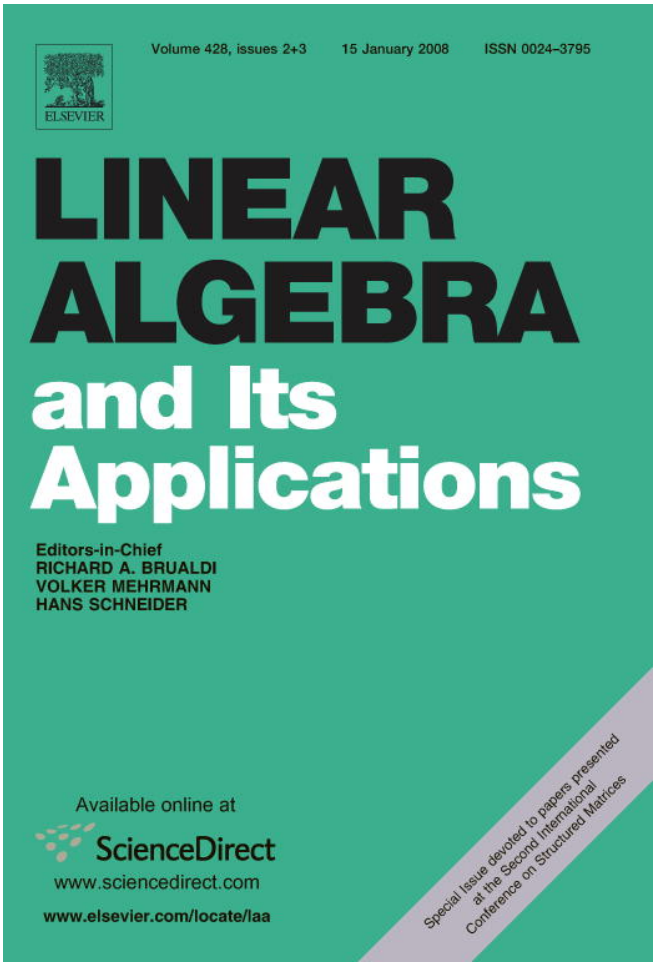

Available online at www.sciencedirect.com
ScienceDirect
Linear Algebra and its Applications 428 (2008) 550–563
LINEAR ALGEBRA AND ITS APPLICATIONS
www.elsevier.com/locate/laa

A fast symmetric SVD algorithm for square Hankel matrices

Wei Xu, Sanzheng Qiao *

Department of Computing and Software, McMaster University, Hamilton, Ont., Canada L8S 4K1

Received 31 August 2006; accepted 25 May 2007
Available online 3 June 2007
Submitted by van Barel

Abstract

This paper presents an $O(n^2 \log n)$ algorithm for computing the symmetric singular value decomposition of square Hankel matrices of order n, in contrast with existing $O(n^3)$ SVD algorithms. The algorithm consists of two stages: first, a complex square Hankel matrix is reduced to a complex symmetric tridiagonal matrix using the block Lanczos method in $O(n^2 \log n)$ flops; second, the singular values and singular vectors of the symmetric tridiagonal matrix resulted from the first stage are computed in $O(n^2)$ flops. The singular vector matrix is given in the form of a product of three or two unitary matrices. The performance of our algorithm is demonstrated by comparing it with the SVD subroutines in Matlab and LAPACK.

AMS classification: 15A18; 65F20; 65F25; 65F50

Keywords: Fast SVD; Hankel matrix; Toeplitz matrix; Symmetric SVD; Takagi factorization

1. Introduction

Complex symmetric matrices arise from many applications such as nuclear magnetic resonance [2], independent component analysis [4], and so on. Complex Hankel matrices are special complex symmetric matrices arising from signal processing [11] and statistics [1]. An $n \times n$ Hankel matrix is of the form:

* Corresponding author.
E-mail addresses: xuw5@mcmaster.ca (W. Xu), qiao@mcmaster.ca (S. Qiao).

0024-3795/$ - see front matter © 2007 Published by Elsevier Inc.
doi:10.1016/j.laa.2007.05.027

$$H = \begin{bmatrix} h_1 & h_2 & h_3 & \cdots & h_n \\ h_2 & h_3 & h_4 & \cdots & h_{n+1} \\ h_3 & h_4 & h_5 & \cdots & h_{n+2} \\ \vdots & \vdots & \vdots & \iddots & \vdots \\ h_n & h_{n+1} & h_{n+2} & \cdots & h_{2n-1} \end{bmatrix},$$

which is determined by its last row and first column, total of $2n-1$ entries, in contrast with $(n^2+n)/2$ entries for a general symmetric matrix.

Due to the special structure of the Hankel matrix, its matrix–vector multiplication can be performed in $O(n \log n)$ flops using the fast Fourier transform (FFT) [12,14]. Also, for any symmetric matrix, there exists a special form of the singular value decomposition called the symmetric singular value decomposition (SSVD) or Takagi factorization [10]. In particular, let H be a Hankel matrix of order n, then there exists the decomposition

$$H = V \Sigma V^{\mathrm{T}},$$

where $V = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n]$ is unitary, $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n), \sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_n \geqslant 0$, is the diagonal singular value matrix. The columns $\mathbf{v}_i$ of V are called the Takagi vectors and σ_i the Takagi values or singular values of H. Analogous to the singular vector matrix, we call V the Takagi vector matrix. Note that the Takagi vectors are the left singular vectors, whereas the left singular vectors need not be the Takagi vectors. For example, consider the 2-by-2 complex symmetric matrix

$$T = \begin{bmatrix} 1 & \mathrm{i} \\ \mathrm{i} & -1 \end{bmatrix}, \quad \text{where } \mathrm{i} = \sqrt{-1}.$$

Its SSVD is

$$T = Q \Sigma Q^{\mathrm{T}} = \begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{1}{2} + \frac{1}{2}\mathrm{i} \\ -\frac{\sqrt{2}}{2}\mathrm{i} & \frac{1}{2} - \frac{1}{2}\mathrm{i} \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2}\mathrm{i} \\ \frac{1}{2} + \frac{1}{2}\mathrm{i} & \frac{1}{2} - \frac{1}{2}\mathrm{i} \end{bmatrix}.$$

Since the Takagi vector matrix Q is unitary, the Takagi vectors, the columns of Q, are left singular vectors of T. However, it can be verified that the left singular vectors given by the eigenvectors in the eigenvalue decomposition

$$T T^{\mathrm{H}} = \begin{bmatrix} \frac{\sqrt{2}}{4} + \frac{\sqrt{6}}{4}\mathrm{i} & -\frac{\sqrt{2}}{2} \\ -\frac{\sqrt{6}}{4} + \frac{\sqrt{2}}{4}\mathrm{i} & \frac{\sqrt{2}}{2}\mathrm{i} \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{4} - \frac{\sqrt{6}}{4}\mathrm{i} & -\frac{\sqrt{6}}{4} - \frac{\sqrt{2}}{4}\mathrm{i} \\ -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2}\mathrm{i} \end{bmatrix}$$

are not the Takagi vectors of T.

The computation of the SVD of a general symmetric matrix of order n requires $O(n^3)$ flops. In this paper, we present an $O(n^2 \log n)$ SSVD algorithm for square Hankel matrices of order n. Since a square Toeplitz matrix can be converted into a Hankel matrix by reversing its columns or rows [12], our fast SSVD algorithm straightforwardly leads to a fast SVD algorithm for square Toeplitz matrices. Our algorithm consists of two stages. In the first stage, a complex square Hankel matrix is reduced to complex symmetric tridiagonal form using symmetric unitary transformations. The

block Lanczos method is used to exploit the Hankel structure. Since the dominant computation of the Lanczos tridiagonalization is matrix–vector multiplication and Hankel matrix–vector multiplication can be performed in $O(n \log n)$ flops using the FFT [12,14], the cost of this stage is $O(n^2 \log n)$. The second stage computes the SSVD of the complex symmetric tridiagonal matrix resulted from the previous stage. This stage consists of three parts: First, the singular values are obtained by the implicit QR method [3,14] in $O(n^2)$ flops; then the singular vectors are computed by the twisted factorization method in $O(n^2)$ flops; finally, the singular vectors are converted into the Takagi vectors in no more than $O(n^2)$ flops. The total cost of this stage is $O(n^2)$. Combining these two stages, we have an $O(n^2 \log n)$ algorithm for computing the SSVD of square Hankel matrices, where the unitary singular vector matrix or the Takagi vector matrix is in the form of a product of two or three unitary matrices. To our best knowledge, this is the first fast SVD algorithm for Hankel matrices.

The rest of this paper is organized as follows. Section 2 describes the block Lanczos tridiagonalization method integrated with the fast Hankel matrix–vector multiplication algorithm. Then, the twisted factorization method for computing the singular vectors is presented in Section 3. The issue of computing the singular vectors associated with a multiple singular value is discussed in Section 4. The conversion of the singular vectors into the Takagi vectors is shown in Section 5. Finally, we compare our SSVD algorithm with the SVD subroutines in Matlab and LAPACK in Section 6 to show that our method is $O(n^2 \log n)$ as opposed to $O(n^3)$ general SVD routines in Matlab and LAPACK.

2. Tridiagonalization

The Lanczos tridiagonalization method is suitable for structured matrices since its major computation is matrix–vector multiplication, where the structure of the matrix can be exploited. In this section, we describe a block Lanczos tridiagonalization method for complex square Hankel matrices. A Hankel matrix is first transformed into block tridiagonal form, which is then reduced to tridiagonal. The method carries out the tridiagonalization in two steps because block algorithms are rich in BLAS 3 operations and render high performance [8].

A complex Hankel matrix of order n, $n = pb$ for some integers p and b, can be reduced to a block tridiagonal matrix by unitary transformations:

$$Q^{\mathrm{H}} H \overline{Q} = J \equiv \begin{bmatrix} M_1 & B_1^{\mathrm{T}} & & 0 \\ B_1 & \ddots & \ddots & \\ & \ddots & \ddots & B_{p-1}^{\mathrm{T}} \\ 0 & & B_{p-1} & M_p \end{bmatrix}, \tag{1}$$

where $Q = [Q_1, Q_2, \ldots, Q_p]$, $Q_j \in \mathbb{C}^{n \times b}$, is unitary, $\overline{Q}$ denotes the complex conjugate of Q, $M_i \in \mathbb{C}^{b \times b}$ is symmetric, and $B_j \in \mathbb{C}^{b \times b}$ upper triangular. Rewriting (1) as $H\overline{Q} = QJ$ and comparing the jth block columns on both sides of the equation, we have

$$H\overline{Q}_j = Q_{j-1} B_{j-1}^{\mathrm{T}} + Q_j M_j + Q_{j+1} B_j, \quad Q_0 B_0 = Q_{p+1} B_p = 0$$

for $j = 1, \ldots, p$, which leads to the block Lanczos outer iteration:

$$Q_{j+1} B_j = H\overline{Q}_j - Q_j M_j - Q_{j-1} B_{j-1}^{\mathrm{T}}.$$

Since Q is unitary, $M_j = Q_j^{\mathrm{H}} H\overline{Q}_j$ for $j = 1, \ldots, p$. Let $R_j = H\overline{Q}_j - Q_j M_j - Q_{j-1} B_{j-1}^{\mathrm{T}}$, then Q_{j+1} and B_j can be obtained from the QR factorization of R_j. Thus, starting with an n-by-b Q_1 with orthonormal columns, we can compute Q and J in (1).

Algorithm 1 (*Block tridiagonalization*). Given an n-by-b starting matrix Q_1 with orthonormal columns and a subroutine for Hankel matrix–matrix multiplication $Y = HX$ for any X, where H is a complex Hankel matrix of order n. This algorithm computes the unitary Q and the blocks M_i and B_i in the block tridiagonal complex symmetric matrix J in (1).

$p = n/b$;
for $j = 1 : p - 1$
  $Y = H\overline{Q}_j$;
  $M_j = Q_j^{\mathrm{H}} Y$;
  $R_j = Y - Q_j M_j - Q_{j-1} B_{j-1}^{\mathrm{T}}$; ($Q_0 = 0$, $B_0 = 0$)
  $Q_{j+1} B_j = R_j$; (QR factorization of R_j)
end
$M_p = Q_p^{\mathrm{H}} H\overline{Q}_p$;

The choice of the block size b is architecture dependent. Typically, $b = 8, 16, 32$. The main computational cost of the above algorithm is the matrix–matrix multiplication $H\overline{Q}_j$. Using the FFT to exploit the Hankel structure of H, the multiplication can be performed in $\mathrm{O}(bn \log n)$ flops [12]. Thus the total cost of the above algorithm is $\mathrm{O}(n^2 \log n)$.

Next, the block tridiagonal J in (1) is tridiagonalized. Since J is symmetric and banded, it can be efficiently tridiagonalized by the algorithm proposed by Schwartz [17] or the classical Lanczos method briefly described as follows. For details, see [15].

The block tridiagonal J can be tridiagonalized by unitary transformations:

$$P^{\mathrm{H}} J\overline{P} = T \equiv \begin{bmatrix} \alpha_1 & \beta_1 & & 0 \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ 0 & & \beta_{n-1} & \alpha_n \end{bmatrix}, \tag{2}$$

where $P = [\mathbf{p}_1, \ldots, \mathbf{p}_n]$ is unitary. Rewriting (2) as $J\overline{P} = PT$ and comparing the jth columns of its both sides, we have

$$J\bar{\mathbf{p}}_j = \beta_{j-1}\mathbf{p}_{j-1} + \alpha_j \mathbf{p}_j + \beta_j \mathbf{p}_{j+1}, \quad \beta_0 \mathbf{p}_0 = 0,$$

which leads to a Lanczos three-term recursion:

$$\beta_j \mathbf{p}_{j+1} = J\bar{\mathbf{p}}_j - \alpha_j \mathbf{p}_j - \beta_{j-1}\mathbf{p}_{j-1}.$$

Since P is unitary, $\alpha_j = \mathbf{p}_j^{\mathrm{H}} A\bar{\mathbf{p}}_j$. Let $\mathbf{r}_j = J\bar{\mathbf{p}}_j - \alpha_j \mathbf{p}_j - \beta_{j-1}\mathbf{p}_{j-1}$, then $\beta_j = \pm\|\mathbf{r}_j\|_2$ and $\mathbf{p}_{j+1} = \mathbf{r}_j/\beta_j$ if $\mathbf{r}_j \neq 0$.

It is well-known that orthogonalization is necessary for practical Lanczos methods. The block tridiagonalization Algorithm 1 is incorporated with the block partial orthogonalization with componentwise orthogonality detection [16] and the tridiagonalization is incorporated with the modified partial orthogonalization [15]. The orthogonalization schemes introduce little extra work. The total cost of the tridiagonalization with orthogonalization is still $\mathrm{O}(n^2 \log n)$.

Alternatively, the tridiagonalization can be carried out in one step using the classical Lanczos method. However, our experiments have shown that the block algorithm renders high performance by exploiting the memory hierarchy [16].

3. Twisted factorization

Now that a Hankel matrix is tridiagonalized, it remains to compute the SSVD of the complex symmetric tridiagonal T in (2) in no more than $O(n^2 \log n)$. It is shown in [3,14] that the implicit QR method computes the singular values in $O(n^2)$ flops, however, it requires $O(n^3)$ for the singular vectors. In this section, we present a twisted factorization method for computing the left singular vectors of T in $O(n^2)$, if the computed singular values are available. Thus, combining the implicit QR and the twisted factorization methods, we can compute the singular values and the left singular vectors of T in $O(n^2)$ flops. There are $O(n^2)$ methods for bidiagonal SVD [9,18]. Since we consider the SSVD of the complex symmetric tridiagonal T, we adapt the twisted method for symmetric tridiagonal eigendecomposition [5,6]. The difference is that we compute the eigenvectors of a Hermitian pentadiagonal matrix TT^{H}. Moreover, our method is implicit in that the pentadiagonal matrix TT^{H} is not explicitly formed.

We assume that the complex symmetric tridiagonal matrix T is irreducible, that is there are no zero entries on its subdiagonal. Otherwise, when there are zero entries on the subdiagonal, we can deflate the matrix T into several small irreducible matrices. It is shown that the multiplicity of any singular value of an irreducible T is at most two [20]. In this section, we consider the case of simple singular values. The issue of multiple singular values is addressed in the next section.

The left singular vectors of T are the eigenvectors of the Hermitian positive semidefinite pentadiagonal $S = TT^{\mathrm{H}}$. As we know, if we have good approximations of eigenvalues, the shifted inverse power iteration is an effective way of computing the associated eigenvectors. We use the singular values of T computed by the implicit QR method as good approximations of the eigenvalues of S. Let λ be an approximation of an eigenvalue λ_i of S, the shifted inverse power method involves solving linear systems with the coefficient matrix $S - \lambda I$, which is ill-conditioned. In the following, we show how to formulate the problem so that we can solve the linear systems efficiently and stably.

We first compute the LDLH decomposition:

$$S = \widehat{L}\widehat{D}\widehat{L}^{\mathrm{H}}, \tag{3}$$

where

$$\widehat{D} = \mathrm{diag}(d_1^*, \ldots, d_n^*), \quad d_i^* \geqslant 0, \quad \text{and} \quad \widehat{L} = \begin{bmatrix} 1 & & & & \\ l_1^* & \ddots & & 0 & \\ m_1^* & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ 0 & & m_{n-2}^* & l_{n-1}^* & 1 \end{bmatrix},$$

which can be obtained, for example, from the QR decomposition of T^{H}. Since T is tridiagonal, this decomposition can be computed in $O(n)$ operations. Then we decompose the shifted $S - \lambda I$ in two ways:

$$S - \lambda I = L D_L L^{\mathrm{H}} = U D_U U^{\mathrm{H}}, \tag{4}$$

where

$$L = \begin{bmatrix} 1 & & & & \\ l_1 & \ddots & & 0 & \\ m_1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ 0 & & m_{n-2} & l_{n-1} & 1 \end{bmatrix}, \quad D_L = \operatorname{diag}(\nu_1, \ldots, \nu_n), \quad \nu_i \in \mathbb{R}$$

and

$$U = \begin{bmatrix} 1 & u_1 & v_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & v_{n-2} \\ & & & \ddots & u_{n-1} \\ & & & & 1 \end{bmatrix}, \quad D_U = \operatorname{diag}(\mu_1, \ldots, \mu_n), \quad \mu_i \in \mathbb{R}.$$

The above two decompositions can be obtained from the LDLH decomposition (3) by comparing the entries. Specifically, comparing the $(i+2, i+1)$-entries in the both sides of the equation $\widehat{L}\widehat{D}\widehat{L}^{\mathrm{H}} - \lambda I = L D_L L^{\mathrm{H}}$, we get

$$m_i^* \bar{l}_i^* d_i^* + l_{i+1}^* d_{i+1}^* = m_i \bar{l}_i \nu_i + l_{i+1} \nu_{i+1}.$$

Comparing the $(i+2, i+2)$-entries, we have

$$|m_i^*|^2 d_i^* + |l_{i+1}^*|^2 d_{i+1}^* + d_{i+2}^* - \lambda = |m_i|^2 \nu_i + |l_{i+1}|^2 \nu_{i+1} + \nu_{i+2}.$$

In summary, we have the following algorithm for the decomposition $S - \lambda I = L D_L L^{\mathrm{H}}$.

Algorithm 2. Given the LDLH decomposition $S = \widehat{L}\widehat{D}\widehat{L}^{\mathrm{H}}$ of a Hermitian pentadiagonal $S = TT^{\mathrm{H}}$, this algorithm computes the decomposition $S - \lambda I = L D_L L^{\mathrm{H}}$

$\nu_1 = d_1^* - \lambda$; % $(1,1)$-entry
$l_1 = l_1^* d_1^* / \nu_1$; % $(2,1)$-entry
$\nu_2 = |l_1^*|^2 d_1^* + d_2^* - \lambda - |l_1|^2 \nu_1$; % $(2,2)$-entry
for $i = 1 : n-2$
  $m_i = m_i^* d_i^* / \nu_i$; % $(i+2, i)$-entry
  $l_{i+1} = (m_i^* \bar{l}_i^* d_i^* + l_{i+1}^* d_{i+1}^* - m_i \bar{l}_i \nu_i)/\nu_{i+1}$; % $(i+2, i+1)$-entry
  $\nu_{i+2} = |m_i^*|^2 d_i^* + |l_{i+1}^*|^2 d_{i+1}^* + d_{i+2}^* - \lambda - |m_i|^2 \nu_i - |l_{i+1}|^2 \nu_{i+1}$;
  % $(i+2, i+2)$-entry
end

The computational cost of this decomposition is $\mathrm{O}(n)$ flops for each λ.

Similarly, we can compute the decomposition $S - \lambda I = U D_U U^{\mathrm{H}}$ in $\mathrm{O}(n)$ flops for each λ.

Now, combining the two decompositions (4), we construct the twisted factorization of the shifted matrix:

$$S - \lambda I = N_k D_k N_k^{\mathrm{H}}, \tag{5}$$

where $D_k = \operatorname{diag}(\nu_1, \ldots, \nu_{k-2}, \xi_k, \gamma_k, \mu_{k+1}, \ldots, \mu_n)$, $\xi_k, \gamma_k \in \mathbb{R}$, and

$$N_k = \begin{bmatrix} 1 & & & & & & & \\ l_1 & \ddots & & & & 0 & & \\ m_1 & \ddots & 1 & & & & & \\ & \ddots & l_{k-2} & 1 & & v_{k-1} & & \\ & & m_{k-2} & \eta_k & 1 & u_k & \ddots & \\ & & & & & 1 & \ddots & v_{n-2} \\ & & 0 & & & & \ddots & u_{n-1} \\ & & & & & & & 1 \end{bmatrix} \tag{6}$$

is a combination of the partial lower triangular L and the partial upper triangular U. Note that for each index k, $1 \leqslant k \leqslant n$, there is a corresponding twisted factorization. However, given the LDLH and UDUH decompositions (4), there are only three unknowns ξ_k, $\gamma_k \in \mathbb{R}$, and $\eta_k \in \mathbb{C}$ in a twisted factorization. The following theorem shows a computation of the three unknowns and a relation between γ_k^{-1} and a Rayleigh quotient of $(S - \lambda I)^{-1}$ when $S - \lambda I$ is invertible.

Theorem 3 [20]. *Given the LDLH and UDUH decompositions* (4) *of* $S - \lambda I$, *defining* $m_{-1} = m_0 = v_0 = l_0 = v_0 = v_{n-1} = \mu_{n+1} = 0$, *then the three unknowns in the twisted factorization* (5) *are given by*

$$\xi_k = v_{k-1} - |v_{k-1}|^2 \mu_{k+1}, \quad k = 1, \ldots, n,$$

$$\eta_k = \begin{cases} 0, & k = 1, \\ (l_{k-1} v_{k-1} - u_k \bar{v}_{k-1} \mu_{k+1})/\xi_k, & k = 2, \ldots, n-1, \\ l_{n-1}, & k = n, \end{cases}$$

$$\gamma_k = \begin{cases} \mu_k - |m_{k-2}|^2 v_{k-2} - \xi_k |\eta_k|^2, & k = 1, \ldots, n-1, \\ v_n, & k = n. \end{cases}$$

Also, when $S - \lambda I$ *is invertible*

$$\gamma_k^{-1} = \mathbf{e}_k^{\mathrm{T}} (S - \lambda I)^{-1} \mathbf{e}_k,$$

where $\mathbf{e}_k$ *is the kth unit vector.*

This theorem shows that γ_k is expected to be small when λ is a good approximation of an eigenvalue of S.

Now, we are ready to present an efficient and stable method for computing the eigenvectors of S, which are the left singular vectors of T, using the approximations of its eigenvalues, which are the squares of the computed singular values of T. Let $S = U \Lambda U^{\mathrm{H}}$ be an eigenvalue decomposition of S, where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ is the eigenvalue matrix and $U = [\mathbf{u}_1, \ldots, \mathbf{u}_n] = [u_{i,j}]$ is the unitary eigenvector matrix. Suppose that λ is an approximation of λ_i, then for each index k of the twisted factorization (5), the solution $\mathbf{z}_k$ for

$$(S - \lambda I)\mathbf{z}_k = \gamma_k \mathbf{e}_k \tag{7}$$

is the result of one inverse power iteration with the shift λ and initial vector $\gamma_k \mathbf{e}_k$. Thus, $\mathbf{z}_k$ gives an approximation of the eigenvector $\mathbf{u}_i$. How good is $\mathbf{z}_k$ as an approximation of $\mathbf{u}_i$? It is shown in [20] that the error in $\mathbf{z}_k$ as an approximation of the eigenvector associated with λ_i is bounded by the sum of $|\lambda - \lambda_i|$ and $|\gamma_k|/\|\mathbf{z}_k\|_2$. Since λ is the computed λ_i from the implicit QR method,

$|\lambda - \lambda_i|$ is expected to be small. So, in the case of simple eigenvalues, we choose the index k so that $|\gamma_k|$ is the smallest among all the twisted factorizations $S - \lambda I = N_i D_i N_i^{\mathrm{H}}$ for $i = 1, \ldots, n$.

As we know, when λ is a good approximation of an eigenvalue λ_i of S, the system (7) is ill-conditioned. However, using the twisted factorization $S - \lambda I = N_k D_k N_k^{\mathrm{H}}$ and noting that the kth column of the twisted factor N_k is $\mathbf{e}_k$, we can reformulate the system (7) into an equivalent but simpler one:

$$N_k^{\mathrm{H}} \mathbf{z}_k = \gamma_k D_k^{-1} N_k^{-1} \mathbf{e}_k = \mathbf{e}_k.$$

Solving $N_k^{\mathrm{H}} \mathbf{z}_k = \mathbf{e}_k$ is not only more efficient than solving (7) but also more stable because the ill-conditioning of (7) caused by the small $|\gamma_k|$ in D_k is avoided. From the structure of N_k^{H} (6), the entries z_j of the solution $\mathbf{z}_k$ are given by

$$\begin{array}{ll} z_k = 1, & \\ z_{k-1} = -\bar{\eta}_k, & k > 1, \\ z_j = -\bar{l}_j z_{j+1} - \bar{m}_j z_{j+2}, & j = k-2, k-3, \ldots, 1, \\ z_{k+1} = \bar{v}_{k-1}\bar{\eta}_k - \bar{u}_k, & v_0 = 0, \\ z_j = -\bar{v}_{j-2} z_{j-2} - \bar{u}_{j-1} z_{j-1}, & j = k+2, k+3, \ldots, n. \end{array} \tag{8}$$

The following algorithm summarizes the procedure of computing the eigenvector given a computed simple eigenvalue λ of S in $\mathrm{O}(n)$ flops.

Algorithm 4 (*Simple eigenvalue case*). Given the LDLH decomposition $S = \widehat{L}\widehat{D}\widehat{L}^{\mathrm{H}}$ of the Hermitian pentadiagonal matrix $S = TT^{\mathrm{H}}$ and $\lambda \approx \lambda_i$, a computed simple eigenvalue of S, this algorithm computes an approximation of the eigenvector corresponding to λ_i.

1. Compute the LDLH and UDUH decompositions (4) of $S - \lambda I$;
2. Applying Theorem 3, for $i = 1, \ldots, n$, compute the twisted factorizations $S - \lambda I = N_i D_i N_i^{\mathrm{H}}$ and find k such that $|\gamma_k| = \min_i |\gamma_i|$;
3. Solve for $\mathbf{z}_k$ in $N_k^{\mathrm{H}} \mathbf{z}_k = \mathbf{e}_k$ using (8);
4. Set $\mathbf{u}_i = \mathbf{z}_k / \|\mathbf{z}_k\|_2$.

Note that a computed eigenvalue of S can be obtained by squaring a singular value of T computed by the implicit QR method.

4. Multiple eigenvalue case

Now, we consider the multiple eigenvalue case, recalling that in our case the multiplicity is at most two. In [20], we have proved that in the case of a multiple eigenvalue $\lambda_i = \lambda_{i+1}$, there exist two indices k_1 and k_2, $k_1 \neq k_2$, so that γ_{k_1} and γ_{k_2} in the twisted factorizations indexed by k_1 and k_2 are small and the associated eigenvectors can be obtained by solving $N_{k_i}^{\mathrm{H}} \mathbf{z}_{k_i} = \mathbf{e}_{k_i}$, $i = 1, 2$. We present the following $\mathrm{O}(n)$ algorithm for computing the eigenvectors associated with a multiple eigenvalue.

Algorithm 5 (*Multiple eigenvalue case*). Given the LDLH decomposition $S = \widehat{L}\widehat{D}\widehat{L}^{\mathrm{H}}$ of the Hermitian pentadiagonal matrix $S = TT^{\mathrm{H}}$ and $\lambda \approx \lambda_i = \lambda_{i+1}$, a computed multiple eigenvalue of S, this algorithm computes approximations of the eigenvectors corresponding to the multiple eigenvalue $\lambda_i = \lambda_{i+1}$.

1. Compute the LDLH and UDUH decompositions (4) of $S - \lambda I$;
2. Applying Theorem 3, for $i = 1, \ldots, n$, compute the twisted factorizations $S - \lambda I = N_i D_i N_i^{\mathrm{H}}$ and find k_1 and k_2, $k_1 \neq k_2$, such that $|\gamma_{k_1}|$ and $|\gamma_{k_2}|$ are the smallest among $|\gamma_i|$;
3. Solve for $\mathbf{z}_{k_1}$ in $N_{k_1}^{\mathrm{H}} \mathbf{z}_{k_1} = \mathbf{e}_{k_1}$ and $\mathbf{z}_{k_2}$ in $N_{k_2}^{\mathrm{H}} \mathbf{z}_{k_2} = \mathbf{e}_{k_2}$ using (8);
4. Set $\mathbf{u}_i = \mathbf{z}_{k_1}/\|\mathbf{z}_{k_1}\|_2$ and $\mathbf{u}_{i+1} = \mathbf{z}_{k_2}/\|\mathbf{z}_{k_2}\|_2$.

The issue of the orthogonality between the computed eigenvectors corresponding to multiple or closely clustered eigenvalues are dealt with by the following two techniques. First, we have found that if the two indices k_1 and k_2 found in step 2 of Algorithm 5 are close, for example $k_2 = k_1 + 1$, the eigenvectors $\mathbf{z}_{k_1}$ and $\mathbf{z}_{k_2}$ can be almost parallel. If k_1 and k_2 are far apart, then $\mathbf{z}_{k_1}$ and $\mathbf{z}_{k_2}$ are linearly independent, which is desirable. Thus we propose the following strategy of selecting k_1 and k_2. We first find k_1 such that $|\gamma_{k_1}| = \min_i |\gamma_i|$. Then, if there is only one isolated second smallest $|\gamma_{k_2}|$, then we choose k_2 as the second index. If there is a cluster of several equally small $|\gamma_i|$ next to $|\gamma_{k_1}|$, then among them we choose an index k_2 which is far apart from k_1. Second, the two eigenvectors $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$ associated with a multiple eigenvalue $\lambda_i = \lambda_{i+1}$ are orthogonalized when they are converted into the Takagi vectors, as shown in the next section.

5. Converting into Takagi vectors

The eigenvectors $\mathbf{u}_i$ of S are the left singular vectors of T, but need not be the Takagi vectors [19]. In this section, we show how to transform the eigenvectors of S into the Takagi vector of T.

Given an eigenvector $\mathbf{u}_j$ of $S = TT^{\mathrm{H}}$, we want to convert it into a vector $\mathbf{q}_j$ satisfying $T\bar{\mathbf{q}}_j = \sigma_j \mathbf{q}_j$. We consider two cases in the conversion. In the case of simple eigenvalues, the corresponding eigenvectors of TT^{H} are uniquely defined up to a scalar factor with unit modulus, which implies that the Takagi vector $\mathbf{q}_j$ is a scalar multiple of the corresponding eigenvector $\mathbf{u}_j$. Let $T\bar{\mathbf{u}}_j = \xi\sigma_j \mathbf{u}_j$ for some scalar ξ, $|\xi| = 1$. Denoting $\xi = \mathrm{e}^{2\mathrm{i}\phi}$ and defining

$$\mathbf{q}_j \equiv \mathrm{e}^{\mathrm{i}\phi}\mathbf{u}_j,$$

we can verify that

$$T\bar{\mathbf{q}}_j = \mathrm{e}^{-\mathrm{i}\phi} T\bar{\mathbf{u}}_j = \mathrm{e}^{-\mathrm{i}\phi}\mathrm{e}^{2\mathrm{i}\phi}\sigma_j \mathbf{u}_j = \mathrm{e}^{\mathrm{i}\phi}\sigma_j \mathrm{e}^{-\mathrm{i}\phi}\mathbf{q}_j = \sigma_j \mathbf{q}_j$$

as desired. Specifically, ξ can be obtained by

$$\xi = \begin{cases} \sigma_j^{-1}\mathbf{u}_j^{\mathrm{H}} T\bar{\mathbf{u}}_j, & \sigma_j \neq 0, \\ 1, & \text{otherwise.} \end{cases} \tag{9}$$

In the case of multiple eigenvalues, $T\bar{\mathbf{u}}_j$ may not equal $\xi\sigma_j\mathbf{u}_j$. We construct

$$\mathbf{q}_j = \alpha_j(T\bar{\mathbf{u}}_j + \sigma_j \mathbf{u}_j), \tag{10}$$

where $\alpha_j = 1/\|T\bar{\mathbf{u}}_j + \sigma_j\mathbf{u}_j\|_2$ is the normalization factor. Then we have

$$T\bar{\mathbf{q}}_j = \alpha_j T(\overline{T\bar{\mathbf{u}}_j + \sigma_j\mathbf{u}_j}) = \alpha_j(T\overline{T}\mathbf{u}_j + \sigma_j T\bar{\mathbf{u}}_j) = \alpha_j(\sigma_j^2\mathbf{u}_j + \sigma_j T\bar{\mathbf{u}}_j) = \sigma_j\mathbf{q}_j.$$

In summary, if $\mathbf{u}_j$ is an eigenvector associated with a simple eigenvalue, then the corresponding Takagi vector $\mathbf{q}_j = \sqrt{\xi}\mathbf{u}_j$, where ξ is given by (9). If $\mathbf{u}_j$ is an eigenvector associated with a multiple eigenvalue, then the corresponding Takagi vector is given by (10).

Now, we discuss the orthogonality of the Takagi vectors of T converted from the eigenvectors of TT^{H}. It is obvious that the orthogonality is maintained among the Takagi vectors corresponding

to distinct singular values, because the eigenvectors corresponding to distinct eigenvalues are orthogonal. Now, assume that $\mathbf{q}_j$ and $\mathbf{q}_{j+1}$ are the Takagi vectors corresponding to a multiple singular value $\sigma_j = \sigma_{j+1}$, recalling that in our case the multiplicity is at most two. The constructions of $\mathbf{q}_j$ and $\mathbf{q}_{j+1}$ from (10) imply that the subspace spanned by $\mathbf{q}_j$ and $\mathbf{q}_{j+1}$ is the same as the one spanned by $\mathbf{u}_j$ and $\mathbf{u}_{j+1}$, since $\mathbf{q}_j$ and $\mathbf{q}_{j+1}$ are the eigenvectors associated with the eigenvalues $\sigma_j^2 = \sigma_{j+1}^2$ of S. Thus, $\mathbf{q}_j$ and $\mathbf{q}_{j+1}$ are orthogonal to $\mathbf{q}_i$, $i \neq j$, $j+1$. However, $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$ may not be orthogonal. We apply the Gram–Schmidt method to orthogonalize these vectors.

The cost of a Takagi vector conversion is $O(n)$, since its computations are tridiagonal matrix–vector multiplication and vector operations. The cost of each orthogonalization is also $O(n)$. Thus the cost of computing each Takagi vector is $O(n)$ and the total cost of computing all n Takagi vectors is $O(n^2)$.

6. Numerical experiments

We have shown an $O(n^2 \log n)$ SSVD algorithm for square Hankel matrices of order n. In this section, we demonstrate the efficiency and accuracy of our algorithm by comparing it with the subroutines in Matlab and LAPACK.

Our experiments in Matlab were carried out on a server with two 2.4 GHz Xeon CPUs, 1 GB RAM and an 80 GB disk. The experiments with our C++ implementation were carried out on a server with ten 400 MHz Ultrasparc II CPUs and 10 GB RAM running Solaris 9 and using `gcc` compiler. The random Hankel matrices tested were generated from random vectors with entries uniformly distributed between 0 and 1. The error in the singular value vector $\hat{\mathbf{s}}$ computed by our algorithm was measured by

$$\Delta_v = \|\hat{\mathbf{s}} - \mathbf{s}\|_2 / n,$$

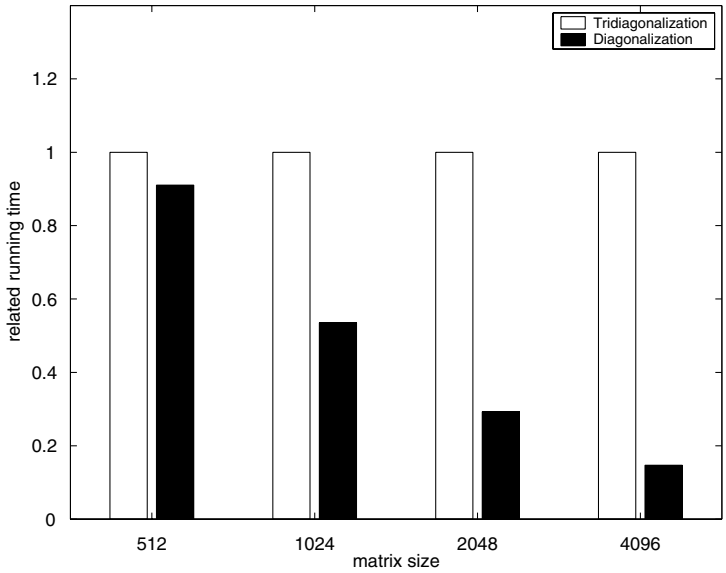

Fig. 1. The relative running times of the two stages in our fast SSVD algorithm for square Hankel matrices. The times are scaled so that the time for the first tridiagonalization stage is one.

Table 1
Running times of the two stages in our fast SSVD algorithm for square Hankel matrices and the diagonalization time relative to the tridiagonalization time

Size	Tridiagonalization	Diagonalization	Relative running time (%)
512	8.42	7.69	91
1024	53.26	30.24	57
2048	404.48	130.67	32
4096	3501.44	487.01	16

where **s** is the singular value vector computed by the Matlab SVD subroutine `svd`. The error in the computed SSVD was measured by

$$\Delta_t = \|\widehat{V}\widehat{\Sigma}\widehat{V}^{\mathrm{T}} - H\|_2/n^2,$$

where $\widehat{\Sigma} = \mathrm{diag}(\hat{\mathbf{s}})$ and $\widehat{V}$ is the computed Takagi vector matrix. The orthogonality of the computed Takagi vectors was measured by

$$\Delta_o = \|\widehat{V}\widehat{V}^{\mathrm{H}} - I\|_2/n^2.$$

Example 1. Four different sizes of Hankel matrices were generated. We ran the Matlab implementation of our fast SSVD algorithm and timed the two stages: tridiagonalization and the SSVD of a symmetric tridiagonal matrix. Fig. 1 plots the relative running times of these two stages. Table 1 lists the running times and relative running time of the two stages. We can see that the tridiagonalization consumes most of the total running time, especially for large matrices. This is expected since the tridiagonalization stage costs $O(n^2 \log n)$ while the second stage costs $O(n^2)$.

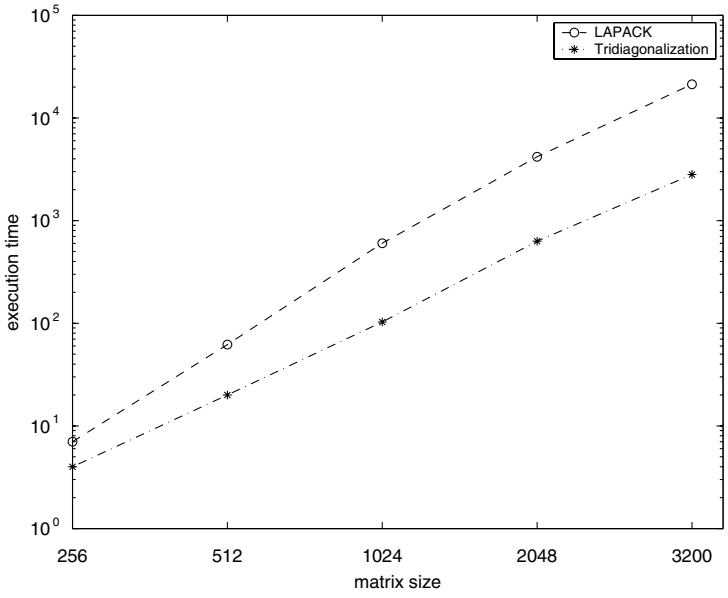

Fig. 2. Performance comparison of our fast tridiagonalization with the LAPACK bidiagonalization subroutine `zgebrd`.

Table 2
Running times of our fast tridiagonalization and the LAPACK bidiagonalization, `zgebrd`

Size	Tridiagonalization	Bidiagonalization
256	4.21	7.00
512	20.93	62.32
1024	102.89	602.31
2048	629.33	4192.30
3200	2815.20	21327.45

Table 3
Accuracy comparison of our fast SSVD algorithm and the SVD subroutine in Matlab

Matrix size	Fast SSVD			SVD in Matlab	
	Δ_o	Δ_v	Δ_t	Δ_o	Δ_t
256	3.8924E−14	3.4404E−15	1.8520E−13	3.2343E−18	5.8992E−18
512	2.1821E−14	1.6345E−14	8.3232E−14	3.2189E−18	6.7237E−18
1024	6.2341E−14	5.9797E−14	1.1890E−13	6.3231E−18	2.4174E−18
2048	3.0023E−14	1.2287E−13	4.9402E−13	1.2873E−18	1.9396E−18
4096	4.3948E−15	1.3323E−14	6.3221E−15	4.3245E−19	4.3239E−19

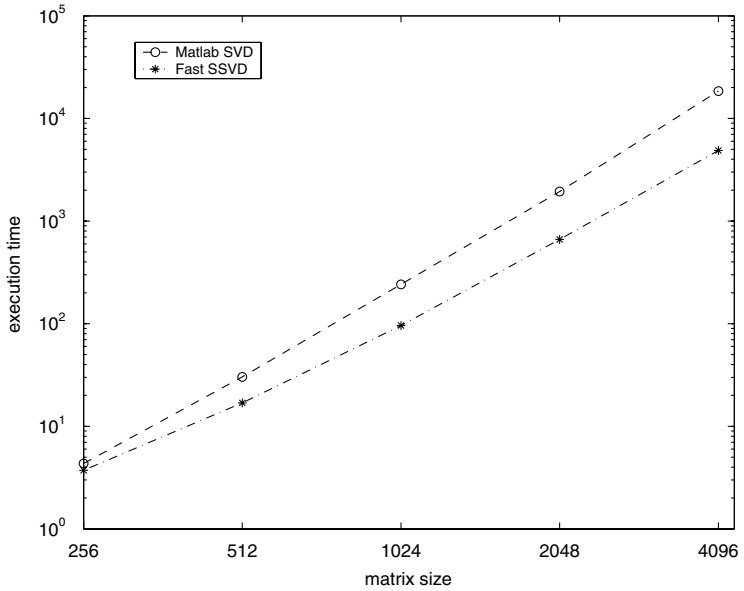

Fig. 3. Comparison of the execution time of SVD subroutine in Matlab with the proposed fast SSVD algorithm on Hankel matrices.

LAPACK has no special subroutines for the SVD of complex Hankel matrices. They are treated as general matrices. There are two steps in computing the SVD of a general matrix: bidiagonalization and diagonalization. Since in our SSVD algorithm, the first stage, tridiagonalization, dominates the total running time, in the following example, we compare the performance of our C++

Table 4
Running times of our fast SSVD algorithm and the SVD subroutine in Matlab

Size	Fast SSVD	SVD in Matlab
256	3.73	4.34
512	16.90	30.28
1024	95.86	241.65
2048	663.04	1942.10
4096	4879.20	18504.33

implementation of the tridiagonalization stage in our algorithm with that of the bidiagonalization subroutine `zgebrd` in LAPACK. In our C++ implementation, the fastest Fourier transformation in the west (FFTW) [7] is used to perform the fast Hankel matrix–vector multiplication.

Example 2. Five different sizes of complex Hankel matrices were generated. Fig. 2 plots the running times of the tridiagonalization stage and the LAPACK bidiagonalization subroutine `zgebrd`. As expected, the gap between two running times widens as the size of the matrix grows. The running times shown in Table 2 demonstrate that our tridiagonalization is approximately $O(n^2 \log n)$, whereas the general bidiagonalization is $O(n^3)$.

Example 3. In this example, we compare the performance of the Matlab implementation of our fast SSVD algorithm with the Matlab SVD subroutine `svd` on random Hankel matrices of five sizes. Table 3 compares the accuracy. Fig. 3 plots the running times of these two algorithms. The running times in Table 4 show that our SSVD is $O(n^2 \log n)$ and Matlab `svd` is $O(n^3)$.

7. Conclusion

We have proposed a novel $O(n^2 \log n)$ SSVD algorithm for square Hankel matrices of order n, in contrast with existing $O(n^3)$ algorithms. A square Hankel matrix is first transformed into symmetric tridiagonal form using the block Lanczos method integrated with orthogonalization schemes for high performance and good orthogonality. Then the SSVD of the symmetric tridiagonal matrix is computed by the implicit QR method for the singular values and the twisted factorization method for the singular vectors. Finally, the singular vectors are converted into the Takagi vectors in the SSVD. The Takagi vector matrix is given in the form of a product of unitary matrices. If the block Lanczos method is used in the tridiagonalization, it is a product of three unitary matrices; if the classical Lanczos method is used, it is a product of two matrices. The issue of the orthogonality of the Takagi vectors associated with a multiple singular value is dealt with by two techniques: an index selection strategy in the twisted factorization method; the orthogonalization process in converting the singular vectors into the Takagi vectors. Our numerical experiments have confirmed the theoretical complexity and shown good accuracy of our algorithm.

Remark. For general rectangular Hankel matrices, the SSVD does not exist. However, the framework presented in this paper can be used to construct an $O(n^2 \log n)$ SVD algorithm for Hankel and Toeplitz matrices. First, a Hankel or Toeplitz matrix is reduced to bidiagonal form in $O(n^2 \log n)$ flops, using the Lanczos bidiagonalization [8,13] integrated with the fast Hankel or Toeplitz matrix–vector multiplication using the FFT. Then, the singular values of the bidiagonal matrix are computed by the implicit QR method in $O(n^2)$ flops. Finally, applying the methods for the SVD of bidiagonal matrices [5,9,18], we obtain the singular vectors in $O(n^2)$ flops.

References

[1] N.I. Akhiezer, M. Krein, Some Questions in the Theory of Moments, AMS, 1962.
[2] A.D. Bain, Chemical exchange in NMR, Process Nucl. Magn. Resonance Spectrosc. 43 (2003) 63–103.
[3] A. Bunse-Gerstner, W.B. Gragg, Singular value decomposition of complex symmetric matrices, J. Comput. Appl. Math. 21 (1988) 41–54.
[4] P. Comon, Independent component analysis, a new concept?, Signal Process. 36 (1994) 287–314.
[5] J.S. Dhillon, B.N. Parlett, Orthogonal eigenvectors and relative gaps, SIAM J. Matrix Anal. Appl. 25 (3) (2004) 858–899.
[6] J.S. Dhillon, B.N. Parlett, Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices, Linear Algebra Appl. 387 (2004) 1–28.
[7] M. Frigo, S.G. Johnson, FFTW for version 3.1, 16 January, 2006. <http://www.fftw.org/fftw3.pdf>.
[8] G.H. Golub, C.F. Van Loan, Matrix Computations, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.
[9] B. Grosser, B. Lang, An O(n^2) algorithm for the bidiagonal SVD, Linear Algebra Appl. 358 (2003) 45–70.
[10] R.A. Horn, C.R. Johnson, Matrix Analysis, Cambridge University Press, Cambridge, 1996.
[11] A.K. Jain, Fundamentals of Digital Image Processing, Prentice-Hall, Inc., Englewood Cliffs, NY, 1989.
[12] T. Kailath, A.H. Sayed, Fast Reliable Algorithms for Matrices with Structures, SIAM, Philadelphia, 1999.
[13] R.M. Larsen, Lanczos bidiagonalization with partial reorthogonalization, Technical Report DAIMI PB-357, Department of Computer Science, Aarhus University, September 1998. <http://soi.stanford.edu/rmunk/PROPACK/index.html>.
[14] F.T. Luk, S. Qiao, A fast singular value algorithm for Hankel matrices, in: V. Olshevsky (Ed.), Fast Algorithms for Structured Matrices: Theory and Applications, Contemporary Mathematics, vol. 323, American Mathematical Society, 2003.
[15] S. Qiao, Orthogonalization techniques for the Lanczos tridiagonalization of complex symmetric matrices, in: Franklin T. Luk (Ed.), Advanced Signal Processing Algorithms, Architectures, and Implementations XIV, Proc. SPIE, vol. 5559, 2004, pp. 423–434.
[16] S. Qiao, G. Liu, W. Xu, Block Lanczos tridiagonalization of complex symmetric matrices, in: Franklin T. Luk (Ed.), Advanced Signal Processing Algorithms, Architectures, and Implementations XV, Proc. SPIE, vol. 5910, 2005, p. 591010.
[17] H.R. Schwartz, Tridiagonalization of a symmetric band matrix, Numer. Math. 12 (1968) 231–241.
[18] P.R. Willems, B. Lang, C. Vomel, Computing the bidiagonal SVD using multiple relatively robust representations, SIAM J. Matrix Anal. Appl. 28 (4) (2006) 907–926.
[19] W. Xu, S. Qiao, A divide-and-conquer method for the Takagi factorization, Technical Report No. CAS 05-01-SQ, McMaster University, Hamilton, Ont., Canada, 2005.
[20] W. Xu, S. Qiao, A twisted factorization method for symmetric SVD of a complex symmetric tridiagonal matrix, Technical Report No. CAS 06-01-SQ, McMaster University, Hamilton, Ont., Canada, 2006.