Singular Value Decomposition

*Project 3 on the Course "Algorithms for Big Data Analysis".

1st Chen Yihang

Peking University

1700010780

Abstract

LinearTime SVD and Prototype for randomized SVD methods are implemented and tested on various datasets. Both method favors the singular vector decreases rapidly. Based on the idea of randomized SVD, SVT (Singular Value Thresholding) method is implemented and tested on a real-life image recovery problem. Satisfactory results are produced.

CONTENTS

I. Randomized SVD techniques

A. LinearTime SVD

The algorithm in [Drineas et al., 2006] can be summarized below, I implemented it in "lineartime.m". The estimated $\mathbf{A}$ is $\mathbf{H}_k'\mathbf{H}_k\mathbf{A}$. According to Theorem 4 in Section 4 in [Drineas et al., 2006], we set

$$p_i = \frac{\|\mathbf{A}^{(i)}\|_2^2}{\|\mathbf{A}\|_F}, \quad c/k = const$$

LinearTime SVD

Given an $m \times n$ matrix $\mathbf{A}$, a target number k of singular vectors, a integer c such that $k \leq c \leq n$, a probability vector $\{p_i\}_{i=1}^n$, this procedure computes an approximate rank-k factorization $\mathbf{U\Sigma V}^\star$, where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal, and $\mathbf{\Sigma}$ is nonnegative and diagonal.

1 **for** $t = 1$ to c.
2 Pick $i_t \in 1, \ldots, n$ with $\mathbb{P}(i_t = \alpha) = p_\alpha, \alpha = 1, \ldots, n$.
3 Set $\mathbf{C}^{(t)} = \mathbf{A}^{i_t}/\sqrt{cp_{i_t}}$
4 **end**
5 Compute the SVD of $\mathbf{C}^T\mathbf{C} = \sum_{t=1}^c \sigma_t^2(\mathbf{C}) y_t y_t^T$.
6 Compute $h_t = \mathbf{C}y_t/\sigma_t(\mathbf{C})$, for $t = 1, \ldots, k$.
7 Return $\mathbf{H}_k$, where $\mathbf{H}_k^{(t)} = h_t$, and $\sigma_t(\mathbf{C}), t = 1, \ldots, k$.

B. Prototype for Randomized SVD

The algorithm in [Halko et al., 2011] can be summarized below, I implemented it in prototype.m. The estimated $\mathbf{A}$ is $\mathbf{Q'QA}$.

II. Matrix Completion via SVD

We have an $n_1 \times n_2$ matrix $\boldsymbol{M}$ with observed entries indexed by the set Ω. Following [Mazumder et al., 2010] we denote the projection $\mathcal{P}_\Omega$ to be the be the $n_1 \times n_2$ matrix with the observed elements of $\boldsymbol{M}$ preserved, and the missing entries replaced with 0. Likewise $\mathcal{P}_\Omega^\perp$ projects onto the complement of the set Ω. We want to solve

$$\begin{aligned} &\text{minimize} && \|\boldsymbol{X}\|_* \\ &\text{subject to} && X_{ij} = M_{ij}, \quad (i,j) \in \Omega, \end{aligned} \tag{II.1}$$

PROTOTYPE FOR RANDOMIZED SVD

Given an $m \times n$ matrix $\mathbf{A}$, a target number k of singular vectors, and an exponent q (say $q = 1$ or $q = 2$), this procedure computes an approximate rank-$2k$ factorization $\mathbf{U\Sigma V}^\star$, where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal, and $\mathbf{\Sigma}$ is nonnegative and diagonal.

Stage A1: Randomized Power Iteration

1 Generate an $n \times 2k$ Gaussian test matrix $\mathbf{\Omega}$.
2 Form $\mathbf{Y} = (\mathbf{AA}^\star)^q\mathbf{A\Omega}$ by multiplying alternately with $\mathbf{A}$ and $\mathbf{A}^\star$.
3 Construct a matrix $\mathbf{Q}$ whose columns form an orthonormal basis for the range of $\mathbf{Y}$.

Stage A2: Randomized Subspace Iteration

1 Generate an $n \times 2k$ Gaussian test matrix $\mathbf{\Omega}$.
2 Form $\mathbf{Y_0} = \mathbf{A\Omega}$ and compute its QR factorization $\mathbf{Y_0} = \mathbf{Q_0R_0}$.
3 **for** $j = 1, 2, \ldots, q$
4 Form $\widetilde{\mathbf{Y_j}} = \mathbf{A}^\star\mathbf{Q}_{j-1}$ and compute its QR factorization $\widetilde{\mathbf{Y_j}} = \widetilde{\mathbf{Q_j}}\widetilde{\mathbf{R_j}}$.
5 Form $\mathbf{Y_j} = \mathbf{A}\widetilde{\mathbf{Q}_{j-1}}$ and compute its QR factorization $\mathbf{Y_j} = \mathbf{Q_jR_j}$.
6 **end**
7 $\mathbf{Q} = \mathbf{Q}_q$

Stage B:

1 Form $\mathbf{B} = \mathbf{Q}^\star\mathbf{A}$.
2 Compute an SVD of the small matrix: $\mathbf{B} = \widetilde{\mathbf{U}}\mathbf{\Sigma V}^\star$.
3 Set $\mathbf{U} = \mathbf{Q}\widetilde{\mathbf{U}}$.

Note: The Stage A1 or A2 computes an $m \times 2k$ matrix $\mathbf{Q}$ whose range approximates the range of $\mathbf{A}$. The computation of $\mathbf{Y}$ in Step 2 in Stage A1 is vulnerable to round-off errors. When high accuracy is required, we must incorporate an orthonormalization step between each application of $\mathbf{A}$ and $\mathbf{A}^\star$; i.e. we adopt Stage A2.

Hence, we are going to minimize

$$\min_X \frac{1}{2}\|\mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X})\|_F^2 + \mu\|\boldsymbol{X}\|_\star \tag{II.2}$$

where the nuclear norm $\|\boldsymbol{X}\|_\star$ is the sum of the singular values of $\boldsymbol{X}$ (a convex relaxation of the rank).

A. The singular value shrinkage operator

Consider the singular value decomposition (SVD) of a matrix $\boldsymbol{X} \in \mathbb{R}^{n_1 \times n_2}$ of rank r

$$\boldsymbol{X} = \boldsymbol{U\Sigma V}^*, \quad \boldsymbol{\Sigma} = diag(\{\sigma_i\}_{1 \le i \le r}), \tag{II.3}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are respectively $n_1 \times r$ and $n_2 \times r$ matrices with orthonormal columns, and the singular values σ_i are positive (unless specified otherwise, we will always assume that the SVD of a matrix is given in the reduced form above). For each $\mu \geq 0$, we introduce the soft-thresholding operator $\mathcal{D}_\mu$ defined as follows:

$$\mathcal{D}_\mu(\boldsymbol{X}) := \boldsymbol{U}\mathcal{D}_\mu(\boldsymbol{\Sigma})\boldsymbol{V}^*, \quad \mathcal{D}_\mu(\boldsymbol{\Sigma}) = diag(\{\sigma_i - \mu)_+\}), \tag{II.4}$$

where t_+ is the positive part of t, namely, $t_+ = \max(0, t)$. In words, this operator simply applies a soft-thresholding rule to the singular values of $\boldsymbol{X}$, effectively shrinking these towards zero. This is the reason why we will also refer to this transformation as the *singular value shrinkage* operator. Even though the SVD may not be unique, it is easy to see that the singular value shrinkage operator is well defined and we do not elaborate further on this issue. In some sense, this shrinkage operator is a straightforward extension of the soft-thresholding rule for scalars and vectors. In particular, note that if many of the singular values of $\boldsymbol{X}$ are below the threshold μ, the rank of $\mathcal{D}_\mu(\boldsymbol{X})$ may be considerably lower than that of $\boldsymbol{X}$, just like the soft-thresholding rule applied to vectors leads to sparser outputs whenever some entries of the input are below threshold.

Theorem 1. For each $\mu \geq 0$ and $\boldsymbol{Y} \in \mathbb{R}^{n_1 \times n_2}$, the singular value shrinkage operator (II.4) obeys

$$\mathcal{D}_\mu(\boldsymbol{Y}) = \arg\min_{\boldsymbol{X}} \left\{ \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \mu\|\boldsymbol{X}\|_* \right\}. \tag{II.5}$$

Proof. Since the function $h_0(\boldsymbol{X}) := \mu\|\boldsymbol{X}\|_* + \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{Y}\|_F^2$ is strictly convex, it is easy to see that there exists a unique minimizer, and we thus need to prove that it is equal to $\mathcal{D}_\mu(\boldsymbol{Y})$. To do this, recall the definition of a subgradient of a convex function $f : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}$. We say that $\boldsymbol{Z}$ is a subgradient of f at $\boldsymbol{X}_0$, denoted $\boldsymbol{Z} \in \partial f(\boldsymbol{X}_0)$, if

$$f(\boldsymbol{X}) \geq f(\boldsymbol{X}_0) + (\boldsymbol{Z}, \boldsymbol{X} - \boldsymbol{X}_0) \tag{II.6}$$

for all $\boldsymbol{X}$. Now $\hat{\boldsymbol{X}}$ minimizes h_0 if and only if $\mathbf{0}$ is a subgradient of the functional h_0 at the point $\hat{\boldsymbol{X}}$, i.e.

$$\mathbf{0} \in \hat{\boldsymbol{X}} - \boldsymbol{Y} + \mu\partial\|\hat{\boldsymbol{X}}\|_*, \tag{II.7}$$

where $\partial\|\hat{\boldsymbol{X}}\|_*$ is the set of subgradients of the nuclear norm. Let $\boldsymbol{X} \in \mathbb{R}^{n_1 \times n_2}$ be an

arbitrary matrix and $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$ be its SVD. It is known that

$$\partial\|\boldsymbol{X}\|_* = \left\{\boldsymbol{U}\boldsymbol{V}^* + \boldsymbol{W} : \ \boldsymbol{W} \in \mathbb{R}^{n_1\times n_2}, \ \ \boldsymbol{U}^*\boldsymbol{W} = 0, \ \ \boldsymbol{W}\boldsymbol{V} = 0, \ \ \|\boldsymbol{W}\|_2 \leq 1\right\}. \tag{II.8}$$

Set $\hat{\boldsymbol{X}} := \mathcal{D}_\mu(\boldsymbol{Y})$ for short. In order to show that $\hat{\boldsymbol{X}}$ obeys (II.7), decompose the SVD of $\boldsymbol{Y}$ as

$$\boldsymbol{Y} = \boldsymbol{U}_0\boldsymbol{\Sigma}_0\boldsymbol{V}_0^* + \boldsymbol{U}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^*,$$

where $\boldsymbol{U}_0$, $\boldsymbol{V}_0$ (resp. $\boldsymbol{U}_1$, $\boldsymbol{V}_1$) are the singular vectors associated with singular values greater than μ (resp. smaller than or equal to μ). With these notations, we have

$$\hat{\boldsymbol{X}} = \boldsymbol{U}_0(\boldsymbol{\Sigma}_0 - \mu\boldsymbol{I})\boldsymbol{V}_0^*$$

and, therefore,

$$\boldsymbol{Y} - \hat{\boldsymbol{X}} = \mu(\boldsymbol{U}_0\boldsymbol{V}_0^* + \boldsymbol{W}), \quad \boldsymbol{W} = \mu^{-1}\boldsymbol{U}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^*.$$

By definition, $\boldsymbol{U}_0^*\boldsymbol{W} = 0$, $\boldsymbol{W}\boldsymbol{V}_0 = 0$ and since the diagonal elements of $\boldsymbol{\Sigma}_1$ have magnitudes bounded by μ, we also have $\|\boldsymbol{W}\|_2 \leq 1$. Hence $\boldsymbol{Y} - \hat{\boldsymbol{X}} \in \mu\partial\|\hat{\boldsymbol{X}}\|_*$, which concludes the proof. □

B. Shrinkage iterations

We are now in the position to introduce the singular value thresholding algorithm. Fix $\tau > 0$ and a sequence $\{\delta_k\}$ of positive step sizes. Starting with $\boldsymbol{Y}_0$, inductively define for $k = 1, 2, \ldots,$

$$\begin{cases} \boldsymbol{X}^k = \mathcal{D}_\tau(\boldsymbol{Y}^{k-1}), \\ \boldsymbol{Y}^k = \boldsymbol{Y}^{k-1} + \delta_k\mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X}^k) \end{cases} \tag{II.9}$$

until a stopping criterion is reached. This shrinkage iteration is very simple to implement. At each step, we only need to compute an SVD and perform elementary matrix operations.

- *Low-rank property.* A remarkable empirical fact is that the matrices in the sequence $\{\boldsymbol{X}^k\}$ have low rank The reason for this phenomenon is, however, simple: because we are interested in large values of μ, the thresholding step happens to 'kill' most of the small singular values and produces a low-rank output. In fact, our numerical

results show that the rank of $\boldsymbol{M}^k$ is nondecreasing with k, and the maximum rank is reached in the last steps of the algorithm.

Thus, when the rank of the solution is substantially smaller than either dimension of the matrix, the storage requirement is low since we could store each $\boldsymbol{M}^k$ in its SVD form (note that we only need to keep the current iterate and may discard earlier values).

- *Sparsity.* Another important property of the soft impute algorithm is that the iteration matrix $\boldsymbol{Y}^k$ is sparse. Since $\boldsymbol{Y}^0 = \boldsymbol{0}$, we have by induction that $\boldsymbol{Y}^k$ vanishes outside of Ω. The fewer entries available, the sparser $\boldsymbol{Y}^k$. Because the sparsity pattern Ω is fixed throughout, one can then apply sparse matrix techniques to save storage. Also, if $|\Omega| = m$, the computational cost of updating $\boldsymbol{Y}^k$ is of order m. Moreover, we can call subroutines supporting sparse matrix computations, which can further reduce computational costs.

  One such subroutine is the SVD. However, note that we do not need to compute the entire SVD of $\boldsymbol{Y}^k$ to apply the singular value thresholding operator. Only the part corresponding to singular values greater than μ is needed. Hence, a good strategy is to apply the iterative Lanczos algorithm to compute the first few singular values and singular vectors. Because $\boldsymbol{Y}^k$ is sparse, $\boldsymbol{Y}^k$ can be applied to arbitrary vectors rapidly, and this procedure offers a considerable speedup over naive methods.

C. Interpretation as a Lagrange multiplier method

In this section, we recast the SVT algorithm as a type of Lagrange multiplier algorithm known as Uzawa's algorithm. An important consequence is that this will allow us to extend the SVT algorithm to other problems involving the minimization of the nuclear norm under convex constraints.

In what follows, we set $f_\tau(\boldsymbol{X}) = \tau\|\boldsymbol{X}\|_* + \frac{1}{2}\|\boldsymbol{X}\|_F^2$ for some fixed $\tau > 0$ and recall that we wish to solve

$$\begin{aligned} &\text{minimize} && f_\tau(\boldsymbol{X}) \\ &\text{subject to} && \mathcal{P}_\Omega(\boldsymbol{X}) = \mathcal{P}_\Omega(\boldsymbol{M}). \end{aligned}$$

The Lagrangian for this problem is given by

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}) = f_\tau(\boldsymbol{X}) + (\boldsymbol{Y}, \mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X})),$$

where $\boldsymbol{Y} \in \mathbb{R}^{n_1 \times n_2}$. Strong duality holds and $\boldsymbol{X}^\star$ and $\boldsymbol{Y}^\star$ are primal-dual optimal if $(\boldsymbol{X}^\star, \boldsymbol{Y}^\star)$ is a saddlepoint of the Lagrangian $\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y})$, i.e. a pair obeying

$$\sup_{\boldsymbol{Y}} \inf_{\boldsymbol{X}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}) = \mathcal{L}(\boldsymbol{X}^\star, \boldsymbol{Y}^\star) = \inf_{\boldsymbol{X}} \sup_{\boldsymbol{Y}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}). \tag{II.10}$$

(The function $g_0(\boldsymbol{Y}) = \inf_{\boldsymbol{X}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y})$ is called the dual function.) Uzawa's algorithm approaches the problem of finding a saddlepoint with an iterative procedure. From $\boldsymbol{Y}_0 = \boldsymbol{0}$, say, inductively define

$$\begin{cases} \mathcal{L}(\boldsymbol{X}^k, \boldsymbol{Y}^{k-1}) = \min_{\boldsymbol{X}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}^{k-1}) \\ \boldsymbol{Y}^k = \boldsymbol{Y}^{k-1} + \delta_k \mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X}^k), \end{cases} \tag{II.11}$$

where $\{\delta_k\}_{k \geq 1}$ is a sequence of positive step sizes. Uzawa's algorithm is, in fact, a subgradient method applied to the dual problem, where each step moves the current iterate in the direction of the gradient or of a subgradient. Indeed, observe that

$$\partial_{\boldsymbol{Y}} g_0(\boldsymbol{Y}) = \partial_{\boldsymbol{Y}} \mathcal{L}(\tilde{\boldsymbol{X}}, \boldsymbol{Y}) = \mathcal{P}_\Omega(\boldsymbol{M} - \tilde{\boldsymbol{X}}), \tag{II.12}$$

where $\tilde{\boldsymbol{X}}$ is the minimizer of the Lagrangian for that value of $\boldsymbol{Y}$ so that a gradient descent update for $\boldsymbol{Y}$ is of the form

$$\boldsymbol{Y}^k = \boldsymbol{Y}^{k-1} + \delta_k \partial_{\boldsymbol{Y}} g_0(\boldsymbol{Y}^{k-1}) = \boldsymbol{Y}^{k-1} + \delta_k \mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X}^k).$$

It remains to compute the minimizer of the Lagrangian (II.11), and note that

$$\arg\min \ f_\tau(\boldsymbol{X}) + (\boldsymbol{Y}, \mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X})) = \arg\min \ \tau \|\boldsymbol{X}\|_* + \frac{1}{2}\|\boldsymbol{X} - \mathcal{P}_\Omega \boldsymbol{Y}\|_F^2. \tag{II.13}$$

However, we know that the minimizer is given by $\mathcal{D}_\tau(\mathcal{P}_\Omega(\boldsymbol{Y}))$ and since $\boldsymbol{Y}^k = \mathcal{P}_\Omega(\boldsymbol{Y}^k)$ for all $k \geq 0$, Uzawa's algorithm takes the form

$$\begin{cases} \boldsymbol{X}^k = \mathcal{D}_\tau(\boldsymbol{Y}^{k-1}) \\ \boldsymbol{Y}^k = \boldsymbol{Y}^{k-1} + \delta_k \mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X}^k), \end{cases}$$

which is exactly the update (II.9). This point of view brings to bear many different mathematical tools for proving the convergence of the singular value thresholding iterations.

D. Numerical experiments

The settings are identical to the reference [Mazumder et al., 2010].

a) Step sizes: In our experiments, we use

$$\delta = 1.2\,\frac{n_1 n_2}{m}, \tag{II.14}$$

We adopt the weight decay as $\delta_k + 1 = \delta/1.005$. i.e. 1.2 times the undersampling ratio.

b) Initial steps: The SVT algorithm starts with $\boldsymbol{Y}^0 = \boldsymbol{0}$, and we want to choose a large τ to make sure that the solution of (II.2) is close enough to a solution of (II.1). Define k_0 as that integer obeying

$$\frac{\tau}{\delta \|\mathcal{P}_\Omega(\boldsymbol{M})\|_2} \in (k_0 - 1, k_0]. \tag{II.15}$$

Since $\boldsymbol{Y}^0 = \boldsymbol{0}$, it is not difficult to see that

$$\boldsymbol{X}^k = \boldsymbol{0}, \quad \boldsymbol{Y}^k = k\delta\,\mathcal{P}_\Omega(\boldsymbol{M}), \quad k = 1, \ldots, k_0.$$

To save work, we may simply skip the computations of $\boldsymbol{X}^1, \ldots, \boldsymbol{X}^{k_0}$, and start the iteration by computing $\boldsymbol{X}^{k_0+1}$ from $\boldsymbol{Y}^{k_0}$.

This strategy is a special case of a *kicking device*; the main idea of such a kicking scheme is that one can 'jump over' a few steps whenever possible. Just like in the aforementioned reference, we can develop similar kicking strategies here as well. Because in our numerical experiments the kicking is rarely triggered, we forgo the description of such strategies.

c) Stopping criteria: Here, we discuss stopping criteria motivated by the first-order optimality conditions or KKT conditions tailored to the minimization problem (II.2). By (II.13) and letting $\partial_{\boldsymbol{Y}} g_0(\boldsymbol{Y}) = \boldsymbol{0}$ in (II.12), we see that the solution $\boldsymbol{X}_\tau^\star$ to (II.2) must also verify

$$\begin{cases} \boldsymbol{X} = \mathcal{D}_\tau(\boldsymbol{Y}), \\ \mathcal{P}_\Omega(\boldsymbol{X} - \boldsymbol{M}) = \boldsymbol{0}, \end{cases} \tag{II.16}$$

where $\boldsymbol{Y}$ is a matrix vanishing outside of Ω^c. Therefore, to make sure that $\boldsymbol{X}^k$ is close to $\boldsymbol{X}_\tau^\star$, it is sufficient to check how close $(\boldsymbol{X}^k, \boldsymbol{Y}^{k-1})$ is to obeying (II.16). By definition, the first equation in (II.16) is always true. Therefore, it is natural to stop (II.9) when the

error in the second equation is below a specified tolerance. We suggest stopping the algorithm when

$$\frac{\|\mathcal{P}_\Omega(\boldsymbol{X}^k - \boldsymbol{M})\|_F}{\|\mathcal{P}_\Omega(\boldsymbol{M})\|_F} \leq \epsilon, \tag{II.17}$$

where ϵ is a fixed tolerance, e.g. 10^{-4}. We provide a short heuristic argument justifying this choice below.

Consider a fixed matrix $\boldsymbol{A} \in \mathbb{R}^{n_1 \times n_2}$. Under the assumption that the column and row spaces of $\boldsymbol{A}$ are not well aligned with the vectors taken from the canonical basis of $\mathbb{R}^{n_1}$ and $\mathbb{R}^{n_2}$ respectively—the *incoherence assumption* in [Candès and Recht, 2009]—then with very large probability over the choices of Ω, we have

$$(1-\epsilon)p\,\|\boldsymbol{A}\|_F^2 \leq \|\mathcal{P}_\Omega(\boldsymbol{A})\|_F^2 \leq (1+\epsilon)p\,\|\boldsymbol{A}\|_F^2, \quad p := m/(n_1 n_2), \tag{II.18}$$

provided that the rank of $\boldsymbol{A}$ is not too large. The probability model is that Ω is a set of sampled entries of cardinality m sampled uniformly at random so that all the choices are equally likely. In (II.18), we want to think of ϵ as a small constant, e.g. smaller than $1/2$. In other words, the 'energy' of $\boldsymbol{A}$ on Ω (the set of sampled entries) is just about proportional to the size of Ω. The near isometry (II.18) is a consequence of Theorem 4.1 in [Candès and Recht, 2009], and we omit the details.

In the matrix completion problem, we know that under suitable assumptions

$$\|\mathcal{P}_\Omega(\boldsymbol{M})\|_F^2 \asymp p\,\|\boldsymbol{M}\|_F^2,$$

which is just (II.18) applied to the fixed matrix $\boldsymbol{M}$ (the symbol $\asymp$ here means that there is a constant ϵ as in (II.18)). Suppose we could also apply (II.18) to the matrix $\boldsymbol{X}^k - \boldsymbol{M}$ (which we rigorously cannot since $\boldsymbol{X}^k$ depends on Ω), then we would have

$$\|\mathcal{P}_\Omega(\boldsymbol{X}^k - \boldsymbol{M})\|_F^2 \asymp p\,\|\boldsymbol{X}^k - \boldsymbol{M}\|_F^2, \tag{II.19}$$

and thus

$$\frac{\|\mathcal{P}_\Omega(\boldsymbol{X}^k - \boldsymbol{M})\|_F}{\|\mathcal{P}_\Omega(\boldsymbol{M})\|_F} \asymp \frac{\|\boldsymbol{X}^k - \boldsymbol{M}\|_F}{\|\boldsymbol{M}\|_F}.$$

In words, one would control the relative reconstruction error by controlling the relative error on the set of sampled locations.

Algorithm 1: Singular Value Thresholding (SVT) Algorithm

Input: sampled set Ω and sampled entries $\mathcal{P}_\Omega(\boldsymbol{M})$, step size δ, tolerance ϵ, parameter τ, increment ℓ, maximum iteration count $k_{\max}$, elementwize lower bound of $\boldsymbol{X}$ ub, elementwize upper bound of $\boldsymbol{X}$ lb
Output: $\boldsymbol{X}^{\text{opt}}$
Description: Recover a low-rank matrix $\boldsymbol{M}$ from a subset of sampled entries

(1) Set $\boldsymbol{Y}^0 = k_0 \delta \mathcal{P}_\Omega(\boldsymbol{M})$ (k_0 is defined in (II.15))
(2) Set $r_0 = 0, \delta_0 = 0$
(3) **for** $k = 1$ **to** $k_{\max}$
(4) Set $s_k = r_{k-1} + 1$
(5) Set $\delta_k = \delta_{k-1}/1.001$
(6) **repeat**
(7) Compute $[\boldsymbol{U}^{k-1}, \boldsymbol{\Sigma}^{k-1}, \boldsymbol{V}^{k-1}]_{s_k}$
(8) Set $s_k = s_k + \ell$
(9) **until** $\sigma^{k-1}_{s_k - \ell} \leq \tau$
(10) Set $r_k = \max\{j : \sigma_j^{k-1} > \tau\}$
(11) Set $\boldsymbol{X}^k = \sum_{j=1}^{r_k} (\sigma_j^{k-1} - \tau) \boldsymbol{u}_j^{k-1} \boldsymbol{v}_j^{k-1}$
(12) Set $\boldsymbol{X}^k = \min(\max(\boldsymbol{X}^k, lb), ub)$.
(13) **if** $\|\mathcal{P}_\Omega(\boldsymbol{X}^k - \boldsymbol{M})\|_F / \|\mathcal{P}_\Omega \boldsymbol{M}\|_F \leq \epsilon$ **then break**
(14) Set $Y_{ij}^k = \begin{cases} 0 & \text{if } (i,j) \notin \Omega, \\ Y_{ij}^{k-1} + \delta(M_{ij} - X_{ij}^k) & \text{if } (i,j) \in \Omega \end{cases}$
(15) **end** *for* k
(16) Set $\boldsymbol{X}^{\text{opt}} = \boldsymbol{X}^k$

III. Numerical Results

A. Part 1

Compute $r \in \{5, 10, 15, 20\}$ largest singular values and their corresponding singular vectors on a random matrix $\mathbf{A}$ generated as "A=randn(2048,20)*rand(20,512)". The prototype SVD is tested in "test1.m" and the linear time SVD is tested in "test2.m". We plot the singular value and sigular vector below.

Since the performance of LinearTime and Prototype differs significantly on this problem. We plot them separatedly. Clearly, Prototype outperforms LinearTime on this problem, and the singular value decreases slowly.

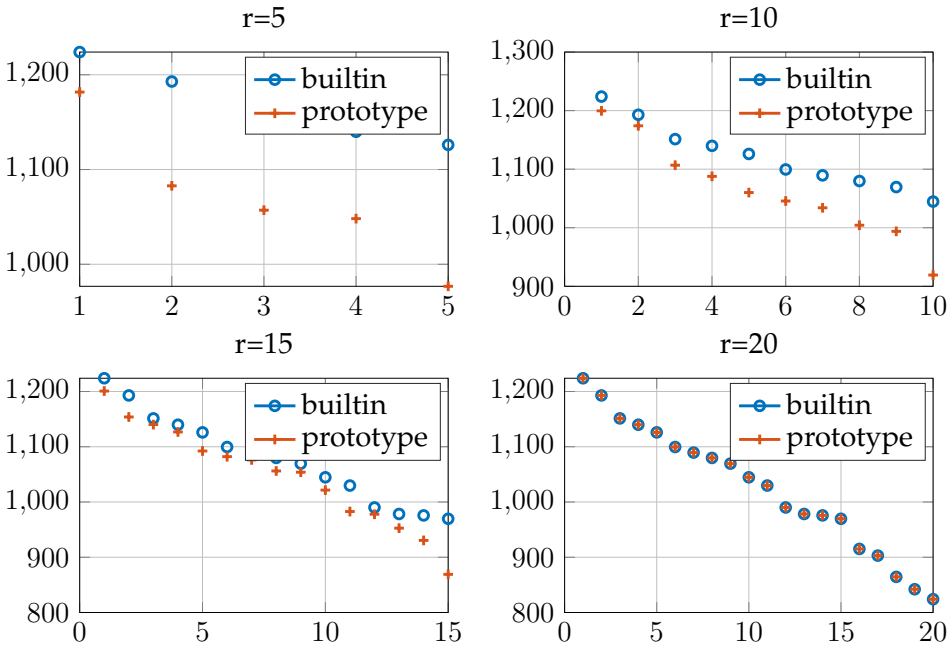


Figure III.1. **Red point**: Singular Values obtained by built-in matlab svd function. **Blue point**: Singular Values obtained by prototype randomized SVD.

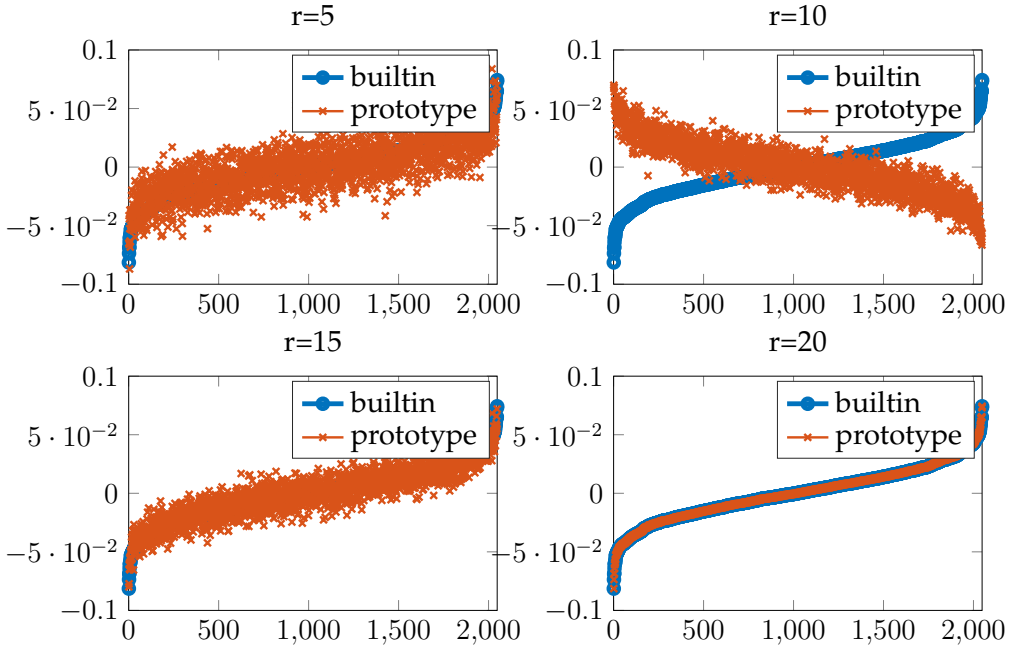


Figure III.2. The vector corresponding to the largest singular value under different r. We sort the index of the vector according to the groundtruth vector. Besides, we do not differentiate vector v and $-v$. **Red point**: Singular Values obtained by built-in matlab svd function. **Blue point**: Singular Values obtained by prototype randomized SVD.

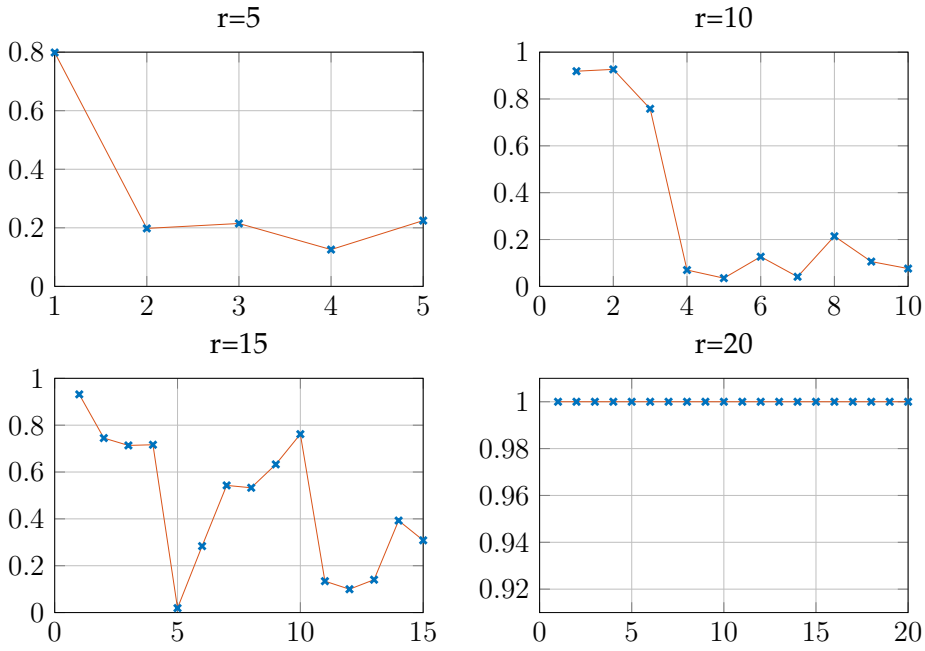


Figure III.3. The absolute value of correlation between computed singular vector and groundtruth singular vector.

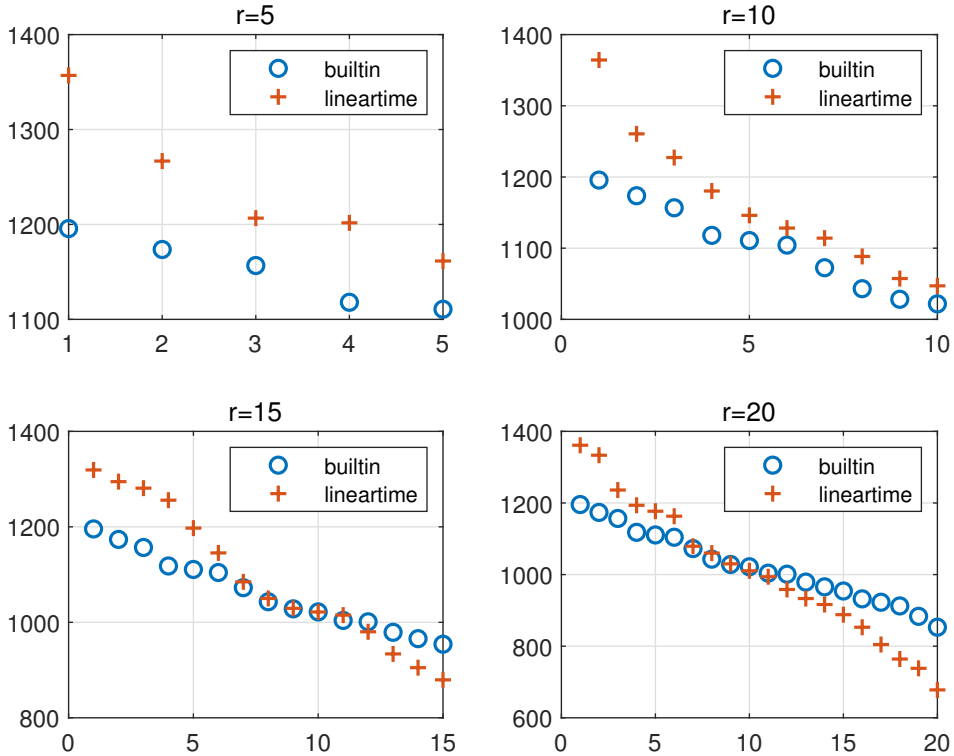


Figure III.4. **Red point**: Singular Values obtained by built-in matlab svd function. **Blue point**: Singular Values obtained by linear time SVD.

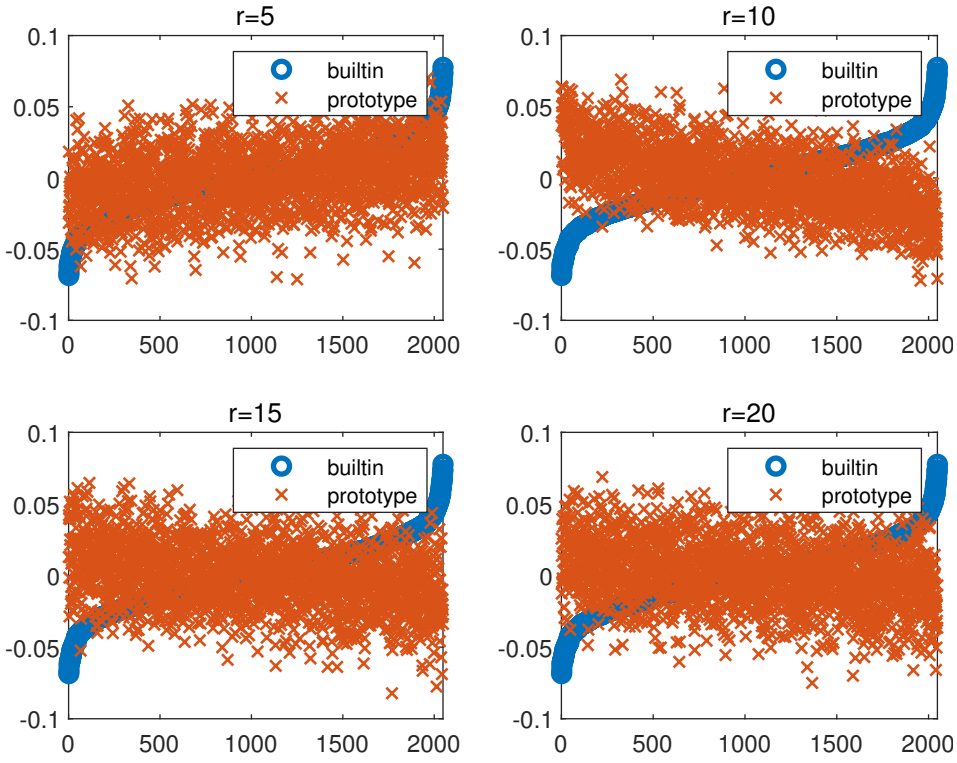


Figure III.5. The vector corresponding to the largest singular value under different r. We sort the index of the vector according to the groundtruth vector. Besides, we do not differentiate vector v and $-v$. **Red point**: Singular Values obtained by built-in matlab svd function. **Blue point**: Singular Values obtained by linear time SVD.

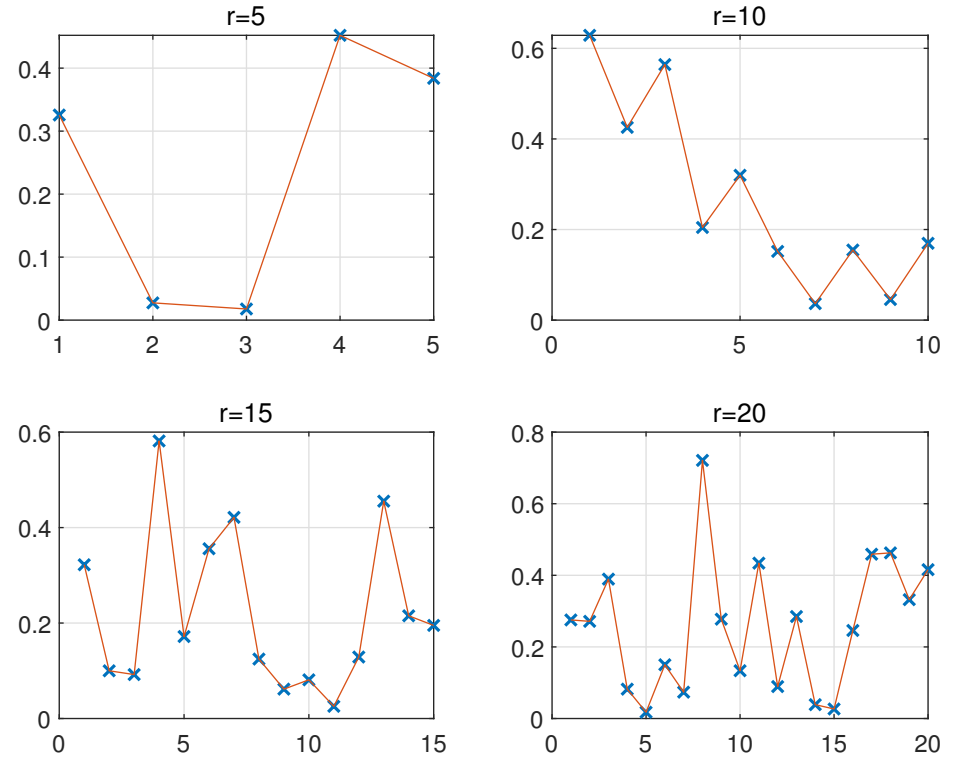


Figure III.6. The absolute value of correlation between computed singular vector and groundtruth singular vector.

B. Part 2

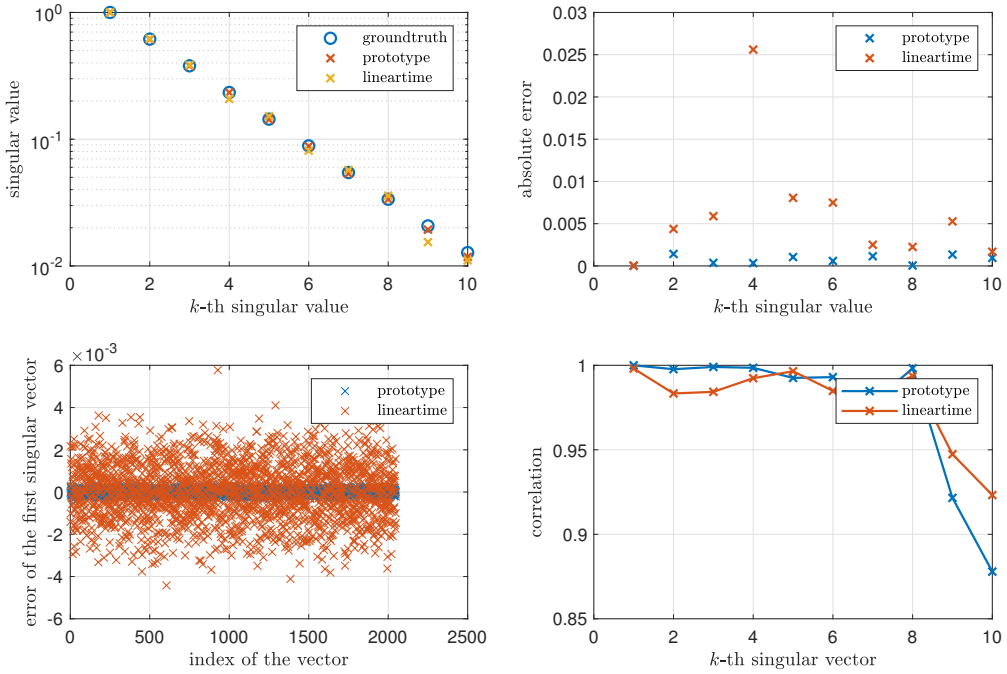
Here, we use "PCAtestmatrix.m"[1]to generate test matrix. We consider Case 1 in [Yu et al., 2017], which is implemeted in function "test3(m,n,k,choice)". Here we set $m = 2048, n = 512$. Four all 6 cases, we test $k = 10, 20, 30, 40$.

Clearly, both method performs well.

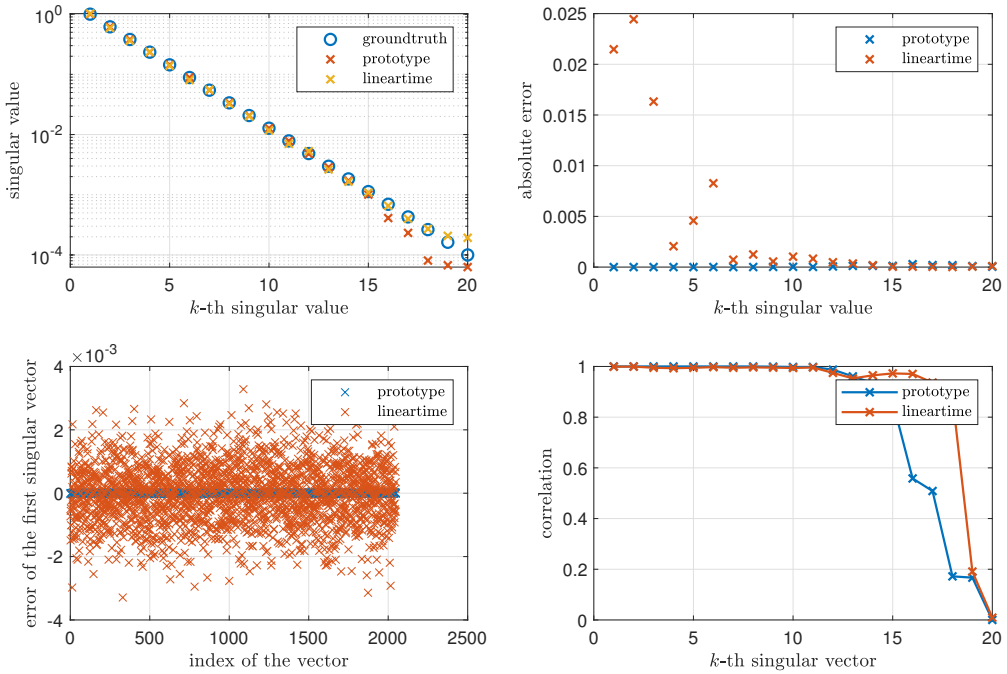
- Case 1. The first 20 singular vector (after logarithmic) decreases linearly. Hence, according to the correlation with the ground truth, only the first 20 singular vector can only be computed accurately.
- Case 2/3. The singular values (after logarithmic) decrease sub-linearly. Both method cannot compute the singular vector except the first few ones. However, the singular vector can be computed accurately.
- Case 4/5. The singular values (after logarithmic) decrease linearly. However, it decreases more slowly than Case 1. The first few singular vectors can be computed rather accurately.
- Case 6. The sigular values lies in different groups. And only those with large values can be approximated accurately.

In summary, both methods favor the rapidly decreasing singular values, where the singular spaces can be separatedly easily.

[1]https://github.com/WenjianYu/rSVD-single-pass

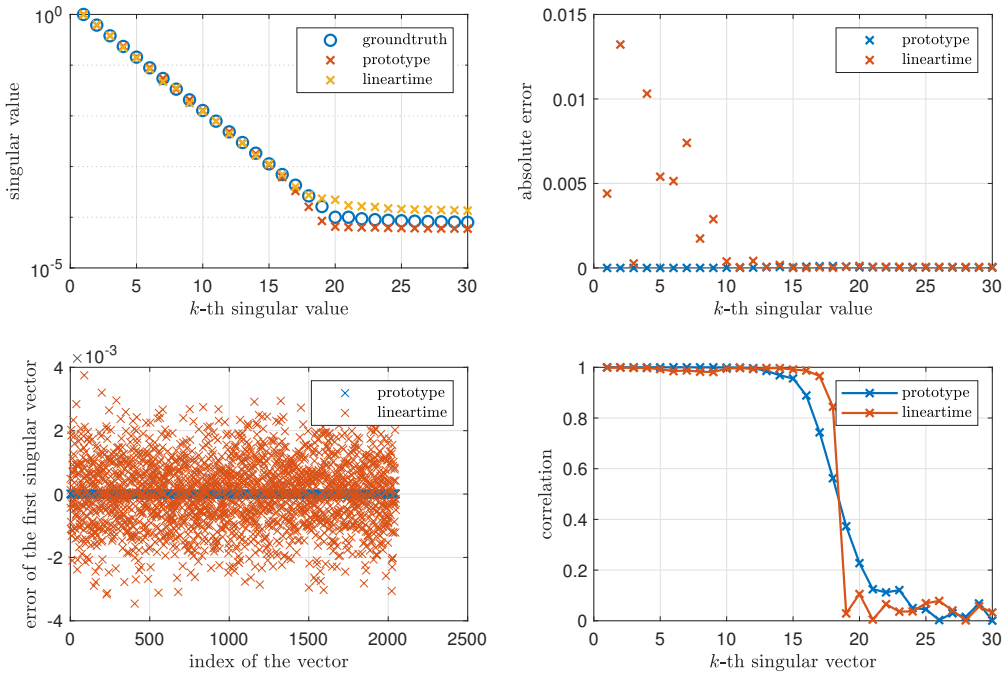


(a) $k = 10$

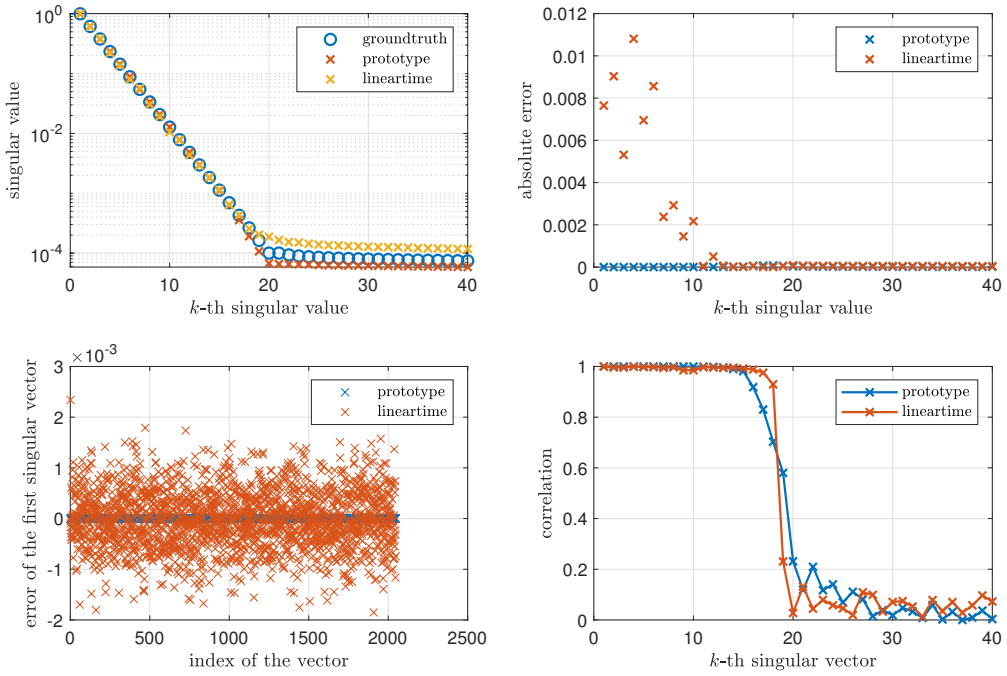


(b) $k = 20$

Figure III.7. PCA test, Case 1, Part 1

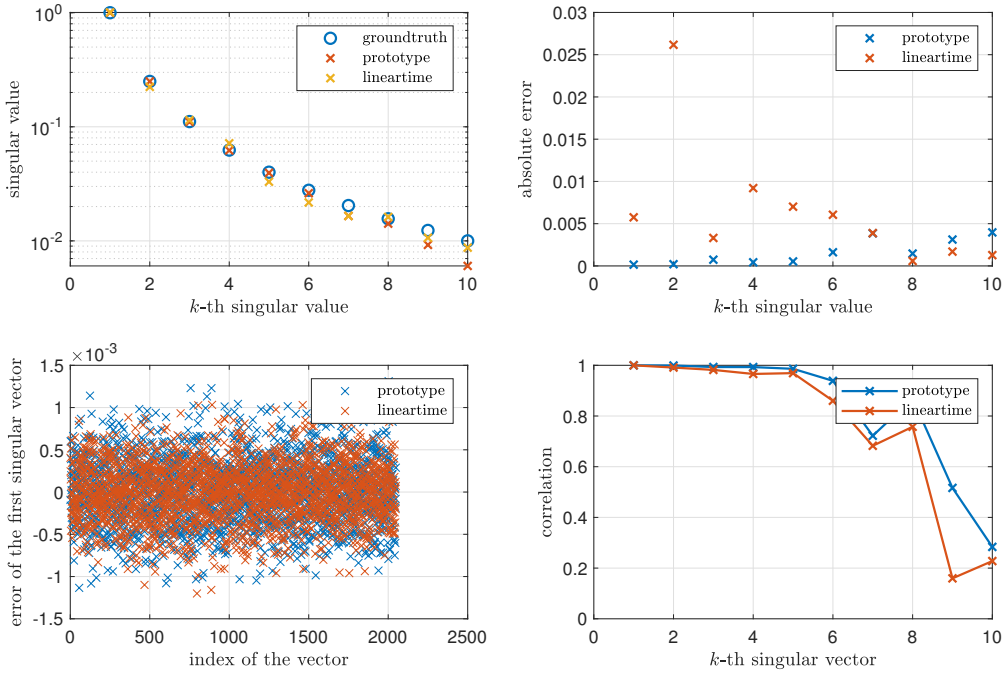


(a) $k = 30$

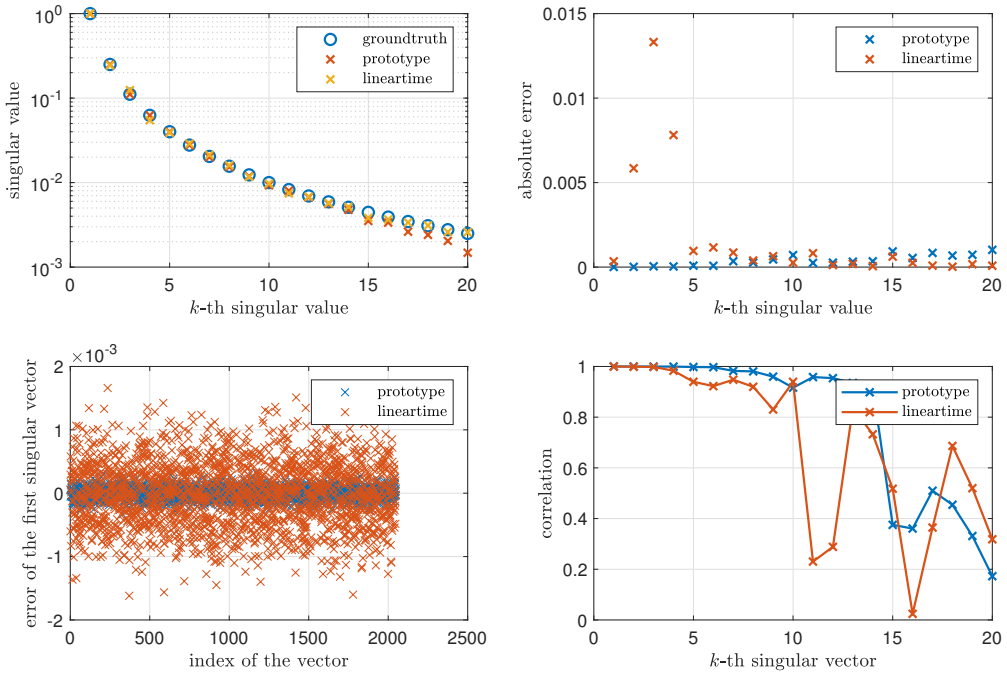


(b) $k = 40$

Figure III.8. PCA test, Case 1, Part 2

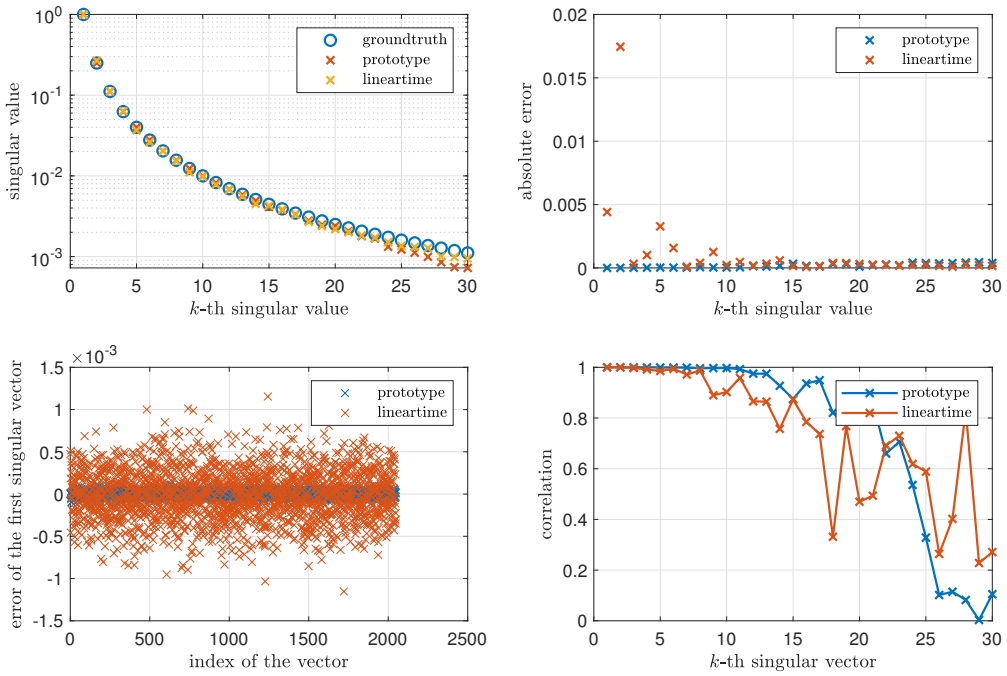


(a) $k = 10$

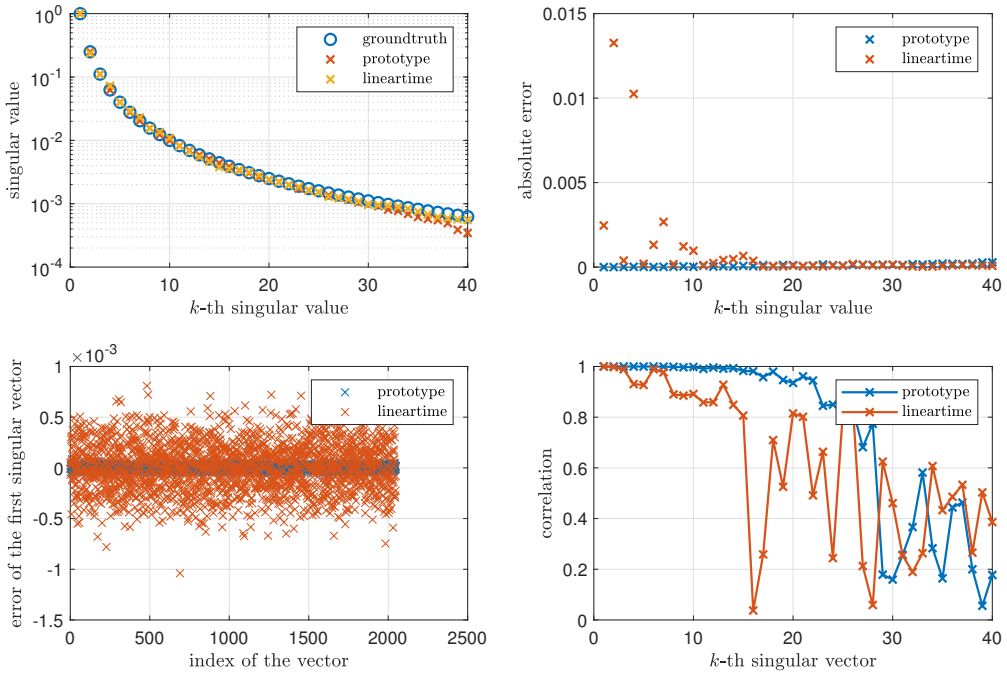


(b) $k = 20$

Figure III.9. PCA test, Case 2, Part 1

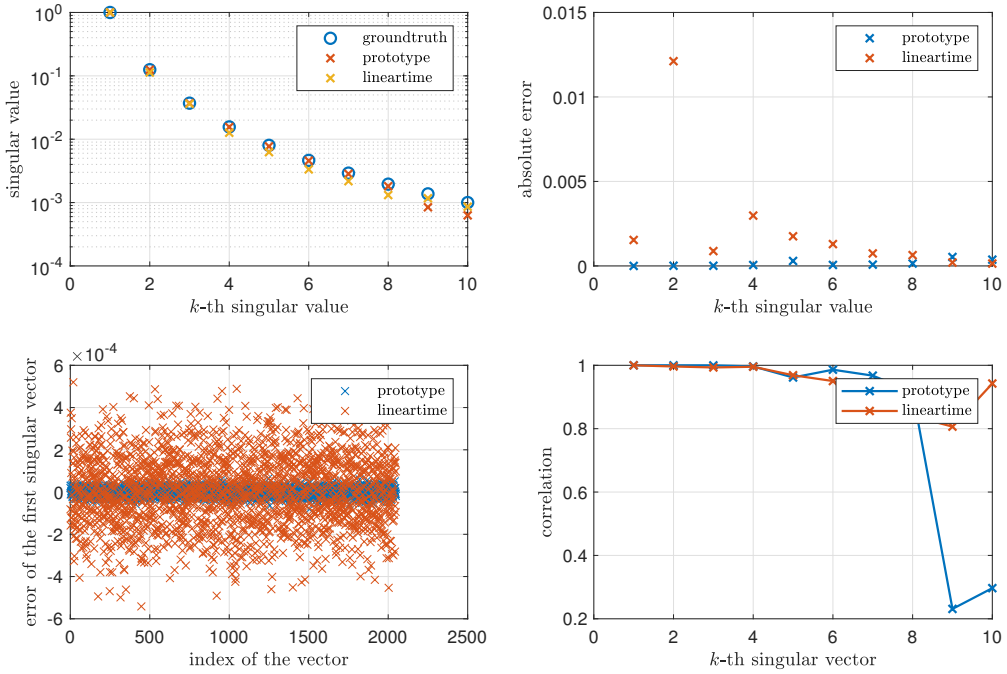


(a) $k = 30$

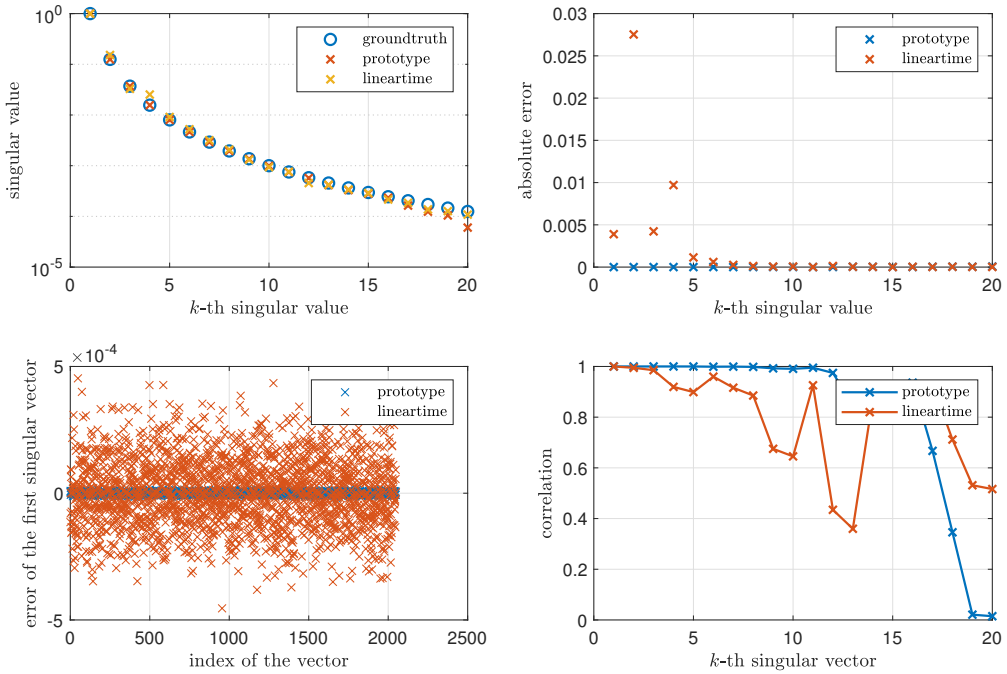


(b) $k = 40$

Figure III.10. PCA test, Case 2, Part 2

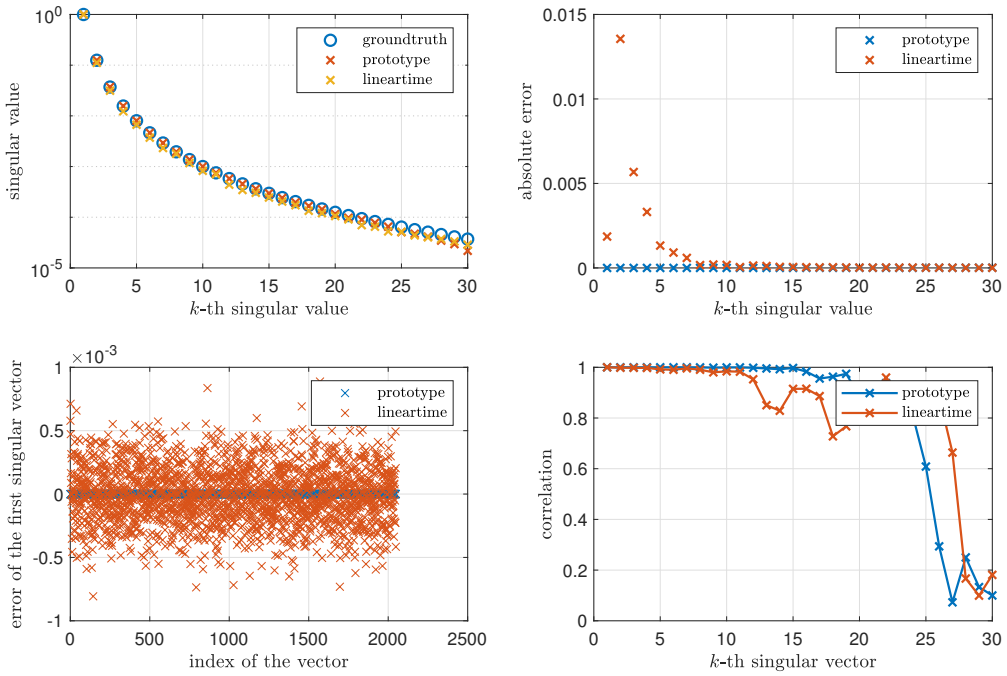


(a) $k = 10$

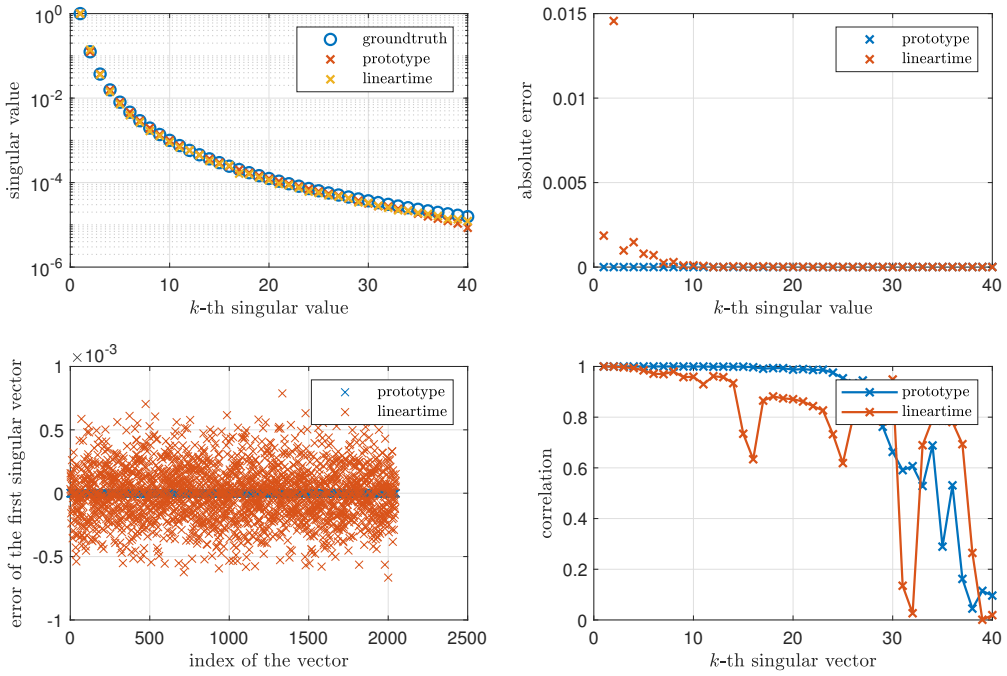


(b) $k = 20$

Figure III.11. PCA test, Case 3, Part 1

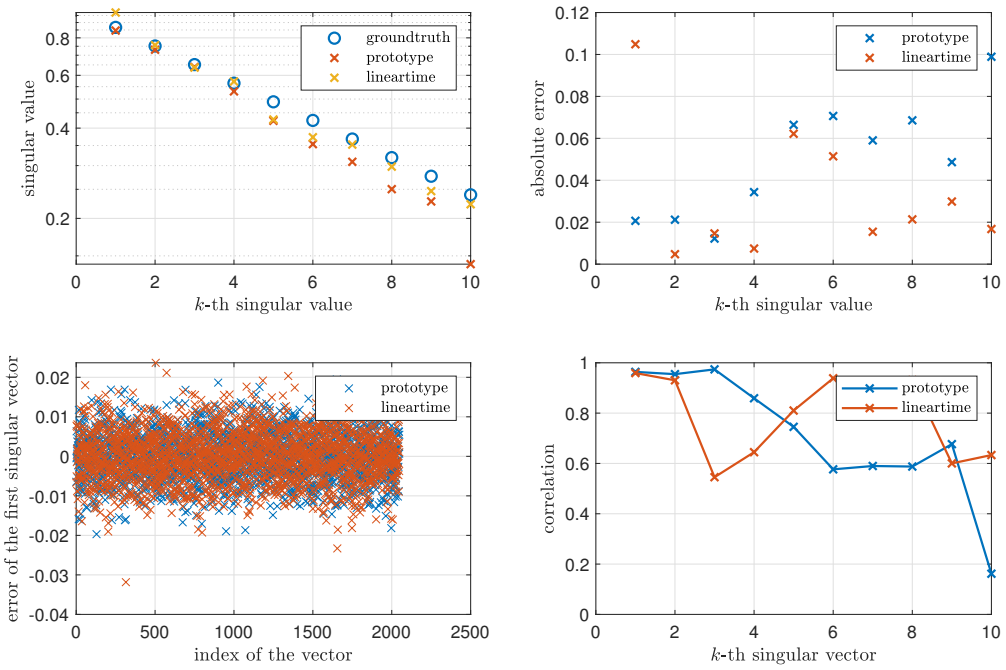


(a) $k = 30$

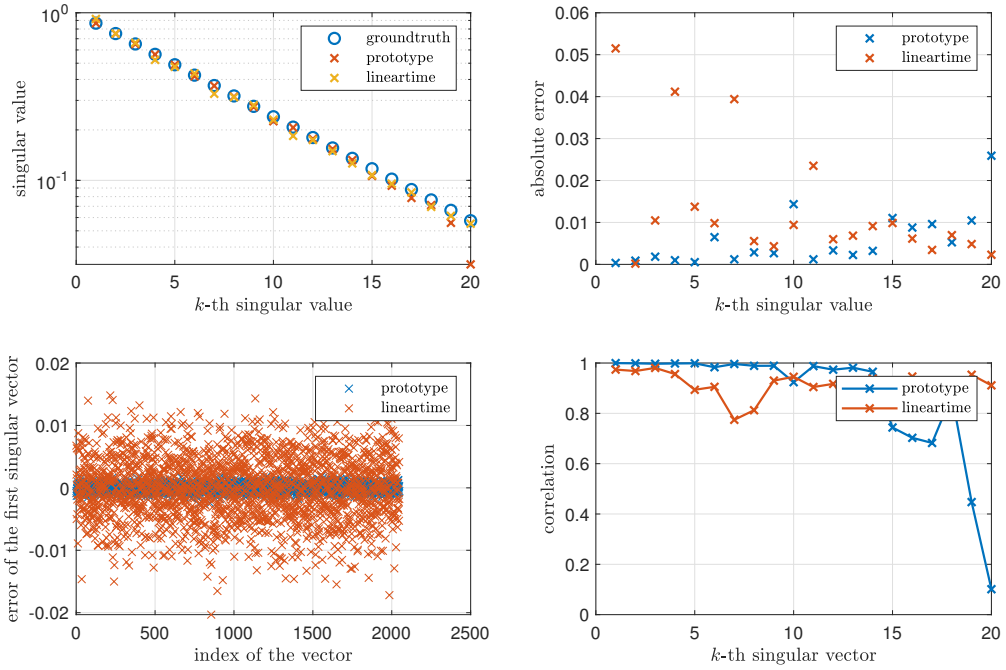


(b) $k = 40$

Figure III.12. PCA test, Case 3, Part 2

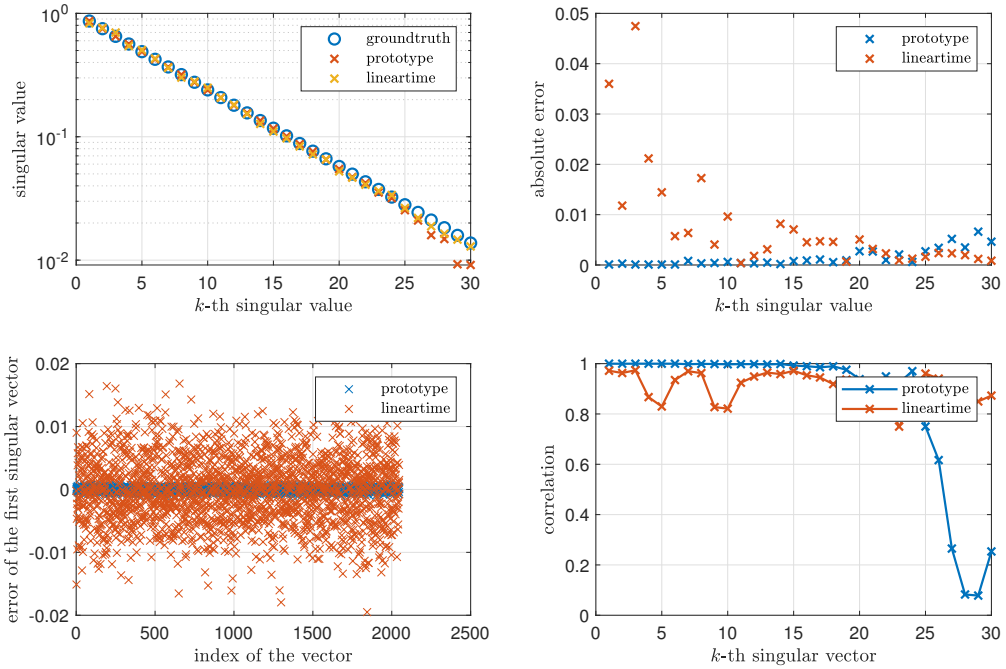


(a) $k = 10$

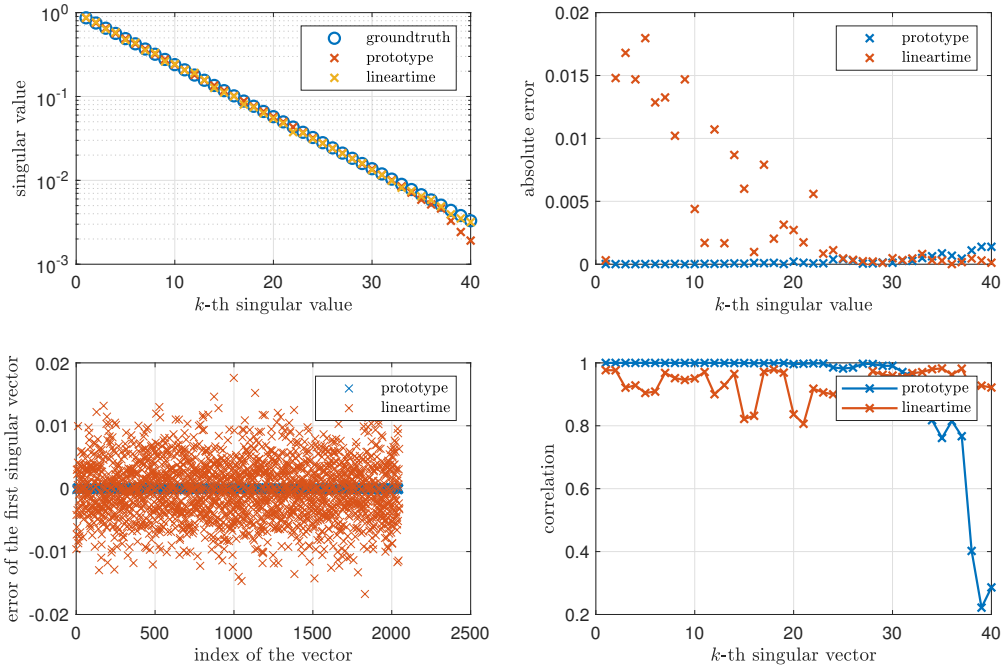


(b) $k = 20$

Figure III.13. PCA test, Case 4, Part 1

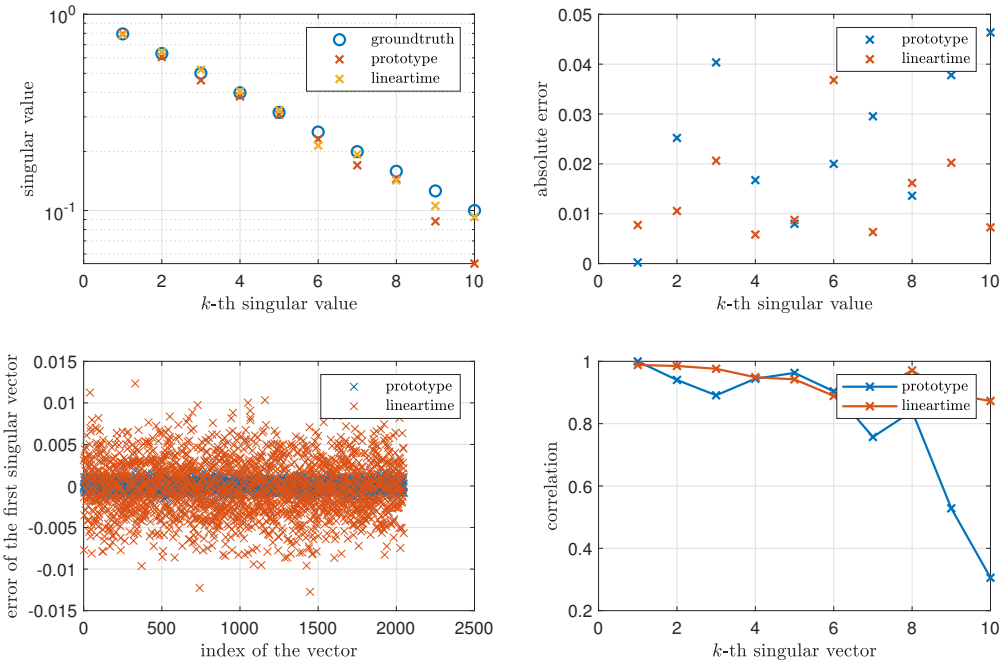


(a) $k = 30$

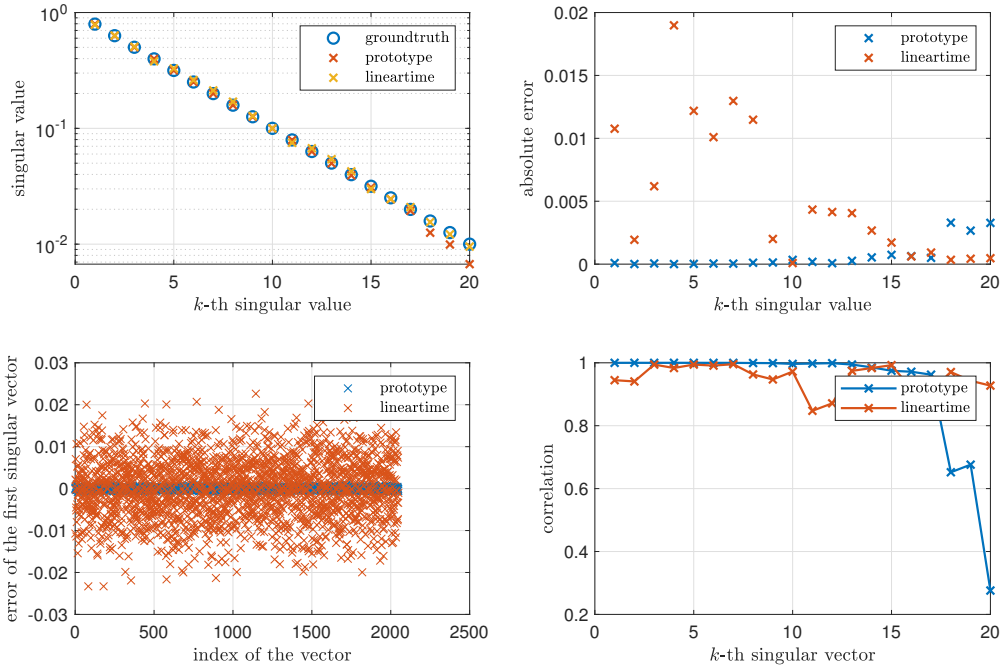


(b) $k = 40$

Figure III.14. PCA test, Case 4, Part 2

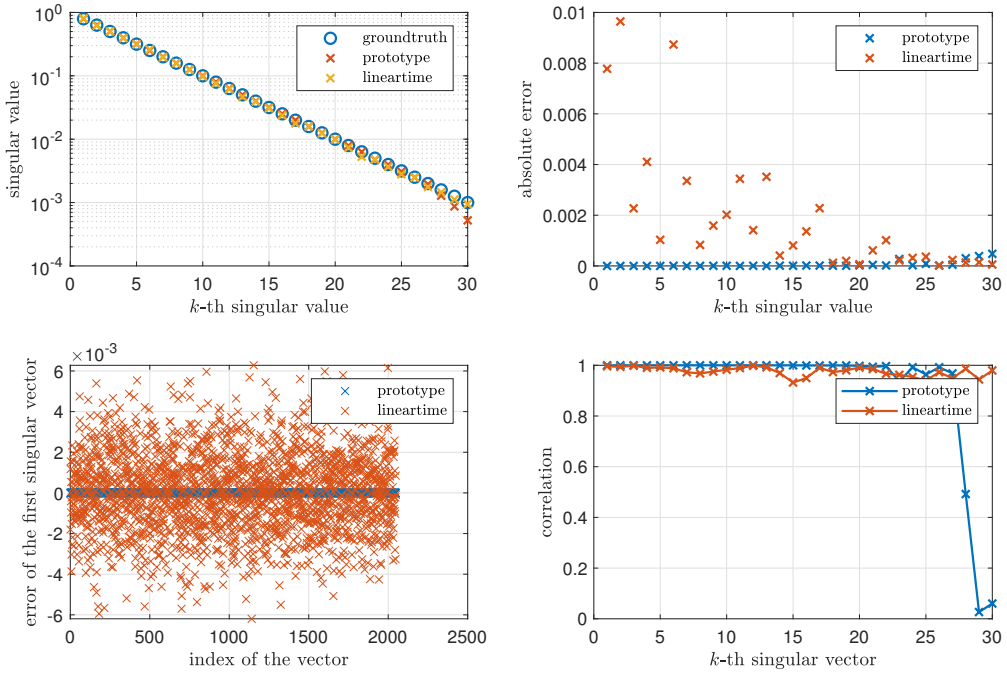


(a) $k = 10$

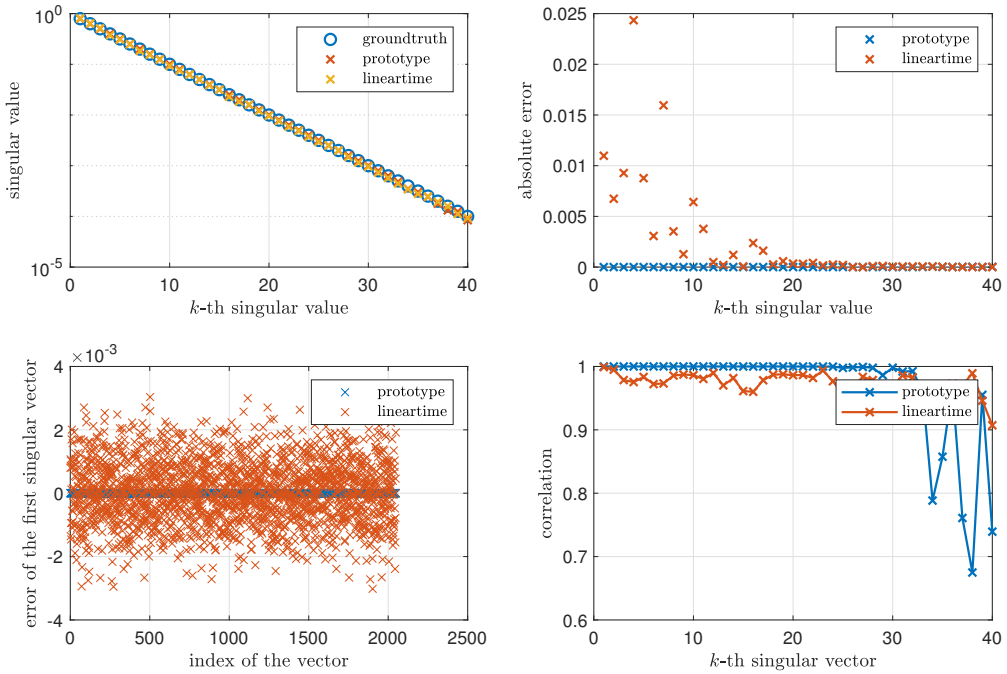


(b) $k = 20$

Figure III.15. PCA test, Case 5, Part 1

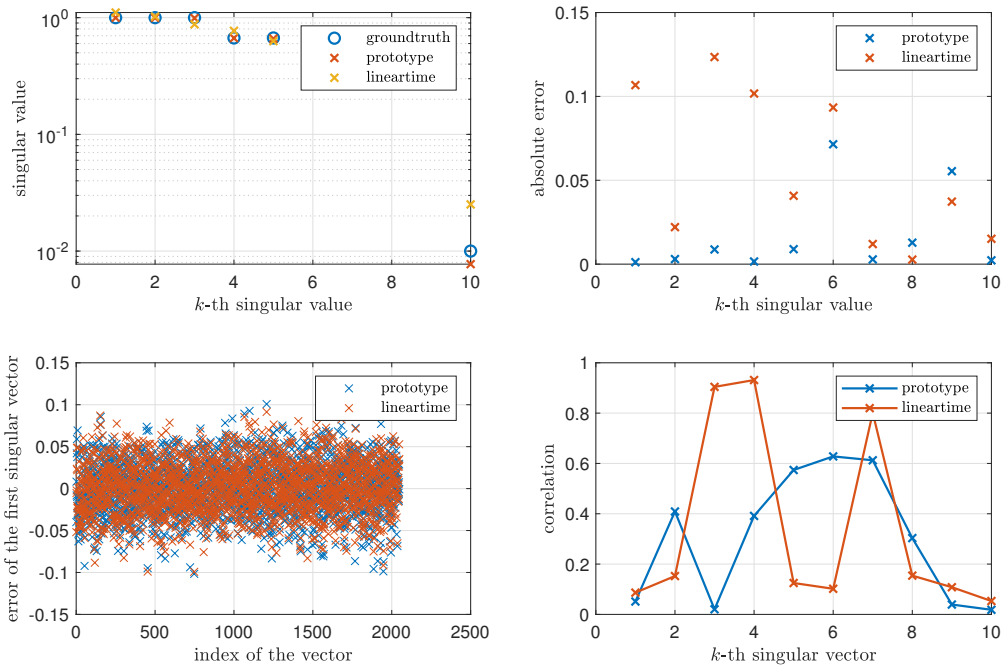


(a) $k = 30$

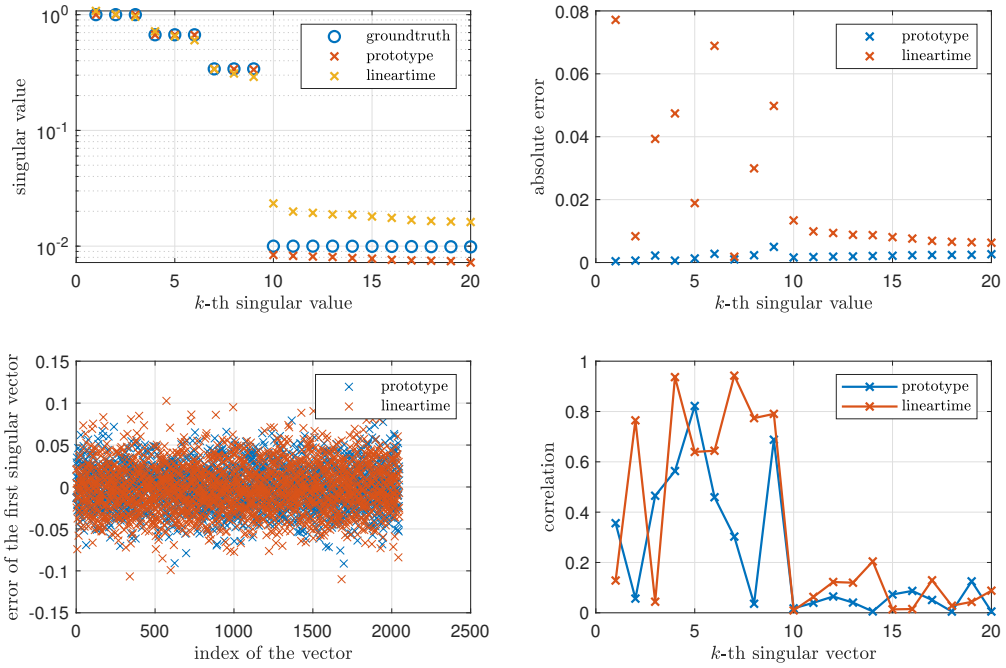


(b) $k = 40$

Figure III.16. PCA test, Case 5, Part 2

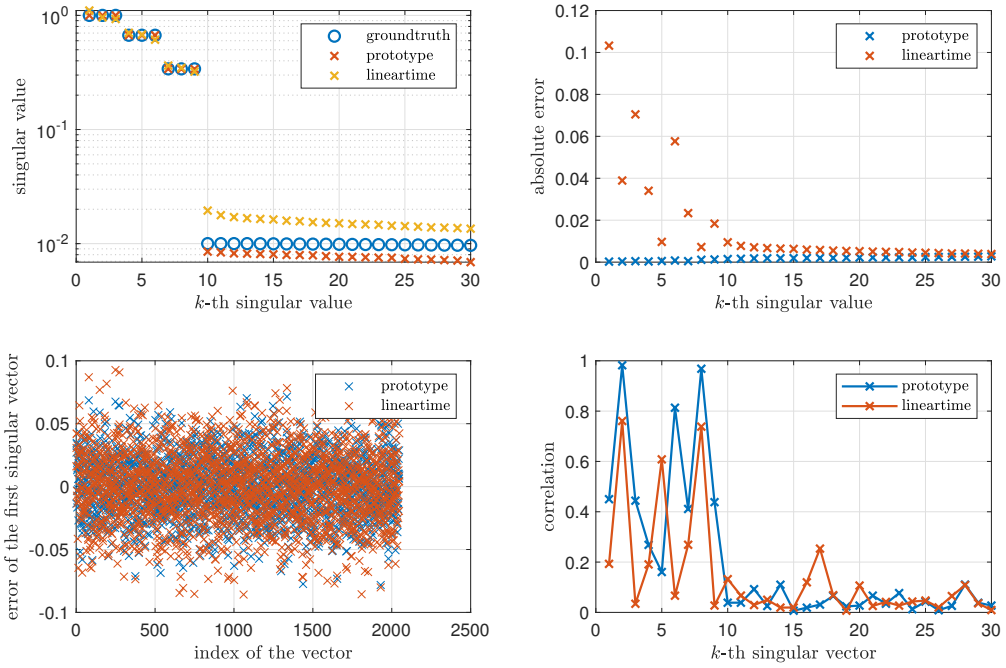


(a) $k = 10$

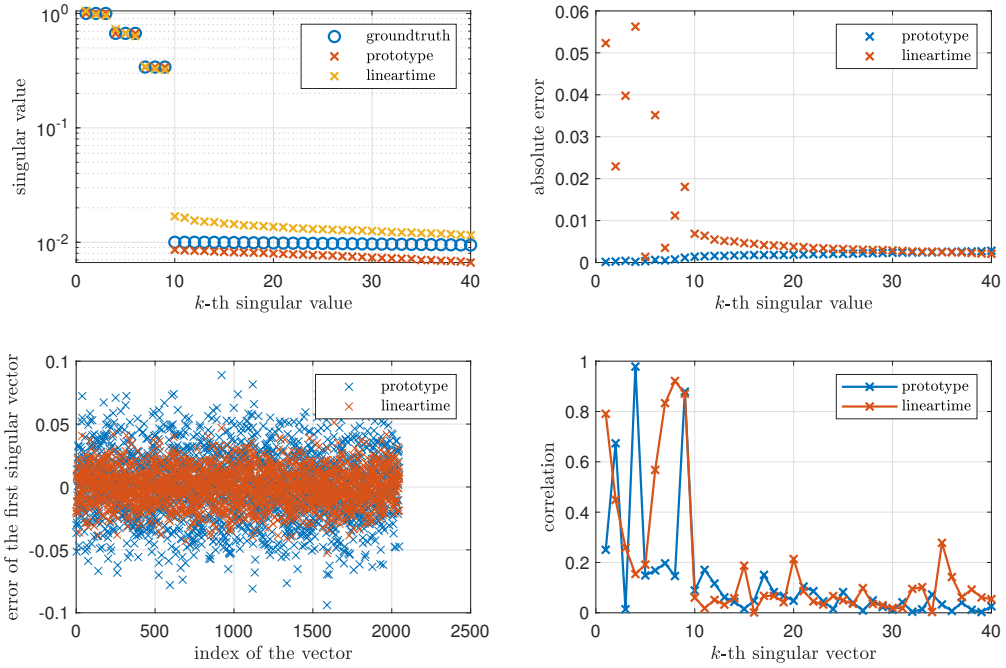


(b) $k = 20$

Figure III.17. PCA test, Case 6, Part 1



(a) $k = 30$



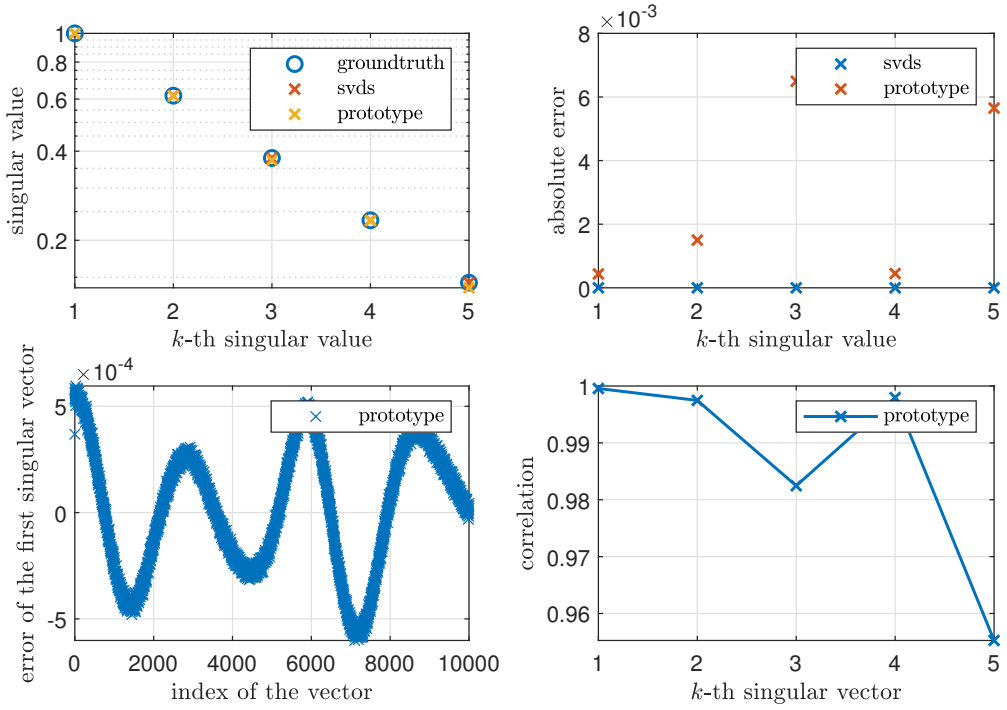
(b) $k = 40$

Figure III.18. PCA test, Case 6, Part 2

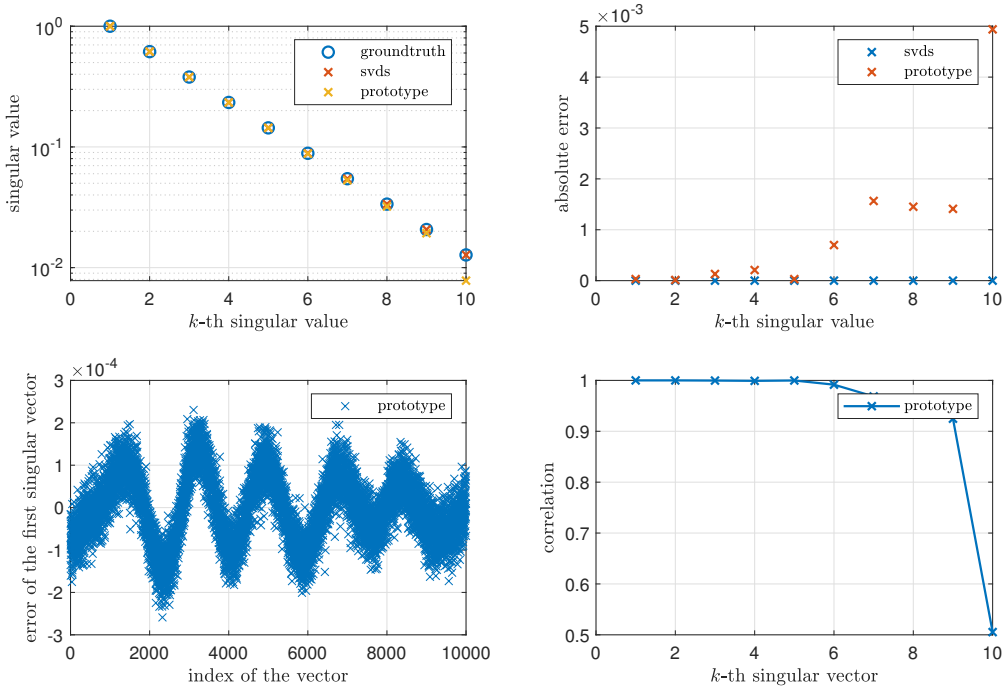
C. Part 3

An large matrix of size $10^4 \times 10^4$ is constructed in "genLargeMatrix.m" and tested in "test5.m". We only conpare the performance of Prototype and matlab built-in function "svds", since from above we find Prototype outperforms LinearTime.

We find that our code "prototype" is comparable to "svds" in terms of CPU time. Besides, the first few singular values and singular vectors can be effectively computed. The main gap between our prototype and "svds" is that given the same desired rank k, the last few singular vectors are highly inaccurate.

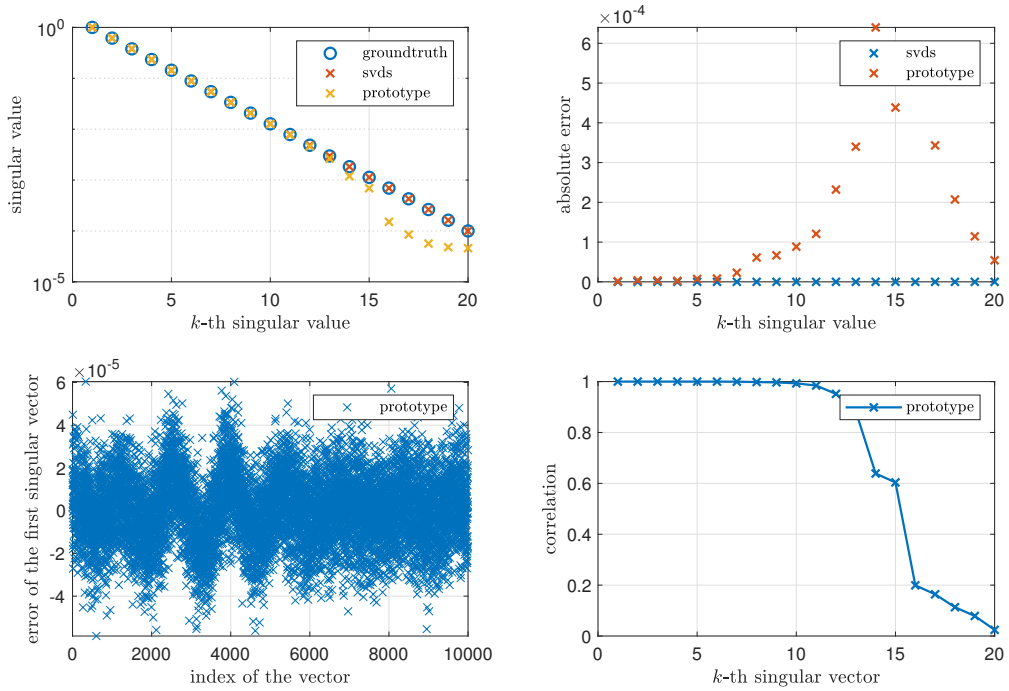


(a) $k = 10$

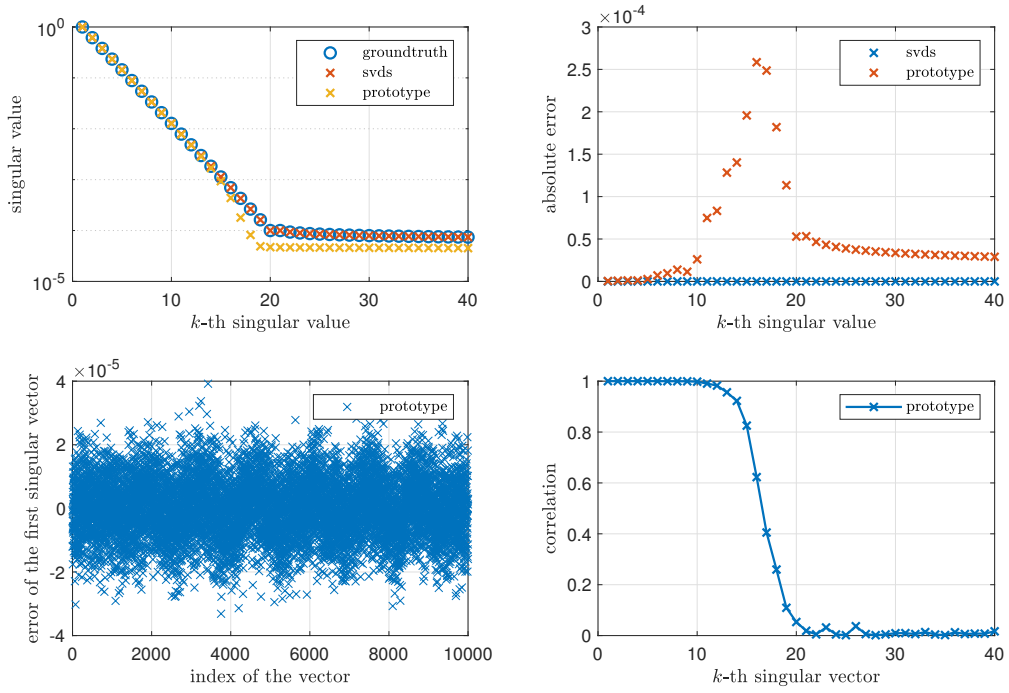


(b) $k = 20$

Figure III.19. Large matrix test, Part 1



(a) $k = 30$



(b) $k = 40$

Figure III.20. Large matrix test, Part 2

D. Part 4

We use figures from Sophie Marceau from Douban [2] as original file. The figure can be seemed as a $2328 \times 1908 \times 3$ tensor, we transform it into a matrix of size 6984×1908 and sample 20% of the matrix as observations. Besides, we add Gaussian noise onto the observed data. We set different levels of tolerance (named "tol") in the programme.

tolerance	SVT iter	error (SVT)	error (on real image)	rank
0.4	11	0.3406	0.3434	3
0.2	35	0.2	0.2039	4
0.1	141	0.0983	0.1080	22
0.05	398	0.0496	0.0656	61

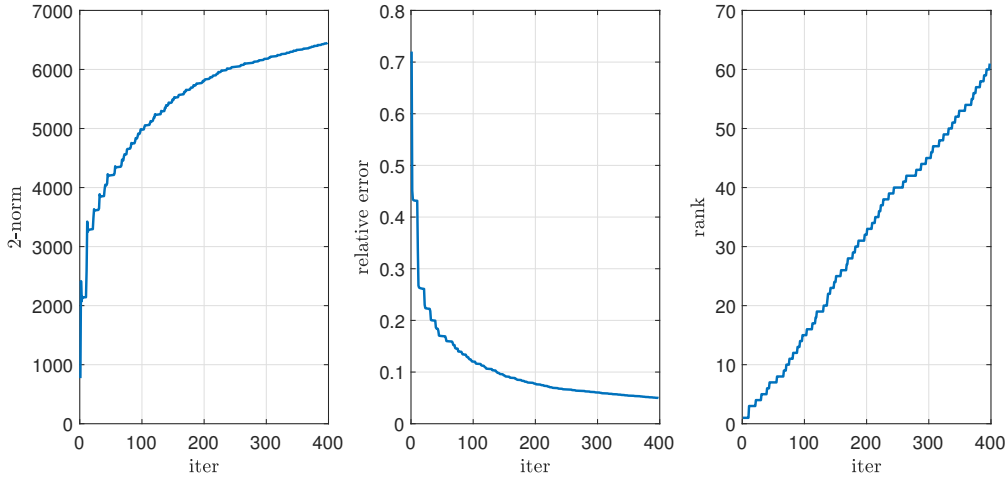
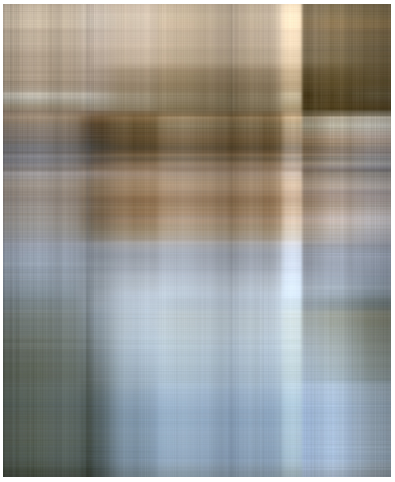


Figure III.21. Tol= 0.05

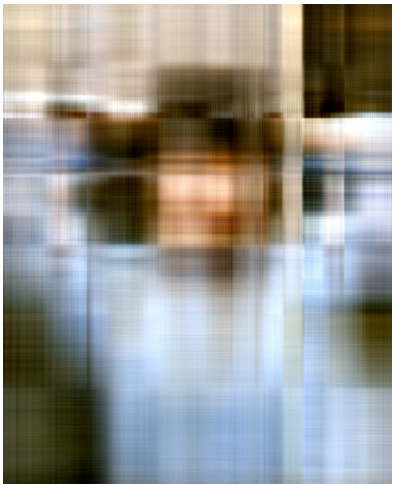
[2] https://movie.douban.com/celebrity/1040543/photo/2583230256/

(a) Original file

(b) Noisy observation



(c) $tol = 0.4$



(d) $tol = 0.2$

(e) $tol = 0.1$

(f) $tol = 0.05$

Figure III.22. Figure reconstruction

REFERENCES

[Candès and Recht, 2009] Candès, E. J. and Recht, B. (2009). Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717.

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

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

[Mazumder et al., 2010] Mazumder, R., Hastie, T., and Tibshirani, R. (2010). Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research*, 11(Aug):2287–2322.

[Yu et al., 2017] Yu, W., Gu, Y., Li, J., Liu, S., and Li, Y. (2017). Single-pass pca of large high-dimensional data. *arXiv preprint arXiv:1704.07669*.