Regularization with randomized SVD for large-scale discrete inverse problems

IOP PUBLISHING INVERSE PROBLEMS

Inverse Problems **29** (2013) 085008 (23pp) doi:10.1088/0266-5611/29/8/085008

Regularization with randomized SVD for large-scale discrete inverse problems

Hua Xiang[1] **and Jun Zou**[2]

[1] School of Mathematics and Statistics, Wuhan University, Wuhan 430072, People's Republic of China
[2] Department of Mathematics, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong

E-mail: hxiang@whu.edu.cn and zou@math.cuhk.edu.hk

Received 7 April 2013, in final form 20 June 2013
Published 18 July 2013
Online at stacks.iop.org/IP/29/085008

Abstract
In this paper, we propose an algorithm for solving the large-scale discrete ill-conditioned linear problems arising from the discretization of linear or nonlinear inverse problems. The algorithm combines some existing regularization techniques and regularization parameter choice rules with a randomized singular value decomposition (SVD), so that only much smaller scale systems are needed to solve, instead of the original large-scale regularized system. The algorithm can directly apply to some existing regularization methods, such as the Tikhonov and truncated SVD methods, with some popular regularization parameter choice rules such as the L-curve, GCV function, quasi-optimality and discrepancy principle. The error of the approximate regularized solution is analyzed and the efficiency of the method is well demonstrated by the numerical examples.

(Some figures may appear in colour only in the online journal)

1. Introduction

We will consider the ill-conditioned linear system $Ax = b$ arising from the discretization of some linear inverse problem [10] or of the linearized system of some nonlinear inverse problem [2, 13], where A is an $m \times n$ matrix and b is obtained from measurement data. For an ill-posed inverse problem, its solution is usually very sensitive to the perturbation in the measurement data b. In order to achieve a meaningful solution that changes stably with respect to the perturbation in the data, one often adopts some regularization techniques [13]. Tikhonov regularization is one of the most popular and effective techniques, which converts the solution of the system $Ax = b$ into the solution of the regularized least-squares system

$$\min_x \{||Ax - b||^2 + \mu^2 ||x||^2\}, \tag{1}$$

where constant μ is the so-called regularization parameter. The minimization problem (1) is equivalent to the system

$$(A^TA + \mu^2 I)x = A^Tb. \tag{2}$$

Suppose that we have the singular value decomposition (SVD) of matrix $A \in \mathbb{R}^{m\times n}(m \geqslant n)$, namely we can write $A = U\Sigma V^T$, where $U = (u_1, \dots, u_m)$, $V = (v_1, \dots, v_n)$ are orthonormal matrices, and $\Sigma = \mathrm{diag}(\sigma_1, \dots, \sigma_n) \in \mathbb{R}^{m\times n}$, with σ_1, $\sigma_2, \dots, \sigma_n$ being the singular values and $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_n$. Then the solution of (2), i.e., the Tikhonov regularized solution x_μ, can be expressed as

$$x_\mu = \sum_{i=1}^{n} f_i \frac{u_i^T b}{\sigma_i} v_i, \tag{3}$$

where $f_i = \sigma_i^2/(\sigma_i^2 + \mu^2)$ is the Tikhonov filter factor [29]. When replacing the filter factors f_i by 0s and 1s appropriately, we have the truncated SVD method (TSVD) [25], which is another popular regularization method using the best low-rank approximation of A. The TSVD regularized solution x_k is given by

$$x_k = \sum_{i=1}^{k} \frac{u_i^T b}{\sigma_i} v_i,$$

where the positive integer k is the truncation parameter and is chosen such that the noise-dominated small singular values are discarded.

As A arises mostly from the discretization of some compact operator, it has singular values of quite small magnitude. One can easily see from (3) that the solution can be easily contaminated by the perturbation in the measurement data b without the regularization, i.e., $\mu = 0$. By introducing the regularization ($\mu \neq 0$), we may make a compromise between the sensitivity of the problem and the perturbation of the measured data and greatly reduce the effect caused by the contamination of the noise in the data.

A key issue for the success of the Tikhonov regularization is how to determine a reasonable regularization parameter μ. There are several popular techniques in the literature for the selection of effective regularization parameters. When the noise level is unknown, we may use some heuristic methods, for example, the so-called L-curve method [26, 31], which uses the plot of $(\log ||Ax_\mu - b||, \log ||x_\mu||)$ over a range of μ, i.e., the norm of the regularized solution versus the corresponding residual norm. If there is a corner on the L-curve, one can take the corresponding parameter μ as the desired regularization parameter. Many other heuristic methods can be found in the literature, such as the generalized cross-validation (GCV) function [15], the quasi-optimality criterion [46, 47], Brezinski–Rodriguez–Seatzu estimators [4], Hanke–Raus rule [24], and so on. When the noise level is known, the discrepancy principle [39, 41], the monotone error rule [45] and the balancing principle [35, 38] may be applied.

Once we have knowledge of the SVD of matrix A, we can determine the regularization parameter by some of the aforementioned techniques, such as the L-curve, the GCV function or the Brezinski–Rodriguez–Seatzu estimators. To see this, we can compute the regularized solution x_μ and the corresponding residual $r_\mu = b - Ax_\mu$ using (3) for a range of μ values. Then either the L-curve, the GCV function or the Brezinski–Rodriguez–Seatzu estimators can be easily applied to determine the desired parameter μ, and the corresponding solution x_μ will be viewed as our final regularized solution. So we can see that the SVD is a simple and efficient tool for solving ill-posed discrete problems by Tikhonov regularization if we can afford the computing of the SVD for the corresponding coefficient matrix A.

However, it is widely known that it may be infeasible or extremely expensive to compute SVD when the concerned discrete inverse problem is of large scale. In this work, we shall investigate how to use the SVD to solve large-scale discrete inverse problems in a more feasible and efficient manner. Certainly, we should not work on the original large systems directly, due to the computational complexity, instability and memory limitation. Our approach is to first greatly reduce the size of the original large-scale discrete system, and then apply some existing regularizations combined with the SVD to solve the much reduced discrete system. Clearly, the solution of the reduced system must still be a good approximation of the original large system in order to ensure this approach works effectively. This is a challenging problem and will be the central focus of this study.

In the remainder of this work, we will introduce some more efficient strategies to deal with the large-scale discrete inverse problems, namely some randomized algorithms. These strategies are based on some new randomized algorithms that have been developed recently in the theoretical computer science community. They can greatly reduce the size of the original discrete inverse problem, but require accessing the large matrix A only twice, which is very crucial when the matrix is of large size. Moreover, these algorithms also work well for noisy data, due to their randomness.

2. Algorithms and complexities

2.1. Randomized algorithms

The randomized algorithm has become more and more popular in the matrix approximation in the last decade. It was realized that low-rank approximations can be obtained by randomly sampling the columns of A according to a probability distribution that depends on the Euclidean norms of the columns [14]. This has motivated many studies in this direction. By choosing columns and rows simultaneously one can obtain the so-called CUR matrix decomposition $A \approx CUR$ [12, 37], where C and R are randomly chosen from the columns and rows of A, respectively, and U is a generalized inverse of their intersections. A low-rank approximation in the form of an interpolative decomposition was derived in [36, 50] by using randomized algorithms.

For large-scale systems, the randomized algorithms can be used to approximate the SVD of the concerned coefficient matrices. This is the major interest of the current work, since we need SVD to help find some reasonable regularization parameters through some parameter choice rules that need to compute the SVD such as the L-curve and the GCV techniques, for use in the Tikhonov regularizations for solving discrete inverse problems.

An approximation to SVD was provided in [50] by means of the interpolative decomposition and was compared with the classical pivoted QR decomposition algorithm in [9]. Randomized algorithms for the principle component analysis (PCA) were given and analyzed in [42]. One such algorithm is listed here; see algorithm 1, which gives the best rank-k approximation of $A \in \mathbb{R}^{m\times n}$, i.e., $A \approx U\Sigma V^T$, and meets the following error estimate

$$||A - U\Sigma V^T|| \leqslant Cm^{1/(4i+2)}\sigma_{k+1}, \tag{4}$$

Algorithm 1. Randomized algorithm for PCA.

```
1. Form a real $l \times m$ Gaussian random matrix $\Omega$, and compute $Y = \Omega(AA^T)^i A$ for some positive
integer $i \geqslant 1$.
2. Using the SVD of $Y$, form a real $n \times k$ matrix $Q$ with orthonormal columns, such that there exists
$Z$ satisfying $||QZ - Y^T|| \leqslant \sigma_{k+1}(Y)$.
3. Compute the $m \times k$ matrix $B = AQ$.
4. Form the SVD of $B$: $B = U\Sigma W^T$.
5. Compute $V = QW$, and obtain the approximation $A \approx U\Sigma V^T$.
```

where σ_{k+1} is the $(k+1)$th greatest singular value of A, and i is a non-negative integer in algorithm 1 specified by the user to enhance the decay of the singular values, and it is sufficient with $i = 1$ or 2 in many applications of PCA [42].

The integer l in algorithm 1 is usually selected such that $k < l \leqslant m - k$, for example, $l = k + 12$ [42]. Although the algorithm is randomized, the estimate (4) holds with very high probability (typically 1–10^{-15}); see [42] for the details and different variants of this algorithm. More about randomized algorithms can be found in the recent review paper [20]. The following algorithm 2 gives an important variant of algorithm 1, which will be fundamental to our subsequent studies. This variant provides an approximation of the SVD of matrix A, and was analyzed in [20]. We shall call it the randomized SVD (or RSVD).

Algorithm 2 (RSVD). Given $A \in \mathbb{R}^{m\times n}(m \geqslant n)$ and $l < n$, compute an approximate SVD: $A \approx U\Sigma V^T$ with $U \in \mathbb{R}^{m\times l}$, $\Sigma \in \mathbb{R}^{l\times l}$ and $V \in \mathbb{R}^{n\times l}$.

```
1. Generate an $n \times l$ Gaussian random matrix $\Omega$.
2. Form the $m \times l$ matrix $Y = A\Omega$.
3. Compute the $m \times l$ orthonormal matrix $Q$ via QR factorization $Y = QR$.
4. Form the $l \times n$ matrix $B$ such that $B = Q^TA$.
5. Compute the SVD of the small matrix $B$: $B = W\Sigma V^T$.
6. Form the $m \times l$ matrix $U = QW$, then $A \approx U\Sigma V^T$.
```

In algorithm 2, the index l is usually selected in the form $l = k + p$, where p is an oversampling parameter and k corresponds to the rank-k specified in advance for the best rank-k approximation of A [20]. To understand algorithm 2 more, we make some remark about each step of the algorithm. Step 2 is used to extract the column information, i.e. the range of A, and yields a matrix with much smaller columns ($l < n$). In step 3, an orthogonal matrix Q is formed to represent the range of A, and it also gives the first approximation of the left singular vectors of A. The matrix A is reduced to a smaller matrix in step 4, and the matrix B^T provides information on $R(A^T) = N(A)^{\perp}$, or the range of right singular vectors. After SVD on the small matrix in step 5, the first left singular vector approximation Q is modified by W in step 6 to obtain the final approximation of the left singular vectors, leading to an SVD approximation.

For the matrices of size $m \times n$ with $m < n$, we propose a variant of algorithm 2, i.e., the following algorithm 3.

Algorithm 3 (RSVD*). Given $A \in \mathbb{R}^{m\times n}(m < n)$ and $l < m$, compute an approximate rank-l SVD: $A \approx U\Sigma V^T$ with $U \in \mathbb{R}^{m\times l}$, $\Sigma \in \mathbb{R}^{l\times l}$ and $V \in \mathbb{R}^{n\times l}$.

```
1. Generate an $l \times m$ Gaussian random matrix $\Omega$.
2. Compute the $l \times n$ matrix $Y = \Omega A$.
3. Compute the $n \times l$ orthonormal matrix $Q$ via QR factorization $Y^T = QR$.
4. Form the $m \times l$ matrix $B = AQ$.
5. Compute the SVD of a small matrix $B$: $B = U\Sigma W^T$.
6. Form the $n \times l$ matrix $V = QW$, then $A \approx U\Sigma V^T$.
```

We have tried several other similar variants of algorithms 2 and 3. For example, we may randomly choose l columns to form the matrix Y in algorithm 2. In this way, we can avoid the matrix–matrix multiplication in step 2, therefore reducing the computational costs nearly by half. Our numerical tests have shown that this variant can succeed for some cases but may fail to give a good SVD approximation for more difficult problems, especially when l is small. We have also tried the double projection of the form QQ^TAPP^T, where Q is defined as in algorithm 2, and P has a similar role to Q, i.e. the range of P should be a good approximation of $R(A^T)$ or $N(A)^\perp$. That is, we can use $A \approx QQ^TAPP^T$ for the approximate SVD of A. This variant works well in our numerical experiments in locating the regularization parameters and finding the regularized solutions, but it nearly doubles the computational costs of algorithm 2. Among all different variants we have tried numerically, we find algorithms 2 and 3 most efficient and robust. Therefore, we shall mainly focus on these two algorithms in the rest of this work.

2.2. Approximate regularized solutions by randomized SVD

In this section, we propose an algorithm that combines the randomized SVD addressed in the previous section with some regularization techniques to solve the large-scale regularized discrete inverse system (2). This seems to be the first time in the literature to solve the discrete inverse problems in such a manner.

As discussed in the introduction, we can use the SVD to determine a desired regularization parameter and find the corresponding solution to the regularized system (2). However, it may not be feasible or efficient to use the SVD for large-scale systems. But in the rest of this section, we show that it is possible to solve the large-scale systems efficiently, with the help of the newly introduced randomized SVD.

Suppose that we have an approximate SVD of matrix A, i.e., $A \approx U\Sigma V^T$, where $U \in \mathbb{R}^{m\times l}$, $\Sigma \in \mathbb{R}^{l\times l}$ and $V \in \mathbb{R}^{n\times l}$. Then by replacing A by its approximate SVD, the regularized system (2) can be approximated by

$$(V\Sigma^2V^T + \mu^2 I)x = V\Sigma U^Tb, \tag{5}$$

which gives an approximate regularized solution of (2):

$$x_\mu = V(\Sigma^2 + \mu^2 I)^{-1}\Sigma U^T b. \tag{6}$$

For convenience, we set $s = (\sigma_1, \dots, \sigma_l)^T$ and $t = (U^Tb)./s$, where $./$ is the componentwise division. Let $.*$ be the componentwise multiplication and $f = (f_1, \dots, f_l)^T$ with $f_i = \sigma_i^2/(\sigma_i^2+\mu^2)$. Then the approximate regularized solution (6), its norm η and residual norm ρ can be evaluated, respectively, by

$$\begin{aligned}
x_\mu &= V(f.*t),\\
\eta &= ||x_\mu|| = ||f.*t||,\\
\rho &= ||b - Ax_\mu|| = ||b - U\Sigma(f.*t)||\\
&= \sqrt{||(I-UU^T)b||^2 + ||UU^Tb - U\Sigma(f.*t)||^2}\\
&= \sqrt{||(I-UU^T)b||^2 + ||UU^Tb - U(f.*(U^Tb))||^2}\\
&= \sqrt{\beta_0^2 + ||(1-f).*\hat{b}||^2},
\end{aligned}$$

where $\hat{b} = U^Tb$ and $\beta_0^2 = ||(I-UU^T)b||^2 = ||b||^2 - ||\hat{b}||^2$.

Using the above formulae and the approximate SVD of A, the entire process of finding a desired regularization parameter μ and an approximate regularized solution of (2) can be summarized in the following algorithm.

Choosing a reasonable regularization parameter is crucial for the success of the Tikhonov regularization. There exist many regularization parameter choice rules in the literature that suit algorithm 4, but we have used only the L-curve rule [26, 31] in the description of the algorithm. The L-curve rule determines a parameter by the plot of $(\log ||Ax_\mu - b||, \log ||x_\mu||)$ over a range of μ.

Algorithm 4. Given $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, compute a regularized solution x_μ.

```
1. Apply RSVD (algorithm 2 for $m \geqslant n$ or algorithm 3 for $m < n$) for an approximate rank-$l$ SVD of
   $A \approx U\Sigma V^T$.
2. Choose the regularization parameter $\mu$.
   2.1 Compute $\hat{b} = U^T b$, $t = \hat{b}./s$, $\beta_0^2 = ||b||^2 - ||\hat{b}||^2$ .
   2.2 Select $p$ values of $\mu$: $\mu_1, \mu_2, \ldots, \mu_p$ .
       For $i = 1 : p$, compute
          $f = (f_1, \ldots, f_l)$ with $f_j = \sigma_j^2/(\sigma_j^2 + \mu_i^2)$,
          $\eta_i = ||f. * t||$,
          $\rho_i = \sqrt{\beta_0^2 + ||(1 - f). * \hat{b}||^2}$.
       End.
   2.3 Form the L-curve by $(\log \rho_i, \log \eta_i)$ and locate the corner of the L-curve and the corresponding
       regularization parameter $\mu$.
3. Compute the approximate regularized solution $x_\mu$.
   3.1 Calculate $f = (f_1, \ldots, f_l)$ with $f_j = \sigma_j^2/(\sigma_j^2 + \mu^2)$ $(j = 1, \ldots, l)$ .
   3.2 Form the regularized solution $x_\mu = V(f. * t)$.
```

Next, we briefly review a few other regularization parameter choice rules. The GCV function [15] is a choice rule that determines the regularization parameter by minimizing the GCV function

$$G = \frac{||(I - A(A^TA + \mu^2 I)^{-1}A^T)b||^2}{(\mathrm{tr}(I - A(A^TA + \mu^2 I)^{-1}A^T))^2} = \frac{||Ax_\mu - b||^2}{\left(m - n + \Sigma_{i=1}^n \mu^2/(\sigma_i^2 + \mu^2)\right)^2}, \tag{7}$$

where we have used (2) and the SVD of A to derive the numerator and denominator (for $m \geqslant n$), respectively. The GCV method intends to balance the data error and the regularization error [26, 29].

Historically, the quasi-optimality criterion [46, 47] might be the first heuristic parameter choice rule. Recently, a family of error estimators E_i and $\hat{E}_i$ of the form

$$E_i^2 = c_0^{i-1}c_1^{6-2i}c_2^{i-4}, \quad \hat{E}_i^2 = c_0^{i-1}c_1^{6-2i}\hat{c}_2^{i-4}$$

were proposed in [4] for any positive integer i, where E_i and $\hat{E}_i$ are the estimators of the error $||e|| := ||x - x_\mu||$ and c_0, c_1, c_2 and $\hat{c}_2$ are four constants given, respectively, by

$$c_0 = \langle r_\mu, r_\mu\rangle, \quad c_1 = \langle r_\mu, Ar_\mu\rangle, \quad c_2 = \langle Ar_\mu, Ar_\mu\rangle, \quad \hat{c}_2 = \langle A^T r_\mu, A^T r_\mu\rangle,$$

with $r_\mu = b - Ax_\mu$ being the residual. Note that $\hat{E}_3$, called *the Auchmuty estimator*, is a lower bound of $||e||$, and can be written explicitly as

$$\hat{E}_3 = ||r_\mu||^2||x_\mu||^{-1}\mu^{-2}$$

by using the fact that $A^T r_\mu = \mu^2 x_\mu$. In general, E_3 and $\hat{E}_3$ are the best of all the estimators E_i and $\hat{E}_i$ [4].

All these above rules do not require data on the noise level explicitly. They are simple to apply, popular and effective in many practical applications, despite some of their drawbacks [3, 23, 34, 48, 49]. For instance, the L-curve corner may not exist. It is under-smoothing for smooth solutions [23], and not convergent in the stochastic discrete data setting as the sample size goes to infinity, where it leads to over-smoothing [49]. On the other hand, the GCV rule may be unstable for the correlated noise, resulting in under-smoothing [3].

When the noise level is known, the discrepancy principle [39, 41], the monotone error rule [45] and the balancing principle [35, 38] can be applied instead. The discrepancy principle selects the regularization parameter such that the residual norm of the regularized solution is about the same as the noise level in the data. The monotone error rule seeks the largest

Table 1. Computational complexity.

	RSVD	RSVD*	RRQR–SVD
Step 1	$O(nl)$	$O(ml)$	$4mnl - 2(m+n)l^2 + \frac{4}{3}l^3$
Step 2	$2mnl$	$2mnl$	$6nl^2 + 20l^3$
Step 3	$4ml^2 - \frac{4}{3}l^3$	$4nl^2 - \frac{4}{3}l^3$	$2ml^2$
Step 4	$2mnl$	$2mnl$	0
Step 5	$6nl^2 + 20l^3$	$6ml^2 + 20l^3$	–
Step 6	$2ml^2$	$2nl^2$	–

computable regularization parameter μ_0 such that the error of the regularized solution in (1) decreases monotonically as μ goes from 0 to μ_0. For the iterative methods of the form $x_{k+1} = x_k + A^T w_k$, the monotone error rule chooses the regularization parameter to be the first index k satisfying $\langle r_k + r_{k+1}, w_k\rangle/(2||w_k||) \leqslant \varepsilon$, with ε being the noise level [22]. The balancing principle aims to balance the regularization error and the propagation noise error.

Comparisons of different parameter choice rules were given in [3, 21, 22], including some recently proposed rules to ensure the convergence of the regularized approximation in the case that the noise level is many times under- or overestimated [22]. It is noted [3] that the best heuristic rules may perform better than the ones that use the noise level.

When iterative methods are used, such as CGLS, LSQR, Landweber's method or Kaczmarz's method, the iterations may be terminated appropriately so that the errors are controlled. In these cases, the iteration number plays the role of the regularization parameter [28, 29].

A well-known result of Bakushinskii [1] states that for an infinite-dimensional ill-posed problem, a parameter choice rule that does not explicitly use the noise level cannot yield a convergent regularization method as the noise level tends to zero. Nevertheless, the exact noise level is often not available for practical problems and the classical parameter choice rules such as the discrepancy principle are not applicable. So the heuristic rules are rather popular and often work reasonably in practice [3, 22].

2.3. Computational complexity

In this section, we shall discuss the computational complexity of the algorithms presented in section 2.1. The cost of each step of algorithm 2 (RSVD) is listed in table 1. Note that we use an economic QR factorization [17] to explicitly form the factor Q in step 3, and apply a thin SVD [17] in step 5. For a given matrix $A \in \mathbb{R}^{m\times n}$, the flops count of the classical SVD based on R-bidiagonalization is about $6mn^2 + 20n^3$ [7, 17], while the cost of algorithm 2 is only about $4mnl$. For the cases where singular values decay rapidly, we can choose $l \ll n$. According to the flops, the ratio of the cost for RSVD over that for the classical SVD is of the order $O(\frac{l}{n})$. Hence, algorithm 2 (RSVD) may be essentially less expensive than the classical SVD (CSVD).

The above statements can be applied to algorithm 3 (RSVD*), which is a variant of algorithm 2 for the case $m < n$. We can see that the major difference lies in step 5, where SVD is applied on a small matrix. Recall that our aim is to perform the SVD for a matrix that has a much smaller size compared to the original discrete system.

Note that if one uses a special structured random sampling matrix, for instance, a subsampled random Fourier transform [50], one may have a substantial gain in terms of execution time and the asymptotic complexity reduces to the order $O(mn\log l)$ for step 2 in algorithm 2.

In order to better understand the performance of the randomized algorithms (algorithms 2 and 3), we shall compare them with a deterministic method based on the rank revealing QR (RRQR) with column pivoting. The rank revealing factorization has been widely applied in the total least-squares problems, subset selection, regularization, low-rank approximation and nonsymmetric eigenproblems; see [8] and the references therein. Although the RRQR factorization may fail to reveal the numerical rank in some cases, it works quite well in practice, like Gaussian elimination with partial pivoting applied to a linear system. In the lth step of RRQR, we first find the column of largest norm and swap it with the lth column. Then the lth orthogonal transformation, for example, Householder matrix, is performed [11]. The RRQR with column pivoting yields

$$A\Pi = Q\begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix},$$

where Π is a permutation matrix, R_{11} is a well-conditioned $l \times l$ upper triangular matrix and R_{22} is sufficiently small. Let σ_l be the lth largest singular value. From the Courant–Fischer minimax theorem [17], we know that $\sigma_l(A) \geqslant \sigma_{\min}(R_{11})$ and $\sigma_{l+1}(A) \leqslant \sigma_{\max}(R_{22})$. Hence if $||R_{22}||$ is small, then A has at least $n - l$ small singular values, and it can be considered to have the numerical rank l, and the first l columns of $A\Pi$ form a well-conditioned basis for the range of A within an approximate accuracy of $\sigma_{l+1}(A)$ [19].

Algorithm 5 (RRQR–SVD). Given $A \in \mathbb{R}^{m\times n}$, use RRQR to achieve an SVD approximation $A \approx U\Sigma V^T$ with $U \in \mathbb{R}^{m\times l}$, $\Sigma \in \mathbb{R}^{l\times l}$ and $V \in \mathbb{R}^{n\times l}$.

```
1. Generate the Householder QR factorization of $A$ with column pivoting: $A\Pi \approx QR$.
2. Generate the thin SVD: $R^T = \hat{V}\Sigma\hat{U}^T$, where $\hat{V} \in \mathbb{R}^{n\times l}$, $\Sigma, \hat{U} \in \mathbb{R}^{l\times l}$.
3. Compute the $m \times l$ matrix $U = Q\hat{U}$.
4. Form the $n \times l$ matrix $V = \Pi\hat{V}$ via the permutation.
```

In algorithm 5, we use RRQR to achieve an approximate rank-l SVD. If we write the matrix $[R_{11}, R_{12}]$ as $R_{l\times n}$ and drop the small term R_{22}, we have the truncated RRQR [28], i.e., $A_{m\times n}\Pi \approx Q_{m\times l}R_{l\times n}$. One may improve the performance of the algorithm by using Stewart's pivoted QLP decomposition [44], but it will increase the total computational complexity.

The flops of algorithm 5 (RRQR–SVD) are also given in table 1, where the Householder QR with column pivoting for an $m \times n$ matrix with rank l needs $4mnl - 2(m+n)l^2 + \frac{4}{3}l^3$ flops [17]. We can see that its computational complexity is about the same order as algorithm 2. But it is very time consuming to permute the data required in RRQR, while most flops for RSVD are spent on the matrix–matrix multiplications, which are the so-called nice BLAS-3 operations, and the original large-scale matrix A is visited only twice. Furthermore, RSVD works as robustly and accurately as RRQR–SVD for almost all the numerical examples that we have tried; see section 4.

The new algorithm 4 can greatly reduce the size of the original problem by using the randomized techniques. There exist other reduction processes, e.g., the ones using the projection onto Krylov subspaces to reduce the size of the original discrete system (2), and iterative methods based on the Arnoldi process [6] are good examples. The application of l steps of the Arnoldi process to the matrix A with the initial vector b yields the decomposition

$$AW_l = W_lH_l + \eta_{l+1}w_{l+1}e_l^T \equiv W_{l+1}\bar{H}_l,$$

where $W_l^TW_l = I, W_le_1 = b/||b||, W_l^Tw_{l+1} = 0, H_l$ is an $l \times l$ upper Hessenberg matrix, and $\bar{H}_l$ is the upper Hessenberg-like matrix formed by H_l and $\eta_{l+1}e_l^T$. We then seek an approximate solution of the form $x_\mu = W_lz$ by minimizing

$$\left|\left|\begin{bmatrix}\bar{H}_l\\ \mu I\end{bmatrix}z - ||b||e_1\right|\right|.$$

As $\bar{H}_l$ is of size $(l+1)\times l$, this is a problem of smaller size.

For a large matrix, another solver based on the Lanczos bidiagonalization and the quadrature rules can be used for the problem reduction [5]. The Lanczos bidiagonalization procedure can be expressed as

$$AV_l = U_lC_l + \delta_{l+1}u_{l+1}e_l^T, \quad A^TU_l = V_lC_l^T,$$

where $V_l^TV_l = I$, $U_l^TU_l = I$, $U_l^Tu_{l+1} = 0$, $U_1e_1 = u_1 = b/||b||$ and C_l is an $l \times l$ lower bidiagonal matrix. Then the regularization techniques are applied for C_l [40]. This is a hybrid regularization using projection first and regularization second, instead of the direct or iterative regularization [28]. The central idea is to reduce the large problem to a much smaller one by projecting A onto some suitably chosen left and right subspaces like TSVD and TGSVD methods [28]. On the other hand, Gauss quadrature rules [16, 18] can also be used to estimate the solution and residual norms. Let $\phi(t) = (t+\mu)^{-2}$, then we can verify that $||b - Ax_\mu||^2 = \mu^2b^T\phi(AA^T)b$, which can be estimated by $\mu^2||b||^2e_1^T\phi(C_lC_l^T)e_1$ by applying the l-point Gaussian quadrature rule [5, 33]. We then need to calculate $(C_lC_l^T+\mu I)^{-1}e_1$, which can be obtained by minimizing

$$\left|\left|\begin{bmatrix}C_l^T\\ \mu^{1/2}I\end{bmatrix}y - \mu^{1/2}e_{l+1}\right|\right|.$$

This reduction process enables us again to solve a problem of smaller size. Here, the l-point Gaussian quadrature gives a lower bound of the residual norm, while the $(l+1)$-point Gauss–Radau quadrature provides a upper bound. The quality of the bounds depends on how well the function ϕ can be approximated by a polynomial on the spectrum of A [33]. Similarly, we can estimate the norm $||x_\mu||$ of the regularized solution and obtain its lower and upper bounds; see [5, 33] for details.

As seen from above, the Krylov subspace methods based on the Arnoldi process or Lanczos procedure can also greatly reduce the size of the large-scale regularized system (2), through which we can seek the approximate solution of the original system (2) by solving a series of smaller problems. But these reduction methods need to access the coefficient matrix A by l or $2l$ times and use the BLAS-2 operations, i.e., the matrix–vector multiplications. The Lanczos procedure even requires the evaluation of matrix–vector products involving the transpose of matrix A and reorthogonalization of the columns of U_l and V_l [6]. These are the processes which are known to require essential CPU times when the size of A is very large.

3. Error estimates

Our proposed regularization method combined with the randomized SVD (i.e., algorithm 4) consists of two stages. The first one is a stochastic process, which generates an approximate SVD of matrix A by a randomized algorithm, while the second stage is deterministic, using the resulting SVD approximation to select the regularization parameter and compute the regularized solution. In order to find out the performance of this new algorithm, we should understand the accuracy of the approximate regularized solution it provides and its performance compared with the classical methods. The accuracies of the randomized SVD and the approximate regularized solution will be analyzed in rest of this section, while its performance will be carried out numerically in the following section. We first introduce an important estimate [20, 42].

Lemma 1 [20, corollary 10.9]**.** *Suppose that $A \in \mathbb{R}^{m\times n}$ has the singular values $\sigma_1 \geqslant \cdots \geqslant \sigma_n$. Let Ω be an $n \times (k+p)$ standard Gaussian matrix with $k+p \leqslant \min\{m,n\}$ and $p \geqslant 4$, and Q be an orthonormal basis for the range of the sampled matrix $A\Omega$. Then*

$$||A - QQ^TA|| \leqslant (1 + 6\sqrt{(k+p)p\log p})\sigma_{k+1} + 3\sqrt{k+p}\left(\Sigma_{j>k}\sigma_j^2\right)^{1/2} \tag{8}$$

with probability not less than $1 - 3p^{-p}$.

Under a very mild assumption on p, the estimate (8) can be simplified [20]:

$$||A - QQ^TA|| \leqslant (1 + 9\sqrt{k+p}\sqrt{\min\{m,n\}})\sigma_{k+1}. \tag{9}$$

We see from algorithm 2 that $U\Sigma V^T = QW\Sigma V^T = QB^T = QQ^TA$; hence $||A - U\Sigma V^T|| = ||A - QQ^TA||$, and lemma 1 indicates that we can obtain a good SVD approximation with very high probability.

Next, we study the accuracy of the regularized solution x_μ generated by algorithm 4 through the approximated SVD. First, we recall that the condition number measures the sensitivity of the least-squares solution and demonstrates how the least-squares solution is affected by the perturbations in A and b.

Lemma 2 [17, theorem 5.3.1]**.** *Suppose A is of full rank, $x_{\rm LS}$ and $\hat{x}_{\rm LS}$ are given by*

$$x_{\rm LS} = \text{argmin}||Ax - b||, \quad \hat{x}_{\rm LS} = \text{argmin}||(A + \delta A)\hat{x} - (b + \delta b)||,$$

where A and δA are in $\mathbb{R}^{m\times n}$ with $m > n$, b and δb are in $\mathbb{R}^m$ and $||\delta A|| < \sigma_n(A)$. Define $\theta \in (0, \pi/2)$ by $\sin\theta = \frac{||b - Ax_{\rm LS}||}{||b||}$ and let $\epsilon = \max\left\{\frac{||\delta A||}{||A||}, \frac{||\delta b||}{||b||}\right\}$. Then we have

$$\frac{||\hat{x}_{\rm LS} - x_{\rm LS}||}{||x_{\rm LS}||} \leqslant \epsilon[\nu_{\rm LS}\sec\theta + (1 + \nu_{\rm LS}\tan\theta)\kappa(A)] + O(\epsilon^2), \tag{10}$$

where $\kappa(A)$ is the condition number $\kappa(A) = \sigma_1(A)/\sigma_n(A)$ and $\nu_{\rm LS} = ||Ax_{\rm LS}||/(\sigma_n(A)||x_{\rm LS}||)$.

Noting that $\nu_{\rm LS} \leqslant \kappa(A)$, the upper bound in (10) can be changed to the following one:

$$\frac{||\hat{x}_{\rm LS} - x_{\rm LS}||}{||x_{\rm LS}||} \leqslant \epsilon(2\sec\theta\kappa(A) + \tan\theta\kappa(A)^2) + O(\epsilon^2). \tag{11}$$

The sensitivity of the least-squares solution is roughly proportional to the quantity $\kappa(A) + \rho_{\rm LS}\kappa(A)^2$, where $\rho_{\rm LS} = ||b - Ax_{\rm LS}||$ and the factor $\kappa(A)^2$ cannot be improved [17].

Now consider the regularized system (2). Clearly, its regularized solution x_μ solves

$$\min\left|\left|\begin{bmatrix} A \\ \mu I\end{bmatrix}x - \begin{bmatrix} b \\ 0\end{bmatrix}\right|\right|. \tag{12}$$

Let $\mathcal{A} = \left[\begin{smallmatrix} A \\ \mu I\end{smallmatrix}\right]$, and its condition number is given by $\kappa(\mathcal{A}) = ||\mathcal{A}||||\mathcal{A}^\dagger||$, where $\mathcal{A}^\dagger$ is the Moore–Penrose generalized inverse [17]. We can establish the following important estimate on $\kappa(\mathcal{A})$.

Lemma 3. *Suppose A is of full rank. Then the condition number of $\mathcal{A}$ associated with the problem (12) is bounded by*

$$\kappa(\mathcal{A}) \leqslant \sqrt{\sigma_1^2 + \mu^2}\left(\frac{\mu}{\sigma_n^2 + \mu^2} + \max_{1\leqslant i\leqslant n}\frac{\sigma_i}{\sigma_i^2 + \mu^2}\right), \tag{13}$$

where $\sigma_1, \sigma_2, \ldots, \sigma_n$ are the singular values of A, and $\sigma_1 \geqslant \cdots \geqslant \sigma_n$.

Proof. Since $\mathcal{A}$ has full column rank, its Moore–Penrose inverse can be expressed as

$$\mathcal{A}^\dagger = (\mathcal{A}^T\mathcal{A})^{-1}\mathcal{A}^T = (A^TA+\mu^2I)^{-1}[A^T, \mu I].$$

The upper bound of the 2-norm of $\mathcal{A}^\dagger$ can be estimated as follows:

$$\begin{aligned}
||\mathcal{A}^\dagger|| &= \max_{||y||=1} ||(A^TA+\mu^2I)^{-1}[A^T,\mu I]y||\\
&= \max_{||[y_1^T,y_2^T]||=1} ||(A^TA+\mu^2I)^{-1}A^Ty_1 + \mu(A^TA+\mu^2I)^{-1}y_2||\\
&\leqslant \max_{||[y_1^T,y_2^T]||=1} ||(A^TA+\mu^2I)^{-1}A^Ty_1|| + \max_{||[y_1^T,y_2^T]||=1} \mu||(A^TA+\mu^2I)^{-1}y_2||\\
&\leqslant \max_{||[y_1^T,y_2^T]||=1} \frac{||(A^TA+\mu^2I)^{-1}A^Ty_1||}{||y_1||} + \max_{||[y_1^T,y_2^T]||=1} \frac{\mu||(A^TA+\mu^2I)^{-1}y_2||}{||y_2||}\\
&\leqslant \max_{y_1} \frac{||(A^TA+\mu^2I)^{-1}A^Ty_1||}{||y_1||} + \max_{y_2} \frac{\mu||(A^TA+\mu^2I)^{-1}y_2||}{||y_2||}\\
&= ||(A^TA+\mu^2I)^{-1}A^T|| + \mu||(A^TA+\mu^2I)^{-1}||.
\end{aligned}$$

Consider the SVD of A, namely $A = U\Sigma V^T$, where $\Sigma = \mathrm{diag}(\sigma_1,\ldots,\sigma_n) \in \mathbb{R}^{m\times n}$, $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices. We can check that

$$A^TA+\mu^2I = V(\Sigma^2+\mu^2I)V^T$$

and

$$(A^TA+\mu^2I)^{-1}A^T = V(\Sigma^2+\mu^2I)^{-1}\Sigma U^T,$$

which implies

$$||(A^TA+\mu^2I)^{-1}A^T|| = \max_{1\leqslant i\leqslant n}\frac{\sigma_i}{\sigma_i^2+\mu^2}, \quad ||(A^TA+\mu^2I)^{-1}|| = \frac{1}{\sigma_n^2+\mu^2}.$$

The 2-norm of $\mathcal{A}$ can be obtained by

$$||\mathcal{A}|| = \max_x \frac{||\mathcal{A}x||}{||x||} = \sqrt{\max_x \frac{||Ax||^2}{||x||^2} + \mu^2} = \sqrt{\sigma_1(A)^2+\mu^2}.$$

From the above, we derive that

$$\kappa(\mathcal{A}) = ||\mathcal{A}||||\mathcal{A}^\dagger|| \leqslant \sqrt{\sigma_1^2+\mu^2}\left(\frac{\mu}{\sigma_n^2+\mu^2} + \max_{1\leqslant i\leqslant n}\frac{\sigma_i}{\sigma_i^2+\mu^2}\right).$$

□

Note that the assumption that A has full rank in lemma 3 is unnecessary, since $\mathcal{A}$ is obviously of full column rank when $\mu \neq 0$. Because of the simple fact that $\frac{\sigma_i}{\sigma_i^2+\mu^2} \leqslant \frac{1}{2\mu}\,(\mu \neq 0)$ for all σ_i $(1 \leqslant i \leqslant n)$, we have the following simplified bound:

$$\kappa(\mathcal{A}) \leqslant \sqrt{\sigma_1^2+\mu^2}\left(\frac{\mu}{\sigma_n^2+\mu^2} + \frac{1}{2\mu}\right),$$

which can be further reduced to $\kappa(\mathcal{A}) \leqslant \frac{3}{2\mu}\sqrt{\sigma_1^2+\mu^2}$. A slightly tighter upper bound can be obtained for the case with $\mu \leqslant \sigma_n$:

$$\kappa(\mathcal{A}) \leqslant \frac{\sigma_n+\mu}{\sigma_n^2+\mu^2}\sqrt{\sigma_1^2+\mu^2}, \tag{14}$$

by noting that $\frac{\sigma_i}{\sigma_i^2+\mu^2} \leqslant \frac{\sigma_n}{\sigma_n^2+\mu^2}$ for $\mu \leqslant \sigma_n$.

For the extreme case of $\mu = 0$, namely without the regularization, the upper bounds in (13) and (14) are reduced to σ_1/σ_n, which is quite large for the ill-conditioned inverse system.

Under an appropriate regularization, the condition number $\kappa(\mathcal{A})$ can be well bounded as it is seen from (13).

By algorithm 2, we obtain an approximate SVD of A, i.e., $A \approx \tilde{A} = U\Sigma V^T$. Then algorithm 4 uses the approximation $\tilde{A}$ to determine the regularization parameter for use in the least-squares system

$$\min \left|\left| \begin{bmatrix} \tilde{A} \\ \mu I \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right|\right|. \tag{15}$$

This can be regarded as a perturbed version of (12). For the sake of simplicity, we define $\tilde{\mathcal{A}} = \left[\begin{smallmatrix} \tilde{A} \\ \mu I \end{smallmatrix}\right]$. Then, we have the following relation for the perturbation:

$$||\tilde{\mathcal{A}} - \mathcal{A}|| = ||\tilde{A} - A|| = ||U\Sigma V^T - A|| = ||A - QQ^TA||.$$

This can be estimated by lemma 1, and we know that it is bounded by the smallest singular values of A.

Now using lemmata 1–3, we can derive the following estimate.

Theorem 1. *Let $\sigma_1 \geqslant \cdots \geqslant \sigma_n$ be the singular values of matrix A, and*

$$c = 1 + 6\sqrt{(k+p)p\log p} + 3\sqrt{(k+p)(n-k)}, \quad C = c/\sigma_1.$$

Assume that algorithm 2 is performed with the Gaussian matrix $\Omega \in \mathbb{R}^{n\times(k+p)}$ to achieve the SVD approximation of matrix A, x_μ and $\hat{x}_\mu$ are the solutions of (12) and (15), respectively, with respect to the regularization parameter μ. Then, we have

$$\frac{||\hat{x}_\mu - x_\mu||}{||x_\mu||} \leqslant C\left(2\sec\theta\kappa + \tan\theta\kappa^2\right)\sigma_{k+1} + O(\sigma_{k+1}^2) \tag{16}$$

with probability not less than $1 - 3p^{-p}$, where κ is the upper bound of $\kappa(\mathcal{A})$ given by (13), and θ is defined by $\sin\theta = \rho_{\mathrm{LS}}/||b||_2$ with $\rho_{\mathrm{LS}} = \sqrt{||b - Ax_\mu||^2 + \mu^2||x_\mu||^2}$.

Proof. Assuming that $\tilde{A} = U\Sigma V^T$ is the SVD approximation of A achieved by algorithm 2 using the Gaussian matrix $\Omega \in \mathbb{R}^{n\times(k+p)}$, we have

$$\tilde{A} = U\Sigma V^T = QQ^TA.$$

Then, it follows from lemma 1 that

$$||\tilde{A} - A|| = ||A - QQ^TA|| \leqslant c\sigma_{k+1}$$

with probability not less than $1 - 3p^{-p}$.

Let $\mathcal{A} = \left[\begin{smallmatrix} A \\ \mu I \end{smallmatrix}\right]$ and define $\delta\mathcal{A} = \left[\begin{smallmatrix} \tilde{A} \\ \mu I \end{smallmatrix}\right] - \left[\begin{smallmatrix} A \\ \mu I \end{smallmatrix}\right]$. We can check that $||\delta\mathcal{A}|| = ||\tilde{A} - A|| \leqslant c\sigma_{k+1}(A)$. Using the fact that $\sigma_1(\mathcal{A}) \geqslant \sigma_1(A)$, we have

$$\frac{||\delta\mathcal{A}||}{||\mathcal{A}||} \leqslant \frac{c\sigma_{k+1}}{\sigma_1(\mathcal{A})} \leqslant C\sigma_{k+1}. \tag{17}$$

Let $\mathcal{B} = \left[\begin{smallmatrix} b \\ 0 \end{smallmatrix}\right]$. By definition, x_μ and $\hat{x}_\mu$ are given, respectively, by

$$x_\mu = \mathrm{argmin}||\mathcal{A}x - \mathcal{B}||, \quad \hat{x}_\mu = \mathrm{argmin}||(\mathcal{A} + \delta\mathcal{A})x - \mathcal{B}||.$$

Then using (17) and (11) or lemma 2, we obtain

$$\frac{||\hat{x}_\mu - x_\mu||}{||x_\mu||} \leqslant C\sigma_{k+1}\left(2\sec\theta\kappa + \tan\theta\kappa^2\right) + O(\sigma_{k+1}^2),$$

where the upper bound, $\kappa = \sqrt{\sigma_1^2 + \mu^2\left(\frac{\mu}{\sigma_n^2+\mu^2} + \max_{1\leqslant i\leqslant n}\frac{\sigma_i}{\sigma_i^2+\mu^2}\right)}$ is given by lemma 3, and $\sin\theta = \rho_{\mathrm{LS}}/||b||_2$ with $\rho_{\mathrm{LS}} = ||\mathcal{B} - \mathcal{A}x_\mu|| = \sqrt{||b - Ax_\mu||^2 + \mu^2||x_\mu||^2}$. □

Since κ is well bounded using the regularization parameter, the upper bound in (16) is of order $O(\sigma_{k+1})$. Hence, the relative error of the regularized solution obtained from algorithm 4 and the regularized solution generated by the classical methods is small.

Remark 3.1. The right-hand side b in theorem 1 can be regarded as the measured data containing noise. To explicitly take into account the noisy data, we use b to stand for the exact data and b^δ for the noisy data. Then, we can consider the regularized solutions of the following two problems (with $\mathcal{B}^\delta = [b^\delta, 0]^T$):

$$x_\mu = \text{argmin}||\mathcal{A}x - \mathcal{B}^\delta||, \quad \hat{x}_\mu = \text{argmin}||(\mathcal{A} + \delta\mathcal{A})x - \mathcal{B}^\delta||,$$

and obtain the same error estimates to (16) by following the arguments in the proof of theorem 1. More explicitly, if we assume the relative noise level of the form $\delta = \frac{||b^\delta - b||}{||b||} \leqslant C\sigma_{k+1}$ and consider the solutions to the following two problems:

$$x_\mu = \text{argmin}||\mathcal{A}x - \mathcal{B}||, \quad \hat{x}_\mu = \text{argmin}||(\mathcal{A} + \delta\mathcal{A})x - \mathcal{B}^\delta||,$$

then we have $\epsilon = \max\left\{\frac{||\delta\mathcal{A}||}{||\mathcal{A}||}, \delta\right\} \leqslant C\sigma_{k+1}$ by using (17), and obtain similar error estimates to (16) by using lemma 2. We note that x_μ above is the regularized solution of the original problem without noise and $\hat{x}_\mu$ is the regularized solution of our randomized algorithm for the noise contaminated problem.

4. Numerical experiments

In this section, we apply the newly proposed algorithm 4 to 14 examples of different linear inverse problems to illustrate the performance of the algorithm for solving the discrete large-scale inverse system $Ax = b$, which is converted to the solution of the least-squares system (2).

Example 1. (Cmrs, [6]). Let C_n be an auxiliary matrix C_n, such that $C_n = (c_{jk})_{j,k=1}^n$ with entries

$$c_{jk} = \exp\left(\pi\frac{2j-1}{4n-2}\cos\pi\frac{2k-1}{2n-1}\right).$$

Suppose C_n has the SVD, $C_n = U_n\Sigma_nV_n^T$, then we define the testing matrix $A \in \mathbb{R}^{2n\times n}$ to be $A = U_{2n}\Sigma V_n^T$, where $\Sigma \in \mathbb{R}^{2n\times n}$ is a diagonal matrix with its diagonal elements being $\exp(-\frac{2}{5}(j-1))$ for $j = 1, 2, \ldots, n$, U_{2n} and V_n are the left and right orthogonal matrices in the SVDs of C_{2n} and C_n, respectively. Let x be a vector with standard normal distributed entries, i.e., $x = \mathsf{randn}(n, 1)$, then we define the right-hand side vector to be $b = Ax$.

Example 2. (Rst, [42]). We form matrix A by $A = U_A\Sigma_AV_A^T$, where U_A and V_A are a $2n \times 2n$ Hadamard matrix (a unitary matrix with entries $\pm 1/\sqrt{2n}$) and an $n \times n$ Hadamard matrix, respectively, and Σ_A is a $2n \times n$ diagonal matrix with its diagonal entries given by

$$\sigma_j = \begin{cases} (\sigma_{k+1})^{\lfloor j/2\rfloor/5}, & j = 1, 2, \ldots, 10, \\ \sigma_{k+1}\frac{n-j}{n-11}, & j = 11, 12, \ldots, n, \end{cases}$$

where $\lfloor j\rfloor$ is the greatest integer less than or equal to j. In our testings, we choose $k = 10$ and $\sigma_{k+1} = 1.0 \times 10^{-6}$, and form vector b by $b = Ax$ with x being a vector of all ones.

Examples 3–14 (Matrices from the regularization tools [30])**.** We choose the remaining 12 testing problems from Hansen's regularization tools (version 4.1) [27, 30]; see table 2. The parameter N_σ in the third column of table 2 represents the number of singular values which are not less than 10^{-6} when the matrix sizes are chosen to be $n = 100$ except in Parallax and Rst. Among these 12 matrices, there are three matrices, namely Deriv2, Heat and Phillips, in

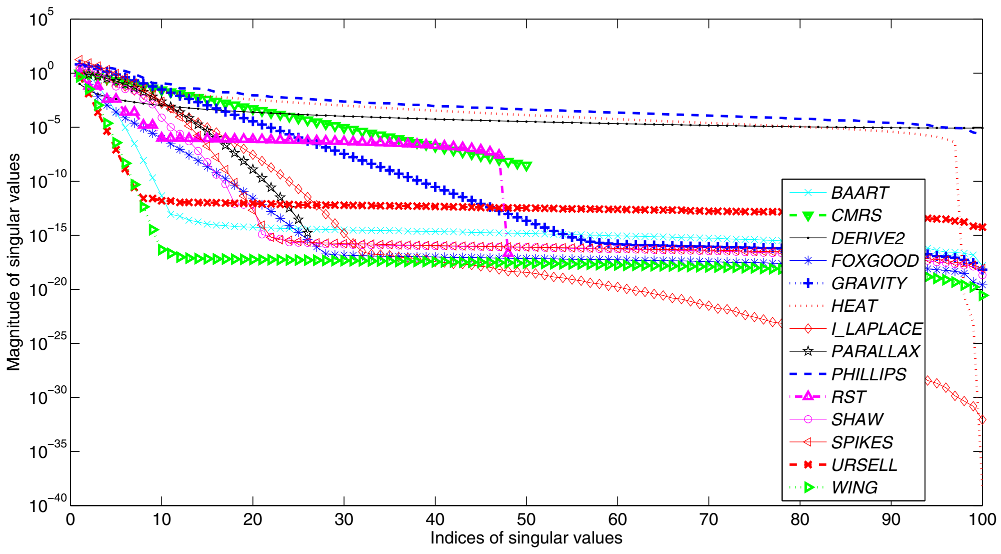

Figure 1. The decay of singular values of the 14 testing problems, with the matrix size set to 100 × 100 except that Parallax, Cmrs and Rst are of sizes 26 × 100, 100 × 50 and 96 × 48, respectively.

Table 2. The 14 testing matrices, with N_σ being the number of singular values larger than 10^{-6}.

No.	Matrix	N_σ	Descriptions (see also [30])
1	Baart	6	Discretization of the first kind Fredholm integral equation
2	Cmrs	35	Matrix of size $n \times (n/2)$ adapted from example 4.1 in [6]
3	Deriv2	100	Computation of the second derivative
4	Foxgood	9	Severely ill-posed test problem
5	Gravity	25	1D gravity surveying problem
6	Heat	95	Inverse heat equation
7	I_laplace	17	Inverse Laplace transformation
8	Parallax	15	Stellar parallax problem with the size of $26 \times n$
9	Phillips	100	Phillips' famous test problem
10	Rst	10	Matrix of size $n \times (n/2)$ adapted from the example in [42]
11	Shaw	12	1D image restoration model
12	Spikes	14	Test problem with a 'spiky' solution
13	Ursell	4	Integral equation with no square integrable solution
14	Wing	4	Test problem with a discontinuous solution

which the singular values are not well separated and decay slowly, while the singular values of other matrices decay rapidly; see figure 1.

In all our numerical experiments, the observation data b^δ are generated from the exact data b by adding noise in the form

$$b^\delta = b + \delta||b||\frac{s}{||s||} = b + \varepsilon\frac{s}{||s||},$$

where s is a random vector, $s = \mathsf{randn}(n, 1)$ if not specified otherwise, $\varepsilon = \delta||b||$ is the so-called noise level and δ is the relative noise level [29].

We shall compare the performance of the new algorithm 4 with some other algorithms, and test two different regularization techniques, i.e., Tikhonov regularization and TSVD, and five different parameter choice strategies, i.e., the L-curve rule, the GCV rule, the quasi-optimality criterion, the Auchmuty estimator $\hat{E}_3$ and the discrepancy principle. The algorithmic performance is checked in the following aspects: the computed regularization parameters compared with the optimal parameters, the accuracies of the regularized solutions compared with the optimal solutions and the CPU times of algorithms.

In the subsequent numerical tables, we shall use the following notation. μ stands for the regularization parameter, err for the relative error $||x - x_\mu||_2/||x||_2$ of the regularized solution x_μ and the exact solution x, and the optimal parameter μ for Tikhonov regularization is the minimizer of $\psi(\rho) = ||x - x_\mu||$ with $\mu = 10^\rho$ by the fminsearch function of Matlab, using as a starting point the logarithm of the parameter furnished by the GCV [4]. The optimal solution for TSVD regularization and its corresponding regularization parameter k are achieved similarly, i.e., we compute the regularized solutions with a range of parameters k and choose the optimal one to minimize the norm of the error between the regularized and exact solutions; *ratio* stands for the norm of the error of the regularized solution divided by the norm of the error corresponding to the optimal solution when the same regularization technique is used with various regularization parameter choice rules; ratio* stands for the error norm of the approximate solution obtained via RSVD divided by the error norm of the approximate solution to the original large-scale system via CSVD when the same regularization technique and same parameter choice rule are used.

Table 3 shows the numerical results when the Tikhonov regularization using RSVD and the classical SVD, respectively, are applied to example 5, Gravity, for size $n = 1000$, while table 4 shows the same as table 3 except that Tikhonov regularization is replaced by the TSVD. One can observe from these two tables that the results by the new algorithm using RSVD are quite comparable with that by CSVD. But as we shall find out later on (cf tables 6–8), the new algorithm is much less expensive.

When the matrices are too large, the classical SVD does not work, either due to the memory limitation or the numerical instability of smallest singular values. For the supercomputer that is accessible to us, we find that when the matrix size reaches about 8000, either CSVD does not work or it generates singular values with very poor accuracies. But the new algorithm 4 with the help of the randomized SVD works well up to the matrix size $n = 100\,000$. Table 5 shows the results by Tikhonov regularization combined with RSVD (i.e., algorithm 4) and TSVD regularization combined with the RSVD for four examples with the matrix sizes $n = 30\,000$ and $n = 100\,000$. The regularization parameters k (for TSVD) or μ (for RSVD), the relative errors and the ratios are also given in table 5, but not the ‘ratio*’, since the matrix sizes are too large to run for the classical SVD. Figure 2 gives the computed solutions for example 11, Shaw, with $n = 100\,000$, when Tikhonov regularization is applied with five different regularization parameter choice strategies. We can see from both table 5 and figure 2 that the new algorithm 4 with RSVD works quite satisfactorily even when the index l is very small, $l = 50$, and all the computed solutions by different regularization parameter choice rules are indistinguishable, except for the one by the Auchmuty estimator.

Next we carry out some numerical experiments to compare the CPU times of different algorithms. The relative noise level is set to be $\delta = 1\%$ unless otherwise specified. The sizes of the testing matrices are all set to be $n = 1000$ except that the matrices for Parallax, Cmrs and Rst are of sizes 26×1000, 1000×500 and 1024×512, respectively. The actual CPU times are recorded, and the following tests are done using Matlab R2008b in a laptop with Intel Core 2 Duo P8400 2.26G and 2GB DDR2.

Table 3. Tikhonov regularization using CSVD and RSVD on the matrix Gravity with size $n = 1000$. Notations 'err', 'ratio' and 'ratio*' are defined in section 4. The results in columns 3–8 on the left correspond to Gaussian noise, while the results in columns 9–14 on the right correspond to uniformly distributed noise in $[-1,1]$.

		Tikhonov + CSVD											
		Optimal	L-curve	GCV	Quasiopt	$\hat{E}_3$	Discrep	Optimal	L-curve	GCV	Quasiopt	$\hat{E}_3$	Discrep
$\delta = 10^{-4}$	μ	6.10E-3	5.84E-4	3.99E-3	6.65E-3	4.37E-2	7.99E-3	5.00E-3	5.84E-4	8.92E-4	1.08E-2	4.29E-2	9.04E-3
	err	5.22E-3	2.29E-2	5.82E-3	5.24E-3	1.24E-2	5.41E-3	3.53E-3	3.70E-2	2.79E-2	5.66E-3	1.24E-2	4.98E-3
	ratio	1	4.39	1.11	1.00	2.38	1.04	1	10.5	7.91	1.60	3.51	1.41
$\delta = 10^{-2}$	μ	4.24E-2	4.72E-2	7.58E-2	9.25E-2	5.42E-1	1.21E-1	1.11E-1	5.44E-2	6.55E-2	1.23E-1	5.42E-1	1.29E-1
	err	1.59E-2	1.61E-2	2.00E-2	2.22E-2	5.75E-2	2.54E-2	2.67E-2	3.44E-2	3.09E-2	2.68E-2	5.65E-2	2.70E-2
	ratio	1	1.01	1.26	1.40	3.62	1.60	1	1.29	1.16	1.00	2.12	1.01
		Tikhonov + RSVD($l = 20$)											
$\delta = 10^{-4}$	μ	6.12E-3	1.55E-4	3.99E-3	6.66E-3	4.37E-2	8.27E-3	4.99E-3	2.86E-4	8.57E-4	1.08E-2	4.29E-2	8.89Ee-3
	err	5.23E-3	5.05E-2	5.84E-3	5.25E-3	1.24E-2	5.46E-3	3.51E-3	5.18E-2	2.87E-2	5.66E-3	1.24E-2	4.91E-3
	ratio	1	9.66	1.12	1.00	2.37	1.04	1	14.8	8.17	1.61	3.53	1.40
	ratio*	1.00	2.21	1.00	1.00	1.00	1.01	0.99	1.40	1.03	1.00	1.00	0.99
$\delta = 10^{-2}$	μ	4.24E-2	8.77E-3	7.58E-2	9.26E-2	5.42E-1	1.21E-1	1.12E-1	8.24E-3	6.54E-2	1.23E-1	5.42E-1	1.29E-1
	err	1.59E-2	8.36E-2	2.00E-2	2.22E-2	5.75E-2	2.54E-2	2.67E-2	2.20E-1	3.09E-2	2.68E-2	5.65E-2	2.70E-2
	ratio	1	5.26	1.26	1.40	3.62	1.60	1	8.26	1.16	1.00	2.12	1.01
	ratio*	1.00	5.19	1.00	1.00	1.00	1.00	1.00	6.40	1.00	1.00	1.00	1.00

Table 4. TSVD regularization using CSVD and RSVD on the matrix GRAVITY with size $n = 1000$. Notations k, 'err', 'ratio' and 'ratio*' are defined in section 4. The results in columns 3–7 on the left correspond to Gaussian noise, while the results in columns 8–12 on the right correspond to uniformly distributed noise in $[-1,1]$.

		TSVD + CSVD									
		Optimal	L-curve	GCV	Quasiopt	$\hat{E}_3$	Optimal	L-curve	GCV	Quasiopt	$\hat{E}_3$
$\delta = 10^{-4}$	k	13	15	12	20	8	13	15	15	13	8
	err	3.30E-3	2.65E-2	4.68E-3	2.08E-1	1.93E-2	4.53E-3	4.39E-2	4.39E-2	4.53E-3	1.93E-2
	ratio	1	8.03	1.42	63.0	5.85	1	9.69	9.69	1.00	4.27
$\delta = 10^{-2}$	k	9	10	7	8	3	8	8	8	7	3
	err	1.71E-2	2.94E-2	2.88E-2	2.19E-2	1.65E-1	2.24E-2	2.24E-2	2.24E-2	2.93E-2	1.65E-1
	ratio	1	1.72	1.68	1.28	9.65	1	1.00	1.00	1.31	7.35
		TSVD + RSVD($l = 20$)									
$\delta = 10^{-4}$	k	13	16	12	16	8	13	14	15	17	8
	err	3.30E-3	2.71E-2	4.68E-3	2.71E-2	1.93E-2	4.53E-3	1.66E-2	4.43E-2	4.68E-2	1.93E-2
	ratio	1	8.21	1.42	8.21	5.85	1	3.66	9.78	1.03	4.27
	ratio*	1.00	1.02	1.00	0.13	1.00	1.00	0.38	1.01	1.03	1.00
$\delta = 10^{-2}$	k	9	7	7	8	3	8	8	8	7	3
	err	1.71E-2	2.88E-2	2.88E-2	2.19E-2	1.65E-1	2.24E-2	2.24E-2	2.24E-2	2.93E-2	1.65E-1
	ratio	1	1.68	1.68	1.28	9.65	1	1.00	1.00	1.31	7.35
	ratio*	1.00	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 5. Tikhonov regularization and TSVD regularization using RSVD for two large-scale cases of sizes $n = 30\,000$ and $n = 100\,000$.

		Tikhonov regularization + RSVD($l = 50$)					TSVD regularization + RSVD($l = 50$)				
($n = 30\,000$, $\delta = 10^{-4}$)		Optimal	L-curve	GCV	Quasiopt	$\hat{E}_3$	Optimal	L-curve	GCV	Quasiopt	$\hat{E}_3$
Foxgood	μ or k	1.45E-04	3.95E-06	1.57E-04	2.17E-04	4.53E-03	5	6	5	5	2
	err	3.67E-04	8.63E-02	3.94E-04	7.78E-04	1.19E-02	1.03E-03	2.33E-03	1.03E-03	1.03E-03	3.11E-02
	ratio	1	236	1.07	2.12	32.5	1	2.26	1.00	1.00	30.3
Gravity	μ or k	2.86E-03	2.59E-05	5.99E-04	2.22E-03	4.37E-02	14	15	14	15	8
	err	3.04E-03	6.55E-02	8.17E-03	3.17E-03	1.24E-02	2.76E-03	4.18E-03	2.76E-03	4.18E-03	1.93E-02
	ratio	1	21.5	2.68	1.04	4.09	1	1.52	1.00	1.52	7.01
Phillips	μ or k	5.95E-03	1.34E-03	4.75E-03	5.36E-03	4.14E-02	25	32	21	4	9
	err	1.46E-03	4.16E-03	1.42E-03	1.38E-03	5.94E-03	1.35E-03	2.59E-03	1.62E-03	3.30E-01	1.29E-02
	ratio	1	2.85	0.97	0.95	4.07	1	1.93	1.21	245	9.60
Shaw	μ or k	1.11E-05	1.46E-05	1.28E-04	2.39E-04	7.23E-03	11	9	9	11	6
	err	1.84E-02	1.94E-02	2.93E-02	3.10E-02	4.74E-02	2.25E-02	3.21E-02	3.21E-02	2.25E-02	1.10E-01
	ratio	1	1.05	1.59	1.68	2.57	1	1.43	1.43	1.00	4.90
		Tikhonov regularization + RSVD($l = 100$)					TSVD regularization + RSVD($l = 100$)				
($n = 100\,000$, $\delta = 10^{-5}$)		Optimal	L-curve	GCV	Quasiopt	$\hat{E}_3$	Optimal	L-curve	GCV	Quasiopt	$\hat{E}_3$
Foxgood	μ or k	3.45E-05	3.22E-07	2.94E-05	5.33E-05	1.20E-03	6	8	6	8	3
	err	1.98E-04	6.48E-02	2.21E-04	3.16E-04	4.45E-03	3.77E-04	8.90E-04	3.77E-04	8.90E-04	7.21E-03
	ratio	1	327	1.11	1.59	22.46	1	2.36	1.00	2.36	19.14
Gravity	μ or k	2.64E-04	1.84E-06	1.98E-04	2.01E-04	1.11E-02	17	19	16	19	10
	err	1.03E-03	4.07E-02	1.06E-03	1.05E-03	5.92E-03	8.73E-04	1.25E-03	1.18E-03	1.25E-03	9.47E-03
	ratio	1	39.6	1.03	1.03	5.75	1	1.43	1.35	1.43	10.84
Phillips	μ or k	1.38E-03	1.62E-04	1.01E-03	1.52E-03	1.18E-02	37	74	37	4	12
	err	4.50E-04	2.73E-03	4.91E-04	4.50E-04	2.06E-03	4.56E-04	2.39E-03	4.56E-04	3.30E-01	4.19E-03
	ratio	1	6.07	1.09	1.00	4.58	1	5.24	1.00	725	9.19
Shaw	μ or k	2.32E-06	8.51E-07	1.04E-05	1.51E-05	2.69E-03	12	13	10	11	7
	err	6.49E-03	3.51E-02	1.78E-02	1.86E-02	4.25E-02	1.71E-02	1.37E-01	1.94E-02	1.92E-02	4.76E-02
	ratio	1	5.41	2.74	2.86	6.56	1	8.03	1.14	1.13	2.79

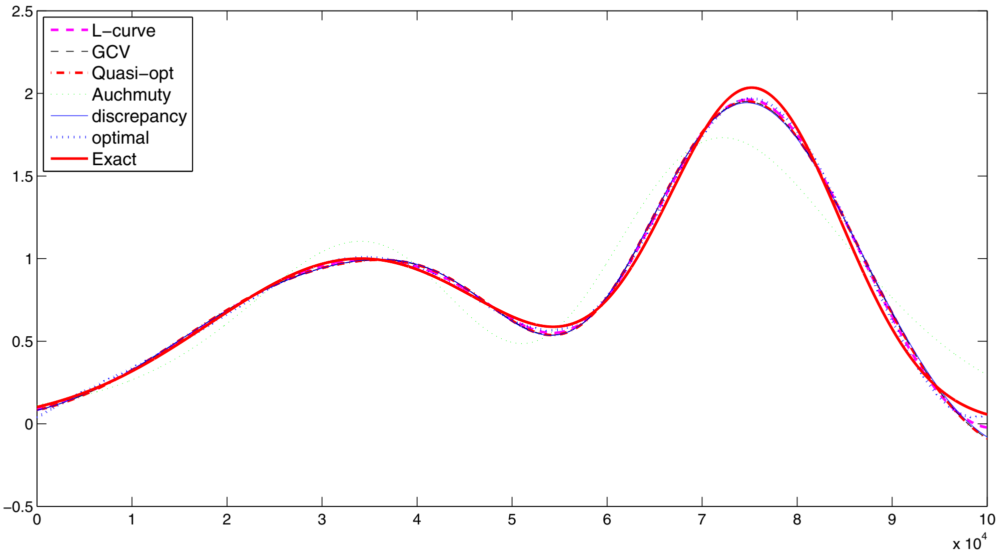

Figure 2. The Tikhonov regularized solutions for example Shaw of size $n = 100\,000$ using RSVD ($l = 50$), with 1% relative noise.

Table 6. Comparison of the computation time and solution accuracy when using RRQR–SVD and RSVD. The size of testing matrices is 2000, and $l = 20$. The CPU time for algorithm 6 to achieve the approximated SVD is $t_{\text{RRQR–SVD}}$ (in seconds); and t_{RSVD} is the CPU time of RSVD. Here err_{Lc} and err_{GCV} are the computed relative errors via using the L-curve and the GCV function, respectively.

	RRQR–SVD			RSVD		
Matrix	$t_{\text{RRQR–SVD}}$	err_{Lc}	err_{GCV}	t_{RSVD}	err_{Lc}	err_{GCV}
Baart	1.37	3.91E-01	3.88E-01	0.057	1.53E-01	1.63E-01
Deriv2	1.36	2.55E-01	2.54E-01	0.057	1.90E-01	2.11E-01
Foxgood	1.38	2.84E-01	2.84E-01	0.057	7.78E-02	3.05E-02
Gravity	1.37	1.58E-01	1.48E-01	0.057	2.95E-01	2.06E-02
Heat	1.39	1.14E-01	1.14E-01	0.058	1.21E-01	9.19E-02
I_laplace	1.36	5.26E-01	5.26E-01	0.130	2.92E-01	2.21E-01
Phillips	1.35	1.15E-01	1.13E-01	0.057	3.37E-02	1.48E-02
Shaw	1.36	2.15E-01	2.15E-01	0.058	4.44E-02	4.85E-02
Spikes	1.66	5.46E-01	5.28E-01	0.057	5.57E-01	5.23E-01
Wing	1.33	8.25E-01	8.25E-01	0.057	5.97E-01	6.04E-01

Tables 7 and 8 show the following CPU times: t_{CSVD} and t_{RSVD} for computing the classical SVD and the randomized SVD, respectively, t_{Lc} (resp. t_{GVC}) for generating the L-curve (resp. the GCV curve). In addition, the computed regularization parameters μ_{Lc} or μ_{GVC} and the relative errors ‘err’ of the regularized solutions are also presented. We have used Hansen’s regularization tools with 200 points to generate each L-curve or GCV curve [30].

As we can see from the total CPU times, i.e., column ‘T’ in table 7, the new algorithm using RSVD is about 30 times faster than the method using the classical SVD when the L-curve rule is used; the new algorithm is about 100 times faster when the GCV rule is used; see table 8. If we just compare the CPU times of SVD, not counting any regularization

Table 7. Comparison of the computation times for the matrix of size 1000 with the L-curve to locate the regularization parameter. In RSVD, we choose $l = 20$. $T = t_{\mathrm{CSVD}} + t_{Lc}$ or $t_{\mathrm{RSVD}} + t_{Lc}$. μ_{Lc} stands for the regularization parameter that corresponds to the L-curve's corner and err is the relative error of the computed solution. There are no exact solutions for the cases PARALLAX and URSELL.

	CSVD + L-curve					RSVD + L-curve				
Matrix	t_{CSVD}	t_{Lc}	T (s)	μ_{Lc}	err	t_{RSVD}	t_{Lc}	T (s)	μ_{Lc}	err
BAART	10.7	0.142	10.8	1.39E-02	1.53E-01	0.019	0.251	0.270	1.63E-03	1.36E-01
CMRS	1.51	0.129	1.63	1.37E-03	9.83E-01	0.012	0.278	0.290	6.60E-02	9.93E-01
DERIV2	10.7	0.125	10.8	6.74E-04	2.00E-01	0.020	0.240	0.260	3.89E-04	1.96E-01
FOXGOOD	11.1	0.125	11.3	4.50E-03	1.78E-02	0.019	0.247	0.266	1.19E-03	3.91E-02
GRAVITY	10.7	0.118	10.9	4.72E-02	1.61E-02	0.020	0.245	0.264	8.77E-03	8.36E-02
HEAT	8.35	0.125	8.47	3.19E-01	8.35E-01	0.019	0.234	0.253	2.12E-03	1.28E-01
I_LAPLACE	5.28	0.214	5.49	6.81E+00	8.46E-01	0.046	0.253	0.299	1.43E-03	3.41E-01
PARALLAX	0.01	0.185	0.19	6.61E-02	–	0.005	0.255	0.260	6.33E-02	–
PHILLIPS	8.98	0.150	9.13	5.15E-02	3.24E-02	0.024	0.297	0.321	2.75E-02	5.76E-02
RST	3.60	0.156	3.75	6.85E-03	1.14E-02	0.013	0.308	0.321	1.86E-03	4.42E-02
SHAW	10.7	0.148	10.8	1.74E-02	6.05E-02	0.020	0.252	0.271	1.69E-03	1.07E-01
SPIKES	10.7	0.148	10.8	5.37E-01	6.47E-01	0.020	0.251	0.271	5.05E-02	6.68E-01
URSELL	11.8	0.153	11.9	1.83E-03	–	0.019	0.246	0.265	3.52E-05	–
WING	10.8	0.152	10.9	1.57E-03	6.02E-01	0.019	0.249	0.268	2.35E-04	6.08E-01

Table 8. Comparison of the computation time for the matrices of size 1000 when using CSVD or RSVD ($l = 20$) together with GCV. Here, μ_{GVC} is the regularization parameter which minimizes the GCV function. Other parameters are similar to those in table 7.

	CSVD + GCV					RSVD + GCV				
Matrix	t_{CSVD}	t_{GVC}	T (s)	μ_{GVC}	err	t_{RSVD}	t_{GVC}	T (s)	μ_{GVC}	err
BAART	10.6	0.053	10.7	5.63E-03	1.23E-01	0.016	0.072	0.088	5.63E-03	1.23E-01
CMRS	1.45	0.058	1.51	6.98E-04	9.83E-01	0.009	0.097	0.106	5.65E-04	9.83E-01
DERIV2	10.7	0.087	10.8	4.42E-04	1.87E-01	0.016	0.081	0.097	4.11E-04	1.92E-01
FOXGOOD	11.2	0.053	11.2	6.07E-03	2.06E-02	0.018	0.079	0.098	6.09E-03	2.06E-02
GRAVITY	10.8	0.053	10.9	7.58E-02	2.00E-02	0.017	0.081	0.097	7.58E-02	2.00E-02
HEAT	8.19	0.092	8.29	1.73E-03	6.62E-02	0.016	0.085	0.101	2.52E-03	9.24E-02
I_LAPLACE	5.26	0.068	5.33	2.06E-02	2.04E-01	0.043	0.072	0.115	2.06E-02	2.04E-01
PARALLAX	0.00	0.077	0.08	5.33E-15	–	0.003	0.078	0.081	1.14E-02	–
PHILLIPS	8.99	0.070	9.06	7.76E-02	2.12E-02	0.017	0.095	0.112	7.78E-02	1.93E-02
RST	3.59	0.070	3.66	1.21E-02	4.96E-03	0.010	0.093	0.103	1.21E-02	4.96E-03
SHAW	10.7	0.073	10.8	9.41E-03	5.18E-02	0.016	0.075	0.091	9.42E-03	5.18E-02
SPIKES	10.7	0.070	10.7	5.11E-01	6.47E-01	0.016	0.072	0.089	5.11E-01	6.47E-01
URSELL	11.8	0.081	11.8	4.52E-08	–	0.016	0.067	0.083	4.13E-08	–
WING	10.8	0.070	10.8	1.48E-03	6.02E-01	0.017	0.072	0.089	1.48E-03	6.02E-01

technique, we find that RSVD (with $l = 20$) is almost 500 times faster than CSVD. The advantage of the new algorithm with RSVD becomes much more significant when the sizes of the systems become larger. For example, when the matrix size is 5000, RSVD is about 1000 times faster than the classical SVD.

We have also compared our randomized algorithm with the deterministic method, algorithm 6 (RRQR–SVD). In order to ensure the fairness of the comparisons for the CPU times, we have implemented step 1 in algorithm 6 in language C by calling the mexFunction in MATLAB. For the matrix of size 2000, CSVD needs more than 80 s on average, and RRQR–SVD is about 60 times faster than CSVD to achieve an SVD approximation, while RSVD is more than 1000 times faster; see table 6. In terms of CPU times, RRQR–SVD performs faster than CSVD, but much slower and less efficiently than RSVD, since it needs to access the data many times to choose the columns of maximum norm and to permute the data, while the product involving A in the randomized algorithm can be evaluated in a single sweep and is amenable to the BLAS-3 operations.

Finally, we make a remark that one may use the GPU technique to accelerate the computation of RSVD, especially steps 2 and 4 in algorithm 2. These two steps are the most expensive steps in RSVD, though they are just two matrix–matrix multiplications. These two steps are well suited for the GPU implementation [32, 43]. We can use the mexFunction in Matlab and CUDA programming to implement these two steps on GPU.

5. Concluding remarks

In this work, we have proposed an algorithm for solving large-scale discrete ill-conditioned linear problems arising from the discretization of linear and nonlinear inverse problems, based on the randomized SVD and some existing regularization techniques. The algorithm preserves basically the same successful locations of the regularization parameters and achieves about the same accurate regularized solutions as the classical SVD, but with much less computational effort. More importantly, the classical SVD may not work or work very poorly for large-scale discrete inverse systems due to the computational complexity, the numerical instability and memory limitation. Compared with the deterministic approach, such as the rank revealing QR factorization, the new algorithm also demonstrates much better performance. The RRQR-based SVD approximation is slower since it needs to access the data many times to choose the columns of the maximum norm and permute the data. By combining the randomized SVD with classical regularization techniques, our new algorithm can convert the large-scale problems to small-scale ones so that the SVD-type methods can still be applied, and with a reasonably acceptable accuracy for the approximated solution. The error estimates of the approximate solutions have been derived, and many numerical experiments have demonstrated that the new algorithm can indeed reduce the problem size greatly and save the entire computational time essentially. The new algorithm admits obvious out-of-core and parallel implementation, hence is also well suited for the GPU acceleration.

Acknowledgments

The authors would like to thank two anonymous referees and the board member for their numerous insightful and constructive comments. In addition, the authors are very grateful to Dr. Zongyan Cao for helping us run several large-scale examples on the supercomputer Deepcomp7000 at Superconducting Center, Computer Network Information Center of Chinese Academy of Sciences. The work of HX was supported by the National Natural Science Foundation of China under grants 10901125, 91130022 and 11161130003. The work of JZ was substantially supported by Hong Kong RGC grant (Project 404611) and CUHK Focused Investment Scheme 2012/2014.

References

[1] Bakushinskii A B 1984 Remarks on choosing a regularization parameter using the quasi-optimality and ratio criterion *USSR Comput. Math. Math. Phys.* **24** 181–2

[2] Banks H and Kunisch K 1989 *Parameter Estimation Techniques for Distributed Systems* (Boston, MA: Birkhäuser)

[3] Bauer F and Lukas M A 2011 Comparing parameter choice methods for regularization of ill-posed problems *Math. Comput. Simul.* **81** 1795–841

[4] Brezinski C, Rodriguez G and Seatzu S 2008 Error estimates for linear systems with applications to regularization *Numer. Algorithms* **49** 85–104

[5] Calvetti D, Golub G H and Reichel L 1999 Estimation of the L-curve via Lanczos bidiagonalization *BIT Numer. Math.* **39** 603–19

[6] Calvetti D, Morigi S, Reichel L and Sgallari F 2000 Tikhonov regularization and the L-curve for large discrete ill-posed problems *J. Comput. Appl. Math.* **123** 423–46

[7] Chan T F 1982 An improved algorithm for computing the singular value decomposition *ACM Trans. Math. Softw.* **8** 72–83

[8] Chan T F and Hansen P C 1992 Some applications of the rank revealing QR factorization *SIAM J. Sci. Stat. Comput.* **13** 727–41

[9] Cheng H, Gimbutas Z, Martinsson P G and Rokhlin V 2005 On the compression of low rank matrices *SIAM J. Sci. Comput.* **26** 1389–404

[10] Colton D and Kress R 1998 *Inverse Acoustic and Electromagnetic Scattering Theory* 2nd edn (Berlin: Springer)

[11] Demmel J W 1997 *Applied Numerical Linear Algebra* (Philadephia, PA: SIAM)

[12] Drineas P, Kannan R and Mahoney M W 2006 Fast Monte Carlo algorithms for matrices: III. Computing a compressed approximate matrix decomposition *SIAM J. Comput.* **36** 184–206

[13] Engl H W, Hanke M and Neubauer A 1996 *Regularization of Inverse Problems* (Dordrecht: Kluwer)

[14] Frieze A, Kannan R and Vempala S 2004 Fast Monte-Carlo algorithms for finding low-rank approximations *J. ACM* **51** 1025–41

[15] Golub G H, Heath M and Wahba G 1979 Generalized cross-validation as a method for choosing a good ridge parameter *Technometrics* **21** 215–23

[16] Golub G H and Meurant G 2010 *Matrices, Moments and Quadrature with Applications* (Princeton, NJ: Princeton University Press)

[17] Golub G H and Van Loan C F 2013 *Matrix Computations* 4th edn (Baltimore, MD: John Hopkins University Press)

[18] Golub G H and von Matt U 1997 Tikhonov regularization for large scale problems *Workshop on Scientific Computing* ed G H Golub, S H Lui, F Luk and R J Plemmons (New York: Springer) pp 3–26

[19] Gu M and Eisenstat S C 1996 Efficient algorithms for computing a strong rank-revealing QR factorization *SIAM J. Sci. Comput.* **17** 848–69

[20] Halko N, Martinsson P and Tropp J 2011 Finding structures with randomness: probabilistic algorithms for constructing approximate matrix decompositions *SIAM Rev.* **53** 217–88

[21] Hamarik U, Palm R and Raus T 2009 On minimization strategies for choice of the regularization parameter in ill-posed problems *Numer. Funct. Anal. Opt.* **30** 924–50

[22] Hamarik U, Palm R and Raus T 2011 Comparison of parameter choices in regularization algorithms in case of different information about noise level *Calcolo* **48** 47–59

[23] Hanke M 1996 Limitations of the L-curve method in ill-posed problems *BIT Numer. Math.* **36** 287–301

[24] Hanke M and Raus T 1996 A general heuristic for choosing the regularization parameter in ill-posed problems *SIAM J. Sci. Comput.* **17** 956–72

[25] Hansen P C 1987 The truncated SVD as a method for regularization *BIT Numer. Math.* **27** 543–53

[26] Hansen P C 1992 Analysis of discrete ill-posed problems by means of the L-curve *SIAM Rev.* **34** 561–80

[27] Hansen P C 1994 Regularization tools: a Matlab package for analysis and solution of discrete ill-posed problems *Numer. Algorithms* **6** 1–35

[28] Hansen P C 1998 *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion* (Philadelphia, PA: SIAM)

[29] Hansen P C 2010 *Discrete Inverse Problems, Insight and Applications* (Philadelphia, PA: SIAM)

[30] Hansen P C 2008 Regularization Tools, Version 4.1 for Matlab 7.3 www2.imm.dtu.dk/∼pch/Regutools

[31] Hansen P C and O'Leary D P 1993 The use of the L-curve in the regularization of discrete ill-posed problems *SIAM J. Sci. Comput.* **14** 1487–503

[32] Kirk D B and Hwu W W 2010 *Programming Massively Parallel Processors: A Hands-On Approach* (San Mateo, CA: Morgan Kaufmann)

[33] Lagomasino G L, Reichel L and Wunderlich L 2008 Matrices, moments, and rational quadrature *Linear Algebra Appl.* **429** 2540–54

[34] Leonov A S and Yagola A G 1997 The L-curve method always produces an unremovable systematic error *Vestn. Mosk. Univ.* 3 **6** 17–9 (in Russian)

[35] Lepskii O V 1990 On a problem of adaptive estimation in Gaussian white noise *Theor. Probab. Appl.* **35** 454–66

[36] Liberty E, Woolfe F, Martinsson P G, Rokhlin V and Tygert M 2007 Randomized algorithms for the low-rank approximation of matrices *Proc. Natl Acad. Sci.* **104** 20167–72

[37] Mahoney M W and Drineas P 2009 CUR matrix decompositions for improved data analysis *Proc. Natl Acad. Sci.* **106** 697–702

[38] Mathe P 2006 The Lepskii principle revisited *Inverse Problems* **22** L11–L15

[39] Morozov V A 1966 On the solution of functional equations by the method of regularization *Sov. Math.—Dokl.* **7** 414–7 (www.ams.org/mathscinet-getitem?mr=0208819)

[40] O'Leary D P and Simmons J A 1981 A bibiagonalization-regularization procedure for large-scale regularization of ill-posed problems *SIAM J. Sci. Stat. Comput.* **2** 474–89

[41] Phillips D L 1962 A technique for the numerical solution of certain integral equations of the first kind *J. Assoc. Comput. Mach.* **9** 84–97

[42] Rokhlin V, Szlam A and Tygert M 2009 A randomized algorithm for principal component analysis *SIAM J. Matrix Anal. Appl.* **31** 1100–24

[43] Sanders J and Kandrot E 2010 *CUDA by example: An Introduction to General-Purpose GPU Programming* (Reading, MA: Addison-Wesley)

[44] Stewart G W 1999 The QLP approximation to the singular value decomposition *SIAM J. Sci. Comput.* **20** 1336–48

[45] Tautenhahn U and Hamarik U 1999 The use of monotonicity for choosing the regularization parameter in ill-posed problems *Inverse Problems* **15** 1487–505

[46] Tikhonov A and Arsenin V 1977 *Solutions of Ill-posed Problems* (New York: Wiley)

[47] Tikhonov A and Glasko V 1965 Use of the regularization method in nonlinear problems *USSR Comput. Math. Math. Phys.* **5** 93–107

[48] Titarenko V N and Yagola A G 2000 Application of the GCV method for well-posed and ill-posed problems *Vestn. Mosk. Univ.* 3 **4** 15–8 (in Russian)

[49] Vogel C R 1996 Non-convergence of the L-curve regularization parameter selection method *Inverse Problems* **12** 535–47

[50] Woolfe F, Liberty E, Rokhlin V and Tygert M 2008 A fast randomized algorithm for the approximation of matrices *Appl. Comput. Harmon. Anal.* **25** 335–66