Thai Journal of Mathematics : 371-385
Special Issue : Annual Meeting in Mathematics 2019

http://thaijmath.in.cmu.ac.th
Online ISSN 1686-0209

A Numerical Study of Efficient Sampling Strategies for Randomized Singular Value Decomposition

Siriwan Intawichai and Saifon Chaturantabut[1]

Department of Mathematics and Statistics, Thammasat University
e-mail : siriwan_amth50@hotmail.com (S. Intawichai)
saifon@mathstat.sci.tu.ac.th (S. Chaturantabut)

Abstract : The randomized singular value decomposition (rSVD) method is a powerful dimension reduction technique that uses random projection matrices to project the data onto lower dimensional subspace. A crucial step of rSVD algorithm is the sampling process, which will be used further to generate a low-dimensional basis for the subspace of available data. To improve computational performance, this paper incorporates rSVD with different efficient sampling strategies, which include Gaussian sampling, uniform sampling, sparse sampling and sampling with K-mean clustering. The numerical tests compare the accuracy and the execution time used in computing optimal low-rank basis by applying rSVD with each of these sampling methods for an image reconstruction application.

Keywords : randomized singular value decomposition; Gaussian sampling; uniform sampling; sparse sampling; K-mean clustering.
2010 Mathematics Subject Classification : 65F15; 65F30.

1 Introduction

The singular value decomposition (SVD) was discovered independently by Beltrami [1] in 1873 and Jordan [2] in 1874 during their research on linear algebra. Both are deemed the progenitors in SVD, however many mathematicians have

This study was supported by Thammasat University Research Fund, Contract No. TUGR 2/12/2562.

[1]Corresponding author.

been working on discovering its properties and developing algorithms for its computation [3, 4]. SVD is a well-known technique for dimensionality reduction which applied for solving many problems, such as low-rank matrix approximations [5, 4] which is a good approximation to the original matrix, least squares problems [6], image processing [7] and model order reduction to compress the high-dimensional data space into a lower-dimensional space [8]. However, the tradition way of computing the full SVD of a high-dimension matrix is often expensive as well as memory intensive.

T. Sarlos (2006) [9], E. Liberty et. al.(2007) [10] and Halko et al. (2011) [13] introduced a more robust approach based on random projections that is the randomized singular value decomposition (rSVD) method. These method can decrease the computational work of extracting low-rank approximations and are more accurate than SVD method. The rSVD algorithms focus on efficiently sampling the important matrix elements. Many sampling strategies, including Gaussian sampling [10]-[14], uniform sampling [15], [16], sparse sampling [17, 18], and sampling with K-means clustering [19, 20] have been proposed. The random sampling method is effective when estimating characteristics of the whole data by a relatively small sample and the Johnson-Lindenstrauss Lemma [13] guarantees that the distances between points can be preserved by the projection from the high dimension space to a lower dimensional subspace. As a result, compared to full SVD, randomized SVD methods are memory efficient and can often obtain good low-rank approximations in a significantly faster way.

The task of randomized singular value decomposition (rSVD) to computing a low-rank approximation to a given matrix can be split naturally into two computational stages. The first is using random sampling methods to construct a low-dimensional subspace that captures the action of the matrix. Here, we are interested in several kind of sampling. The second is to restrict the matrix to the subspace and then compute a standard factorization (QR, eigendecomposition, SVD, etc.) of the reduced matrix.

In this paper, we present the randomized singular value decomposition (rSVD) methods that using three different approaches for constructing the reduce matrix, QR decomposition, eigendecomposition and SVD. Moreover, for each rSVD method, we incorporate the algorithm with different efficient sampling strategies, which include Gaussian sampling, Uniform sampling, Sparse sampling, and sampling with K-mean clustering. The results of rSVD are shown by the numerical experiments and the comparisons of the computation time and accuracy by applying rSVD with each of these sampling methods with image processing problem.

The remainder of this paper is organized as follows. In Section 2, we give a detailed overview of rSVD and consider the three approaches for constructing the reduce matrix as reviewed in Section 2.1 for the rSVD by using QR decomposition, Section 2.2 for the rSVD by using eigendecomposition, and Section 2.3 for the rSVD by using SVD. The different sampling strategies are considered in Section 3, including Gaussian sampling in Section 3.1, uniform sampling in Section 3.2, Sparse sampling in Section 3.3 and in Section 3.4 for the sampling with K-mean clustering. Then, we consider an application of image processing problem in image

reconstruction in Section 4. In Section 5, we show the numerical experiments of the rSVD with each of these sampling methods for constructing the projection basis and compare the CPU times together with the reconstruction errors. Finally, some concluding remarks are given in Section 6.

2 The Overview of Randomized Singular Value Decomposition

To begin with, we review some classical deterministic matrix decomposition methods and give defining terms and notations.

The singular value decomposition (SVD) of a matrix $X \in \mathbb{R}^{m\times n}$ with rank r can be expressed as

$$X = USV^T, \tag{2.1}$$

where $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ are matrices with orthonormal columns. The column vectors of U and V are left and right singular vectors, respectively, denoted as $u_i \in \mathbb{R}^m$ and $v_i \in \mathbb{R}^n$. $S \in \mathbb{R}^{r\times r}$ is a diagonal matrix with $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0$, are known as singular values.

The truncated X_k can be obtained by computing full SVD and then truncating it by selecting the top k dominant singular values and their corresponding singular vectors such that

$$X_k = U_k S_k V_k^T = \sum_{i=1}^{k} \sigma_i u_i v_i^T, \tag{2.2}$$

where $k < r$, and $U_k \in \mathbb{R}^{m\times k}$, $V_k \in \mathbb{R}^{n\times k}$ are matrices with orthonormal columns. $S_k \in \mathbb{R}^{k\times k}$ is a diagonal matrix with the singular values $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_k > 0$ on its diagonal entries. For a fixed dimension $k < r$, X_k is the optimal solution of least-squares problem:

$$\min_{W\in\mathbb{R}^{m\times n}} \|X - W\|_2^2, \quad \text{rank}(W) = k, \tag{2.3}$$

and the corresponding minimum error is given by $\|X - X_k\|_2^2 = \sum_{\ell=k+1}^{r} \sigma_\ell^2$. In this case, the orthonormal basis of rank k formed by the k columns in $U_k = [u_1, \ldots, u_k] \in \mathbb{R}^{n\times k}$ is called **proper orthogonal decomposition** (POD) basis of dimension k for the matrix X.

We define the randomized singular value decomposition (rSVD) of X as,

$$\widehat{X}_k = \widehat{U}_k \widehat{S}_k \widehat{V}_k^T, \tag{2.4}$$

where $k < r$, and $\widehat{U}_k \in \mathbb{R}^{m\times k}$, $\widehat{V}_k \in \mathbb{R}^{n\times k}$ are matrices with orthonormal columns. $\widehat{S}_k \in \mathbb{R}^{k\times k}$ is a diagonal matrix with the singular values $\widehat{\sigma}_1 \geq \widehat{\sigma}_2 \geq ... \geq \widehat{\sigma}_k > 0$. Details are given as follows.

Define the random projection of a matrix as, $Y = X\Omega$, where Ω is a random matrix, which is obtained by the sampling strategies explained later in this section. The rSVD algorithm as considered by [13] explores approximate matrix

factorizations using random projections, separating the process into two stages: in the first stage, random sampling is used to obtain a reduced matrix whose range approximates the range of the data matrix, in the second stage, the reduced matrix is factorized. The first stage of the method is common to many approximate matrix factorization methods: for a given $\epsilon > 0$, we wish to find a matrix Q with orthonormal columns such that

$$\|X - QQ^T X\|_2^2 \leq \epsilon. \tag{2.5}$$

Without loss of generality, we assume $Q \in \mathbb{R}^{m\times\ell}$, $\ell \ll n$. The columns of Q form an orthogonal basis for the range of $X\Omega$ which is an approximation to the range of X, where Ω is a matrix composed of the random vectors.

The second stage of the rSVD method is to compute the SVD of the reduced matrix $B := Q^T X$. Suppose $B = \widetilde{U}\widehat{S}\widehat{V}^T$ is the SVD of B, which can be obtained from the orthogonal projection of X onto the low dimensional subspace spanned by columns of Q. Then, we finally obtain the approximated POD basis or $\widehat{U}$ of X from the product $Q\widetilde{U}$.

We would like the basis matrix Q to contain as few columns as possible, but it is even more important to have an accurate approximation of the input matrix. However, there are many methods for constructing a matrix Q such as QR, SVD, and eigendecomposition. In this work, we will consider three approaches for computing this matrix as shown next.

2.1 The rSVD by using QR decomposition

In this section, we considered the rSVD algorithm by using QR decomposition for constructing Q. From above, we can summarize the computational steps of rSVD algorithm using QR decomposition in Algorithm 1.

Algorithm 1: The rSVD-QR algorithm [13]

INPUT	: A data matrix $X \in \mathbb{R}^{m\times n}$ with target rank k and an oversampling parameter p
OUTPUT	: The rSVD of X: $\widehat{U}, \widehat{S}, \widehat{V}$
Step 1.	Draw a random matrix Ω with dimension $n \times (k+p)$.
Step 2.	Form the matrix product $Y = X\Omega$.
Step 3.	Construct Q from the QR decomposition of Y.
Step 4.	Set $B = Q^T X$.
Step 5.	Compute an SVD of the small matrix: $B = \widetilde{U}\widehat{S}\widehat{V}^T$.
Step 6.	Set $\widehat{U} = Q\widetilde{U}$.

2.2 The rSVD by using Eigendecomposition

In this section, we considered the rSVD algorithm by using eigendecomposition. Recall, the eigenvalue decomposition of a square matrix $Z \in \mathbb{R}^{\ell\times\ell}$, is defined as $Z = TDT^{-1}$, where $T \in \mathbb{R}^{\ell\times\ell}$ is a nonsingular matrix whose i^{th} column is an

eigenvector of Z and $D \in \mathbb{R}^{\ell \times \ell}$ is a diagonal matrix whose i^{th} diagonal entry is the corresponding eigenvalue.

Given a matrix $X \in \mathbb{R}^{m \times n}$, with rank k, we define a random matrix $\Omega \in \mathbb{R}^{n \times \ell}$ and suppose that $Y = X\Omega$. We first form a smaller matrix $Z := Y^T Y \in \mathbb{R}^{\ell \times \ell}$, then computing the eigendecomposition $Z = TDT^{-1}$. Since Z is real symmetric matrix, the eigenvalues are non-negative real numbers and the eigenvectors are orthogonal to each other, i.e $T^{-1} = T^T$. Note that $T \in \mathbb{R}^{\ell \times \ell}$ is the right singular matrix of Y and $\sqrt{D} \in \mathbb{R}^{\ell \times \ell}$ is the diagonal matrix that contains the singular values of Y on its diagonal entries.

If $\ell > k$, we truncate the decomposition, by extracting the first k dominant eigenvectors (or right singular vectors), so that we get $T_k \in \mathbb{R}^{\ell \times k}$ and singular values $\widetilde{S} = \sqrt{D_k} \in \mathbb{R}^{k \times k}$.

From the SVD of $X = USV^T$, it follows that the left and right singular vectors can be derived from each other as $U = XVS^{-1}$ and $V = X^T US^{-1}$, respectively. Thus, the approximate left singular vectors of Y is $\widetilde{U} := YT_k\widetilde{S}^{-1}$. By setting $B = \widetilde{U}^T X$, we can compute the SVD of B so that $B := Q\widehat{S}\widehat{V}^T$.

Hence, $\widetilde{U}(\widetilde{U}^T X) = \widetilde{U}B =: \widetilde{U}[Q\widehat{S}\widehat{V}^T] =: (\widetilde{U}Q)\widehat{S}\widehat{V}^T =: \widehat{U}\widehat{S}\widehat{V}^T \approx X$. The computational steps are summarized in Algorithm 2.

Algorithm 2: The rSVD-Eigen algorithm

INPUT	: A data matrix $X \in \mathbb{R}^{m \times n}$ with target rank k and an oversampling parameter p
OUTPUT	: The rSVD of X: $\widehat{U}, \widehat{S}, \widehat{V}$
Step 1.	Draw a random matrix $\Omega \in \mathbb{R}^{n \times \ell}$ where $\ell = k + p$.
Step 2.	Form the matrix product $Y = X\Omega$ and then the square matrix $Z = Y^T Y$.
Step 3.	(Ensure symmetry by $Z = \frac{1}{2}(Z + Z^T)$)
Step 4.	Computing the eigendecomposition, $Z = TDT^T$.
Step 5.	Truncating $T_k \in \mathbb{R}^{\ell \times k}$, and calculating $\widetilde{S} = \sqrt{D_k}$ where $D_k \in \mathbb{R}^{k \times k}$.
Step 6.	Approximate, $\widetilde{U} := YT_k\widetilde{S}^{-1}$ then set $B = \widetilde{U}^T X$.
Step 7.	Compute an SVD of $B = Q\widehat{S}\widehat{V}$.
Step 8.	Set $\widehat{U} = \widetilde{U}Q$.

2.3 The rSVD by using SVD

In this section, we considered the rSVD algorithm by applying SVD on the reduced matrix Y. For a given matrix $X \in \mathbb{R}^{m \times n}$ with rank k, we define a random matrix Ω and suppose that $Y = X\Omega$. First, we compute the SVD of Y as $Y = QSV^T$. Then, we form $B = Q^T X$ and compute the SVD of $B = \widetilde{U}\widehat{S}\widehat{V}^T$. Finally, we obtain the approximated POD basis or the left singular vectors of X from the product $Q\widetilde{U}$. The computational steps are summarized in Algorithm 3.

Algorithm 3: The rSVD-SVD algorithm

INPUT	: A data matrix $X \in \mathbb{R}^{m\times n}$ with target rank k and an oversampling parameter p
OUTPUT:	: The rSVD of X: $\widehat{U}, \widehat{S}, \widehat{V}$
Step 1.	Draw a random matrix $\Omega \in \mathbb{R}^{n\times \ell}$ where $\ell = k + p$.
Step 2.	Form the matrix $Y = X\Omega$
Step 3.	Computing the truncated SVD of Y then $Y = QSV^T$
Step 4.	Form $B = Q^T X$
Step 5.	Compute SVD of $B = \widetilde{U}\widehat{S}\widehat{V}^T$.
Step 6.	Set $\widehat{U} = Q\widetilde{U}$.

Note that, Algorithm 3 presents a different implementation that is slightly more computationally expensive than Algorithm 2. However, the accuracy and numerical stability of Algorithm 3 is improved by computing the deterministic SVD of Y in Step 3, instead of forming $Z = Y^T Y$ and using the eigendecomposition of the smaller matrix Z. In practice, this algorithm is suitable for fat matrices, i.e. $m < n$. However, for tall and skinny matrices, i.e. $m > n$ which have a large dimension m it becomes more efficient to form the smaller matrix Z first, as outlined in Algorithm 2.

3 Random test matrices

In this section, we will present the random sampling strategies that are used to draw the random matrix Ω, projecting a high-dimensional data to a lower-dimensional space. Specifically, we seek a matrix with independent identically distributed (i.i.d.) entries from certain distribution, which ensures that its columns are linearly independent with high probability. The interesting choices for constructing the random test matrix as explained next.

3.1 Gaussian random matrix

Gaussian random matrix is the simplest random matrix ([10]-[14]) to construct a random test matrix is to draw entries from the standard normal distribution $\mathcal{N}(0, 1)$, (Mean $\mu = 0$, Standard deviation $\sigma = 1$). Then, the probability density function of x is given by $f(x) = \dfrac{1}{\sqrt{2\pi}} \exp^{-\dfrac{x^2}{2}}$, for $-\infty < x < \infty$.

This matrix is the most popular to be used because it has random entries that are i.i.d. However, it is expensive to generate and dense matrix multiplications are computationally intensive.

3.2 Uniform random matrix

The uniform random matrix ([15, 16]) is consisting of the random variables which follow the uniform distribution with parameters a and b. We write $x \sim \mathcal{U}(a, b)$.

The random variable x is uniformly distributed on the interval $[a, b]$ if it has the following probability density function $f(x) = \dfrac{1}{b-a}$, for $a \leq x \leq b$ and otherwise $f(x) = 0$. In particular, we use $x \sim \mathcal{U}(0, 1)$, i.e. the random variables are uniformly distributed on the interval $[0, 1]$ or the unit uniform random variable.

3.3 Sparse matrix

The sparse matrix [17, 18] is from the sparse random projections, originally introduced by Achlioptas [5]. This matrix has only a few nonzero random entries, which are i.i.d. This matrix is better than a dense random test matrix in term of computational complexity. The sparse matrix can be constructed by drawing entries ω_{ij} from the following distribution

$$\omega_{ij} = \sqrt{c} \begin{cases} -1, & \text{with prob.} \frac{1}{2c}; \\ 0, & \text{with prob.} 1 - \frac{1}{c}; \\ 1, & \text{with prob.} \frac{1}{2c}. \end{cases}$$

The parameter c controls the density of the nonzero entries, e.g. $c = 2, 3$. In this paper, we use $c = 3$ and, therefore, the sparse matrix will have entries $-\sqrt{3}, 0$, and $\sqrt{3}$ with probability $\frac{1}{6}, \frac{1}{3}$, and $\frac{1}{6}$, respectively.

3.4 K-mean sampling matrix

K-means clustering, or Lloyd's algorithm [19, 20] is an iterative, data-partitioning algorithm. It assigns n observations to exactly one of k clusters defined by centroids, (k- partition of the columns of the original matrix). Given $x_1, x_2, ..., x_m \in \mathbb{R}^n$, we wish to have k points ($\mu_1, \mu_2, ..., \mu_k \in \mathbb{R}^n$) that satisfy $\sum_{i=1}^{n} \min_{j=1:k} \|x_i - \mu_j\|^2$. In this paper, we use K-mean sampling matrix whose column vectors are formed by $\mu_1, \mu_2, ..., \mu_k \in \mathbb{R}^n$.

4 Application in image reconstruction

This section describes the standard projection-based approach [21] to reconstruct missing data of incomplete image. This approach uses the least-squares method for estimating missing data with an optimal low-rank basis in Euclidean norm, which is also called proper orthogonal decomposition (POD) basis.

Approximating an incomplete data vector will use a projection onto a subspace spanned by a basis that represents the related complete data vectors. First, we define complete and incomplete data vectors. Let $\{x_1, x_2, \ldots, x_{n_s}\} \subset \mathbb{R}^n$ be a complete data set and form a matrix of a complete data $X = [x_1, x_2 \ldots x_{n_s}] \in \mathbb{R}^{n \times n_s}$.

Suppose $\widehat{x} \in \mathbb{R}^n$ is an incomplete sample and $n = n_c + n_g$ where n_c, n_g are the numbers of known and unknown components respectively.

Suppose that $C = [e_{c_1}, \ldots, e_{c_{n_c}}] \in \mathbb{R}^{n\times n_c}$ and $G = [e_{g_1}, \ldots, e_{g_{n_g}}] \in \mathbb{R}^{n\times n_g}$, where $e_{c_i}, e_{g_i} \in \mathbb{R}^n$ are the c_i-th, g_i-th column of the identity matrix I_n, for $\{c_1, c_2, \ldots, c_{n_c}\}, \{g_1, g_2, \ldots, g_{n_g}\} \subset \{1, 2, \ldots, n\}$ are the indices of the known and unknown components.

Let $\widehat{x}_c := C^T\widehat{x} \in \mathbb{R}^{n_c}$ and $\widehat{x}_g := G^T\widehat{x} \in \mathbb{R}^{n_g}$. Then, the known components and the unknown components are given in the vectors $\widehat{x}_c$ and $\widehat{x}_g$, respectively.

Note that, pre-multiplying C^T is equivalent to extracting the n_c rows corresponding to the indices $c_1, \ldots, c_{n_c}$. Similarly, G^T is equivalent to extracting the n_g rows corresponding to the indices $g_1, \ldots, g_{n_g}$.

The missing components contained in $\widehat{x}_g$ will be approximated by first projecting $\widehat{x}$ onto the column span of a basis matrix U with rank k

$$\widehat{x} \;\approx\; Ua, \qquad \text{or} \qquad \widehat{x}_c \approx U_c a \;\text{ and }\; \widehat{x}_g \approx U_g a,$$

for some coefficient vector $a \in \mathbb{R}^k$, and where $U_c := C^T U \in \mathbb{R}^{n_c\times k}$, $U_g := G^T U \in \mathbb{R}^{n_g\times k}$.

The known components contained in $\widehat{x}_c$ are then used to determine the coefficient vector a through the approximation $\widehat{x}_c \approx U_c a$ from the following least-squares problem:

$$\min_{a\in\mathbb{R}^k} \|\widehat{x}_c - U_c a\|_2^2. \tag{4.1}$$

The solution of the above problem is given by $a = U_c^\dagger \widehat{x}_c$, where $U_c^\dagger = (U_c^T U_c)^{-1} U_c^T$ is the Moore-Penrose psudoinverse. That is,

$$\widehat{x}_g \;\approx\; U_g a \;=\; U_g U_c^\dagger \widehat{x}_c. \tag{4.2}$$

The steps described above, which will called POD-LS approach, are summarized in Algorithm 4 below.

Algorithm 4: Standard POD-LS approach for approximating missing data [21]

INPUT	**:** - Complete data set $\{x_j\}_{j=1}^{n_s} \subset \mathbb{R}^n$ and, dimension $k \le rank(\{x_j\}_{j=1}^{n_s})$ - Incomplete data $\widehat{x} \in \mathbb{R}^n$ with known entries in $\widehat{x}_c \in \mathbb{R}^{n_c}$ and unknown entries in $\widehat{x}_g \in \mathbb{R}^{n_g}$, where $n = n_c + n_g$.
OUTPUT	**:** Approximation of $\widehat{x}_g$
Step 1.	Create snapshot matrix : $X = [x_1, \ldots, x_{n_s}] \in \mathbb{R}^{n\times n_s}$ and let $r = rank(X)$
Step 2.	Construct basis U of rank $k \le r$ for x.
Step 3.	Find coefficient vector a from $\widehat{x}_c$ using least-squares problem in (4.1): $\min_{a\in\mathbb{R}^k} \|\widehat{x}_c - U_c a\|_2^2$.
Step 4.	Compute the approximation $\widehat{x}_g \;\approx\; U_g a$.

Next, we consider the optimal basis which is obtained from the singular value decomposition (SVD) because it is optimal in the least-squares sense. It follows

from [11] and [21] that the basis defined above can be obtained from the left singular vector of the matrix X.

Let $X = [x_1, \ldots, x_{n_s}] \in \mathbb{R}^{n \times n_s}$ and $k < r = rank(X)$. The SVD of X is $X = USV^T$, where $U = [u_1, \ldots, u_r] \in \mathbb{R}^{n \times r}$ and $V = [v_1, \ldots, v_r] \in \mathbb{R}^{n_s \times r}$ are matrices with orthonormal columns and $S = diag(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{r \times r}$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$. As stated previously, the optimal solution of least-squares problem $\min_W \|X - W\|_2^2$, $\text{rank}(W) = k$ is $X_k = U_k S_k V_k^T$ with minimum error $\|X - X_k\|_2^2 = \sum_{\ell=k+1}^{r} \sigma_\ell^2$. The optimal orthonormal basis of rank k (the POD basis) is formed by the columns in $U_k = [u_1, \ldots, u_k] \in \mathbb{R}^{n \times k}$, $k \leq r$. However, we will use the rSVD methods in the previous section to find the approximate optimal orthonormal basis, which can significantly reduce the computational complexity, and therefore computational time, of the standard approach for computing SVD.

5 Numerical Experiments

In this section, we show the numerical experiments of the three rSVD methods with each of these sampling approaches for constructing the projection basis and compare the CPU times together with the reconstruction errors. Here, we use the relative error in 2-norm.

We use a standard test image in the field of image processing, called *Lena* picture, to compare the accuracy and efficiency of the basis used in missing data reconstruction. We consider this image with 10% and 30% missing pixels and each image will be considered as a matrix with gray scale color, which are shown in Figure 1.

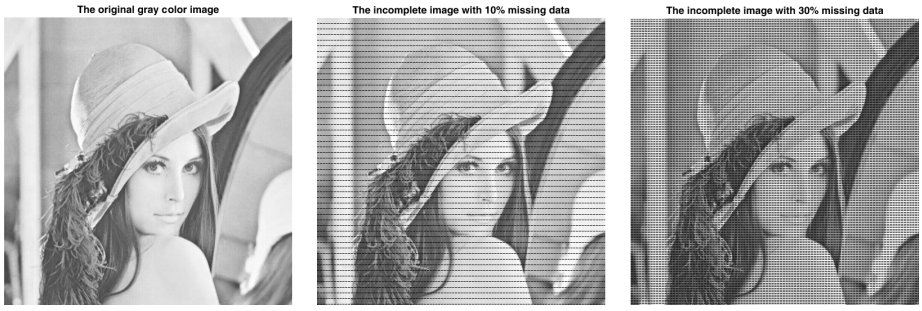


Figure 1: The original image and the incomplete image with 10% missing data and 30% missing data.

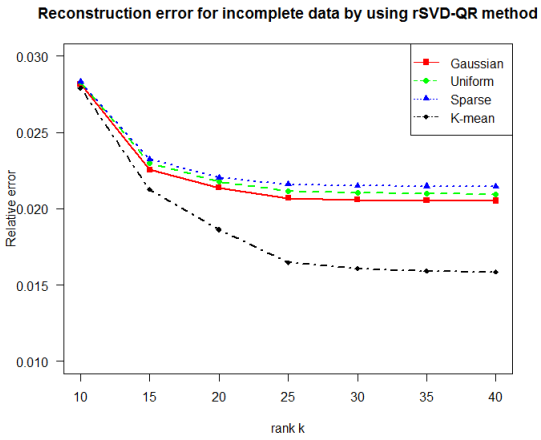
The reconstructed images from three different approaches are shown below when $k = 40$ is used, for example, the reconstructed images of the rSVD-QR method with four sampling are shown in Figure 2.

The error and computational time for constructing basis set by using the rSVD-QR, the rSVD-eigen, and the rSVD-SVD methods with four sampling techniques including Gaussian sampling, uniform sampling, sparse sampling and the sampling

Figure 2: The reconstructed images of 10% missing data by using the rSVD-QR method

with K-mean clustering are shown in Figure 3, Figure 4 and Figure 5. In each of these figures, the accuracy of these approaches are shown through the plots of the relative errors for different truncated dimension k. It can be seen that the accuracy can be improved by increasing dimension k up to certain value where the stagnation occurs. In the table of each figure, the comparison of computational time is given in terms of both actual CPU time (seconds) and speedup, which is scaled with the maximum CPU time of K-mean approach for each fixed dimension k.

These results show that the K-mean sampling strategies gives less error than the others in all numerical examples. However, the computational times of the three rSVD methods with K-mean sampling take longer than other sampling approaches. More discussion of the results is also provided in the next section.



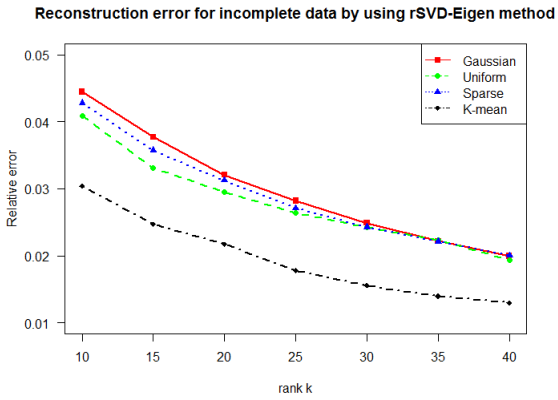
k	CPU Time (in seconds)				Speedup			
	Gauss	Uniform	Sparse	K-mean	Gauss	Uniform	Sparse	K-mean
10	0.0163	0.0158	0.0156	0.2658	16.3	16.8	17.1	1.0
15	0.0194	0.0197	0.0195	0.4027	20.7	20.4	20.6	1.0
20	0.0247	0.0243	0.0236	0.4680	18.9	19.3	19.9	1.0
25	0.0291	0.0291	0.0282	0.5735	19.7	19.7	20.3	1.0
30	0.0329	0.0337	0.0322	0.7395	22.5	22.0	23.0	1.0
35	0.0371	0.0376	0.0367	0.8369	22.5	22.3	22.8	1.0
40	0.0416	0.0420	0.0408	0.9604	23.1	22.9	23.6	1.0

Figure 3: [Numerical Test 1] Relative error and CPU time of the rSVD-QR method for computing basis used in the reconstruction of the Lenna picture when 30% of pixels are missing.

6 Conclusions and Discussion

This work performed numerical studies of the rSVD methods that use three different approaches for constructing the reduce matrix, which are QR decomposition, eigendecomposition and SVD. Moreover, for each rSVD method, we incorporate the algorithm with different efficient sampling strategies, which include Gaussian sampling, uniform sampling, sparse sampling, and sampling with K-mean clustering.

By applying these approaches in reconstructing missing image data with the least-squares approximation, using the K-mean sampling was shown to give minimum error for computing projection basis when compared with Gaussian sampling, uniform sampling and sparse sampling. By visual inspection, the computational time is hardly noticeable. This is because the size of image is small and the rSVD methods approximates the column space of the input matrix. However, in most practical applications that require to process a large number of high-dimensional images, this computational efficiency become significant as shown through the

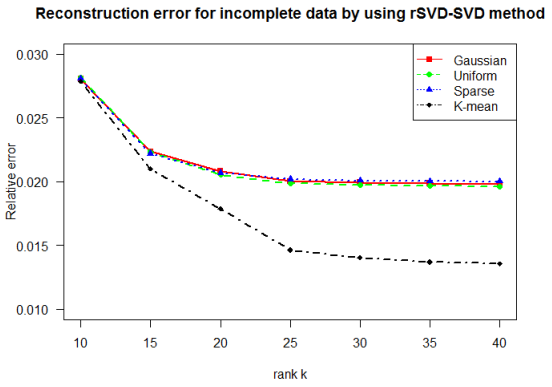


	CPU Time (in seconds)				Speedup			
k	Gauss	Uniform	Sparse	K-mean	Gauss	Uniform	Sparse	K-mean
10	0.0121	0.0126	0.0119	0.3192	26.3	25.3	26.8	1.0
15	0.0151	0.0157	0.0151	0.3156	21.0	20.1	20.9	1.0
20	0.0183	0.0188	0.0177	0.4966	27.1	26.4	28.1	1.0
25	0.0223	0.0220	0.0212	0.5657	25.4	25.7	26.7	1.0
30	0.0250	0.0257	0.0246	0.7340	29.3	28.6	29.8	1.0
35	0.0287	0.0290	0.0282	0.8540	29.8	29.4	30.3	1.0
40	0.0317	0.0328	0.0314	1.0139	32.0	30.9	32.3	1.0

Figure 4: [Numerical Test 2] Relative error and CPU time of the rSVD-Eigen method for computing basis used in the reconstruction of the Lenna picture when 30% of pixels are missing.

speedup comparison in the tables of Figure 3, Figure 4 and Figure 5. We observed a trade-off between the accuracy and the computational time. In particular, while the K-mean sampling approach gives higher accuracy, it takes roughly more than 16-32 longer CPU time than the other three sampling techniques for each truncated dimension k.

The numerical procedures presented in this work can be extended to use with large-scale image processing problems or other applications that require to use SVD computation to speedup the overall simulation time while maintaining the accuracy of the original problems.



k	CPU Time (in seconds)				Speedup			
	Gauss	Uniform	Sparse	K-mean	Gauss	Uniform	Sparse	K-mean
10	0.0164	0.0161	0.0154	0.3011	18.4	18.7	19.6	1.0
15	0.0194	0.0195	0.0190	0.3453	17.8	17.7	18.2	1.0
20	0.0239	0.0240	0.0233	0.4871	20.4	20.3	20.9	1.0
25	0.0279	0.0279	0.0273	0.5689	20.4	20.4	20.8	1.0
30	0.0322	0.0324	0.0317	0.7104	22.1	21.9	22.4	1.0
35	0.0362	0.0365	0.0359	0.9297	25.7	25.5	25.9	1.0
40	0.0405	0.0408	0.0400	0.9779	24.2	24.0	24.4	1.0

Figure 5: [Numerical Test 3] Relative error and CPU time of the rSVD-SVD method for computing basis used in the reconstruction of the Lenna picture when 30% of pixels are missing.

Acknowledgment: The authors would like to thank the reviewers for many insightful comments to improve this paper. This study was supported by Thammasat University Research Fund, Contract No. TUGR 2/12/2562.

References

[1] E. Beltrami, Sulle Funzioni Bilineari, Giornale di Matematiche ud uso Degli Studenti Delle Universita 11 (1873) 98–106.

[2] C. Jordan, Mémoire sur les formes bilinéaires, Journal de Mathématiques Pures et Appliquées 19 (1874) 35–54.

[3] G.W. Stewary, On the early history of the singular value decomposition, Siam Review 35 (1993) 551–566.

[4] G.H. Golub, C.F. Van Loan, Matrix Computations, 3rd ed., Johns Hopkins University Press, Baltimore, Maryland, 1996.

[5] D. Achlioptas, F. McSherry, Fast computation of low-rank matrix approximations, J. Assoc. Comput. Mach. 54 (2007) https://doi.org/10.1145/1219092.1219097.

[6] G.H. Golub, C. Reinsch, Singular value decomposition and least squares solutions (1971), 134–151.

[7] H.C. Andrews, C.L. Patterson, Singular value decompositions and digital image processing 24 (1976), 26–53.

[8] K.V. Ravi Kanth, D. Agrawal, A.E. Abbadi, A. Singh, Dimensionality reduction for similarity searching in dynamic databases, In ACM SIGMOD Conference Proceedings (1998), 166–176.

[9] T. Sarlos, Improved Approximation Algorithms for Large Matrices via Random Projections. In Foundations of Computer Science. 47th Annual IEEE Symposium (2006), 143–152.

[10] E. Liberty, F. Woolfe, P.G. Martinsson, V. Rokhlin, M. Tygert. Randomized algorithms for the low-rank approximation of matrices, In Proc. Natl. Acad. Sci. 104 (2007) 20167–20172.

[11] S. Volkwein, Proper Orthogonal Decomposition: Theory and Reduced-Order Modelling. Lecture notes, University of Konstanz.

[12] M.W. Mahoney, Randomized algorithms for matrices and data, Found. Trends Mach. Learning 3 (2011) 123–224.

[13] N. Halko, P.G. Martinsson, J.A. Tropp, Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions." SIAM Review 53 (2) (2011) 217–288.

[14] P.G. Martinsson, Randomized Methods for Matrix Computations and Analysis of High Dimensional Data, (2016) 1–55 https://arXiv:1607.01649 .

[15] P. Drineas, R. Kannan, M.W. Mahoney, Fast Monte-Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix, SIAM J. Comput. 36 (2006) 158–183.

[16] S. Eriksson-Bique, M. Solbrig, M. Stefanelli, S. Warkentin, R. Abbey, I. Ipsen, Importance sampling for a Monte Carlo matrix multiplication algorithm, with application to information retrieval, SIAM J. Sci. Comput. 33 (2011) 1689–1706.

[17] D. Achlioptas, Database-friendly random projections: Johnson-Lindenstrauss with binary coins, Journal of Computer and System Sciences 66 (4) (2003) 671–687.

[18] P. Li, T.J. Hastie, K.W. Church, Very sparse random projections, In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2006) 287–296.

[19] C. Boutsidis, A. Zouzias, P. Drineas, Random projections for k-means clustering, In NIPS 2010, 2010.

[20] C. Boutsidis, A. Zouzias, M.W. Mahoney, P. Drineas, Randomized dimensionality reduction for k-means clustering. IEEE Transactions on Information Theory 61 (2) (2015) 1045–1062.

[21] S. Intawichaiy, S. Chaturantabut, An application of randomized singular value decomposition on image reconstruction using least-squares approach, Preprint in Thai Journal of Mathematics, 2018.

(Received 17 June 2019)
(Accepted 24 December 2019)