

An Improved Approximation Algorithm for the Column Subset Selection Problem *

Christos Boutsidis [†] Michael W. Mahoney [‡] Petros Drineas [§]

Abstract

We consider the problem of selecting the “best” subset of *exactly* k columns from an $m \times n$ matrix A . In particular, we present and analyze a novel two-stage algorithm that runs in $O(\min\{mn^2, m^2n\})$ time and returns as output an $m \times k$ matrix C consisting of exactly k columns of A . In the first stage (the *randomized* stage), the algorithm randomly selects $O(k \log k)$ columns according to a judiciously-chosen probability distribution that depends on information in the top- k right singular subspace of A . In the second stage (the *deterministic* stage), the algorithm applies a deterministic column-selection procedure to select and return exactly k columns from the set of columns selected in the first stage. Let C be the $m \times k$ matrix containing those k columns, let P_C denote the projection matrix onto the span of those columns, and let A_k denote the “best” rank- k approximation to the matrix A as computed with the singular value decomposition. Then, we prove that

$$\|A - P_C A\|_2 \leq O\left(k^{\frac{3}{4}} \log^{\frac{1}{2}}(k) (n - k)^{\frac{1}{4}}\right) \|A - A_k\|_2,$$

with probability at least 0.7. This spectral norm bound improves upon the best previously-existing result (of Gu and Eisenstat [23]) for the spectral norm version of this Column Subset Selection Problem. We also prove that

$$\|A - P_C A\|_F \leq O\left(k\sqrt{\log k}\right) \|A - A_k\|_F,$$

with the same probability. This Frobenius norm bound is only a factor of $\sqrt{k \log k}$ worse than the best previously existing existential result and is roughly $O(\sqrt{k!})$ better than the best previous algorithmic result (both of Deshpande et al. [12]) for the Frobenius norm version of this Column Subset Selection Problem.

1 Introduction

We consider the problem of selecting the “best” set of *exactly* k columns from an $m \times n$ matrix A . More precisely, we consider the following COLUMN SUBSET SELECTION PROBLEM (CSSP):

Definition 1 (The CSSP) *Given a matrix $A \in \mathbb{R}^{m \times n}$ and a positive integer k , pick k columns of A forming a matrix $C \in \mathbb{R}^{m \times k}$ such that the residual*

$$\|A - P_C A\|_{\xi}$$

is minimized over all possible $\binom{n}{k}$ choices for the matrix C . Here, $P_C = CC^+$ denotes the projection onto the k -dimensional space spanned by the columns of C and $\xi = 2$ or F denotes the spectral norm or Frobenius norm.

*A conference proceedings version of this paper will appear in SODA 2009 [5]; the main extension with this version is the empirical evaluation we present in Section 5.

[†]Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, boutsc@cs.rpi.edu.

[‡]Department of Mathematics, Stanford University, Stanford, CA, mmahoney@cs.stanford.edu.

[§]Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, drinep@cs.rpi.edu.

That is, the goal of the CSSP is to find a subset of exactly k columns of A that “captures” as much of A as possible, with respect to the spectral norm and/or Frobenius norm, in a projection sense. The CSSP has been studied extensively in numerical linear algebra, where it has found applications in, e.g., scientific computing [7]. More recently, a relaxation has been studied in theoretical computer science, where it has been motivated by applications to large scientific and internet data sets [16].

1.1 Complexity of the CSSP

We briefly comment on the complexity of the problem. Clearly, in $O(n^k)$ time we can generate all possible matrices C and thus solve the problem exactly, i.e., find the optimal solution. However, from a practical perspective, in data analysis applications of the CSSP (see Section 1.2), n is often in the order of hundreds or thousands. Thus, in practice, algorithms that run in $O(n^k)$ time are prohibitively slow even if k is, from a theoretical perspective, a constant. Finally, the NP-hardness of the CSSP is an interesting open problem. Note, though, that a similar problem asking for the k columns of the $m \times n$ matrix A that maximize the volume of the parallelepiped spanned by the columns of C , is provably NP-hard [11].

1.2 The CSSP in statistical data analysis

In data applications, where the input matrix A models m objects represented with respect to n features, the CSSP corresponds to unsupervised feature selection. Standard motivations for feature selection include facilitating data visualization, reducing training times, avoiding overfitting, and facilitating data understanding.

Consider, in particular, Principal Components Analysis (PCA), which is the predominant linear dimensionality reduction technique, and which has been widely applied on datasets in all scientific domains, from the social sciences and economics, to biology and chemistry. In words, PCA seeks to map or embed data points from a high dimensional Euclidean space to a low dimensional Euclidean space while keeping all the relevant linear structure intact. PCA is an unsupervised dimensionality reduction technique, with the sole input parameters being the coordinates of the data points and the number of dimensions that will be retained in the embedding (say k), which is typically a constant independent of m and n ; often it is $k \ll \{m, n\}$ too. Data analysts often seek a subset of k actual features (that is, k actual columns, as opposed to the k eigenvectors or eigenfeatures returned by PCA) that can accurately reproduce the structure derived by PCA. The CSSP is the obvious optimization problem associated with such unsupervised feature selection tasks.

We should note that similar formulations appeared in [25, 35, 37, 39, 28, 1]. In addition, applications of such ideas include: (i) [36], where a “compact CUR matrix decomposition” was applied to static and dynamic data analysis in large sparse graphs; (ii) [26, 27, 14], where these ideas were used for compression and classification of hyperspectral medical data and the reconstruction of missing entries from recommendation systems data in order to make high-quality recommendations; and (iii) [32], where the concept of “PCA-correlated SNPs” (Single Nucleotide Polymorphisms) was introduced and applied to classify individuals from throughout the world without the need for any prior ancestry information. See Section 5 for an empirical evaluation of our main algorithm; and see [4] for a detailed evaluation of our main algorithm as an unsupervised feature selection strategy in three application domains of modern statistical data analysis (finance, document-term data, and genetics).

1.3 Our main results

We present a novel two-stage algorithm for the CSSP. This algorithm is presented in detail in Section 3 as Algorithm 1. In the first stage of this algorithm (the *randomized stage*), we randomly select $O(k \log k)$ columns of V_k^T , i.e., of the transpose of the $n \times k$ matrix consisting of the top k right singular vectors of A , according to a judiciously-chosen probability distribution that depends on information in the top- k right singular subspace of A . Then, in the second stage (the *deterministic stage*), we apply a deterministic column-selection procedure to select exactly k columns from the set of columns of V_k^T selected by the first stage. The algorithm then returns the corresponding k columns of A . In Section 4 we prove the following theorem.

Theorem 1 *There exists an algorithm (the two-stage Algorithm 1) that approximates the solution to the CSSP. This algorithm takes as input an $m \times n$ matrix A of rank $\rho \leq \min\{m, n\}$ and a positive integer k ; it runs in $O(\min\{mn^2, m^2n\})$ time; and it returns as output an $m \times k$ matrix C consisting of exactly k columns of A such that with probability at least 0.7:*

$$\begin{aligned} \|A - P_C A\|_2 &\leq O\left(k^{3/4} \log^{1/2}(k) (\rho - k)^{1/4}\right) \|A - A_k\|_2, \\ \|A - P_C A\|_F &\leq O\left(k \log^{1/2} k\right) \|A - A_k\|_F. \end{aligned}$$

Here, $P_C = CC^+$ denotes a projection onto the column span of the matrix C , and A_k denotes the best rank- k approximation to the matrix A as computed with the singular value decomposition.

Note that we can trivially boost the success probability in the above theorem to $1 - \delta$ by repeating the algorithm $O(\log(1/\delta))$ times. Note also that the running time of our algorithm is linear in the larger of the dimensions m and n , quadratic in the smaller one, and independent of k . Thus, it is practically useful and efficient.

To put our results into perspective, we compare them to the best existing results for the CSSP. Prior work provided bounds of the form

$$\|A - P_C A\|_\xi \leq p(k, n) \|A - A_k\|_\xi, \quad (1)$$

where $p(k, n)$ is a polynomial on n and k . For $\xi = 2$, i.e., for the spectral norm, the best previously-known bound for approximating the CSSP is $p(k, n) = O\left(\sqrt{1 + k(n - k)}\right)$ [23], while for $\xi = F$, i.e., for the Frobenius norm, the best bound is $p(k, n) = \sqrt{(k + 1)!}$ [12]. Both results are algorithmically efficient, running in time polynomial in all three parameters m , n , and k ; the former runs in $O(\min\{mn^2, m^2n\})$ time and the latter runs in $O(mnk + kn)$ time. Thus, our approach asymptotically improves the best previously-known result for the spectral norm version of the CSSP by a factor of $n^{1/4}$. (Here we assume that k is independent of n , as is typically the case in data applications of these techniques.) Our approach also provides an algorithmic bound for the Frobenius norm version of the CSSP that is roughly $O(\sqrt{k!})$ better than the best previously-known algorithmic result. It should be noted that [12] also proves that by exhaustively testing all $\binom{n}{k}$ possibilities for the matrix C , the best one will satisfy eqn. (1) with $p(k, n) = \sqrt{k + 1}$. Our algorithmic result is only $O(\sqrt{k \log k})$ worse than this existential result. A similar existential result for the spectral norm version of the CSSP is proved in [24] with $p(k, n) = \sqrt{1 + k(n - k)}$. Obviously, the result in [12] is a lower bound for the Frobenius norm version of the CSSP. On the other hand, the result in [24] is the best existing existential result for the spectral norm version of the CSSP. A lower bound for the spectral norm version of the CSSP, which should be between $\|A - A_k\|_2$ and $\sqrt{1 + k(n - k)}\|A - A_k\|_2$, is an interesting open problem. These results are summarized in Table 1.

Existential Result				Algorithmic Result		
	Ref	$p(k,n)$	Time	Ref	$p(k,n)$	Time
$\xi = 2$	[24]	$\sqrt{k(n-k)+1}$	$O(n^k)$	[23]	$O(k^{\frac{1}{2}}(n-k)^{\frac{1}{2}})$	$O(mn^2)$
$\xi = F$	[13]	$\sqrt{k+1}$	$O(n^k)$	[13]	$\sqrt{(k+1)!}$	$O(mnk)$

Our Result	
$p(k,n)$	Time
$O(k^{\frac{3}{4}}(n-k)^{\frac{1}{4}})$	$O(mn^2)$
$O(k\sqrt{\log k})$	$O(mn^2)$

Table 1: Comparison of the results of this paper with the state-of-the-art existential and algorithmic results for the CSSP. Here, $p(k, n)$ is involved in the approximation $\|A - P_{CA}\|_{\xi} \leq p(k, n) \|A - A_k\|_{\xi}$. (In addition, $m \geq n$ for this table.)

Finally, we should emphasize that a novel feature of the algorithm that we present in this paper is that it combines in a nontrivial manner recent algorithmic developments in the theoretical computer science community with more traditional techniques from the numerical linear algebra community in order to obtain improved bounds for the CSSP.

2 Background and prior work

2.1 Notation and linear algebra

Let $[n]$ denote the set $\{1, 2, \dots, n\}$. For any matrix $A \in \mathbb{R}^{m \times n}$, let $A_{(i)}, i \in [m]$ denote the i -th row of A as a row vector, and let $A^{(j)}, j \in [n]$ denote the j -th column of A as a column vector. In addition, let $\|A\|_F^2 = \sum_{i,j} A_{ij}^2$ denote the square of its Frobenius norm, and let $\|A\|_2 = \sup_{x \in \mathbb{R}^n, x \neq 0} |Ax|_2 / |x|_2$ denote its spectral norm. If $A \in \mathbb{R}^{m \times n}$, then the Singular Value Decomposition (SVD) of A can be written as

$$\begin{aligned}
 A &= U_A \Sigma_A V_A^T \\
 &= \begin{pmatrix} U_k & U_{\rho-k} \end{pmatrix} \begin{pmatrix} \Sigma_k & \mathbf{0} \\ \mathbf{0} & \Sigma_{\rho-k} \end{pmatrix} \begin{pmatrix} V_k^T \\ V_{\rho-k}^T \end{pmatrix}.
 \end{aligned}$$

In this expression, $\rho \leq \min\{m, n\}$ denotes the rank of A , $U_A \in \mathbb{R}^{m \times \rho}$ is an orthonormal matrix, Σ_A is a $\rho \times \rho$ diagonal matrix, and $V_A \in \mathbb{R}^{n \times \rho}$ is an orthonormal matrix. Also, Σ_k denotes the $k \times k$ diagonal matrix containing the top k singular values of A , $\Sigma_{\rho-k}$ denotes the $(\rho - k) \times (\rho - k)$ matrix containing the bottom $\rho - k$ singular values of A , V_k denotes the $n \times k$ matrix whose columns are the top k right singular vectors of A , and $V_{\rho-k}$ denotes the $n \times (\rho - k)$ matrix whose columns are the bottom $\rho - k$ right singular vectors of A , etc.

The $m \times k$ orthogonal matrix U_k consisting of the top k left singular vectors of A is the “best” set of k linear combinations of the columns of A , in the sense that $A_k = P_{U_k} A = U_k \Sigma_k V_k^T$ is the “best” rank k approximation to A . Here, $P_{U_k} = U_k U_k^T$ is a projection onto the k -dimensional space spanned by the columns of U_k . In particular, A_k minimizes $\|A - A'\|_{\xi}$, for both $\xi = 2$ and F , over all $m \times n$ matrices A' whose rank is at most k . We also denote $A_{\rho-k} = U_{\rho-k} \Sigma_{\rho-k} V_{\rho-k}^T$. We will use the notation $\|\cdot\|_{\xi}$ when writing an expression that holds for both the spectral and the Frobenius norm. We will subscript the norm by 2 and F when writing expressions that hold for one norm or the other. Finally, the Moore-Penrose generalized inverse, or pseudoinverse, of A , denoted by A^+ , may be expressed in terms of the SVD as $A^+ = V_A \Sigma_A^{-1} U_A^T$.

2.2 Related prior work

Since solving the CSSP exactly is a hard combinatorial optimization problem, research has historically focused on computing approximate solutions to it. Since $\|A - A_k\|_\xi$ provides an immediate lower bound for $\|A - P_C A\|_\xi$, for $\xi = 2, F$ and for any choice of C , a large number of approximation algorithms have been proposed to select a subset of k columns of A such that the resulting matrix C satisfies

$$\|A - A_k\|_\xi \leq \|A - P_C A\|_\xi \leq p(k, n) \|A - A_k\|_\xi$$

for some function $p(k, n)$. Within the numerical linear algebra community, most of the work on the CSSP has focused on spectral norm bounds and is related to the so-called RANK REVEALING QR (RRQR) FACTORIZATION:

Definition 2 (The RRQR factorization) *Given a matrix $A \in R^{m \times n}$ ($m \geq n$) and an integer k ($k \leq n$), assume partial QR factorizations of the form:*

$$A\Pi = QR = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix},$$

where $Q \in R^{m \times n}$ is an orthonormal matrix, $R \in R^{n \times n}$ is upper triangular, $R_{11} \in R^{k \times k}$, $R_{12} \in R^{k \times (n-k)}$, $R_{22} \in R^{(n-k) \times (n-k)}$, and $\Pi \in R^{n \times n}$ is a permutation matrix. The above factorization is called a RRQR factorization if it satisfies

$$\begin{aligned} \frac{\sigma_k(A)}{p_1(k, n)} &\leq \sigma_{\min}(R_{11}) \leq \sigma_k(A) \\ \sigma_{k+1}(A) &\leq \sigma_{\max}(R_{22}) \leq p_2(k, n)\sigma_{k+1}(A), \end{aligned}$$

where $p_1(k, n)$ and $p_2(k, n)$ are functions bounded by low degree polynomials in k and n .

The work of Golub on pivoted QR factorizations [21] was followed by much research addressing the problem of constructing an efficient RRQR factorization. Most researchers improved RRQR factorizations by focusing on improving the functions $p_1(k, n)$ and $p_2(k, n)$ in Definition 2. Let Π_k denote the first k columns of a permutation matrix Π . Then, if $C = A\Pi_k$ is an $m \times k$ matrix consisting of k columns of A , it is straightforward to prove that

$$\|A - P_C A\|_\xi = \|R_{22}\|_\xi,$$

for both $\xi = 2, F$. Thus, in particular, when applied to the spectral norm, it follows that

$$\|A - P_C A\|_2 \leq p_2(k, n)\sigma_{k+1}(A) = p_2(k, n) \|A - A_k\|_2,$$

i.e., any algorithm that constructs an RRQR factorization of the matrix A with provable guarantees also provides provable guarantees for the CSSP. See Table 2 for a summary of existing results, and see [19] for a survey and an empirical evaluation of some of these algorithms. More recently, [29, 38] proposed random-projection type algorithms that achieve the same spectral norm bounds as prior work while improving the running time.

Within the theoretical computer science community, much work has followed that of Frieze, Kannan, and Vempala [20] on selecting a small subset of representative columns of A , forming a matrix C , such that the projection of A on the subspace spanned by the columns of C is as close to A as possible. The algorithms from this community are randomized, which means that they come with a failure probability, and focus mainly on the Frobenius norm. It is worth noting that they provide a strong tradeoff between the number of selected columns and the desired approximation

Method	Reference	$\mathbf{p}(\mathbf{k}, \mathbf{n})$	Time
Pivoted QR	[Golub, 1965] [21]	$\sqrt{(n-k)2^k}$	$O(mnk)$
High RRQR	[Foster, 1986] [18]	$\sqrt{n(n-k)2^{n-k}}$	$O(mn^2)$
High RRQR	[Chan, 1987] [6]	$\sqrt{n(n-k)2^{n-k}}$	$O(mn^2)$
RRQR	[Hong and Pan, 1992] [24]	$\sqrt{k(n-k)+k}$	$O(n^k)$
Low RRQR	[Chan and Hansen, 1994] [8]	$\sqrt{(k+1)n}2^{k+1}$	$O(mn^2)$
Hybrid-I RRQR	[Chandrasekaran and Ipsen, 1994] [9]	$\sqrt{(k+1)(n-k)}$	$O(n^k)$
Hybrid-II RRQR	[9]	$\sqrt{(k+1)(n-k)}$	$O(n^k)$
Hybrid-III RRQR	[9]	$\sqrt{(k+1)(n-k)}$	$O(n^k)$
Strong RRQR	[Gu and Eisenstat, 1996] [23]	$\sqrt{k(n-k)+1}$	$O(n^k)$
Strong RRQR	[23]	$O(\sqrt{k(n-k)+1})$	$O(mn^2)$
DGEQPY	[Bischof and Orti, 1998] [3]	$O(\sqrt{(k+1)^2(n-k)})$	-
DGEQPX	[3]	$O(\sqrt{(k+1)(n-k)})$	$O(n^k)$
SPQR	[Stewart, 1999] [34]	-	-
PT Algorithm 1	[Pan and Tang, 1999] [31]	$O(\sqrt{(k+1)(n-k)})$	-
PT Algorithm 2	[31]	$O(\sqrt{(k+1)^2(n-k)})$	-
PT Algorithm 3	[31]	$O(\sqrt{(k+1)^2(n-k)})$	-
Pan Algorithm 2	[Pan, 2000] [30]	$O(\sqrt{k(n-k)+1})$	$O(mn^2)$

Table 2: Accuracy of deterministic algorithms for the CSSP. A dash means that the algorithm either runs in $O(n^k)$ time, or the authors do not provide a running time bound. (In addition, $m \geq n$ for this table.)

accuracy. A typical scenario for these algorithms is that the desired approximation error (see ϵ below) is given as input, and then the algorithm selects the minimum number of appropriate columns in order to achieve this error. One of the most relevant results for this paper is a bound of [12], which states that there exist exactly k columns in any $m \times n$ matrix A such that

$$\|A - CC^+A\|_F \leq \sqrt{k+1} \|A - A_k\|_F.$$

Here, C contains exactly k columns of A . The only known algorithm to find these k columns is to try all $\binom{n}{k}$ choices and keep the best. This existential result relies on the so-called volume sampling method [12, 13]. In [13], an adaptive sampling method is used to approximate the volume sampling method and leads to an $O(mnk + kn)$ algorithm which finds k columns of A such that

$$\|A - CC^+A\|_F \leq \sqrt{(k+1)!} \|A - A_k\|_F.$$

As mentioned above, much work has also considered algorithms choosing slightly more than k columns. This relaxation provides significant flexibility and improved error bounds. For example, in [13], an adaptive sampling method leads to an $O(mn(k/\epsilon^2 + k^2 \log k))$ algorithm, such that

$$\|A - CC^+A\|_F \leq (1 + \epsilon) \|A - A_k\|_F$$

holds with high probability for some matrix C consisting of $O(k/\epsilon^2 + k^2 \log k)$ columns of A . Similarly, in [15, 16], Drineas, Mahoney, and Muthukrishnan leverage the subspace sampling method to give an $O(\min\{mn^2, m^2n\})$ algorithm such that

$$\|A - CC^+A\|_F \leq (1 + \epsilon) \|A - A_k\|_F \quad (2)$$

holds with high probability if C contains at most $O(k \log k / \epsilon^2)$ columns of A .

3 A two-stage algorithm for the CSSP

In this section, we present and describe Algorithm 1, our main algorithm for approximating the solution to the CSSP. This algorithm takes as input an $m \times n$ matrix A and a rank parameter k . After an initial setup, the algorithm has two stages: a randomized stage and a deterministic stage. In the *randomized stage*, a randomized procedure is run to select $O(k \log k)$ columns from the $k \times n$ matrix V_k^T , i.e., the transpose of the matrix containing the top- k right singular vectors of A . The columns are chosen by randomly sampling according to a judiciously-chosen nonuniform probability distribution that depends on information in the top- k right singular subspace of A . Then, in the *deterministic stage*, a deterministic procedure is employed to select exactly k columns from the $O(k \log k)$ columns chosen in the randomized stage. The algorithm then outputs exactly k columns of A that correspond to those columns chosen from V_k^T . Theorem 1 states that the projection of A on the subspace spanned by these k columns of A is (up to bounded error) close to the best rank k approximation to A .

3.1 Detailed description of our main algorithm

In more detail, Algorithm 1 first computes a probability distribution p_1, p_2, \dots, p_n over the set $\{1, \dots, n\}$, i.e., over the columns of V_k^T , or equivalently over the columns of A . The probability distribution depends on information in the top- k right singular subspace of A . In particular, for all $i \in [n]$, define

$$p_i = \frac{\frac{1}{2} \left\| (V_k)_{(i)} \right\|_2^2}{\sum_{j=1}^n \left\| (V_k)_{(j)} \right\|_2^2} + \frac{\frac{1}{2} \left\| \left(\Sigma_{\rho-k} V_{\rho-k}^T \right)^{(i)} \right\|_2^2}{\sum_{j=1}^n \left\| \left(\Sigma_{\rho-k} V_{\rho-k}^T \right)^{(j)} \right\|_2^2}, \quad (3)$$

and note that $p_i \geq 0$, for all $i \in [n]$, and that $\sum_{i=1}^n p_i = 1$. We will describe the computation of probabilities of this form below.

In the *randomized stage*, Algorithm 1 employs the following randomized column selection algorithm to choose $O(k \log k)$ columns from V_k^T to pass to the second stage. Let $c = \Theta(k \log k)$ be a positive integer.¹ For each $i \in [n]$, independently, the algorithm keeps the i -th column of V_k^T with probability $\min\{1, cp_i\}$. Additionally, if the i -th column is kept, then a scaling factor equal to $1/\sqrt{\min\{1, cp_i\}}$ is kept as well. Thus, at the end of this process, we will be left with \tilde{c} columns of V_k^T and their corresponding scaling factors. Notice that due to random sampling, \tilde{c} will generally be different than c . However, it can be proved that if $c = \Theta(k \log k)$ then $\tilde{c} = O(k \log k)$ with constant probability.

In order to conveniently represent the \tilde{c} selected columns and the associated scaling factors, we will use the following sampling matrix formalism. First, define an $n \times \tilde{c}$ sampling matrix S_1 as follows: S_1 is initially empty; for all i , in turn, if the i -th column of V_k^T is selected by the random sampling process, then e_i (an n -vector of all-zeros, except for its i -th entry which is set to one) is appended to S_1 . Next, define the $\tilde{c} \times \tilde{c}$ diagonal rescaling matrix D_1 as follows: if the i -th column of V_k^T is selected, then a diagonal entry of D_1 is set to $1/\sqrt{\min\{1, cp_i\}}$. Thus, we may view the randomized stage as outputting the matrix $V_k^T S_1 D_1$ consisting of a small number of rescaled columns of V_k^T , or simply as outputting S_1 and D_1 .

¹Recall that the Θ -notation can be used to denote an asymptotically tight bound: $f(n) \in \Theta(g(n))$ or $f(n) = \Theta(g(n))$ if there exists positive constants c_1, c_2 , and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$. This is similar to the way in which the big- O -notation can be used to denote an asymptotic upper bound: $f(n) = O(g(n))$ if there exists positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$. Finally, the Ω -notation can be used to denote an asymptotic lower bound: $f(n) = \Omega(g(n))$ if there exists positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$.

In the *deterministic stage*, Algorithm 1 employs a deterministic column selection algorithm to the output of the first stage in order to choose *exactly* k columns from the input matrix A . To do so, we run the Algorithm 1 of [30] on the $k \times \tilde{c}$ matrix $V_k^T S_1 D_1$, i.e., the column-scaled version of the columns of V_k^T chosen in the first stage.² Thus, a matrix $V_k S_1 D_1 S_2$ is formed, or equivalently, in the sampling matrix formalism described previously, a new matrix S_2 is constructed. Its dimensions are $\tilde{c} \times k$, since it selects exactly k columns out of the \tilde{c} columns returned after the end of the randomized stage. The algorithm then returns the corresponding k columns of the original matrix A , i.e., after the second stage of the algorithm is complete, the $m \times k$ matrix $C = AS_1 S_2$ is returned as the final output.

Input: $m \times n$ matrix A , integer k .

Output: $m \times k$ matrix C with k columns of A .

1. Initial setup:

- Compute the top k right singular vectors of A , denoted by V_k .
- Compute the sampling probabilities p_i , for $i \in [n]$, using eqn. (3) or (4).
- Let $c = \Theta(k \log k)$.

2. Randomized Stage:

- For $i = 1, \dots, n$, keep the i -th index with probability $\min\{1, cp_i\}$. If the i -th index is kept, keep the scaling factor $\sqrt{\min\{1, cp_i\}}$.
- Form the sampling matrix S_1 and the rescaling matrix D_1 (see text).

3. Deterministic Stage:

- Run Algorithm 1 of Pan [30] (see also Lemma 3.5 in [30]) on the matrix $V_k^T S_1 D_1$ in order to select exactly k columns of $V_k^T S_1 D_1$, thereby forming the sampling matrix S_2 (see text).
- Return the corresponding k columns of A , i.e., return $C = AS_1 S_2$.

Algorithm 1: A two-stage algorithm for the CSSP.

3.2 Running time analysis

We now discuss the running time of our algorithm. Note that manipulating the probability distribution (3) yields:

$$p_i = \frac{\|(V_k)_{(i)}\|_2^2}{2k} + \frac{\|(A)^{(i)}\|_2^2 - \|(AV_k V_k^T)^{(i)}\|_2^2}{2 \left(\|A\|_F^2 - \|AV_k V_k^T\|_F^2 \right)}. \quad (4)$$

²Most deterministic algorithms for the CSSP operate on matrices that are $m \times n$ with $m \geq n$. In our case, in the second stage, we need to apply a deterministic column selection algorithm to a matrix with more columns than rows. Even though, to the best of our understanding, theoretical bounds for most of the algorithms reviewed in Section 2 hold even if $m < n$, for our theoretical analysis we opt to employ Algorithm 1 (and the related Lemma 3.5) of [30] which is explicitly designed to work for $m < n$. We will consider implementations of other deterministic procedures for our empirical evaluation; see Section 5.

Thus, knowledge of V_k , i.e., the $n \times k$ matrix consisting of the top- k right singular vectors of A , suffices to compute the p_i 's.³ By (4), $O(\min\{mn^2, m^2n\})$ time suffices for our theoretical analysis; in practice, of course, Lanczos/Arnoldi algorithms could be used to speed up the algorithm. Note also that in order to obtain a Frobenius norm bound of the form in Theorem 1, our theoretical analysis holds if the sampling probabilities are of the form:

$$p_i = \left\| (V_k)_{(i)} \right\|_2^2 / k. \quad (5)$$

That is, the Frobenius norm bound of Theorem 1 holds even if the second term in the sampling probabilities of (3) or (4) is omitted. Finally, the deterministic stage of our algorithm (using Algorithm 1 of [30]) takes $O(k^3 \log k)$ time, since $V_k^T S_1 D_1$ has w.h.p. $O(k \log k)$ columns.

An interesting open problem would be to identify other suitable importance sampling probability distributions that avoid the computation of a basis for the top- k right singular subspace.

3.3 Intuition underlying our main algorithm

Intuitively, we achieve improved bounds for the CSSP because we apply the deterministic algorithm to a lower dimensional matrix (the matrix $V_k^T S_1 D_1$ with $O(k \log k)$ columns, as opposed to the matrix A with n columns) in which the columns are “spread out” in a “nice” manner. To see this, note that the probability distribution of equation (5), and thus one of the two terms in the probability distribution of (3) or (4), equals (up to scaling) the diagonal elements of the projection matrix onto the span of the top- k right singular subspace. In diagnostic regression analysis, these quantities have a natural interpretation in terms of *statistical leverage*, and thus they have been used extensively to identify “outlying” data points [10]. Thus, the importance sampling probabilities that we employ in the randomized stage of our main algorithm provide a bias toward more “outlying” columns, which then provide a “nice” starting point for the deterministic stage of our main algorithm. (This also provides intuition as to why using importance sampling probabilities of the form (5) leads to relative-error low-rank matrix approximation bounds of the form (2); see [15, 16].)

4 Proof of Theorem 1

In this section, we provide a proof of Theorem 1. We start with an outline of our proof, pointing out conceptual improvements that were necessary in order to obtain improved bounds. An important condition in the first phase of the algorithm is that when we sample columns from the $k \times n$ matrix V_k^T , we obtain a $k \times \tilde{c}$ matrix $V_k^T S_1 D_1$ that does not lose any rank. To do so, we will apply a result from matrix perturbation theory to prove that if $c = \Theta(k \log k)$ then $|\sigma_k^2(V_k^T S_1 D_1) - 1| \leq 1/2$. (See Lemma 1 below.) Then, under the assumption that $V_k^T S_1 D_1$ is full rank, we will prove that the $m \times k$ matrix C returned by the algorithm will satisfy:

$$\|A - P_C A\|_\xi \leq \|A - A_k\|_\xi + \sigma_k^{-1}(V_k^T S_1 D_1 S_2) \|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\|_\xi$$

for both $\xi = 2, F$. (See Lemma 2 below.) Next, we will provide a bound on $\sigma_k^{-1}(V_k^T S_1 D_1 S_2)$. In order to get a strong accuracy guarantee for the overall algorithm, the deterministic column selection algorithm must satisfy

$$\sigma_k(V_k^T S_1 D_1 S_2) \geq \frac{\sigma_k(V_k^T S_1 D_1)}{p(k, \tilde{c})} > 0,$$

³Actually, from (4) it is clear that *any* orthogonal matrix spanning the top- k right singular subspace suffices.

where $p(k, \tilde{c})$ is a polynomial in both k and \tilde{c} . Thus, for our main theorem, we will employ Algorithm 1 of Pan [30], which guarantees the above bound with $p(k, \tilde{c}) = \sqrt{k(\tilde{c} - k) + 1}$.⁴ (See Lemma 3 below.) Finally, we will show, using relatively straightforward matrix perturbation techniques, that $\left\| \Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1 \right\|_{\xi}$ is not too much more, in a multiplicative sense, than $\|A - A_k\|_{\xi}$, where we note that the factors differ for $\xi = 2, F$. (See Lemmas 4 and 5 below.) By combining these results, the main theorem will follow.

We should note here that existing proofs for the relative error bound of eqn. (2) of Section 2 break down if $o(k \log k)$ columns of A are selected. ($\Omega(k \log k)$ columns seem necessary to guarantee that the matrix of the sampled columns preserves a certain rank constraint that does not seem easy to circumvent.) Thus, extending the theoretical computer science results to pick exactly k columns does not seem easy using existing techniques. On the other hand, it should be noted that if we allow existing numerical linear algebra algorithms to pick more than k columns, it is not clear whether relative error approximation guarantees of the form described in eqn. (2) can be obtained. However, a hybrid approach that first selects roughly $k \log k$ columns, and then nails down exactly k columns using a deterministic algorithm, seems to combine the best of both worlds and thus achieve the stated improvement.

4.1 The rank of $V_k^T S_1 D_1$

The following lemma provides a bound on the singular values of the matrix $V_k^T S_1 D_1$ computed by the *randomized phase* of Algorithm 1, from which it will follow that the matrix $V_k^T S_1 D_1$ is full rank. To prove the lemma, we apply a recent result of Rudelson and Vershynin on approximating the spectral norm of an operator [33, 16]. Note that probabilities of the form (5) actually suffice to establish Lemma 1. Note also that, by the Coupon Collecting Problem, we cannot set the column sampling parameter c to be less than $\Theta(k \log k)$ (in worst case) at this step.

Lemma 1 *Let S_1 and D_1 be constructed using Algorithm 1. Then, there exists a choice for $c = \Theta(k \log k)$ such that with probability at least 0.9,*

$$\sigma_k(V_k^T S_1 D_1) \geq 1/2.$$

In particular, $V_k^T S_1 D_1$ has full rank.

Proof: In order to bound $\sigma_k(V_k^T S_1 D_1)$, we will bound $\|V_k^T S_1 D_1 D_1 S_1^T V_k - I_k\|_2$. Towards that end, we will use Theorem 7 of [16] with $\beta = 1/2$. This theorem (followed by Markov's inequality) guarantees that given our construction of S_1 and D_1 , with probability at least 0.9,

$$\begin{aligned} \|V_k^T S_1 D_1 D_1 S_1^T V_k - I_k\|_2 &\leq O(1) \sqrt{\frac{\log c}{\beta c}} \|V_k\|_F \|V_k\|_2 \\ &\leq O(1) \sqrt{\frac{k \log c}{c}}. \end{aligned}$$

Standard matrix perturbation theory results [22] now imply that for all $i = 1, \dots, k$,

$$|\sigma_i^2(V_k^T S_1 D_1) - 1| \leq O(1) \sqrt{\frac{k \log c}{c}} \leq 1/2$$

for some $c = \Theta(k \log k)$.

⁴To be exact, in the parlance of this paper, Lemma 3.5 of [30] guarantees that $p(k, \tilde{c}) = \sqrt{\mu^2 k(\tilde{c} - k) + 1}$, for a user-controlled parameter $\mu \geq 1$. [30] suggests using $\mu = 1 + u$, where u is the machine precision. We note that by choosing a larger μ the deterministic step of our algorithm becomes (up to constant factors) faster. See [30] for details.

4.2 Bounding the spectral and Frobenius norms of $A - P_C A$

Lemma 2 *Let S_1 , D_1 , and S_2 be constructed as described in Algorithm 1 and recall that $C = AS_1 S_2$. If $V_k^T S_1 D_1$ has full rank, then for $\xi = 2, F$,*

$$\|A - P_C A\|_\xi \leq \|A - A_k\|_\xi + \sigma_k^{-1} (V_k^T S_1 D_1 S_2) \|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\|_\xi.$$

Proof: (Note that this proof differs from our original proof presented in [5].) We seek to bound the spectral and Frobenius norms of $A - P_C A$, where $C = AS_1 S_2$ is constructed by Algorithm 1. To do so, first notice that scaling the columns of a matrix (equivalently, post-multiplying the matrix by a diagonal matrix) by any non-zero scale factors does not change the subspace spanned by the columns of the matrix. Thus,

$$\begin{aligned} A - P_C A &= A - (AS_1 S_2) (AS_1 S_2)^+ A \\ &= A - (AS_1 D_1 S_2) (AS_1 D_1 S_2)^+ A \\ &= A - (AS) (AS)^+ A, \end{aligned} \tag{6}$$

where, in the last line, we have introduced the convenient notation $\mathcal{S} = S_1 D_1 S_2 \in R^{n \times k}$ that we will use throughout the remainder of this proof. In the sequel we seek to bound the residual

$$\|A - P_C A\|_\xi = \|A - (AS) (AS)^+ A\|_\xi. \tag{7}$$

First, note that

$$(AS)^+ A = \arg \min_{X \in R^{k \times n}} \|A - ASX\|_\xi.$$

This implies that in (7) we can replace $(AS)^+ A$ with any other $k \times n$ matrix and the equality with an inequality. In particular we replace $(AS)^+ A$ with $(A_k \mathcal{S})^+ A_k$, where A_k is the best rank- k approximation to A :

$$\begin{aligned} \|A - P_C A\|_\xi &= \|A - AS(AS)^+ A\|_\xi \\ &\leq \|A - AS(A_k \mathcal{S})^+ A_k\|_\xi. \end{aligned}$$

Let $A_{\rho-k} = U_{\rho-k} \Sigma_{\rho-k} V_{\rho-k}^T$. Then, $A = A_k + A_{\rho-k}$ and, using the triangle inequality,

$$\begin{aligned} \|A - P_C A\|_\xi &= \|A_k + A_{\rho-k} - (A_k + A_{\rho-k}) \mathcal{S} (A_k \mathcal{S})^+ A_k\|_\xi \\ &\leq \underbrace{\|A_k - A_k \mathcal{S} (A_k \mathcal{S})^+ A_k\|_\xi}_{\gamma_1} + \underbrace{\|A_{\rho-k}\|_\xi}_{\gamma_2} + \underbrace{\|A_{\rho-k} \mathcal{S} (A_k \mathcal{S})^+ A_k\|_\xi}_{\gamma_3}. \end{aligned} \tag{8}$$

We now bound γ_1 , γ_2 , and γ_3 . First, for γ_1 , note that:

$$\begin{aligned} \gamma_1 &= \|A_k - A_k \mathcal{S} (A_k \mathcal{S})^+ A_k\|_\xi \\ &= \|U_k \Sigma_k V_k^T - U_k \Sigma_k (V_k^T \mathcal{S}) (U_k \Sigma_k V_k^T \mathcal{S})^+ U_k \Sigma_k V_k^T\|_\xi \\ &= \|U_k \Sigma_k V_k^T - U_k \Sigma_k (V_k^T \mathcal{S}) (V_k^T \mathcal{S})^+ (U_k \Sigma_k)^+ U_k \Sigma_k V_k^T\|_\xi \end{aligned} \tag{9}$$

$$= \|\Sigma_k - \Sigma_k (V_k^T \mathcal{S}) (V_k^T \mathcal{S})^+ (U_k \Sigma_k)^+ U_k \Sigma_k\|_\xi \tag{10}$$

$$= \|\Sigma_k - \Sigma_k\|_\xi = 0. \tag{11}$$

In (9), we replaced $(U_k \Sigma_k V_k^T \mathcal{S})^+$ by $(V_k^T \mathcal{S})^+ (U_k \Sigma_k)^+$. This follows since the statement of our lemma assumes that the matrix $V_k^T S_1 D_1$ has full rank; also, the construction of S_2 guarantees

that the columns of $V_k^T S_1 D_1$ that are selected in the second stage of Algorithm 1 are linearly independent; thus, the $k \times k$ matrix $V_k^T \mathcal{S} = V_k^T S_1 D_1 S_2$ has full rank and thus is invertible. In (10), U_k and V_k^T can be dropped without increasing a unitarily invariant norm, while (11) follows since $V_k^T \mathcal{S}$ is a full-rank $k \times k$ matrix.

Next, note that $\gamma_2 = \|A_{\rho-k}\|_\xi = \|A - A_k\|_\xi$. Finally, to conclude the proof, we bound γ_3 as follows:

$$\begin{aligned} \gamma_3 &= \|A_{\rho-k} \mathcal{S} (A_k \mathcal{S})^+ A_k\|_\xi \\ &= \|U_{\rho-k} \Sigma_{\rho-k} V_{\rho-k}^T \mathcal{S} (U_k \Sigma_k V_k^T \mathcal{S})^+ U_k \Sigma_k V_k^T\|_\xi \\ &= \|\Sigma_{\rho-k} V_{\rho-k}^T \mathcal{S} (V_k^T \mathcal{S})^+\|_\xi \end{aligned} \tag{12}$$

$$\leq \|\Sigma_{\rho-k} V_{\rho-k}^T \mathcal{S}\|_\xi \left\| (V_k^T \mathcal{S})^{-1} \right\|_2 \tag{13}$$

$$= \sigma_k^{-1} (V_k^T \mathcal{S}) \|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\|_\xi. \tag{14}$$

(12) follows by the orthogonality of $U_{\rho-k}$ and V_k and the fact that $V_k^T \mathcal{S}$ is a $k \times k$ invertible matrix (see above); (13) follows from the fact that for any two matrices X and Y and $\xi = 2, F$, $\|XY\|_\xi \leq \|X\|_\xi \|Y\|_2$; and (14) follows since $\mathcal{S} = S_1 D S_2$ and S_2 is an orthogonal matrix.

4.3 Upper bounds for $\sigma_k^{-1} (V_k^T S_1 D_1 S_2)$ and $\|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\|_\xi$, $\xi = 2, F$

Lemma 3 *Let S_1 , D_1 , and S_2 be constructed using Algorithm 1. If $c = \Theta(k \log k)$, then with probability at least 0.9,*

$$\sigma_k^{-1} (V_k^T S_1 D_1 S_2) \leq 2\sqrt{k(\tilde{c} - k) + 1}.$$

Proof: From Lemma 1 we know that $\sigma_i (V_k^T S_1 D_1) \geq 1/2$ holds for all $i = 1, \dots, k$ with probability at least 0.9. The deterministic construction of S_2 (see Algorithm 1 and Lemma 3.5 in [30]) guarantees that

$$\sigma_k (V_k^T S_1 D_1 S_2) \geq \frac{\sigma_k (V_k^T S_1 D_1)}{\sqrt{k(\tilde{c} - k) + 1}} \geq \frac{1}{2\sqrt{k(\tilde{c} - k) + 1}}.$$

Lemma 4 ($\xi = 2$) *If S_1 and D_1 are constructed as described in Algorithm 1, then with probability at least 0.9,*

$$\|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\|_2 \leq \left(1 + O(1) \sqrt{\frac{(\rho - k + 1) \log c}{c}}\right)^{1/2} \|A - A_k\|_2.$$

Proof: Define $\Gamma = \Sigma_{\rho-k} V_{\rho-k}^T V_{\rho-k} \Sigma_{\rho-k}$. We manipulate $\|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\|_2^2$ as follows:

$$\begin{aligned} \|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\|_2^2 &= \|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1 D_1 S_1^T V_{\rho-k} \Sigma_{\rho-k}\|_2 \\ &= \|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1 D_1 S_1^T V_{\rho-k} \Sigma_{\rho-k} - \Gamma + \Gamma\|_2 \\ &\leq \|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1 D_1 S_1^T V_{\rho-k} \Sigma_{\rho-k} - \Gamma\|_2 + \|\Sigma_{\rho-k}^2\|_2. \end{aligned}$$

Given our construction of S_1 and D_1 , and applying Markov's inequality and Theorem 7 of [16] with $\beta = 1/2$, we get that with probability at least 0.9,

$$\begin{aligned} &\|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1 D_1 S_1^T V_{\rho-k} \Sigma_{\rho-k} - \Sigma_{\rho-k} V_{\rho-k}^T V_{\rho-k} \Sigma_{\rho-k}\|_2 \\ &\leq O(1) \sqrt{\frac{\log c}{\beta c}} \|\Sigma_{\rho-k}\|_F \|\Sigma_{\rho-k}\|_2. \end{aligned}$$

Thus, by combining these expressions, we have that

$$\|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\|_2^2 \leq O(1) \sqrt{\frac{\log c}{c}} \|\Sigma_{\rho-k}\|_F \|\Sigma_{\rho-k}\|_2 + \|\Sigma_{\rho-k}\|_2^2.$$

Using $\|\Sigma_{\rho-k}\|_2 = \|A - A_k\|_2$ and $\|\Sigma_{\rho-k}\|_F \leq \sqrt{\rho - k + 1} \|A - A_k\|_2$ concludes the proof of the lemma.

Lemma 5 ($\xi = F$) *If S_1 and D_1 are constructed as described in Algorithm 1, then with probability at least 0.9, $\|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\|_F \leq 4 \|A - A_k\|_F$.*

Proof: It is straightforward to prove that with our construction of S_1 and D_1 , the expectation of $\|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\|_F^2$ is equal to $\|\Sigma_{\rho-k} V_{\rho-k}^T\|_F^2$. In addition, note that the latter quantity is exactly equal to $\|A - A_k\|_F^2$. Applying Markov's inequality, we get that with probability at least 0.9,

$$\|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\|_F^2 \leq 10 \|A - A_k\|_F^2,$$

which implies the lemma.

4.4 Completing the proof of Theorem 1

To prove the Frobenius norm bound in Theorem 1 we use Lemmas 1, 2, 3, and 5. Notice that Lemmas 1 and 3 fail with probability at most 0.1, and that Lemma 5 fails with probability at most 0.1. Overall, by combining all these and applying the standard union bound, it follows that the Frobenius norm bound in Theorem 1 holds with probability at least 0.7. The proof of the spectral norm bound in Theorem 1 is similar, except for employing Lemma 4 instead of Lemma 5.

5 Empirical Evaluation

In this section, we describe an empirical evaluation of a Matlab implementation of Algorithm 1. Our primary goal is to confirm its usefulness in practice and to provide initial evidence for its performance with respect to existing deterministic column selection techniques. (Recall that we have evaluated our main algorithm as an unsupervised feature selection strategy in three data analysis applications [4]. In addition, a more detailed empirical evaluation elucidating many of the technical issues will be forthcoming.) We will be interested in the behavior of our algorithm under different parameters, e.g., the size of the input matrix, the number of columns to be selected, etc. We will experiment with four carefully constructed families of matrices; the main reason underlying our choice of these four classes of matrices is that they have been used in prior literature in order to evaluate the performance of various RRQR algorithms.

5.1 Experimental setup: matrices

We consider the following four classes of matrices.

LOGDIST: This is an $n \times n$ matrix A that is constructed as a product of three matrices, i.e., $A = U \Sigma V^T$. U and V are $n \times n$ random orthogonal matrices, constructed as described in [29]. Σ is an $n \times n$ diagonal matrix, whose diagonal entries are logarithmically equally spaced points between 10^0 and $10^{-\log(n)}$. In our experiments we worked with five different randomly chosen such matrices; we report results on one of those matrices, since we observed essentially no variance in the results.

SCALERANDOM: This is an $n \times n$ matrix with elements uniformly distributed in $[-1, 1]$ and normalized by dividing the j -th row of the matrix for all $j \in [n]$ by the factor $(20 \cdot 2.2 \cdot 10^{-16})^{\frac{j}{n}}$, following the lines of [23]. In our experiments we worked with five different randomly chosen such matrices; we report results on one of those matrices, since we observed essentially no variance in the results.

GKS: This is an $n \times n$ upper triangular matrix whose j -th diagonal element is equal to $1/\sqrt{j}$, and whose (i, j) -th off-diagonal element is equal to $-1/\sqrt{j}$, for $j > i$ [23].

KAHAN: This is an $n \times n$ matrix A which is constructed as product of two matrices, i.e., $A = SK$. S is a diagonal $n \times n$ matrix with diagonal elements equal to $1, \zeta, \zeta^2, \dots, \zeta^{n-1}$. K is an $n \times n$ upper triangular matrix, whose diagonal elements are equal to one, and whose off-diagonal elements (the upper triangular part) are equal to $-\phi$. Such matrices were also used in the numerical experiments of [23], with $\phi = 0.285$ and $\zeta^2 = 1 - \phi^2$. We use the same values for ϕ and ζ in our numerical experiments.

In our numerical experiments we work with three different values of $n = 100, 384,$ and 768 , following the experimental setup of [23]. For $n = 384$, Figure 1 illustrates the top singular values of typical matrices chosen from each of the above families. The corresponding linear structure is plotted in Figure 2. The linear structure of a matrix A whose rank is equal to ρ is measured via the function

$$f(k) = \sum_{i=1}^k \sigma_i^2(A) / \|A\|_F^2, \quad (15)$$

where k is a positive integer ranging from 1 up to ρ . Intuitively, $f(k)$ measures the percentage of the spectrum of the matrix that is captured by its top k singular values.

5.2 Experimental setup: algorithms

We compare our randomized column selection method to the following four deterministic column selection methods. We also provide pointers to publicly available software implementing these methods in Table 3.

1. Pivoted QR: We employed MATLAB’s `qr` function. This function implements the algorithm described by Golub in [21]. The best known bound for the spectral norm of the residual error for this algorithm is proved in [23]:

$$\|A - CC^+A\|_2 \leq \sqrt{n-k} 2^k \|A - A_k\|_2.$$

2. qrxp: `qrxp` is the algorithm implemented as the LAPACK routine `DGEQPX` in ACM Algorithm 782 [3, 2]. We will use the MatLab implementation of the Fortran routine `DGEQPX` from [19]. The best known bound for the spectral norm of the residual error for this algorithm is:

$$\|A - CC^+A\|_2 \leq 4\sqrt{(k+1)(n-k)} \|A - A_k\|_2.$$

3. High RRQR: High RRQR was devised by Chan in [6]. A MatLab implementation is available from [17]. The best known bound for the spectral norm of the residual error for this algorithm is:

$$\|A - CC^+A\|_2 \leq \sqrt{k(n-k)} 2^{n-k} \|A - A_k\|_2.$$

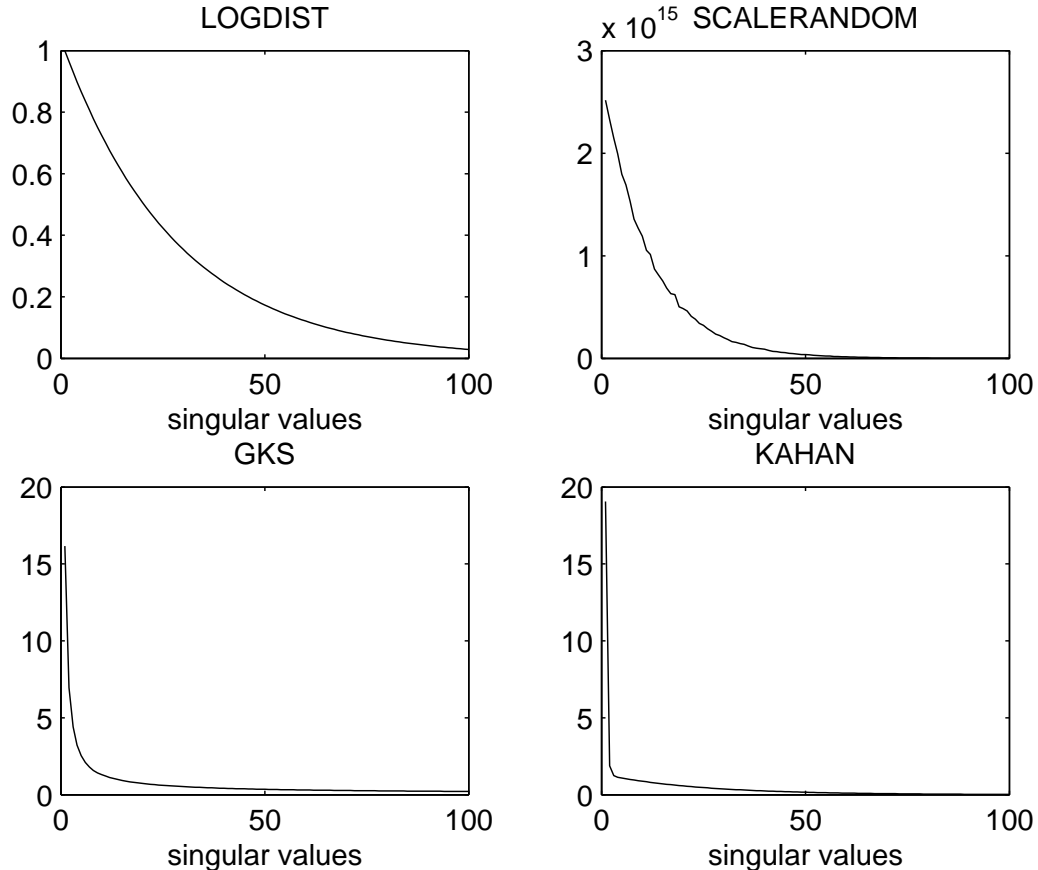


Figure 1: The top 100 singular values of the four classes of matrices of our experiments, for $n = 384$.

- 4. Low RRQR:** Low RRQR was proposed by Chan and Hansen in [8]. A MatLab implementation is available from [17]. The best known bound for the spectral norm of the residual error for this algorithm is:

$$\|A - CC^+A\|_2 \leq \sqrt{(k+1)n}2^{k+1} \|A - A_k\|_2.$$

5.3 Description of our experiments

In an *experiment*, we compare a deterministic column selection method \mathcal{D} with our two-stage randomized column selection algorithm of Section 3, where the same deterministic method \mathcal{D} is used in the second step of our two-stage algorithm. An *experiment* is described by the six-tuple $\{A, \xi, \mathcal{D}, n, k, c\}$, where A is the input matrix, $\xi = 2$ or F specifies a matrix norm, \mathcal{D} denotes one of the four deterministic column selection methods of Table 3, n is the size of A , k is the number of columns to be selected from A , and c denotes the number of columns selected in the first stage of our two-stage algorithm. We ran $4 \times 2 \times 4 \times 3 \times 3 \times 5 = 1440$ *experiments*: these numbers correspond to four families of matrices (Section 5.1), two choices of norms (spectral and Frobenius), four deterministic column selection methods (Section 5.2), three values of n , three values of k , and five values of c (Table 4). In addition, due to the randomization which comes with

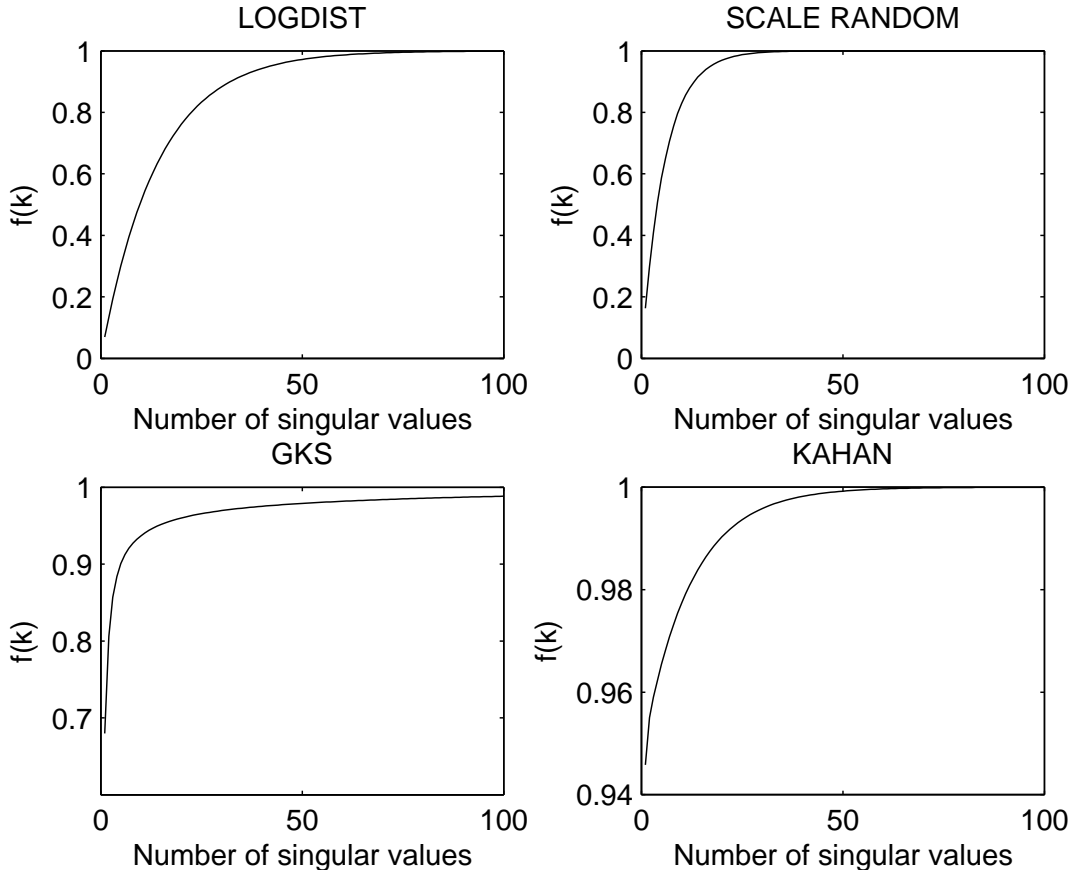


Figure 2: The linear structure of the four classes of matrices of our numerical experiments, for $n = 384$ (see eqn. (15) for a definition of $f(k)$).

a potential failure probability, in order to guarantee a success probability of at least $1 - 10^{-20}$, for every experiment, we repeated the above procedure 40 times—which was typically much more than necessary—and returned the best result as the output of the experiment. The platform used in our experiments was a 2.0 GHz Pentium IV with 1GB RAM.

5.4 Discussion of the results of our experiments

In this section we discuss the results of our numerical experiments by (i) highlighting the superiority of our two-stage algorithm against the deterministic algorithms of Table 3, (ii) pointing out the effectiveness of the randomized step of our method, and (iii) discussing effects of all six parameters $\{A, \xi, \mathcal{D}, n, k, c\}$ in the performance of our column selection method. Finally, we briefly comment on the running times of the column selection methods of our study.

Consider, for the sake of concreteness, a typical *experiment* out of the total of 1440 experiments that we ran, e.g., the experiment with description

$$\{A, \xi, \mathcal{D}, n, k, c\} = \{LOGDIST, 2, qrxp, 384, 10, 90\}.$$

Each experiment generates two results ϵ_1 and ϵ_2 . ϵ_1 is the residual error of our two-stage column selection method, while ϵ_2 is the residual error of the deterministic column selection method

Method	Reference	Software	$\ A - CC^+A\ _2 \leq$
Pivoted QR	[21]	Mathworks	$\sqrt{n - k}2^k \ A - A_k\ _2$
qrxp	[3, 2]	[19]	$4\sqrt{(k + 1)(n - k)} \ A - A_k\ _2$
High RRQR	[6]	[17]	$\sqrt{k(n - k)}2^{n-k} \ A - A_k\ _2$
Low RRQR	[8]	[17]	$\sqrt{(k + 1)n}2^{k+1} \ A - A_k\ _2$

Table 3: The deterministic column selection algorithms.

n	k	c
100	10, 20, 30	40, 50, 70, 90, 100
384	10, 30, 50	70, 100, 200, 300, 400
768	20, 60, 100	120, 200, 300, 500, 700

Table 4: Experimental parameters

applied to the matrix A . In our example here, ϵ_1 would correspond to the residual error of our two-stage column selection algorithm, with qrxp employed in the second stage, whereas ϵ_2 would correspond to the error of qrxp applied directly to the original matrix.

Remark: All residual errors are normalized by dividing them by $\|A - A_k\|_\xi$, which implies that ϵ_1 and ϵ_2 are at least one. Clearly, a residual error of exactly one would imply that the residual error of the CSSP is exactly equal to the residual error of the best rank- k approximation. This, for example, would be the case for orthogonal matrices.

5.4.1 The 2-step algorithm in practice

Our numerical experiments confirm the improved performance of our two-stage algorithm when compared to the four deterministic column selection methods of Table 3. The improved theoretical approximation bounds of our method nicely coincide with an improved empirical performance for both the spectral and the Frobenius norm. In particular, for fixed $\{A, \xi, n, k\}$ we are interested to see whether the best instance of our method (there are $4 \times 5 = 20$ instances due to the four different deterministic methods that are used in the second step, and the five different choices for the parameter c) gives a lower residual error than the residual error of the best deterministic method. (Recall, there are four deterministic methods.) Overall, there are $4 \times 2 \times 3 \times 3 = 72$ such “games” (four matrices A , two values of ξ , three values of n , and three values of k). Our method is the “winner” in 55 of those, with improvements that range from 10% to 100%. (In order to declare our algorithm the “winner” of a “game,” we required that the best instance of our method is better than the best instance of the four deterministic methods.) See Tables 5 and 6 for a summary of the results of all 1440 *experiments* and observe that our two-stage algorithm gives better results than the deterministic methods in almost all cases. A notable exception is the spectral norm case for KAHAN matrices, where our method performs worse than its deterministic counterparts. (This is probably since the KAHAN matrices are so low rank and our rank parameter was set to be 10 or more.) It is worth noting that in all 17 cases where our two-stage algorithm is unable to give improved empirical performance, (i) its performance is not much worse than that of the best deterministic method (at most 20% worse), and (ii) the best deterministic column selection method was the Low RRQR algorithm.

n	LOGDIST	SCALERANDOM	GKS	KAHAN
	k=10, 20, 30	k=10, 20, 30	k=10, 20, 30	k=10, 20, 30
100	1, 1, 1	1, 1, 1	1, 1, 0	0, 0, 0
384	1, 1, 1	1, 1, 1	1, 0, 0	0, 0, 0
768	1, 1, 1	1, 1, 1	1, 0, 0	0, 0, 0

Table 5: Results of the “games” of our experiments for the spectral norm. A “1” means that our method was the “winner” of a “game”.

n	LOGDIST	SCALERANDOM	GKS	KAHAN
	k=10, 20, 30	k=10, 20, 30	k=10, 20, 30	k=10, 20, 30
100	1, 1, 1	1, 1, 1	1, 1, 1	1, 1, 1
384	1, 1, 1	1, 1, 1	1, 1, 0	1, 1, 1
768	1, 1, 1	1, 1, 1	0, 1, 0	1, 1, 1

Table 6: Results of the “games” of our experiments for the Frobenius norm. A “1” means that our method was the “winner” of a “game”.

5.4.2 The randomized step of our two-stage algorithm

In this section, we point out an interesting observation regarding the first stage of our algorithm. The randomized step of our algorithm, which can be viewed as a preprocessing step before the application of a deterministic column selection method, is capable of making competitive (in practice) even the less powerful deterministic column selection methods. For example, High RRQR, which is by far the worst deterministic column selection method (both in theory and in practice), can be very competitive when it is combined with the randomized step of our algorithm. More specifically, its performance is comparable to that of the best deterministic method, which typically is the Low RRQR method. In Figure 3 we plot the results of an application of all column selection algorithms on the KAHAN matrix with $k = 20$, $n = 100$, and $\xi = 2$. One can observe the significant improvements that are achieved in Pivoted QR, qrxp, and High RRQR, when the randomized step of our algorithm is applied before the deterministic step. Note also that the common observation that the Pivoted QR method can fail in the KAHAN matrix [23] is confirmed here, since the Pivoted QR method gives a residual error of about 7.5. However, this error goes down to 1.7 when the randomized step of our method is combined with the Pivoted QR method.

5.4.3 Performance of our two-stage algorithm under different experimental conditions

We briefly comment on the effect of each of the six parameters of an *experiment* on the theoretical and empirical performance of our two-stage algorithm.

The effect of the input matrix A : The input matrix A , theoretically, does not affect the performance of our method because our analysis is not based on any assumptions on A . Empirically, we didn’t observe any significant effect of A on the performance of our algorithm.

The effect of the norm ξ : Theoretically, the Frobenius norm approximation bound of our algorithm, which is independent of the size of A , is more accurate than the spectral norm bound, which depends on the fourth root of the size of A . This fact is observed in our numerical experiments in all cases. For example, in Figure 4 we fix $\{A, \mathcal{D}, k, n\} =$

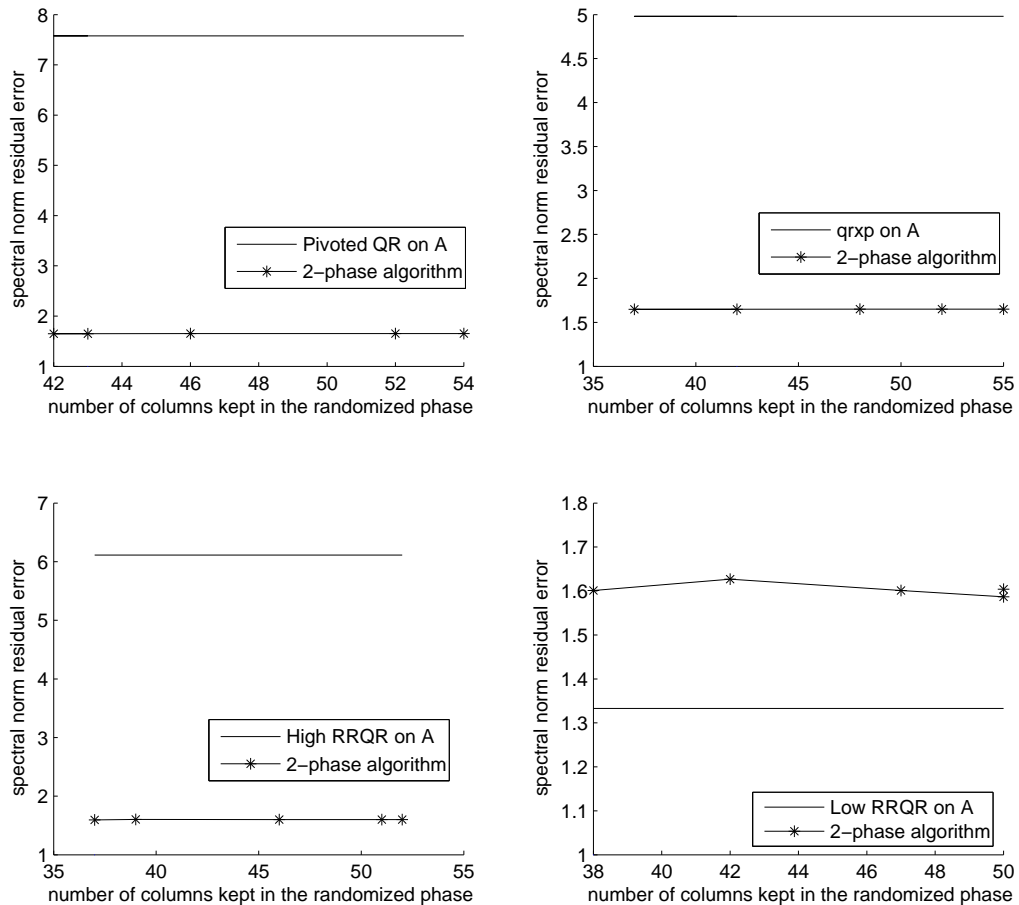


Figure 3: The results of the *experiments* $\{KAHAN, 2, \mathcal{D}, 20, 100, c\}$.

$\{LOGDIST, Pivoted - QR, 60, 768\}$ and observe the performance of our method for the spectral and the Frobenius norms, for all five values of c . Figure 4 confirms that the Frobenius norm approximations are more accurate (≈ 1.4) than spectral norm approximations (≈ 1.6).

The effect of the deterministic column selection method \mathcal{D} : Theoretically, the accuracy of the deterministic step of our method is crucial in order to obtain the improved approximation bounds for the CSSP. More specifically, we are seeking a deterministic column selection method with the theoretical properties of the method of Pan [30] or Gu and Eisenstat [23]. However, in practice, all four deterministic methods give comparable results when they are employed in the deterministic stage of our algorithm. For example, in Figure 5 we fixed $\{A, \xi, k, n\} = \{SCALERANDOM, F, 30, 384\}$ and observe that even the worst deterministic method (High RRQR) gives results that are close to the best ones, when combined with the randomized step of our two-stage approach.

The effect of the size of the matrix n and the number of columns to be selected k : We discuss these two parameters together because what is really important is the *relative* size

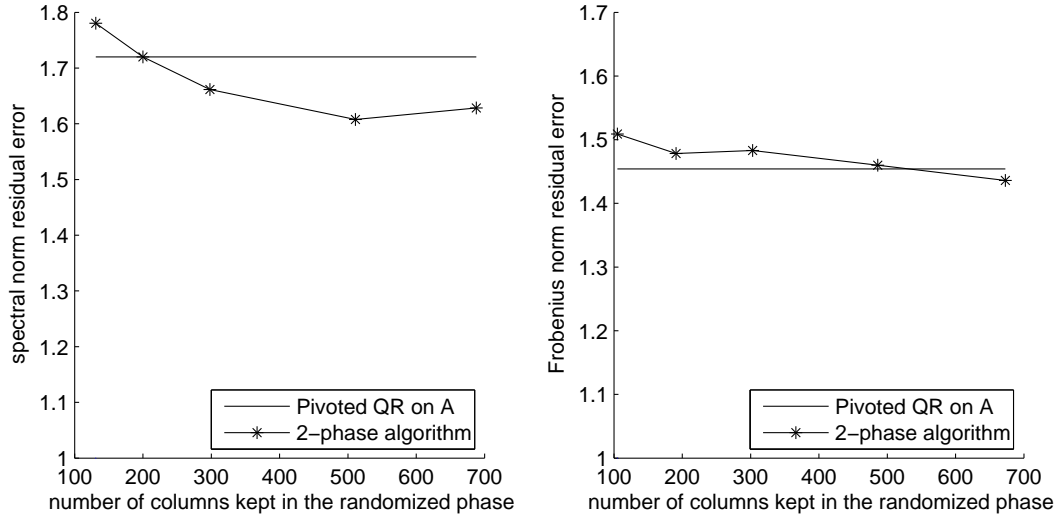


Figure 4: The results of the *experiments* $\{LOGDIST, \xi, Pivoted - QR, 60, 768, c\}$. The plot at the left corresponds to the spectral norm, while the one at the right to the Frobenius norm.

of n with respect to k . Theoretically, in order to obtain our improved approximation bounds we assumed that $k \ll n$, as is typically the case in applications. Note that when k is close to n , our bounds are not better than previous state-of-the-art bounds. This fact is observed in our numerical experiments where the performance of our algorithm is better when k is *much smaller* than n . To illustrate this fact, we compare results for $\{A, \xi, \mathcal{D}, k, n\} = \{GKS, 2, qrxp, 20, 100\}$ and $\{A, \xi, \mathcal{D}, k, n\} = \{GKS, 2, qrxp, 20, 768\}$ in Figure 6; observe that in the first case our method is approximately 3 times better than the deterministic method while in the second case is approximately 6 times better.

The effect of the number of columns c selected in the first stage of our algorithm: This is a crucial parameter of our two-stage method. Theoretically, c should be $O(k \log k)$; however, in practice, a small constant times k suffices. We further noticed another interesting phenomenon. Starting from $c = k$, while increasing c , the accuracy of our algorithm improves until an optimum point is reached. Then, if we further increase c to approximate n , the performance of our algorithm drops and eventually becomes equal to the performance of the deterministic method directly applied to the matrix V_k^T . Thus, there exists an optimal choice for the parameter c in the randomized step of our algorithm, which, theoretically is $O(k \log k)$. This manifests itself in the plots of Figure 7, where we evaluated ten different values of c in order to obtain more accurate results. Finally, it should be noted that sometimes the performance of our method seems independent of c (see Figure 7).

5.4.4 Empirical running time analysis

Our empirical study focused on the residual errors incurred by the application of the various column selection methods on various matrices. In terms of running time comparisons, our method is three to five times slower than the deterministic column selection algorithms, mainly due to the 40 repetitions of the randomized and the deterministic steps. Finally, note that except for the Pivoted QR method, the other three deterministic column selection algorithms also require the computation of the singular values and vectors of A .

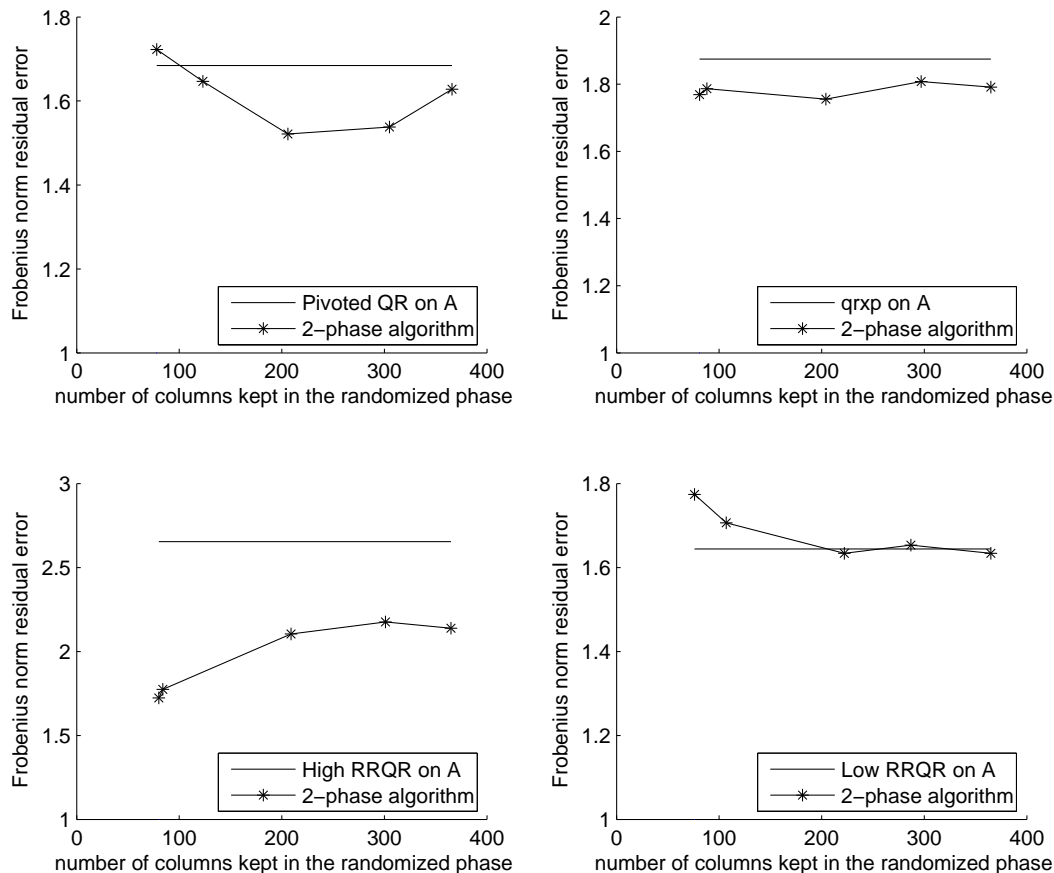


Figure 5: The results of the *experiments* $\{SCALERANDOM, F, D, 30, 384, c\}$.

Acknowledgements

We are grateful to Daniel Spielman for useful discussions as well as his encouragement and enthusiasm for the results of this paper.

References

- [1] A. Ben-Hur and I. Guyon. Detecting stable clusters using principal component analysis. *Methods in Molecular Biology*, 224:159–182, 2003.
- [2] C. H. Bischof and G. Quintana-Ortí. Algorithm 782: Codes for rank-revealing QR factorizations of dense matrices. *ACM Transactions on Mathematical Software*, 24(2):254–257, 1998.
- [3] C. H. Bischof and G. Quintana-Ortí. Computing rank-revealing QR factorizations of dense matrices. *ACM Transactions on Mathematical Software*, 24(2):226–253, 1998.

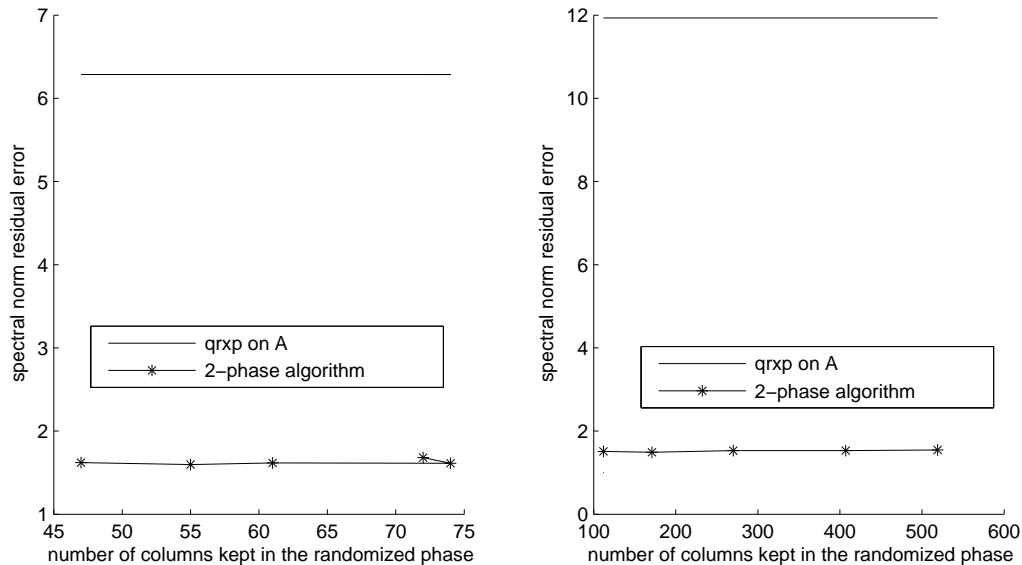


Figure 6: The results of the experiments $\{GKS, 2, qrxp, 20, 100, c\}$ (left) and $\{GKS, 2, qrxp, 20, 768, c\}$ (right).

- [4] C. Boutsidis, M.W. Mahoney, and P. Drineas. Unsupervised feature selection for principal components analysis. In *Proceedings of the 14th Annual ACM SIGKDD Conference*, pages 61–69, 2008.
- [5] C. Boutsidis, M.W. Mahoney, and P. Drineas. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 968–977, 2009.
- [6] T.F. Chan. Rank revealing QR factorizations. *Linear Algebra and Its Applications*, 88/89:67–82, 1987.
- [7] T.F. Chan and P.C. Hansen. Some applications of the rank revealing QR factorization. *SIAM Journal on Scientific and Statistical Computing*, 13:727–741, 1992.
- [8] T.F. Chan and P.C. Hansen. Low-rank revealing QR factorizations. *Numerical Linear Algebra with Applications*, 1:33–44, 1994.
- [9] S. Chandrasekaran and I. C. F. Ipsen. On rank-revealing factorizations. *SIAM Journal on Matrix Analysis and Applications*, 15:592–622, 1994.
- [10] S. Chatterjee and A.S. Hadi. *Sensitivity Analysis in Linear Regression*. John Wiley & Sons, New York, 1988.
- [11] A. Civril and M. Magdon-Ismail. Finding maximum volume sub-matrices of a matrix. Technical Report 07-08, Rensselaer Polytechnic Institute Department of Computer Science, Troy, NY, 2007.
- [12] A. Deshpande, L. Rademacher, S. Vempala, and G. Wang. Matrix approximation and projective clustering via volume sampling. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1117–1126, 2006.

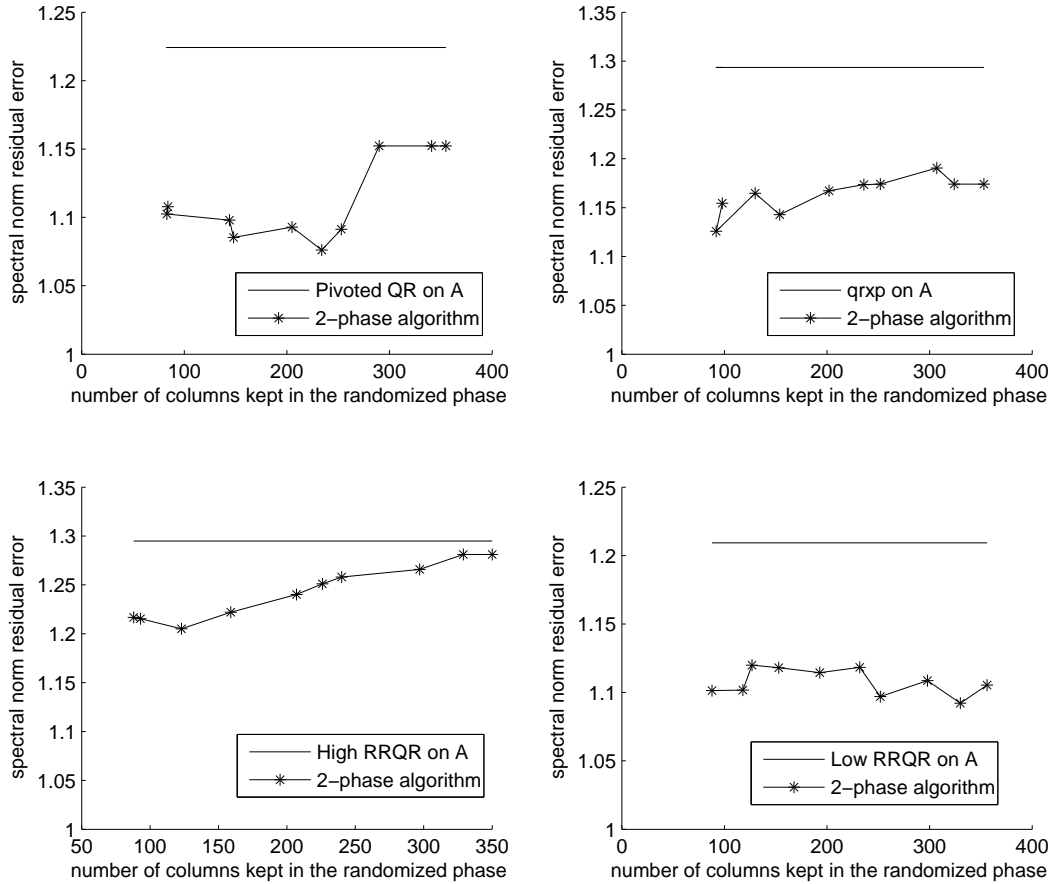


Figure 7: The results of the *experiments* $\{LOGDIST, 2, d, 10, 384, c\}$.

- [13] A. Deshpande and S. Vempala. Adaptive sampling and fast low-rank matrix approximation. In *Proceedings of the 10th International Workshop on Randomization and Computation*, pages 292–303, 2006.
- [14] P. Drineas, I. Kerenidis, and P. Raghavan. Competitive recommendation systems. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 82–90, 2002.
- [15] P. Drineas, M.W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-based methods. In *Proceedings of the 10th International Workshop on Randomization and Computation*, pages 316–326, 2006.
- [16] P. Drineas, M.W. Mahoney, and S. Muthukrishnan. Relative-error CUR matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30:844–881, 2008.
- [17] R. D. Fierro, P. C. Hansen, and P. S. K. Hansen. UTV tools: Matlab templates for rank-revealing UTV decompositions. *Numerical Algorithms*, 20(2-3):165–194, 1999.
- [18] L. V. Foster. Rank and null space calculations using matrix decomposition without column interchanges. *Linear Algebra and Its Applications*, 74:47–71, 1986.

- [19] L. V. Foster and X. Liu. Comparison of rank revealing algorithms applied to matrices with well defined numerical ranks. *Manuscript. 2006*.
- [20] A. Frieze, R. Kannan, and S. Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 370–378, 1998.
- [21] G. Golub. Numerical methods for solving linear least squares problems. *Numerische Mathematik*, 7:206–216, 1965.
- [22] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 1989.
- [23] M. Gu and S.C. Eisenstat. Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM Journal on Scientific Computing*, 17:848–869, 1996.
- [24] Y. P. Hong and C. T. Pan. Rank-revealing QR factorizations and the singular value decomposition. *Mathematics of Computation*, 58:213–232, 1992.
- [25] W. J. Krzanowski. Selection of variables to preserve multivariate data structure, using principal components. *Applied Statistics*, 36(1):22–33, 1987.
- [26] M.W. Mahoney, M. Maggioni, and P. Drineas. Tensor-CUR decompositions for tensor-based data. In *Proceedings of the 12th Annual ACM SIGKDD Conference*, pages 327–336, 2006.
- [27] M.W. Mahoney, M. Maggioni, and P. Drineas. Tensor-CUR decompositions for tensor-based data. *SIAM Journal on Matrix Analysis and Applications*, 30:957–987, 2008.
- [28] K. Z. Mao. Identifying critical variables of principal components for unsupervised feature selection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35(2):339–344, 2005.
- [29] P.-G. Martinsson, V. Rokhlin, and M. Tygert. A randomized algorithm for the approximation of matrices. Technical Report YALEU/DCS/TR-1361, Yale University Department of Computer Science, New Haven, CT, June 2006.
- [30] C.-T. Pan. On the existence and computation of rank-revealing LU factorizations. *Linear Algebra and Its Applications*, 316:199–222, 2000.
- [31] C. T. Pan and P. T. P. Tang. Bounds on singular values revealed by QR factorizations. *BIT Numerical Mathematics*, 39:740–756, 1999.
- [32] P. Paschou, E. Ziv, E.G. Burchard, S. Choudhry, W. Rodriguez-Cintron, M.W. Mahoney, and P. Drineas. PCA-correlated SNPs for structure identification in worldwide human populations. *PLoS Genetics*, 3:1672–1686, 2007.
- [33] M. Rudelson and R. Vershynin. Sampling from large matrices: an approach through geometric functional analysis. *Journal of the ACM*, 54(4):Article 21, 2007.
- [34] G.W. Stewart. Four algorithms for the efficient computation of truncated QR approximations to a sparse matrix. *Numerische Mathematik*, 83:313–323, 1999.
- [35] H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar. Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research*, 3:1399–1414, 2003.

- [36] J. Sun, Y. Xie, H. Zhang, and C. Faloutsos. Less is more: Compact matrix decomposition for large sparse graphs. In *Proceedings of the 7th SIAM International Conference on Data Mining*, 2007.
- [37] L. Wolf and A. Shashua. Feature selection for unsupervised and supervised inference: The emergence of sparsity in a weight-based approach. *Journal of Machine Learning Research*, 6:1855–1887, 2005.
- [38] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert. A fast randomized algorithm for the approximation of matrices. Technical Report YALEU/DCS/TR-1380, Yale University Department of Computer Science, New Haven, CT, 2007.
- [39] Z. Zhao and H. Liu. Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the 24th International Conference on Machine Learning*, pages 1151–1157, 2007.