A fast and stable algorithm for downdating the singular value decomposition

Article *in* Computers & Mathematics with Applications · November 2014

DOI: 10.1016/j.camwa.2014.09.008

Shengguo Li
National University of Defense Technology
32 PUBLICATIONS **273** CITATIONS

Xiang-Ke Liao
National University of Defense Technology
211 PUBLICATIONS **4,615** CITATIONS

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

A fast and stable algorithm for downdating the singular value decomposition

Jieyuan Zhang [a,c], Shengguo Li [b,*], Lizhi Cheng [a,c], Xiangke Liao [b], Guangquan Cheng [d]

[a] College of Science, National University of Defense Technology, Changsha 410073, China

[b] College of Computer Science, National University of Defense Technology, Changsha 410073, China

[c] The State Key Laboratory for High Performance Computation, National University of Defense Technology, Changsha 410073, China

[d] Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, China

A R T I C L E I N F O

Article history:
Received 31 January 2014
Received in revised form 18 August 2014
Accepted 13 September 2014
Available online xxxx

Keywords:
Downdating SVD
HSS matrices
Cauchy-like matrices
Numerical stability

A B S T R A C T

In this paper, we modify a classical downdating SVD algorithm and reduce its complexity significantly. We use a structured low-rank approximation algorithm to compute an hierarchically semiseparable (HSS) matrix approximation to the eigenvector matrix of a diagonal matrix plus rank-one modification. The complexity of our downdating algorithm is analyzed. We further show that the structured low-rank approximation algorithm is backward stable. Numerous experiments have been done to show the efficiency of our algorithm. For some matrices with large dimensions, our algorithm can be much faster than that using plain matrix–matrix multiplication routine in Intel MKL in both sequential and parallel cases.

1. Introduction

Let A be an $M \times N$ real matrix, $M > N$, and assume that its SVD is defined as

$$A = U\Sigma V^T = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} \Sigma_1 \\ 0 \end{pmatrix} V^T = U_1 \Sigma_1 V^T, \tag{1}$$

where $U = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \in R^{M\times M}$ and $V \in R^{N\times N}$ are orthogonal matrices, $U_1 \in R^{M\times N}$, and $\Sigma_1 \in R^{N\times N}$ is a diagonal matrix, whose diagonal entries are the singular values of A. In this paper, we propose a fast algorithm to compute the SVD of A' satisfying

$$A = \begin{pmatrix} A' \\ a^T \end{pmatrix}, \tag{2}$$

where a^T is the bottom row of A. Note that the case of deleting a column of A can be considered similarly.

* Corresponding author. Tel.: +86 13687340094.
E-mail address: nudtlsg@nudt.edu.cn (S. Li).

http://dx.doi.org/10.1016/j.camwa.2014.09.008
0898-1221/© 2014 Elsevier Ltd. All rights reserved.

Computing the SVD of A' is well-known as the *downdating SVD* problem [1], which has been widely applied in signal processing, image analysis and least square problems (see [2–5] for more details). When using Latent Semantic Indexing (LSI) for information retrieval, some outdated terms and/or documents can be removed from the term-by-document matrix [6], which is also reduced to the downdating SVD problem.

Most algorithms for downdating SVD (see [1,3]) are expensive, costing $O(N^3)$ flops. For simplicity of analysis, we assume that M and N are in the same order, $M = O(N)$. In this paper, we show that the complexity of downdating SVD problem (2) can be reduced to $O(N^2r)$ flops,

$$A' = U'\Sigma'V'^T = \begin{pmatrix} U_1' & U_2' \end{pmatrix} \begin{pmatrix} \Sigma_1' \\ 0 \end{pmatrix} V'^T = U_1'\Sigma_1'V'^T, \tag{3}$$

where r is a moderate constant (see Section 3.2 for details), and $U_1' \in R^{(M-1)\times N}$. The technique used in this paper is similar to that in [7,8]. The main observation is that some intermediate matrices appeared in the algorithm in [1] are Cauchy-like matrices and have off-diagonal low-rank property, see Eqs. (7) and (8). We use hierarchically semiseparable (HSS) matrices [9–11] to approximate these matrices, and then use fast HSS matrix–matrix multiplication algorithm [12] to update the singular vectors. Note that the updating SVD problem in [3,13] can be accelerated similarly.

To fully take advantage of these two properties, a structured low-rank approximation method is proposed for Cauchy-like matrices in [7,8], which is called *SRRSC* (structured rank-revealing Schur complement factorization). SRRSC only works on several vectors, therefore its memory cost is linear $O(N)$. The complexity of using SRRSC to construct HSS approximations is analyzed in Section 4.2, which is shown to be $O(N^2r)$, where N is the dimension of the matrix and r is a modest constant as above. In Section 4.3, the rounding error analysis of SRRSC is included, which shows that SRRSC is backward stable.

Numerous experiments have been done in Section 5 to show the efficiency of our algorithm. Since both the HSS construction algorithm and HSS matrix–matrix multiplication algorithm are good for parallel, we further implement the proposed downdating SVD algorithm by using *OpenMP* on the shared memory multicore platform. For matrices with large dimensions, the proposed downdating algorithm can be 3x–5x faster than that using plain matrix–matrix multiplication routine in Intel MKL in the serial and parallel cases.

2. Preliminaries

The HSS matrix is an important kind of rank-structured matrices. It explores the low-rank property of off-diagonal blocks, which was first discussed in [10,11]. In this section, we briefly summarize some key concepts of the HSS structure following the notation in [14,15,9].

2.1. Tree structure

Suppose that H is a general $N \times N$ matrix, and $I = \{1, 2, \dots, N\}$ is the set of its row and column indices. Let $\mathscr{T}$ be a binary tree with $2n-1$ nodes labeled as $i = 1, 2, \dots, 2n-1$, in which the root node is numbered $2n-1$ and the number of leaf nodes is n. Here $\mathscr{T}$ is assumed to be in post order, which means that the ordering of non-leaf node i satisfies $i_1 < i_2 < i$, where i_1 and i_2 are its left and right child respectively. Each node i is associated with a subset of indices $t_i \subseteq I$. Thus t_i has the following properties:

1. *Each non-leaf node satisfies* $t_{i_1} \cup t_{i_2} = t_i$ *and* $t_{i_1} \cap t_{i_2} = \varnothing$, *and the leaf nodes satisfy* $\cup_{\text{all leaves } i} \; t_i = I$.
2. $t_i = I$, *when node i is the root of* $\mathscr{T}$.

Following the notation in [14], let $H_{t_it_j}$ denote the submatrix of H corresponding to row index set t_i and column index set t_j.

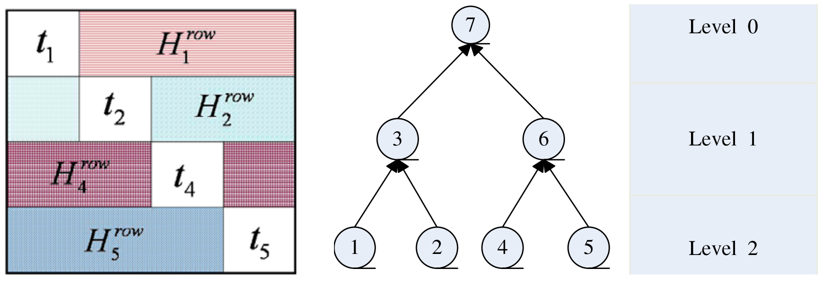
2.2. Generators

If matrix H of order N is represented in HSS form and its corresponding HSS tree is $\mathscr{T}$, there exist matrices D_i, U_i, V_i, R_i, W_i and B_i (called *HSS generators*) associated with each node i of $\mathscr{T}$ satisfying

$$\begin{aligned} &D_{2n-1} = H, \qquad U_{2n-1} = \varnothing, \qquad V_{2n-1} = \varnothing, \\ &D_i = H_{t_it_i} = \begin{pmatrix} D_{i_1} & U_{i_1}B_{i_1}V_{i_2}^T \\ U_{i_2}B_{i_2}V_{i_1}^T & D_{i_2} \end{pmatrix}, \\ &U_i = \begin{pmatrix} U_{i_1}R_{i_1} \\ U_{i_2}R_{i_2} \end{pmatrix}, \qquad V_i = \begin{pmatrix} V_{i_1}W_{i_1} \\ V_{i_2}W_{i_2} \end{pmatrix}. \end{aligned} \tag{4}$$

For each node i of $\mathscr{T}$, its corresponding *HSS block row and column* are defined as follows:

$$H_i^{row} = H_{t_i \times (I \setminus t_i)} \quad \text{and} \quad H_i^{column} = H_{(I\setminus t_i)\times t_i}, \tag{5}$$



(a) Row partition. (b) HSS tree.

Fig. 1. A 4×4 HSS matrix H.

which are also called *HSS block* for simplicity. We call the maximum (numerical) rank of all HSS blocks the *HSS rank*, which is denoted by r. If HSS rank is small compared with the matrix size, matrix H is said to have *off-diagonal low-rank property*.

Assume H is a block 4×4 HSS matrix, then it would have the following form

$$H = \begin{pmatrix} \begin{pmatrix} D_1 & U_1B_1V_2^T \\ U_2B_2V_1^T & D_2 \end{pmatrix} & \begin{pmatrix} U_1R_1 \\ U_2R_2 \end{pmatrix} B_3 \begin{pmatrix} W_4^TV_4^T & W_5^TV_5^T \end{pmatrix} \\ \begin{pmatrix} U_4R_4 \\ U_5R_5 \end{pmatrix} B_6 \begin{pmatrix} W_1^TV_1^T & W_2^TV_2^T \end{pmatrix} & \begin{pmatrix} D_4 & U_4B_4V_5^T \\ U_5B_5V_4^T & D_5 \end{pmatrix} \end{pmatrix}, \tag{6}$$

and its corresponding HSS tree is shown in Fig. 1.

The HSS construction can be implemented by following the HSS tree level by level from bottom to up. For the nodes at the same level, all HSS block rows or columns can be compressed simultaneously. We refer to [10–12] for more details of HSS construction algorithm. At each level, after all HSS rows and columns are compressed, the generators B_i are stored and the nonzero off-diagonal blocks are merged together. After a matrix is represented in HSS form, there exist fast algorithms for multiplying it with a vector in $O(Nr)$ flops [12] (in parallel). For simplicity, the HSS matrix multiplication algorithm is not included here, see [12] for details.

3. Structured downdating SVD

We only show how to accelerate the computation of V'. The left singular vector matrix U' can be accelerated similarly, see [1]. By (1)–(3), we have

$$V\Sigma_1^2V^T = A^TA = A'^TA' + aa^T = V'\Sigma_1'^2V'^T + aa^T. \tag{7}$$

Let $z = V^Ta$, and (7) can be written as:

$$V'\Sigma_1'^2V'^T = V(\Sigma_1^2 - zz^T)V^T. \tag{8}$$

If the eigendecomposition of $\Sigma_1^2 - zz^T$ is $H\Omega^2H^T$, then V' and Σ_1' can be easily computed, $V' = VH$ and $\Sigma_1' = \Omega$. Note that the main cost lies in the computation of singular vector matrix V' which requires one matrix–matrix multiplication, while the singular values can be computed by solving a secular equation in $O(N^2)$ flops. See the following section for more details.

3.1. *The eigendecomposition of $\Sigma_1^2 - zz^T$*

As is well-known that the eigenvalues and eigenvectors of $\Sigma_1^2 - zz^T$ can be computed by the following lemmas, where $\Sigma_1 = diag(\sigma_1, \ldots, \sigma_N)$, with $\sigma_1 \geq \cdots \geq \sigma_N \geq 0$, and $z = (z_1, \ldots, z_N)^T$. Owing to (8), the eigenvalues of $\Sigma_1^2 - zz^T$ are guaranteed to be nonnegative.

Lemma 1 (*Bunch and Nielsen [2]*). *Assume that $H\Omega^2H^T$ is the eigendecomposition of matrix $\Sigma_1^2 - zz^T$, where $\Sigma_1 = diag(\sigma_1, \ldots, \sigma_N)$, $\Omega = diag(\omega_1, \ldots, \omega_N)$, $H = (h_1, \ldots, h_N)$, and $z^T\Sigma_1^{-2}z \leq 1$. Then the eigenvalues $\{\omega_i^2\}_{i=1}^N$ satisfy the interlacing property*

$$\sigma_1 > \omega_1 > \sigma_2 > \cdots > \sigma_N > \omega_N \geq 0, \tag{9}$$

Table 1
The ranks of different off-diagonal blocks of H.

k	1	2	3	4	5	6	7	8	9	10	11	12
dim($\times$**100**)	1	2	3	4	5	6	7	8	9	10	11	12
rank	16	18	21	22	23	22	24	23	23	24	23	25

and the following equation

$$f(\omega) \equiv -1 + \sum_{j=1}^{N} \frac{z_j^2}{\sigma_j^2 - \omega^2} = 0. \tag{10}$$

The eigenvectors are given by

$$h_i = \left(\frac{z_1}{\sigma_1^2 - \omega_i^2}, \dots, \frac{z_N}{\sigma_N^2 - \omega_i^2} \right)^T \Big/ \sqrt{\sum_{j=1}^{N} \frac{z_j^2}{(\sigma_j^2 - \omega_i^2)^2}}. \tag{11}$$

Note that the singular vectors cannot be computed directly from Eq. (11) since they may loss orthogonality [1]. One way to solve this problem is to recompute the vector z, denoted by $\hat{z}$, by using the following lemma.

Lemma 2 (*See [16,1]*). *Given a diagonal matrix $\Sigma_1 = diag(\sigma_1, \dots, \sigma_N)$ and a set of numbers $\{\hat{\omega}_i^2\}_{i=1}^N$ satisfying the interlacing property*

$$\sigma_1 > \hat{\omega}_1 > \sigma_2 > \cdots > \sigma_N > \hat{\omega}_N \geq 0, \tag{12}$$

there exists a vector $\hat{z} = (\hat{z}_1, \dots, \hat{z}_N)^T$ such that the eigenvalues of $\Sigma_1^2 - \hat{z}\hat{z}^T$ are $\{\hat{\omega}_i^2\}_{i=1}^N$. The components of $\hat{z}$ are given by

$$|\hat{z}_i| = \sqrt{(\sigma_i^2 - \hat{\omega}_N^2) \prod_{j=1}^{i-1} \frac{\hat{\omega}_j^2 - \sigma_i^2}{\sigma_j^2 - \sigma_i^2} \prod_{j=i}^{N-1} \frac{\hat{\omega}_j^2 - \sigma_i^2}{\sigma_{j+1}^2 - \sigma_i^2}}, \quad 1 \leq i \leq N, \tag{13}$$

where the sign of $\hat{z}_i$ can be chosen arbitrarily.

3.2. Low-rank property

Recall from Lemma 1, we find that H is a Cauchy-like matrix,

$$H = \begin{pmatrix} \frac{\alpha_1 z_1}{\sigma_1^2 - \omega_1^2} & \cdots & \frac{\alpha_N z_1}{\sigma_1^2 - \omega_N^2} \\ \frac{\alpha_1 z_2}{\sigma_2^2 - \omega_1^2} & \vdots & \frac{\alpha_N z_2}{\sigma_2^2 - \omega_N^2} \\ \vdots & \vdots & \vdots \\ \frac{\alpha_1 z_N}{\sigma_N^2 - \omega_1^2} & \cdots & \frac{\alpha_N z_N}{\sigma_N^2 - \omega_N^2} \end{pmatrix}, \tag{14}$$

where $\alpha_i = 1/\sqrt{\sum_{j=1}^N \frac{z_j^2}{(\sigma_j^2 - \omega_i^2)^2}}$. Furthermore, it has off-diagonal low-rank structures, which is illustrated in the following example.

Example 1. Assume H is a 2500×2500 Cauchy-like matrix like $(\frac{u_i v_j}{d_i - w_j})_{ij}$, where u, v, d and w are 2500×1 random vectors, and the entries of d, w are interlacing: $d_1 > w_1 > d_2 > \cdots > d_{2500} > w_{2500}$. The numerical ranks of the off-diagonal blocks $H_k = H(m \times k + 1 : end, 1 : m \times k)$ with $m = 100$, are shown in Table 1, and the column dimensions are displayed in the second row of Table 1 (the column dimension is smaller than the row dimension). Here we use Matlab routine and $\varepsilon = 1.0e^{-13}$ to compute the numerical ranks. The results in Table 1 show that the off-diagonal ranks can be smaller than the sizes of the corresponding submatrices.

Remark 1. If the vectors d and w are interlacing, matrix H in Example 1 usually has off-diagonal low-rank property. But it is not theoretically true. In [7], we show that the off-diagonal ranks depend on the distribution of the vectors d and w (probably u and v too). For example, when d and w are clustered, the off-diagonal rank may nearly equal the size of the corresponding submatrix.

Algorithm 1 SRRSC factorization

Let G be an $m \times K$ Cauchy-like matrix which satisfies (16) for $k = 0$, its generators are d, w, u and v. Let $u^{(0)} = u$, and assume G has numerical rank $r < \min(m, K)$.

```
for $k = 1, \ldots, \min(m, K)$
(1) use complete pivoting strategy to choose the largest entry $|G^{(k-1)}(\ell, h)|$, and exchange the first and $\ell$-th entries
    of $u(k : m)$ and $d(k : m)$, and exchange the first and $h$-th entries of $v(k : K)$ and $w(k : K)$;
(2) if $|G^{(k-1)}(1, 1)|$ is negligible, then
        $r = k - 1$;
        return;
(3) else
    $u^{(k)}(k+1:m) = u^{(k)}(k+1:m) \cdot \frac{d^{(k)}(k+1:m) - d^{(k)}(k)}{d^{(k)}(k+1:m) - w^{(k)}(k)}$;
    $v^{(k)}(k+1:K) = v^{(k)}(k+1:K) \cdot \frac{w^{(k)}(k+1:K) - w^{(k)}(k)}{w^{(k)}(k+1:K) - d^{(k)}(k)}$;
    $y^{(k)}(1:k-1) = y^{(k)}(1:k-1) \cdot \frac{w^{(k)}(k) - d^{(k)}(1:k-1)}{d^{(k)}(k) - d^{(k)}(1:k-1)}$;
    $y^{(k)}(k) = \frac{d^{(k)}(k) - w^{(k)}(k)}{u^{(0)}(k)}$;
    for $j = 1 : k-1$
        $y^{(k)}(k) = y^{(k)}(k) \cdot \frac{w^{(k)}(j) - d^{(k)}(k)}{d^{(k)}(j) - d^{(k)}(k)}$;
    end for
end for
```

3.3. Structured low-rank approximation

Some HSS construction methods have been introduced in [9–11]. Based on the special properties of Cauchy-like matrices, we introduce an effective structured low-rank approximation method for Cauchy-like matrices in this section, which is called SRRSC in [7]. The complexity of SRRSC is analyzed in Section 4.1.

Definition 3 (*See [17,18]*)**.** Let G be an $m \times K$ matrix, $m \le K$, the following factorization of G is called kth Schur complement factorization of G (k is an integer, $1 \le k < m$),

$$G = \begin{pmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{pmatrix} = \begin{pmatrix} I_k & \\ Z^{(k)} & I_{m-k} \end{pmatrix} \begin{pmatrix} G_{11} & G_{12} \\ & G^{(k)} \end{pmatrix}, \tag{15}$$

and $G^{(k)}$ is the kth Schur complement.

The Schur complements of a Cauchy-like matrix are well known to have displacement structures [19,20], and therefore we can only work on its generators. The generators of these Schur complements can be computed recursively.

Theorem 4. *The kth Schur complement $G^{(k)}$ satisfies*

$$D_k G^{(k)} - G^{(k)} W_k = u^{(k)}(k+1:m) \cdot v^{(k)T}(k+1:K), \tag{16}$$

with $D_k = diag(d_{k+1}, \ldots, d_m)$ and $W_k = diag(w_{k+1}, \ldots, w_K)$. Then

$$u^{(k)}(k+j) = u^{(k-1)}(k+j) \cdot \frac{d_{k+j} - d_k}{d_{k+j} - w_k}, \tag{17}$$

$$v^{(k)}(k+l) = v^{(k-1)}(k+l) \cdot \frac{w_{k+l} - w_k}{w_{k+l} - d_k}, \tag{18}$$

with $1 \le j \le m-k$, $1 \le l \le K-k$, $G^{(0)} = G$, $u^{(0)} = u$ and $v^{(0)} = v$.

For $1 \le j \le m-k$ and $1 \le l \le K-k$, the generators of $G^{(k)}$ are

$$u^{(k)}(k+j) = u^{(0)}(k+j) \cdot \prod_{i=1}^{k} \frac{d_{k+j} - d_i}{d_{k+j} - w_i}, \tag{19}$$

$$v^{(k)}(k+l) = v^{(0)}(k+l) \cdot \prod_{i=1}^{k} \frac{w_{k+l} - w_i}{w_{k+l} - d_i}. \tag{20}$$

Furthermore, the matrix $Z^{(k)}$ in (15) has displacement structure too, whose generators can also be computed recursively.

Theorem 5. *The generators of $Z^{(k)}$ satisfy the following displacement equation*

$$D_2^{(k)} Z^{(k)} - Z^{(k)} D_1^{(k)} = u^{(k)}(k+1:m) \cdot y^{(k)}(1:k), \tag{21}$$

where $D_1^{(k)} = diag(d_1, \ldots, d_k)$, $D_2^{(k)} = diag(d_{k+1}, \ldots, d_m)$, $u^{(k)}(k+1:m)$ are computed recursively by (17), and

$$y^{(k)}(i) = \begin{cases} y^{(k-1)}(i) \cdot \dfrac{w_k - d_i}{d_k - d_i} & \text{if } 1 \le i \le k-1, \\ \displaystyle\prod_{j=1}^{k-1} \frac{d_k - w_j}{d_k - d_j} \cdot \frac{d_k - w_k}{u^{(0)}(i)} & \text{if } i = k. \end{cases} \tag{22}$$

Note that if matrix (15) which corresponds to an HSS block is rank-deficient, at some step the entries of $G^{(k)}$ would become negligible. Thus by ignoring $G^{(k)}$, we can get a low-rank approximation to G,

$$G \approx \begin{pmatrix} I \\ Z^{(k)} \end{pmatrix} \begin{pmatrix} G_{11} & G_{12} \end{pmatrix}. \tag{23}$$

To have better numerical stability, we incorporate pivoting into the factorization (15) since pivoting does not destroy the Cauchy-like structure. The complete pivoting is best for stability but it may deteriorate the speed. For some more efficient pivoting strategies, we refer to [7,20,18]. The whole structure of SRRSC is illustrated in Algorithm 1.

Algorithm 2 HSS construction algorithm for Cauchy-like matrices

Suppose that HSS tree $\mathcal{T}$ is a full binary tree and it has $L+1$ levels and the leaf nodes are at level L. Assume that H is a Cauchy-like matrix with generators $u, v, d, w \in R^N$, $H(i,j) = \frac{u_i v_j}{d_i - w_j}$.

```
for node $i$ at level $L$ (leaf nodes)
    1. compute the main diagonal block $D_i$ of $H$ via its generators;
    2. identify the generators of $i$th HSS block row $H_i^{row}$ and column $H_i^{col}$, and compute their low-rank
       approximations by using SRRSC,
           $H_i^{row} \approx U_i \tilde{H}_i^{row}, \quad H_i^{col} \approx \tilde{H}_i^{col} V_i^T.$
    3. store the generators of Cauchy-like sub-matrices $\tilde{H}_i^{row}$ and $\tilde{H}_i^{col}$, and store the generators of HSS matrix $U_i$
       and $V_i$.
end for
for $\ell = L-1 : -1, 1$ (parent nodes)
  for node $i$ at level $\ell$
    1. merge the generators of $\tilde{H}_{i_1}^{row}$ and $\tilde{H}_{i_2}^{row}$, and compute a low-rank approximation of $H_i^{row}$,
           $H_i^{row} = \begin{pmatrix} \hat{H}_{i_1}^{row} \\ \hat{H}_{i_2}^{row} \end{pmatrix} \approx \begin{pmatrix} R_{i_1} \\ R_{i_2} \end{pmatrix} \tilde{H}_i^{row},$
       where $\tilde{H}_i^{row}$ is a certain submatrix of $H_i^{row}$.
    2. merge the generators of $\tilde{H}_{i_1}^{col}$ and $\tilde{H}_{i_2}^{col}$, and compute a low-rank approximation of $H_i^{col}$,
           $H_i^{col} = \begin{pmatrix} \hat{H}_{i_1}^{col} & \hat{H}_{i_2}^{col} \end{pmatrix} \approx \tilde{H}_i^{col} \begin{pmatrix} W_{i_1}^T & W_{i_2}^T \end{pmatrix},$
       and store the generators of $\tilde{H}_i^{row}$ and $\tilde{H}_i^{col}$, and the generators for the HSS matrix.
    3. compute submatrices $B_{i_1}$ and $B_{i_2}$ via their generators.
  end for
end for
```

4. Complexity and stability analysis

4.1. Complexity of SRRSC

We assume that G is an $m \times K$ matrix with entries $G_{ij} = \frac{u_i v_j}{d_i - w_j}$ and its numerical rank is r. For simplicity, we assume that all pivots are chosen by complete pivoting strategy. Similar to Gaussian elimination with complete pivoting (GECP), SRRSC has two main steps: *complete pivoting* and *updating the parameters*.

Complete pivoting Before factorization, the complete pivoting should be done first, which can be performed by searching column by column. At the kth step of SRRSC, the complexity of pivoting would be

$$\gamma_s(m-k+1)(K-k+1), \tag{24}$$

where $\gamma_s = 2$ is the number of operations of computing $\frac{u_\ell}{d_\ell - w_j}$.

Updating the parameters The SRRSC is computed by operating on three parameters, u, v and y. At step k, u is updated by (19), v is by (20) and y is by (22). The cost is

$$\gamma_u(m-k) + \gamma_v(K-k) + \gamma_y(2(k-1)+1) \approx \gamma(m+K), \tag{25}$$

where $\gamma_u = \gamma_v = \gamma_y = \gamma = 4$ is the number of operations updating one entry of u (v, y), for example there are four operations in (17). For simplicity we let $\gamma_s \equiv \gamma = 4$.

Total cost of SRRSC

$$\begin{aligned} C_{srrsc} &= \gamma \sum_{k=1}^{r}[(m+K)+(m-k+1)(K-k+1)] \\ &= \gamma\left[(mK+2m+2K+1)r + \sum_{k}^{r}(k^2-(m+K+2)k)\right] \\ &\approx \gamma\left[mKr + \frac{r^3}{3} - \frac{1}{2}(m+K)r^2\right] \\ &\le \gamma\left[mKr - \frac{1}{2}Kr^2\right], \end{aligned} \tag{26}$$

where we ignored the lower order terms and used the relation that $r \le m$.

Remark 2. 1. Note that matrix G corresponds to an HSS block of H in (15) and the following subsection, which is assumed to be off-diagonal low rank.
2. The constants, γ_u, γ_v and γ_y, may be a little different in the downdating SVD problem since, for example, $d_i - w_j$ must be computed by

$$d_i - w_j = \begin{cases} (d_i - d_j) - \gamma_j & \text{if } i \le j \\ (d_i - d_{j+1}) + \mu_j & \text{if } i > j, \end{cases} \tag{27}$$

where $\gamma_i = w_i - d_i$ and $\mu_i = d_{i+1} - w_i$, which can be returned by calling LAPACK routine `dlaed4`, for $i = 1, \dots, N$.

4.2. *Complexity of constructing HSS*

The analysis is similar to those in [14,15] but with some differences. All the previous analysis are assuming that matrix H has been formed explicitly. Our new HSS construction algorithm works on the generators u, v, d and w and does not form H explicitly. Our analysis is based on the following assumptions and notation.

Assume that the HSS tree is a full binary tree and that it has L levels, $(0, 1, \dots, L-1)$. There are $n = 2^L$ leaf nodes, and the dimension of matrix H is N. All the HSS blocks have rank r. Each leaf node at the bottom level has m rows and $r = O(m)$. We follow the HSS tree level by level and start from the leaf nodes. Let LN denote the set of all leaf nodes. Let N_i be the number of columns in the ith HSS block row.

Complexity for leaf nodes Each leaf node has four generators D, U, V and B. Since the compressions of rows and columns are similar, we only analyze the row compressions in detail.

Generator D: D_i can be computed directly. The cost for leaf nodes is

$$C_D^f \approx \gamma \cdot m \cdot m \cdot n \approx \gamma N r, \tag{28}$$

where $\gamma = 4$ and we used $r = O(m)$.

Generator U: Since $U = P\begin{bmatrix} I \\ Z \end{bmatrix}$ where P is a permutation matrix, the cost includes that of computing SRRSC and that of computing entries of Z.

$$\begin{aligned} C_{srrsc}^f &= \sum_{i \in LN} \gamma r\left[mN_i - \frac{r}{2}N_i\right] = \gamma n r\left(m - \frac{r}{2}\right)(N-m) \\ &\approx \frac{\gamma}{2}N^2 r - \frac{1}{2}N r^2, \end{aligned} \tag{29}$$

where we used $r = O(m)$.

Since the (i,j)th entry of Z is $Z_{ij} = \frac{u_i y_j}{d_i - d_j}$, after obtaining the generators of Z the cost of computing Z is

$$\sum_{i \in LN} \gamma (m-r) r \approx \gamma N r.$$

Thus, the cost of computing U for all leaf nodes is

$$C_U^f = \frac{\gamma}{2} N^2 r + O(Nr - Nr^2).$$

Generator V: It needs the same operations as U, $C_V^f = C_U^f$.

Generator B: Each B_i at the bottom level is a $r \times r$ matrix. The cost is

$$C_B^f = \sum_{i \in LN} \gamma \cdot r \cdot r \approx \gamma N r,$$

where we used $r = O(m)$ and there are n leaf nodes.

Therefore, the total cost for leaf nodes is

$$C^f = C_D^f + C_U^f + C_V^f + C_B^f = \gamma N^2 r + O(Nr - Nr^2).$$

Complexity for parent nodes For a parent node i, let $\widetilde{U}_i = \begin{bmatrix} R_{i_1} \\ R_{i_2} \end{bmatrix}$, $\widetilde{V}_i = \begin{bmatrix} W_{i_1} \\ W_{i_2} \end{bmatrix}$ and B_i. Then, similar to the analysis of leaf nodes, we can compute the cost of parent nodes level by level. Note that there are 2^ℓ nodes at level ℓ, and the row dimension of each HSS block row is $2r$. No operation is needed for the root node. The total cost for all parent nodes is

$$C^p = C_{\widetilde{U}}^p + C_{\widetilde{V}}^p + C_B^p = \frac{3}{2} \gamma N^2 r + O(Nr^2).$$

Therefore, the total complexity of constructing HSS matrix is about

$$C = C^f + C^p = \frac{5}{2} \gamma N^2 r + O(Nr^2) = 10 N^2 r + O(Nr^2).$$

Remark 3. The HSS construction algorithms based on SRRSC need $O(N^2 r)$ operations. The main advantage of using SRRSC is that it only needs $O(N)$ memory which can potentially work on very large matrices. It is easy to see that the complexity of our downdating algorithm, shown in Algorithm 3, is also $O(N^2 r)$ flops.

4.3. Numerical stability of SRRSC

An error analysis for the LU factorization of Cauchy-like matrices appeared in [21]. Since the Cauchy-like matrices considered in this paper have displacement rank one, using the same techniques as in [21], it is easy to show that the backward stability of SRRSC is related to that of the classical Gaussian elimination, see also [22]. Therefore, the algorithm is backward stable, which is also illustrated as follows.

Theorem 6. *Assume that no overflows and underflows were encountered during the computation and ε is a unit roundoff. Then the computed factors $\hat{Z}^{(k)}$, $\hat{G}^{(k)}$ satisfies $\hat{Z}^{(k)} = Z^{(k)} + \Delta Z^{(k)}$ and $\hat{G}^{(k)} = G^{(k)} + \Delta G^{(k)}$, where*

$$|\Delta Z^{(k)}| \le \gamma_{8k+1} |Z^{(k)}|, \qquad |\Delta G^{(k)}| \le \gamma_{8k+3} |G^{(k)}| \quad \left(\gamma_k = \frac{k\varepsilon}{1 - k\varepsilon} \right). \tag{30}$$

Proof. Assume that permutation has been done beforehand. By Theorems 4 and 5,

$$u^{(k)}(k+l) = u^{(0)}(k+l) \cdot \prod_{j=1}^{k} \frac{d_{k+l} - d_j}{d_{k+l} - w_j}, \tag{31}$$

$$y^{(k)}(i) = \prod_{j=1}^{i-1} \frac{d_i - w_j}{d_i - d_j} \cdot \prod_{j=i+1}^{k} \frac{d_i - w_j}{d_i - d_j} \cdot \frac{d_i - w_i}{u^{(0)}(i)}. \tag{32}$$

A straightforward error analysis implies

$$\hat{Z}^{(k)}(i,j) = fl\left(\frac{u^{(k)}(k+i) y^{(k)}(j)}{d_{k+i} - d_j} \right) = Z^{(k)}(i,j) \delta_{i,j}^{(k)}, \tag{33}$$

where

$$\frac{(1-\varepsilon)^{6k+1}}{(1+\varepsilon)^{2k}} \le \delta_{i,j}^{(k)} \le \frac{(1+\varepsilon)^{6k+1}}{(1-\varepsilon)^{2k}}. \tag{34}$$

The similar arguments can be applied to $\hat{G}^{(k)}$ and get its rounding error bound. □

Corollary 7. *The computed factorization* $\hat{G} = \begin{pmatrix} I_k & \\ \hat{Z}^{(k)} & I_{m-k} \end{pmatrix} \begin{pmatrix} \hat{G}_{11} & \hat{G}_{12} \\ & \hat{G}^{(k)} \end{pmatrix}$ *satisfies* $\hat{G} = G + \Delta E^{(k)}$, *if we define* $L = \begin{pmatrix} I_k & \\ Z^{(k)} & I_{m-k} \end{pmatrix}$ *and* $F = \begin{pmatrix} G_{11} & G_{12} \\ & G^{(k)} \end{pmatrix}$, *then*

$$|\Delta E^{(k)}| \le [(16k+4)\varepsilon + O(\varepsilon^2)]|L||F|. \tag{35}$$

Proof. From the standard error analysis, the computed $\hat{G} = \hat{L}\hat{F}$ satisfies

$$\hat{L}\hat{F} = (L + \Delta L)(F + \Delta F) = LF + \Delta E^{(k)} \tag{36}$$

where

$$|\Delta L| \le \gamma_{8k+1}|L|, \qquad |\Delta F| \le \gamma_{8k+3}|F|. \tag{37}$$

The bound (35) can be derived from (30) and (37). □

Algorithm 3 Downdating the SVD

```
Given an $M \times N$ ($M > N$) matrix $A$ and its SVD, $A = U\Sigma V^T$, assume the bottom vector $a^T$ is to be deleted.
  1. Extract the submatrix $\Sigma_1$ from $\Sigma$ and compute $z = V^T a$;
  2. Compute the eigendecomposition of $\Sigma_1^2 - zz^T$ by solving secular equations and store the generators of matrix $H$;
  3. Use SRRSC to construct an HSS approximation to $H$
         $H \approx \hat{H}$,
     where $\hat{H}$ is an HSS matrix;
  4. Compute the right singular vectors of $A'$ by using fast HSS matrix–matrix multiplications
         $V' = V\hat{H}$.
```

5. Numerical results

We use our algorithm to solve the following specific problem,

Given V, Σ_1 and a, compute V' and Σ_1'.

The algorithm that we used is summarized in Algorithm 3.

The following examples are tested on a laptop with 4 GB memory and Intel(R) Core(TM) i7-2640M CPU. For compilation we used Intel fortran compiler (`ifort`) and the optimization flag `-O2`, and then linked the codes to the optimized BLAS and LAPACK library, Intel MKL (composer_xe_2013_sp1).

Example 2. Let A be an $(N+1) \times N$ Gaussian random matrix with zero mean and unit variance, and assume that its SVD is $A = U\Sigma V^T$. The singular values and right singular vector matrix V' of A' are desired, where A' is the matrix after removing the bottom row of A. By choosing different dimensions N, the orthogonality of right singular vectors computed by using HSS matrix techniques and plain matrix–matrix multiplications are included in Table 2, where the baseline is from Intel MKL. The orthogonality is measured by $\|(V')^T V' - I\|_2$, shown in the row denoted by *Orth*.

The comparisons of running times in seconds are shown in the third row denoted by *Time* in Table 2. For matrices with dimensions around 8000, our algorithm can be `5.4x` faster than that using the plain matrix–matrix multiplication routine in Intel MKL. The larger the matrix is, the bigger speedup our algorithm has. Computing matrix V' requires $3N^2$ memory when using plain matrix–matrix multiplications. While, using HSS techniques only requires $N^2 + O(N)$ memory. The row denoted by *Memory* in Table 2 shows the ratios of memory cost when using Algorithm 3 to compute V' over $3N^2$.

Since the HSS construction and HSS matrix multiplication algorithms are naturally parallelizable, we implement these two algorithms by using OpenMP and compare our algorithms with the multi-threaded Intel MKL. Since the CPU of the laptop that we used has four cores, we let `OMP_NUM_THREADS=4`. The results are shown in Table 3. In the parallel case, when the dimensions of matrices are around 8000, our algorithm can be 3.0x faster than that using multi-threaded Intel MKL.

Table 2
The comparison of orthogonality and speedups with MKL, k denotes one thousand.

Dim		$1k \times 1k$	$3k \times 3k$	$4k \times 4k$	$5k \times 5k$	$8k \times 8k$
Orth.	HSS	1.7e−14	3.5e−14	4.9e−14	5.4e−14	7.4e−14
	MKL	1.7e−14	3.7e−14	5.1e−14	5.9e−14	8.0e−14
Time	HSS	7.8e−02	0.7e+00	1.2e+00	1.9e+00	5.0e+00
	MKL	9.4e−02	1.7e+00	3.7e+00	6.9e+01	2.7e+01
Speedup		1.2x	2.5x	3.0x	3.6x	5.4x
Memory		0.65	0.53	0.46	0.45	0.41

Table 3
The results for parallel implementation, k denotes one thousand.

Dim		$1k \times 1k$	$3k \times 3k$	$4k \times 4k$	$5k \times 5k$	$8k \times 8k$
Orth.	HSS	1.6e−14	3.5e−14	4.9e−14	5.5e−14	8.0e−14
	MKL	1.7e−14	3.7e−14	5.1e−14	5.9e−14	7.5e−14
Time	HSS	5.9e−02	7.5e−01	8.9e−01	1.4e+00	3.6e+00
	MKL	6.3e−02	1.3e+00	1.7e+00	3.1e+00	1.1e+01
Speedup		1.1x	1.7x	1.9x	2.2x	3.0x

6. Conclusion

In this paper, we propose an accelerated downdating SVD algorithm by using HSS matrix techniques, which was first introduced in [1]. The improved downdating algorithm is summarized in Algorithm 3, which can reduce the complexity of floating-point operations and storage cost significantly, especially for matrices with large dimensions. We use SRRSC [8,7] to construct HSS approximations. In Section 4, the complexities of SRRSC and HSS construction algorithm based on SRRSC are analyzed in detail. Furthermore, the rounding error analysis of SRRSC is also included, which is shown to be backward stable. Comparisons with sequential and multi-threaded Intel MKL show that using HSS techniques to compute V' can be 3x–5x faster than using plain matrix–matrix multiplications implemented in Intel MKL for large matrices.

Acknowledgments

This work is partially supported by National Natural Science Foundation of China (No. 11401580), NSF of Hunan Province in China (No. 13JJ2001 and 13JJ4011) and Doctoral Foundation of Ministry of Education of China (No. 20114307120021).

References

[1] M. Gu, S.C. Eisenstat, Downdating the singular value decomposition, SIAM J. Matrix Anal. Appl. 16 (1995) 793–810.
[2] J.R. Bunch, C.P. Nielsen, Updating the singular value decomposition, Numer. Math. 31 (1978) 111–129.
[3] M. Moonen, P. Van Dooren, J. Vandewalle, A singular value decomposition updating algorithm for subspace tracking, SIAM J. Matrix Anal. Appl. 13 (1992) 1015–1038.
[4] J. Barlow, P.A. Yoon, H. Zha, An algorithm and a stability theory for downdating the ULV decomposition, BIT 36 (1996) 14–40.
[5] G. Stewart, Determining rank in the presense of error, in: Linear Algebra for Large Scale and Real-time Applications, vol. 232, 1993, pp. 275–291.
[6] D. Witter, M. Berry, Downdating the latent semantic indexing model for conceptual information retrieval, Comput. J. 41 (1998) 589–601.
[7] S. Li, M. Gu, L. Cheng, X. Chi, M. Sun, An accelerated divide-and-conquer algorithm for the bidiagonal singular value problem, SIAM J. Matrix Anal. Appl. 35 (2014) 1038–1057.
[8] M. Gu, A numerically stable superfast Toeplitz solver, in: Presentations in Berkeley Matrix Computations and Scientific Computing Seminar and SIAM LA09, 2009, http://math.berkeley.edu/~mgu/Seminar/Fall2009/ToeplitzSeminarTalk.pdf.
[9] J. Xia, S. Chandrasekaran, M. Gu, X. Li, Fast algorithms for hierarchically semiseparable matrices, Numer. Linear Algebra Appl. 17 (2010) 953–976.
[10] S. Chandrasekaran, M. Gu, T. Pals, Fast and Stable Algorithms for Hierarchically Semiseparable Representations, Tech. Report, University of California, Berkeley, CA, 2004.
[11] S. Chandrasekaran, M. Gu, T. Pals, A fast ULV decomposition solver for hierarchical semiseparable representations, SIAM J. Matrix Anal. Appl. 28 (2006) 603–622.
[12] W. Lyons, Fast algorithms with applications to PDEs (Ph.D. thesis), University of California, Santa Barbara, 2005.
[13] M. Gu, S.C. Eisenstat, A Stable and Fast Algorithm for Updating the Singular Value Decomposition, Tech. Report, RR-966, Yale University, 1994.
[14] J. Xia, M. Gu, Robust approximate Cholesky factorization of rank-structured symmetric positive definite matrices, SIAM J. Matrix Anal. Appl. 31 (2010) 2899–2920.
[15] S. Li, M. Gu, C.J. Wu, J. Xia, New efficient and robust HSS Cholesky factorization of symmetric positive definite matrices, SIAM J. Matrix Anal. Appl. 33 (2012) 886–904.
[16] K. Löwner, Über monotone Matrixfunktionen, Math. Z. 38 (1934) 176–216.
[17] R.W. Cottle, Manifestations of the Schur complement, Linear Algebra Appl. 8 (1974) 189–211.
[18] C. Pan, On the existence and computation of rank revealing LU factorizations, Linear Algebra Appl. 316 (2000) 199–222.
[19] I. Gohberg, T. Kailath, V. Olshevsky, Fast Gaussian elimination with partial pivoting for matrices with displacement structure, Math. Comp. 64 (1995) 1557–1576.
[20] M. Gu, Stable and efficient algorithms for structured systems of linear equations, SIAM J. Matrix Anal. Appl. 19 (1998) 279–306.
[21] D. Sweet, R. Brent, Error analysis of a fast partial pivoting method for structured matrices, in: Proceedings of the SPIE-1995, vol. 2563, pp. 266–280.
[22] T. Boros, T. Kailath, V. Olshevsky, Pivoting and backward stability of fast algorithms for solving Cauchy linear equations, Linear Algebra Appl. 343 (2002) 63–99.