Empirical Study of Randomized Algorithms for Low-rank Approximation

Shuncheng Yuan

June 4, 2022

Advisor: Ioana Dumitriu

Abstract

In 2011, Halko, Martinsson, and Tropp published a survey presenting the theory behind and performance of fast randomized algorithms for low-rank matrix decomposition. The current paper discusses an empirical study to evaluate their work, including checking error bounds and exploring possibility to strengthen the bounds, testing failure probabilities, and finding factors that empirically influence the quality of algorithm but are not mentioned in their paper.

1 Introduction

Data science developed extremely fast in the last few decades, and large matrices appear more and more frequently. Not only are those matrices large, but they are usually dense. Traditional deterministic tools, like $O(n^3)$ algorithms, will be inefficient to deal with large and dense matrices. However, those matrices usually have low numerical rank. This means that columns or rows of those matrices are numerically dependent, or more rigorously, those matrices have a few dominant singular values, and the rest of singular values are very small. With this important feature, randomized algorithms are one key to resolve the inefficiency issue. For example, Principal Component Analysis (PCA) is a very popular data analysis method these days. It is used to reduce the dimension of data set. If the design matrix $X \in M^{m\times n}$ has singular value decomposition $X = U\Sigma V^*$, then columns of $U\Sigma$ are call principal components, and U is used to project data onto a much smaller dimensional affine space [8]. Usually, the design matrix is large dimensional but with low numerical rank. Cost of traditional algorithm for SVD is generally cubic, meaning the cost is $O(n^3)$ for square matrices, but if a randomized algorithm is used, then the cost is $O(mnk)$ or even $O(mn\,log(k))$, where k is the numerical rank.

The idea is that if a matrix $A \in M^{m\times n}$ has low numerical rank, then it can be approximately factored into 2 smaller matrices

$$A \approx B \times C,$$

where $B \in M^{m\times k}$ and $C \in M^{k\times n}$, $k \ll min\{m, n\}$.

One way to do this is to smartly extract k columns of A and form a smaller matrix $Q \in M^{m\times k}$ whose range is approximately range of A, and then form QQ^*A. In essence, we are looking for a projection that projects $range(A)$ to an approximate range. Now Q can be recognized as B above, and Q^*A can be recognized as C above.

We can do matrix decomposition more quickly given this factorization. For example, if we want to find an SVD of $A \approx QQ^*A$, we can do SVD on a smaller matrix. This is faster because each operation is on small matrices. Here is a way to approximate the SVD of A given the factorization $A \approx B \times C$

- Form $Y = Q^*A$
- Do SVD for Y: $\tilde{U}\Sigma V^* = Y$
- Form $U = Q\tilde{U}$

This gives an approximate SVD of $A \approx U\Sigma V^*$.

The way to form such a matrix Q is how randomness comes into the place. The idea is to "scramble" columns of A randomly and then sample those scrambled columns. In other words, we are using a random matrix Ω and a random sampling matrix R to form a sampled matrix Y in the following way

$$Y = \underbrace{A \times \Omega}_{scrambling} \times \underbrace{R}_{sampling}.$$

This "scrambling" helps retain most of the information. Then we use orthogonalization to get a matrix with orthogonal columns that have the same range as the sampled matrix. Using such Q will provide a sufficiently good approximation, i.e.

$$||A - QQ^*A|| \leq \epsilon \quad \text{or} \quad ||A - QQ^*A||_F \leq \epsilon$$

In short, the idea is as following.

- Form an $m \times l$ random matrix Ω
- Form $Y = A\Omega$
- Find orthonormal Q with same range as Y. One way is QR-decomposition $QR = Y$

There are many choices we can make that create variant of the algorithm here. Usually, we can choose the random matrix as a standard Gaussian matrix, this makes error analysis has been very tractable. We can also use other random matrices such as SRFT which can make algorithm faster than Gaussian with price of larger error and harder error analysis with much fewer random bits. the above Algorithm requires us to know the numerical rank in advance, and this does not usually happen. More often, we are facing a fixed precision problem. In this case, we need to sample more columns and check the error smartly. When singular values do not decay fast enough, error will be significant, and in this case we may use a power method to reduce error. There are also ways to increase the speed further if the matrix has special structure, such as being Hermitian or positive semi-definite. When the matrix is extremely exceedingly large, there is also a single-pass algorithm that avoids revisiting the input matrix; this feature is known as "minimizing communication".

Although there are many variants, I will present only some of them, because the primary goal of this paper is to check, using experiments, the error bounds; analyze the result of experiments; and verify some unproved claims found in a survey by Halko, Martinsson, and Tropp [4] who collected the randomized algorithms from many sources. Thus, most part of the contents before experiments are from them, and I will reorganize and reinterpret the way they present the theory.

2 Preliminary

2.1 Notations

I will introduce most of the notation used in the paper here, and some of the definitions will be introduced later.

- $||x|| = \sqrt{\sum_{i=1}^{n} x_i^2}$ is 2-norm of a vector $x \in \mathbb{R}$.
- $||A|| = \max_{x \neq 0} \frac{||Ax||}{||x||}$ is induced 2-norm or spectral norm of a matrix.
- $||A||_F = \sqrt{\sum_{ij} a_{ij}^2}$ is Frobenius norm of a matrix.
- A^* denotes the Hermitian conjugate of matrix A. That means $a_{ij}^* = \bar{a_{ji}}$.
- $A^\dagger$ is the pseudo-inverse of matrix A.
- P_Y is an orthoprojector onto space $range(Y)$.
- I_m is an $m \times m$ identity matrix; m will be removed if there is no confusion.

2.2 Definitions and Properties

2.3 Orthonormal matrices

An orthonormal matrix A is a matrix with orthonormal columns (i.e. $A^*A = \mathrm{I}$). In addition, if the matrix is a square matrix, then it is called an orthogonal matrix. An orthogonal matrix satisfies $A^*A = AA^* = I$

2.3.1 Standard Gaussian matrices

A standard Gaussian matrix Ω is a matrix with *i.i.d* entries following standard normal distribution $N(0,1)$. Gaussian matrices are invariant under rotation. This means if U and V are orthonormal matrices, then $U^*\Omega V$ is still a standard normal matrix. Gaussian matrices also have the following properties:

$$\left(\mathbb{E}||S\Omega T||_F^2\right)^{\frac{1}{2}} = ||S||_F||T||_F, \tag{1}$$

and

$$\mathbb{E}||S\Omega T|| \leq ||S||||T||_F + ||S||_F||T||, \tag{2}$$

where S and T are real matrices. The first one is easy to prove using basic linear algebra. The work of the second one is done by Gordon [3, 2].

2.3.2 Projectors

A matrix P is a projector if it satisfies $P^2 = P$. In addition, if it satisfies $P = P^*$, then it is called an orthoprojector. If a matrix Q is orthonormal, then QQ^* is an orthoprojector whose range is the same as $range(Q)$.

2.3.3 Singular Value Decomposition

Given an $m \times n$ matrix A, we can always decompose it in the form $A = U\Sigma V^*$, where $U \in M^{m\times m}$ and $V \in M^{n\times n}$ are orthogonal matrices, and Σ has non-zero entries $\Sigma_{ii} = \sigma_i$ only on the main diagonal. Such a decomposition is called a singular value decomposition, or SVD, and $\sigma_1 \geq \sigma_2 \geq ...$ are called singular values. Two key facts are that

$$||A|| = \sigma_1 \text{ and } ||A||_F = \sqrt{\sum \sigma_i^2}.$$

Singular values can be used to describe the optimal error using rank-k approximations. In other words, if $rank(A) = r$, then for all $k \leq r$, we have

$$\inf_{rank(B)\leq k} ||A - B|| = \sigma_{k+1}$$

and

$$\inf_{rank(B)\leq k} ||A - B||_F = \sqrt{\sigma_{k+1}^2 + ... + \sigma_r^2}.$$

2.3.4 Pseudo-inverse

If $A \in M^{m\times n}$ has SVD $A = U\Sigma V^*$, then the pseudo-inverse of A can be defined by $A^\dagger = V\Sigma^\dagger U^*$, where $\Sigma^\dagger$ is an $n \times m$ matrix with non-zero entries $\Sigma_{ii}^\dagger = \frac{1}{\sigma_i}$ only on the main diagonal.

3 Algorithms

All the algorithms we present here have two stages; stage one gives an orthonormal matrix Q such that $||A - QQ^*A||$ is small, while stage two uses the ideas in the Introduction to compute matrix factorization such as the SVD we see in the Introduction.

Stage One

There are many ways of sampling to get such a matrix Q. One of the simplest cases is when we know the numerical rank k of the matrix $A \in M^{m\times n}$ in advance, or when the numerical rank is required to be a fixed number in a problem, and generate randomness using a Gaussian matrix. In this case, we will generate a $n \times (k + p)$ Gaussian matrix Ω and left multiply it with the matrix A to create both the "scrambling" effect and the sampling. Here p is an oversampling parameter which reduces error and probability of failure of probabilistic error bounds without adding too much computational work. Usually, this parameter is chosen to be $p = 5$ or $p = 10$, but in general the parameter can be larger if the dimension of the matrix is larger, or the speed of decay of spectrum is slower, or the random matrix involves less randomness to improve error. This is intuitive but will be discussed more formally in the error analysis part.

Algorithm 1 Find an orthonoamal Q such that $||A - QQ^*A||$ is small

```
Given an $m \times n$ matrix $A$ with targeted rank $k$, output an $m \times (k + p)$ orthonormal matrix $Q$ such
that $||A - QQ^*A||$ is small, where $p$ is an oversampling parameter.
Generate an $n \times (k + p)$ standard Gaussian matrix $\Omega$
$Y = A\Omega$
$QR = Y$
output $Q$
```

In many cases, we do not know the rank in advance or want to deal with a fixed precision problem, and we do not know how much sampling effect we want. Fortunately, there is a cheap way to have a rough estimation of error, which is guaranteed by the following lemma. The proof of this lemma can be found in Woolfe, Liberty, Rokhlin and Tygert's [9].

Lemma. *Suppose B is an $m \times n$ real matrix, r is a positive integer and $\alpha \in (1, +\infty]$. If $\omega_1, \omega_2, ..., \omega_r$ are i.i.d. following $N(0, I_n)$ then*

$$P\left(||B|| \leq \alpha\sqrt{\frac{2}{\pi}} \max_{i=1,...,r} ||B\omega_i||\right) \geq 1 - \alpha^{-r}$$

If we take $B = A - QQ^*A$, and $\alpha = 10$, we have

$$\epsilon = ||A - QQ^*A|| \leq 10\sqrt{\frac{2}{\pi}} \max_{i=1,\ldots,r} ||(I - QQ^*)A\omega_i||$$

with probability at least $1 - 10^{-r}$. This is a quick way to check the error because computing vector norms is much quicker than computing the spectral norm of matrices. With this speedy way, we can start with q being a small number in above algorithm and then increase it, if the error is not satisfactory. It may also happen that we are given a small numerical rank but the spectrum decays slowly. In this case we can still increase oversampling parameter to improve error, but there is an iterative method that can quickly lower the error. If $A = U\Sigma V^*$, then let

$$B = (AA^*)^q A = (U\Sigma^{2q}U^*)U\Sigma V^* = U\Sigma^{2q+1}V^*;$$

B has the same singular vectors as A does, but the singular values decay geometrically faster relative to the larger one. The idea of the iterative method is to apply the previous algorithm to B.

Algorithm 2 Improve error in algorithm 1 using power method

```
Given an $m \times n$ matrix $A$ with targeted rank $k$, output an $m \times (k+p)$ orthonormal matrix $Q$ such that $||A - QQ^*A||$ is small, where $p$ is an oversampling parameter. $q$ is a positive integer.
Generate an $n \times (k+p)$ standard Gaussian matrix $\Omega$
$Y = (AA^*)^q A\Omega$ using alternating application.
$QR = Y$
output $Q$
```

The cost of the first algorithm is in general $O(mnl)$. This is because cost of forming $Y = A\Omega$ is $O(mnl)$, and this step dominates the cost in the algorithm. If we change the way we generate the random matrix Ω, we can reduce the cost to $O(mn\,log(l))$ for which one might theoretically pay a price of losing some accuracy. One alternative way to generate Ω is subsampled random Fourier transform, or SRFT. An $n \times l$ SRFT Ω has form

$$\Omega = \sqrt{\frac{n}{l}}DFR.$$

D is an $n \times n$ diagonal matrix whose entries are $exp(iU_j)$ where $U_1, ..., U_n$ are i.i.d $Unif(0, 2\pi)$, which uniformly sample points on unit circle and provides random initial phase. F is an $n \times n$ whose entries are $F_{pq} = \frac{1}{\sqrt{n}}exp(-2\pi i(p-1)(q-1)/n)$ for $p, q = 1, ..., n$, which is a unitary discrete Fourier transform. R is an $n \times l$ matrix that uniformly samples l columns from an $n \times n$ identity matrix.

With this special way to generate randomness, we can apply Fast Fourier Transform (FFT), which computes Discrete Fourier Transform faster and ultimately decreases the cost of forming $Y = A\Omega$ in the algorithm. This idea is from [7]. There are other ways to generate Ω, such as using random Givens rotations and subsampled

Algorithm 3 Speed up Algorithm 1 using SRFT

```
Given an $m \times n$ matrix $A$ with targeted rank $k$, output an $m \times (k+p)$ orthonormal matrix $Q$ such that $||A - QQ^*A||$ is small, where $p$ is an oversampling parameter.
Generate an $n \times (k+p)$ SRFT matrix $\Omega$
$Y = A\Omega$ using FFT
$QR = Y$
output $Q$
```

Hadamard transforms, but only the above SRFT method will be empirically tested in this paper.

Stage Two

In stage one, we get an $n \times l$ orthonormal matrix Q with small l and

$$||A - QQ^*A|| \leq \epsilon. \tag{3}$$

With this Q, we have a low-rank approximations $A = BC$ is we take $B = Q$ and $C = Q^*A$. Doing SVD on these smaller matrices is much cheaper. This algorithm gives an SVD that has the same number of positive singular values as the rank of Q. If we need a rank k that is smaller than this number, we can simply keep the first k singular values and zero out the rest in Σ. This operation only increases the error by σ_{k+1}. Doing PCA is one example where we need to do this. Again, the content in stage 2 is rich. For example, when A is Hermitian, there is a way to speed up the algorithm more with a little sacrifice of error, and a single-pass algorithm also appears here to relieve the pressure of fast memory. Because the main focus is to test and analyze randomized methods in stage 1 using experiments, only the above algorithm will be introduced and used later in the experiments.

Algorithm 4 Find an approximate SVD of A

```
Given matrices $A$ and $Q$ satisfying (3.1), Compute the SVD of $A$
$B = Q^*A$
$\tilde{U}\Sigma V^* = Y$
$U = Q\tilde{U}$
output $U\Sigma V^* \approx A$
```

4 Error Analysis

This section is essentially based on Halko, Martinsson, and Tropp's work [4]; I will reorganize the ideas in order to conveniently analyze error bounds later. In this section, only error bounds for algorithm using a Gaussian matrix will be discussed thoroughly. There are some results of bounds using SRFT, but the conditions significantly restrict the cases where we can use it. We will discuss the upper bound for expected error. Then, to remove concerns about large deviation, I will present probabilistic error bounds.

The error bounds for algorithms 1 and 2 relies on the following general theorem introduced by Halko, Martinsson, and Tropp [4]. Detailed proof is also provided in their paper. It is technical, there is not much room to analyze it in combination with experiments, so it will be skipped in this paper.

Theorem 1. *Define $Y := A\Omega$, where A is an $m \times n$ matrix with SVD $A = U\Sigma V^*$, and Ω is an $n \times l$ test matrix. For a fixed natural number k, partition Σ and V in a following way*

$$\Sigma V^* = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^* \\ V_2^* \end{bmatrix}$$

where Σ_1 is $k \times k$ and V_1^* is $k \times n$. Let $\Omega_1 = V_1^*\Omega$ and $\Omega_2 = V_2^*\Omega$. If Ω_1 has full rank, then

$$||(I - P_Y A)||^2 \leq ||\Sigma_2||^2 + ||\Sigma_2\Omega_2\Omega_1^\dagger||^2. \tag{4}$$

We know that $||\Sigma_2|| = \sigma_{k+1}$, which is optimal error for rank k approximation, so to find the bounds, we need to find upper bounds of pseudo-inverse of a Gaussian matrix.

Theorem 2. *Let Ω be a $k \times (k + p)$ standard Gaussian matrix with $k \geq 2$ and $p \geq 2$. Then we have*

$$\mathbb{E}[||G^\dagger||_F^2] = \frac{k}{p-1} \tag{5}$$

and

$$\mathbb{E}[||G^\dagger||] \leq \frac{e\sqrt{k+p}}{p} \tag{6}$$

This is an average bound. Proof of the first equality can be found in [6], and second one can be found in [1]. Now we have tools to develop the expected error bound.

Theorem 3. *Let A be an $m \times n$ matrix with singular values $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq ...$, and let $Y = A\Omega$, where Ω is an $n \times (k + p)$ standard Gaussian matrix with $k, p \geq 2$ and $k + p \leq min\{m, n\}$. Then*

$$\mathbb{E}||(I - P_Y)A||_F \leq \sqrt{1 + \frac{k}{p-1}} \left(\sum_{j>k} \sigma_j^2 \right)^{\frac{1}{2}} \tag{7}$$

Proof. Keeping the notation in Theorem 1, since V^* has orthonormal rows, and Ω is a standard Gaussian matrix, ΩV^* is also a standard Gaussian matrix. As ΩV_1^* and ΩV_2^* are blocks of $V^*\Omega$, they are independent and standard Gaussian matrices. We have

$$\mathbb{E}||(I - P_Y)A||_F \leq \left(\mathbb{E}||(I - P_Y)A||_F^2\right)^{\frac{1}{2}} \leq \left(||\Sigma_2||^2 + ||\Sigma_2\Omega_2\Omega_1^\dagger||^2\right)^{\frac{1}{2}}$$

The first inequality is Jensen's inequality, and the second one is due to Theorem 1. Then, using a commonly used conditional expectation trick, we have

$$\left(||\Sigma_2||^2 + ||\Sigma_2\Omega_2\Omega_1^\dagger||^2\right) = \mathbb{E}\left(\mathbb{E}\left[||\Sigma_2\Omega_2\Omega_1^\dagger||_F^2|\Omega_1\right]\right) = \mathbb{E}\left(||\Sigma_2||_F^2||\Omega_1^\dagger||_F^2\right)$$

$$= ||\Sigma_2||_F^2\mathbb{E}||\Omega_1^\dagger||_F^2 = \frac{k}{p-1}||\Sigma_2||_F^2,$$

The second equality is a property of standard Gaussian matrice. See equation (2.1), and the last equality is equation (4.2). As $||\Sigma_2||_F^2 = \sum_{j>k} \sigma_j^2$, doing some algebra completes the proof. □

Again, the proof is from Halko, Martinsson, and Tropp's paper, this is introduced because it will be analyzed later. Similarly, we have error bound for spectral error.

Theorem 4. *With the same setting in Theorem 3, we have*

$$\mathbb{E}||(I-P_Y)A|| \leq \left(1+\sqrt{\frac{k}{p-1}}\right)\sigma_{k+1} + \frac{e\sqrt{k+p}}{p}\left(\sum_{j>k}\sigma_j^2\right)^{\frac{1}{2}} \tag{8}$$

The proof of this theorem is similar to above on. The difference it uses equation (2.2) and both equations in Theorem 2. If we want to find probabilistic error bounds for algorithm 4.1, we also need probabilistic error bound for the pseudo-inverse of a Gaussian matrix. The proof of the following theorem can be found in [6, 5]. The first bound is slack and will be one of the reasons that the probabilistic error are loose.

Theorem 5. *Let Ω be a $k \times (k+p)$ standard Gaussian matrix with $p \geq 4$. Then, for all $t \geq 1$ we have*

$$\mathbb{P}\left(||G^\dagger||_F] \geq \frac{12k}{p}t\right) \leq 4t^{-p} \tag{9}$$

and

$$\mathbb{P}\left(||G^\dagger|| \geq \frac{e\sqrt{k+p}}{p+1}t\right) \leq t^{-(p+1)} \tag{10}$$

The following theorems describe probabilistic error bound. Details of proof will be skipped. The idea is the same as proving expected error bounds: using some conditioning tricks and additionally applying concentration for functions of Gaussian matrices.

Theorem 6. *With the same setting in Theorem 3, assume $p \geq 4$ and $s, t \geq 1$. Then*

$$||(I-P_Y)A||_F \leq \left(1+t\sqrt{\frac{12k}{p}}\right)\left(\sum_{j>k}\sigma_j^2\right)^{\frac{1}{2}} + ut\frac{e\sqrt{k+p}}{p+1}\sigma_{k+1} \tag{11}$$

with failure probability at most $5t^{-p} + 2e^{-u^2/2}$.

Theorem 7. *With the same setting in Theorem 3, assume $p \geq 4$ and $s, t \geq 1$. Then*

$$||(I-P_Y)A|| \leq \left[\left(1+t\sqrt{\frac{12k}{p}}\right)\sigma_{k+1} + t\frac{e\sqrt{k+p}}{p+1}\sqrt{\sum_{j>k}\sigma_j^2}\right] + ut\frac{e\sqrt{k+p}}{p+1}\sigma_{k+1}. \tag{12}$$

with failure probability at most $5t^{-p} + e^{-u^2/2}$

In above two bounds, The term in square bracket on the right is like the expectation, while $ute\sqrt{k+p}\sigma_{k+1}/(p+1)$ is like deviation. To use a probabilistic bound, we can set $t = e$ and $u = \sqrt{2p}$ or any other reasonable choices. This choice will be applied later in the experiment.

There are also bounds for the power scheme.

Theorem 8. *With the same setting in Theorem 3, let $B = (AA^*)^q A$ for some natural number q and $Z = B\Omega$. Then*

$$\mathbb{E}||(I-P_Z)A|| \leq \left[\left(1+\sqrt{\frac{k}{p-1}}\right)\sigma_{k+1}^{2q+1} + \frac{e\sqrt{k+p}}{p}\left(\sum_{j>k}\sigma_j^{4q+2}\right)^{\frac{1}{2}}\right]^{\frac{1}{2q+1}} \tag{13}$$

If we enlarge this error and the error in Theorem 3, we can see that the power scheme reduces the constant factor in front of the optimal bound exponentially fast. The error bound above is strengthened to

$$\mathbb{E}||(I-P_Z)A|| \leq \left(1+\sqrt{\frac{k}{p-1}} + \frac{e\sqrt{k+p}}{p}\sqrt{min\{m,n\}-k}\right)^{\frac{1}{2q+1}}\sigma_{k+1},$$

and the error bound in Theorem 3 can be strengthened to

$$\mathbb{E}||(I-P_Y)A|| \leq \left(1+\sqrt{\frac{k}{p-1}} + \frac{e\sqrt{k+p}}{p}\sqrt{min\{m,n\}-k}\right)\sigma_{k+1}.$$

There is an error analysis of algorithm using SRFT as well, but it is not as well studied as the error using Gaussian matrices.

Theorem 9. *Let A be an $m \times n$ matrix with singular values $\sigma_1 \geq \sigma_2 \geq \ldots$. Let Ω be an $n \times l$ SRFT matrix satisfying*

$$4\left(\sqrt{k}+\sqrt{8log(kn)}\right]^2 log(k) \leq l \leq n,$$

and construct $Y = A\Omega$. Then

$$||(I - P_Y)A|| \leq \sqrt{1 + 7n/l}\sigma_{k+1} \tag{14}$$

$$||(I - P_Y)A||_F \leq \sqrt{1 + 7n/l}\left(\sum_{j>k}\sigma_j^2\right)^{\frac{1}{2}} \tag{15}$$

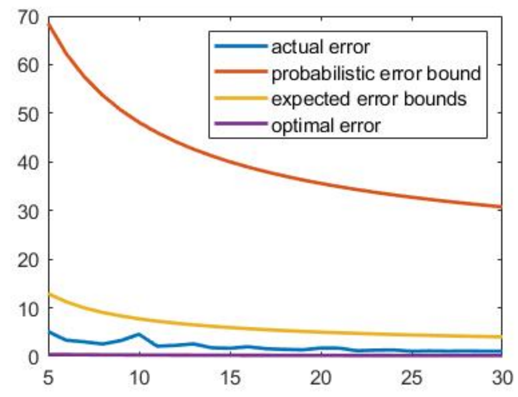
with failure probability at most $O(k^{-1})$

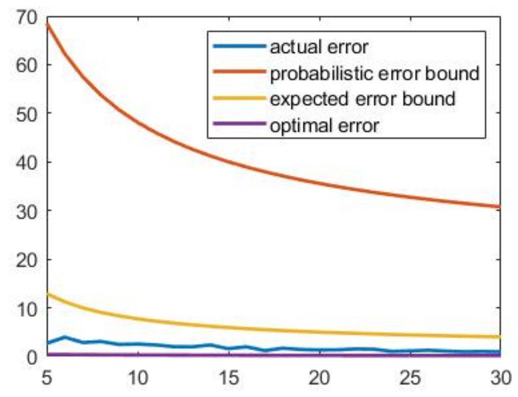
The conditions in the above theorem are very restrictive. When the number of the columns is 1000, then the number k cannot be larger than 11. Therefore, this error bound in most of cases cannot be applied. However, as we will see in the experiment, although less randomness means higher error, the error is not so large compared with well-studied error using Gaussian.

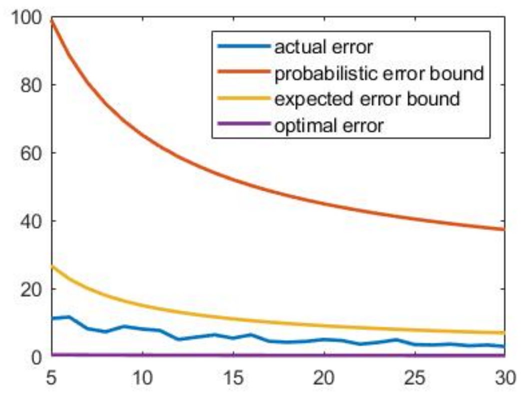
5 Experiments and Analysis

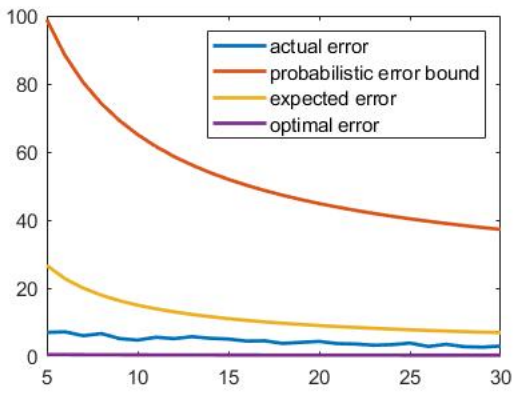
5.1 Testing Error Bounds Using Algorithm 1

As we saw in the error bounds using Gaussian matrix above, the approximation error significantly depends on the speed of decay of singular values. If the singular values decay fast enough, σ_{k+1} and k are small. Ideally, if the decay is so quick that there is a big fall or gap in singular values, then the error is expected to be small. Also, if the spectrum decays fast, $\sum_{j>k}\sigma_j^2$ will be small as well. Taking this into consideration, there are six 3000×3000 matrices in the first experiment. Three of them have 30 relatively large singular values $39, 38, ..., 10$; the other three have 30 smaller singular values $31, 30, ..., 2$. In each group of three matrices, the tail singular values follow sequences $a_n = (n^{-0.5})_{n\geq1}$, $b_n = ((log(n+1))^{-1})_{n\geq1}$, and $c_n = (log((log(n+10)))^{-1})_{n\geq1}$ respectively. In other words, all matrices have small numerical rank, and there is a large gap in singular values in three matrices; in other 3 matrices, the gap is very small, and there are three levels of decay of singular values. The first experiment aims to check if the theoretical error bounds are tight.









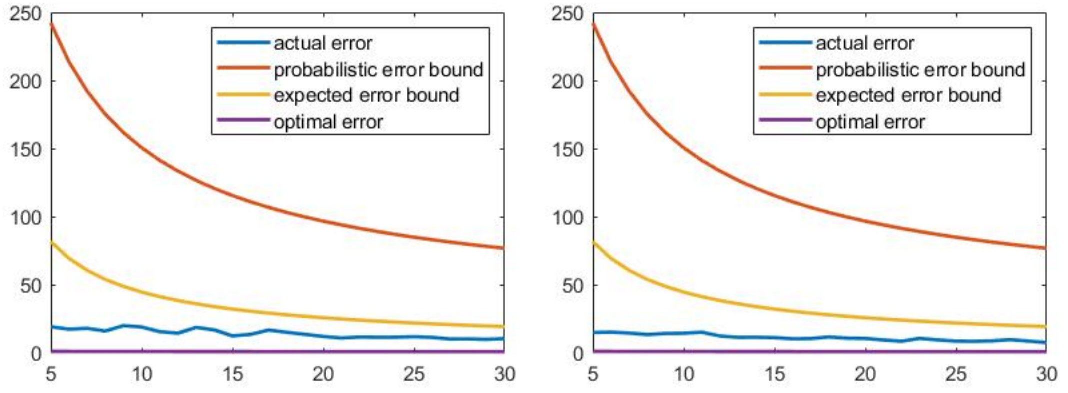
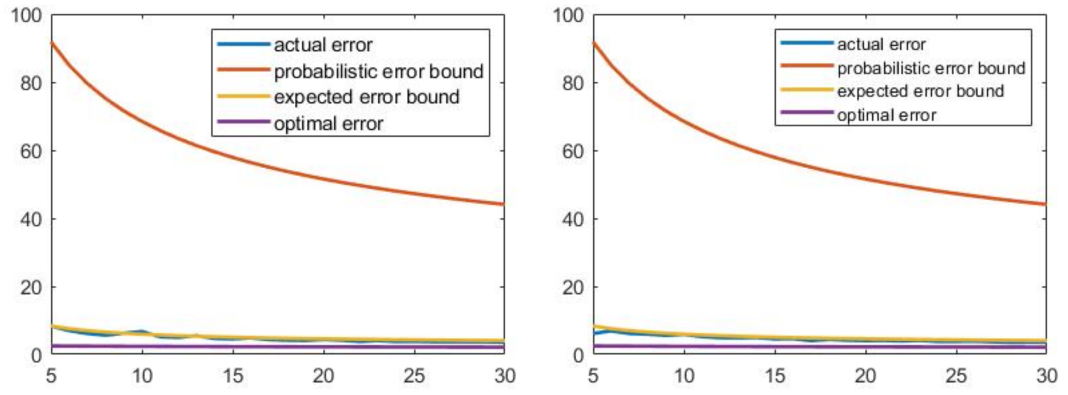
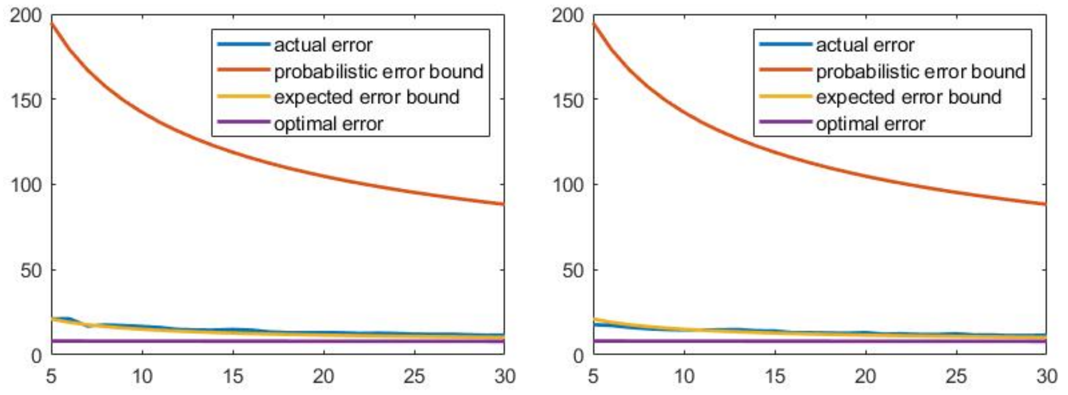


Figure 1: The horizontal axis represents the oversampling parameter p, and the vertical axis represents errors in the spectral norm. Matrices on the left have large gap in singular values. Tail singular values of matrices in the first/second/third row follow $a_n/b_n/c_n$.





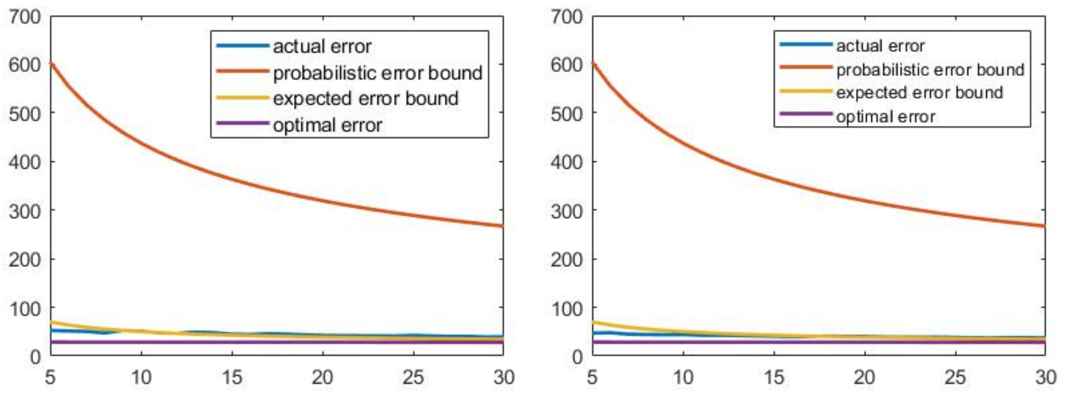


Figure 2: The horizontal axis represents the oversampling parameter p, and the vertical axis represents errors in the Frobenius norm. Matrices on the left have large gap in singular values. Tail singular values of matrices in the first/second/third row follow $a_n/b_n/c_n$.

From these graphs, the first observation is that the probabilistic error bound is very loose. It is natural to derive loose probabilistic error bound, because deviation need to be considered, and a sufficiently small failure probability. However, we can see that the gap between expected error bound and probabilistic error bound, which relates to the deviation, is too large compared with how much the error can vary. Certainly, this experiment only sampled once in each matrix, so when the sampling parameter is small, it is totally possible that the error goes up and approach the bound. We will see later that this does not happen at least in these matrices.

Another observation is that, the expected error bound is quite tight, especially if we measure the error using Frobenius norm. If we look at the proof of Theorem 3, which gives the expected error bound in the Frobenius norm, there are only two inequalities; both equation (1) and (5) are equalities. One inequality is Jensen's inequality, and the other is due to Theorem 1. These indicate that the inequality in the Theorem 1 in the last section is very tight, because Jensen's inequality can be tight if the random variable is not convex enough. Moreover, this means that the bounds in Theorem 5 are loose, so the probabilistic error bound can be improved once we have better understanding of probabilistic bounds of norms of pseudo-inverse of Gaussian matrix. But expected bounds seems reliable as well, as we test failure probability of probabilistic error bound later, we will also check how likely is the error goes above expected error bounds.

Also the bounds well predicted the factors that influence the error. If we look through these plots vertically, we can see that the rate of decay of tail singular values significantly influences the error, even in spectral norm. In spectral norm, the best error is always σ_{k+1} which is independent of the whole tail singular values. This means the slower the spectrum decays, the further the error is away from the optimal error. Horizontally, we see the error seem not significantly depend on how large the gap is. But the method still applies better with large gaps. One can quickly realize that with similar approximation error, smaller dominant singular values is more sensitive to errors, which results in inaccurate matrix decomposition in stage II. We will see that the power method perfectly fixes this problem without too much additional cost.

5.2 Testing the Reliability of Power Scheme

In last section, we have seen that the the slower the singular values decays, the larger the error is. The best rank k approximation generate error σ_{k+1}. If we denote the error as $C\sigma_{k+1}$, the factor C could be roughly 20 in matrix with lowest decay. Compared with the dominant singular values, this is a terrible number. This can be seen by simply checking the approximate singular values.

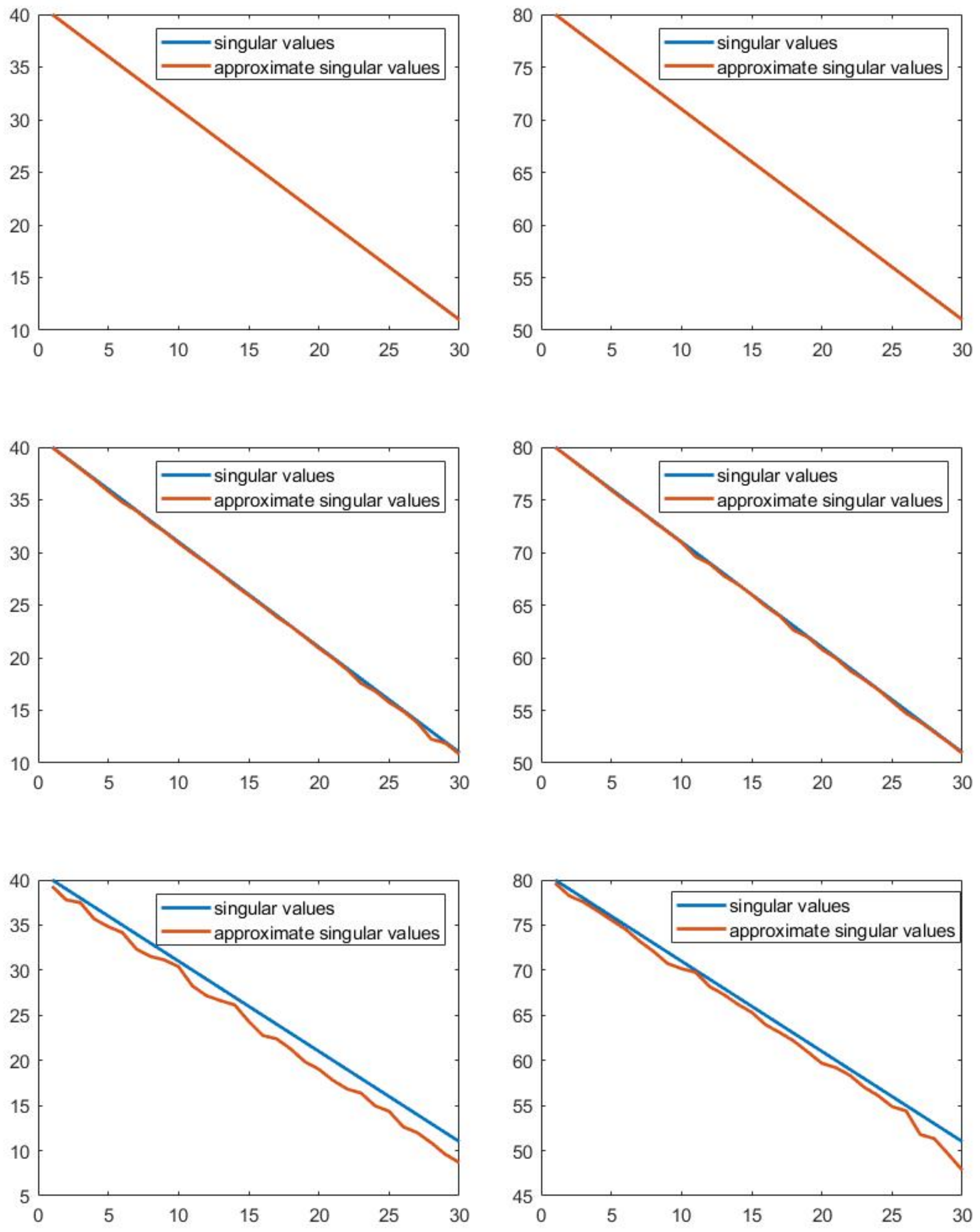


Figure 3: The horizontal axis represents the order of singular values, and the vertical axis represents singular values. Tail singular values of matrices in the first/second/third row follow $a_n/b_n/c_n$. The oversampling parameter is 5 in this experiment.

We see the errors of approximating dominant singular values are sensitive to the quality of projector Q. In case the quality of Q is low, effectiveness of power scheme must be guaranteed to improve the error. The next experiment will be checking how well the power method work for the matrices designed above.

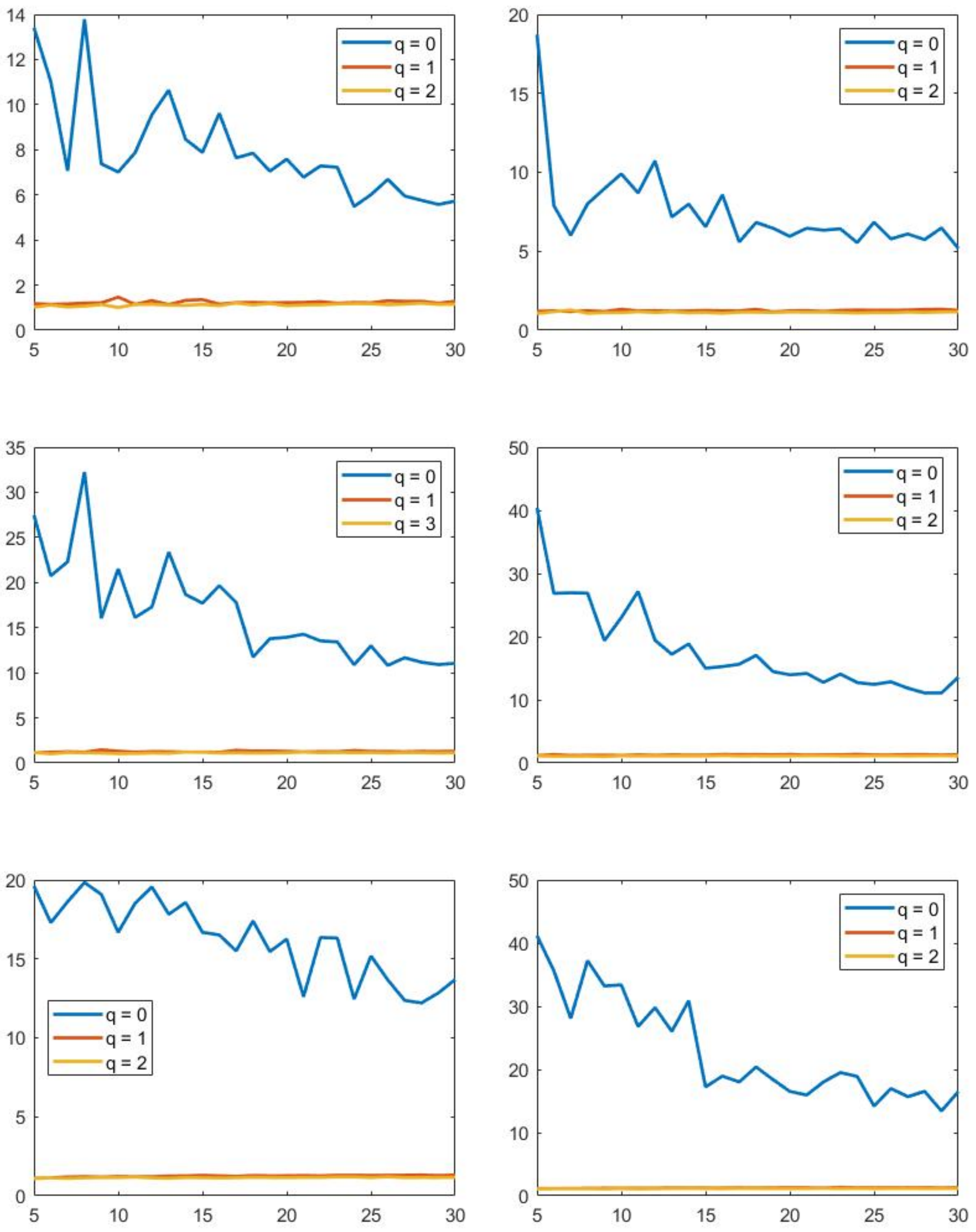


Figure 4: The horizontal axis represents the oversampling parameter p, and the vertical axis represents the error in spectral norm. Dominant singular values are the same as those in Figure 5.3 in the same position. Tail singular values of matrices in the first/second/third row follow $a_n/b_n/c_n$. The oversampling parameter is 5 in this experiment.

From the result, we see that the performance of power methods is very good. Based on the theoretical bound, the factors should be $C^{1/(2q+1)}$, but the experiment showed that even if we use iteration once, the factor suddenly reduces to approximately 1. This phenomenon was predicted by Halko, Martinsson and Tropp. They claimed that if the tail singular values are dominated by $Kn^{\frac{1+\epsilon}{4q+2}}$, for some positive number ϵ and constant K, then the power method will work much better than the bound predicts. Although the rate of decay in above experiment is logarithmic, when the size of the matrices is small, there is not much difference between two rates, especially there is a constant factor in the former rate. We will see later that if we just increase the size of matrices a bit, the case is completely different.

5.3 Testing the Algorithm Using SRFT

The error analysis of algorithm using Gaussian is thoroughly developed, though some bounds are generally loose. One major drawback of using Gaussian is the cost is $O(nmk)$, but the cost using SRFT is $O(nm\, log(k))$ due to the reliance on FFT. If the numerical rank is large enough, using SRFT will be much faster. However, as we see in Halko, Martinsson and Tropp's paper, the error analysis of algorithm using SRFT is not well developed. There is a restrictive condition in the theorem, making the error bounds useless in most cases. Intuitively, we expected the error using SRFT will be much worse than using Gaussian, because there is not much source of randomness in SRFT, so that the "scrambling" effect is of poorer quality. However, the experiments below show that in some cases, the performance are almost the same.

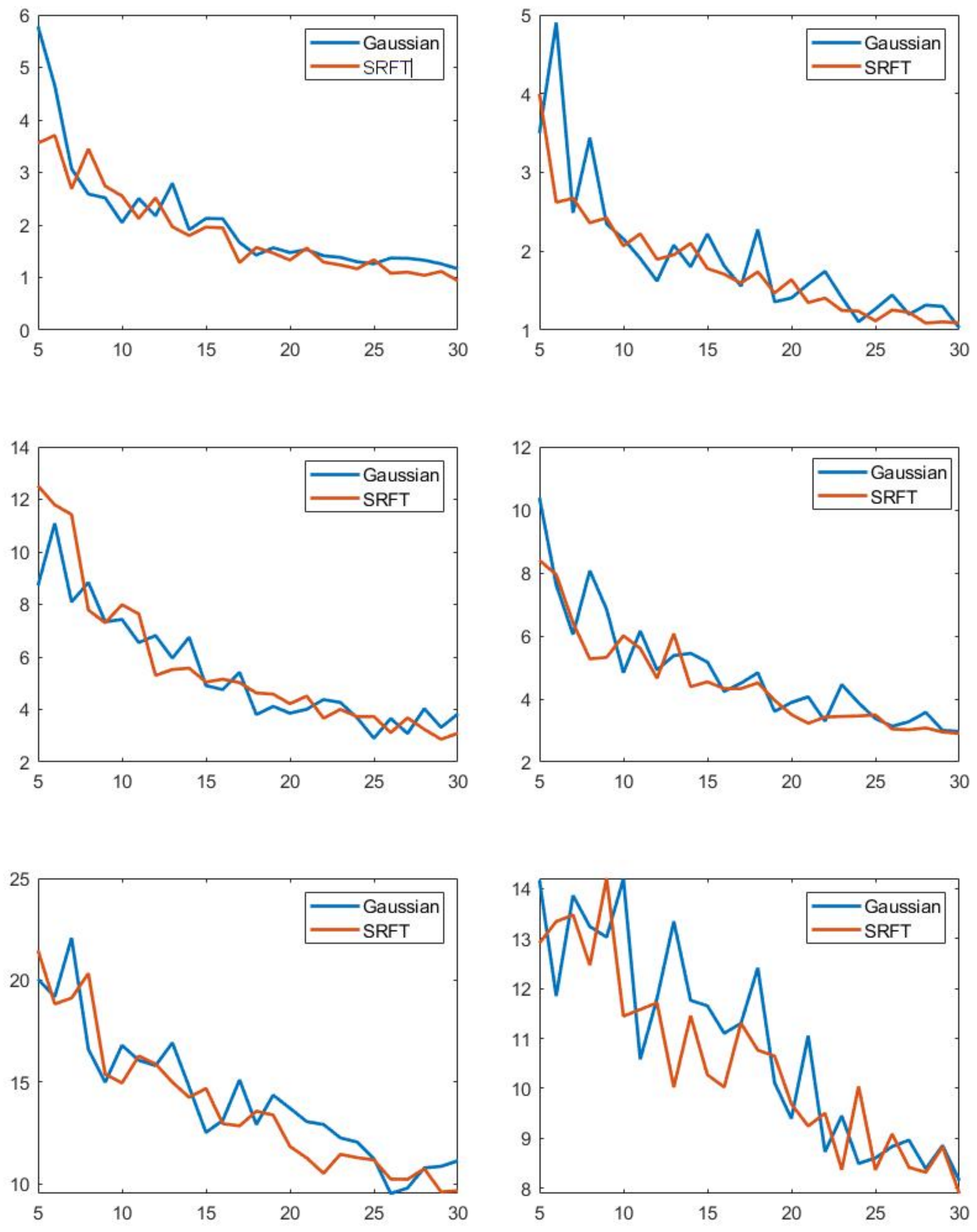


Figure 5: The horizontal axis represents oversampling parameter p, and the vertical axis represents errors in spectral norm. Matrices on the left have large gap in singular values. Tail singular values of matrices in the first/second/third row follow $a_n/b_n/c_n$.

As we can see from the plots, it hard to say one's performance is better than the other. Using the same oversampling the parameter, there is no obvious advantage, but also when the oversampling parameter is increased, the errors decrease at the same rate. At least with these artificially created matrices, SRFT does not generate as bad error as the intuition. It even contradicts Halko, Martinsson and Tropp's claims that when the tail singular values decay slowly, the error will be worse than the one using Gaussian. Although using SRFT is faster, there is no method to quickly reduce error when the decay of spectrum is small. But because this method is faster, there is no harm to add more sampling to reduce the error. As we see from the plots, when the parameter is twice of numerical rank, the error to generate projector Q is quickly approaching the best error we can reach which also happens on Gaussian. Again, this result seems too good to be true, some more experiments finding the factors that have influence on difference of using Gaussian and using SRFT are needed.

5.4 Testing Failure Probability

So far, the experiment testing error bounds only have sample size 1. Given that the failure probability is not negligible when the oversampling parameter is small, one concern is the above result is a coincidence. The following experiment aims to eliminate this concern. In this experiment, the size of matrix will be changed to 1000×1000 due to limited computing power. The dominant and tail singular values are still the same as the first experiments. For each matrix, the sample size is 1000.

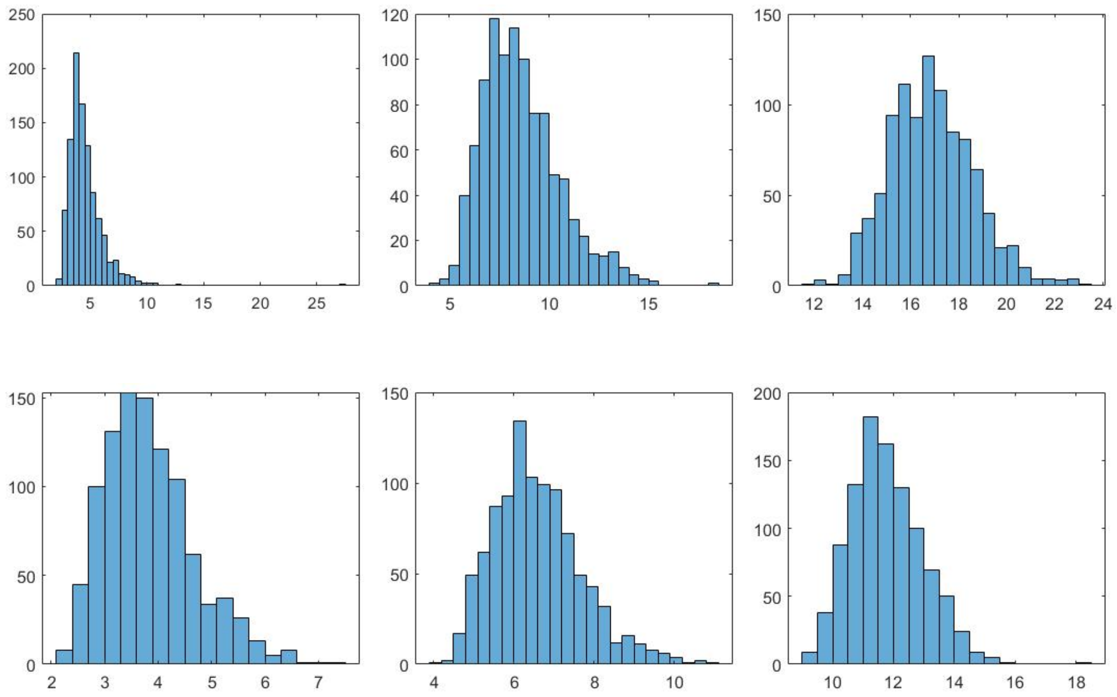


Figure 6: The horizontal axis in the histogram represents error using Gaussian matrix in spectral norm. The matrices in the first row have large gap in singular values. Tail singular values of matrices in the first/second/third columns follow $a_n/b_n/c_n$. The oversampling parameter is 5.

The probabilistic error bounds in three kinds of decay speeds are roughly 70, 100, and 250 respectively. Based on this result, not only are probabilistic error bounds like deterministic error bounds, but also the largest errors are still far away from the error bounds. Errors are so small that if we use expected error bounds as probabilistic error bounds, the failure is a rare event. This reminds us that we can adjust the s, t parameters in the probabilistic error bounds to gain a tighter bound with sacrifice of increasing failure probability, but this is unnecessary as we see expected error bound is reliable enough. Another key observation is that the distribution of error in spectral norm depend on the size of gap in singular value, which is a factor that is not apparent when the sample size is 1 in the first experiments. More importantly, this factor does not appear in the error bounds. Generally, the larger the gap is, the larger errors are. This should not be a big concern especially if we only care about the first few dominant singular values or eigenvalues; the previous experiments showed that if the gap is larger, the singular value/ eigenvalue approximation is less sensitive to the error.

Similar thoughts can also be used as a criteria to check if SRFT is a reliable way to generate randomness. Again, this will be compared with errors using Gaussian, and only the errors to approximate matrices with larger gap will be compared.

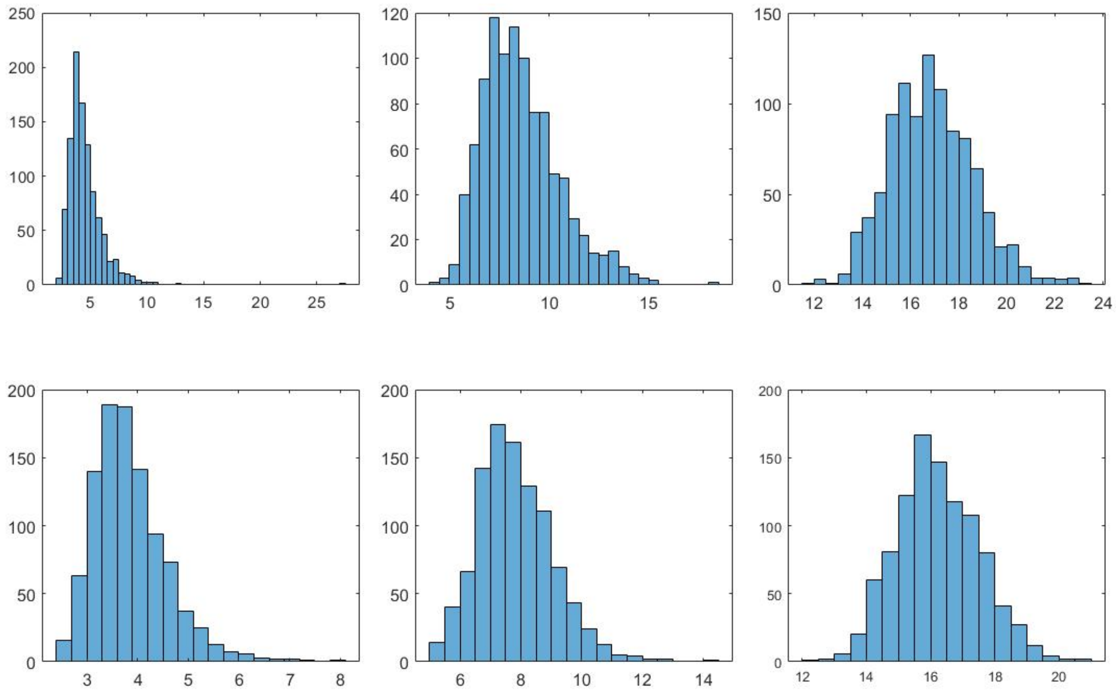


Figure 7: The horizontal axis in the histogram represents error using SRFT in spectral norm. The matrices in the first row have large gap in singular values. Tail singular values of matrices in the first/second/third columns follow $a_n/b_n/c_n$. The oversampling parameter is 5.

These results are counter-intuitive again. With larger sample size, it becomes obvious that errors using SRFT is better in terms of distribution. The mean is smaller, and deviation is not larger.

5.5 Additional Tests

The following experiments aim to find the factors that make the performance of the power scheme and SRFT so good in practice. The first step is to test on slightly bigger matrices. In the following experiment, the size of matrix is 10000×10000; many open source data are smaller than this size. There are 100 dominant singular values $20, 19.9, ..., 10.1$, and the rest of singular values follows sequence c_n. The following table shows error using power scheme with 10 different samples Ω. The oversampling parameter is 5. This time, the power

	q = 0	q = 1	q = 2	optimal error
sample 1	18.2045291573607	8.81984916338680	2.22404059507884	1.01978144053823
sample 2	17.5766175572607	10.0896271606239	2.23975646405150	1.01978144053823
sample 3	17.9070290233242	9.46530246748910	2.23330662991463	1.01978144053823
sample 4	17.7989906000247	7.22263686356695	2.22068840237818	1.01978144053823
sample 5	17.7126720019033	11.2129088770791	2.30702879888204	1.01978144053823
sample 6	17.8398813531354	10.1971565097982	2.26771846434051	1.01978144053823
sample 7	17.9269378161891	10.6925181213061	2.29464240767117	1.01978144053823
sample 8	17.7495443124032	11.6330842078662	2.36175428657884	1.01978144053823
sample 9	17.8392865907637	10.2292210517578	2.26156886696664	1.01978144053823
sample 10	17.6646462740368	9.05874409981834	2.23674566825358	1.01978144053823

scheme does not work as well as the prediction. When $q = 1$, the error is only reduced to half of $q = 0$ (without using power method). This is worse than the predicted rate, but we cannot say the error bound is incorrect because to get $C^{\frac{1}{2q+1}}$ rate, many things are enlarged. Thus, the slower the singular values decay, the more iterations are needed to get desired error. Although the rate is not reliable, taking $q = 2$ or $q = 3$ is empirically sufficient to reduce error significantly as we see in the table, since many data are of smaller size and have thinner tail of singular values.

With the same matrix, the following table contains error using SRFT and Gaussian, each has 10 samples. The oversampling parameter is 5.

Gaussian	SRFT
17.9781251440279	17.7291423458222
18.0677132353478	17.8180454238921
17.5460276235669	17.6874052618001
17.8645388002046	17.4597075148310
17.8350845399760	17.6647404990870
17.9805896358771	17.7604950638704
17.7305460815219	17.6587313040270
17.5888019711260	17.6407536568126
17.9294241358220	17.7888944310882
17.7825540102251	17.7513596729028

We see, even for matrices of this size and this slow rate of decay of tail singular values, which is a factor that nullify SRFT, there is not much difference between Gaussian and SRFT.

Another factor that affect errors is oversampling parameter. The following table contains error using SRFT and Gaussian, each has 10 samples. The oversampling parameter is 400 which is $4k$.

Gaussian	SRFT
11.3369873756133	11.4317303719488
11.1034004275146	11.3161649690591
11.3223803041069	11.3833080297183
11.4950481382659	11.2145445081250
11.3336646821692	11.2563329904522
11.5531933722054	11.2471833527772
11.2572650261345	11.4575613553698
11.2779563357270	11.4341038795775
11.3359152802412	11.4054871484074
11.2430629091554	11.3329353658352

There is not much difference. Certainly, it is not sufficient to conclude that this difference does not depend on the size of matrices and decay rate of spectrum, but given that many data are of this size and the tail is much thinner, the quality of SRFT can be guaranteed, even if the oversampling parameter is chosen to be large to reduce error.

6 Acknowledgment

I would like to thank Professor Ioana Dumitriu, who introduced this theory of randomized algorithms to me, provided guidance of research, and giving me advice of designing experiments.

References

[1] Zizhong Chen and Jack J Dongarra. Condition numbers of Gaussian random matrices. *SIAM Journal on Matrix Analysis and Applications*, 27(3):603–620, 2005.

[2] Yehoram Gordon. Gaussian processes and almost spherical sections of convex bodies.

[3] Yehoram Gordon. Some inequalities for Gaussian processes and applications. *Israel Journal of Mathematics*, 50(4):265–289, 1985.

[4] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[5] Per-Gunnar Martinssona, Vladimir Rokhlinb, and Mark Tygertb. A randomized algorithm for the approximation of matrices. Technical report, Rapport technique, Technical Report 1361, Department of Computer Science . . . , 2006.

[6] Robb J. Muirhead. *Aspects of Multivariate Statistical Theory.* Wilet New York, NY, 2005.

[7] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *2006 47th annual IEEE symposium on foundations of computer science (FOCS'06)*, pages 143–152. IEEE, 2006.

[8] J. Friedman T. Hastie, R. Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition.* Springer New York, NY, 2009.

[9] Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert. A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis*, 25(3):335–366, 2008.