Vol. 47 No. SIG 7(ACS 14) IPSJ Transactions on Advanced Computing Systems May 2006

Regular Paper

An Implementation of the Block Householder Method

Hiroshi Murakami†

When large matrix problems are treated, the locality of storage reference is very important. Usually higher locality of storage reference is attained by means of block algorithms. This paper introduces an implementation of block Householder transformation based on the block reflector (Schreiber, 1988) or "GG^T" representation rather than on the method using "WY^T" representations or compact "WY^T" or "YTY^T" (Bischof, 1993, etc.). This version of block Householder transformation can be regarded as a most natural extension of the original non-blocked Householder transformation, with the matrix elements of the algorithm changed from numbers to small matrices. Thus, an algorithm that uses the non-blocked version of Householder transformation can be converted into the corresponding block algorithm in the most natural manner. To demonstrate the implementation of the Householder method based on the block reflector described in this paper, block tridiagonalization of a dense real symmetric matrix is carried out to calculate the required number of eigenpairs, following the idea of the two-step reduction method (Bischof, 1996, etc.).

1. Introduction

Large matrix problems usually require large amounts of storage space, and not all of the data can be placed in local storage at the same time. Consequently, most of them must be held in lower-hierarchy storage that is larger but takes longer to access. In such cases, blocked algorithms have the great advantage that they increase the locality of storage reference and markedly reduce the data transfer between hierarchies. With high-speed local storage such as cache memory, even if the storage space is small, the computation can proceed smoothly on account of the very high locality of the reference.

Several block Householder transformation methods already exists, such as those based on the "WY^T" representations [1),4),6),7)], and compact "WY^T" representation or "YTY^T" representation [15)20),21)], for the block reflector or the "GG^T" representation [14)].

WY^T representation:

The WY^T representation is the aggregation of the series of ordinary Householder transformations $Q_i = I - v^{(i)}(v^{(i)})^T,\ i = 1, \cdots, r \in \mathbf{R}^{m\times m}$. The equation $Q = Q_1Q_2\cdots Q_r$ can be represented in the matrix form $Q = I + WY^T$, where $W, Y \in \mathbf{R}^{m\times r}$, and the representation is computed by using the recurrence relation for W and Y for increasing r (§5.1.1-5.1.2 of Golub and van Loan [11)]). The calculation of application QA or QAQ^T is rich in matrix-matrix multiplications and higher performance is obtained with the level-3 BLAS routines, which have high locality of memory reference. Note that aggregated Q is orthogonal but not symmetric.

Compact WY^T representation:

Similarly, compact WY^T representation is the aggregation of the ordinary Householder transformations in the form $Q = I + YTY^T$, $Y \in \mathbf{R}^{m\times r}$, $T \in \mathbf{R}^{r\times r}$, and is upper triangular. When $m \gg r$, the amount of storage needed for representation is less than that for the original WY^T representation, since W need not be held and T is small.

Block reflector:

Using the matrix of rank r orthonormal vectors $U \in \mathbf{R}^{m\times r}$, the block reflector has the form $H = I - 2UU^T$. The matrix H is orthogonal and symmetric.

Since $C \in \mathbf{R}^{m\times n}$ is a rank r matrix, if a matrix U of rank r is chosen suitably, all rows of HC are eliminated to zeros except the first r rows. The rank of the first r rows is r and full.

The reflector version of the block Householder transformation can be regarded as a natural extension of the original non-blocked Householder transformation, with the matrix elements of the algorithm changed from numbers to small matrices. Therefore, an algorithm using the non-blocked version of Householder transformation would have a corresponding blocked version algorithm, and could be converted naturally.

The block reflector method of Schreiber and Parlett [14)] is implemented and explained in this

† Faculty of Urban Liberal Arts, Tokyo Metropolitan University

paper, but it is constructed by using singular value decomposition (SVD) rather than polar decomposition, since SVD is more familiar and more widely studied in the area of numerical computation.

In this paper, as an example of the block Householder method, the eigensolver of a dense real symmetric matrix similar to the two-step reduction method [2),3)] is tested. The eigenproblem of a symmetric dense matrix is first condensed into a problem of a symmetric block tridiagonal matrix by the block Householder transformations, and is then solved by the band eigensolver for the required eigenvalues and the corresponding eigenvectors. The eigenpairs of the original dense symmetric matrix can be obtained from the eigenpairs of a block tridiagonal matrix by application of the backward block Householder transformations.

In the next section, we will explain a construction of the block reflector of Schreiber and Parlett based on the SVD, which is different from Algorithm 2 of Schreiber and Parlett [14)] based on polar decomposition. Note that both block reflectors constructed in the form $H = I - 2UU^T$ are the same; the only difference is the orthogonal factor in the matrix U.

2. Block Householder Transformation

In this section, a SVD-based algorithm for constructing the block Householder vector U to give the two-sided block Householder transformation $H = I - 2UU^T$ will be presented.

For simplicity of description, matrices of complex numbers are not treated in this paper. We write $I_k \in \mathbf{R}^{k\times k}$ to denote the identity matrix of size k, and $E_k \in \mathbf{R}^{m\times k}$ to denote the generalized identity matrix whose (i,j)-th element is $\delta_{i,j}$.

2.1 Definition of a Block Reflector

For a given matrix $C \in \mathbf{R}^{m\times b}$ and $r = \mathrm{rank}(C) \le b$, there exists a matrix $U \in \mathbf{R}^{m\times r}$ such that $U^TU = I_r$ and also all rows of HC are eliminated to zeros except the first r rows, where $H = I_m - 2UU^T$. The matrix H is symmetric and orthogonal, since $H^T = H$ and $H^TH = I_m$. The matrix U is the block Householder vector constructed from C, and H is the matrix of block Householder transformation.

2.2 Construction of a Block Reflector

Assuming $b \le m$, for the given matrix $C \in \mathbf{R}^{m\times b}$, the algorithm below constructs the block Householder vector $U \in \mathbf{R}^{m\times r}$ and $\beta \in \mathbf{R}^{r\times b}$, such that $\mathrm{rank}(\beta) = r$ and $H = I_m - 2UU^T$ is the block reflector which has the elimination property $HC = E_r\beta$.

2.2.1 Algorithm

The following steps from STEP-1 to STEP-4 compute the block Householder vector U and β, the top non-zero part of HC.

STEP-1: Let "any" orthogonal decomposition of C be $C \Rightarrow XZ$, and let $r = \mathrm{rank}(C) \le b$. ($X \in \mathbf{R}^{m\times r}$ and $Z \in \mathbf{R}^{r\times b}$.) Usually, in-place decomposition is available and X overwrites C.
If $r = 0$, let U and β be rank zero and finish computation. Hereafter it that $r > 0$ is assumed.

STEP-2: Let $\widehat{X} \in R^{r\times r}$ denote the first r rows of X. Let $\widehat{X} \Rightarrow WDV$ be SVD of $\widehat{X}$. Here $W, V \in \mathbf{R}^{r\times r}$ are orthogonal matrices and $D \in R^{r\times r}$ is the diagonal matrix of singular values.
Let $\beta = (-WV)Z$.

STEP-3: Let $Y = X + E_rWV \in \mathbf{R}^{m\times r}$; that is, Y overwrites X by adding WV to part of the first r rows.

STEP-4: Let "any" orthonormalization of Y be $U \in \mathbf{R}^{m\times r}$ and let $U^TU = I_r$.
The computation of U can be made much more efficient by using the level-3 BLAS routine DGEMM where $U = Y\,[V^T\{2(I_r + D)\}^{(-1/2)}]$ instead of orthonormalization. In-place computation is easy if columns are partitioned into blocks.
U is the block Householder vector of rank r to give $HC = E_r\beta$.
By the above chain of overwrites, U may be written in the place of C.

2.2.2 Properties of Intermediate Quantities

Properties of the intermediate quantities in the above construction from STEP-1 to STEP-4 of the algorithm are explained here. Suppose that $m, b, r \in \mathbf{N}$ and $0 \le r \le b \le m$.

In STEP-1: X consists of r orthonormal vectors and $X^TX = I_r$. Z is full rank i.e. $\mathrm{rank}(Z) = r$.

Note that in the case where $r = 0$, $C = 0$. U is $\mathbf{R}^{m\times 0}$ and $H = I_m \in \mathbf{R}^{m\times m}$ is identity. β is $\mathbf{R}^{0\times b}$. Thus there is no problem.

In STEP-2: $I_r = X^TX = \widehat{X}^T\widehat{X} + (X - E_r\widehat{X})^T(X - E_r\widehat{X}) = V^TD^2V + (X -$

$E_r\widehat{X})^T(X - E_r\widehat{X})$, and therefore $I_r - D^2 = \{(X - E_r\widehat{X})V^T\}^T\{(X - E_r\widehat{X})V^T\}$. Since the right-hand side is symmetric semi-positive definite, so is $I_r - D^2$. Therefore the singular values do not exceed 1. (The standard selection of non-negative signs for the singular values in SVD makes the calculation in later steps numerically most stable as the sign selection in the ordinary Householder transform.)

In STEP-3: The relation $Y^TY = V^T2(I_r + D)V$ is derived from both $X^TX = I_r$ and $X^TE_r = \widehat{X}^T$. Y is well-conditioned and $\mathrm{cond}(Y) \leq \sqrt{2}$, since all diagonals of $2(I_r + D)$ are between 2 and 4.

In STEP-4: Let "any" orthonormalization of Y be written as $Y \Rightarrow US$. The matrix $S \in \mathbf{R}^{r\times r}$ is invertible, since $\mathrm{cond}(S) \leq \sqrt{2}$.

The matrix U can be multiplied by any orthogonal matrix of size r from the right side. The special selection $U = Y[V^T\{2(I_r + D)\}^{(-1/2)}]$ also gives an orthonormalization.

From $U^TU = I_r$, the symmetry and orthogonality of $H = I_m - 2UU^T \in \mathbf{R}^{m\times m}$ follows. Moreover, as a block Householder transformation, H satisfies the following elimination property.

Lemma 1 *For a given matrix $X \in \mathbf{R}^{m\times r}$ such that $X^TX = I_r$, the matrix H constructed following STEP-2 to STEP-4 transforms the matrix X into $HX = E_r(-WV)$, which means that all rows of HX are zeros except for the first r rows, and the matrix of the first r rows of HX is $(-WV)$.*

Proof *Since $Y = X + E_rWV$ and $\widehat{X} = WDV$, $E_r^TY = \widehat{X} + WV = W(I_r + D)V$.*
From the relation $(E_rWV)^TY = V^TW^T(E_r^TY) = V^T(I_r + D)V = (1/2)Y^TY$, by transposition we get $Y^TE_rWV = (1/2)Y^TY$, and also from $E_rWV = Y - X$ we get $Y^T(Y - 2X) = 0$.
Since S is not singular and $Y^T = S^TU^T$, $U^T(Y - 2X) = 0$; that is, $2U^TX = U^TY$.
Since $U^TY = S$ and $US = Y$, $UU^TY = Y$, and thus we get $2UU^TX = U(2U^TX) = UU^TY = Y$.
Therefore $HX = X - 2UU^TX = X - Y = E_r(-WV)$, which was to be proved. ∎

Using the above Lemma 1, the matrix H is constructed from the orthonormal X of rank r, and therefore $HX = E_r(-WV)$ holds. Hence, $HC = HXZ = E_r(-WV)Z = E_r\beta$. Here $\beta \in \mathbf{R}^{r\times b}$ is defined as $\beta = (-WV)Z$, which is full rank, i.e. $\mathrm{rank}(\beta) = r$. Therefore, the matrix of the first r rows of HC is β, and the others are zeros.

If the Householder QR-decomposition is used for the orthonormalization of C, then the relation $C\Pi = XR$ holds. Here, $R \in \mathbf{R}^{r\times b}$ is upper triangular and Π is the permutation matrix of column pivoting. When decomposition is without pivoting, Π is just an identity. In that case, the QR-decomposition of β is given by $\beta\Pi = (-WV)R$, since $(-WV) \in \mathbf{R}^{r\times r}$ is an orthogonal matrix.

2.3 Another Construction

Note that even when the matrix C is rank-deficient, by taking the extended orthonormal basis of C the matrix X can be made rank b and $X \in \mathbf{R}^{m\times b}$. The extended orthonormal basis may be constructed quite naturally by the Householder QR-decomposition, for example. In that case, the (effective) rank of the matrix $Z \in \mathbf{R}^{b\times b}$ is r. Applying the above procedure of Lemma 1 (replacing r with b), from X the extended orthonormal basis of rank b in the matrix H is constructed and $HX = E_b(-WV)$. Thus, $HC = HXZ = E_b(-WVZ) = E_b\beta$. Here, the matrix $\beta = -WVZ \in \mathbf{R}^{b\times b}$ and $\mathrm{rank}(\beta) = r$. Therefore, the matrix of the first b rows of HC is β, and all other rows are zeros.

If the Householder QR-decomposition is used, the orthogonal decomposition is $C\Pi = XR$. Here, Π is the permutation matrix of column pivoting and X is the matrix of the extended orthonormal basis of rank b. The matrix $R \in \mathbf{R}^{b\times b}$ is upper triangular, and all the rows are zeros except the first r rows. The last zero-valued $(b-r)$ rows of R correspond to the redundant orthonormal basis. In this case, the QR-decomposition of β is $\beta\Pi = (-WV)R$, where the matrix $(-WV) \in \mathbf{R}^{b\times b}$ is orthogonal. $R \in \mathbf{R}^{b\times b}$ is upper triangular and all rows are zeros except the first r rows.

In both of the above methods, the block Householder reflector H is constructed from the given general matrix C. If $\mathrm{rank}(C) = r$, the first r rows of HC are of rank r, and the other rows of HC are eliminated to zeros.

If we regard matrix $C \in \mathbf{R}^{m\times b}$ as a block vector whose elements are small matrices of size b, then C has $n = \lceil m/b \rceil$ block elements. (The last fragmented element of a block vector may not be square b-by-b, but we may always use a b-by-b matrix to store the fragmented elements, and zeros are used to pad the unoccupied places in the matrix.) Then, by the block Householder transformation, the block vector C is transformed into the one such that the first block

element is $\beta \in \mathbf{R}^{b\times b}$ and all other elements are zeros. If $r = \mathrm{rank}(C)$, then the rank of β is also r and all the rows of β except the first r are zeros. The matrix U, which represents the Householder matrix H, can also be regarded as a block vector. The matrix $A \in \mathbf{R}^{m\times m}$ can also be regarded as a block matrix of n-by-n of b-by-b small matrices.

3. Application of a Block Reflector

In this section, it is shown that by successive applications of the block reflectors, the dense symmetric matrix A can be condensed into the block tridiagonal matrix T in a manner similar to the ordinary Householder tridiagonalization and that, after the eigensolutions of the block tridiagonal matrix T have been obtained, the eigenvectors of T can be transformed by backward block reflections into eigenvectors of A.

The flow of this two-step reduction eigensolver is as follows:

(1) [**BHSHLD**] By the block Householder transformations, the symmetric matrix A is transformed into the block tridiagonal matrix T. The information needed to construct the transformations is stored in a place in A for later use in backward transformations.

(2) [**BT2BND**] T is transformed into the symmetric band matrix B in order to halve the bandwidth.

(3) The number of required eigenpairs of the band matrix B is obtained by using Murata's method as follows:

- (a) [**TRIDIA**] B is transformed into the ordinal tridiagonal matrix t by Murata's band Householder tridiagonalization.
- (b) [**BISECT**] The required eigenvalues of t are calculated by the Sturm bisection method.
- (c) [**INVITR**] For each of the obtained eigenvalues of t, the corresponding eigenvector of B is obtained by using inverse iterations of the band LU-decomposition with selective reorthogonalization.

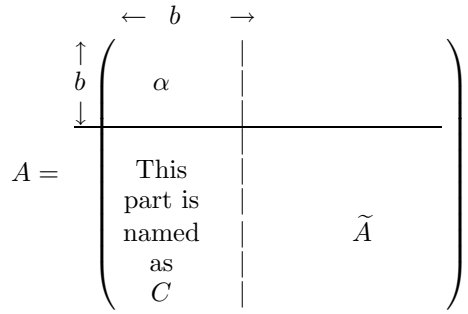
(4) [**BND2BT**] The obtained eigenvectors of B are simultaneously transformed backward into eigenvectors of T by the inverse of the transformation previously used in step (2).

(5) [**BCKTRS**] The eigenvectors of T are simultaneously transformed backward into eigenvectors of A by the block Householder transformations used and stored previously in a place in A in step (1), in reverse order.

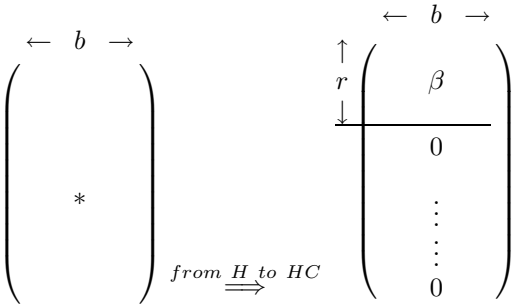
BHSHLD, BT2BND, TRIDIA, BISECT, INVITR, BND2BT and BCKTRS in the above are the names of computation steps. In this section, further explanations of these steps will be given.

3.1 Block Tridiagonalization of a Dense Matrix [BHSHLD]

For a given dense symmetric matrix A, the solution of the eigenproblem of A is considered. The matrix is partitioned by a small matrix of size b. The first block column vector of A, excluding the first element, is regarded as the block column vector C (see the figure below).



The block transformation H is thus determined by the construction described in the previous section as STEP-1 to STEP-4 of the algorithm, so that all elements of HC except the first r rows are eliminated, where $r(\leq b)$ is the effective rank of C (see the figure below).



The calculation HAH transforms the principle block column or row into tridiagonal form (see the figure below).

$$
\begin{array}{c}
\begin{array}{cc} & \leftarrow b \rightarrow \quad \leftarrow r \rightarrow \end{array} \\
\left(\begin{array}{cc|ccc}
\uparrow & & & & \\
b & \alpha & \beta^T & 0 & \cdots \\
\downarrow & & & & \\
\hline
\uparrow & & & & \\
r & \beta & & & \\
\downarrow & & & & \\
 & 0 & & H\widetilde{A}H & \\
 & & & & \\
 & \vdots & & & \\
 & & & & \\
 & & & &
\end{array}\right)
\end{array}
$$

As in ordinary tridiagonalization, this block transformation is applied until the matrix is in a block tridiagonal form. At each step, the transformation reduces the size of the matrix by a block. It is not necessary to take the block size of each column to be uniform. Continuing this operation finally yields the expected block tridiagonal matrix (see the figure below).

$$
H^{(n-2)} \cdots H^{(3)} H^{(2)} H^{(1)} A H^{(1)} H^{(2)} H^{(3)} \cdots H^{(n-2)}
$$

$$
= \begin{pmatrix}
\alpha_1 & \beta_1^T & 0 & \cdots & & & & \\
\beta_1 & \alpha_2 & \beta_2^T & 0 & \cdots & & & \\
0 & \beta_2 & \alpha_3 & \beta_3^T & 0 & & & \\
\vdots & 0 & \beta_3 & \alpha_4 & \beta_4^T & & & \\
 & \vdots & & & & & & \\
 & & 0 & \ddots & \ddots & & & \vdots \\
 & & & & & \ddots & \ddots & 0 \\
 & & & & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1}^T \\
 & & & & \cdots & 0 & \beta_{n-1} & \alpha_n
\end{pmatrix}
$$

3.2 Practical Computation of HAH

Inside the first column of the block matrix HAH, α is a small matrix in the first position and is unchanged, and the first block element of HC with rank $r = rank(C)$ will be the small matrix β in the next position. All the other block elements in the first block column are eliminated.

Therefore, we need not compute the transform of the first block column, but from the top of the column α is selected and stored, and then in the construction of the block Householder vector U the matrix β is obtained and is also stored, and the part C is overwritten by U for later use in the backward transform.

If we denote by $\widetilde{A}$ the part of A whose size is reduced by one block, the update $A \leftarrow HAH$ for that part of A can be represented by using the auxiliary block vector P and the symmetric small matrix G as follows:

$$
\begin{cases}
P \leftarrow \widetilde{A}U, \\
G \leftarrow U^T P, \\
P \leftarrow (-2)(P - UG)), \\
\widetilde{A} \leftarrow \widetilde{A} + UP^T + PU^T.
\end{cases}
$$

This computation is rich in the matrix-matrix multiplications of small matrices of size b, and can be implemented by using the level-3 BLAS routines [5] to obtain high speed and locality of storage reference. In practice, the symmetry of the block matrix A is used and only the lower half of A is stored as column block vectors, because the small matrix of the block matrix A can be accessed contiguously in direction of the column (see the figure below).

$$
\begin{pmatrix}
* & & & & \\
\downarrow & & & & \\
* & * & & & \\
\downarrow & \downarrow & & & \\
* & * & * & & \\
\downarrow & \downarrow & \downarrow & & \\
* & * & * & * & \\
\downarrow & \downarrow & \downarrow & \downarrow & \\
* & * & * & * & *
\end{pmatrix}
$$

Backward transformation of eigenvectors [BCKTRS]: This operation is also similar to the ordinary Householder method.

After the required eigenvectors of the block tridiagonal matrix T have been calculated, they must be transformed back into the eigenvectors of the original matrix A.

To transform the eigenvector y of T into the eigenvector x of the original matrix A, the block Householder vectors $\{U^{(i)}\}$ which were previously determined and stored are retrieved in reverse order. Using $\{U^{(i)}\}$, the block Householder transformations $\{H^{(i)}\}$ are constructed and applied to y as $x \leftarrow H^{(1)} H^{(2)} \cdots H^{(n-2)} y$ from right to left.

For efficiency, many eigenvectors should be combined and simultaneously transformed. The application of the block reflector $H^{(i)}$ to a set of vectors Y simultaneously can be made inplace as $Y \leftarrow H^{(i)}Y$, and is also rich in matrix-matrix multiplications. The reflection is iterated for $i = (n-2), \cdots, 1$. Initially, Y is a combined set of eigenvectors of T, and transformed to a combined set of corresponding eigenvectors of A.

3.3 Considerations

Storage: For the case when eigenvectors are required later, the block Householder vector U and its effective rank must be reproduced in the backward transformations. Therefore, at each step of the forward transformation, the block Householder vector U is stored in the column of A which the transformation has just eliminated. The small matrices α_i and β_i can be stored by overwriting the i-th block elements to the places U_i and P_i , since they are shrinking block vectors.

Amount of computation: Here, N is the true size of the matrix A, and the computational order of the block tridiagonalization of A is $O(N^3)$, which is the same order as that realized by the non-blocked tridiagonalization. When $b \ll N$, if the multiplications and additions are counted separately, it is about $(4/3)N^3$ floating point operations.

Amount of storage transfer: Block tridiagonalization of block size b reduces the total amount of the storage transfer by a factor of about b, because the number of the sweeps of the transforming matrix A is reduced from $N-2$ to $\lceil N/b \rceil - 2$. The same is true for backward transformation.

Storage referencing is also reduced by a factor of b, since the amount of floating computation is not greatly affected when the blocking is done. If the value of b is chosen appropriately, the speed of computation and the speed of storage transfer can be balanced.

However (with the combination of methods used in the current implementation), adoption of an excessively large value of b should be avoided, especially when ℓ, the required number of eigenvectors, is very large, because in the later steps of calculation, where the eigenproblem of the block tridiagonal matrix is solved by Murata's band eigensolver [12),19)], which uses the band matrix Householder tridiagonalization and the band LU-decomposition inverse iterations, the amounts of computation are $O(N^2 b)$ and $O(Nb^2\ell)$, respectively.

The current code of the band eigensolver [19)] was written for vector architecture machines and does not use blocked algorithms and assumed the higher memory bandwidth. For the block tridiagonalization, the number of memory accesses may be $O(N^3/b)$.

Case-1. If ℓ is proportional to N, that is, $\ell = O(N)$, even if it is only a small fraction of N, then asymptotically the inverse iteration process makes more memory accesses than the band Householder method, and the balancing condition between $O(N^3/b)$ and $O(Nb^2\ell)$ gives $b = O(N^{1/3})$. In that case, the total number of memory accesses will be $O(N^{8/3})$.

Case-2. If ℓ is a constant and independent of N (even if the value of ℓ is large), then asymptotically the band Householder method makes more memory accesses than the inverse iteration process, and the balancing condition between $O(N^3/b)$ and $O(N^2 b)$ gives $b = O(N^{1/2})$. In this case, the total number of memory accesses will be $O(N^{5/2})$.

Therefore (for the current combination of the methods, namely the eigenvalue solver given by the band matrix Householder tridiagonalization and the inverse iteration given by band LU-decomposition iterations), $b = O(N^{1/2})$ is optimal when ℓ is a constant of N and $b = O(N^{1/3})$ when $\ell = O(N)$.

When the number of required eigenpairs is large, if the recently developed divide-and-conquer algorithms for the band matrix or the block tridiagonal matrix [8),9)] were used, the situation could be different, especially when the number of required eigenpairs is very large and $\ell = O(N)$. The inverse iterations with LU-decomposition approach would be useful only in cases where the number of required eigenpairs is not excessive.

3.4 Transform Eigenproblem of a Block Tridiagonal into a Band Matrix [BT2BND]

3.4.1 Reduction of a Block Tridiagonal Matrix to a Band Matrix:

In the following, it is assumed for simplicity that the block sizes of all columns are uniform.

A block tridiagonal matrix T of block size b can be regarded naturally as a band matrix of width $2b-1$. Therefore, the method of solution for the eigenproblem of the symmetric band matrix is directly applicable to the symmetric block tridiagonal matrix T. But the matrix T of bandwidth $2b-1$ can be reduced to the symmetric band matrix of width b with a small amount of computation.

In an experiment, the code of Murata's band Householder method and the method of inverse iterations by LU-decomposition [12),19)] where used to solve the band eigenproblem. In the method, the band matrix is stored in packed form. If w is the bandwidth of the packed band matrix to Murata's program, the count of operations in the band Householder method

is $O(N^2w)$ and in the inverse iterations it is $O(Nw^2\ell)$, where ℓ is the number of eigenvectors required. (The operation count of the Sturm bisection is independent of w.) Therefore, if the bandwidth w is reduced here in advance from $2b-1$ to b, the band eigensolver runs faster.

The symmetric block tridiagonal matrix T can be written as

$$T = \begin{pmatrix} \alpha_1 & \beta_1^T & & & & \\ \beta_1 & \alpha_2 & \beta_2^T & & & \\ & \beta_2 & \alpha_3 & \beta_3^T & & \\ & & \beta_3 & \ddots & \ddots & \\ & & & \ddots & \ddots & \beta_{n-1}^T \\ & & & & \beta_{n-1} & \alpha_n \end{pmatrix}$$

In the first step, paying attention to the second row, β_1 is QR-decomposed as $\beta_1 \rightarrow Q_1 R_1$, where Q represents the orthogonal matrix and R represents the right triangular matrix. (In the actual computation here, the transformation matrix Q_1 may not formed explicitly but kept in "factored form".) After the QR-factorization of β_1, the orthogonal transformation

$$\mathrm{diag}[I, Q_1^T, I, \cdots, I] \;\; T \;\; \mathrm{diag}[I, Q_1, I, \cdots, I]$$

$$= \begin{pmatrix} \alpha_1 & R_1^T & & & & \\ R_1 & Q_1^T\alpha_2 Q_1 & (\beta_2 Q_1)^T & & & \\ & \beta_2 Q_1 & \alpha_3 & \beta_3^T & & \\ & & \beta_3 & \ddots & \ddots & \\ & & & \ddots & \ddots & \beta_{n-1}^T \\ & & & & \beta_{n-1} & \alpha_n \end{pmatrix}$$

is applied.

Analogously, at the k-th step, the sub-diagonal element of the $(k+1)$-th row is QR-decomposed by the Householder-QR method without pivoting, and Q_k and R_k are then determined.

The factor of QR-decomposition Q_k is stored in "factored form" and R_k is overwritten to the place where β_k was stored. The transpose of Q_k is multiplied from the left to the $(k+1)$-th row, and Q_k is multiplied from the right to the $(k+1)$-th column.

By symmetry, only the data in the lower half of the main diagonal block elements and in the left sub-diagonal block elements are used in the calculation. After the $(n-1)$-th step has been completed, the block tridiagonal matrix T transformed into the symmetric band matrix B of bandwidth b is obtained as follows:

$$B = \mathrm{diag}[I, Q_1^T, \cdots, Q_{n-1}^T]\; T\; \mathrm{diag}[I, Q_1, \cdots, Q_{n-1}]$$

$$= \begin{pmatrix} \gamma_1 & R_1^T & & & & \\ R_1 & \gamma_2 & R_2^T & & & \\ & R_2 & \gamma_3 & R_3^T & & \\ & & R_3 & \ddots & \ddots & \\ & & & \ddots & \ddots & R_{n-1}^T \\ & & & & R_{n-1} & \gamma_n \end{pmatrix}.$$

3.4.2 Backward Transformation of Eigenvectors [BND2BT]:

To obtain y, the eigenvector of T, from z, the eigenvector of the symmetric band matrix B, the backward transformation: $y \leftarrow \mathrm{diag}[I, {Q_1}^T, \cdots, {Q_{n-1}}^T]\, z$ is used.

To enhance the locality of references, many eigenvectors are simultaneously transformed in practice, and y and z are regarded as block vectors in which many columns consisting of eigenvectors are transformed at once.

3.4.3 Amount of Computation Needed to Reduce the Bandwidth:

The order of each computation of the QR-factorization of a small matrix is $O(b^3)$ and the multiplication of each Q_k is $O(b^3)$. As $n = N/b$, the total computation of the forward transformation is $O(b^2 \times N)$ and the total amount of computation carried out to apply the reverse transformation to the eigenvectors is $O(b \times \ell \times N)$, where ℓ is the number of eigenvectors to be transformed back.

3.5 Solution of the Eigenproblem of a Symmetric Band Matrix:

To tridiagonalize the band matrix, instead of the Rutishauser-Schwarz method in (§II/8 of Wilkinson and Reinsch [17]), the band Householder method proposed by Murata [12] is used.

The eigenproblem of the symmetric band matrix is solved by a combination of the following methods:

- **[TRIDIA]** Murata's band matrix Householder tridiagonalization.
- **[BISECT]** Sturm bisection for eigenvalues of a tridiagonal matrix.
- **[INVITR]** inverse iteration band eigensolver by LU-decomposition with selective reorthogonalization.

The Fortran 77 code for these three steps,

TRIDIA, BISECT, and INVITR, is taken from a book by Murata, et al. [19], which was written for the vector supercomputers at that time.

Note that a new algorithm that is an improved version of Murata's method was tested and described in Bischof, et al. [3]. However, only the code in Murata, et al. [19] is used in this paper.

Let ℓ be the required number of eigenpairs. The computational order used to compute ℓ eigenpairs by this specific method is

$$\begin{aligned}&\text{tridiagonalization} + \text{bisection}\\ &+\text{inverse iterations}\\ &= O(N^2 w) + O(N\ell \log_2(1/\epsilon)) + O(Nw^2\ell)\,,\end{aligned}$$

where w is the bandwidth of the matrix, N is the scalar dimension of the matrix, and ϵ is the tolerance for the separation of eigenvalues by the bisection. By the previous transformation to reduce the bandwidth, the value of w is reduced from $2b-1$ to b.

If the bandwidth is reduced, the total amount of computation for the solution of a block tridiagonal matrix will be reduced, even if the reduction of the bandwidth is consumed.

3.5.1 Storage Consideration:

The storage requirement for Murata's band Householder method is

- $N\times(w+1)$ words for the band matrix B,
- $3N + (3w+1)\times N = (3w+4)\times N$ of real words, and N integer words for the other working storage,
- $N\times\ell$ words to store the eigenvectors.

The storage for eigenvectors and the working storage for real type data can be overwritten.

4. Experiments

4.1 Accuracy Checks

4.1.1 Checking of Eigenvalues after the Block Tridiagonalization

As a test of the numerical error property of the block tridiagonalization, the eigenvalues were calculated and compared for both the original symmetric matrix A and the block tridiagonal matrix T condensed from A by the block Householder method described in this paper.

The eigenvalues of both A and T were computed by the dense symmetric eigensolver NUMPAC routine HOQRUD [13], which uses the standard method of Householder tridiagonalization followed by the QR-iterations, and is written in Fortran 77 and it is modified for this experiment to use higher-precision variables for the inner product accumulations in order to reduce errors.

Table 1 Maximum differences of eigenvalues A and T ($N = 3600$).

b	Frank	Hilbert	random
20	9.3E-10	2.4E-15	1.6E-12
40	5.6E-09	1.3E-15	1.6E-12
60	2.8E-09	8.9E-16	1.4E-12
80	1.9E-09	2.2E-15	3.4E-12
100	2.3E-10	1.6E-15	3.9E-12

Table 2 Maximum differences of all eigenvalues ($N = 3600$).

b	Frank	Hilbert	random
20	8.4E-09	4.7E-15	6.1E-12
40	2.0E-08	4.0E-15	9.0E-13
60	1.6E-08	3.1E-15	8.9E-12
80	8.4E-09	3.1E-15	1.1E-11
100	6.5E-09	4.9E-15	8.1E-13

As test matrices of size $N = 3600$, the Frank matrix $a_{i,j} = N + 1 - \max(i,j)$, the Hilbert matrix $a_{i,j} = 1/(i+j-1)$, and the symmetric matrix of uniform random numbers were used. The maximum differences in magnitude between all corresponding eigenvalues of A and T are tabulated in **Table 1** for the block sizes $b = 20, 40, 60, 80, 100$ used in the block reflectors.

4.1.2 Checking of Eigenvalues Computed by the Bisection

It is not so time consuming to compute all eigenvalues of the (ordinary) tridiagonal matrix of size N which is obtained by Murata's Householder tridiagonalization of the band matrix. Since these eigenvalues are those of the original matrix A, all eigenvalues of the (ordinary) tridiagonal matrix were compared with all eigenvalues of A directly computed by the standard dense matrix eigensolver. The maximum differences between them are tabulated in **Table 2**.

4.1.3 Checking of the Total Eigensolver

To test the property of the numerical error of the total eigensolver, the Frank matrix, the Hilbert matrix, and the random matrix were chosen as test matrices, and the accuracy of the solved eigenpairs was checked by conducting the following tests:

Checked items

(1) Error of the orthonormality of vectors:

The maximum error in the orthonormality err_{orth} of the solved eigenvectors was defined as the largest value of $|\langle v^{(\mu)}, v^{(\nu)}\rangle - \delta_{\mu,\nu}|$, and checked to determine whether it was small

Table 3 Errors of the 100 smallest eigenpairs (Frank matrix, $N = 3600$).

b	err_{orth}	$dmax$	$rmax$
20	1.3E-15	3.2E-10	3.0E-10
40	2.0E-15	3.2E-10	3.8E-10
60	1.2E-15	3.2E-10	2.4E-10
80	1.3E-15	3.2E-10	3.2E-10
100	1.1E-15	3.2E-10	2.3E-10

Table 4 Errors of the 100 largest eigenpairs (Frank matrix, $N = 3600$).

b	err_{orth}	$dmax$	$rmax$
20	1.7E-10	2.1E-09	1.2E-04
40	3.2E-10	1.1E-08	6.9E-04
60	2.8E-10	7.5E-09	6.6E-04
80	6.5E-10	1.9E-09	4.7E-04
100	1.5E-09	8.1E-10	6.2E-04

Table 5 Errors of the 100 smallest eigenpairs (Hilbert matrix, $N = 3600$).

b	err_{orth}	$dmax$	$rmax$
20	4.4E-15	2.2E-16	7.7E-12
40	3.2E-15	3.1E-15	1.4E-11
60	3.6E-15	1.0E-15	2.8E-11
80	3.8E-15	6.7E-16	2.1E-11
100	6.1E-15	1.3E-17	2.2E-11

Table 6 Errors of the 100 largest eigenpairs (Hilbert matrix, $N = 3600$).

b	err_{orth}	$dmax$	$rmax$
20	4.5E-11	4.4E-15	3.1E-12
40	1.3E-10	3.6E-15	6.7E-12
60	1.7E-10	3.8E-15	7.5E-12
80	4.1E-11	3.8E-15	6.5E-12
100	6.9E-11	4.2E-15	8.4E-12

enough.

(2) Comparison of the eigenvalues with those obtained by a standard solver:

The accuracy of the eigenvalues obtained by the present method was examined by a comparison with the eigenvalues of A obtained directly by the routine HOQRUD in NUMPAC [13].

The two sets of eigenvalues were sorted in order, and the distances between $\lambda^{(\mu)}$ and $\widehat{\lambda}^{(\nu)}$ were computed as $|\lambda^{(\mu)} - \widehat{\lambda}^{(\nu)}|$.

For the obtained eigenvalues, the maximum distance was $dmax$.

(3) Residual of the eigenvalue equation:

For each eigenpair (λ, v), the residual of the eigenvalue equation is the vector $r = Av - \lambda v$, and its 2-norm res $= ||r||_2$ was defined. The value of res is the upper bound of the distance between the approximated eigenvalue λ and some true eigenvalue.

$rmax$ was then defined as the maximum values of res among the solved eigenpairs, and checked to determine whether it was sufficiently small.

Checking of results

The Frank matrix, the Hilbert matrix, and the random matrix were chosen as the most typical test matrices for this check. The matrix size N was set to 3600, and 100 eigenpairs from both the smallest and largest eigenvalues were calculated with the block sizes $b = 20, 40, 60, 80, 100$. The test results were as follows:

Frank matrix (Table 3 and Table 4):

For this matrix, the magnitude of the largest element was 3600. The 100 smallest eigenvalues calculated were between about 0.25 and 0.250476, and the 100 largest eigenvalues calculated were between about 1.3E2 and 5.3E6.

- The largest errors of the orthonormality between the solved eigenvectors were under 2.0E-15 for the smallest eigenpairs, and under 1.5E-9 for the largest eigenpairs.
- The largest distance between the calculated eigenvalues and those computed by the standard eigensolver for the dense matrix A was under 3.2E-10 for the smallest eigenpairs, and under 1.1E-8 for the largest eigenpairs.
- The 2-norm of the residual $rmax$ was under 3.8E-10 for the smallest eigenpairs, and under 6.9E-4 for the largest eigenpairs.

Hilbert matrix (Table 5 and Table 6):

For this matrix, the largest element in magnitude was 1. The Hilbert matrix was positive definite but had many eigenvalues very close to zero when the size of the matrix was large. The 100 smallest eigenvalues obtained were indeed very close to zeros, and most of the 100 largest eigenvalues obtained were also very close to zero. In fact, only 32 of the calculated eigenvalues were above 1.0E-15, and the largest eigenvalue was about 2.5.

- The errors of orthonormality between the solved eigenvectors were under 6.1E-15 for the smallest eigenpairs, and under 1.7E-10 for the largest eigenpairs.
- The distances between the solved eigenvalues and those computed by the standard method were under 3.1E-15 for the smallest eigenpairs, and under 4.4E-15 for the largest eigenpairs.
- The 2-norm of the residual $rmax$ was under 2.8E-11 for the smallest eigenpairs, and under 8.4E-12 for the largest eigenpairs.

Random matrix (Table 7 and Table 8):

For this matrix, the magnitude of the largest

Table 7 Errors of the 100 smallest eigenpairs (random matrix, $N = 3600$).

b	err_{orth}	$dmax$	$rmax$
20	8.9E-16	9.3E-13	2.8E-05
40	1.0E-15	7.5E-13	3.0E-05
60	1.1E-15	8.3E-13	8.6E-06
80	1.6E-15	8.8E-13	9.2E-06
100	1.3E-15	8.1E-13	1.5E-04

Table 8 Errors of the 100 largest eigenpairs (random matrix, $N = 3600$).

b	err_{orth}	$dmax$	$rmax$
20	1.4E-14	1.1E-12	5.9E-04
40	8.7E-15	1.1E-12	3.3E-05
60	1.3E-14	5.2E-12	1.3E-05
80	7.5E-14	6.6E-12	3.5E-05
100	9.1E-15	5.0E-12	2.0E-04

element was about 1.0. The magnitude of the largest eigenvalue was about -3.4E1 for the 100 smallest eigenpairs, and about 1.8E3 for the 100 largest eigenpairs.

- The errors of orthonormality between the solved eigenvectors were under 1.6E-15 for the smallest eigenpairs, and under 7.5E-14 for the largest eigenpairs.
- The distances between the calculated eigenvalues and those by the standard method were under 9.3E-13 for the smallest eigenpairs, and under 6.6E-12 for the largest eigenpairs.
- The 2-norm of the residual $rmax$ was under 1.5E-4 for the smallest eigenpairs, and under 5.9E-4 for the largest eigenpairs.

4.2 Timing Measurement

The system environment of this experiment was as follows:

CPU: Intel Pentium 4 2.6C (Northwood, HT) with SSE2, L1 data cache 8 Kbytes, L2 cache 512 Kbytes. The maximal performance of the Intel Pentium 4 2.6C with SSE2 in IEEE double-precision computation is 2 floating point operations per CPU clock, which is 5.2 Gflops, since the clock frequency of the CPU is 2.6 GHz. (Note that 5.2 Gflops is just an ideal value, for example, when there is no access to main memory, no branch effect, etc.)

Main memory: 4 pieces of DDR-3200 type 512 Mbyte memory modules with a total capacity of 2 Gbytes in dual-channel mode.

Compiler: The Intel Fortran compiler v9.0.031 for Linux IA32 was used with the option "-O3 -xN".

Level-3 BLAS routines [5]: Intel MKL (Math Kernel Library) version 8.0 for IA32. (In the block Householder tridiagonalization kernel, to compute HAH^T for the given U and $H = I - 2UU^T$, the level-3 BLAS routines DGEMM, DSYMM and DSYR2K were used.)

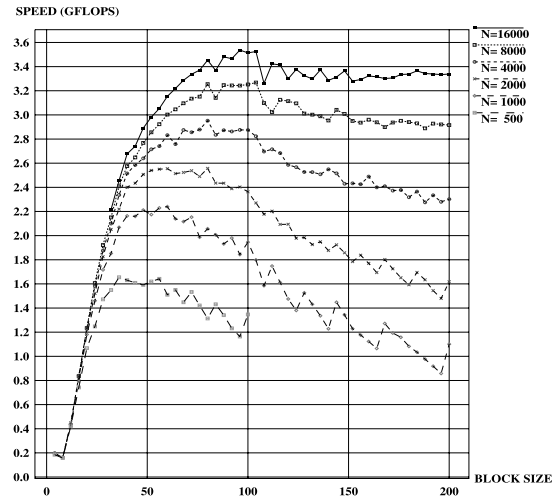


Fig. 1 Speed of block tridiagonalization.

SVD routine: "SVDD" in NUMPAC (Nagoya University Mathematica PACage) [13] was used. The algorithm that SVDD uses is the Golub-Reinsch SVD [10]. The original code of SVDD is 205 lines in Fortran 77. The flop count of Golub-Reinsch SVD is about $21s^3$ for a square matrix of size s, where $s = b$ or $s = r < b$ (§5.4.5 of Golub and van Loan [11]).

In experiments, a symmetric random matrix (with a uniform random number in the unit interval) was used to measure the elapsed time needed to solve an eigenproblem.

In the experiments described in this paper,

- **Figure 1** shows the "equivalent speed", which is defined by "as if" the block tridiagonalization of a symmetric matrix of size N were $(4/3)N^3$ floating point operations. This equivalent speed in Gflops is shown on the vertical axis, and the block size b is shown on the horizontal axis by steps from 4 up to 200. The graph has six plotted curves corresponding to matrix sizes of $N = 500$, 1000, 2000, 4000, 8000, and 16000. As we increased the block size, the observed speed increased linearly in the region where the value of b was small, then saturated, and finally decreased. If b was fixed, then the larger the value of N, the faster the speed.
  This graph shows that about 60% to 70% of the peak performance of Pentium 4 2.6C (the theoretical peak is 5.2 Gflops) was ob-

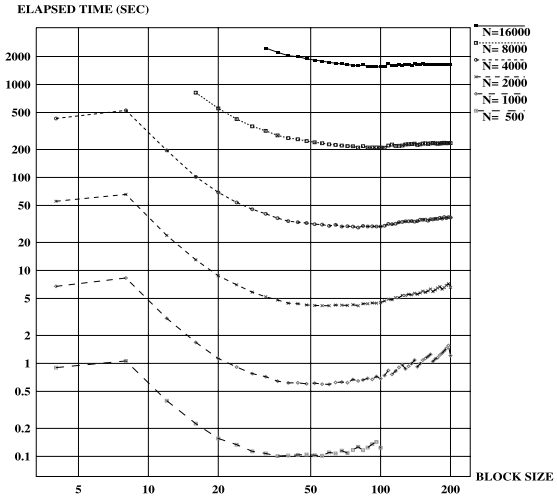


Fig. 2 Elapsed time of block tridiagonalization.

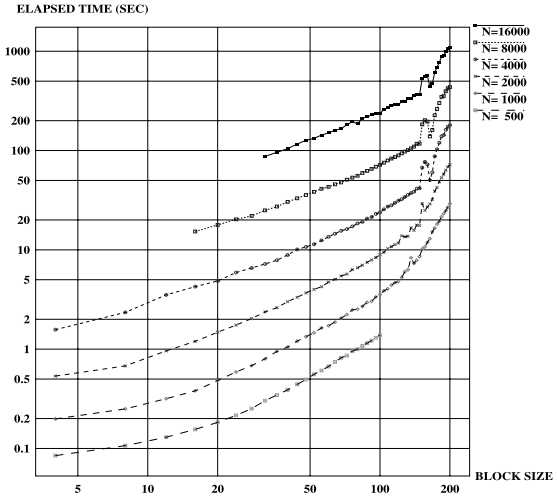


Fig. 3 Elapsed time for solution of eigenpairs of the block tridiagonal ($\ell = 100$).

tained when the block size was 50 or more and the matrix size was 1000 or more. Most of the computation of the block rank-2 update kernel in the block Householder was rich in level-3 BLAS computations, and this measurement was close to being a performance test of level-3 BLAS routines used for size b matrices.

- **Figure 2** is a graph of the elapsed time of the block tridiagonalization on the vertical axis, and the block size on the horizontal axis. Both axe are labeled with logarithmic values and the graph is plotted for various values of N.
- **Figure 3** is a graph showing the elapsed time for the 100 solutions of eigenpairs on the vertical axis, and the block size on the horizontal axis. Both axe are labeled with logarithmic values and graph is plotted for various values of N. This elapsed time includes the transformation from a block tridiagonal matrix into a band matrix, the band eigensolver of 100 eigenvalues and vectors, and the back transformation of 100 eigenvectors from a band matrix into a block-tridiagonal matrix.

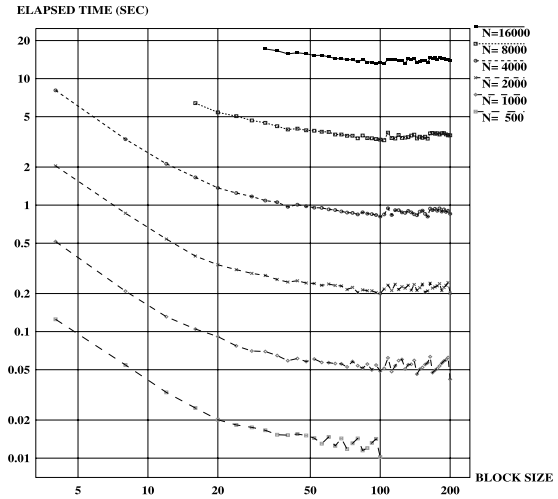


Fig. 4 Elapsed time for block backward transformation of eigenvectors ($\ell = 100$).

- **Figure 4** is a graph showing the elapsed time of the block backward transformation of 100 eigenvectors from those of the block tridiagonal matrix already obtained by the previous procedure into those of the original dense symmetric matrix, and the block size on the horizontal axis. Both axe are labeled with logarithmic values, and the graph is plotted for various values of N.
- **Figure 5** is a graph showing the total elapsed time for the solution of 100 eigenpairs of the original dense symmetric random matrix of size N by the implemented method combined with the current method, and the block size on the horizontal axis . Both axe are labeled with logarithmic values, and the graph is plotted for the various values of N. For each value of N, the best block sizes for minimizing the total elapsed time are as follows: $b = 20$ for $N = 500$, $b = 28$ for $N = 1000$, $b = 36$ for $N = 2000$, $b = 52$ for $N = 4000$, $b = 80$ for $N = 8000$, and $b = 80$ for $N = 16000$.

From the above graphs of our experiments:

- **Block tridiagonalization:** The elapsed

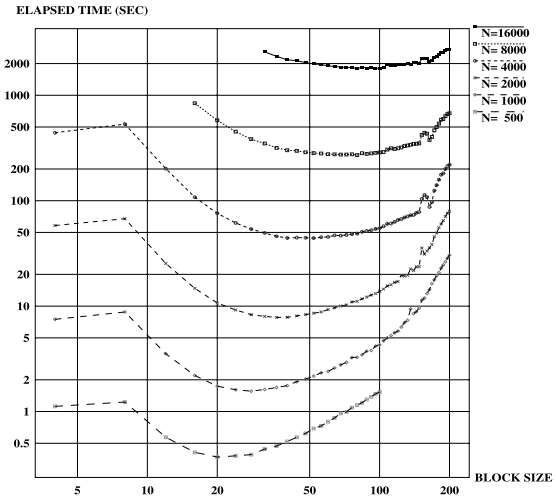


Fig. 5 Total elapsed time for solution of eigenpairs of a dense matrix ($\ell = 100$).

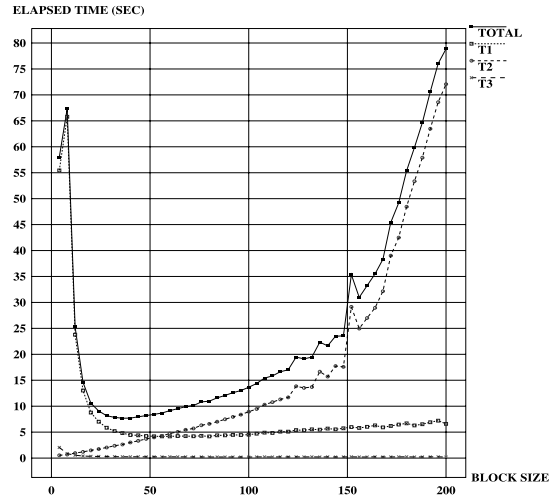


Fig. 7 Elapsed time of three steps ($N = 2000, \ell = 100$).

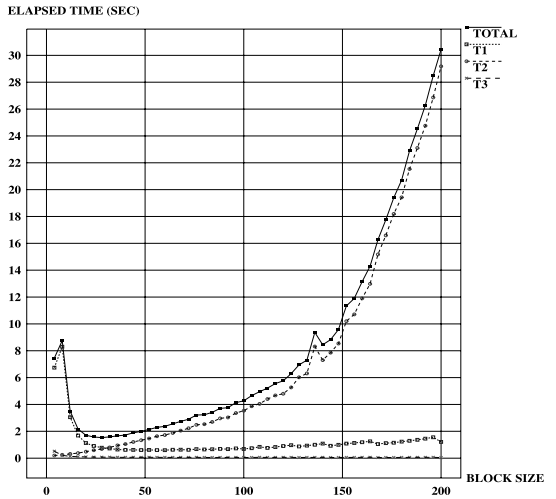


Fig. 6 Elapsed time of three steps ($N = 1000, \ell = 100$).

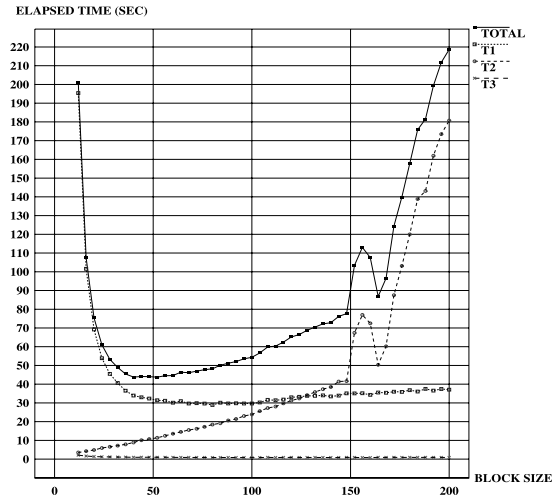


Fig. 8 Elapsed time of three steps ($N = 4000, \ell = 100$).

time is almost proportional to N^3, and is nearly inversely proportional to b for the region in which the value of b is small. The value of b that minimizes the elapsed time of this step seems to be proportional to $N^{1/2}$ according to the graph. Above that values, the elapsed time grows. (In the next subsection, the reason for this is considered.)

- **Block tridiagonal eigensolver:** The elapsed time needed to obtain the value ℓ of eigenpairs by using the band eigensolver with LU-decomposition inverse iteration with selective reorthogonalization method seems to be proportional to $N^2 b$, according to the graph.
- **Backward transform:** The elapsed time for the backward transformation of ℓ eigenvectors seems to be proportional to N^2, according to the graph. It is almost inversely proportional to b for the region in which the value of b is small.

In the graph for each matrix size, **Fig. 6** for $N = 1000$, **Fig. 7** for $N = 2000$, **Fig. 8** for $N = 4000$, **Fig. 9** for $N = 8000$, and **Fig. 10** for $N = 16000$, the elapsed times are plotted for three major steps: "block tridiagonalization" (T1), "block tridiagonal eigensolver" (T2), "backward transform" (T3), and their total time (TOTAL). (T1 = BHSHLD; T2 = BT2BND + (TRIDIA + BISECT + INVITR) + BND2BT; T3 = BCKTRS.)

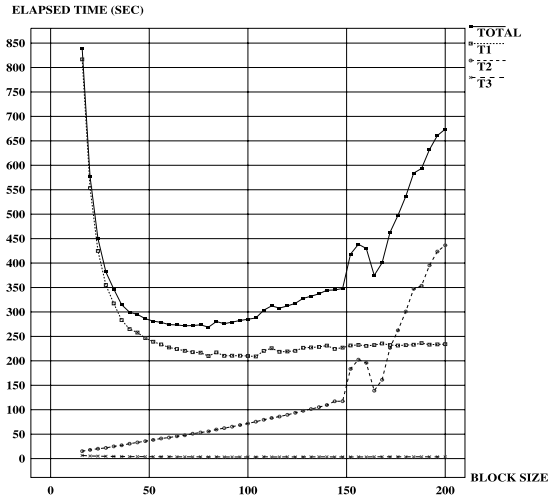


Fig. 9 Elapsed time of three steps ($N = 8000, \ell = 100$).

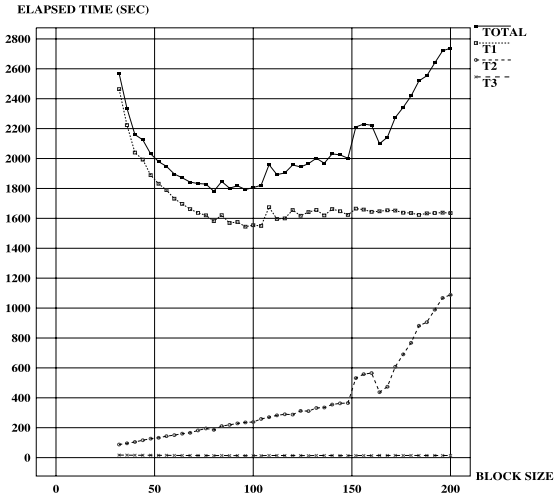


Fig. 10 Elapsed time of three steps ($N = 16000, \ell = 100$).

For all these timings, ℓ, the number of eigenpairs to be solved, is fixed at 100.

The curve of (T3) is always at the bottom and almost negligible when N becomes large for fixed $\ell = 100$. The curve of (T2) is initially below the plot of (T1) for smaller block sizes, but for larger block sizes it increases and tends to go above (T1). The curve of (TOTAL) is always at the top, since it is the sum of the others. Note that (T1) is independent of ℓ, (T3) is proportional to ℓ, and (T2) contains the process of inverse iterations, which is proportional to ℓ.

In **Table 9**, when the number of eigenpairs to be computed is fixed at $\ell = 100$, the best block sizes $b = 20$, 28, 36, 52, 80, and 80 are chosen to minimize the total elapsed time for each matrix size of $N = 500$, 1000, 2000, 4000, 8000, and 16000. (Note that the best block size depends on the value of ℓ.) In that combination, the elapsed times measured for each seven steps in the total computation of the eigensolutions are tabulated.

Table 9 Elapsed time for the calculation steps (in seconds).

N	500	1000	2000	4000	8000	16000
b	20	28	36	52	80	80
BHSHLD	0.156	0.777	4.808	31.43	209.76	1581.59
BT2BND	0.003	0.011	0.034	0.13	0.59	1.17
TRIDIA	0.032	0.194	1.238	6.83	38.05	150.73
BISECT	0.040	0.073	0.137	0.26	0.50	0.99
INVITR	0.103	0.396	1.188	4.11	16.45	32.49
BND2BT	0.003	0.006	0.015	0.03	0.08	0.17
BCKTRS	0.020	0.070	0.258	0.96	3.38	13.57
TOTAL	0.359	1.529	7.683	43.75	268.83	1780.74

Table 10 Summary of the source code in this experiment.

Source file	Function of code	Lines
MAIN	Main program for this experiment	143
GENMAT	Test matrix generator	124
BHSHLD	Total forward block H-transform	61
MKBREF	Construction of block reflector	133
HQRORT	H-QR orthogonalization (for the block vector)	263
SVDD	NUMPAC routine SVDD for SVD	334
BREFLECT	Block reflection kernel	102
STEP2	Eigensolver for block tridiagonal	294
BT2BND	Reduce bandwidth of the block tridiagonal to band matrix, and later vector backward transform	285
MUHAUS	Murata's band eigensolver (TRIDIA+BISECT+INVITR)	559
BCKTRS	Total backward block H-transform	64
	sub total in the above	2362
REPORT	Report routines about the run	107
CHECKS	Routines for checking	628
HOQRVD	NUMPAC routine HOQRVD/UD for checking (standard eigensolver)	200
	grand total	3297

4.3 Code Used in the Experiment

In this experiment, all the code used was written in Fortran 90 except for the BLAS3 routines (Intel MKL), which were supplied in the form of binary objects. The total size of the source code, including comments and macro definitions, is about 3300 lines, and if the routines from NUMPAC and Murata's book are excluded, it is about 2200 lines. (Note that the routines from NUMPAC and Murata's in books written in Fortran 77 were restructured into Fortran 90 syntax and then used.) A summary of the source files with short explanations is shown in **Table 10**. The source code of the block reflection kernel written using BLAS3

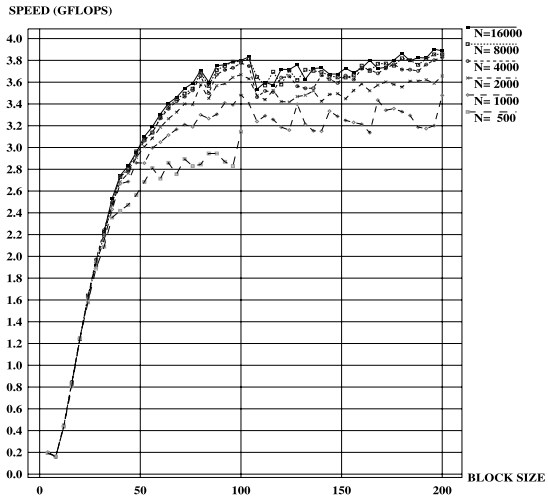


Fig. 11 Speed of block tridiagonalization (dummy U is used).

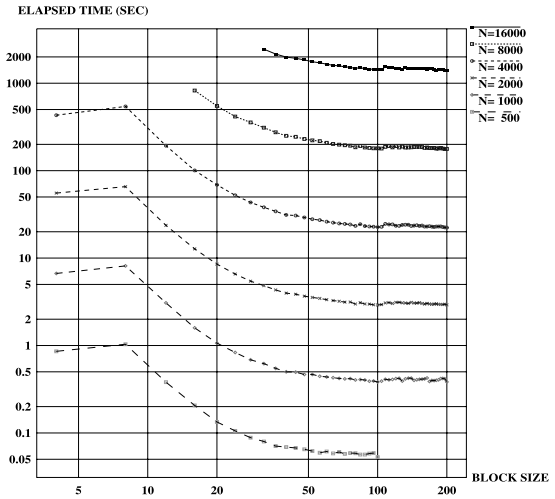


Fig. 12 Elapsed time of block tridiagonalization (dummy U is used).

routines is shown in **Fig. 13** as example. The OpenMP directives lines are left In the sample source code, which is for SMP machines, and not for the single-CPU (Pentium 4 2.6C) system used in this experiment.

4.4 Note on Performance Degradation

When the block size b becomes larger than about $O(N^{1/2})$, the performance of the block tridiagonalization decreases, as can be seen in the graph of the experiment (Fig. 1).

In the current implementation used in the experiment, the ordinary Householder-QR method (with column pivoting, and backward accumulation to generate an orthonormal basis explicitly) [18], (§5.1.6 of Golub and van Loan [11]) is used for the orthogonal decomposition of the matrix C in STEP-1 of the algorithm in this paper, which by nature lacks locality of memory reference and can be very slow to compute when the matrix C is large. (STEP-2 and STEP-3, and also STEP-4, have higher memory reference locality as a result of the matrix multiplication.)

To show that the performance degradation in the current implementation results from the computation of block Householder vectors, a controlled experiment was conducted in which the elapsed time was measured with the following single change: at each stage of block transformation, the block Householder vector U was merely copied from a generic dummy block vector without computation, and HAH was computed as if U were the correct block Householder vector. Thus, from the total elapsed time of block tridiagonalization the elapsed time needed to construct U was completely excluded.

As before, **Fig. 11** shows the "equivalent speed" and **Fig. 12** shows the "elapsed time" of the block tridiagonalization without the construction of block Householder vectors. The behavior of the graphs for this experiment is completely different from that of the graphs for the original experiment (Fig. 1 and Fig. 2). In the new graphs, it can be recognized that the "equivalent speed" does not decrease for the larger block sizes. By this simple comparison, it can be understood that the performance degradation for the large block sizes seen in Fig. 1 results mainly from the construction of the block Householder vectors.

If the ordinary Householder-QR is used for the orthogonal decomposition in the construction of the block Householder vector, to complete the block tridiagonalization of a matrix of size N, the flop count of the two-sided block transformations is $O(N^3)$; on the other hand, it can be shown that the flop count of the construction of the block Householder vectors using the Householder-QR method is $O(N^2b)$. Therefore, as long as $b \ll N$ holds, the construction of the block Householder vector makes a negligible contribution to the flop count.

However, to complete the block tridiagonalization, the number of memory accesses made by the two-sided Householder block transformations is $O(N^3/b)$. On the other hand, the number of accesses for the Householder-QR orthogonal decomposition to construct the block Householder vectors is $O(N^2b)$. These two or-

ders of memory access meet when $b = O(N^{1/2})$.

This indicates that if the Householder-QR method is used for the orthogonal decomposition in the construction of the block reflector, the performance of the block tridiagonalization is highly degraded when a value as large as $O(N^{1/2})$ or more is used for the block size b.

Therefore, for a larger block size b, some other orthogonal decomposition algorithm must be used rather than the ordinary Householder-QR method. It should have the highest locality of memory reference and also numerical error properties as good as those of the ordinary Householder-QR method.

The block Householder-QR factorization (§5.2.2 of Golub and van Loan [11]) [4] would be a method of choice. With some good block size b' ($b' < b$), the memory access could be reduced by a factor of b' and would run faster if a few blocks of b' vectors could be held in fast memory. The block Householder reflector described in this paper could also be used to make block version of Householder-QR factorization, including the explicit formation of the basis.

Another completely different possibility would be to use the method described in Stathopoulos and Wu [16], which is based on the classical SVD with orthonormality corrections, is rich in matrix-matrix multiplications of size b, and can be implemented using the level-3 BLAS routines DGEMM and DSYRK.

5. Discussion

Recently, the potential speed of CPU chips has been greatly enhanced by many advances in hardware architecture. However, the storage transfer speed into and out of a CPU chip is much slower than that inside the CPU. For large-scale computation, storage transfer is usually a bottleneck, especially when the size of local storage on the chip is insufficient, as it always is for the large computations. In such situations, block algorithms usually help to avoid performance degradation in the computation. Even a cheap Intel Pentium 4 2.6C is able to transform a symmetric matrix of size $16,000$ stored in the main memory into block tridiagonal form within 2000 seconds.

In this paper, we have demonstrated the benefits of the block Householder method only for the eigenproblem of a dense symmetric matrix. However, other algorithms that use the Householder transformation could also be used.

6. Conclusion

A block Householder transformation was implemented and tested for the eigenproblem of block-tridiagonalizing a dense symmetric matrix. The block algorithm approach is effective for solving large problems. In comparison with non-blocked algorithms, the locality of storage reference is enhanced by a factor of b, where b is the block size. This enhancement of the locality greatly reduces the amount of storage transfer, and enables the calculation to be performed in shorter elapsed time.

If we define the block tridiagonalization process of size-N symmetric matrix as $(4/3)N^3$ floating point operations, an "equivalent speed" can be defined. In the experiment, for the case of a matrix of size $N = 16000$, the block tridiagonalization can be performed at an "equivalent speed" of about 3.5 Gflops by a Pentium 4 2.6C system whose "theoretical" peak performance is 5.2 Gflops. The Intel MKL 8.0 for IA32 is used for the level-3 BLAS routines.

In our current implementation of the block tridiagonalization process for a very large block size of $O(N^{1/2})$ or more, clearly a performance degradation is observed. The reason is that, inside the current implementation of the construction of the block reflector, the ordinary Householder-QR method is used for orthogonal decomposition in STEP-1 of the algorithm. Although the flop count in the Householder-QR method is negligible, since the ordinary Householder-QR method lacks locality of memory reference, the number of storage accesses is not negligible but is comparable to the other parts of the computation. This explains why the performance decreases for large block sizes of more than $b = O(N^{1/2})$. To overcome this problem, it is necessary to use a better orthogonal decomposition method that is not only accurate but also has the highest locality of storage reference. One candidate could be a block Householder-QR factorization method that uses some block Householder transformation (for example, WY^T, YTY^T, and also the block reflector itself).

References

1) Bischof, C.H.: A Summary of Block Schemes for Reducing a General Matrix to Hessenberg Form, Technical Report ANL/MCS-TM-175, Argonne National Laboratory (1993).
2) Bischof, C.H. and Lang, B.: The SBR Tool-

box: Software for Successive Band Reduction, *ACM Trans. Math. Software*, Vol.26, No.4, pp.602–616 (2000). Also Preprint, ANL/MCS-P587-0496, Argonne National Laboratory (1996).
3) Bischof, C.H., Lang, B. and Sun, X.: A Framework for Symmetric Band Reduction, *ACM Trans. Math. Software*, Vol.26, No.4, pp.581–601 (2000). Also Preprint, ANL/MCS-P586-0496, Argonne National Laboratory (1996).
4) Bischof, C.H. and van Loan, C.: The WY Representation for Products of Householder Matrices, *SIAM J. Sci. Stat. Comput.*, Vol.8, No.1, pp.2–13 (1987).
5) Dongarra, J.J., Croz, J.D., Hammarling, S. and Duff, L.S.: A Set of Level 3 Basic Linear Algebra Subprograms, *ACM Trans. Math. Software*, Vol.16, No.1, pp.1–17 (1990).
6) Dongarra, J.J., Hammarling, S.J. and Sorensen, D.C.: LAPACK Working Note #2; Block Reduction of Matrices to Condensed Forms for Eigenvalue Computations, Technical Report ANL/MCS-TM-99, Argonne National Laboratory (1987).
7) Dongarra, J.J. and Sorensen, D.C.: Block Reduction of Matrices to Condensed Forms for Eigenvalue Computations, *J. Comput. Appl. Math.*, Vol.27, pp.215–227 (1989).
8) Gansterer, W.N., Schneid, J. and Ueberhuber, C.W.: A Divide-and-Conquer Method for Symmetric Banded Eigenproblems; Part-I: Theoretical Results, Technical Report AURORA TR1999-12, Vienna Univ. of Tech. (1998).
9) Gansterer, W.N., Ward, R.C. and Muller, R.P.: An Extension of the Divide-and-Conquer Method for a Class of Symmetric Block-Tridiagonal Eigenproblems, *ACM Trans. Math. Software*, Vol.28, No.1, pp.45–58 (2002).
10) Golub, G.H. and Reinsch, C.: Singular Value Decomposition and Least Square Solutions, *Numerische Mathematik*, Vol.14, pp.403–420 (1970).
11) Golub, G.H. and van Loan, C.F.: *Matrix Computations*, 3rd Ed., The Johns Hopkins Univ. Press (1983).
12) Murata, K. and Horikoshi, K.: A New Method for the Tridiagonalization of the Symmetric Band Matrix, *Information Processing in Japan*, Vol.15, pp.108–112 (1975).
13) Ninomiya, I., et al.: NUMPAC (Nagoya University Mathematical PACkage), NetNUMPAC Guide Page.
http://netnumpac.fuis.fukui-u.ac.jp
14) Schreiber, R. and Parlett, B.: Block Reflectors: Theory and Computation, *SIAM J. Numer. Anal.*, Vol.25, No.1, pp.189–205 (1988).
15) Schreiber, R. and van Loan, C.: A Storage Efficient YW Representation for Products of Householder Transformations, *SIAM J. Sci. Stat. Comput.*, Vol.10, No.1, pp.53–57 (1989).
16) Stathopoulos, A. and Wu, K.: A Block Orthogonalization Procedure with Constant Synchronization Requirements, *SIAM J. Sci. Comput.*, Vol.23, pp.2165–2182 (2002).
17) Wilkinson, J.H. and Reinsch, C., e.: *Handbook for Automatic Computation, Vol.2, Linear Algebra*, Springer-Verlag, New York (1971).
18) 村田健郎：線形代数と線形計算法序説，サイエンス社 (1986).
Murata, K.: Traslation of Japanese title: "Introduction to the Linear Algebra and Computation Method" (in Japanese), Science Corp., Tokyo, Japan (1986).
19) 村田健郎，小国　力，唐木幸比古：スーパーコンピュータ 科学技術計算への適用，丸善 (1985).
Murata, K. and Oguni, T. and Karaki, Y.: *Supercomputers and Their Applications for Scientific & Engineering Computations* (in Japanese), Maruzen, Tokyo, Japan (1985).
20) 山本有作：キャッシュマシン向け三重対角化アルゴリズムの性能予測方式，*IPSJ SIG Technical Reports*, Vol.2005, No.81, pp.25–30 (2005).
Yamamoto, Y.: Performance Predicition of a Tridiagonalization Algorithm for Cache-based Microprocessors (in Japanese), *IPSJ SIG Technical Reports*, Vol.2005, No.81, pp.25–30 (2005).
21) 山本有作：キャッシュマシン向け対称密行列固有値解法の性能・精度評価，情報処理学会論文誌 ACS, Vol.46, No.SIG3(ACS8), pp.81–91 (2005).
Yamamoto, Y.: Performance and Accuracy of Algorithms for Computing the Eigenvalues of Real Symmetric Matrices on Cache-based Microprocessors (in Japanese), *IPSJ Advanced Computing Systems* (*ACS*), Vol.46, No.SIG3(ACS8), pp.81–91 (2005).

7. Addenda

After this paper was accepted, it was found that the set of solved eigenvectors of the band matrix by the routine "INVITR", when the size of matrix was large, had a tendency to contain some eigenvectors that had large errors in orthonormality or residual. The errors can be much reduced by the following modification to "INVITR". The part of the original below.

```
------------------------------------------
    DO 360 K=1,N
       KMAX=IWK(K)
       SUM=-V(KMAX,JJ)
       V(KMAX,JJ)=V(K,JJ)
       JS=MAX0(1,K-NLD)
       IF (JS .GT. (K-1)) GO TO 350
```

Table 11 Errors of the 100 smallest eigenpairs (Frank matrix, $N = 3600$).

b	err_{orth}	$rmax$
20	1.8E-15	3.1E-10
40	1.8E-15	3.7E-10
60	1.6E-15	2.2E-10
80	1.9E-15	2.9E-10
100	1.8E-15	2.8E-10

```
      MMK=M-K
      DO 340 J=JS,K-1
340      SUM=SUM+WK(MMK+J,K)*V(J,JJ)
350   V(K,JJ)=-SUM
360 CONTINUE
```

was replaced to as below.

```
    DO 360 K=1,N
      KMAX=IWK(K)
      IE=MIN0(K+NLD, N)
      T=-V(KMAX,JJ)
      V(KMAX,JJ)=V(K,JJ)
      V(K,JJ)=-T
      IF ((K+1) .GT. IE) GO TO 360
      DO 340 I=K+1,IE
340      V(I,JJ)=V(I,JJ)+T*WK(M-I+K,I)
360 CONTINUE
```

The minimum number of inverse iterations for each eigenvector was also raised from once to twice to get higher accuracy. For this, another line in the original "INVITR" below

```
IF (VNORM .GE. 1.0D0) GO TO 380
```

was replaced to as below.

```
IF(VNORM.GE.1.D0.AND.ITER.GE.2) GOTO 380
```

The above modification reduced the errors of eigenvectors of the symmetric band matrix B, and also reduced the errors of eigenvectors of the original dense symmetric matrix A.

To show that the eigenvectors were much improved by the modification, the maximum errors of orthonormality err_{orth} and the maximum 2-norm of the residual $rmax$ were computed for the Frank matrix, Hilbert matrix, and random matrix (symmetric, uniform random) of size $N = 3600$, for the block sizes $b = 20$, 40, 60, 80 and 100.

- For the Frank matrix of 100 smallest eigenpairs, there is not so much difference between Table 3 and **Table 11**.
- For the Frank matrix of 100 largest eigenpairs, there is much difference between Table 4 and **Table 12**. The maximum error of orthonormality err_{orth} is much improved from the level of about 1E-10 to about 1E-15, and the maximum 2-norm of residual $rmax$ is also much improved from the level of about 1E-4 to about 1E-8.
- For the Hilbert matrix of 100 smallest eigenpairs, there is not so much differences between Table 5 and **Table 13**.
- For the Hilbert matrix of 100 largest eigenpairs, there is much difference between Table 6 and **Table 14**. The err_{orth} is much improved from the level of about 1E-10 to about 1E-13. The $emax$ is also much improved from the level of about 1E-11 to about 1E-14.
- For the random matrix of 100 smallest eigenpairs, there is much difference between Table 7 and **Table 15**. The err_{orth} is almost same level, but $rmax$ is much improved from the level of about 1E-5 to

Table 12 Errors of the 100 largest eigenpairs (Frank matrix, $N = 3600$).

b	err_{orth}	$rmax$
20	2.1E-15	5.8E-09
40	2.8E-15	1.4E-08
60	1.6E-15	1.0E-08
80	2.0E-15	6.6E-09
100	3.3E-15	9.0E-09

Table 13 Errors of the 100 smallest eigenpairs (Hilbert matrix, $N = 3600$).

b	err_{orth}	$rmax$
20	2.6E-14	3.0E-12
40	2.3E-14	2.9E-12
60	1.9E-14	1.7E-12
80	1.1E-14	2.7E-12
100	2.5E-14	2.0E-12

Table 14 Errors of the 100 largest eigenpairs (Hilbert matrix, $N = 3600$).

b	err_{orth}	$rmax$
20	5.0E-14	9.0E-15
40	4.8E-14	6.7E-15
60	1.9E-14	8.7E-15
80	5.8E-14	7.4E-15
100	3.7E-14	7.8E-15

Table 15 Errors of the 100 smallest eigenpairs (random matrix, $N = 3600$).

b	err_{orth}	$rmax$
20	7.1E-15	4.1E-12
40	4.8E-15	4.6E-12
60	4.4E-15	4.6E-12
80	5.8E-15	6.3E-12
100	3.7E-15	5.8E-12

```
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! SAMPLE CODE OF BLOCK REFLECTION: H A H^{T} -> A, WHERE H = I - 2 U U^{T}.
! COPYRIGHT NOTICE: THIS CODE CAN BE FREELY COPIED,MODIFIED,OR DISTRIBUTED.
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SUBROUTINE BREFLECT ( KK, BLKSZ, NBLK, A, RANK, U, P, BS )
IMPLICIT NONE
INTEGER,INTENT(IN) :: KK ! FOR THE KK-TH STEP OF THE BLOCK H-TRANSFORM.
INTEGER,INTENT(IN) :: BLKSZ, NBLK ! SIZE OF BLOCK AND NUMBER OF BLOCKS.
REAL(8),INTENT(INOUT) :: A(BLKSZ,BLKSZ,(NBLK+1)*NBLK/2)
   ! HOLDS ONLY THE LOWER TRIANGULAR PART OF THE BLOCK SYMMETRIC MATRIX A.
   ! FOR IB >= JB, THE (IB,JB)-TH BLOCK IS STORED IN A(:,:,INDX(IB,JB)).
   ! FOR IB == JB, ONLY THE LOWER TRIANGULAR PART OF DIAGONAL BLOCK
   !               IS STORED IN A(:,:,INDX(IB,IB)).
INTEGER,INTENT(IN) :: RANK ! THE EFFECTIVE RANK OF BLOCK VECTOR U.
REAL(8),INTENT(IN) :: U(BLKSZ,BLKSZ,NBLK) ! BLOCK H-VECTOR. (READ-ONLY)
   ! THE PART U(:,:,1:KK) WILL NOT BE ACCESSED BY THIS ROUTINE,
   ! AND MAY HOLD ALP(1),...,ALP(KK) ALREADY.
REAL(8)            :: P(BLKSZ,BLKSZ,NBLK) ! USED AS WORK AREA.
   ! THE PART P(:,:,1:KK) WILL NOT BE ACCESSED INSIDE THIS ROUTINE,
   ! AND MAY HOLD BET(1),...,BET(KK) ALREADY.
INTEGER,INTENT(IN) :: BS(NBLK) ! THE ARRAY OF THE SIZE OF EACH BLOCK.
   ! BS(1),...,BS(NBLK-1) ARE SET TO BLKSZ, AND BS(NBLK) IS SET TO
   ! N - BLKSZ * (NBLK-1), WHICH IS THE SIZE OF THE LAST (FRAGMENT) BLOCK.
!==========================================================================
EXTERNAL DSYMM, DGEMM, DSYR2K  ! LEVEL-3 BLAS ROUTINES.
REAL(8) G(RANK,RANK), TMP(BLKSZ,BLKSZ)
INTEGER IB, JB, INDX ;  INDX(IB,JB) = IB + (JB - 1) * (2 * NBLK - JB) / 2
  ! STATEMENT FUNCTION OF THE INDEX TO THE (IB,JB)-TH MATRIX BLOCK OF THE
  ! SYMMETRIC BLOCK MATRIX STORED USING THE SYMMETRY. IB >= JB IS ASSUMED.
!==========================================================================
     IF (RANK == 0) RETURN ! DO NOTHING FOR THE NULL TRANSFORMATION.
!--------------------------------------------------------------------------
! COMPUTE: A * U ---> P.
!$OMP PARALLEL DO
      DO  IB = KK+1, NBLK
         P(:,1:RANK,IB) = 0.D0
      ENDDO
!$OMP END PARALLEL DO
      DO  JB = KK+1, NBLK
         ! FOR DIAGONAL BLOCK (IB == JB)
         CALL DSYMM ('L', 'L', BS(JB), RANK, 1.D0,                       &
                 A(1,1,INDX(JB,JB)), SIZE(A,1), U(1,1,JB), SIZE(U,1),    &
                 1.D0, P(1,1,JB), SIZE(P,1))
         ! FOR OFF-DIAGONAL BLOCKS. (IB > JB)
!$OMP PARALLEL DO PRIVATE(TMP) SHARED(RANK,BLKSZ)
         DO  IB = JB+1, NBLK
            CALL DGEMM ('N', 'N', BS(IB), RANK, BS(JB), 1.D0,            &
                    A(1,1,INDX(IB,JB)), SIZE(A,1), U(1,1,JB), SIZE(U,1), &
                    1.D0, P(1,1,IB), SIZE(P,1))
            CALL DGEMM ('T', 'N', BS(JB), RANK, BS(IB), 1.D0,            &
                    A(1,1,INDX(IB,JB)), SIZE(A,1), U(1,1,IB), SIZE(U,1), &
                    0.D0, TMP, SIZE(TMP,1))
!$OMP CRITICAL
```

```
054|             P(:BS(JB),1:RANK,JB) = &                                          |
055|                     P(:BS(JB),1:RANK,JB) + TMP(:BS(JB),1:RANK)               |
056| !$OMP END CRITICAL                                                           |
057|          ENDDO                                                              |
058| !$OMP END PARALLEL DO                                                       |
059|       ENDDO                                                                 |
060| !-------------------------------------------------------------------------- |
061| ! COMPUTE: (U^{T} * P) --> G. NOTE, G IS SYMMETRIC SINCE G = U^{T} A U.     |
062|       G(1:RANK,1:RANK) = 0.D0                                               |
063| !$OMP PARALLEL DO PRIVATE(TMP) SHARED(RANK,BLKSZ)                           |
064|       DO  IB = KK+1, NBLK                                                   |
065|          CALL DGEMM ('T', 'N', RANK, RANK, BS(IB), 1.D0,      &             |
066|                  U(1,1,IB), SIZE(U,1), P(1,1,IB), SIZE(P,1),  &             |
067|                  0.D0, TMP, SIZE(TMP,1))                                    |
068| !$OMP CRITICAL                                                              |
069|          G(1:RANK,1:RANK) = G(1:RANK,1:RANK) + TMP(1:RANK,1:RANK)           |
070| !$OMP END CRITICAL                                                          |
071|       ENDDO                                                                 |
072| !$OMP END PARALLEL DO                                                       |
073| !-------------------------------------------------------------------------- |
074| ! COMPUTE: (P - U * G) * (-2) --> P. NOTE, THE SYMMETRY OF G IS USED.       |
075| !$OMP PARALLEL DO SHARED(BLKSZ,RANK)                                        |
076|       DO  IB = KK+1, NBLK                                                   |
077|          CALL DSYMM ('R', 'L', BS(IB), RANK, -1.D0, &                       |
078|             G, SIZE(G,1), U(1,1,IB), SIZE(U,1), 1.D0, P(1,1,IB), SIZE(P,1)) |
079|          P(:BS(IB),1:RANK,IB) = P(:BS(IB),1:RANK,IB) * (-2.D0)             |
080|       ENDDO                                                                 |
081| !$OMP END PARALLEL DO                                                       |
082| !-------------------------------------------------------------------------- |
083| ! COMPUTE: A + U * P^{T} + P * U^{T} ---> A.                                |
084|       DO  JB = KK+1, NBLK                                                   |
085|          ! FOR DIAGONAL BLOCK (IB == JB) :                                  |
086|          CALL DSYR2K ('L', 'N', BS(JB), RANK, 1.D0,           &             |
087|                  U(1,1,JB), SIZE(U,1), P(1,1,JB), SIZE(P,1),  &             |
088|                  1.D0, A(1,1,INDX(JB,JB)), SIZE(A,1))                       |
089|          ! FOR OFF-DIAGONAL BLOCK (IB > JB) :                               |
090| !$OMP PARALLEL DO SHARED(BLKSZ,RANK)                                        |
091|          DO  IB = JB+1, NBLK                                                |
092|             CALL DGEMM ('N', 'T', BS(IB), BS(JB), RANK, 1.D0,    &          |
093|                     U(1,1,IB), SIZE(U,1), P(1,1,JB), SIZE(P,1),  &          |
094|                     1.D0, A(1,1,INDX(IB,JB)), SIZE(A,1))                    |
095|             CALL DGEMM ('N', 'T', BS(IB), BS(JB), RANK, 1.D0,    &          |
096|                     P(1,1,IB), SIZE(P,1), U(1,1,JB), SIZE(U,1),  &          |
097|                     1.D0, A(1,1,INDX(IB,JB)), SIZE(A,1))                    |
098|          ENDDO                                                              |
099| !$OMP END PARALLEL DO                                                       |
100| !-------------------------------------------------------------------------- |
101|       ENDDO                                                                 |
102| END SUBROUTINE BREFLECT                                                     |
```

Fig. 13 Sample block reflection kernel in fortran 90.

Table 16 Errors of the 100 largest eigenpairs (random matrix, $N = 3600$).

b	err_{orth}	$rmax$
20	5.9E-15	4.0E-12
40	3.6E-15	4.1E-12
60	9.9E-15	5.1E-12
80	4.1E-15	5.9E-12
100	8.4E-15	6.1E-12

about 1E-11.

- For the random matrix of 100 largest eigenpairs, there is also much difference between Table 8 and **Table 16**. The err_{orth} is almost same level, but $rmax$ is much improved from the level of about 1E-5 to about 1E-11.

By the above comparisons, the modification to "INVITR" had an effect to reduce the error of eigenvectors was shown. (The reason why the set of eigenvectors solved by the original INVITR, when the matrix size was large, often contained not so accurate eigenvectors must be investigated.)

(Received October 3, 2005)
(Accepted January 24, 2006)

Hiroshi Murakami is currently an Assistant of Department of Mathematics and Information Sciences, Graduate Schools of Science and Engineering, Faculty of Urban Liberal Arts, Tokyo Metropolitan University. His research interests include efficient or accurate solutions for the problems in Mathematics or Sciences by numerical or symbolic methods, and their parallel implementations. He received the Doctor of Science degree in Chemistry from Hokkaido University in 1992.