Randomized Singular Value Decomposition Algorithm

Ruicheng Ao 1900012179

February 24, 2022

1 Problem description

For a matrix $A \in \mathbb{R}^{m\times n}$, its singular value decomposition (SVD) can be formulated as

$$A = U\Sigma V^\top = \sum_{k=1}^{\min(m,n)} \sigma_k u_k v_k^\top, \tag{1}$$

where $\sigma_1 \geq \sigma_2 \geq \ldots, \sigma_{\min(m,n)} \geq 0$ are so-called singular values of A and U, V have orthogonal columns. In some situations, it is requisited to know the k largest singular values of A. Traditional methods based on numerical algebra can obtain such decomposition with a cost of $\mathcal{O}(mnk)$ [2]. To further reduce the time cost, many randomized procedures are investigated. We mainly focus on two of them, namely, Linear Time SVD [1] and Prototype Randomized SVD [3].

2 Algorithm description

In this section, we give detailed descriptions about the two SVD algorithms mentioned above.

2.1 Linear Time SVD

The linear time SVD approximate $A \in \mathbb{R}^{m\times n}$ with a matrix $C \in \mathbb{R}^{m\times c}$, where c only depends on k. The columns of C are randomly sampled from those of A according to importance. The general framework is given in Algorithm 1.

2.2 Propotype Randomized SVD

The Prototype Randomized SVD method utilizes random sampling to identify a subsapce that captures most characteristics of a matrix. The sampling reduces the computational cost by projection through an orthogonal matrix obtained from the QR decomposition. After that, the reduced matrix is directly manipulated to obtain the decomposition. We demonstrate the framework in Algorithm 2

3 Numerical experiments

In this section, we test the above two algorithms on two datasets to see the efficiency of SVD. Then they are applied to accelerate the matrix completion problem.

Algorithm 1 Linear Time SVD

Input $A \in \mathbb{R}^{m\times n}$, number of samples c

Set importance weight $p_i = \|A_i\|_2^2/\|A\|_F^2$ for sampling
for $l = 1, 2, \ldots, c$ **do**
  Select $i_l \in \{1, 2, \ldots, n\}$ with probability distribution $\mathcal{P}(i_l = j) = p_j$
  Set $C_l = A_{i_l}/\sqrt{cp_{i_l}}$
end for
Compute the $k-$largest sigular values of $C = U_c\Sigma_c V_c^\top$
if Postprocessing **then**
  Set $Y = A^\top U_c$
  Compute QR decomposition $Y = QR$
  Compute SVD for $R^\top = U_r\Sigma_r V_r^\top$
  Set $U_k = U_c U_r, V_k = QV_r$
end if
return The approximate $k-$largest singular values $\Sigma_k = \Sigma_r$ with left and right singular values U_k, V_k.

Algorithm 2 Prototype Randomized SVD

Input $A \in \mathbb{R}^{m\times n}$, preprocessing exponential q, number of additional sampling p

Generate a Gaussian test matrix $\Omega \in \mathbb{R}^{n\times(k+p)}$
Set $Y = (AA^\top)^q A\Omega$
Compute QR decomposition $Y = QR$
Projection $B = Q^\top A$
Compute SVD $B = U_B\Sigma_B V_B T^\top$
Set $U_k = QU_B$
return The $k-$largetst singular values $\Sigma_k = \Sigma_B$ with the left and right vectors $U_k, V_k = V_B$

Algorithm	baseline	Linear time			Prototype randomized		
	-	$c = 2k$	$c = 10k$	$c = 50k$	$q = 0$	$q = 1$	$q = 2$
$k = 5$	$1.19e-02$	$2.04e-02$	$2.92e-02$	$7.15e-02$	$1.24e-01$	$8.52e-02$	$5.07e-02$
$k = 10$	$1.69e-02$	$1.78e-02$	$5.79e-02$	$1.45e-01$	$1.94e-01$	$1.33e-01$	$4.16e-02$
$k = 15$	$1.78e-02$	$2.71e-02$	$1.39e-01$	$1.58e-01$	$1.67e-01$	$1.62e-01$	$6.26e-02$
$k = 20$	$2.02e-02$	$3.79e-02$	$1.81e-01$	$1.75e-01$	$1.92e-01$	$1.78e-01$	$7.03e-02$

Table 1: CPU time for Linear time SVD and Prototype randomized SVD with different parameters and k

Algorithm	Linear time			Prototype randomized		
	$c = 2k$	$c = 10k$	$c = 50k$	$q = 0$	$q = 1$	$q = 2$
$k = 5$	$1.13e-01$	$1.05e-01$	$3.87e-02$	$6.52e-02$	$3.89e-02$	$2.35e-02$
$k = 10$	$7.22e-02$	$4.76e-02$	$2.56e-02$	$7.99e-16$	$8.29e-16$	$8.48e-16$
$k = 15$	$3.88e-02$	$2.76e-02$	$1.06e-02$	$8.34e-16$	$8.38e-16$	$9.88e-16$
$k = 20$	$8.49e-16$	$8.40e-16$	$7.33e-16$	$8.84e-16$	$6.73e-16$	$1.09e-15$

Table 2: Relative error of eigenvalue for Linear time SVD and Prototype randomized SVD with different parameters and k on dataset 1

Synthetic datasets The first dataset is generated by the matlab code

$$
\begin{aligned}
&m = 2048;\\
&n = 512;\\
&p = 20;\\
&A = randn(m, p) \star randn(p, n);
\end{aligned}
$$

The second comes from [3] by utilizing its code from github (not copy directly). We test our algorithms on 2 of the distributions for comparison (required 1). The matrix size is chosen to be 4096×4096 in dataset 2. The parameters are chosen to be $c = 2k, 10k, 50k$ for linear SVD and $q = 0, 1, 2$ for Prototype SVD. The parameter p in Algorithm 2 is set to be k, which doubles the sizes of sampling matrix. For each A, the number of the largest eigenvalues we compute is chosen to be $k = 5, 10, 15, 20$.

The results are displayed as below. Figures 1-12 show the magnitude of each eigenvalues compared with the MATLAB default function *svds*. The tables 1-12 record the relative error of eigenvalues and the corresponding eigen vectors, as well as the CPU time compared with the baseline *svds*.

Algorithm	Linear time			Prototype randomized		
	$c = 2k$	$c = 10k$	$c = 50k$	$q = 0$	$q = 1$	$q = 2$
$k = 5$	$1.31e-02$	$1.27e-02$	$9.22e-03$	$1.19e-02$	$1.05e-02$	$6.48e-03$
$k = 10$	$1.11e-02$	$1.12e-02$	$7.89e-03$	$1.79e-15$	$3.87e-15$	$4.74e-15$
$k = 15$	$1.11e-02$	$1.04e-02$	$6.10e-03$	$1.46e-15$	$1.14e-15$	$1.93e-15$
$k = 20$	$1.44e-15$	$2.72e-15$	$1.27e-15$	$3.52e-15$	$2.20e-15$	$2.48e-15$

Table 3: Relative $\|\cdot\|_F$ error of U for Linear time SVD and Prototype randomized SVD with different parameters and k on dataset 1

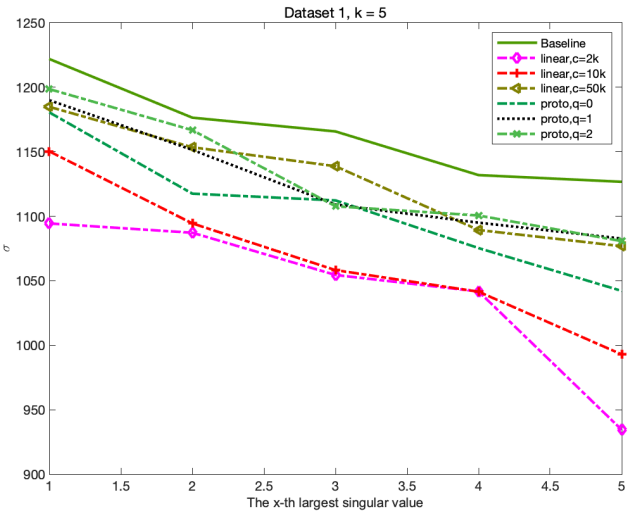

Figure 1: $k = 5$

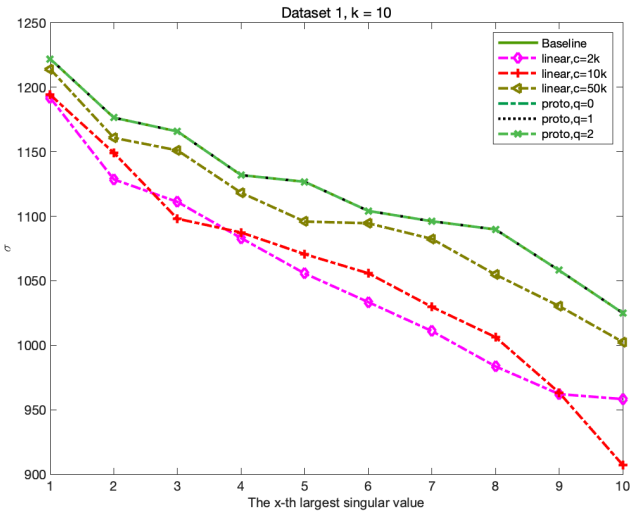

Figure 2: $k = 10$

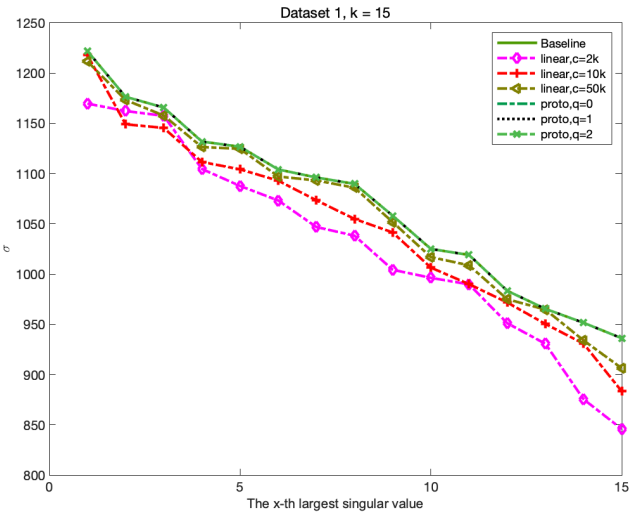

Figure 3: $k = 15$

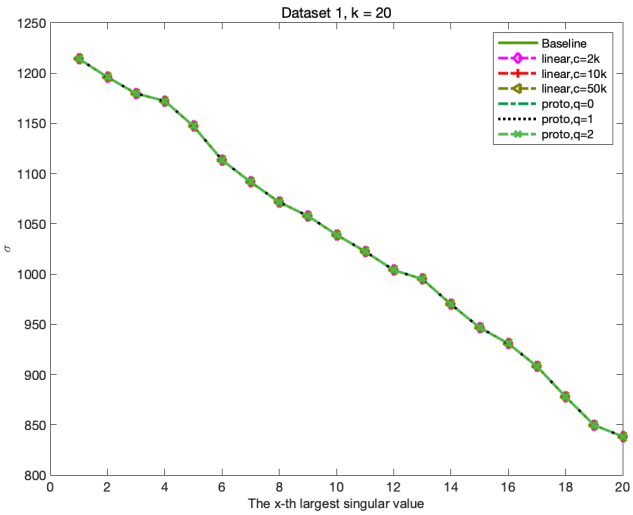

Figure 4: $k = 20$

Results on dataset 1 with different k

Algorithm	Linear time			Prototype randomized		
	$c = 2k$	$c = 10k$	$c = 50k$	$q = 0$	$q = 1$	$q = 2$
$k = 5$	$1.30e-02$	$1.26e-02$	$9.01e-03$	$1.18e-02$	$1.04e-02$	$6.21e-03$
$k = 10$	$1.09e-02$	$1.11e-02$	$7.71e-03$	$1.80e-15$	$3.88e-15$	$4.73e-15$
$k = 15$	$1.10e-02$	$1.03e-02$	$5.99e-03$	$1.47e-15$	$1.15e-15$	$1.93e-15$
$k = 20$	$1.44e-15$	$2.72e-15$	$1.28e-15$	$3.52e-15$	$2.20e-15$	$2.48e-15$

Table 4: Relative $\|\cdot\|_F$ error of V for Linear time SVD and Prototype randomized SVD with different parameters and k on dataset 1

Algorithm	baseline	Linear time			Prototype randomized		
	-	$c = 2k$	$c = 10k$	$c = 50k$	$q = 0$	$q = 1$	$q = 2$
$k = 5$	$1.07e+00$	$1.16e+00$	$1.13e+00$	$2.48e-01$	$7.98e-01$	$7.77e-01$	$2.03e-01$
$k = 10$	$1.24e+00$	$1.22e+00$	$1.34e+00$	$3.68e-01$	$8.75e-01$	$9.35e-01$	$3.79e-01$
$k = 15$	$1.47e+00$	$1.47e+00$	$1.75e+00$	$5.26e-01$	$1.32e+00$	$1.35e+00$	$6.82e-01$
$k = 20$	$1.51e+00$	$1.69e+00$	$1.70e+00$	$7.04e-01$	$1.28e+00$	$1.28e+00$	$6.99e-01$

Table 5: CPU time for Linear time SVD and Prototype randomized SVD with different parameters and k

Algorithm	baseline	Linear time			Prototype randomized		
	-	$c = 2k$	$c = 10k$	$c = 50k$	$q = 0$	$q = 1$	$q = 2$
$k = 5$	$1.16e+00$	$1.16e+00$	$1.13e+00$	$2.38e-01$	$7.66e-01$	$7.72e-01$	$1.95e-01$
$k = 10$	$1.27e+00$	$1.21e+00$	$1.25e+00$	$3.81e-01$	$9.73e-01$	$9.70e-01$	$3.96e-01$
$k = 15$	$1.35e+00$	$1.41e+00$	$1.52e+00$	$5.21e-01$	$1.11e+00$	$1.11e+00$	$5.49e-01$
$k = 20$	$1.37e+00$	$1.54e+00$	$1.71e+00$	$6.81e-01$	$1.35e+00$	$1.22e+00$	$6.84e-01$

Table 6: CPU time for Linear time SVD and Prototype randomized SVD with different parameters and k

Algorithm	Linear time			Prototype randomized		
	$c = 2k$	$c = 10k$	$c = 50k$	$q = 0$	$q = 1$	$q = 2$
$k = 5$	$2.44e-02$	$1.95e-03$	$6.47e-04$	$1.25e-04$	$1.27e-08$	$6.11e-14$
$k = 10$	$2.64e-04$	$1.31e-04$	$2.20e-05$	$5.00e-05$	$1.56e-14$	$5.59e-16$
$k = 15$	$4.13e-04$	$6.66e-05$	$1.29e-05$	$2.24e-04$	$1.06e-09$	$2.12e-07$
$k = 20$	$3.57e-04$	$2.19e-04$	$1.06e-04$	$3.03e-04$	$2.37e-05$	$2.31e-04$

Table 7: Relative error of eigenvalue for Linear time SVD and Prototype randomized SVD with different parameters and k on dataset 2 with distribution 1

Algorithm	Linear time			Prototype randomized		
	$c = 2k$	$c = 10k$	$c = 50k$	$q = 0$	$q = 1$	$q = 2$
$k = 5$	$6.24e-03$	$2.44e-03$	$8.57e-04$	$3.59e-03$	$1.53e-06$	$1.28e-08$
$k = 10$	$3.45e-03$	$1.62e-03$	$2.88e-04$	$4.59e-04$	$1.59e-06$	$1.47e-09$
$k = 15$	$1.75e-03$	$2.79e-04$	$5.37e-04$	$2.60e-04$	$3.27e-07$	$1.40e-09$
$k = 20$	$1.23e-03$	$5.21e-04$	$1.41e-04$	$1.88e-04$	$3.90e-07$	$3.91e-09$

Table 8: Relative error of eigenvalue for Linear time SVD and Prototype randomized SVD with different parameters and k on dataset 2 with distribution 2

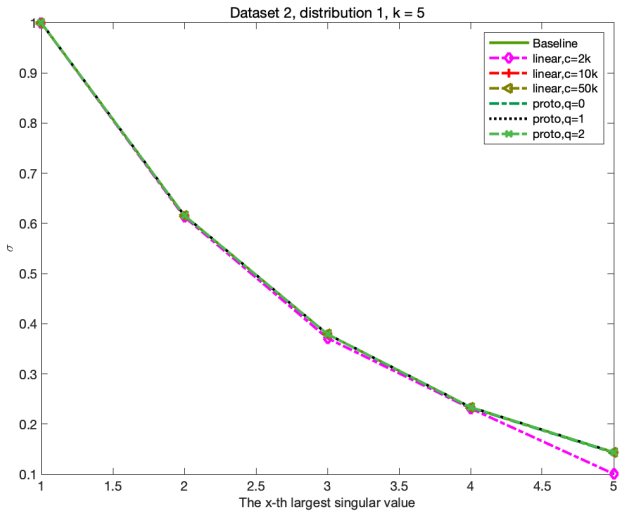

Figure 5: $k = 5$

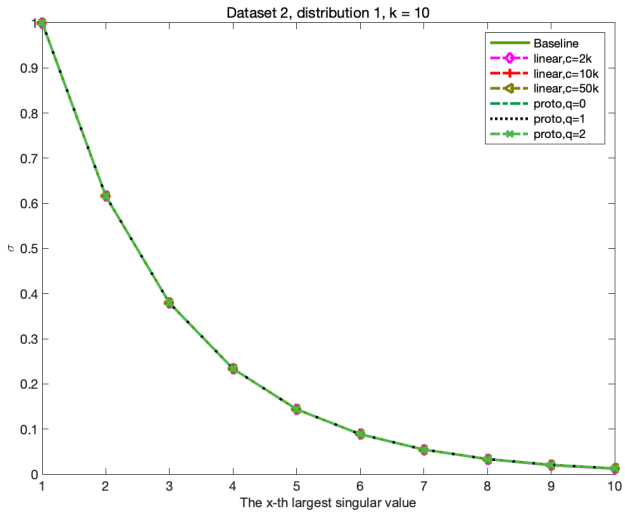

Figure 6: $k = 10$

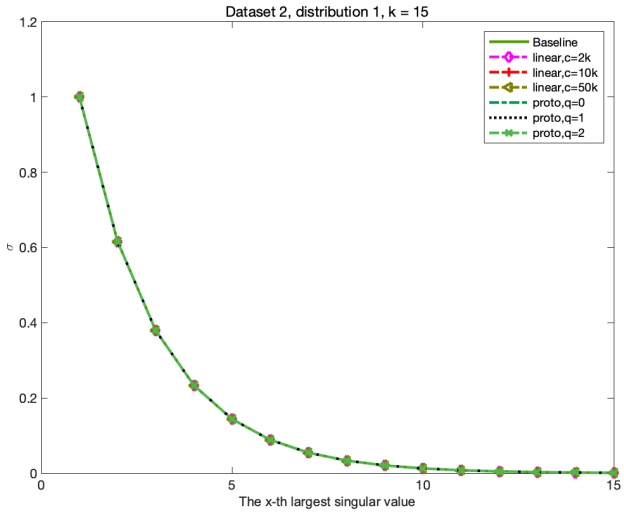

Figure 7: $k = 15$

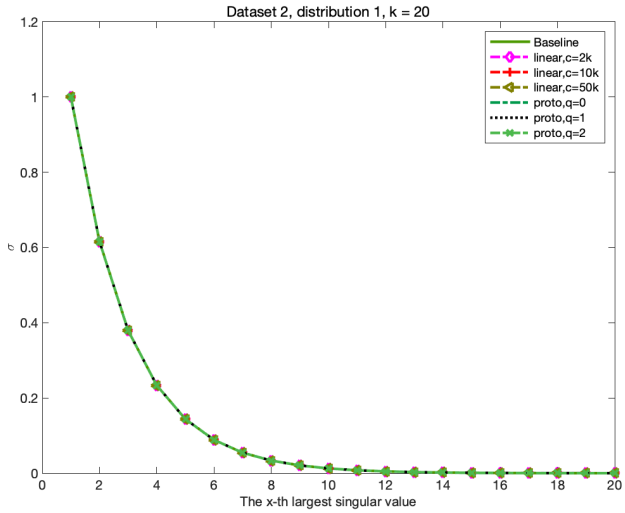

Figure 8: $k = 20$

Results on dataset 2, distribution 1 with different k

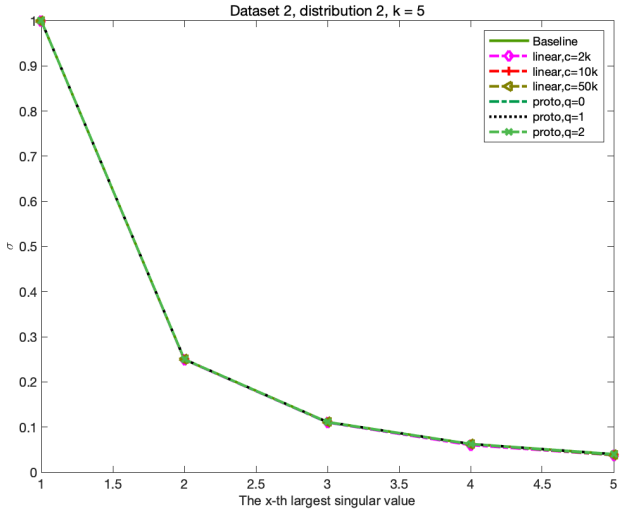

Figure 9: $k = 5$

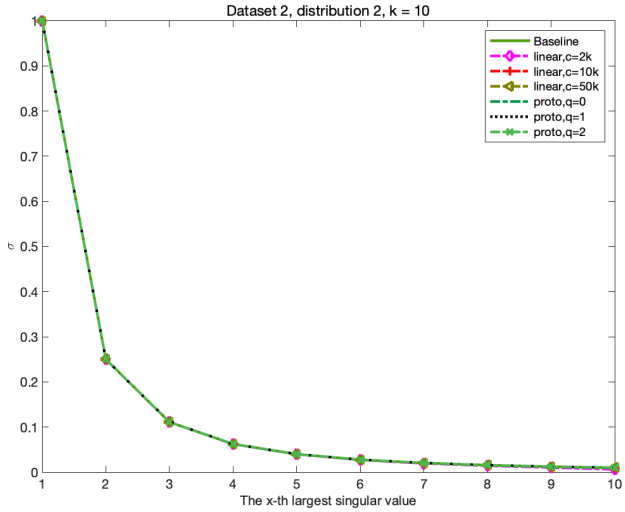

Figure 10: $k = 10$

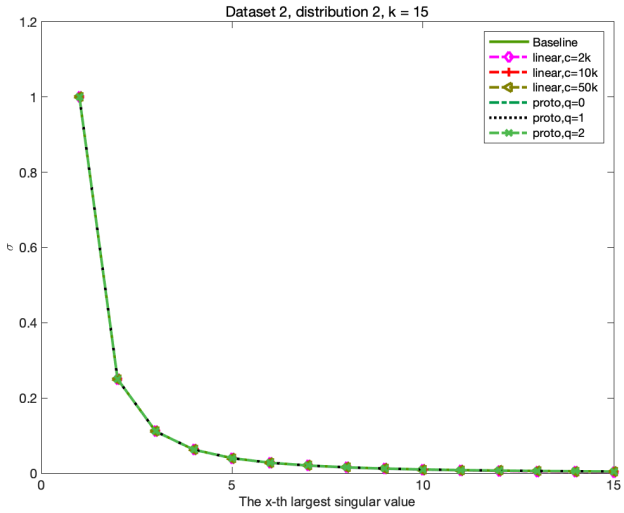

Figure 11: $k = 15$

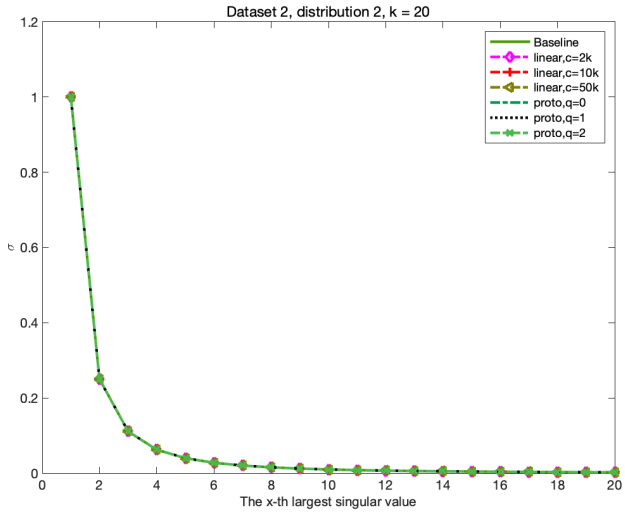

Figure 12: $k = 20$

Results on dataset 2, distribution 1 with different k

Algorithm	Linear time			Prototype randomized		
	$c = 2k$	$c = 10k$	$c = 50k$	$q = 0$	$q = 1$	$q = 2$
$k = 5$	$3.63e-03$	$1.07e-03$	$4.30e-04$	$2.34e-04$	$2.12e-06$	$4.65e-09$
$k = 10$	$8.07e-04$	$6.64e-04$	$1.78e-04$	$3.50e-04$	$5.90e-09$	$5.84e-10$
$k = 15$	$2.84e-03$	$1.15e-03$	$4.97e-04$	$2.07e-03$	$4.08e-06$	$5.67e-05$
$k = 20$	$6.30e-03$	$5.64e-03$	$4.72e-03$	$5.75e-03$	$3.16e-03$	$5.63e-03$

Table 9: Relative $\|\cdot\|_F$ error of U for Linear time SVD and Prototype randomized SVD with different parameters and k on dataset 2 with distribution 1

Algorithm	Linear time			Prototype randomized		
	$c = 2k$	$c = 10k$	$c = 50k$	$q = 0$	$q = 1$	$q = 2$
$k = 5$	$2.80e-03$	$2.17e-03$	$1.10e-03$	$2.01e-03$	$4.12e-05$	$3.77e-06$
$k = 10$	$4.93e-03$	$3.64e-03$	$1.17e-03$	$1.31e-03$	$7.05e-05$	$1.98e-06$
$k = 15$	$5.81e-03$	$1.55e-03$	$2.91e-03$	$1.30e-03$	$3.96e-05$	$2.38e-06$
$k = 20$	$5.81e-03$	$4.73e-03$	$1.44e-03$	$1.37e-03$	$5.22e-05$	$4.69e-06$

Table 10: Relative $\|\cdot\|_F$ error of U for Linear time SVD and Prototype randomized SVD with different parameters and k on dataset 2 with distribution 2

Algorithm	Linear time			Prototype randomized		
	$c = 2k$	$c = 10k$	$c = 50k$	$q = 0$	$q = 1$	$q = 2$
$k = 5$	$2.02e-03$	$6.35e-04$	$1.45e-04$	$1.88e-05$	$1.21e-07$	$2.57e-10$
$k = 10$	$2.49e-04$	$3.59e-04$	$4.65e-05$	$1.96e-05$	$2.99e-11$	$8.94e-12$
$k = 15$	$7.72e-04$	$2.90e-04$	$3.74e-05$	$1.95e-04$	$2.03e-07$	$5.80e-06$
$k = 20$	$4.82e-03$	$4.45e-03$	$4.33e-03$	$3.74e-03$	$3.01e-03$	$4.60e-03$

Table 11: Relative $\|\cdot\|_F$ error of V for Linear time SVD and Prototype randomized SVD with different parameters and k on dataset 2 with distribution 1

Algorithm	Linear time			Prototype randomized		
	$c = 2k$	$c = 10k$	$c = 50k$	$q = 0$	$q = 1$	$q = 2$
$k = 5$	$8.59e-04$	$1.35e-03$	$5.10e-04$	$4.77e-04$	$8.59e-06$	$9.38e-07$
$k = 10$	$4.51e-03$	$3.27e-03$	$6.96e-04$	$5.06e-04$	$1.96e-05$	$4.43e-07$
$k = 15$	$5.37e-03$	$9.52e-04$	$2.66e-03$	$4.98e-04$	$9.63e-06$	$4.68e-07$
$k = 20$	$5.32e-03$	$4.53e-03$	$1.05e-03$	$5.52e-04$	$1.18e-05$	$1.18e-06$

Table 12: Relative $\|\cdot\|_F$ error of V for Linear time SVD and Prototype randomized SVD with different parameters and k on dataset 2 with distribution 2

We observe that the Linear time SVD works not very well on dataset 1 when c is small, however, it shows good performance when k or c is large. Both algorithms behave well on dataset 2, from that the tables show that they can exactly recover the eigenvalues even when $k = 5$. An explanation for better behavior when k is larger is that larger k results in larger sample sets, which leads to better simulation of A. Another interpretation is that the rank of A is exactly 20 in the first dataset, thus when $k \geq 10$, the restoration of the information of A is complete for the Prototype SVD and $k = 20$ for Linear time SVD.

On the other hand, we find that for problems of small scale (rank) like dataset 1, the random SVD has little advantage, while for much larger matrix in dataset 2, the random algorithms outperform the baseline MATLAB SVD, which is corresponding to our intuition that, it is hard to defeat MATLAB when the problem is of small scale, while we can do much better for larger problem.

References

[1] Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM Journal on computing*, 36(1):158–183, 2006.

[2] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

[3] 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.