# 225B Term Project - Randomized SVD

James Albert Nguyen

March 2025

## 1 Introduction

Matrix approximation offers an intuitive solution to the expanding reliance on massive data. Such techniques intend to find a low-rank matrix that preserves most of the action of the original one. In the domains of data analysis and machine learning, common practices like QR-Factorization and Singular Value Decomposition (SVD) express these matrices as factorizations to extract prominent features whilst reducing memory footprint and computational complexity. The combination of these methods, however, can be expensive. Thus, we discuss the benefits of randomization for low-rank matrix approximation to enhance resource efficiency in computing fields.

Randomization in numerical linear algebra presents a relatively modern approach to highly expensive factorizations. While some matrices – such as the Gaussian random matrix – are well-distributed and full-rank, it is common to encounter large matrices that can be reduced to the dimension of a dominant subspace that accounts for most of its action. We quantify the size of a matrix's dominant subspace by its numerical rank, determined by the number of singular values greater than some significance threshold. In particular, we consider some matrix $A \in \mathbb{C}^{m \times n}$ and are interested in using random-sampling techniques to obtain a rank-$k$ (where $k < n$) orthonormal basis matrix $Q \in \mathbb{C}^{m \times k}$ that closely approximates the action of $A$. Under appropriate conditions, we minimize the following quantity:

$$||A - QQ^*A||_2$$

## 2 Background

### 2.1 Existing Methods

To provide a stronger motivation for using randomization in low-rank matrix approximation, we consider existing direct methods for obtaining one. In the special case of symmetric matrices, we can leverage the Eigenvalue Decomposition method [3]. While this offers an efficient solution, it is not generalizable to the asymmetric case. A traditional procedure for obtaining an approximation is

to reconstruct the Truncated Singular Value Decomposition (T-SVD). An improved approach utilizes rank-revealing QR-factorizations to avoid needing to compute the SVD itself [1]. However, this method has its own limitations; As it is constructed from a permutation of a matrix $A$, it requires iterative methods to permute and is restricted to the subspace described by the selected columns.

### 2.2 Modern Application

To address the concerns mentioned in the previous subsection, we introduce randomization to the process of constructing low-rank approximations. The results of *Finding Structure with Randomness (2009)* by Halko, Martinsson, and Tropp discuss improvements in accuracy, numerical stability, and complexity that can be achieved through random sampling methods [2]. In the following exposition, we provide experimental and theoretical results that showcase the superiority of randomized methods compared to direct ones.

## 3 Summary

We first motivate the topic by considering scenarios for which such methods would be advantageous. To do so, we refer to the application of Singular Value Decomposition for data analysis and data compression. Matrix factorizations such as SVD can help us with feature extraction and dimensionality reduction, which can be particularly useful within domains such as image and video processing. However, beyond taking an individual factorization of an image, the repeated application of such a factorization requires computing the full SVD of a (typically) large input many times. To simplify the complexity of this process, we are interested in approximating the basis of the input matrix $A$ before computing the SVD such that the overall complexity is reduced proportional to the size of the reduced matrix.

This discussion will mainly cover two versions of the approximation problem: (i) the fixed-rank problem, and (ii) the fixed-precision problem. These two problems address the need to approximate a matrix by one of a predetermined rank $k$ or within a given approximation error $\epsilon$.

### 3.1 Fixed-Rank Approximation

In the simplest case, we suppose that the desired rank of the approximate basis is given. Suppose we want to find a rank-$k$ orthonormal basis $Q \in \mathbb{C}^{m \times k}$ that approximates $A$ by $QQ^*A$. The main idea of this method is to draw a matrix $\Omega$ whose vectors are randomly sampled from some random distribution such that $Y = A\Omega$ is a random sample from the $range(A)$. Then, the resulting $Q$ matrix from the QR-factorization $Y = QR$ yields an orthonormal basis of the column space of $A$. By varying the number of columns of $\Omega$, we can adjust the rank and size of our basis matrix $Q$.

For some desired rank $k$, we generally are interested in drawing a number of extra vectors from our random distribution as an oversampling parameter $p$. This oversampling feature is inspired the variance introduced with randomization. By the optimality of the truncated-SVD, we know that there exists a minimal rank-$k$ solution $\tilde{X}$ to the expression:

$$\underset{X \in \mathbb{C}^{m \times n},\ rank(X)=k}{\arg\min} ||A - X||_2$$

However, random sampling methods will almost surely not produce this exact minimal solution. By adding an oversampling parameter $p$, we obtain a basis $Q \in \mathbb{C}^{m \times l}$, where $l = k+p$, that closely approximates the minimal solution. The choice of $p$ depends on several factors, including the random matrix properties of the chosen sampling distribution, the size of the input matrix $A$, and the rate of singular value decay of $A$. It is relevant to note that the Gaussian distribution – as used for all examples throughout this exposition – requires little oversampling due to the inherent linear independence and well-distributed nature of random Gaussian vectors. However, there exist other choices of sampling distributions that are structured to accelerate computation at the cost of some approximation accuracy; refer to Subsampled Random Fourier Transform (SRFT) [**?**].

**Algorithm 1** Fixed-Rank Basis Approximation

1: **Input:** $A \in \mathbb{C}^{m \times n}, l$ (desired rank $k$ plus oversampling parameter $p$)
2: **Output:** $Q \in \mathbb{C}^{m \times l}$ orthogonal basis
3: **function** FIXEDRANK($A, l$)
4: Generate $\Omega \in \mathbb{C}^{n \times l}$ matrix with $l$ columns of length-$n$ Gaussian vectors
5: $Y = A\Omega$
6: Compute QR-factorization $Y = QR$
7: **return** $Q$
8: **end function**

From Algorithm 1, one would obtain a rank-$l$ approximation of the matrix $A$. Clearly, the approximation accuracy of the matrix depends heavily on the numerical rank of $A$. However, we will also address the sensitivity of the algorithm to its rate of singular value decay.

## 3.2 Fixed-Rank: Sensitivity to Singular Value Decay Rate

When implementing this algorithm, it is important to consider the effect of the conditioning of the input matrix $A$ on the accuracy of the approximation $QQ^*A$. When considering the potential for a low-rank matrix approximation, it is often of interest to observe the rate of singular value decay; a rapid decay rate indicates a small number of singular vectors, and thus the ability to be approximated well by a lower-rank matrix. For the cases in which this is not true, Algorithm 1 tends to struggle. To account for this vulnerability, we introduce the power method for basis approximation.

Intuitively, we can apply repeated left-hand matrix multiplication of $AA^*$ to the range sample $Y$ from Algorithm 1. The resulting product $(AA^*)^q A\Omega$ denotes a scaled sample of $range(A)$ whose singular vectors are preserved, yet the singular values decay at a much faster rate. Consider the following adaptation:

**Algorithm 2** Fixed-Rank Power-Iteration

**Input:** $A \in \mathbb{C}^{m \times n}, l$ (desired rank $k$ plus oversampling parameter $p$), $q$
**Output:** $Q \in \mathbb{C}^{m \times l}$ orthogonal basis
**function** FIXEDRANK($A, l, q$)
    Generate $\Omega \in \mathbb{C}^{n \times l}$ matrix with $l$ columns of length-$n$ Gaussian vectors
    $Y = (AA^*)^q A\Omega$
    Compute QR-factorization $Y = QR$
    **return** $Q$
**end function**

Algorithm 2 improves upon the accuracy of matrix approximations whose singular values decay at a slow rate. However, when performed in floating point arithmetic, any singular values scaled to be near machine error become susceptible to numerical round-off errors. Thus, the implementation of this algorithm is highly advised against. Instead, we propose the following algorithm:

**Algorithm 3** Fixed-Rank Power-Iteration

**Input:** $A \in \mathbb{C}^{m \times n}, l$ (desired rank $k$ plus oversampling parameter $p$), $q$
**Output:** $Q \in \mathbb{C}^{m \times l}$ orthogonal basis
**function** FIXEDRANKPOWER($A, l, q$)
    Generate $\Omega \in \mathbb{C}^{n \times l}$ matrix with $l$ columns of length-$n$ Gaussian vectors
    $Y = A\Omega$
    $Q_0 \leftarrow Q$ s.t. $Y = QR$
    **for** $i$ in $1, \ldots, q$ **do**
        $\tilde{Q}_i \leftarrow Q$ s.t. $A^* Q_{i-1} = QR$
        $Q_i \leftarrow Q$ s.t. $A\tilde{Q}_{i-1} = QR$
    **end for**
    **return** $Q_q$
**end function**

It follows that the adjusted procedure is algebraically equivalent to Algorithm 2; however, it does not encounter the same numerical round-off errors because it orthonormalizes the basis with each iteration.

### 3.3 Fixed-Precision Algorithm

A more common application of low-rank matrix approximation assumes that the desired approximation rank is unknown. Oftentimes, obtaining a good understanding of the numerical rank of the original matrix would require analyzing the singular value decay and other eigenvalue properties. However, this would

counterintuitively require us to compute the full-SVD. In practice, it is much more likely that we know the level of accuracy to which we would like to approximate a matrix. To approach this fixed-precision problem, we explore a method that adaptively constructs the basis matrix until a specified approximation error can be guaranteed to a determinable degree of certainty.

To accomplish this, we consider adjusting our matrix $Y$ with each iteration such that it is orthogonal to the range of $QQ^*A$. Then, normalizing $y \in Y$ would give us an additional basis vector until we construct a sufficient basis matrix $Q$. In particular, we seek to obtain an $m \times l$ basis $Q^{(l)}$ such that:

$$||A - Q^{(l)}(Q^{(l)})^*A|| < \epsilon \tag{1}$$

for some desired approximation error $\epsilon$. Similarly to the relationship described by the oversampling parameter needed for the fixed-precision problem, the resulting $Q \in \mathbb{C}^{m \times l}$ will likely be of slightly higher dimension than the minimal basis $X \in \mathbb{C}^{m \times k}$ that satisfies the inequality.

To determine whether or not the current iterate $Q^{(j)}$ sufficiently approximates the action of $A$, we probabilistically bound the approximation error as a function of the norm of our range vectors in $Y$. The criteria is inspired by the following lemma:

**Lemma 1.** *Let $B$ be a real $m \times n$ matrix. Fix a positive integer $r$ and a real number $\alpha > 1$. Draw an independent family $\{\omega^{(i)} : i = 1, \ldots, r\}$ of standard Gaussian vectors. Then*

$$||B|| \leq \alpha\sqrt{\frac{2}{\pi}} \max_{i-1,\ldots,r} ||B\omega^{(i)}||$$

*except with probability $\alpha^{-r}$.*

*Proof.* See [4].

As we iteratively add basis vectors to $Q$, we would notice that the orthogonalized sample vectors of $Y$ decrease in norm. Then, for some reliability parameter $r$ – as defined in Lemma 1 –, we can inexpensively evaluate the approximation error of our current basis to an almost sure certainty. The following algorithm takes advantage of this property to implement a proper stopping criteria for the fixed-precision algorithm.

**Algorithm 4** Fixed-Precision Approximation

**Input:** $A \in \mathbb{C}^{m\times n}, r$ (reliability parameter), $\epsilon$ (desired approximation error)
**Output:** $Q \in \mathbb{C}^{m\times l}$ orthogonal basis satisfying (1) with probability $1 - \min\{m,n\}10^{-r}$.
**function** FIXEDPRECISION($A, l, q$)
    Generate $\omega^{(1)}, \ldots, \omega^{(r)}$ length $n$ Gaussian vectors.
    **for** $i = 1, \ldots, r$ **do** $y^{(i)} = A\omega^{(i)}$
        $j = 0$
        $Q^{(0)} = [\,]$
        **while** $\max\{||y^{(j+1)}||, \ldots, ||y^{(j+r)}||\} > \epsilon/10\sqrt{\frac{2}{\pi}}$ **do**
            $j = j + 1$
            Overwrite $y^{(j)}$ by $(I - Q^{(j-1)}(Q^{(j-1)})^*)y^{(j)}$.
            $q^{(j)} = y^{(j)}/||y^{(j)}||$.
            $Q^{(j)} = [Q^{(j-1)}\ q^{(j)}]$.
            Draw additional $\omega^{(j+r)}$ of length $n$.
            $y^{(j+r)} = (I - Q^{(j)}(Q^{(j)})^*)A\omega^{(j+r)}$.
            **for** $i = (j+1), \ldots, (j+r-1)$, **do**
                Overwrite $y^{(i)}$ by $y^{(i)} - q^{(j)}\langle q^{(j)}, y^{(j)}\rangle$.
            **end for**
        **end while**
    **end for**
    **return** $Q^{(j)}$
**end function**

Although similar power iteration methods can be applied to the Algorithm 4, we leave those adaptations as exercises.

## 3.4 Extensions to Singular Value Decomposition

Using any of the described random sampling methods to obtain a low-rank approximation, we explore how to simplify existing factorization algorithms.

Consider some $m \times n$ matrix $A$ with numerical rank $k$. Then using any of Algorithm 1,3, or 4, we obtain an $m \times l$ orthonormal basis matrix $Q$ such that $A \approx QQ^*A$. Note that computing $A = U\Sigma V^*$ is of complexity $O(mn^2)$. Given that $Q^*A \in \mathbb{C}^{m\times l}$, we can compute $Q^*A = \hat{U}\hat{\Sigma}\hat{V}^*$ costs only $O(nl^2)$. Therefore, computing $QQ^*A = Q(\hat{U}\hat{\Sigma}\hat{V}^*)$ has a total cost:

$$T(Q, A) = O(mnl) + O(nl^2) = O(mnl).$$

Whenever the numerical rank of a matrix is significantly less than $n$, we make significant enhancements in the complexity of matrix factorizations.

# 4 Numerical Experiments

In this section, we apply the above algorithms to obtain results comparing the approximation errors of the obtained $QQ^*A$ and $Q\hat{U}\hat{\Sigma}\hat{V}^*$.

## 4.1 Experiment Setup

All experiments will be performed on a $200 \times 200$ matrix $A$. Recall from Section 3.2 that the algorithm is particularly sensitive to matrices that possess a slow rate of singular value decay. To avoid complications, we obtain $A$ from a discretized Laplace integral of the form:

$$[S\sigma](x) = c \int_{\Gamma_1} \log|x - y|\sigma(y)\mathrm{d}A(y), \qquad x \in \Gamma_2$$

As previously mentioned, all experiments are conducted using the Gaussian distribution to produce a sample from $range(A)$.

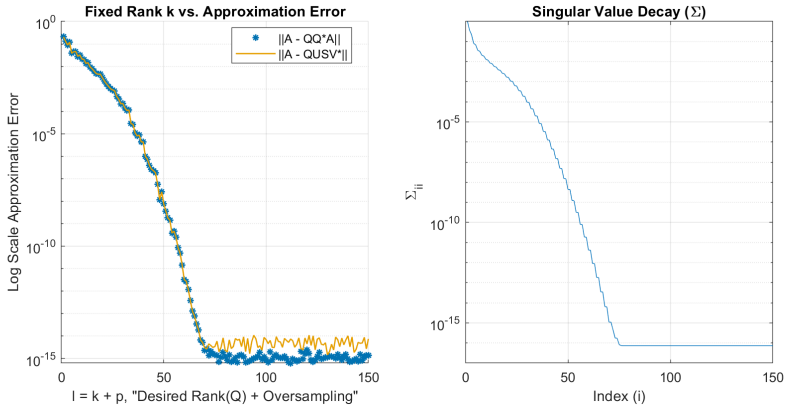


Figure 1: Algorithm 1 implemented for $k = 5, 10, \ldots, 150$; $p = 5$ to approximate a Laplace integral operator. The horizontal axis indicates the number of random samples used to generate the orthonormal basis $Q$, and the vertical axis indicates the approximation error on logarithmic scale. A comparison of the singular value decay is provided.

A quick glance at Figure 4.1 reveals an observable similarity between the curve of the approximation error and the decay rate of the singular values. This relationship corresponds with our expectation that a rapid singular value decay is indicative of lower-rank representability.

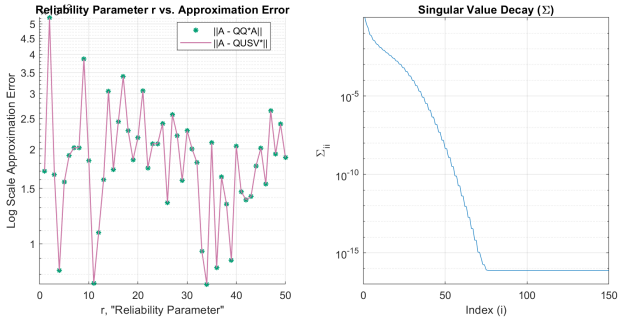


Figure 2: Algorithm 4 implemented for reliability parameter $r = 1, \ldots, 50$; tolerance $\epsilon = 1e-10$ to approximate a Laplace integral operator. The horizontal axis indicates the number of range vectors used to bound the approximation error at each iteration. The vertical axis indicates the order of magnitude of the log-scale approximation error, beginning at 1e-12.

Figure 2 gives an example of the approximation error for an approximate basis at each $r = 1, \ldots, 50$. We notice that, given a tolerance of $1e-10$, the error is expected to satisfy

$$||A - QQ^*A|| < 1e-10$$

with probability at least $1 - 200(10^{-r})$.

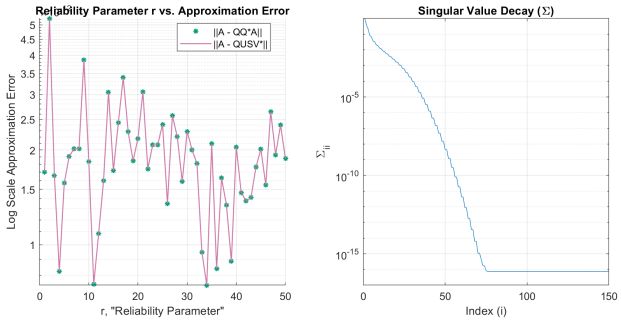


Figure 3: Algorithm 3 implemented for $q = 1, \ldots, 60$ to approximate a random Gaussian matrix. The horizontal axis indicates the number of applications of the power iteration.

To explore the ill-conditioned case of an input matrix with a slow rate of singular value decay, we generate a random Gaussian matrix $A \in \mathbb{C}^{200 \times 200}$. Figure 3 reveals that repeated application of the power iteration decreases the approximation error, given such a matrix.

# References


[1] Tony F. Chan and Per Christian Hansen. Computing truncated singular value decomposition least squares solutions by rank revealing qr-factorizations. *SIAM Journal on Scientific and Statistical Computing*, 11(3):519–530, 1990.

[2] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

[3] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM, 1997.

[4] Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert. A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis*, 25:335–366, 11 2008.