Efficient Spectrum-Revealing CUR Matrix Decomposition

Cheng Chen
Shanghai Jiao Tong University

Ming Gu
University of California, Berkeley

Zhihua Zhang
Peking University

Weinan Zhang
Shanghai Jiao Tong University

Yong Yu
Shanghai Jiao Tong University

Abstract

The CUR matrix decomposition is an important tool for low-rank matrix approximation. It approximates a data matrix though selecting a small number of columns and rows of the matrix. Those CUR algorithms with gap-dependent approximation bounds can obtain high approximation quality for matrices with good singular value spectrum decay, but they have impractically high time complexities. In this paper, we propose a novel CUR algorithm based on truncated LU factorization with an efficient variant of complete pivoting. Our algorithm has gap-dependent approximation bounds on both spectral and Frobenius norms while maintaining high efficiency. Numerical experiments demonstrate the effectiveness of our algorithm and verify our theoretical guarantees.

1 Introduction

CUR matrix decomposition is a well known method for low-rank approximation (Mahoney and Drineas, 2009; Drineas et al., 2008). It approximates the data matrix $\mathbf{A}$ as the product of three matrices: $\mathbf{A} \approx \mathbf{CUR}$, where $\mathbf{C}$ and $\mathbf{R}$ are composed of sampled columns and sampled rows of matrix $\mathbf{A}$ respectively. Compared with other low-rank approximation methods such as truncated Singular Value Decomposition (SVD), CUR decomposition can better preserve the sparsity of the input matrix and facilitate the interpretation of the computed results. Because of these advantages, CUR decomposition is more attractive than SVD in text mining, collaborative filtering, bioinformatics, image and video processing (Drineas et al., 2008; Xu et al., 2015; Wang and Zhang, 2012, 2013; Mahoney et al., 2008; Mackey et al., 2011).

In recent years, many variants of the CUR decomposition have been developed (Drineas et al., 2008; Wang and Zhang, 2012, 2013; Boutsidis and Woodruff, 2017; Bien et al., 2010; Anderson et al., 2015; Drineas et al., 2006; Wang et al., 2016; Song et al., 2019). Many randomized CUR algorithms adopt random sampling on columns and rows of data matrices. They aim to achieve $1+\varepsilon$ relative error bounds on Frobenius norm with a failure probability δ (Drineas et al., 2008; Wang and Zhang, 2013; Boutsidis and Woodruff, 2017), i.e., $\|\mathbf{A}-\mathbf{CUR}\|_F^2 \leq (1+\varepsilon)\|\mathbf{A}-\mathbf{A}_k\|_F^2$, where k is the target rank of approximation. These bounds are tight theoretically, but are less useful in practical applications because they require $O(k/\varepsilon)$ columns and rows. For example, Algorithm 3 of (Boutsidis and Woodruff, 2017) requires to select $4k+4820k/\varepsilon$ columns and rows, but only gets a success probability of 0.16. If we choose $k = 10$ and $\varepsilon = 0.5$ (very common setting), the number of selected columns and rows is close to 10^6. For real-world data sets, it is impractical to select so many columns and rows. Anderson et al. (2015) proposed a deterministic column selection method for CUR decomposition with gap-dependent approximation bounds. These bounds allow their method only to choose $\ell = k+O(1)$ columns and rows when the given matrix has a rapidly decaying spectrum. However, their algorithm is highly time-consuming and requires $O(\ell mn(m+n))$ time, where ℓ is the number of selected columns and rows.

A natural question now arises: can we develop an efficient CUR algorithm which is as efficient as previous randomized CUR algorithms while maintaining gap-dependent approximation bounds? In this paper, we address this question via presenting a novel CUR algorithm, namely Spectrum-Revealing CUR (SRCUR) decomposition. Our method borrows the idea from a

Proceedings of the 23[rd]International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

classic numerical linear algebra method, i.e., pivoted LU factorization (Trefethen and Bau III, 1997). Unlike existing CUR algorithms which sample columns and rows separately (Drineas et al., 2008; Wang and Zhang, 2013; Boutsidis and Woodruff, 2017; Anderson et al., 2015), the pivoting procedure of LU factorization selects columns and rows simultaneously and considers the inter-connectivity of selected columns and rows. However, the vanilla pivoting procedure is costly and does not bound the relative error. To make use of the advantages and bypass the disadvantages of pivoted LU, we propose Fast Spectrum-Revealing LU (FSRLU) factorization, which is efficient and has theoretical guarantees. Our SRCUR algorithm first adopts FSRLU to select columns and rows and then computes the intersection matrix. The main contributions of this paper are summarized as follows:

- We propose FSRLU algorithm which can efficiently compute truncated LU factorization with complete pivoting. We also analyze the stability and time complexity of FSRLU.
- We propose SRCUR algorithm based on FSRLU. We provide error analysis for SRCUR algorithm in both spectral and Frobenius norms. The proofs show that our method has gap dependency error bounds with regard to the quadratic of the spectral gap $\sigma_{k+1}/\sigma_{p+1}$. These error bounds indicate that our method only needs to select $\ell = k+O(1)$ columns and rows for data matrices with rapidly decaying singular values.
- SRCUR is much faster than existing gap-dependent CUR algorithms. The time complexity of our method is $O(\text{nnz}(\mathbf{A})\log n)+\tilde{O}(\ell^2(m+n))$, which is comparable to the state-of-the-art randomized CUR algorithms.
- We validate our method with numerical experiments on four real-world datasets, which demonstrate that our algorithms are substantially faster than existing CUR algorithms while maintaining good performance.

The remainder of the paper is organized as follows. In Section 2, we describe notation and preliminaries used in this paper. In Section 3 we introduce related work. Then we present our algorithms in Section 4 and show theoretical analysis in Section 5. Experimental results are given in Section 6. Conclusions and discussions are provided in Section 7.

2 Notation and Preliminaries

We use $\mathbf{I}_m$ to denote $m \times m$ identity matrix, and $\mathbf{0}$ to denote a zero vector or matrix of appropriate size. For a vector $\mathbf{x} = (x_1, \ldots, x_n)^\top$, let $\|\mathbf{x}\|_2 = \sqrt{\sum_i x_i^2}$ be the ℓ_2-norm of $\mathbf{x}$. For a matrix $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{m\times n}$, let $\|\mathbf{A}\|_2$ denote the spectral norm and $\|\mathbf{A}\|_F$ denote the Frobenius norm. We use $\text{nnz}(\mathbf{A})$ to denote the number of nonzero elements of $\mathbf{A}$. Let $\det(\mathbf{A})$ be the determinant of $\mathbf{A}$ and $\text{adj}(\mathbf{A})$ be the adjugate matrix of $\mathbf{A}$. We use $\tilde{O}$ to hide logarithmic factors.

Let ρ be the rank of matrix $\mathbf{A}$. The reduced SVD of $\mathbf{A}$ is defined as

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \sum_{i=1}^{\rho} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top,$$

where σ_i are positive singular values in the descending order. That is, $\sigma_i(\mathbf{A})$ is the i-th largest singular value of $\mathbf{A}$. Let $\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$ denote the best rank-k approximation of $\mathbf{A}$ and $\mathbf{A}^\dagger = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^\top$ denote the *Moore-Penrose* pseudo-inverse of $\mathbf{A}$. $\kappa(\mathbf{A}) = \sigma_{\max}/\sigma_{\min}$ to denote the condition number of matrix $\mathbf{A}$. Here $\sigma_{\max}$ is the maximum singular value and $\sigma_{\min}$ is the minimum non-zero singular value of $\mathbf{A}$.of $\mathbf{A}$ when $\mathbf{A}$ is square.

We use the Matlab colon notation for a block matrix. Let $\mathbf{A}_{i,:}$ be the i-th row of $\mathbf{A}$ and $\mathbf{A}_{:,j}$ be the j-th column of $\mathbf{A}$. We use $\mathbf{A}_{i:j,:}$ to denote $\left[\mathbf{A}_{i,:}^\top, \mathbf{A}_{i+1,:}^\top, \ldots, \mathbf{A}_{j,:}^\top\right]^\top$ and $\mathbf{A}_{:,p:q}$ to denote $[\mathbf{A}_{:,p}, \mathbf{A}_{:,p+1}, \ldots, \mathbf{A}_{:,q}]$. Let $\mathbf{A}_{i:j,p:q}$ denote the block matrix $\begin{pmatrix} a_{ip} & \ldots & a_{iq} \\ \vdots & \ddots & \vdots \\ a_{jp} & \ldots & a_{jq} \end{pmatrix}$.

Johnson-Lindenstrauss (JL) transform (Johnson and Lindenstrauss, 1984) is a powerful tool for dimensionality reduction. It has been proved to preserve the vector norm within an ε-error ball. We present the Johnson-Lindenstrauss Lemma as follows:

Lemma 1. *For any vector $\mathbf{x} \in \mathbb{R}^d$, $0 < \varepsilon, \delta < 1/2$, there exists a JL transform matrix $\mathbf{S} \in \mathbb{R}^{p\times d}$, with $p = \Theta(\log(1/\delta)\varepsilon^{-2})$, for which satisfies*

$$(1-\varepsilon)\|\mathbf{x}\|_2^2 \le \|\mathbf{S}\mathbf{x}\|_2^2 \le (1+\varepsilon)\|\mathbf{x}\|_2^2,$$

with probability $1-\delta$.

3 Related Work

In Section 3.1, we review some representative methods of CUR matrix decomposition. In Section 3.2, we introduce truncated LU factorization with complete pivoting.

3.1 CUR Matrix Decomposition

CUR decomposition approximates the data matrix with actual columns and rows. Given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$,

CUR decomposition selects a subset of columns $\mathbf{C} \in \mathbb{R}^{m\times c}$ and a subset of rows $\mathbf{R} \in \mathbb{R}^{r\times n}$ and computes an intersection matrix $\mathbf{U} = \mathbf{C}^\dagger\mathbf{A}\mathbf{R}^\dagger$. Then, $\widehat{\mathbf{A}} = \mathbf{CUR}$ is a low-rank approximation of matrix $\mathbf{A}$. Some works try to compute an approximation $\widehat{\mathbf{A}}_k$ with exact rank k (Boutsidis and Woodruff, 2017; Anderson et al., 2015). The rank-constrained approximation is more comparable to the truncated SVD decomposition.

The essential of CUR decomposition is how to select columns and rows effectively and efficiently. The traditional approaches select columns according to deterministic pivoting rules (Stewart, 1999; Chan, 1987; Gu and Eisenstat, 1996; Berry et al., 2005), but these methods do not have good approximation errors. Recently, randomized algorithms for CUR decomposition aims to obtain $1+\varepsilon$ relative error bounds in Frobenius norm (Mahoney and Drineas, 2009; Drineas et al., 2008; Wang and Zhang, 2013; Boutsidis and Woodruff, 2017; Mahoney et al., 2011):

$$\|\mathbf{A} - \mathbf{CUR}\|_F^2 \le (1+\varepsilon)\|\mathbf{A} - \mathbf{A}_k\|_F^2.$$

Drineas et al. (2008) randomly sampled columns and rows according to their leverage scores. They achieve $1+\varepsilon$ relative error bounds in Frobenius norm. Wang and Zhang (2013) proposed an adaptive CUR algorithm with better theoretical results, and their results were further improved by (Boutsidis and Woodruff, 2017; Song et al., 2019). However, these methods have to choose $O(k/\varepsilon)$ rows and columns to achieve the $1+\varepsilon$ error bound, which is usually impractical for real-world data set.

Anderson et al. (2015) proposed unweighted column selection for CUR decomposition which provides spectral gap error bounds as the following form:

$$\|\mathbf{A} - \mathbf{CUR}\|_\xi^2 \le (1+O(\omega^2))\|\mathbf{A} - \mathbf{CUR}\|_\xi^2,$$

for $\xi \in \{2, F\}$. Here ω is a number that relies on the rate of spectrum decay of $\mathbf{A}$ and the number of oversampling $l-k$. Their method only needs to choose $\ell = k+O(1)$ columns and rows to obtain a good rank-k approximation when the data matrix has rapid spectrum decay. However, their algorithm is very expensive and requires $O(\ell mn(m+n))$ time to obtain a rank-k approximation.

There have been several investigations of ways to construct the approximation. Wang et al. (2016) proposed an efficient method to compute $\mathbf{U}$ by matrix sketching. Anderson et al. (2015) proposed a numerically stable algorithm to construct $\widehat{\mathbf{A}}$ and $\widehat{\mathbf{A}}_k$. This method avoids computing the pseudo-inverse of matrices by performing QR factorization on $\mathbf{C}$ and $\mathbf{R}$. We present their algorithm in Appendix A.

Some other variants of CUR algorithms including tensor CUR decomposition (Song et al., 2019) and CUR with ℓ_1-norm (Song et al., 2017) are less relevant to our work.

3.2 LU factorization with Complete Pivoting

LU factorization factors a matrix as a product of a unit lower triangular matrix and an upper triangular matrix with row permutation and/or column permutation. Partial pivoting is widely used in numerical algebra area because it is more efficient. However, it may fail on rank deficient matrices. The complete pivoting, even though very expensive, is necessary for those matrices to guarantee that factorized matrices are well-defined. Recently, Melgaard and Gu (2015) proposed a randomized LU factorization with complete pivoting by leverage random projection.

Rank-revealing LU algorithms (Pan, 2000; Miranian and Gu, 2003) are good quality approximations. They guarantee that the approximation capture the rank of the data matrix within a low polynomial factor in k, m and n, which could be very large. Spectrum-revealing LU (Anderson and Gu, 2017) performs extra swaps to ensure spectral gap error bounds, but it also introduces extra computational costs.

4 Our Approaches

In this section, we propose our main algorithms. We first propose the EELM algorithm which quickly estimates the element with largest magnitude of a given matrix. Then we propose the FSRLU algorithm to efficiently computes the spectrum-revealing LU factorization. Finally, we propose our SRCUR algorithm based on the FSRLU.

4.1 Estimating the Element with Largest Magnitude

Finding the element with largest magnitude in a matrix has many applications (Higham and Relton, 2016). Usually computing $\|\mathbf{A}\|_{\max} \in \mathbb{R}^{m\times n}$ from its definition has $O(mn)$ cost. However, if we can only access matrix $\mathbf{A}$ implicitly, such as through a product $\mathbf{A} = \mathbf{BC}$ or an inverse $\mathbf{A} = \mathbf{B}^{-1}$, the computational cost can be prohibitive if we need to calculate all elements of matrix $\mathbf{A}$. The method in (Higham and Relton, 2016) uses matrix-vector product to estimate the position of the element with the largest magnitude. But their method does not have theoretical guarantees on the estimated value.

We propose an efficient algorithm to estimate the element with the largest magnitude, called EELM. Given matrix $\mathbf{A}$ and projected matrix $\mathbf{R} = \mathbf{\Omega A}$ ($\mathbf{\Omega} \in \mathbb{R}^{p\times m}$ is a JL transform matrix), EELM first finds the col-

Algorithm 1 Estimating the Element with Largest Magnitude (EELM)

Input: Input matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{R} \in \mathbb{R}^{p\times n}$.
Output: row index r, column index c, estimated value x.

1: $c = \arg\max_{i\le j\le n} \|\mathbf{R}_{:,j}\|_2$.
2: Compute the c-th column of $\mathbf{A}$.
3: $r = \arg\max_{i\le j\le m} |\mathbf{A}_{j,c}|$.
4: $x = |\mathbf{A}_{r,c}|$.

umn with the maximum ℓ_2-norm in the sketched matrix $\mathbf{R}$ and then chooses the largest element in the corresponding column of the original matrix $\mathbf{A}$. We describe EELM in Algorithm 1. Since JL transform preserves the vector norm well, the estimated value can be bounded as the following theorem:

Theorem 1. *The estimated value x given by Algorithm 1 satisfies $x \ge \sqrt{\frac{1-\varepsilon}{mn(1+\varepsilon)}}\|\mathbf{A}\|_F$ with probability $1-\delta$.*

As shown in the next section, ε only affect the time complexity of our algorithm but have little influence on the error bounds. Thus, in the rest of our paper, we assume that $\varepsilon = 0.5$ and $\delta = 0.01$ for convenience.

4.2 Fast Spectrum-Revealing LU Factorization

In this section, we describe our FSRLU algorithm which consists of two parts: Fast Pivoting for Truncated LU factorization (FPTLU) and Fast Spectrum-Revealing Pivoting (FSRP). FPTLU computes a rough pivoting for truncated LU, and FSRP performs extra operations on the result of FPTLU in order to bound the error.

4.2.1 Fast Pivoting for Truncated LU Factorization

Truncated LU factorization aims to compute $\mathbf{\Pi}_1\mathbf{A}\mathbf{\Pi}_2 = \widehat{\mathbf{L}}\widehat{\mathbf{U}} + \begin{pmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{S}\end{pmatrix}$, where $\mathbf{\Pi}_1$ and $\mathbf{\Pi}_2$ are permutation matrices, $\mathbf{S}$ is the Schur complement. In each iteration, the vanilla complete pivoting of LU factorization choose the largest magnitude element in the Schur complement as the pivot. This procedure is very expensive. To reduce the computational cost, we adopt EELM algorithm to estimate pivots of LU. We present FPTLU in Algorithm 2. Note that line 5-7 of Algorithm 2 is standard LU update. Note that the sketched matrix $\mathbf{B}$ can be updated efficiently as in (Anderson and Gu, 2017) (see Appendix B).

Algorithm 2 Fast Pivoting for Truncated LU factorization

Input: data matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, target number of columns and rows ℓ.
Output: $\widehat{\mathbf{L}} \in \mathbb{R}^{m\times \ell}$, $\widehat{\mathbf{U}} \in \mathbb{R}^{\ell\times n}$, $\mathbf{\Pi}_1$, $\mathbf{\Pi}_2$.

1: Generate JL transform matrix $\mathbf{\Omega} \in \mathbb{R}^{p_1\times m}$,compute $\mathbf{B} = \mathbf{\Omega}\mathbf{A}$, $\mathbf{\Pi}_1 = \mathbf{I}_m$, $\mathbf{\Pi}_2 = \mathbf{I}_n$.
2: **for** $i = 1, \dots, \ell$ **do**
3: $[c, r, x]$ = EELM($\mathbf{A}$,$\mathbf{B}$).
4: Swap i-th and c-th column of $\mathbf{A}$, update $\mathbf{\Pi}_2$, Swap i-th and r-th row of $\mathbf{A}$, update $\mathbf{\Pi}_1$.
5: $\mathbf{A}_{i:m,i} = \mathbf{A}_{i:m,i} - \mathbf{A}_{i:m,1:i-1}\mathbf{A}_{1:i-1,i}$, $\mathbf{A}_{i+1:m,i} = \mathbf{A}_{i+1:m,i}/\mathbf{A}_{i,i}$.
6: $\mathbf{A}_{i,i+1:n} = \mathbf{A}_{i,i+1:n} - \mathbf{A}_{i,1:i-1}\mathbf{A}_{1:i-1,i+1:n}$,
7: Update $\mathbf{B}_{:,i+1:n}$.
8: **end for**
9: Let $\widehat{\mathbf{L}}$ be the lower triangular part of $\mathbf{A}_{:,1:\ell}$ with unit diagonal.
10: Let $\widehat{\mathbf{U}}$ be the upper triangular part of $\mathbf{A}_{1:\ell,:}$.

4.2.2 Fast Spectrum-Revealing Pivoting

Given a truncated LU factorization in the following form,

$$\mathbf{\Pi}_1\mathbf{A}\mathbf{\Pi}_2 = \begin{pmatrix}\mathbf{L}_{11} & & \\ \mathbf{l}_{21}^\top & 1 & \\ \mathbf{L}_{31} & & \mathbf{I}\end{pmatrix}\begin{pmatrix}\mathbf{U}_{11} & \mathbf{u}_{12} & \mathbf{U}_{13}\\ & \alpha & \mathbf{s}_{12}^\top\\ & \mathbf{s}_{21} & \mathbf{S}_{22}\end{pmatrix},$$

where $\widehat{\mathbf{L}} = \begin{pmatrix}\mathbf{L}_{11}\\ \mathbf{l}_{21}\\ \mathbf{L}_{31}\end{pmatrix}$, $\widehat{\mathbf{U}} = \begin{pmatrix}\mathbf{U}_{11} & \mathbf{u}_{12} & \mathbf{U}_{13}\end{pmatrix}$ and $\mathbf{S} = \begin{pmatrix}\alpha & \mathbf{s}_{12}^\top\\ \mathbf{s}_{21} & \mathbf{S}_{22}\end{pmatrix}$, FSRP aims to find the element α with the largest magnitude in $\mathbf{S}$ and the largest element β in matrix $|\overline{\mathbf{A}}_{11}^{-\top}|$ in each iteration. Here $\overline{\mathbf{A}}_{11} = \begin{pmatrix}\mathbf{L}_{11} & \\ \mathbf{l}_{21}^\top & 1\end{pmatrix}\begin{pmatrix}\mathbf{U}_{11} & \mathbf{u}_{12}\\ & \alpha\end{pmatrix}$. Since computing $\mathbf{S}$ is very expensive, we use EELM to estimate α and β rather than compute their accurate value. We present the FSRP in Algorithm 3. In each iteration, the sketched matrix $\mathbf{R}$ and $\mathbf{W}$ only needs a rank-2 update after each swap. We leverage techniques in (Gondzio, 1992) to turn $\mathbf{L}$ and $\mathbf{U}$ back into trapezoidal form after the swaps are performed. We show details of updating $\widehat{\mathbf{L}}$ and $\widehat{\mathbf{U}}$ in Appendix C.

4.3 Spectrum-Revealing CUR Decomposition

The permutation matrices produced by FSRP algorithm indicate which columns and rows we should choose for CUR decomposition. Namely, we compute matrix $\mathbf{C}$ and $\mathbf{R}$ as

$$\mathbf{C} = \mathbf{\Pi}_1^\top\widehat{\mathbf{L}}\widehat{\mathbf{U}}_{:,1:\ell} \quad \text{and} \quad \mathbf{R} = \widehat{\mathbf{L}}_{1:\ell,:}\widehat{\mathbf{U}}\mathbf{\Pi}_2^\top.$$

Algorithm 3 Fast Spectrum-revealing Pivoting

Input: Truncated LU factorization $\mathbf{\Pi}_1\mathbf{A}\mathbf{\Pi}_2 \approx \widehat{\mathbf{L}}\widehat{\mathbf{U}}$ and tolerance f.
Output: $\widehat{\mathbf{L}} \in \mathbb{R}^{m\times\ell}$, $\widehat{\mathbf{U}} \in \mathbb{R}^{\ell\times n}$, $\mathbf{\Pi}_1$, $\mathbf{\Pi}_2$.

1: Generate JL transform matrix $\mathbf{\Omega}_1 \in \mathbb{R}^{p_2\times(m-\ell)}$, $\mathbf{\Omega}_2 \in \mathbb{R}^{p_3\times(\ell+1)}$.
2: $\mathbf{R} = \mathbf{\Omega}_1\mathbf{S}$, $[s_r, s_c, \alpha] = \mathrm{EELM}(\mathbf{S}, \mathbf{R})$.
3: Swap the $(\ell+1)$-th row and the $(\ell+s_r)$-th row of $\widehat{\mathbf{L}}$ and update $\mathbf{\Pi}_1$ correspondingly.
4: Swap the $(\ell+1)$-th column and the $(\ell+s_c)$-th column of $\widehat{\mathbf{U}}$ and update $\mathbf{\Pi}_2$ correspondingly.
5: $\mathbf{W} = \mathbf{\Omega}_2\overline{\mathbf{A}}_{11}^{-\top}$, $[a_r, a_c, \beta] = \mathrm{EELM}(\overline{\mathbf{A}}_{11}^{-\top}, \mathbf{W})$.
6: **while** $\alpha\beta > f$ **do**
7: Exchange the a_r-th row and the $(\ell+1)$-th row of $\mathbf{A}$ and update $\widehat{\mathbf{L}}$.
8: Exchange the a_c-th column and the $(\ell+1)$-th column of $\mathbf{A}$ and update $\widehat{\mathbf{U}}$.
9: Update $\mathbf{\Pi}_1$, $\mathbf{\Pi}_2$, $\mathbf{S}$, $\mathbf{R}$ and perform step 3,4,5.
10: Update $\mathbf{W}$, let $[a_r, a_c, \beta] = \mathrm{EELM}(\overline{\mathbf{A}}_{11}^{-\top}, \mathbf{W})$.
11: **end while**

Algorithm 4 Spectrum-Revealing CUR decomposition

Input: Data matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, target rank k, target column and row number ℓ, $f > 1$.
Output: $\widehat{\mathbf{A}} \in \mathbb{R}^{m\times n}$ and $\widehat{\mathbf{A}}_k \in \mathbb{R}^{m\times n}$.

1: Perform Algorithm 2 to compute a truncated LU factorization $\mathbf{\Pi}_1\mathbf{A}\mathbf{\Pi}_2 \approx \widehat{\mathbf{L}}\widehat{\mathbf{U}}$.
2: Perfrom Algorithm 3 to update $\mathbf{\Pi}_1$, $\mathbf{\Pi}_2$, $\widehat{\mathbf{L}}$ and $\widehat{\mathbf{U}}$.
3: Compute $\mathbf{C} = \mathbf{\Pi}_1^\top\widehat{\mathbf{L}}\widehat{\mathbf{U}}_{:,1:\ell}$ and $\mathbf{R} = \widehat{\mathbf{L}}_{1:\ell,:}\widehat{\mathbf{U}}\mathbf{\Pi}_2^\top$.
4: Perform StableCUR Algorithm to obtain $\widehat{\mathbf{A}}$ and $\widehat{\mathbf{A}}_k$.

To make our algorithm more stable, we construct the approximation matrices by StableCUR (Anderson et al., 2015), which is shown in Appendix A. We show our SRCUR algorithm in Algorithm 4.

5 Theoretical Analysis

In this section, we analyze time complexity of our algorithm in Section 5.1 and provide spectral gap error bounds on singular values, 2-norm and Frobenius norm in Section 5.2.

5.1 Time complexity

It is obvious that the time cost of EELM is $O(m+pn)$. Then we discuss the time complexity of FPTLU. If we choose Gaussian random projection matrix as the JL transform matrix, we only require $O(p_1\mathrm{nnz}(\mathbf{A}))$ to initialize matrix $\mathbf{B}$. In each iteration, FPTLU require $O(p_1 n)$ time to update $\mathbf{B}$ and $O(\ell^2(m+n))$ time to perform standard LU update(line 5-7). Since $p_1 = O(\log n)$, FPTLU can compute a truncated LU factorization in $O(\mathrm{nnz}(\mathbf{A})\log n + \ell^2(m+n))$ time.

To analyze the complexity of the FSRP, we first give the following theorem which shows the upper bound of number of iterations.

Theorem 2. *If the input matrix $\widehat{\mathbf{L}}$ and $\widehat{\mathbf{U}}$ are obtained by Algorithm 2, Algorithm 3 performs at most $O(\ell\log(mn))$ number of iterations with probability* 0.99.

We require $O(\mathrm{nnz}(\mathbf{A})\log n + \ell(m+n)\log n)$ to initialize matrix $\mathbf{R}$ and $O(\ell^2\log\ell)$ to initialize matrix $\mathbf{W}$. In each iteration, the FSRP costs $O(pn+\ell(m+n)+\ell^2\log\ell)$. Using Theorem 2, we know that the time complexity of FSRP is $O(\mathrm{nnz}(\mathbf{A})\log\ell + \ell pn\log(mn) + \ell^2(m+n)\log(mn) + \ell^3\log(mn)\log\ell)$. The StableCUR require $O(\ell^2(m+n))$ time to construct the CUR approximation. Therefore, our SRCUR algorithm needs $O(\mathrm{nnz}(\mathbf{A})\log n) + \tilde{O}(\ell^2(m+n))$ time in total.

5.2 Spectral Gap Error Bounds

We present the norm error bounds in Theorem 3 and the singular value bound in Theorem 4. The detailed proofs are presented in the supplementary material due to page limit.

Theorem 3. *For $\gamma = O(f\ell\sqrt{mn})$, Algorithm 4 satisfies:*

$$\|\mathbf{A} - \widehat{\mathbf{A}}\|_2 \le \|\mathbf{A} - \widehat{\mathbf{A}}\|_F \le \gamma\sigma_{\ell+1}(\mathbf{A}), \tag{1}$$

$$\|\mathbf{A} - \widehat{\mathbf{A}}_k\|_F^2 \le \left(1 + \frac{2\gamma^2\sigma_{\ell+1}^2(\mathbf{A})}{\sum_{i=k+1}^{\mathrm{rank}(\mathbf{A})}\sigma_i^2(\mathbf{A})}\right)\|\mathbf{A} - \mathbf{A}_k\|_F^2, \tag{2}$$

$$\|\mathbf{A} - \widehat{\mathbf{A}}_k\|_2^2 \le \left(1 + 2\gamma^2\frac{\sigma_{\ell+1}^2(\mathbf{A})}{\sigma_{k+1}^2(\mathbf{A})}\right)\|\mathbf{A} - \mathbf{A}_k\|_2^2, \tag{3}$$

with probability 0.98.

Theorem 4. *For $1 \le j \le \ell$ and $\gamma = O(f\ell\sqrt{mn})$, Algorithm 4 satisfies:*

$$\sigma_j(\mathbf{A}) \ge \sigma_j(\widehat{\mathbf{A}}) \ge \sqrt{1 - 2\gamma^2\left(\frac{\sigma_{\ell+1}(\mathbf{A})}{\sigma_j(\mathbf{A})}\right)^2}\,\sigma_j(\mathbf{A})$$

with probability 0.98.

5.3 Discussion

We would like to emphasize the differences between our work and those methods which achieve $1+\varepsilon$ relative error bounds (Drineas et al., 2008; Wang and Zhang, 2013; Boutsidis and Woodruff, 2017). First, $1+\varepsilon$ relative error bounds only bound Frobenius norm, while

we provide bounds on spectral norm, Frobenius norm and singular values. In addition, our bounds are data dependent, while $1+\varepsilon$ relative error bounds depend on the universal constant ε.

Now we compare our bounds with DetUCS algorithm (Anderson et al., 2015). First, both our bounds and those of DetUCS are related to the quadratic of the spectral gap $\sigma^2_{\ell+1}(\mathbf{A})/\sigma^2_{k+1}(\mathbf{A})$. Thus, our bounds have the same order as that of DetUCS. However, DetUCS costs $O(\ell mn(m+n))$ time, which is much higher than our costs.

Finally, we discuss the differences between our method and SRLU (Anderson and Gu, 2017). First, our method and SRLU are designed for different tasks. The goal of SRLU is to obtain a high quality LU factorization, while we aim to compute CUR decomposition. Also, SRLU does not have time guarantees on the extra swap procedure. Thus, our FSRLU is faster than SRLU while maintaining the same theoretical guarantee. Though SRLU provided bounds on $\|\mathbf{\Pi}_1\mathbf{A}\mathbf{\Pi}_2 - \widehat{\mathbf{L}}\mathbf{M}\widehat{\mathbf{U}}\|_\xi$, where $\mathbf{M} = \widehat{\mathbf{L}}^\dagger\mathbf{A}\widehat{\mathbf{U}}^\dagger$ and $\xi \in \{2, F\}$. Their approximation is not rank-constrained and worse than our result.

6 Experiments

In this section, we empirically evaluate the performance of our SRCUR algorithm. We choose Gaussian random projection matrix as the JL transform matrix in SRCUR. The baseline methods include uniform sampling (Williams and Seeger, 2001), subspace sampling (Drineas et al., 2008), near-optimal sampling (Wang and Zhang, 2013; Boutsidis and Woodruff, 2017) and deterministic unweighted column selection (Anderson et al., 2015).

In our empirical evaluation, we consider following six measures: $\|\mathbf{A}-\widehat{\mathbf{A}}\|_F/\|\mathbf{A}-\mathbf{A}_k\|_F$, $\|\mathbf{A}-\widehat{\mathbf{A}}\|_2/\|\mathbf{A}-\mathbf{A}_k\|_2$, $\|\mathbf{A} - \widehat{\mathbf{A}}_k\|_F/\|\mathbf{A} - \mathbf{A}_k\|_F$, $\|\mathbf{A} - \widehat{\mathbf{A}}_k\|_2/\|\mathbf{A} - \mathbf{A}_k\|_2$, $\sigma_k(\widehat{\mathbf{A}})/\sigma_k(\mathbf{A})$ and running time. We compare CUR algorithms on the following four datasets: The Extended Yale Face Database B (Georghiades et al., 2001), Dexter (Guyon et al., 2005), Sido[1] and Reuters[2]. We summarize the information of these data sets in Table 1 and show the singular value distribution in Figure 1. We choose $\ell = k+1, 2k, \cdots, 10k$ for all datasets, where $k = \lceil \min(m,n)/200 \rceil$. We report our results in Figures 2-5. All results of randomized algorithm are averaged over 5 runs. The experiments are all performed in Matlab R2017b.

The figures show that our algorithm is much more efficient than state-of-the-art CUR algorithms. The main reason is that state-of-the-art CUR algorithms all require to perform SVD or randomized SVD while SRCUR is SVD-free. The randomized SVD, though has good time complexity Clarkson and Woodruff (2017), is much less efficient than LU update because the constant of SVD is extreme large. Furthermore, the approximation errors of SRCUR algorithm are better than or comparable with other algorithms. Especially, SRCUR performs better than other methods when ℓ is close to k. This result matches our theoretical analysis in Section 5.

Table 1: Summary of datasets for CUR matrix decomposition

Dataset	m	n	%nnz
Extend Yale Face B	2414	1024	97.84
Dexter	2600	20000	4.8
Sido	4932	12678	9.84
Reuters	8293	18933	0.25

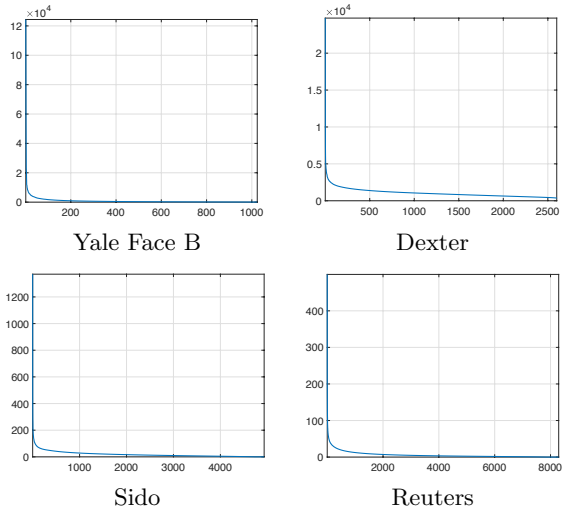
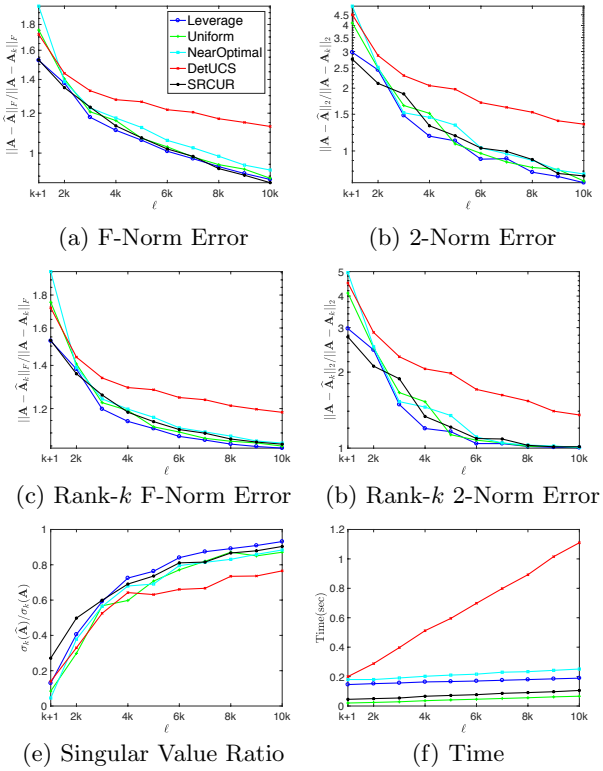


Figure 1: Singular value distribution

7 Conclusions

In this paper, we have proposed Spectrum-Revealing CUR decomposition algorithm. We have built the bridge between CUR decomposition and LU factorization, and leveraged truncated LU factorization to select columns and rows for CUR decomposition. We have develop spectral gap error bound on spectral norm and Frobenius norm and singular values. These bounds confirm high quality approximations computed by our methods for matrices with fast enough singular value decay, a property which is typically observed in practical data matrices. We have also conducted experiments to empirically demonstrate the effectiveness and reliability of our algorithms on four machine learning datasets.

[1] http://www.causality.inf.ethz.ch/data/SIDO.html
[2] http://www.zjucadcg.cn/dengcai/Data/TextData.html

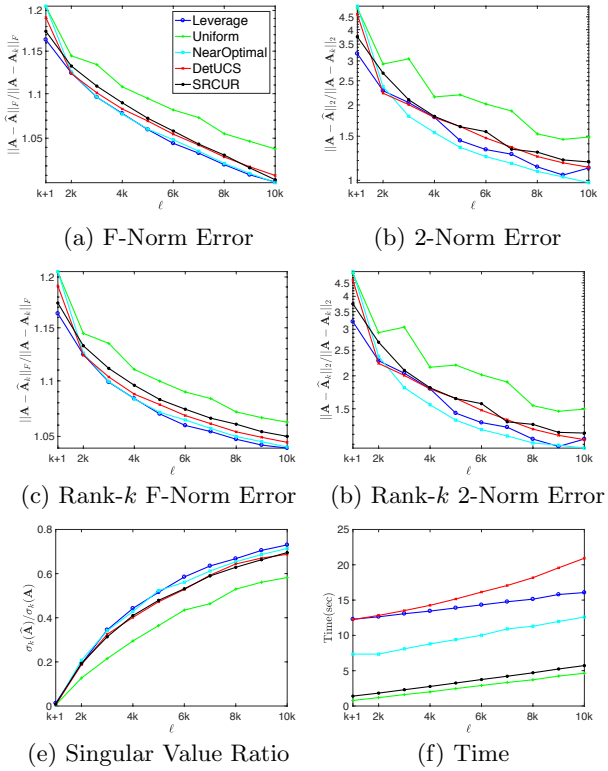


(a) F-Norm Error (b) 2-Norm Error

(c) Rank-k F-Norm Error (b) Rank-k 2-Norm Error

(e) Singular Value Ratio (f) Time

Figure 2: Results of CUR decomposition algorithms comparison on Yale Face B data matrix.

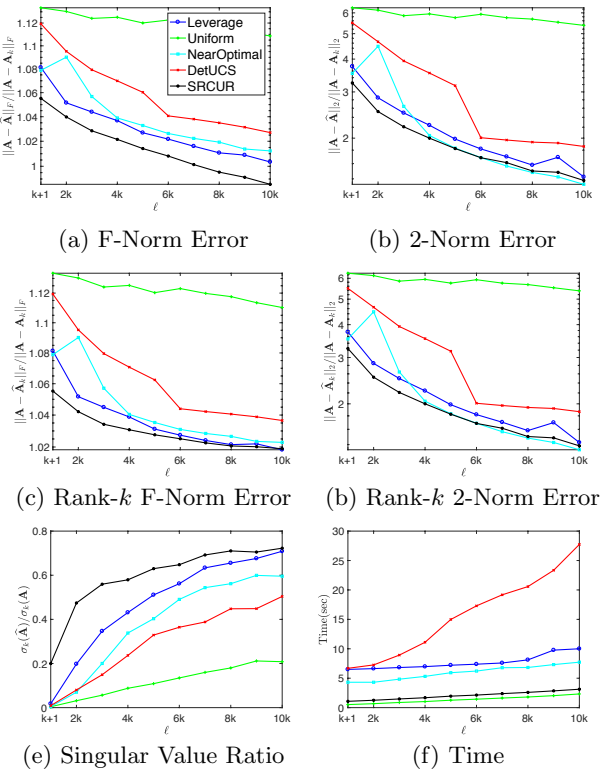


(a) F-Norm Error (b) 2-Norm Error

(c) Rank-k F-Norm Error (b) Rank-k 2-Norm Error

(e) Singular Value Ratio (f) Time

Figure 4: Results of CUR decomposition algorithms comparison on Sido data matrix.

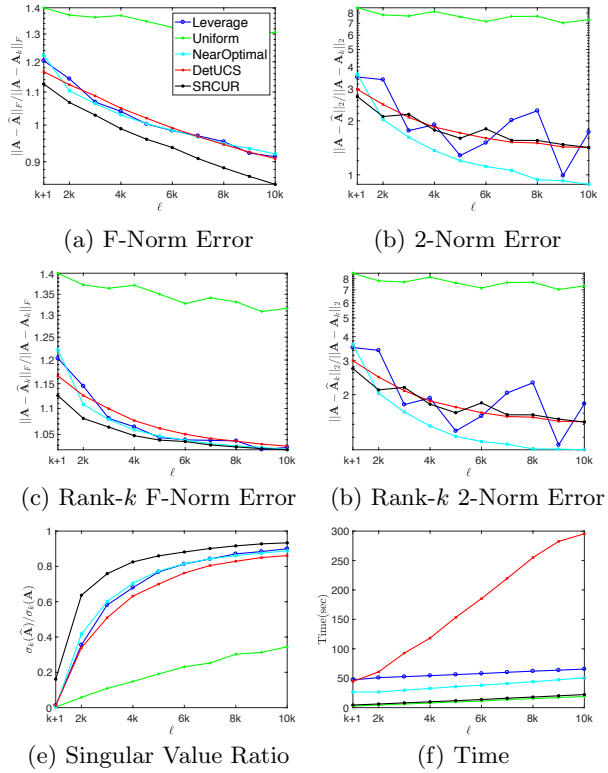


(a) F-Norm Error (b) 2-Norm Error

(c) Rank-k F-Norm Error (b) Rank-k 2-Norm Error

(e) Singular Value Ratio (f) Time

Figure 3: Results of CUR decomposition algorithms comparison on Dexter data matrix.



(a) F-Norm Error (b) 2-Norm Error

(c) Rank-k F-Norm Error (b) Rank-k 2-Norm Error

(e) Singular Value Ratio (f) Time

Figure 5: Results of CUR decomposition algorithms comparison on Reuters data matrix.

Acknowledgements

The work is supported by the National Natural Science Foundation of China (61702327, 61772333, 61632017). Ming Gu is supported by the NSF (1760316). Zhihua Zhang is supported by the National Natural Science Foundation of China (No. 11771002) and Beijing Natural Science Foundation (Z190001). Yong Yu is the corresponding author.

References

Anderson, D. and Gu, M. (2017). An efficient, sparsity-preserving, online algorithm for low-rank approximation. In *International Conference on Machine Learning*, pages 156–165.

Anderson, D. G., Du, S., Mahoney, M., Melgaard, C., Wu, K., and Gu, M. (2015). Spectral gap error bounds for improving cur matrix decomposition and the nyström method. In *AISTATS*.

Berry, M. W., Pulatova, S. A., and Stewart, G. (2005). Algorithm 844: Computing sparse reduced-rank approximations to sparse matrices. *ACM Transactions on Mathematical Software (TOMS)*, 31(2):252–269.

Bien, J., Xu, Y., and Mahoney, M. W. (2010). Cur from a sparse optimization viewpoint. In *Advances in Neural Information Processing Systems*, pages 217–225.

Boutsidis, C. and Woodruff, D. P. (2017). Optimal cur matrix decompositions. *SIAM Journal on Computing*, 46(2):543–589.

Chan, T. F. (1987). Rank revealing qr factorizations. *Linear algebra and its applications*, 88:67–82.

Clarkson, K. L. and Woodruff, D. P. (2017). Low-rank approximation and regression in input sparsity time. *Journal of the ACM (JACM)*, 63(6):54.

Drineas, P., Kannan, R., and Mahoney, M. W. (2006). Fast monte carlo algorithms for matrices iii: Computing a compressed approximate matrix decomposition. *SIAM Journal on Computing*, 36(1):184–206.

Drineas, P., Mahoney, M. W., and Muthukrishnan, S. (2008). Relative-error cur matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30(2):844–881.

Georghiades, A. S., Belhumeur, P. N., and Kriegman, D. J. (2001). From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE transactions on pattern analysis and machine intelligence*, 23(6):643–660.

Gondzio, J. (1992). Stable algorithm for updating dense lu factorization after row or column exchange and row and column addition or deletion. *Optimization*, 23(1):7–26.

Gu, M. (2015). Subspace iteration randomization and singular value problems. *SIAM Journal on Scientific Computing*, 37(3):A1139–A1173.

Gu, M. and Eisenstat, S. C. (1996). Efficient algorithms for computing a strong rank-revealing qr factorization. *SIAM Journal on Scientific Computing*, 17(4):848–869.

Guyon, I., Gunn, S., Ben-Hur, A., and Dror, G. (2005). Result analysis of the nips 2003 feature selection challenge. In *Advances in neural information processing systems*, pages 545–552.

Higham, N. J. and Relton, S. D. (2016). Estimating the largest elements of a matrix. *SIAM Journal on Scientific Computing*, 38(5):C584–C601.

Johnson, W. B. and Lindenstrauss, J. (1984). Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1.

Mackey, L. W., Jordan, M. I., and Talwalkar, A. (2011). Divide-and-conquer matrix factorization. In *Advances in Neural Information Processing Systems*, pages 1134–1142.

Mahoney, M. W. and Drineas, P. (2009). Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702.

Mahoney, M. W. et al. (2011). Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224.

Mahoney, M. W., Maggioni, M., and Drineas, P. (2008). Tensor-cur decompositions for tensor-based data. *SIAM Journal on Matrix Analysis and Applications*, 30(3):957–987.

Melgaard, C. and Gu, M. (2015). Gaussian elimination with randomized complete pivoting. *arXiv preprint arXiv:1511.08528*.

Miranian, L. and Gu, M. (2003). Strong rank revealing lu factorizations. *Linear algebra and its applications*, 367:1–16.

Pan, C.-T. (2000). On the existence and computation of rank-revealing lu factorizations. *Linear Algebra and its Applications*, 316(1-3):199–222.

Song, Z., Woodruff, D. P., and Zhong, P. (2017). Low rank approximation with entrywise l1-norm error. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 688–701. ACM.

Song, Z., Woodruff, D. P., and Zhong, P. (2019). Relative error tensor low rank approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2772–2789. Society for Industrial and Applied Mathematics.

Stewart, G. (1999). Four algorithms for the the efficient computation of truncated pivoted qr approximations to a sparse matrix. *Numerische Mathematik*, 83(2):313–323.

Trefethen, L. N. and Bau III, D. (1997). *Numerical linear algebra*, volume 50. Siam.

Wang, S. and Zhang, Z. (2012). A scalable cur matrix decomposition algorithm: Lower time complexity and tighter bound. In *Advances in Neural Information Processing Systems*, pages 647–655.

Wang, S. and Zhang, Z. (2013). Improving cur matrix decomposition and the nyström approximation via adaptive sampling. *Journal of Machine Learning Research*, 14(1):2729–2769.

Wang, S., Zhang, Z., and Zhang, T. (2016). Towards more efficient spsd matrix approximation and cur matrix decomposition. *Journal of Machine Learning Research*, 17(210):1–49.

Williams, C. K. and Seeger, M. (2001). Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688.

Xu, M., Jin, R., and Zhou, Z.-H. (2015). Cur algorithm for partially observed matrices. In *ICML*, pages 1412–1421.

A Stable CUR Decomposition

Algorithm 5 StableCUR (Anderson et al., 2015)

Input: $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{C} \in \mathbb{R}^{m\times c}$, $\mathbf{R} \in \mathbb{R}^{r\times n}$ and target rank k.
Output: $\widehat{\mathbf{A}} \in \mathbb{R}^{m\times n}$ and $\widehat{\mathbf{A}}_k \in \mathbb{R}^{m\times n}$.

1: Do QR factorization on $\mathbf{R}^\top$ to obtain a basis of rows of $\mathbf{R}$, $\mathbf{R} = \mathbf{R}_r\mathbf{Q}_r$;
2: Do QR factorization on $\mathbf{C}$ to obtain a basis of columns of $\mathbf{C}$, $\mathbf{C} = \mathbf{Q}_c\mathbf{R}_c$;
3: $\mathbf{E} = \mathbf{Q}_c^\top\mathbf{A}\mathbf{Q}_r^\top$, $\widehat{\mathbf{A}} = \mathbf{Q}_c\mathbf{B}\mathbf{Q}_r$;
4: Compute rank-k truncated SVD on $\mathbf{E}$ to obtain $\mathbf{B}_k$;
5: $\widehat{\mathbf{A}}_k = \mathbf{Q}_c\mathbf{E}_k\mathbf{Q}_r$.

B Update of $\mathbf{B}_{:,i+1:n}$ in Algorithm 2

We can partition $\mathbf{B}_{:,i:n}$ as follows:

$$\mathbf{B}_{:,i:n} = \overset{\begin{matrix}1 & n-i\end{matrix}}{\begin{pmatrix}\mathbf{B}_1 & \mathbf{B}_2\end{pmatrix}} = \overset{\begin{matrix}1 & n-i\end{matrix}}{\begin{pmatrix}\mathbf{\Omega}_1 & \mathbf{\Omega}_2\end{pmatrix}} \overset{\begin{matrix}1 & n-i\end{matrix}}{\begin{pmatrix}\mathbf{L}_{11} & \\ \mathbf{L}_{21} & \mathbf{I}\end{pmatrix}} \overset{\begin{matrix}1 & n-i\end{matrix}}{\begin{pmatrix}\mathbf{U}_{11} & \mathbf{U}_{12} \\ & \mathbf{S}\end{pmatrix}},$$

then we have

$$\mathbf{B}_{:,i+1:n} = \mathbf{B}_2^{new} = \mathbf{\Omega}_2\mathbf{S} = \mathbf{B}_2 - \begin{pmatrix}\mathbf{\Omega}_1 & \mathbf{\Omega}_2\end{pmatrix}\begin{pmatrix}\mathbf{L}_{11} \\ \mathbf{L}_{21}\end{pmatrix}\mathbf{U}_{12}. \tag{4}$$

C Swapping Algorithms

Let

$$\overline{\mathbf{L}} = \begin{pmatrix}\mathbf{L}_{11} & \\ \mathbf{l}_{21}^\top & 1 \\ \mathbf{L}_{31} & \mathbf{0}\end{pmatrix} \text{ and } \overline{\mathbf{U}} = \begin{pmatrix}\mathbf{U}_{11} & \mathbf{u}_{12} & \mathbf{U}_{13} \\ & \alpha & \mathbf{s}_{12}^\top\end{pmatrix},$$

then we present Row Swap Algorithm and Column Swap Algorithm in Algorithm 6 and Algorithm 7, respectively.

Algorithm 6 Row Swap

1: Move a_r-th row to the position ℓ+1 in $\overline{\mathbf{L}}$ and shift rows $a_r+1, a_r+2, \ldots, \ell$ of one position up.
2: **for** $i = a_r$ to ℓ **do**
3: Let $x = \overline{\mathbf{L}}_{i,i}$, $c = \overline{\mathbf{U}}_{i,i+1}$, $d = \overline{\mathbf{U}}_{i+1,i+1}$.
4: $\mathbf{H}_1 = \begin{pmatrix} x & 1 \\ \frac{d}{cx+d} & -\frac{c}{cx+d}\end{pmatrix}$, $\mathbf{H}_2 = \begin{pmatrix} x & 1 \\ 0 & \sqrt{x^2+1}\end{pmatrix}$.
5: **if** $x = 0$ or $\kappa(\mathbf{H}_1) < \kappa(\mathbf{H}_2)$ **then**
6: $\mathbf{H}_i = \mathbf{H}_1$.
7: Exchange the i-th and the $(i+1)$-th column of $\overline{\mathbf{U}}$.
8: **else**
9: $\mathbf{H}_i = \mathbf{H}_2$.
10: **end if**
Let $\overline{\mathbf{L}}_{:,i:i+1} = \overline{\mathbf{L}}_{:,i:i+1}\mathbf{H}_i^{-1}$, $\overline{\mathbf{U}}_{i:i+1,:} = \mathbf{H}_i\overline{\mathbf{U}}_{i:i+1,:}$
11: **end for**
12: Update permutation matrix $\mathbf{\Pi}_1$ and $\mathbf{\Pi}_2$ according to the row and column exchange.
13: Let $\widehat{\mathbf{L}} = \overline{\mathbf{L}}_{:,1:\ell}$ and $\widehat{\mathbf{U}} = \overline{\mathbf{U}}_{1:\ell,:}$.

Algorithm 7 Column Swap

Move a_c-th column to the position ℓ+1 in $\overline{\mathbf{U}}$ and shift columns a_c+1, a_c+2, ..., ℓ of one position to the left.
for $i = a_c$ to ℓ **do**
  Let $a = \overline{\mathbf{U}}_{i,i}$, $b = \overline{\mathbf{U}}_{i+1,i}$, $z = \overline{\mathbf{L}}_{i+1,i}$, $t = b/a$.
  Let $\mathbf{H}_1 = \begin{pmatrix} z & 1 \\ \frac{b}{az+b} & -\frac{a}{az+b} \end{pmatrix}$, $\mathbf{H}_2 = \begin{pmatrix} 1 & 0 \\ -\frac{b}{a} & 1 \end{pmatrix}$.
  if $a = 0$ or $\kappa(\mathbf{H}_1) < \kappa(\mathbf{H}_2)$ **then**
    Let $\mathbf{H}_i = \mathbf{H}_1$.
    Exchange the i-th and the $(i+1)$-th row of $\overline{\mathbf{L}}$.
  else
    Let $\mathbf{H}_i = \mathbf{H}_2$.
  end if
  Let $\overline{\mathbf{L}}_{:,i:i+1} = \overline{\mathbf{L}}_{:,i:i+1}\mathbf{H}_i^{-1}$, $\overline{\mathbf{U}}_{i:i+1,:} = \mathbf{H}_i\overline{\mathbf{U}}_{i:i+1,:}$.
end for
Update permutation matrix $\mathbf{\Pi}_1$ and $\mathbf{\Pi}_2$ according to the row and column exchange.
Let $\widehat{\mathbf{L}} = \overline{\mathbf{L}}_{:,1:\ell}$ and $\widehat{\mathbf{U}} = \overline{\mathbf{U}}_{1:\ell,:}$.

Now we analyze the correctness and stability of Algorithm 6 and Algorithm 7. We first analyze Algorithm 6. After the operation in line 1, the $\overline{\mathbf{L}}$ has been transformed to an Hessenberg matrix with subdiagonal nonzeros at columns $a_r + 1, a_r + 2, \ldots, \ell + 1$. The rest operations aim to find a matrix $\mathbf{H}$ and a permutation matrix $\mathbf{P}$ such that

$$\overline{\mathbf{L}}_{new} = \overline{\mathbf{L}}\mathbf{H}^{-1} \quad \text{and} \quad \overline{\mathbf{U}}_{new} = \mathbf{H}\overline{\mathbf{U}}\mathbf{P}$$

In the i-th iteration of Algorithm 6, we have

$$\mathbf{L}_i = \overline{\mathbf{L}}_{i:i+1,i+i+1} = \begin{pmatrix} x & 1 \\ z & v \end{pmatrix} \quad \text{and} \quad \mathbf{U}_i = \overline{\mathbf{U}}_{i:i+1,i+i+1} = \begin{pmatrix} a & c \\ 0 & d \end{pmatrix}$$

We aim to choose $\mathbf{H}_i$ and $\mathbf{P}_i$ such that $\mathbf{L}_i\mathbf{H}_i^{-1}$ is a lower triangular matrix and $\mathbf{H}_i\mathbf{U}_i\mathbf{P}_i$ is an upper triangular matrix. We provide two solutions. The first is $\mathbf{H}_1 = \begin{pmatrix} x & 1 \\ \frac{d}{cx+d} & -\frac{c}{cx+d} \end{pmatrix}$ with $\mathbf{P}_i = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. The second is $\mathbf{H}_2 = \begin{pmatrix} x & 1 \\ 0 & \sqrt{x^2+1} \end{pmatrix}$ with $\mathbf{P}_i = \mathbf{I}$. Since $d \neq 0$, at least one of $\mathbf{H}_1$ and $\mathbf{H}_2$ is well defined and nonsingular. This result means that Algorithm 6 is stable. Further, if both $\mathbf{H}_1$ and $\mathbf{H}_2$ are nonsingular, we choose the one with smaller condition number.

The analysis of Algorithm 7 is similar to that of Algorithm 7. Thus we do not present it in detail.

D Proof of Theorem 1

According to Lemma 1, $\mathbf{\Omega}$ satisfies $(\varepsilon, \delta/n)$-JL property. Thus for each column $\mathbf{a}$ in $\mathbf{A}$, we have

$$\Pr\left((1-\varepsilon)\|\mathbf{a}\|^2 \leq \|\frac{1}{\sqrt{p}}\mathbf{\Omega}\mathbf{a}\|^2 \leq (1+\varepsilon)\|\mathbf{a}\|^2\right) \geq 1 - \frac{\delta}{n}.$$

Assume that $c^* = \arg\max_{i \leq j \leq n} \|\mathbf{A}_{:,j}\|_2$. Then we have

$$\begin{aligned}
\|\mathbf{A}_{:,c}\|_2 &\geq \frac{1}{\sqrt{1+\varepsilon}}\|\frac{1}{\sqrt{p}}\mathbf{R}_{:,c}\|_2 \\
&\geq \frac{1}{\sqrt{1+\varepsilon}}\|\frac{1}{\sqrt{p}}\mathbf{R}_{:,c^*}\|_2 \qquad \text{because the optimality of } c \\
&\geq \sqrt{\frac{1-\varepsilon}{1+\varepsilon}}\|\mathbf{A}_{:,c^*}\|_2 \\
&\geq \sqrt{\frac{1-\varepsilon}{n(1+\varepsilon)}}||\mathbf{A}||_F.
\end{aligned}$$

with probability at least $\left(1-\frac{\delta}{n}\right)^n > 1-\delta$.
Thus, we can get

$$x = \|\mathbf{A}_{*,c}\|_{\max} \geq \sqrt{m}\|\mathbf{A}_{*,c}\|_2 \geq \sqrt{\frac{1-\varepsilon}{mn(1+\varepsilon)}}\|\mathbf{A}\|_F.$$

E Proof of Theorem 2

Let $\mathbf{A}_{11}$ be the top left $\ell \times \ell$ matrix of $\mathbf{\Pi}_1\mathbf{A}\mathbf{\Pi}_2$, then we have the following lemma:

Lemma 2. *Each swap in Algorithm 3 will increase the determinant* $\det(\mathbf{A}_{11})$ *by at least* f.

Proof. For each iteration, let $\mathbf{A}'_{11}$ be the top left $\ell \times \ell$ matrix of $\mathbf{\Pi}_1\mathbf{A}\mathbf{\Pi}_2$ after swap. Then, we have

$$\alpha = \frac{\det(\overline{\mathbf{A}}_{11})}{\det(\mathbf{A}_{11})}$$

and

$$\beta = (\overline{\mathbf{A}}_{11}^{-\top})_{a_r,a_c} = \frac{(\mathrm{adj}(\mathbf{A}_{11}))_{a_r,a_c}}{\det(\overline{\mathbf{A}}_{11})} = \frac{\det(\mathbf{A}'_{11})}{\det(\overline{\mathbf{A}}_{11})}.$$

Since the condition that we perform the iteration is $\alpha\beta > f$, we can obtain

$$\frac{\det(\mathbf{A}'_{11})}{\det(\mathbf{A}_{11})} > f.$$

□

Then, we prove Theorem 2. After performing k steps of TLURP, we have

$$\mathbf{\Pi}_1\mathbf{A}\mathbf{\Pi}_2 = \begin{pmatrix}\mathbf{L}_{11}^k \\ \mathbf{L}_{21}^k & \mathbf{I}\end{pmatrix}\begin{pmatrix}\mathbf{U}_{11}^k & \mathbf{U}_{12}^k \\ & \mathbf{S}_k\end{pmatrix}$$

where $\mathbf{L}_{11}^k$ and $\mathbf{U}_{11}^k$ are $k\times k$ matrix and $\mathbf{S}_k$ is the Schur complement. Let $\mathbf{A}_k = \mathbf{L}_{11}^k\mathbf{U}_{11}^k$, we can get $\frac{\det(\mathbf{A}_{k+1})}{\det(\mathbf{A}_k)} = \gamma_k$, where γ_k is the pivot chosen in $\mathbf{S}_k$.
The pivot of TLURP is selected by EELM. According to Theorem 1, we can achieve that

$$\gamma_\ell \geq \sqrt{\frac{1-\varepsilon}{(m-\ell)(n-\ell)(1+\varepsilon)}}\|\mathbf{S}_\ell\|_F \geq \sqrt{\frac{1-\varepsilon}{mn(1+\varepsilon)}}\|\mathbf{S}_\ell\|_F.$$

Since $\|\mathbf{S}_\ell\|_F \geq \|\mathbf{S}_\ell\|_2 \geq \sigma_{\ell+1}(\mathbf{A})$, we can obtain

$$\det(\mathbf{A}_\ell) \geq \left(\sqrt{\frac{1-\varepsilon}{mn(1+\varepsilon)}}\right)^\ell \prod_{i=1}^{\ell}\sigma_i(\mathbf{A}).$$

Assume that $\overline{\mathbf{A}}_\ell$ is the top left $\ell \times \ell$ matrix of $\mathbf{A}$ after performing Fast SRP. Then we have $\sigma_i(\overline{\mathbf{A}}_\ell) \leq \sigma_i(\mathbf{A})$ for $i = 1, \ldots, \ell$. So we can get, $\det(\overline{\mathbf{A}}_\ell) \leq \prod_{i=1}^{\ell}\sigma_i(\mathbf{A})$.
Since each swap increase the determinant $\det(\mathbf{A}_\ell)$ by at least a factor f, number of swaps is at most

$$\log_f \frac{\det(\overline{\mathbf{A}}_\ell)}{\det(\mathbf{A}_\ell)} \leq \log_f \left(\sqrt{\frac{mn(1+\varepsilon)}{1-\varepsilon}}\right)^\ell = \ell \log_f \sqrt{\frac{mn(1+\varepsilon)}{1-\varepsilon}}$$

with probability $1-\delta$. Since we assume $\varepsilon = \frac{1}{2}$ and $\delta = 0.01$, we obtain Theorem 2.

F Proof of Theorem 3

We first prove the following lemma:

Lemma 3. *For $\gamma = O(f\ell\sqrt{mn})$, the result of Fast SRP algorithm satisfies*

$$\|\mathbf{S}\|_F = \|\mathbf{\Pi}_1\mathbf{A}\mathbf{\Pi}_2 - \widehat{\mathbf{L}}\widehat{\mathbf{U}}\|_F \leq \gamma\sigma_{\ell+1}(\mathbf{A})$$

with probability 0.98.

Proof. Since α and β in Algorithm 3 is selected by Algorithm 1, according to Theorem 1, we have

$$\alpha \geq \sqrt{\frac{1-\varepsilon}{(m-\ell)(n-\ell)(1+\varepsilon)}}\|\mathbf{S}\|_F$$

with probability $1-\delta$ and

$$\beta \geq (\ell+1)\sqrt{\frac{1-\varepsilon}{1+\varepsilon}}\|(\overline{\mathbf{A}}_{11})^{-\top}\|_F$$

with probability $1-\delta$. Then, we have

$$\begin{aligned}
\|\mathbf{S}\|_F &= \frac{\|\mathbf{S}\|_F}{\alpha}\frac{\|\mathbf{A}_{11}^{-1}\|_F}{\beta}\alpha\beta\|\mathbf{A}_{11}^{-1}\|_F^{-1} \\
&\leq \frac{1+\varepsilon}{1-\varepsilon}(\ell+1)\sqrt{(m-\ell)(n-\ell)}\alpha\beta\|\mathbf{A}_{11}^{-1}\|_2^{-1} \\
&= \frac{1+\varepsilon}{1-\varepsilon}(\ell+1)\sqrt{(m-\ell)(n-\ell)}\alpha\beta\sigma_{\min}(\mathbf{A}_{11}) \\
&\leq \frac{1+\varepsilon}{1-\varepsilon}(\ell+1)\sqrt{(m-\ell)(n-\ell)}\alpha\beta\sigma_{\ell+1}(\mathbf{A})
\end{aligned}$$

with probability $1-2\delta$.
The last inequality is from the interlacing property of the singular values.
Since $\alpha\beta < f$, $\varepsilon = \frac{1}{2}, \delta = 0.01$, we have

$$||\mathbf{S}||_F \leq \gamma\sigma_{\ell+1}(\mathbf{A})$$

with probability at least 0.98. □

Let $\widetilde{\mathbf{A}} = \mathbf{\Pi}_1\mathbf{A}\mathbf{\Pi}_2$ and $\mathbf{M} = \widehat{\mathbf{L}}^{\dagger}\widetilde{\mathbf{A}}\widehat{\mathbf{U}}^{\dagger}$.Then we have

$$\widehat{\mathbf{L}}\mathbf{M}\widehat{\mathbf{U}} = \mathbf{\Pi}_1\mathbf{C}\mathbf{U}\mathbf{R}\mathbf{\Pi}_2. \tag{5}$$

Assume that $\widetilde{\mathbf{A}} = \widetilde{\mathbf{L}}\widetilde{\mathbf{U}}$ is the full LU factorization of matrix $\widetilde{\mathbf{A}}$, where $\widetilde{\mathbf{L}}_{:,1:\ell} = \widehat{\mathbf{L}}$ and $\widetilde{\mathbf{U}}_{1:\ell,:} = \widehat{\mathbf{U}}$.

Perform QR factorization on $\widetilde{\mathbf{L}}$ and $\widetilde{\mathbf{U}}^{\top}$, we have $\widetilde{\mathbf{L}} = \mathbf{Q}_L\mathbf{R}_L =: \begin{pmatrix}\mathbf{Q}_1^L & \mathbf{Q}_2^L\end{pmatrix}\begin{pmatrix}\mathbf{R}_{11}^L & \mathbf{R}_{12}^L \\ & \mathbf{R}_{22}^L\end{pmatrix}$ and $\widetilde{\mathbf{U}} = \mathbf{L}_U\mathbf{Q}_U =: \begin{pmatrix}\mathbf{L}_{11}^U & \\ \mathbf{L}_{21}^U & \mathbf{L}_{22}^U\end{pmatrix}\begin{pmatrix}\mathbf{Q}_1^U \\ \mathbf{Q}_2^U\end{pmatrix}$.

Then

$$\widehat{\mathbf{L}}\mathbf{M}\widehat{\mathbf{U}} = \mathbf{Q}_1^L\left(\mathbf{Q}_1^L\right)^{\top}\widetilde{\mathbf{A}}\left(\mathbf{Q}_1^U\right)^{\top}\mathbf{Q}_1^U.$$

According to Lemma 3 and Theorem 3 of (Anderson and Gu, 2017), we can get

$$\|\mathbf{A}-\widehat{\mathbf{A}}\|_2 \leq \|\mathbf{A}-\widehat{\mathbf{A}}\|_F = \|\mathbf{A}-\mathbf{C}\mathbf{U}\mathbf{R}\|_F = \|\mathbf{\Pi}_1\mathbf{A}\mathbf{\Pi}_2 - \widehat{\mathbf{L}}\mathbf{M}\widehat{\mathbf{U}}\|_F \leq \|\mathbf{S}\|_F \leq \gamma\sigma_{\ell+1}(\mathbf{A}).$$

Then we get formula (1). Let $\mathbf{D} = \widetilde{\mathbf{A}} - \widehat{\mathbf{L}}\widehat{\mathbf{U}}$. Note that

$$\begin{aligned}
\widetilde{\mathbf{A}}^{\top}\mathbf{Q}_2^L &= \left(\widehat{\mathbf{L}}\widehat{\mathbf{U}} + \mathbf{C}\right)^{\top}\mathbf{Q}_2^L \\
&= \left(\mathbf{Q}_1^L\mathbf{R}_{11}^L\mathbf{L}_{11}^U\mathbf{Q}_1^U + \mathbf{C}\right)^{\top}\mathbf{Q}_2^L \\
&= \left(\mathbf{Q}_1^U\right)^{\top}\left(\mathbf{L}_{11}^U\right)^{\top}\left(\mathbf{R}_{11}^L\right)^{\top}\left(\mathbf{Q}_1^L\right)^{\top}\mathbf{Q}_2^L + \mathbf{D}^{\top}\mathbf{Q}_2^L \\
&= \mathbf{D}^{\top}\mathbf{Q}_2^L.
\end{aligned} \tag{6}$$

Analogously

$$\widetilde{\mathbf{A}}\left(\mathbf{Q}_2^U\right)^\top = \mathbf{D}\left(\mathbf{Q}_2^U\right)^\top. \tag{7}$$

Thus,

$$\begin{aligned}
& \|\widetilde{\mathbf{A}} - (\widehat{\mathbf{L}}\mathbf{M}\widehat{\mathbf{U}})_k\|_F^2 \\
= \; & \|\widetilde{\mathbf{A}} - \mathbf{Q}_1^L\left((\mathbf{Q}_1^L)^\top\widetilde{\mathbf{A}}(\mathbf{Q}_1^U)^\top\right)_k\mathbf{Q}_1^U\|_F^2 \\
\leq \; & \|\widetilde{\mathbf{A}} - \mathbf{Q}_1^L(\mathbf{Q}_1^L)^\top\widetilde{\mathbf{A}}_k(\mathbf{Q}_1^U)^\top\mathbf{Q}_1^U\|_F^2 \\
= \; & \|\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}_k + \widetilde{\mathbf{A}}_k - \mathbf{Q}_1^L(\mathbf{Q}_1^L)^\top\widetilde{\mathbf{A}}_k + \mathbf{Q}_1^L(\mathbf{Q}_1^L)^\top\widetilde{\mathbf{A}}_k - \mathbf{Q}_1^L(\mathbf{Q}_1^L)^\top\widetilde{\mathbf{A}}_k(\mathbf{Q}_1^U)^\top\mathbf{Q}_1^U\|_F^2 \\
= \; & \|\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}_k + \widetilde{\mathbf{A}}_k - \mathbf{Q}_1^L(\mathbf{Q}_1^L)^\top\widetilde{\mathbf{A}}_k\|_F^2 + \|\mathbf{Q}_1^L(\mathbf{Q}_1^L)^\top\widetilde{\mathbf{A}}_k - \mathbf{Q}_1^L(\mathbf{Q}_1^L)^\top\widetilde{\mathbf{A}}_k(\mathbf{Q}_1^U)^\top\mathbf{Q}_1^U\|_F^2 \\
& +2\mathrm{tr}\left((\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}_k)^\top\mathbf{Q}_1^L(\mathbf{Q}_1^L)^\top\widetilde{\mathbf{A}}_k(\mathbf{I} - (\mathbf{Q}_1^U)^\top\mathbf{Q}_1^U)\right) \\
= \; & \|\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}_k\|_F^2 + \|(\mathbf{I} - \mathbf{Q}_1^L(\mathbf{Q}_1^L)^\top)\widetilde{\mathbf{A}}_k\|_F^2 + \|\mathbf{Q}_1^L(\mathbf{Q}_1^L)^\top\widetilde{\mathbf{A}}_k(\mathbf{I} - (\mathbf{Q}_1^U)^\top\mathbf{Q}_1^U)\|_F^2 \\
& +2\mathrm{tr}\left((\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}_k)^\top\mathbf{Q}_1^L(\mathbf{Q}_1^L)^\top\widetilde{\mathbf{A}}_k(\mathbf{I} - (\mathbf{Q}_1^U)^\top\mathbf{Q}_1^U)\right) \\
= \; & \|\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}_k\|_F^2 + \|\mathbf{Q}_2^L(\mathbf{Q}_2^L)^\top\widetilde{\mathbf{A}}_k\|_F^2 + \|\mathbf{Q}_1^L(\mathbf{Q}_1^L)^\top\widetilde{\mathbf{A}}_k(\mathbf{Q}_2^U)^\top\mathbf{Q}_2^U\|_F^2 \\
& +2\mathrm{tr}\left((\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}_k)^\top\mathbf{Q}_1^L(\mathbf{Q}_1^L)^\top\widetilde{\mathbf{A}}_k(\mathbf{Q}_2^U)^\top\mathbf{Q}_2^U\right) \\
= \; & \|\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}_k\|_F^2 + \|\mathbf{Q}_2^L(\mathbf{Q}_2^L)^\top\widetilde{\mathbf{A}}\|_F^2 - \|\mathbf{Q}_2^L(\mathbf{Q}_2^L)^\top(\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}_k)\|_F^2 + \|\widetilde{\mathbf{A}}_k(\mathbf{Q}_2^U)^\top\mathbf{Q}_2^U\|_F^2 \\
& -\|\mathbf{Q}_2^L(\mathbf{Q}_2^L)^\top\widetilde{\mathbf{A}}_k(\mathbf{Q}_2^U)^\top\mathbf{Q}_2^U\|_F^2 + 2\mathrm{tr}\left((\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}_k)^\top\mathbf{Q}_2^L(\mathbf{Q}_2^L)^\top\mathbf{Q}_2^L(\mathbf{Q}_2^L)^\top\widetilde{\mathbf{A}}_k(\mathbf{Q}_2^U)^\top\mathbf{Q}_2^U\right) \\
= \; & \|\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}_k\|_F^2 + \|\mathbf{Q}_2^L(\mathbf{Q}_2^L)^\top\widetilde{\mathbf{A}}\|_F^2 + \|\widetilde{\mathbf{A}}_k(\mathbf{Q}_2^U)^\top\mathbf{Q}_2^U\|_F^2 \\
& -\|\mathbf{Q}_2^L(\mathbf{Q}_2^L)^\top(\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}_k) - \mathbf{Q}_2^L(\mathbf{Q}_2^L)^\top\widetilde{\mathbf{A}}_k(\mathbf{Q}_2^U)^\top\mathbf{Q}_2^U\|_F^2 \\
\leq \; & \|\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}_k\|_F^2 + \|\mathbf{Q}_2^L(\mathbf{Q}_2^L)^\top\widetilde{\mathbf{A}}\|_F^2 + \|\widetilde{\mathbf{A}}_k(\mathbf{Q}_2^U)^\top\mathbf{Q}_2^U\|_F^2 \\
\leq \; & \|\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}_k\|_F^2 + \|\mathbf{Q}_2^L(\mathbf{Q}_2^L)^\top\widetilde{\mathbf{A}}\|_F^2 + \|\widetilde{\mathbf{A}}(\mathbf{Q}_2^U)^\top\mathbf{Q}_2^U\|_F^2 \\
= \; & \|\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}_k\|_F^2 + \|\mathbf{Q}_2^L(\mathbf{Q}_2^L)^\top\mathbf{D}\|_F^2 + \|\mathbf{D}(\mathbf{Q}_2^U)^\top\mathbf{Q}_2^U\|_F^2 \\
\leq \; & \|\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}_k\|_F^2 + 2\|\mathbf{D}\|_F^2 \\
= \; & \|\widetilde{\mathbf{A}} - \widetilde{\mathbf{A}}_k\|_F^2 + 2\|\mathbf{S}\|_F^2 \\
= \; & \|\mathbf{A} - \mathbf{A}_k\|_F^2 + 2\|\mathbf{S}\|_F^2 \\
\leq \; & \sum_{i=k+1}^{\mathrm{rank}(\mathbf{A})}\sigma_i^2(\mathbf{A}) + 2\gamma^2\sigma_{\ell+1}^2(\mathbf{A}).
\end{aligned}$$

The second equation is due to $(\mathbf{A} - \mathbf{A}_k)\mathbf{A}_k^\top = 0$ and $\mathbf{Q}_2^L(\mathbf{Q}_2^L)^\top(\mathbf{I} - \mathbf{Q}_2^L(\mathbf{Q}_2^L)^\top)^\top = 0$.
According to Theorem 3.4 in (Gu, 2015), we can get

$$\|\widetilde{\mathbf{A}} - (\widehat{\mathbf{L}}\mathbf{M}\widehat{\mathbf{U}})_k\|_2^2 \leq \sigma_{k+1}^2(\mathbf{A}) + 2\gamma^2\sigma_{\ell+1}^2(\mathbf{A}).$$

Using Equation 5, we can get the conclusion of Theorem 3.

G Proof of Theorem 4

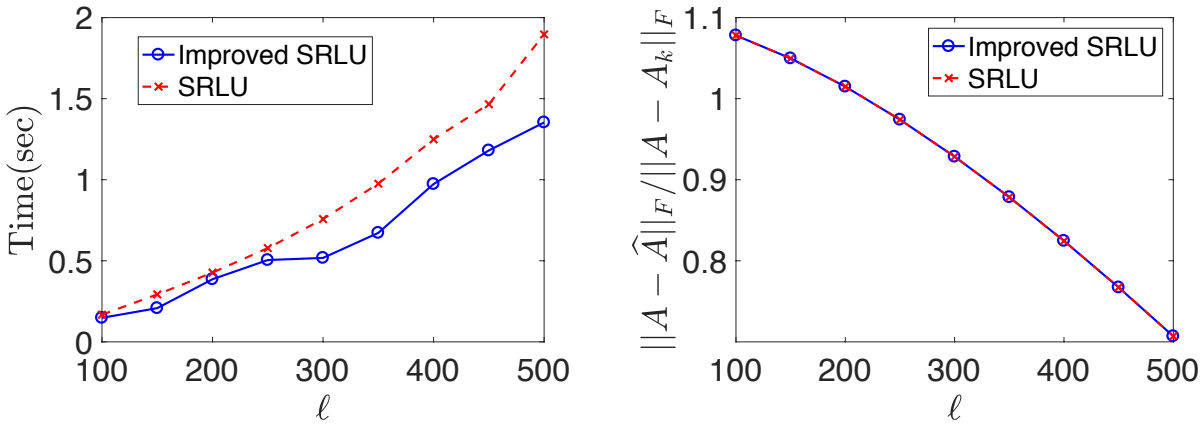
According to Equation 5, we have

$$\widehat{\mathbf{L}}\mathbf{M}\widehat{\mathbf{U}} = \mathbf{\Pi}_1\mathbf{C}\mathbf{U}\mathbf{R}\mathbf{\Pi}_2.$$

Thus, $\sigma_k(\mathbf{CUR}) = \sigma_k(\widehat{\mathbf{L}}\mathbf{M}\widehat{\mathbf{U}})$. Using Theorem 7 in (Anderson and Gu, 2017), we can get the conclusion of Theorem 4.

H Experiments of Improved SRLU

In this section, We compare our Improved SRLU with original SRLU (Anderson and Gu, 2017). Figure 6 shows that Improved SRLU has almost the same approximation errors as SRLU, but Improved SRLU is more efficient.



(a) Time

(b) Frobenius Norm Error

Figure 6: The comparison of Improved SRLU and SRLU on a random 2000×2000 matrix.