^+AA_R^+$ can be computed in sublinear time.

In the main algorithm analyzed in this paper, we uniformly sample rows and columns to produce a skeleton of the form $A_C(A_C^+AA_R^+)A_R$, not $A_C(A_C^+AA_R^+)A_R$. One major difference is that the skeleton $A_C(A_C^+AA_R^+)A_R$ can be computed in $\tilde{O}(k^3)$ time.\footnote{Note that $\tilde{O}$ is the $O$ notation with log factors dropped.}

Note that the matrix skeleton output by our algorithms is represented by the index sets $R, C$ and matrix $Z$, not $A_R, A_C$.

Finally, let us mention that the term "skeleton" may refer to other factorizations. Instead of $A \simeq A_CZA_R$, we can have $A \simeq Z_1A_RZ_2$, where $Z_1, Z_2$ are arbitrary $m \times k$ and $k \times n$ matrices [15]. As $O(mnk + nk)$ space is needed to store $Z_1, Z_2$, this representation does not seem as appealing as the representation $A_CZA_R$ in memory-critical situations. Nevertheless, it is numerically more stable and has found several applications [25].

Alternatively, when $A = MBN$, where $M, B, N$ are $n \times n$ matrices, we can approximate $M$ as $M_CP$, $N$ as $DN_R$, where $M_C$ has $k$ columns of $M$ and $N_R$ has $k$ rows of $N$. Thus, $A \simeq M(C(P)D)N_R$, effectively replacing $B$ with the $k \times k$ matrix $\tilde{B} := PBD$. Bremer calls $\tilde{B}$ a skeleton and uses it to approximate scattering matrices [9].

1.4. Notation. The matrices we consider in this paper take the form

$$A = (X_1 \quad X_2) \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} Y_1^* \\ Y_2^* \end{pmatrix},$$

where $X = (X_1 \quad X_2)$ and $Y = (Y_1 \quad Y_2)$ are unitary matrices, with columns being "spread," and the blocks $A_{12}, A_{21},$ and $A_{22}$ are in some sense small. By "spread," we mean $\tilde{O}(1)$-coherent.

**Definition 1.1.** Let $X \in \mathbb{C}^{n \times k}$ be a matrix with $k$ orthonormal columns. Denote $\|X\|_{\text{max}} = \max_{ij} |X_{ij}|$. We say $X$ is $\mu$-coherent if $\|X\|_{\text{max}} \leq (\mu/n)^{1/2}$.

This notion is well known in compressed sensing [11] and matrix completion [10, 29].

Formally, let

$$\Delta_k := \begin{pmatrix} 0 & A_{12} \\ A_{21} & A_{22} \end{pmatrix},$$

and assume that

$$\varepsilon_k := \|\Delta_k\|$$

is small.

That means $A$ can be represented using only $O(k^2)$ data if we allow an $\varepsilon_k$ error in the operator norm. Note that $\varepsilon_k$ is equivalent to $\max(\|X_1^*A\|, \|AY_2\|)$ up to constants.

To prevent clutter, we have suppressed the dependence on $k$ from the definitions of $X_1, Y_1, A_{11}, A_{12}$ etc.

If (1) is the SVD of $A$, then $\varepsilon_k = \sigma_{k+1}(A)$. It is good to keep this example in mind as it simplifies many formulas that we see later.
An alternative to $\varepsilon_k$ is

$$
(3) \\
\varepsilon'_k := \sum_{i=1}^{m} \sum_{j=1}^{n} |(\Delta_k)_{ij}|.
$$

In other words, $\varepsilon'_k$ is the $\ell^1$ norm of $\Delta_k$ reshaped into a vector. We know $\varepsilon_k \leq \varepsilon'_k \leq mn\varepsilon_k$. The reason for introducing $\varepsilon'_k$ is that it is common for $(\Delta_k)_{ij}$ to decay rapidly such that $\varepsilon'_k \ll mn\varepsilon_k$. For such scenarios, the error guarantee of Algorithm 1 is much stronger in terms of $\varepsilon'_k$ than in terms of $\varepsilon_k$, as we will see in the next section.

### 1.5. Main result.

Random subsets of rows and columns are only representative of the subspaces of the matrix $A$ under the incoherence assumption mentioned earlier; otherwise Algorithm 1 may fail. For example, if $A = X_1 A_{11} Y_1^*$ and $X_1 = (I_{k \times k})$, then $A_{R_1}$ is going to be zero most of the time, and so is $A_{C} Z A_{R_1}$. Hence, it makes sense that we want $X_{1,R_1}$ to be “as nonsingular as possible” so that little information is lost. In particular, it is well known that if $X_1, Y_1$ are $O(1)$-coherent, i.e., spread, then sampling $\ell = O(k)$ rows will lead to $X_{1,R_1}, Y_{1,C}$: being well conditioned.\(^3\)

Here is our main result. It is proved in section 2.

**Theorem 1.1.** Let $A$ be given by (1) for some $k > 0$. Assume $m \geq n$ and $X_1, Y_1$ are $\mu$-coherent where $\mu = O(1)$ with respect to $m, n$. Recall the definitions of $\varepsilon_k, \varepsilon'_k$ in (2) and (3). Let $\ell \geq 10\mu k \log m$ and $\lambda = \frac{(mn)^{1/2}}{\ell}$. Then with probability at least $1 - 4km^{-2}$, Algorithm 1 returns a skeleton that satisfies

$$
(4) \quad \|A - A_{C} Z A_{R_1}\| = O(\lambda \delta + \lambda \varepsilon_k + \varepsilon'_k \lambda/\delta).
$$

If, furthermore, the entire $X$ and $Y$ are $\mu$-coherent, then with probability at least $1 - 4m^{-1}$,

$$
(5) \quad \|A - A_{C} Z A_{R_1}\| = O(\lambda \delta + \varepsilon'_k + \varepsilon'_k^2/\lambda \delta)).
$$

The right-hand sides of (4) and (5) can be minimized with respect to $\delta$. For (4), pick $\delta = \Theta(\varepsilon_k)$ so that

$$
(6) \quad \|A - A_{C} Z A_{R_1}\| = O(\varepsilon_k \lambda) = O(\varepsilon_k (mn)^{1/2}/\ell).
$$

For (5), pick $\delta = \Theta(\varepsilon'_k/\lambda)$ so that

$$
(7) \quad \|A - A_{C} Z A_{R_1}\| = O(\varepsilon'_k).
$$

Here are some possible scenarios where $\varepsilon'_k = o(\varepsilon_k \lambda)$ and (7) is strictly stronger than (6):

- The entries of $\Delta_k$ decay exponentially or there are only $O(1)$ nonzero entries as $m, n$ increases. Then $\varepsilon'_k = \Theta(\varepsilon_k)$.

- Say $n = m$ and (1) is the SVD of $A$. Suppose the singular values decay as $m^{-1/2}$. Then $\varepsilon'_k = O(\varepsilon_k m^{1/2})$.

---

\(^3\)Assume $\ell = O(k)$. Then $\|Y_1\|_{\infty,\infty} = O(n^{-1/2})$ is a sufficient condition for $Y_{1,C}$: to be well conditioned with high probability. This condition can be relaxed in at least two ways. First, all we need is that for each row $i$, $(\sum_j |(Y_1)_{ij}|^2)^{1/2} \leq (\mu k/n)^{1/2}$. This would allow a few entries of each row of $Y_1$ to be bigger than $O(n^{-1/2})$. Second, we can allow a few rows of $Y$ to violate the previous condition [2].

\(^4\)More generally, the algorithm runs in $O(n^3) = O(k^3 \mu^3 \log^3 m)$ time which is $o(mn)$ or sublinear if $k\mu = o((mn)^{1/3})$. 

---

*Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.*
One important question remains: How can we guess $\varepsilon_k$, in order to then choose $\delta$? Unfortunately, we are not aware of any $O(k^3)$ algorithm that can accurately estimate $\varepsilon_k$. Here is one possible heuristic for choosing $\delta$ for the case where (1) is the SVD. Imagine $A_{RC} \simeq X_{1,R}A_{11}Y_{1,C}$. As we will see, the singular values of $X_{1,R}, Y_{1,C}$ are likely to be on the order of $(\ell/m)^{1/2}, (\ell/n)^{1/2}$. Therefore, it is not unreasonable to view $\varepsilon_k \simeq \sigma_{k+1}(A) \simeq \lambda \sigma_{k+1}(A_{RC})$.

Another approach is to begin with a big $\delta$, run the $O(k^3)$ algorithm, check $\|A - A_{CZ}A_{R}\|$, divide $\delta$ by two and repeat the whole process until the error does not improve. However, calculating $\|A - A_{CZ}A_{R}\|$ is expensive and other tricks are needed. This seems to be an open problem.

The $O(k^3)$ algorithm is among the fastest algorithms for computing skeleton representations that one can expect to have. With more work, can the accuracy be improved? In section 3, we sketch two such algorithms. These two algorithms have, for the most part been analyzed in previous work: though their ability to perform, in sublinear-time complexity was not explicitly stated in those references, this fact should not come as a surprise. The first algorithm samples $\ell \simeq k \log m$ rows, columns, then reduces it to exactly $k$ rows, columns using a RRQR decomposition, with an operator norm error of $O(\varepsilon_k(mk)^{1/2})$. This idea is also employed in Boutsidis, Mahoney, and Drineas [7] which pertains to the column subset selection problem. In the second algorithm, we uniformly sample $\ell \simeq k \log m$ rows to get $A_{R}$, then run RRQR on $A_{R}$ to select $k$ columns of $A$. The overall error is $O(\varepsilon_k mn^{1/2})$. This is similar to the algorithm proposed in [27, 39] and more details can be found at the end of section 3.2.

Using the proof framework in section 2, we will derive error estimates for the above two algorithms. As mentioned earlier these error guarantees are not new, but (i) they concern provable sublinear-time complexity algorithms, (ii) they work for a more general model (1), and (iii) our proofs are also motivated differently. In section 3.3, we compare these three algorithms.

1.6. More on incoherence. If either $X$ or $Y$ is not $O(1)$-coherent, we can use the idea of a randomized Fourier transform [1] to impose incoherence. The idea is to multiply them on the left by the unitary Fourier matrix with randomly rescaled columns. This has the effect of “blending up” the rows of $X, Y$—at the possible cost of requiring linear-time complexity. The following is a standard result that can be proved using Hoeffding’s inequality.

**Proposition 1.2.** Let $X \in \mathbb{C}^{n \times k}$ with orthonormal columns. Let $D = \text{diag}(d_1, \ldots, d_n)$, where $d_1, \ldots, d_n$ are independent random variables such that $Ed_i = 0$ and $|d_i| = 1$. Let $F$ be the unitary Fourier matrix and let $\mu = \alpha n$ for some $\alpha > 0$. Define $U := FDX$. Then $\|U\|_{\max} \leq (\mu/n)^{1/2}$ with probability at least $1 - 2(nk)n^{-2\alpha}$.

In other words, no matter what $X$ is, $U = FDX$ would be $O(1)$-coherent with high probability. Hence, we can write a wrapper around Algorithm 1. Call this Algorithm 1’. Let $F \in \mathbb{C}^{n \times n}$ and $F' \in \mathbb{C}^{m \times m}$ be unitary Fourier transforms.

1. Let $B := F'D_2AD_1F'$, where $D_1, D_2$ are diagonal matrices with independent entries that are $\pm 1$ with equal probability.

2. Feed $B$ to the $O(k^3)$ algorithm and obtain $B \simeq B_{CZ}B_{R}$.

3. It follows that $A \simeq (AD_1F'_{CZ})Z(F_{R}D_2A)$.

The output $(AD_1F'_{CZ})Z(F_{R}D_2A)$ is not a matrix skeleton, but the amount of space needed is $O(n) + O(k^3)$ which is still better than $O(nk)$. Note that we are not storing $AD_1F'_{CZ}$ just as we do not store $A_{CZ}$ in Algorithm 1. Let $T_A$ be the cost of
matrix-vector multiplication of $A$. Then Algorithm 1' runs in $O(TA_k + mk + k^3)$ time. The most expensive step is computing $B_{RC}$ and it can be carried out as follows.

Compute $D_1(F^*S_C)$ in $O(nk)$ time. Multiply the result by $A$ on the left in $\tilde{O}(T_Ak)$ time. Multiply the result by $D_2$ on the left in $O(nk)$ time. Multiply the result by $F'$ on the left in $O(nk)$ time using FFT. Multiply the result by $S'_R$ on the left in $\tilde{O}(k^2)$ time.

2. Error estimates for $\tilde{O}(k^3)$ algorithm.

2.1. Notation. $S_C, S_R \in \mathbb{R}^{n \times k}$ are both column selector matrices. They are column subsets of permutation matrices. The subscripts “$R$” and “: $C$” denote a row subset and a column subset respectively, e.g., $A_R := S_R^T A$ and $A_C := A S_C$, while $A_{RC}$ is a row and column subset of $A$. Transposes and pseudoinverses are taken after the subscripts, e.g., $A_R^*$ means $(A_R)^*$.

2.2. Two principles. Our proofs are built on two principles. The first principle is due to Rudelson [31] in 1999. Intuitively, it says the following:

Let $Y$ be a $n \times k$ matrix with orthonormal columns. Let $Y_C$ be a random row subset of $Y$. Suppose $Y$ is $\mu$-coherent with $\mu = O(1)$, and $|C| = \ell \gtrsim \mu k$.

Then with high probability, $(\frac{2}{\pi})^{1/2} Y_C$ is like an isometry.

To be precise, we quote [37, Lemma 3.4]. Note that their $M$ is our $\mu k$.

**Theorem 2.1.** Let $Y \in \mathbb{C}^{n \times k}$ with orthonormal columns. Suppose $Y$ is $\mu$-coherent and $\ell \gtrsim \alpha \mu k$ for some $\alpha > 0$. Let $Y_C$ be a random $\ell$-row subset of $Y$. Each row of $Y_C$ is sampled independently, uniformly. Then

$$P \left( \|Y_C^+\| \geq \sqrt{\frac{n}{(1-\delta)^\ell}} \right) \leq k \left( \frac{e^{-\delta}}{(1-\delta)^{1-\delta}} \right)^\alpha \text{ for any } \delta \in [0,1)$$

and

$$P \left( \|Y_C\| \geq \sqrt{\frac{(1+\delta')\ell}{n}} \right) \leq k \left( \frac{e^{\delta'}}{(1+\delta')^{1+\delta'}} \right)^\alpha \text{ for any } \delta' \geq 0.$$

To be concrete, if $\delta = 0.57$ and $\delta' = 0.709$ and $\ell \geq 10k\mu \log n$, then

$$P \left( \|Y_C^+\| \leq 1.53(n/\ell)^{1/2} \text{ and } \|Y_C\| \leq 1.31(\ell/n)^{1/2} \right) \geq 1 - 2kn^{-2}.$$  

(8)

We will use (8) later. Let us proceed to the second principle, which says the following:

Let $C$ be an arbitrary index set. If $\|A_C\|$ is small, then $\|A\|$ is also small, provided that we have control over $\|AY_2\|$ and $\|Y^+_{1,C}\|$ for some unitary matrix $(Y_1 \ Y_2)$.

The roadmap is as follows. If we ignore the regularization step, then what we want to show is that $A \simeq A_C A_{RC}^+ A_R$. But when we take row and column restrictions on both sides, we have trivially $A_{RC} = A_{RC} A_{RC}^+ A_{RC}$. Hence, we desire a mechanism to go backwards, that is to infer that “$E := A - A_C A_{RC}^+ A_R$ is small” from “$E_{RC}$ is small.” We begin by inferring that “$E$ is small” from “$E_C$ is small.”

**Lemma 2.2.** Let $A \in \mathbb{C}^{m \times n}$ and let $Y = (Y_1 \ Y_2) \in \mathbb{C}^{m \times n}$ be a unitary matrix such that $Y_1$ has $k$ columns. Select $\ell \geq k$ rows of $Y_1$ to form $Y_{1,C} := S_C^T Y_1 \in \mathbb{C}^{\ell \times k}$. Assume $Y_{1,C}$ has full column rank. Then

$$\|A\| \leq \|Y^+_{1,C}\|\|A_C\| + \|Y^+_{1,C}\|\|AY_2 Y_{2,C}\| + \|AY_2\|.$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
Proof. Note that $Y^*_{1,C} Y^*_{1,C} = I_{k \times k}$. Now,

$$
\|A\| \leq \|AY_1\| + \|AY_2\|
= \|AY_1 Y^*_{1,C} Y^*_{1,C} \| + \|AY_2\|
\leq \|AY_1 Y^*_{1,C} SC \| \|Y^*_{1,C}\| + \|AY_2\|
\leq \|(A - AY_2 Y^*_{2,C}) SC\| \|Y^*_{1,C}\| + \|AY_2\|
\leq \|A_{RC}\| \|Y^*_{1,C}\| + \|AY_2 Y^*_{2,C}\| \|Y^*_{1,C}\| + \|AY_2\|
\tag*{\square}
$$

Lemma 2.2 can be extended in two obvious ways. First, we can deduce that "$A$ is small if $A_{RC}$ is small." Second, we can deduce that "$A$ is small if $A_{RC}$ is small." This is what the next lemma establishes. (Although its form looks unduly complicated, control over all the terms is in fine necessary.)

**Lemma 2.3.** Let $A \in C^{m \times n}$, and let $X = (X_1, X_2) \in C^{n \times m}$, and let $Y = (Y_1, Y_2) \in C^{n \times n}$ be unitary matrices such that $X_1, Y_1$ each has $k$ columns. Select $\ell \geq k$ rows and columns indexed by $R, C$, respectively. Assume $X_{1,R}, Y_{1,C}$ have full column rank. Then

$$
\|A\| \leq \|X_{1,R}^+\| \|A_{RC}\| + \|Y_{1,C}^+\| \|X_{2,R}^+ A Y_{2,C}^+\| + \|X_{2}^+ A\|
$$

and

$$
\|A\| \leq \|X_{1,R}^+\| \|Y_{1,C}^+\| \|A_{RC}\|
+ \|X_{1,R}^+\| \|Y_{1,C}^+\| \|X_{2,R}^+ A Y_{2,C}^+\| + \|X_{2}^+ A Y_{2,C}^+\|
= \|X_{1,R}^+\| \|Y_{1,C}^+\| \|S^T R X_1^+ A Y_{1,R}^+ Y_{1,C}^+\|
+ \|X_{1,R}^+\| \|X_{2,R}^+ A Y_{2,C}^+\| + \|X_{2}^+ A Y_{2,C}^+\|
\leq \|X_{1,R}^+\| \|S^T R X_1^+ A Y_{1,R}^+ Y_{1,C}^+\| \|Y_{1,C}^+\|
+ \|X_{1,R}^+\| \|X_{2,R}^+ A Y_{2,C}^+\| + \|X_{2}^+ A Y_{2,C}^+\|
= \|X_{1,R}^+\| \|Y_{1,C}^+\| \|S^T R (A - X_2^+ A Y_{2,C}^+ - X_1^+ A Y_{1,C}^+ Y_{1,R}^+) S C\|
+ \|X_{1,R}^+\| \|X_{2,R}^+ A Y_{2,C}^+\| + \|X_{2}^+ A Y_{2,C}^+\|.
$$

Proof. The top inequality is obtained by applying Lemma 2.2 to $A^*$. The proof of the bottom inequality is similar to the proof of Lemma 2.2. For completeness,

$$
\|A\| \leq \|X_{1,R}^+ A Y_{1,R}\| + \|X_{1,R}^+ A Y_{2}\| + \|X_{2}^+ A Y_{1}\| + \|X_{2}^+ A Y_{2}\|
= \|X_{1,R}^+ X_{1,R}^+ A Y_{1,R} Y_{1,C}^+ Y_{1,R}^+\|
+ \|X_{1,R}^+ X_{1,R}^+ A Y_{2}\| + \|X_{2}^+ A Y_{1,R} Y_{1,C}^+ Y_{1,R}^+\| + \|X_{2}^+ A Y_{2}\|
\leq \|X_{1,R}^+\| \|S^T R X_1^+ A Y_{1,R}^+ Y_{1,C}^+\| \|Y_{1,C}^+\|
+ \|X_{1,R}^+\| \|X_{2,R}^+ A Y_{2,C}^+\| + \|X_{2}^+ A Y_{2,C}^+\|
= \|X_{1,R}^+\| \|Y_{1,C}^+\| \|S^T R (A - X_2^+ A Y_{2,C}^+ - X_1^+ A Y_{1,C}^+ Y_{1,R}^+) S C\|
+ \|X_{1,R}^+\| \|X_{2,R}^+ A Y_{2,C}^+\| + \|X_{2}^+ A Y_{2,C}^+\|.
$$
Apply the triangle inequality to \(\|S_{F}(A - X_{2}X_{2}^{*}AY_{1}Y_{1}^{*} - X_{1}X_{1}^{*}AY_{2}Y_{2}^{*} - X_{2}X_{2}^{*}AY_{2})S_{C}\|\) and we are done. \(\square\)

We conclude this section with a useful corollary. It says that if \(PA_{C}\) is a good low rank approximation of \(A_{C}\) for some \(P \in \mathbb{C}^{m \times m}\), then \(PA\) may also be a good low rank approximation of \(A\).

**Corollary 2.4.** Let \(A \in \mathbb{C}^{m \times n}\) and \(P \in \mathbb{C}^{m \times m}\). Let \(Y = (Y_{1} \ Y_{2})\) be a unitary matrix such that \(Y_{1}\) has \(k\) columns. Let \(Y_{1,C} = S_{C}^{*}Y_{1} \in \mathbb{C}^{\ell \times k}\), where \(\ell \geq k\). Assume \(Y_{1,C}\) has full column rank. Let \(I \in \mathbb{C}^{m \times m}\) be the identity. Then

\[
\|A - PA\| \leq \|Y_{1,C}^{+}\|\|A_{C} - PA_{C}\| + \|Y_{1,C}^{+}\|\|I - P\|\|AY_{2}Y_{2,C}^{*}\| + \|I - P\|\|AY_{2}\|.
\]

In particular, if \(P\) is the orthogonal projection \(A_{C}A_{C}^{*}\), then

\[
\|A - A_{C}A_{C}^{+}A\| \leq \|Y_{1,C}^{+}\|\|AY_{2}Y_{2,C}^{*}\| + \|AY_{2}\|.
\]

**Proof.** To get the first inequality, apply Lemma 2.2 to \(A - PA\). The second inequality is immediate from the first inequality since \(\|A_{C} - A_{C}A_{C}^{+}A_{C}\| = 0\). \(\square\)

For the special case where \(X, Y\) are singular vectors of \(A\), (9) can be proved using the technique that \(\|A - A_{C}A_{C}^{+}A\| = \min_{D}\|A - A_{C}D\|\) and choosing an appropriate \(D\). See Boutsidis, Mahoney, and Drineas [7].

Note that (9) can be strengthened to \(\|A - A_{C}A_{C}^{+}A\|^{2} \leq \|AY_{2}Y_{2,C}^{*}Y_{1,C}^{+}\|^{2} + \|AY_{2}\|^{2}\) by modifying the first step of the proof of Lemma 2.2 from \(\|A\| \leq \|AY_{1}\| + \|AY_{2}\|\) to \(\|A\|^{2} \leq \|AY_{1}\|^{2} + \|AY_{2}\|^{2}\). A similar result for the case where \(X, Y\) are singular vectors can be found in Halbo, Martinsson, and Tropp [24]. The originality of our results is that they hold for a more general model (1).

### 2.3. Proof of Theorem 1.1.

The proof is split into two main parts. The first part is probabilistic. We will apply the first principle to control the largest and smallest singular values of \(Y_{1,C}\), \(X_{1,R}\), and other similar quantities. The second part, mainly linear algebra, uses these bounds on \(Y_{1,C}\), and \(X_{1,R}\), to help control the error \(\|A - A_{C}B_{RC}^{+}A_{R}\|\).

#### 2.3.1. Probabilistic part.

Let \(\lambda_{X} = (\mathcal{P}_{m}^{2})^{1/2}\) and \(\lambda_{Y} = (\mathcal{P}_{n}^{2})^{1/2}\). To prove the first part of Theorem 1.1, i.e., (4), we apply Theorem 2.1. From (8), it is clear that the assumption of Theorem 1.1 guarantee that \(\|Y_{1,C}\| = O(\lambda_{Y}^{-1})\), \(\|Y_{1,C}^{+}\| = O(\lambda_{Y})\), \(\|X_{1,R}\| = O(\lambda_{X}^{-1})\), \(\|X_{1,R}^{+}\| = O(\lambda_{X})\) hold simultaneously with probability at least \(1 - 4km^{-2}\).

For the second part of Theorem 1.1, i.e., (5), we need to refine (1) as follows. Let \(X = (\tilde{X}_{1}, \ldots, \tilde{X}_{[m/k]}\) and \(Y = (\tilde{Y}_{1}, \ldots, \tilde{Y}_{[n/k]}\)\), where \(\tilde{X}_{1}, \ldots, \tilde{X}_{[m/k]-1}\) and \(\tilde{Y}_{1}, \ldots, \tilde{Y}_{[n/k]-1}\) have \(k\) columns, and \(\tilde{X}_{[m/k]}, \tilde{Y}_{[n/k]}\) have \(\leq k\) columns. Note that \(\tilde{X}_{1} = X_{1}, \tilde{Y}_{1} = Y_{1}, \tilde{A}_{11} = A_{11}\), where \(X_{1}, Y_{1}, A_{11}\) are defined in (1). Rewrite (1) as

\[
A = (\tilde{X}_{1}, \ldots, \tilde{X}_{[m/k]}) \begin{pmatrix}
\tilde{A}_{11} & \cdots & \tilde{A}_{1,[n/k]} \\
\vdots & \ddots & \vdots \\
\tilde{A}_{[m/k],1} & \cdots & \tilde{A}_{[m/k],[n/k]}
\end{pmatrix} \begin{pmatrix}
\tilde{Y}_{1}^{*} \\
\vdots \\
\tilde{Y}_{[n/k]}^{*}
\end{pmatrix}
\]
By applying Theorem 2.1 to every $\tilde{X}_i$, $\tilde{Y}_j$ and doing a union bound, we see that with probability at least $1 - 4m^{-1}$, we will have $\|Y_{j,c}\| = O(\lambda_Y^{-1})$, $\|Y^+_{j,c}\| = O(\lambda_Y)$, $\|X_{i,R}\| = O(\lambda_X^{-1})$, $\|X^+_{i,R}\| = O(\lambda_X)$ for all $i,j$.

**2.3.2. Deterministic part: Introducing $B$, an auxiliary matrix.** Recall that in Algorithm 1, we compute the SVD of $A_{RC}$ as $U_1 \Sigma_1 V^*_1 + U_2 \Sigma_2 V^*_2$ and invert only $U_1 \Sigma_1 V^*_1$ to get the center matrix $Z$.

Define $B \in \mathbb{C}^{m \times n}$ such that $B_{RC} = U_1 \Sigma_1 V^*_1$ and all other entries of $B$ are the same as $A$’s. In other words, define $E \in \mathbb{C}^{m \times n}$ such that $E_{RC} = -U_2 \Sigma_2 V^*_2$ and all other entries of $E$ are zeros, and then let $B = A + E$.

The skeleton returned is $A_{C}B_{RC}^+A_{R}$. By construction,

$$\|A - B\| \leq \delta; \quad \|B_{RC}^+\| \leq \delta^{-1}.$$ 

Our objective is to bound $\|A - A_{C}B_{RC}^+A_{R}\|$, but it is $\|B - B_{C}B_{RC}^+B_{R}\|$ that we have control over by the second principle. Recall that $B_{RC} = B_{RC}B_{RC}B_{RC}$ is to be lifted to $B \simeq B_{C}B_{RC}B_{R}$ by Lemma 2.3. Thus, we shall first relate $\|A - A_{C}B_{RC}^+A_{R}\|$ to quantities involving only $B$ by a perturbation argument:

$$\begin{align*}
\|A - A_{C}B_{RC}^+A_{R}\| & \leq \|A - B\| + \|B - B_{C}B_{RC}^+B_{R}\| \\
& \quad + \|B_{C}B_{RC}^+B_{R} - A_{C}B_{RC}^+B_{R}\| + \|A_{C}B_{RC}^+B_{R} - A_{C}B_{RC}^+A_{R}\| \\
& \quad \leq \delta + \|B - B_{C}B_{RC}^+B_{R}\| \\
& \quad + \|(B - A)S_C\| \|B_{RC}^+B_{R}\| + \|A_{C}B_{RC}^+\| \|S_C^T(B - A)\| \\
& \quad \leq \delta + \|B - B_{C}B_{RC}^+B_{R}\| \\
& \quad + \|B_{RC}B_{R}\| + \|B_{C}B_{RC}^+\| + \|(A - B)S_C\| \delta^{-1} \delta \\
& \leq 2\delta + \|B - B_{C}B_{RC}B_{R}\| + \delta \|B_{RC}B_{R}\| + \delta \|B_{C}B_{RC}^+\|.
\end{align*}$$

(11)

**2.3.3. Deterministic part: Bounds on $\|B_{RC}^+B_{R}\|$ and $\|B_{C}B_{RC}^+\|$.** It remains to bound $\|B - B_{C}B_{RC}^+B_{R}\|$, $\|B_{RC}B_{R}\|$, and $\|B_{C}B_{RC}^+\|$. In this subsection, we obtain bounds for the last two quantities. By the second principle, we do not expect $\|B_{RC}B_{R}\|$ to be much bigger than $\|B_{RC}^+B_{R}\| \leq 1$. Specifically, by Lemma 2.2, we have

$$\begin{align*}
\|B_{RC}^+B_{R}\| & \leq \|Y_{1,c}\| \|B_{RC}^+B_{R}\| + \|Y^+_{1,c}\| \|B_{RC}^+B_{R}Y_{2}Y^*_2\| + \|B_{RC}^+B_{R}Y_{2}\| \\
& \leq \|Y_{1,c}\| + \|Y^+_{1,c}\| \|B_{RC}^+\| (\|B_{R} - A_{R}\|Y_{2}Y^*_2) + \|A_{R}Y_{2}\|) \\
& \quad + \|B_{RC}^+\| (\|B_{R} - A_{R}\|Y_{2}) + \|A_{R}Y_{2}\|) \\
& \leq \|Y_{1,c}\| + \|Y^+_{1,c}\| \delta^{-1}(\delta + \|A_{R}Y_{2}\|) + \delta^{-1}(\delta + \|A_{R}Y_{2}\|) \\
& \leq 1 + 2\|Y^+_{1,c}\| + \|Y^+_{1,c}\| \delta^{-1} \|A_{R}Y_{2}\| + 2\|A_{R}Y_{2}\|.
\end{align*}$$

By the first principle, the following holds with high probability:

$$\begin{align*}
\|B_{RC}^+B_{R}\| & = O(\lambda_Y + \lambda_Y \delta^{-1} \|A_{R}Y_{2}\| + \delta^{-1} \|A_{R}Y_{2}\|).
\end{align*}$$

(12)
The same argument works for $\|B_C B_{RC}^+\|$. With high probability,

$$\|B_C B_{RC}^+\| = O(\lambda_X + \lambda_Y \delta^{-1} \|X_{2,R} X^*_2 A_C\| + \delta^{-1} \|X^*_2 A_C\|).$$

2.3.4. Deterministic part: Bounding $\|B - B_C B_{RC}^+ B_{R^c}\|$. Bounding the third quantity requires more work, but the basic ideas are the same. Recall that the second principle suggests that $\|B - B_C B_{RC}^+ B_{R^c}\|$ cannot be too much bigger than $\|B_{RC} - B_{RC} B_{RC}^+ B_{RC}\| = 0$. Applying Lemma 2.3 with $B - B_C B_{RC}^+ B_{R^c}$ in the role of $A$ yields

$$\|B - B_C B_{RC}^+ B_{R^c}\| \leq \left\|X_{1,R}^+\right\| \left\|Y_{1,C}^+\right\| \left\|Y_{1,C}\right\| (\|X_{2,R} X^*_2 B\| + \|X_{2,R} X^*_2 B_{RC}\| \|B_{RC}^+ B_{R^c}\|)$$

$$+ \left\|X_{1,R}^+\right\| \left\|Y_{1,C}^+\right\| \left\|X_{1,R}\right\| (\|BY_{2,C}^+\| + \|B_{R} Y_{2,C}^+\| \|B_{RC}^+\|)$$

$$+ \left\|X_{1,R}^+\right\| \left\|Y_{1,C}^+\right\| (\|X_{2,R} X^*_2 B Y_{2,C}^+\| + \|X_{2,R} X^*_2 B_{RC}\| \delta^{-1} \|B_{R} Y_{2,C}^+\|)$$

$$+ \left\|X_{1,R}^+\right\| \left\|X_{1,R}\right\| (\|BY_{2}\| + \|B_{RC}^+ B_{R^c}\| \|B_{R^c} Y_{2}\|)$$

$$+ \left\|Y_{1,C}^+\right\| \left\|Y_{1,C}\right\| (\|X_{2}^* B\| + \|B_{RC}^+ B_{R^c}\| \|X_{2}^* B_{RC}\|)$$

$$+ \|X_{2}^* B Y_{2}\| + \|X_{2}^* B_{RC}\| \delta^{-1} \|B_{R^c} Y_{2}\|.$$  

In the expression above, we have paired $\|X_{1,R}\|$ with $\|X_{1,R}^+\|$, and $\|Y_{1,C}\|$ with $\|Y_{1,C}^+\|$ because the first principle implies that their products are $O(1)$ with high probability. This implies that $\|B - B_C B_{RC}^+ B_{R^c}\|$ is less than a constant times,

$$\lambda_X (\|X_{2,R} X^*_2 B\| + \|X_{2,R} X^*_2 B_{RC}\| \|B_{RC}^+ B_{R^c}\|)$$

$$+ \lambda_Y (\|BY_{2,C}^+\| + \|B_{R} Y_{2,C}^+\| \|B_{RC}^+\|)$$

$$+ \lambda_X \lambda_Y (\|X_{2,R} X^*_2 B Y_{2,C}^+\| + \|X_{2,R} X^*_2 B_{RC}\| \delta^{-1} \|B_{R} Y_{2,C}^+\|)$$

$$+ \|BY_{2}\| + \|B_{RC}^+ B_{R^c}\| \|B_{R^c} Y_{2}\|$$

$$+ \|X_{2}^* B\| + \|B_{RC}^+ B_{R^c}\| \|X_{2}^* B_{RC}\|$$

$$+ \|X_{2}^* B_{RC}\| \delta^{-1} \|B_{R^c} Y_{2}\|.$$  

We have dropped $\|X_{2}^* B Y_{2}\|$ because it is dominated by $\|X_{2}^* B\|$. Equations (13) and (12) can be used to control $\|B_C B_{RC}^+\|$ and $\|B_{RC}^+ B_{R^c}\|$. Before doing so, we
want to replace \( B \) with \( A \) in all the other terms. This will introduce some extra \( \delta \)'s. For example, \( \|X_{2.R:2}X_2^*B\| \leq \|X_{2.R:2}X_2^*B - X_{2.R:2}X_2^*A\| + \|X_{2.R:2}X_2^*A\| \leq \delta + \|X_{2.R:2}X_2^*A\| \). Doing the same for other terms, we have that \( \|B - B_C B_{RC}^+ B_R\| \) is with high probability less than a constant times,

\[
\begin{align*}
\lambda X (\delta + \|X_{2.R:2}X_2^*A\| + (\delta + \|X_{2.R:2}X_2^*A_C\|) \|B_{RC}^+ B_R\|) \\
+ \lambda Y (\delta + \|AY_2Y_2^*C\| + (\delta + \|A_R:2Y_2^*C\|) \|B_C B_{RC}^+\|) \\
+ \lambda X Y (\delta + \|X_{2.R:2}AY_2Y_2^*C\| + \|X_{2.R:2}X_2^*A_C\|) \\
+ \|A_R:2Y_2Y_2^*C\| + \|X_{2.R:2}X_2^*A_C\| \|\epsilon^{-1}\|A_R:2Y_2\|
\end{align*}
\]

Several terms can be simplified by noting that \( \delta \leq \lambda X \delta \leq \lambda X Y \delta \) and \( \|X_2^*A_C\| \leq \|X_2^*A\| \). We shall also use the estimates on \( |B_C B_{RC}^+| \) and \( |B_{RC}^+ B_R| \), from (13) and (12). This leads to

\[
\lambda X (\delta + \|X_{2.R:2}X_2^*A\| + (\delta + \|X_{2.R:2}X_2^*A_C\|) \|A_R:2Y_2Y_2^*C\|) + \lambda Y (\delta + \|AY_2Y_2^*C\| + (\delta + \|A_R:2Y_2Y_2^*C\|) \|X_2^*A_C\|) + \lambda X Y (\delta + \|X_{2.R:2}AY_2Y_2^*C\| + \|X_{2.R:2}X_2^*A_C\|) + \|A_R:2Y_2Y_2^*C\| + \|X_2^*A_C\| \|\epsilon^{-1}\|A_R:2Y_2\|.
\]

Collect the terms by their \( \lambda X, \lambda Y \) factors and drop the smaller terms to obtain that with high probability,

\[
(14) \quad \|B - B_C B_{RC}^+ B_R\| \\
= O(\lambda X (\delta + \|X_{2.R:2}X_2^*A\| + \|A_R:2Y_2\| + \delta^{-1} \|X_{2.R:2}X_2^*A_C\| \|A_R:2Y_2\|) \\
+ \lambda Y (\delta + \|AY_2Y_2^*C\| + \delta^{-1} \|X_2^*A_C\|) + \|A_R:2Y_2Y_2^*C\|) \\
+ \lambda X Y (\delta + \|X_{2.R:2}AY_2Y_2^*C\| + \|X_{2.R:2}X_2^*A_C\|) + \|A_R:2Y_2Y_2^*C\|) \\
+ \delta^{-1} \|X_{2.R:2}X_2^*A_C\| \|A_R:2Y_2Y_2^*C\| + \|X_{2.R:2}X_2^*AY_2Y_2^*C\|) \\
+ \|\delta^{-1} \|X_2^*A_C\| \|\epsilon^{-1}\|A_R:2Y_2\|).
\]

2.3.5. Deterministic part: Conclusion of the proof. We now have control over all three terms \( \|B - B_C B_{RC}^+ B_R\|, \|B_{RC}^+ B_R\|, \|B_C B_{RC}^+\| \). Substitute (12), (13), (14) into (11). As the right-hand side of (14) dominates \( \delta \) multiplied by the right-hand side of (12), (13), we conclude that with high probability, \( \|A - A_C B_{RC}^+ A_R\| \) is also bounded by the right-hand side of (14).

To obtain the basic bound, (4), we note that all the “normed” terms on the right-hand side of (14), e.g., \( \|A_R:2Y_2^*C\| \) and \( \|X_2^*A\| \), are bounded by \( \epsilon_k \). It follows that with high probability, \( \|A - A_C B_{RC}^+ A_R\| = O(\lambda X \lambda Y (\delta + \epsilon + \delta^{-1} \epsilon^2)) \).
To obtain the other bound, (5), we need to bound each “normed term” of (14) differently. Recall (10). Consider $\|X_{2,R}X_{2}\ast\|_C$. We have

$$X_{2,R}X_{2}^{\ast}A_{C} = (\tilde{X}_{2,R},\ldots,\tilde{X}_{[m/k],R})$$

\[
\begin{pmatrix}
\tilde{A}_{21} & \ldots & \tilde{A}_{2,[n/k]}
\vdots & \ddots & \vdots
\tilde{A}_{[m/k],1} & \ldots & \tilde{A}_{[m/k],[n/k]}
\end{pmatrix}
\cdot
\begin{pmatrix}
\tilde{Y}_{1,C} \\
\vdots \\
\tilde{Y}_{[n/k],C}
\end{pmatrix}.
\]

In section 2.3.1, we show that with high probability, $\|\tilde{X}_{i,R}\| = O(\lambda_{X}^{-1})$ and $\|\tilde{Y}_{j,C}\| = O(\lambda_{Y}^{-1})$ for all $i, j$. Recall the definition of $\varepsilon_{k}'$ in (3). It follows that with high probability,

$$\|X_{2,R}X_{2}^{\ast}A_{C}\| \leq \sum_{i=2}^{[m/k]} \sum_{j=1}^{[n/k]} \|\tilde{X}_{i,R}\| \|\tilde{A}_{ij}\| \|\tilde{Y}_{j,C}\| \leq \lambda_{X}^{-1}\lambda_{Y}^{-1}\varepsilon_{k}.'$$

Apply the same argument to other terms on the right-hand side of (14), e.g., $\|X_{2,R}X_{2}^{\ast}AY_{2,C}^{\ast}\| = O(\lambda_{X}^{-1}\lambda_{Y}^{-1}\varepsilon_{k}')$ and $\|X_{2}^{\ast}A_{C}\| = O(\lambda_{Y}^{-1}\varepsilon_{k}')$ with high probability. Monotonically, a $R$ in the subscript leads to a $\lambda_{X}^{-1}$ and a $C$ in the subscript leads to a $\lambda_{Y}^{-1}$.

Recall that $\|A_{C}B_{RC}^{\ast}A_{R}\|$ is bounded by the right-hand side of (14). Upon simplifying, we obtain that $\|A - A_{C}B_{RC}^{\ast}A_{R}\| = O(\lambda_{X}\lambda_{Y}\delta + \varepsilon_{k}' + \lambda_{X}\lambda_{Y}\delta^{-1}\varepsilon_{k}^{'2})$, i.e., (5). The proof is complete.

3. Alternative sublinear-time algorithms.

3.1. Second algorithm. In Algorithm 2, uniform random sampling first helps to trim down $A$ to two factors, $A_{C}$ and $A_{R}^{\ast}$ with $|C| = |R| = \ell = O(k)$, and then rank-revealing QR decompositions (RRQR) are used on $A_{C}$ and $A_{R}^{\ast}$ to further reduce the small dimension to exactly $k$.

For dense matrices, the most expensive step in Algorithm 2 is the multiplication of $A$ by $A_{R}^{\ast}$. However, for structured matrices, the most expensive steps of Algorithm 2 are likely to be the RRQR factorization of $A_{C}$ and $A_{R}^{\ast}$ and the inversion of $A_{C}^{\ast}, A_{R}^{\ast}$, which all take $O(mk^2)$ time. The overall running time is $O(T_{A}k) + O(mk^2)$, where $T_{A}$ is the cost of a matrix-vector multiplication.

Note that in the MATLAB code, the call $rrqr(A,k)$ is assumed to return an index set of size $k$ specifying the selected columns. One can use Algorithm 782 [6] or its MATLAB port [34].

It can be easily shown [19] that once $A_{C}^{\ast}, A_{R}^{\ast}$, are fixed, the choice of $Z = A_{C}^{\ast}A_{R}^{\ast}$ is optimal in the Frobenius norm (not operator norm), that is, $Z = \arg_{W\epsilon \in \mathbb{C}^{\ell \times k}} \|A - A_{C}^{\ast}W A_{R}^{\ast}\|_{F}$. Unsurprisingly, the error estimate is better than in Theorem 1.1.

**Theorem 3.1.** Let $A$ be given by (1) for some $k > 0$. Assume $m \geq n$ and $X_{1}, Y_{1}$ are $\mu$-coherent where $\mu = O(1)$ with respect to $m, n$. Recall the definition of $\varepsilon_{k}$ in (2). Let $\ell \geq 10mk\log m$. With probability at least $1 - 4km^{-2}$, Algorithm 2 returns a skeleton that satisfies

$$\|A - A_{C}^{\ast}Z A_{R}^{\ast}\| = O(\varepsilon_{k}(mk)^{1/2}).$$
Algorithm 2. \(O(T_Ak) + \tilde{O}(mk^2)\)-time algorithm

Input: A matrix \(A \in \mathbb{C}^{m \times n}\) that is approximately rank \(k\), and user-defined parameter \(\ell = \hat{O}(k)\). Assume \(m \geq n\).

Output: Column index set \(C'\) of size \(k\), row index set \(R'\) of size \(k\), center matrix \(Z\) of a matrix skeleton \(A_{C',ZA_{R'}}\).

Steps:
1. Let \(C\) be a random index set of size \(\ell\) chosen uniformly from \(\{1, \ldots, n\}\). Explicitly form \(A_C\).
2. Let \(R\) be a random index set of size \(\ell\) chosen uniformly from \(\{1, \ldots, m\}\). Explicitly form \(A_R\).
3. Run RRQR on \(A_C\) to select \(k\) columns of \(A_C\). Denote the result as \(A_{C'}\) where \(C' \subseteq C\) indexes the \(k\) selected columns of \(A\). This takes \(O(mk^2)\) time and \(O(mk)\) space.
4. Run RRQR on \(A_R^*\) to select \(k\) rows of \(A_R^*\). Denote the result as \(A_{R'}\) where \(R' \subseteq R\) indexes the \(k\) selected rows of \(A\). This takes \(O(nk)\) time and \(O(nk)\) space.
5. Compute \(Z = A_{C'}^T (AA_{R'}^*)\). This takes \(O(T_Ak + mk^2)\) time and \(O(mk)\) space, where \(T_A\) is the time needed to apply \(A\) to a vector.

MATLAB code:
```matlab
function [Cp,Z,Rp]=skeleton2(A,l)
    C=rankperm(n,l); ind=rrqr(A(:,C),k); Cp=C(ind);
    R=randperm(m,l); ind=rrqr(A(R,:),k); Rp=R(ind);
    Z=pinv(A(:,Cp))*(A*pinv(A(Rp,:)));
end
```

Proof. Let \(P = A_{C'}A_{C'}^+ \in \mathbb{C}^{m \times m}\). RRQR [23] selects \(k\) columns from \(A_C\) such that

\[
\|A_C - PA_C\| \leq f(k, \ell)\sigma_{k+1}(A_C) \leq f(k, \ell)\sigma_{k+1}(A) \leq f(k, \ell)\varepsilon_k,
\]

where \(f(k, \ell) := \sqrt{1 + 2k(\ell - k)}\). We have used the fact that \(\sigma_{k+1}(A) = \sigma_{k+1}(A_{11}, A_{12}) \leq \sigma_1(A_{12}) \leq \varepsilon_k\). See interlacing theorems [26, Corollary 3.1.3].

Recall from (8) that \(\|Y_{1,C}^+\| = O((n/\ell)^{1/2})\) with probability at least \(1 - 2km^{-2}\). Apply Corollary 2.4 to obtain that with high probability,

\[
\|A - PA\| \leq O(\lambda_Y)\|A_C - PA_C\| + O(\lambda_Y)\|A_C - PA_{C'}\| + O(\varepsilon_k)\|Y_{1,C}^+\| + O(\varepsilon_k(n/\ell)^{1/2}f(k, \ell)) = O(\varepsilon_kn/\ell)^{1/2}f(k, \ell).
\]

Let \(P' = A_{R'}^+A_{R'}\). By the same argument, \(\|A - AP'\| = O(\varepsilon_k^{1/2})\) with the same failure probability. Combine both estimates. With probability at least \(1 - 4km^{-2}\),

\[
\|A - A_{C'}A_{C'}^+AA_{R'}^+A_{R'}\| = \|A - PA\| \leq \|A - PA\| + \|PA - PAP'\|
\]

\[
\leq \|A - PA\| + \|A - AP'\| = O(\varepsilon_kn/\ell)^{1/2}f(k, \ell).
\]

\(\square\)

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
Algorithm 3. $\tilde{O}(nk^2)$-time algorithm

Input: A matrix $A \in \mathbb{C}^{m \times n}$ that is approximately rank $k$, and user-defined parameter $\ell = \tilde{O}(k)$.

Output: Column index set $C'$ of size $k$, row index set $R$ of size $\ell$, center matrix $Z$ of a matrix skeleton. Implicitly, we have the matrix skeleton $A_{C'}ZA_{R'}$.

Steps:
1. Let $R$ be a random index set of size $\ell$ chosen uniformly from $\{1, \ldots, m\}$. Explicitly form $A_R$.
2. Run RRQR on $A_R$ and obtain a column index set $C'$. Note that $A_R \simeq A_{RC'}(A_{RC'}^TA_{R'})$, where $A_{RC'}$ contains $k$ columns of $A_R$. This takes $\tilde{O}(nk^2)$ time and $\tilde{O}(nk)$ space.
3. Compute $Z = A_{R'}^k$. This takes $\tilde{O}(k^3)$ time and $\tilde{O}(k^2)$ space.

MATLAB code:

```matlab
function [Cp,Z,R]=skeleton3(A,l)
R=randperm(m,1); Cp=rrqr(A(R,:),k); Z=pinv(A(R,Cp));
end
```

Many algorithms that use the skeleton $A_{C'}(A_{C'}^TA_{C'}^+)A_{R'}$, e.g., in [28], seek to select columns indexed by $C$ such that $\|A - A_{C'}A_{C'}^+A\|$ is small. Here, we further select $k$ out of $\ell = \tilde{O}(k)$ columns, which is also suggested in [7]. Their estimate on the error in the operator norm is $O(k \log^{1/2}k)\varepsilon_k + O(k^{3/4} \log^{1/4}k)\|A - A_k\|_F$, where $A_k$ is the optimal rank $k$ approximation to $A$. In general, $\|A - A_k\|_F$ could be as large as $(n - k)^{1/2}\varepsilon_k$, which makes our bound better by a factor of $k^{1/4}$. Nevertheless, we make the extra assumption that $X_1, Y_1$ are incoherent.

**3.2. Third algorithm.** Consider the case where only $X_1$ is $\tilde{O}(1)$-coherent. See Algorithm 3. It computes a skeleton with $\tilde{O}(k)$ rows and $k$ columns in $\tilde{O}(nk^2 + k^3)$ time. Intuitively, the algorithm works as follows. We want to select $k$ columns of $A$ but running RRQR on $A$ is too expensive. Instead, we randomly choose $\tilde{O}(k)$ rows to form $A_R$, and select our $k$ columns using the much smaller matrix $A_{R'}$. This works because $X_1$ is assumed to be $\tilde{O}(1)$-coherent and choosing almost any $\tilde{O}(k)$ rows will give us a good sketch of $A$.

**Theorem 3.2.** Let $A$ be given by (1) for some $k > 0$. Assume $m \geq n$ and $X_1$ is $\mu$-coherent with respect to $m, n$. ($Y_1$ does not need to be incoherent.) Recall the definition of $\varepsilon_k$ in (2). Let $\ell \geq 10\mu k \log m$. Then, with probability at least $1 - 2km^{-2}$, Algorithm 3 returns a skeleton that satisfies

$$\|A - A_{C'}ZA_{R'}\|_F = O(\varepsilon_k mn^{1/2}).$$

**Proof.** We perform RRQR on $A_R$ to obtain $A_R \simeq A_{RC'}D$, where $D = A_{RC'}^TA_{R'}$ and $C'$ indexes the selected $k$ columns. We want to use the second principle to “undo the row restriction” and infer that $A \simeq A_{C'}D$, the output of Algorithm 3. The details are as follows.

Strong RRQR [23] guarantees that

$$\|A_{R'} - A_{RC'}D\| \leq \sigma_{k+1}(A_{R'})f(k,n) \leq \sigma_{k+1}(A)f(k,n) \leq \varepsilon_k f(k,n)$$

and

$$\|D\| \leq f(k,n),$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
where \( f(k, n) = \sqrt{1 + 2k(n - k)} \). Prepare to apply a transposed version of Corollary 2.4, i.e.,

\[
\| A - AP \| \leq \left\| X_{1, R}^+ \right\|=\| A_{R^c} - A_{R^c} P \|
+ \left\| X_{1, R}^+ \right\|=\| I - P \| \left\| X_{2, R}^+ X_{1, R}^+ A \right\| + \left\| I - P \right\| \left\| X_{2} A \right\|.
\]

Let \( P = S_{C^c} D \), so that \( \| P \| \leq \| D \| \leq f(k, n) \). Note that \( AP = A_{C^c} A^+_{R^c} A_{R^c} \). By (8), with probability at least \( 1 - 2km^{-2} \), \( \| X_{1, R}^+ \| = O((m/\ell)^{1/2}) \). By (15),

\[
\| A - AP \| \leq O(\lambda_X) \| A_{R^c} - A_{R^c} D \| + O(\lambda_X)(1 + \| P \|)\varepsilon_k + (1 + \| P \|)\varepsilon_k
= O(\varepsilon_k f(k, n)(m/\ell)^{1/2}) = O(\varepsilon_k (mn)^{1/2}).
\]

If \( X_1 \) is not incoherent and we fix it by multiplying on the left by a randomized Fourier matrix \( F \) (cf. section 1.6), then we arrive at the algorithm in [27]. The linear algebraic part of their proof combined with the first principle will lead to similar bounds. What we have done here is split the proof into three simple parts: (1) show that \( X_{1, R} \) is incoherent, (2) use the first principle to show that \( X_{1, R} \) is “sufficiently nonsingular,” (3) apply the second principle.

### 3.3. Comparison of the Three Algorithms

Here is a summary of the three algorithms studied in this paper. Assume \( m \geq n \). Recall that \( A \simeq X_1 A_{11} Y_1^* \). For Algorithms 1 and 2, assume that \( X_1, Y_1 \) are both incoherent. For Algorithm 3, assume that \( X_1 \) is incoherent. See Table 1.

Recall that \( T_A \) is the cost of applying \( A \) to a vector. If \( T_A = \tilde{O}(nk) \) and \( m = O(n) \), then the running time of Algorithms 2 and 3 are comparable and we would recommend using Algorithm 2 because it has a better error guarantee.

Otherwise, if \( T_A \) is on the order of \( mn \), then Algorithm 2 is much slower than Algorithms 1 and 3, and is not recommended. Compared to Algorithm 3, and in that scenario, Algorithm 1 is much faster and has better error guarantees, so we view it as the better choice. The advantages of Algorithm 3 are that it selects exactly \( k \) columns and does not require \( Y_1 \) to be incoherent.

If we cannot afford using \( \tilde{O}(mk) \) memory or having a running time that scales with \( m, n \), then Algorithm 1 is the only possible choice here. Although Theorem 1.1 suggests that the error for Algorithm 1 grows with \( (mn)^{1/2} \), we believe that in practice, the error usually increases with \( m^{1/2} \). See section 4 for some numerical results.

Finally, we remind the reader that these recommendations are made based on error guarantees which are not always tight.

### 4. Examples

#### 4.1. First Toy Example: Convolution

This first example shows that in Algorithm 1 it is crucial to regularize when inverting \( A_{RC} \) because, otherwise, the error

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>No. of rows</th>
<th>No. of columns</th>
<th>Upper bound on error in the operator norm</th>
<th>Running time</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 1</td>
<td>( \ell = \tilde{O}(k) )</td>
<td>( \ell = \tilde{O}(k) )</td>
<td>( O(\varepsilon_k (mn)^{1/2}) )</td>
<td>( \tilde{O}(k^2) )</td>
<td>( \tilde{O}(k^2) )</td>
</tr>
<tr>
<td>Algorithm 2</td>
<td>( k )</td>
<td>( k )</td>
<td>( O(\varepsilon_k (mk)^{1/2}) )</td>
<td>( O(T_A k) + \tilde{O}(mk^2) )</td>
<td>( \tilde{O}(mk) )</td>
</tr>
<tr>
<td>Algorithm 3</td>
<td>( \tilde{O}(k) )</td>
<td>( k )</td>
<td>( O(\varepsilon_k (mn)^{1/2}) )</td>
<td>( \tilde{O}(nk^2) )</td>
<td>( \tilde{O}(nk) )</td>
</tr>
</tbody>
</table>
in the operator norm can blow up. In fact, even when $A$ is positive definite and we pick $C = R$ as in the work of Gittens [21], we encounter the same need to regularize. The reason is that due to numerical errors, $A_{RC}$ tends to be ill-conditioned when $A_{RC}$ has more rows and columns than the rank of $A$. In other words, numerical errors introduce spurious small but nonzero singular values in $A_{RC}$ and inverting the components corresponding to these small singular values leads to large errors.

The experiment is set up as follows. Let $A = X \Sigma X^* \in \mathbb{C}^{n \times n}$, where $X$ is the unitary Fourier matrix and $\Sigma$ is a diagonal matrix of singular values. Note that every entry of $X$ is of magnitude $n^{-1/2}$ and $X$ is 1-coherent. Fix $n = 301$, $\ell = 100$, and $k = 10, 30, 50$. Pick $\varepsilon = \varepsilon_k = \sigma_{k+1} = \cdots = \sigma_n = 10^{-15}$. Pick the largest $k$ singular values to be logarithmically spaced between 1 and $\varepsilon$. Note that $A$ is Hermitian and positive definite. In each random trial, we randomly shuffle the singular values, pick $\ell$ random rows and columns, and measure $\|A - A_{RC} Z A_R\|$. The only parameters being varied are $k$ and $\delta$. Note that although $R \neq C$ in this experiment, similar results are obtained when $R = C$.

From (4) in Theorem 1.1, we expect that when variables such as $n, m, \ell, k$ are fixed,

\begin{equation}
\log \|A - A_{RC} Z A_R\| \sim \log (\delta^{-1} (\varepsilon_k + \delta)^2) = -\log \delta + 2 \log (\varepsilon_k + \delta).
\end{equation}

Consider a plot of $\|A - A_{RC} Z A_R\|$ versus $\delta$ on a log-log scale. According to the above equation, when $\delta \ll \varepsilon_k$, the first term dominates and we expect to see a line of slope $-1$, and when $\delta \gg \varepsilon_k$, $\log (\varepsilon_k + \delta) \approx \log \delta$, and we expect to see a line of slope $+1$. Indeed, when we plot the experimental results in Figure 1, we see a right-angled

\begin{figure}
\centering
\includegraphics[width=\textwidth]{image1.png}
\caption{Log-log plot of the empirical mean of the error in operator norm by the $\tilde{O}(k^3)$ algorithm versus $\delta$, a regularization parameter. This relationship between the error and $\delta$ agrees with Theorem 1.1. See (16). More importantly, the error blows up for small $\delta$, which implies that the regularization step should not be omitted.}
\end{figure}
JIAWEI CHIU AND LAURENT DEMANET

Fig. 2. The above is a log-log plot of the empirical mean of the error in operator norm versus $n$, the size of square matrix $A$. Fix $k = 10$, $\ell = 40$, and $A = X\Sigma Y^*$, where $X, Y$ are unitary Fourier matrices with randomly permuted columns and $\Sigma$ is the diagonal matrix of singular values. The top $k$ singular values are set to 1 and the others are set to $\varepsilon = 10^{-6}$. When we run Algorithm 1 with $\delta = \varepsilon, \varepsilon/\sqrt{n}, \varepsilon/n$, the expected errors seem to grow with $n^{0.55}, n^{0.51}, n^{0.69}$, respectively. For Algorithms 2 and 3, the expected errors seem to grow with $n^{0.52}, n^{0.43}$, respectively. The errorbars correspond to $\frac{1}{\sqrt{n}}$ of the standard deviations obtained empirically. Observe that the error in Algorithm 3 fluctuates much more than Algorithm 1 with $\delta = \varepsilon, \varepsilon/\sqrt{n}$.

V-curve.

The point here is that the error in the operator norm can blow up as $\delta \to 0$.

A curious feature of Figure 1 is that the error curves resemble staircases. As we decrease $k$, the number of distinct error levels seems to decrease proportionally. A possible explanation for this behavior is that the top singular vectors of $A_C$ match those of $A$, and as $\delta$ increases from $\sigma_i(A)$ to $\sigma_{i-1}(A)$ for some small $i$, smaller components will not be inverted and the error is all on the order of $\sigma_i(A)$.

4.2. Second toy example. For the second experiment, we consider $A = X\Sigma Y^*$ where $X, Y$ are unitary Fourier matrices with randomly permuted columns and $\Sigma$ is the diagonal matrix of singular values. Fix $k = 10$, $\ell = 40$. The singular values are set such that the largest $k$ singular values are all 1 and the other singular values are all $\varepsilon = 10^{-6}$. We consider all three algorithms. For Algorithm 1, we set $\delta$ in three different ways: $\delta = \varepsilon, \delta = \varepsilon/\sqrt{n}$, and $\delta = \varepsilon/n$.

We plot the error $\| A - A_C Z A_R \|$ versus $n$ in Figure 2. The numerical results show that if we pick $\delta = \varepsilon/\sqrt{n}$ for Algorithm 1, then the estimated mean error is almost the same as that of Algorithm 2—they both scale with $n^{0.51}$, with $k, \ell$ fixed. On the other hand, if we pick $\delta = \varepsilon$ as suggested by (4) of Theorem 1.1, the expected error seems to grow with $n^{0.55}$, which is slightly worse than Algorithm 2 but much better than described in (6).

The expected error of Algorithm 3 seems to grow with $n^{0.43}$, which is the best in
this experiment. However, compared with Algorithms 2 and 1 with \( \delta = \varepsilon, \varepsilon/\sqrt{n} \), its error is not as concentrated around the mean.

4.3. Smooth kernel. Consider a one-dimensional integral operator with a kernel \( K \) that is analytic on \([-1,1]^2\). Define \( A \) as \( (A)_{ij} = cK(x_i,y_j) \), where the nodes \( x_1, \ldots, x_n \) and \( y_1, \ldots, y_n \) are uniformly spaced in \([-1,1]\). First, suppose \( K = \sum_{1 \leq i,j \leq 6} c_{ij}T_i(x)T_j(y) + 10^{-3}T_{10}(x)T_{10}(y) + 10^{-9}N \), where \( T_i(x) \) is the ith Chebyshev polynomial and \( N \) is the random Gaussian matrix, i.e., noise. The coefficients \( c_{ij} \)'s are chosen such that \( \| A \| \simeq 1 \). Pick \( n = m = 10^3 \) and slowly increase \( \ell \), the number of rows and columns sampled by the \( \tilde{O}(k^3) \) algorithm. As shown in Figure 3, the skeleton representation \( A_CZAR \) converges rapidly to \( A \) as we increase \( \ell \).

Next, consider \( K(x,y) = c \exp(xy) \). Let \( n = 900 \) and pick \( c \) to normalize \( \| A \| = 1 \). We then plot the empirical mean of the error of the \( \tilde{O}(k^3) \) algorithm against \( \ell \) on the left of Figure 4. Notice that the error decreases exponentially with \( \ell \).

To understand what is happening, imagine that the grid is infinitely fine. Let \( \varphi_1, \varphi_2, \ldots \) be Legendre polynomials. Recall that these polynomials are orthogonal on
[−1, 1]. Define the matrices $X, Y$ as $(X)_{ij} = \varphi_j(x_i)$ and $(Y)_{ij} = \varphi_j(y_i)$. Assume the $\varphi_i$’s are scaled such that $X, Y$ are unitary. It is well known that if we expand $K$ in terms of Chebyshev polynomials or Legendre polynomials [8] or prolate spheroidal wave functions [40], the expansion coefficients will decay exponentially. This means that the entries of $X^* A Y$ should decay exponentially away from the topleft corner and $\varepsilon_k' = \Theta(\varepsilon_k)$ (cf. (2) and (3)). We confirm this by plotting $\varepsilon_k, \varepsilon_k'$ versus $k$ on the right of Figure 4. The actual $X, Y$ used to obtain this plot are obtained by evaluating the Legendre polynomials on the uniform grid and orthonormalizing. It can be verified that the entries of $X, Y$ are of magnitude $O((k/n)^{1/2})$, which implies a coherence $\mu \approx k$, independent of $n$. The implication is that the algorithm will continue to perform well as $n$ increases.

As $\ell$ increases, we can apply Theorem 1.1 with a larger $k$. Since $\varepsilon_k, \varepsilon_k'$ decrease exponentially, the error should also decrease exponentially. However, as $k$ increases beyond $\simeq 15$, $\varepsilon_k$ stagnates and nothing can be gained from increasing $\ell$. In general, as $\varepsilon_k$ decreases, we should pick a smaller $\delta$. But when $k \gtrsim 15$, choosing a smaller $\delta$ does not help and may lead to worse results due to the instability of pseudoinverses. This is evident from Figure 4.

A recent paper by Platte, Trefethen, and Kuijlaars [30] states that we cannot have an exponential decrease of the error without a condition number that grows exponentially. In our case, the random selection of columns and rows corresponds to selecting interpolation points randomly, and $\delta$ serves as a regularization parameter of the interpolation method. Due to the regularization, we can only expect an exponential decrease of the error up to a limit dependent on $\delta$. 
4.4. **Fourier integral operators.** In [12], Candes, Demanet, and Ying consider how to efficiently apply two-dimensional Fourier integral operators of the form

\[ Lf(x) = \int_\xi a(x, \xi) e^{2\pi i \Phi(x, \xi)} \hat{f}(\xi) d\xi, \]

where \( \hat{f}(\xi) \) is the Fourier transform of \( f \), \( a(x, \xi) \) is a smooth amplitude function, and \( \Phi \) is a smooth phase function that is homogeneous, i.e., \( \Phi(x, \lambda \xi) = \lambda \Phi(x, \xi) \) for any \( \lambda > 0 \). Say there are \( N^2 \) points in the space domain and also the frequency domain.

The main idea is to split the frequency domain into \( N \) wedges, perform a Taylor expansion of \( \Phi(x, \cdot) \) about \( \xi \mid \xi_j \), where \( j \) indexes a wedge, and observe that the residual phase \( \Phi_j(x, \xi) := \Phi(x, \xi) - \Phi(x, \xi_j) \cdot \xi \) is nonoscillatory. Hence, the matrix \( A^{(j)}_{st} := \exp(2\pi i \Phi_j(x_s, \xi_t)) \) can be approximated by a low rank matrix, i.e., \( \exp(2\pi i \Phi_j(x, \xi)) \) can be written as \( \sum_{q=1}^r f_q(x) g_q(\xi) \) where \( r \), the separation rank, is independent of \( N \).

By switching the order of summations, the authors arrive at \( O(N^{2.5}) \) algorithms for both the preprocessing and the evaluation steps. See [12] for further details.

What we are concerned with here is the approximate factorization of \( A^{(j)} \). This is a \( N^2 \) by \( N^{1.5} \) matrix since there are \( N^2 \) points in the space domain and \( N^2/\sqrt{N} \) points in one wedge in the frequency domain. In [12], a slightly different algorithm is proposed:

1. Uniformly and randomly select \( \ell \) rows and columns to form \( A_R \) and \( A_C \).
2. Perform SVD on \( A_C \). Say \( A_C = U_1 A_1 V_1^* + U_2 A_2 V_2^* \), where \( U, V \) are unitary and \( \|A_2\| \leq \delta \), a user specified parameter.
3. Return the low rank representation \( U_1 U_{1,R}^+ A_R \).

In the words of the authors, “this randomized approach works well in practice although we are not able to offer a rigorous proof of its accuracy, and expect one to be non-trivial” [12].

We are now in a position to explain why this randomized approach works well. Consider (1) and (2). Let \( B \) be a perturbation of \( A \) such that \( B_C = U_1 A_1 V_1^* \) and \( A - B \leq \delta \). Since \( A_1 \) is invertible, the output can be rewritten as

\[ U_1 U_{1,R}^+ A_R = B_C B_{RC}^+ A_R. \]

By following the proof of Theorem 1.1, we see that

\[ \|A - B_C B_{RC}^+ A_R\| = O(\|B - B_C B_{RC}^+ B_R\|) \]

and that all the estimates in Theorem 1.1 must continue to hold.

The analysis presented here, therefore, answers the questions posed in [12]. We believe that the assumption of incoherence of the generating vectors is precisely the right framework to express the error guarantees of the skeleton in such situations.

An important subclass of Fourier integral operators is pseudodifferential operators. These are linear operators with pseudodifferential symbols that obey certain smoothness conditions [35]. In [16], a similar randomized algorithm is used to derive low rank factorizations of such smooth symbols. It is likely that the method works well here in the same way as it works well for a smooth kernel as discussed in the previous section.

**Acknowledgments.** We would also like to thank the anonymous referees for making many valuable suggestions.
REFERENCES


