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

Siriwan Intawichai and Saifon Chaturantabut

Department of Mathematics and Statistics, Thammasat University

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...
been working on discovering its properties and developing algorithms for its computation \cite{3, 4}. SVD is a well-known technique for dimensionality reduction which applied for solving many problems, such as low-rank matrix approximations \cite{5, 6, 7, 8}, which is a good approximation to the original matrix, least squares problems \cite{6}, image processing \cite{7} and model order reduction to compress the high-dimensional data space into a lower-dimensional space \cite{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) \cite{9}, E. Liberty et. al.(2007) \cite{10} and Halko et al. (2011) \cite{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 \cite{10-11}, uniform sampling \cite{16}, \cite{12}, sparse sampling \cite{17, 18}, and sampling with K-means clustering \cite{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 \cite{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
A Numerical Study of Efficient Sampling Strategies for Randomized ... 373

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

$$X = USV^T, \quad (2.1)$$

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 $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$, 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

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

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 $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_k > 0$ on its diagonal entries. For a fixed dimension $k < r$, $X_k$ is the optimal solution of least-squares problem:

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

and the corresponding minimum error is given by $\|X - X_k\|_2^2 = \sum_{i=k+1}^{r} \sigma_i^2$. 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,

$$\hat{X}_k = \hat{U}_k \hat{S}_k \hat{V}_k^T, \quad (2.4)$$

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

Define the random projection of a matrix as, $Y = X \Omega$, where $\Omega$ 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 ma-
trix 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
\[
\|X - QQ^TX\|_2^2 \leq \epsilon.
\] (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 $\Omega$ is a matrix composed of the random vectors.
The second stage of the rSVD method is to compute the SVD of the reduced
matrix $B := Q^TX$. Suppose $B = \tilde{U}\hat{S}\hat{V}^T$ 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 $\hat{U}$ of $X$
from the product $Q\hat{U}$.
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 ma-
trix. 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.

\begin{algorithm}
\caption{The rSVD-QR algorithm [13]}
\begin{tabular}{ll}
\textbf{INPUT} & : A data matrix $X \in \mathbb{R}^{m \times n}$ with target rank $k$
& and an oversampling parameter $p$
\textbf{OUTPUT} & : The rSVD of $X$: $\hat{U}, \hat{S}, \hat{V}$
\end{tabular}
\begin{itemize}
\item Step 1. Draw a random matrix $\Omega$ with dimension $n \times (k + p)$.
\item Step 2. Form the matrix product $Y = X\Omega$.
\item Step 3. Construct $Q$ from the QR decomposition of $Y$.
\item Step 4. Set $B = Q^TX$.
\item Step 5. Compute an SVD of the small matrix: $B = \tilde{U}\hat{S}\hat{V}^T$.
\item Step 6. Set $\hat{U} = Q\hat{U}$.
\end{itemize}
\end{algorithm}

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 $\tilde{S} = \sqrt{D_k} \in \mathbb{R}^{k \times k}$.

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 $\tilde{U} := YT_k \tilde{S}^{-1}$. By setting $B = \tilde{U}^T X$, we can compute the SVD of $B$ so that $B := Q\tilde{S}\tilde{V}^T$. Hence, $U(\tilde{U}^T X) = \tilde{U}B =: \tilde{U}[Q\tilde{S}\tilde{V}^T] =: (\tilde{U}Q)\tilde{S}\tilde{V}^T =: \tilde{U}\tilde{S}\tilde{V}^T \approx X$. 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 $X$: $\tilde{U}, \tilde{S}, \tilde{V}$

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Draw a random matrix $\Omega \in \mathbb{R}^{n \times \ell}$ where $\ell = k + p$.</td>
</tr>
<tr>
<td>2</td>
<td>Form the matrix product $Y = X\Omega$ and then the square matrix $Z = Y^T Y$.</td>
</tr>
<tr>
<td>3</td>
<td>(Ensure symmetry by $Z = \frac{1}{2}(Z + Z^T)$)</td>
</tr>
<tr>
<td>4</td>
<td>Computing the eigendecomposition, $Z = TDT^{-1}$.</td>
</tr>
<tr>
<td>5</td>
<td>Truncating $T_k \in \mathbb{R}^{\ell \times k}$, and calculating $\tilde{S} = \sqrt{D_k}$ where $D_k \in \mathbb{R}^{k \times k}$.</td>
</tr>
<tr>
<td>6</td>
<td>Approximate, $\tilde{U} := YT_k \tilde{S}^{-1}$ then set $B = \tilde{U}^T X$.</td>
</tr>
<tr>
<td>7</td>
<td>Compute an SVD of $B = Q\tilde{S}\tilde{V}$.</td>
</tr>
<tr>
<td>8</td>
<td>Set $\tilde{U} = \tilde{U}Q$.</td>
</tr>
</tbody>
</table>

#### 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 $\Omega$ 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 $B = USV^T$. Finally, we obtain the approximated POD basis or the left singular vectors of $X$ from the product $Q\tilde{U}$. 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 $X$: $\hat{U}, \hat{S}, \hat{V}$

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 $B = Q^TX$.

Step 5. Compute SVD of $B = \tilde{U}\tilde{S}\tilde{V}^T$.

Step 6. Set $\hat{U} = Q\tilde{U}$.

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 $Z = Y^TY$ 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 $\Omega$, 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 $f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$, 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 $f(x) = \frac{1}{b-a}$, for $a \leq x \leq b$ and otherwise $f(x) = 0$. In particular, we use $x \sim U(0, 1)$, 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 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 $\omega_{ij}$ from the following distribution

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

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 $x_1, x_2, \ldots, x_m \in \mathbb{R}^n$, we wish to have $k$ points ($\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$) 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 $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$.

### 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 $\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^n$ be a complete data set and form a matrix of a complete data $X = [x_1, x_2 \ldots x_n] \in \mathbb{R}^{n \times n}$.

Suppose $\hat{x} \in \mathbb{R}^n$ 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 \( e_{c_i}, e_{g_j} \in \mathbb{R}^n \) are the \( c_i \)-th, \( g_j \)-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 \( \hat{x}_c := C^T \bar{x} \in \mathbb{R}^{n_c} \) and \( \hat{x}_g := G^T \bar{x} \in \mathbb{R}^{n_g} \). Then, the known components and the unknown components are given in the vectors \( \hat{x}_c \) and \( \hat{x}_g \), 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 \( \hat{x}_g \) will be approximated by first projecting \( \hat{x} \) onto the column span of a basis matrix \( U \) with rank \( k \)

\[ \hat{x} \approx U a, \quad \text{or} \quad \hat{x}_c \approx U_c a \quad \text{and} \quad \hat{x}_g \approx U_g a, \]

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 \( \hat{x}_c \) are then used to determine the coefficient vector \( a \) through the approximation \( \hat{x}_c \approx U_c a \) from the following least-squares problem:

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

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

\[
\hat{x}_g \approx U_g a = U_g U_c^\dagger \hat{x}_c. \tag{4.2}
\]

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 \( k \leq \text{rank}(\{x_j\}_{j=1}^{n_s}) \)
- Incomplete data \( \tilde{x} \in \mathbb{R}^n \) with known entries in \( \tilde{x}_c \in \mathbb{R}^{n_c} \) and unknown entries in \( \tilde{x}_g \in \mathbb{R}^{n_g} \), where \( n = n_c + n_g \).

**OUTPUT**
- Approximation of \( \tilde{x}_g \)

**Step 1.** Create snapshot matrix \( X = [x_1, \ldots, x_{n_s}] \in \mathbb{R}^{n \times n_s} \) and let \( r = \text{rank}(X) \).

**Step 2.** Construct basis \( U \) of rank \( k \leq r \) for \( x \).

**Step 3.** Find coefficient vector \( a \) from \( \tilde{x}_c \) using least-squares problem in \( \| \tilde{x}_c - U_c a \|_2^2 \).

**Step 4.** Compute the approximation \( \tilde{x}_g \approx U_g a \).

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 $X = [x_1, \ldots, x_n] \in \mathbb{R}^{n \times n_s}$ and $k < r = \text{rank}(X)$. The SVD of $X$ is

$$X = U S V^T,$$

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 $S = \text{diag} (\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{r \times r}$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$. As stated previously, the optimal solution of least-squares problem

$$\min_W \|X - W\|^2_2,$$

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.](image)

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.
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

![Reconstruction error for incomplete data by using rSVD-QR method](image_url)
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.
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


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