Block Lanczos Tridiagonalization of Complex Symmetric Matrices

Sanzheng Qiao, Guohong Liu, Wei Xu

Department of Computing and Software, McMaster University, Hamilton, Ontario L8S 4L7

ABSTRACT

The classic Lanczos method is an effective method for tridiagonalizing real symmetric matrices. Its block algorithm can significantly improve performance by exploiting memory hierarchies. In this paper, we present a block Lanczos method for tridiagonalizing complex symmetric matrices. Also, we propose a novel componentwise technique for detecting the loss of orthogonality to stablize the block Lanczos algorithm. Our experiments have shown our componentwise technique can reduce the number of orthogonalizations.

Keywords: Complex symmetric matrix, block Lanczos algorithm, singular value decomposition (SVD), Takagi factorization.

1. INTRODUCTION

For any complex symmetric matrix A of order n, there exist a unitary $Q \in C^{n\times n}$ and an order n nonnegative diagonal $\Sigma = \mathrm{diag}(\sigma_1, ..., \sigma_n)$, where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$, such that

$$A = Q\Sigma Q^{\mathrm{T}} \quad \text{or} \quad Q^{\mathrm{H}} A \bar{Q} = \Sigma.$$

This special form of singular value decomposition (SVD) is called Takagi factorization.[1,2]

The computation of the Takagi factorization consists of two stages: tridiagonalization and diagonalization.[3] A complex symmetric matrix is first reduced to complex symmetric and tridiagonal form. There are various tridiagonalization schemes. When A is sparse or structured, the Lanczos method is preferred. Since the Lanczos algorithm involves only matrix-vector multiplication, sparsity and structures can be exploited to develop fast tridiagonalization algorithms.[4]

The second stage, diagonalization of the complex symmetric tridiagonal matrix computed in the first stage, can be implemented by the implicit QR method[3,4] or the more efficient divide-and-conquer method.[5]

Block algorithms in which blocks of vectors instead of single vectors are used are rich in matrix-matrix (level 3 BLAS) operations. Consequently, performance is improved by exploiting memory hierarchies.

This paper presents a stable block Lanczos tridiagonalization algorithm for complex symmetric matrices. The block Lanczos tridiagonalization algorithm consists of two stages: block tridiagonalization and tridiagonalization. A complex symmetric matrix is first reduced to complex symmetric and block tridiagonal form. The second stage reduces the block tridiagonal complex symmetric matrix to complex symmetric tridiagonal form. In Sections 2 and 3, we describe the two stages of block Lanczos tridiagonalization algorithm respectively. As we know, the generic Lanczos method suffers from the loss of orthogonality of the computed Q in the presence of rounding error. Orthogonalization is necessary for practical Lanczos method. Orthogonalization techniques for the single-vector Lanczos algorithm are discussed in.[6,7] Based on a model of estimating the orthogonality of Q described in Section 4, two orthogonalization schemes for block tridiagonalization stage are proposed in Sections 5 and 6. In Section 7, we describe the orthogonalization scheme for the tridiagonalization stage. Finally, Section 8 demonstrates our numerical experiments.

Send correspondence to S. Qiao: E-Mail: qiao@mcmaster.ca

2. BLOCK TRIDIAGONALIZATION

Let A be an n-by-n complex symmetric matrix and assume $n = p \times b$. Then there exists the decomposition

$$Q^H A\bar{Q} = J = \begin{bmatrix} M_1 & B_1^T & & \dots & 0 \\ B_1 & M_2 & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & B_{p-1}^T \\ 0 & \dots & & B_{p-1} & M_p \end{bmatrix} \tag{1}$$

where

$$Q = [Q_1, Q_2, \dots, Q_p], \quad Q_i \in C^{n\times b},$$

is unitary, $M_i \in C^{b\times b}$ are symmetric, and $B_i \in C^{b\times b}$ upper triangular. Rewriting (1) as

$$A\bar{Q} = QJ \tag{2}$$

and comparing the jth block columns on both sides of (2), we have

$$A\bar{Q}_j = Q_{j-1}B_{j-1}^T + Q_j M_j + Q_{j+1}B_j, \quad Q_0B_0 = Q_{p+1}B_p = 0,$$

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

$$Q_{j+1}B_j = A\bar{Q}_j - Q_jM_j - Q_{j-1}B_{j-1}^T. \tag{3}$$

From the orthogonality of Q, we have

$$M_j = Q_j^H A\bar{Q}_j$$

for $j = 1...p$. Let $R_j = A\bar{Q}_j - Q_jM_j - Q_{j-1}B_{j-1}^T \in C^{n\times b}$, then $Q_{j+1}B_j = R_j$ is a QR factorization of R_j.

Algorithm 1 (Block Tridiagonalization). *Given an n-by-b starting matrix S of orthonormal columns and a subroutine for matrix-matrix multiplication $Y = AX$ for any X, where A is an n-by-n complex symmetric matrix. This algorithm computes the diagonal blocks of the block tridiagonal complex symmetric matrix J in (1) and a unitary Q such that $J = Q^H A\bar{Q}$*

$p = n/b$;
$Q_0 = 0$; $B_0 = 0$;
$Q_1 = S$;
for $j = 1$ to $p-1$
  $Y = A\bar{Q}_j$;
  $M_j = Q_j^H Y$;
  $R_j = Y - Q_jM_j - Q_{j-1}B_{j-1}^T$;
  $Q_{j+1}B_j = R_j$; (QR factorization of R_j)
end
$M_p = Q_p^H A\bar{Q}_p$.

3. TRIDIAGONALIZATION

We follow Berry[8] to adopt the single vector Lanczos tridiagonalization recursion for reducing the block tridiagonal complex symmetric matrix to tridiagonal form.

Let J be the p-by-p block tridiagonal complex symmetric matrix in (1) resulted from Algorithm 1. We can find a unitary $P = [\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n]$, $\mathbf{p}_i \in C^{n\times 1}$, such that

$$P^H J\bar{P} = T = \begin{bmatrix} \alpha_1 & \beta_1 & & \dots & 0 \\ \beta_1 & \alpha_2 & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \beta_{n-1} \\ 0 & \dots & & \beta_{n-1} & \alpha_n \end{bmatrix}. \tag{4}$$

Rewriting (4) as

$$J\bar{P} = PT \tag{5}$$

and comparing the jth columns on both sides of (5), we have

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

for $j = 1, ..., n$, which leads to the inner Lanczos recursion:

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

From the orthogonality of P we have

$$\alpha_j = \mathbf{p}_j^H J\bar{\mathbf{p}}_j$$

for $j = 1, ..., n$. Let $\mathbf{r}_j = J\bar{\mathbf{p}}_j - \alpha_j\mathbf{p}_j - \beta_{j-1}\mathbf{p}_{j-1} \in C^{n\times 1}$, then $\beta_j = \pm\|\mathbf{r}_j\|_2$ and $\mathbf{p}_{j+1} = \mathbf{r}_j/\beta_j$ if $\mathbf{r}_j \neq 0$. The symmetric block tridiagonal structure of J is exploited in the calculation of $J\bar{\mathbf{p}}_j$ for efficiency.

Algorithm 2 (Tridiagonalization). *Given a starting vector* $\mathbf{b}$ *and a subroutine for symmetric block tridiagonal matrix-vector multiplication* $\mathbf{y} = J\mathbf{x}$ *for any* $\mathbf{x}$*, where* J *is the complex symmetric and block tridiagonal matrix in (1), this algorithm computes the diagonals of the complex symmetric and tridiagonal matrix* T *and the unitary* P *in (4).*

$\mathbf{p}_0 = 0$; $\beta_0 = 0$;
$\mathbf{p}_1 = \mathbf{b}/\|\mathbf{b}\|_2$;
for $j = 1$ to n
  $\mathbf{y} = J\bar{\mathbf{p}}_j$;
  $\alpha_j = \mathbf{p}_j^{\mathrm{H}}\mathbf{y}$;
  $\mathbf{r}_j = \mathbf{y} - \alpha_j\mathbf{p}_j - \beta_{j-1}\mathbf{p}_{j-1}$;
  $\beta_j = \|\mathbf{r}_j\|_2$;
  if $\beta_j = 0$, quit; end
  $\mathbf{p}_{j+1} = \mathbf{r}_j/\beta_j$;
end.

4. MODEL OF ESTIMATING ORTHOGONALITY

The algorithms described in the previous sections assume exact arithmetic. In the presence of rounding error, however, the computed Q loses orthogonality. Thus, orthogonalization is necessary for practical Lanczos methods. Obviously, to detect the loss of orthogonality we must measure the orthogonality. Although Q^HQ can be a measurement for the orthogonality, it is too expensive to compute in every iteration. In this section, we present a model of estimating the orthogonality of Q computed by our block tridiagonalization algorithm. Denoting $W_{k,j} = Q_k^HQ_j$, we propose an efficient recursion of $W_{k,j}$ including rounding errors without explicitly computing $Q_k^HQ_j$.

Incorporating rounding errors into the jth iteration of (3), we have

$$Q_{j+1}B_j + F_j = A\bar{Q}_j - Q_jM_j - Q_{j-1}B_{j-1}^T, \qquad j = 1, ..., p, \tag{6}$$

where F_j represents the rounding error at step j. From (6), we have

$$Q_{j+1}B_j = A\bar{Q}_j - Q_jM_j - Q_{j-1}B_{j-1}^T - F_j$$

and

$$Q_{k+1}B_k = A\bar{Q}_k - Q_kM_k - Q_{k-1}B_{k-1}^T - F_k.$$

Premultiplying the above two equations with Q_k^H and Q_j^H respectively, we get

$$W_{k,j+1}B_j = Q_k^HA\bar{Q}_j - W_{k,j}M_j - W_{k,j-1}B_{j-1}^T - Q_k^HF_j \tag{7}$$

and

$$W_{j,k+1}B_k = Q_j^H A\bar{Q}_k - W_{j,k}M_k - W_{j,k-1}B_{k-1}^T - Q_j^H F_k. \tag{8}$$

From (8) we get

$$Q_j^H A\bar{Q}_k = W_{j,k+1}B_k + W_{j,k}M_k + W_{j,k-1}B_{k-1}^T + Q_j^H F_k. \tag{9}$$

Since the transpose of $Q_j^H A\bar{Q}_k$ is $Q_k^H A\bar{Q}_j$ in (7) and $W_{k,j}^T = \bar{W}_{j,k}$, substituting $Q_k^H A\bar{Q}_j$ in (7) with (9) results in

$$\begin{aligned} W_{k,j+1}B_j &= B_k^T\bar{W}_{k+1,j} + M_k\bar{W}_{k,j} + B_{k-1}\bar{W}_{k-1,j} \\ &\quad -W_{k,j}M_j - W_{k,j-1}B_{j-1}^T + G_{k,j}, \end{aligned} \tag{10}$$

for $k = 1, ..., j-1$, where $W_{j-1,j-1} = \bar{W}_{j,j} = I$ and $G_{k,j} = F_k^T\bar{Q}_j - Q_k^H F_j$ represents the local rounding error. The above equation (10) shows that $W_{k,j+1}$ can be obtained by $W_{k-1,j}$, $W_{k,j}$, $W_{k+1,j}$, and $W_{k,j-1}$ computed in the previous two iterations. In this model, $W_{k,j}$ measures the orthogonality $Q_k^H Q_j$ including rounding error. In the following two sections, we show how to use $W_{k,j}$ to detect the loss of orthogonality.

5. NORMWISE DETECTION

Now that we have established a model of estimating the orthogonality, we propose a scheme for detecting the loss of orthogonality using the norm $\|W_{k,j}\|_2$. Taking the norm on the both sides of (10), we get

$$\begin{aligned} \|W_{k,j+1}\|_2 &\leq \|B_j^{-1}\|_2(\|B_k\|_2\|W_{k+1,j}\|_2 \\ &+ \|B_{k-1}\|_2\|W_{k-1,j}\|_2 + \|B_{j-1}\|_2\|W_{k,j-1}\|_2 \\ &+ (\|M_k\|_2 + \|M_j\|_2)\|W_{k,j}\|_2 + \|G_{k,j}\|_2). \end{aligned} \tag{11}$$

We use this inequality to derive an upper bound $\omega_{k,j}$ for $\|W_{k,j}\|_2$. From (11) we have

$$\omega_{k,j+1} = \tilde{\beta}_j(\beta_k\omega_{k+1,j} + \beta_{k-1}\omega_{k-1,j} + \beta_{j-1}\omega_{k,j-1} + (\alpha_k + \alpha_j)\omega_{k,j}), \tag{12}$$

for $k = 1, ...j-1$, where

$\alpha_k = \|M_k\|_2$
$\beta_k = \|B_k\|_2$
$\tilde{\beta}_k = 1/\sigma_b(B_k)$, where $\sigma_b(B_k)$ is the smallest singular value of B_k

Following,[9] for $G_{k,j}$ in (10), we assume

$$\|G_{k,j}\|_2 \leq \epsilon b,$$

where ϵ is the roundoff unit. Assuming that Q_j and Q_{j+1} are almost orthogonal, we set

$$\omega_{j,j+1} = \epsilon b.$$

This completes our algorithm for computing the estimates $\omega_{k,j+1}$ for $k = 1, ..., j$.

We choose $\sqrt{\epsilon}$ as the threshold for the loss of orthogonality. In the modified partial orthogonalization scheme,[6] Q_{j+1} is orthogonalized against Q_k and its neighboring Q_i when $\omega_{k,j+1} \geq \sqrt{\epsilon}$. The neighborhood $[l_k, u_k]$, in which k lies, is the largest interval such that $\omega_{i,j+1}$ is larger than a predetermined tolerance for all i in $[l_k, u_k]$. The tolerance is chosen as $\epsilon^{3/4}$, a value between ϵ and $\sqrt{\epsilon}$. Whenever we perform orthogonalization in iteration j, we always carry out orthogonalization in the subsequent iteration $j+1$ so that the next block generated by the recurrence is almost orthogonal to its predecessors.

Algorithm 3 (Normwise Detection). *Given an n-by-b starting matrix S of orthonormal columns and a subroutine for matrix-matrix multiplication $Y = AX$ for any X, where A is an n-by-n complex symmetric matrix. This algorithm computes the diagonal blocks of the block tridiagonal complex symmetric matrix J in (1) and a unitary Q such that $J = Q^H A\bar{Q}$*

$steps = n/b$;
$Q_0 = 0$; $B_0 = 0$;
$Q_1 = S$;
$\epsilon_s = \epsilon b$;
$\omega_{1,2} = \epsilon_s$;
for $j = 1$ to $steps$
    $Y = A\bar{Q}_j$;
    $M_j = Q_j^H Y$;
    $R_j = Y - Q_j M_j - Q_{j-1} B_{j-1}^T$;
    $Q_{j+1} B_j = R_j$;   (QR factorization of R_j)
    Compute $\omega_{k,j+1}$ for $k = 1, ..., j-1$ using (12);
    Set $\omega_{j,j+1} = \epsilon_s$;
    Set $\omega_{j+1,j+1} = 1$;
    $k = 1$;
    while $k \leq j$
        if $|\omega_{k,j+1}| > \sqrt{\epsilon}$
            Find the neighborhood $[l_k, u_k]$ of k;
            $k = u_k + 1$;
        else
            $k = k + 1$;
        end
    end
    for each interval $[l_k, u_k]$
        Orthogonalize R_j against $Q_{l_k}, ..., Q_{u_k}$;   (See Algorithm 4)
        Reset $\omega_{i,j+1} = \epsilon\, b$, $i = l_k, ..., u_k$;
        Adjust the neighborhood to $[l_k - 1, u_k + 1]$ for the next iteration;
    end
    if orthogonalization was performed
        Recalculate QR factorization $Q_{j+1} B_j = R_j$;
    end
end.

When the loss of orthogonality is detected, the modified Gram-Schmidt method is used to orthogonalize two blocks.

Algorithm 4 (Orthogonalization of two matrices). *Given two m-by-n complex matrices R and Q, this algorithm orthogonalizes R against Q using the modified Gram-Schmidt method.*

for each column $\mathbf{r}_i$ in R
    for each column $\mathbf{q}_j$ in Q
        orthogonalize $\mathbf{r}_i$ against $\mathbf{q}_j$ using the modified Gram-Schmidt method;
    end
end.

6. COMPONENTWISE DETECTION

Normwise estimation combines the orthogonalities of the vectors in the same block. It cannot reveal the individual orthogonalities within a block. Consequently, unnecessary orthogonalization may be carried out. In this section, we present a componentwise detection scheme based on (10). Let $w_{x,y}$ be the (x, y)-entry of $W_{k,j+1}$, then $w_{x,y} > \sqrt{\epsilon}$ means that the orthogonality between $\mathbf{q}_x$ in Q_k and $\mathbf{q}_y$ in Q_{j+1} is lost. Thus it detects loss of orthogonality more precisely than the normwise scheme.

Our componentwise orthogonalization scheme is based on the model (10). For the term $G_{k,j}$ in (10), we use a block version of modified partial reorthogonalization scheme in[6]:

$$G_{k,j} = \epsilon(B_k + B_j)(\Theta_{\rm r} + i\Theta_{\rm i}), \quad \Theta_{\rm r},\ \Theta_{\rm i} \in N(0, 0.3), \tag{13}$$

where $(B_k + B_j)(\Theta_{\rm r} + i\Theta_{\rm i})$ means that each entry in $B_k + B_j$ is multiplied by a normally distributed random number with zero mean and variance 0.3. Also, we set $W_{j,j+1}$ so that

$$W_{j,j+1}B_j = b\epsilon B_1(\Psi_{\rm r} + i\Psi_{\rm i}), \quad \Psi_{\rm r},\ \Psi_{\rm i} \in N(0, 0.6). \tag{14}$$

Now that we have described the computation of $W_{k,j+1}$ for $k = 1, ..., j$, in the following we discuss the determination of the orthogonalization intervals. Analogous to the normwise method, when the absolute value of a component $w_{x,y}$ of $W_{k,j+1}$ exceeds $\sqrt{\epsilon}$, we find the largest interval $[l_k, u_k]$ such that $k \in [l_k, u_k]$ and for all $i \in [l_k, u_k]$ $W_{i,j+1}$ has at least one component larger than a tolerance. Based on our experiments, we chose $\epsilon^{7/8}$, a value between ϵ and $\sqrt{\epsilon}$, as the tolerance. Our experiments have also shown that for each j, there is usually only one interval, if it exists, and the lower end of the interval is very close to one and the upper end is near j. Thus, to save search time, we always set the lower end to 1 and search for the upper end starting from j.

To incorporate rounding error, after the columns of Q_{j+1} are orthogonalized against the columns of Q_k, we set

$$W_{k,j+1} = \epsilon(\Omega_{\rm r} + i\Omega_{\rm i}), \quad \Omega_{\rm r},\ \Omega_{\rm i} \in N(0, 1.5), \tag{15}$$

that is the entries of $W_{k,j+1}$ are ϵ multiplied by normally distributed random numbers with zero mean and variance 1.5.

As Algorithm 3, whenever we perform the modified partial orthogonalization in iteration j, we always carry out orthogonalization in the subsequent iteration $j + 1$. Suppose that the orthogonalization interval in iteration j is $[1, u]$. As shown in (10), the computation of $W_{k,j+1}$ requires $W_{k-1,j}$, $W_{k,j}$, and $W_{k+1,j}$, thus we expand the orthogonalization interval $[1, u]$ to $[1, u + 1]$ for the subsequent iteration.

Algorithm 5 (Componentwise Detection). *Given an n-by-b starting matrix S of orthonormal columns and a subroutine for matrix-matrix multiplication $Y = AX$ for any X, where A is an n-by-n complex symmetric matrix. This algorithm computes the diagonal blocks of the block tridiagonal complex symmetric matrix J in (1) and a unitary Q such that $J = Q^H A\bar{Q}$*

```
steps = n/b;
Q_0 = 0; B_0 = 0;
Q_1 = S;
doOrtho = 0;
second = 0;
for j = 1 to steps
    Y = A Q̄_j;
    M_j = Q_j^H Y;
    R_j = Y − Q_j M_j − Q_{j−1} B_{j−1}^T;
    Q_{j+1} B_j = R_j;   (QR factorization of R_j)
    if second == 0
        k = 1;
        while (k ≤ j) and (doOrtho != 1)
            if 1 ≤ k ≤ j − 1
                Compute W_{k,j+1} using (10) (13);
            else
                Compute W_{j,j+1} using (14);
            end
            if one component of W_{k,j+1} exceeds √ϵ
                doOrtho = 1;
```

end
$k = k + 1$;
end
if $doOrtho == 1$
$thresh = 0$;
$k = j$;
while ($k \geq 2$) and ($thresh$!= 1)
if $1 \leq k \leq j - 1$
Compute $W_{k,j+1}$ using (10) (13);
else
Compute $W_{j,j+1}$ using (14);
end
if one component of $W_{k,j+1}$ exceeds $\epsilon^{7/8}$
$up = k$;
$thresh = 1$;
end
$k = k - 1$;
end
end
end
if ($doOrtho == 1$) or ($second == 1$)
Orthogonalize R_j against $Q_1, ..., Q_{up}$;
Reset $W_{1,j+1}, ..., W_{up,j+1}$ using (15);
Recalculate QR factorization $Q_{j+1}B_j = R_j$;
if ($second == 1$)
$second = 0$;
else
$second = 1$;
$doOrtho = 0$;
Adjust the neighborhood to $[1, up + 1]$ for the next iteration;
end
end
end.

7. ORTHOGONALIZATION IN TRIDIAGONALIZATION

We apply the modified partial orthogonalization technique described in[6] to the second stage, the tridiagonalization of the block tridiagonal J. The symmetric and block tridiagonal structure of J is exploited in matrix-vector multiplication for efficiency.

ALGORITHM 6 (ORTHOGONALIZATION IN TRIDIAGONALIZATION). *Given a starting vector* $\mathbf{b}$ *and a subroutine for symmetric block tridiagonal matrix-vector multiplication* $\mathbf{y} = J\mathbf{x}$ *for any* $\mathbf{x}$*, where* J *is the complex symmetric block tridiagonal matrix in (1). Using the modified partial orthogonalization, this algorithm computes the diagonals of the complex symmetric and tridiagonal matrix* T *and the unitary* P *in (4).*

$\mathbf{p}_0 = 0$; $\beta_0 = 0$; $\omega_{1,1} = 1$;
$\mathbf{p}_1 = \mathbf{b}/\|\mathbf{b}\|_2$;
for $j = 1$ to n
$\mathbf{y} = J\bar{\mathbf{p}}_j$;
$\alpha_j = \mathbf{p}_j^{\mathrm{H}}\mathbf{y}$;
$\mathbf{r}_j = \mathbf{y} - \alpha_j\mathbf{p}_j - \beta_{j-1}\mathbf{p}_{j-1}$;
$\beta_j = \|\mathbf{r}_j\|_2$;
Compute the estimates $\omega_{k,j+1}$ for $\mathbf{p}_k^H\mathbf{p}_{j+1}$, for $k = 1, ..., j$;

Set $\omega_{j+1,j+1} = 1$;
for each $k = 1, ..., j$ such that $|\omega_{k,j+1}| > \sqrt{\epsilon}$
    Find the neighborhood $[l_k, u_k]$ of k;
end
for each interval $[l_k, u_k]$
    Orthogonalize $\mathbf{r}_j$ against $\mathbf{p}_i$, $i = l_k, ..., u_k$;
    Reset $\omega_{i,j+1}$, $i = l_k, ..., u_k$;
    Adjust the neighborhood to $[l_k - 1, u_k + 1]$ for the next iteration;
end
if orthogonalization was performed
    Recalculate $\beta_j = \|\mathbf{r}_j\|_2$;
end
if $\beta_j = 0$, quit; end
$\mathbf{p}_{j+1} = \mathbf{r}_j / \beta_j$;
end.

8. EXPERIMENTS

Algorithms 3, 5 and 6 were programmed in MATLAB. The random complex symmetric matrices in the following examples were generated as follows. First, a set of n random numbers with normal Gaussian distribution with zero mean and variance 1 was generated. Their absolute values were chosen as the singular values $\sigma_1, ..., \sigma_n$. Then, a random unitary matrix U of order n was generated to form a complex symmetric matrix $A = U\Sigma U^{\mathrm{T}}$. The starting matrix S of orthonormal columns was generated from the QR factorization of a random complex n-by-b matrix. The error in the orthogonality of Q was measured by $\|I - Q^H Q\|_{\mathrm{F}}/n^2$. The error in the tridiagonalization $T = Q^H A\bar{Q}$ was measured by $\|Q^H A\bar{Q} - T\|_{\mathrm{F}}/n^2$. When a vector $\mathbf{q}_j$ was orthogonalized against $\mathbf{q}_k$, it was counted as one orthogonalization.

Example 1. Algorithm 3 (normwise detection) and Algorithm 6 were run on a random complex symmetric matrix of order 2048. Table 1 shows the total number of orthogonalizations and errors for various block sizes.

block size	total number of orthogonalizations	error in orthogonality	error in factorization
2	602542	$2.66E-13$	$2.44E-13$
4	633830	$1.34E-13$	$1.27E-13$
8	662991	$8.81E-14$	$8.38E-14$
16	677998	$2.79E-14$	$2.63E-14$
32	810810	$3.53E-14$	$3.21E-14$

Table 1. Efficiency and accuracy of Algorithms 3 and 6 on a complex symmetric matrix of order 2048.

Example 2. Algorithm 5 (componentwise detection) and Algorithm 6 were run on a random complex symmetric matrix of order 2048. Table 2 shows the total number of orthogonalizations and errors for various block sizes. This example shows that the componentwise algorithm performed fewer orthogonalizations than the normwise algorithm.

Example 3. The single vector Lanczos algorithm with modified partial orthogonalization in[6] and the two block algorithms were run on a random complex symmetric matrix of size 2048. The block size in the block algorithms was set to 32. Table 3 shows the total number of orthogonalizations, errors, and execution time in seconds. This example shows that the block algorithms are almost twice as fast as the single vector algorithm, although the block algorithms performed more orthogonalizations than the single vector algorithm.

Example 4. To compare the performance, we generated random complex symmetric matrices of various sizes and ran the single vector algorithm and the two block algorithms. Figure 1 depicts the run times which are

block size	total number of orthogonalizations	error in orthogonality	error in factorization
2	437969	$8.13E-14$	$7.92E-14$
4	444056	$1.32E-12$	$2.28E-13$
8	425837	$8.00E-14$	$7.72E-14$
16	433081	$1.40E-13$	$1.38E-13$
32	413307	$1.64E-13$	$1.52E-13$

Table 2. Efficiency and accuracy of Algorithms 5 and 6 on a complex symmetric matrix of order 2048.

algorithm	total number of orthogonalizations	error in orthogonality	error in factorization	run time (seconds)
single vector	213477	$5.56E-14$	$5.15E-14$	2151
normwise	807009	$4.81E-14$	$4.40E-14$	1287
componentwise	453775	$4.09E-14$	$3.79E-14$	1075

Table 3. Comparison of block Lanczos algorithms with single vector Lanczos algorithm.

normalized so that the execution time of the single vector algorithm is 100. In all cases, the block size in the block algorithms is 32. This example shows that the block algorithm with componentwise detection has the best performance for large matrices.

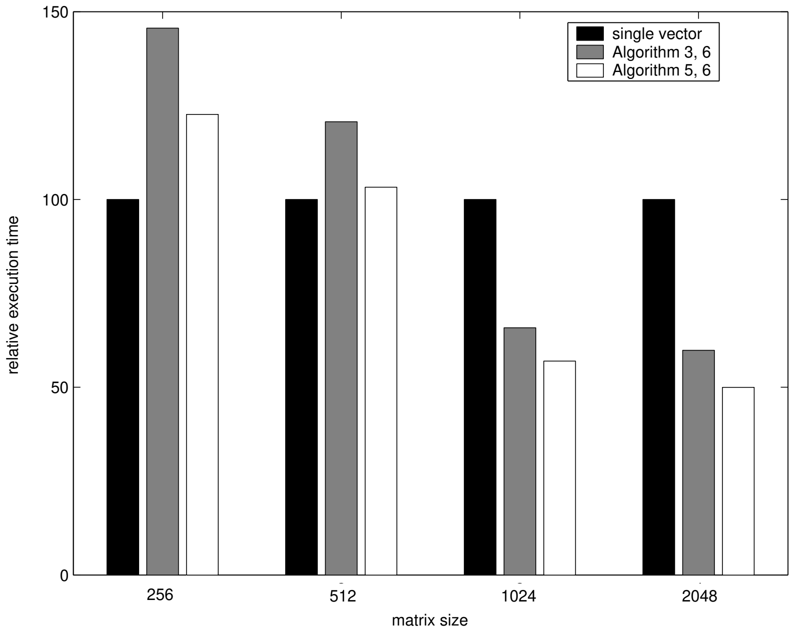

Figure 1. Comparison of the efficiency of block Lanczos algorithms with single vector Lanczos algorithm. The y axis shows the execution times normalized to the execution time of the single vector algorithm.

Example 5. This example shows the effect of block size on the efficiency of the block Lanczos algorithms. A complex symmetric matrix of order 2048 was generated and tried on the single vector Lanczos algorithm and the two block Lanczos algorithms with various block sizes. Figure 2 shows the execution times of these algorithms in seconds. Although the performance depends on the architecture of the machine, this example illustrates that the block Lanczos algorithms with large block sizes are very efficient.

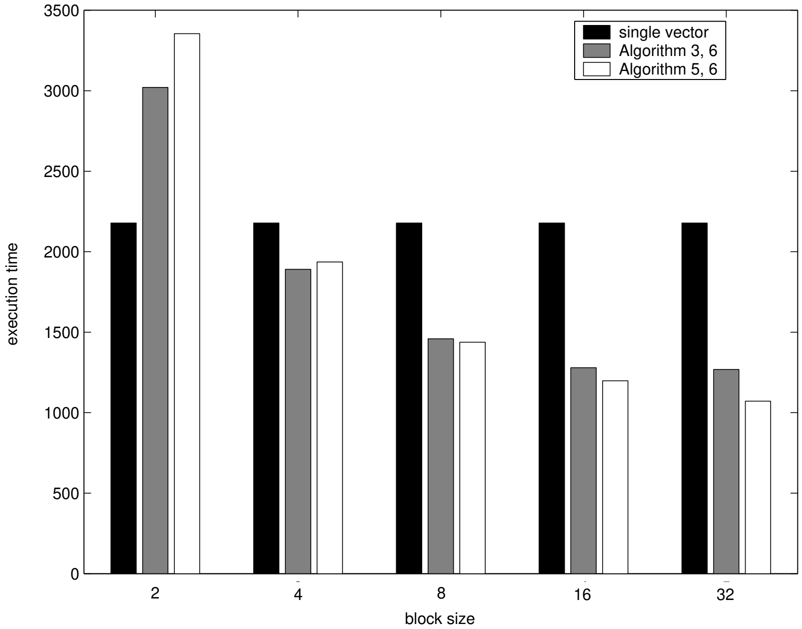

Figure 2. The effect of the block size on the efficiency of block Lanczos algorithms. The y axis is the execution time in seconds.

Conclusion. In this paper, we have presented the block Lanczos tridiagonalization algorithms of a complex symmetric matrix. Experimental results show that the block Lanczos tridiagonalization algorithms are more efficient than the single vector Lanczos tridiagonalization algorithm for large matrices and large block sizes. Our experiments also show that the componentwise detection for the loss of orthogonality in block tridiagonalization is more efficient than the normwise detection.

ACKNOWLEDGMENTS

This work is partially supported by Natural Sciences and Engineering Research Council of Canada.

REFERENCES

1. R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.
2. T. Takagi, "On an algebraic problem related to an analytic theorem of Carathédory and Fejér and on an allied theorem of Landau," *Japan J. Math.* **1**, pp. 82–93, 1924.
3. A. Bunse-Gerstner and W. B. Gragg, "Singular value decompositions of complex symmetric matrices," *Journal of Computational and Applied Mathematics* **21**, pp. 41–54, 1988.
4. F. T. Luk and S. Qiao, "A fast singular value algorithm for Hankel matrices," in *Fast Algorithms for Structured Matrices: Theory and Applications, Contemporary Mathematics 323*, V. Olshevsky, ed., pp. 169–177, American Mathematical Society, 2003.
5. W. Xu and S. Qiao, "A divide-and-conquer method for the Takagi factorization," Tech. Rep. CAS 05-01-SQ, Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada L8S 4K1, February 2005.
6. S. Qiao, "Orthogonalization techniques for the Lanczos tridiagonalization of complex symmetric matrices," in *Advanced Signal Processing Algorithms, Architectures, and Implementations XIV*, F. T. Luk, ed., **5559**, pp. 423–434, SPIE, August 2004.
7. H. D. Simon, "The Lanczos algorithm with partial reorthogonalization," *Mathematics of Computation* **42**, pp. 115–142, 1984.

8. M. Berry, G. O'Brien, V. Krishna, and S. Varadhan, "SVDPACKC (version 1.0) user's guide," Tech. Rep. CS-93-194, Department of Computer Science, University of Tennessee, 1993.
9. R. Grimes, J. Lewis, and H. Simon, "A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems," *SIAM J. matrix Anal. Appl.* **15**, pp. 228–272, 1994.