A Parallel Direct Linear Equation Solver for Nonsymmetric Tridiagonal Matrices [*]

Y. Yamamoto[†], *M. Igai*[‡], *and K. Naono*[§]

1 Introduction

The problem of solving linear simultaneous equations with a tridiagonal coefficient matrix arises in many areas of scientific computing. Typical applications include computation of eigenvectors of a tridiagonal matrix by the inverse iteration method, solution of a partial differential equation by the ADI method and interpolation by spline functions. When the size of the matrix is large, it is appropriate to accelerate the solution with the use of parallel computers. Many approaches for the parallel solution of tridiagonal matrices have been proposed so far, including the dissection method [4], the cyclic reduction method [6][8] and a method based on the QR decomposition [10].

In the cyclic reduction method, the odd-numbered variables in the equations are eliminated first and the number of equations is reduced by half. This procedure is repeated until the number of equations is sufficiently small. This method has a large degree of parallelism and has been successfully implemented on vector processors [7]. Also, extensions to block tridiagonal matrices [8] and band matrices [12] have been proposed. In this method, however, the order in which the variables are eliminated is predetermined and pivoting for numerical stabilization cannot be incorporated. This makes it difficult to apply this method to general nonsymmetric tridiagonal matrices.

[*]The authors are grateful to Dr. Sigeo Ihara for providing us with the opportunity to start this study and to Dr. Yasuhiro Inagami, Dr. Satoshi Itoh and Mr. Nobuhiro Ioki for kind support.

[†]Central Research Laboratory, Hitachi Ltd.

[‡]Hitachi ULSI Systems Corp.

[§]Central Research Laboratory, Hitachi Ltd.

The dissection method is based on the Cholesky decomposition and extracts the parallelism in the elimination operation by renumbering the variables and equations [4]. It can be extended to nonsymmetric tridiagonal matrices by using the LU decomposition instead of the Cholesky decomposition. However, because the order of variable elimination is fixed also in this method, the type of matrices to which it is applicable is limited to symmetric positive definite matrices or diagonal dominant matrices.

The approach based on the QR decomposition [10], on the other hand, can be applied to general nonsymmetric tridiagonal matrices. However, it is known that the QR decomposition also needs pivoting to produce accurate solution when the matrix is nearly singular [11]. Hence, the applicability of this method without pivoting is also limited.

In this paper, we propose a new parallel direct solver for nonsymmetric tridiagonal matrices. Our method is a variant of the dissection method that can incorporate partial pivoting and can solve linear simultaneous equations with general nonsymmetric tridiagonal coefficient matrices on parallel machines efficiently and accurately.

A parallel direct solver for nonsymmetric tridiagonal matrices with pivoting has also been proposed in [9]. However, in this method, there is a tradeoff between the ratio of the sequential part in the algorithm and the number of interprocessor synchronizations. More specifically, if we denote the number of columns that have to be eliminated sequentially by $2r$ and the number of interprocessor synchronizations by q, pq is equal to the matrix size N. In contrast, our method has the advantage that it needs only one interprocessor synchronization and the number of columns that have to be eliminated sequentially is independent of N.

The paper is organized as follows: In section 2, we briefly review the conventional dissection method applied to tridiagonal matrices along with the difficulty arising in the case of nonsymmetric matrices. Our new parallel direct solver which incorporates partial pivoting is introduced in section 3. Numerical results on the Hitachi SR8000, a shared-memory parallel computer with 8 processors, can be found in section 4. Concluding remarks are given in the final section.

2 The Dissection Method and Its Limitation

2.1 Tridiagonal solver based on the dissection method

We consider a problem of solving a linear simultaneous equation $T\mathbf{x} = \mathbf{b}$, where T is a nonsymmetric tridiagonal matrix of order N. In the dissection method, we first transform T to $T' = PTP^t$ with a permutation matrix P and then solve a new equation $(PTP^t)(P\mathbf{x}) = P\mathbf{b}$ by Cholesky decomposition. P is determined so that the parallelism in the decomposition phase is maximized.

As is well known [4], a matrix A with symmetric nonzero pattern can be represented by a non-directed graph G_A. G_A has N vertices that correspond to rows of A and G_A has an edge between two vertices i and j if and only if $A_{ij} \neq 0$. The graph G_T corresponding to T is a chain, as shown in Fig. 1(a). We can identify the simultaneous permutation of rows and columns $T' = PTP^t$ with renumbering of G_T.

To solve $T\mathbf{x} = \mathbf{b}$ on a parallel computer with p processors using the dissection method, we divide G_T into p subregions and $p-1$ boundary vertices. Then we renumber the vertices so that the vertices in the first subregion are numbered first, those in the second subregion are numbered second, and so on, and the $p-1$ boundary vertices are given numbers from $N-p+2$ to N. The graph G_T with new vertex numbers is shown in Fig. 1(b) for the case of $p = 3$. Here, the boundary vertices are represented by shaded circles.

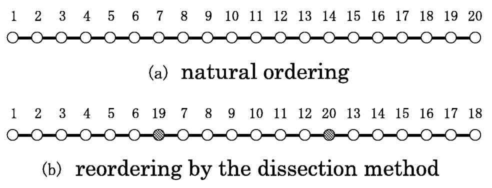

Figure 1. *A graph associated with a tridiagonal matrix.*

By applying the corresponding permutation of rows and columns to T, we obtain a matrix shown in Fig. 2. It can be seen from the figure that the original tridiagonal matrix T is transformed into a bordered block diagonal matrix with three diagonal blocks. When pivoting is not used, the Cholesky decomposition of each diagonal block can be performed independently. Thus we can solve the tridiagonal equation $T\mathbf{x} = \mathbf{b}$ in parallel using p processors.

2.2 Problems in the case of nonsymmetric matrices

When solving linear simultaneous equations with nonsymmetric coefficient matrix using direct methods, it is in general necessary to perform pivoting to ensure accuracy and numerical stability [5]. The most commonly used method for pivoting is the partial pivoting, which chooses the element in the pivot column with the largest modulus as the pivot element. In this subsection, we study how the parallelism in the dissection method for $T\mathbf{x} = \mathbf{b}$ is affected when the partial pivoting is introduced.

Assume we apply Gaussian elimination with partial pivoting to a matrix shown in Fig. 2. The nonzero pattern after the first 6 columns (which corresponds to the vertices in the first subregion in the graph of Fig. 1(b)) have been eliminated is shown in Fig. 3. The actual nonzero pattern depends on the sequence of the row numbers of the pivot elements chosen; $\sigma = (n_1, n_2, \ldots, n_6)$, where $i \le n_i \le N$. The nonzero pattern displayed in the figure is the union of nonzero patterns over all possible σ's. In the figure, the elements modified by the elimination operation are denoted by squares with oblique lines, while the elements generated by fill-ins are denoted by black squares.

From the figure, we can see that the element in the 7th column and the 2nd row from the last has been modified due to the elimination. However, in the elimination of the 7th column, this element is one of the candidates of the pivot element, because we choose the pivot element from all the elements in the column below the diagonal. This means that we cannot start the elimination of the 7th

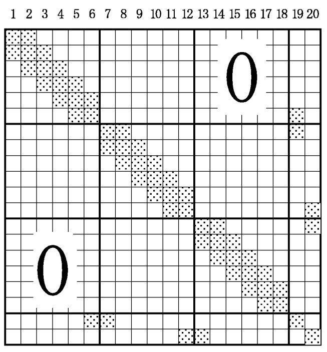

Figure 2. *A tridiagonal matrix reordered by the dissection method.*

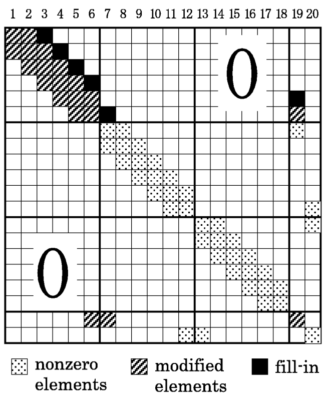

Figure 3. *Nonzero structure after elimination by the 6th column.*

column until the value of this element has been determined, that is, until the first six columns have been eliminated. Hence introduction of partial pivoting causes dependence of the elimination operations in the second subregion on those in the first subregion, thereby destroying the parallelism.

3 A Parallel Tridiagonal Solver with Partial Pivoting

3.1 The basic idea

In this section, we propose a new parallel direct solver for nonsymmetric tridiagonal matrices that can incorporate partial pivoting. We achieve this by modifying the reordering scheme in the conventional dissection method.

In the example shown in the previous subsection, the dependence of the elimination operations was caused due to the existence of the nonzero element in the 7th column and the 2nd row from the last. This element is nonzero because the rightmost vertex in the first subregion is connected with the leftmost vertex in the second subregion through a boundary vertex (vertex 19 in Fig. 1(b)). In our algorithm, we dissolve this dependence by renumbering the vertices again in each subregion of Fig. 1(b). More specifically, in each subregion, the vertex numbers of all the "purely inner" vertices, which are not adjacent to any boundary vertices, are decremented by one, and the leftmost vertex in the subregion is given the second largest number in the subregion. By reordering all the subregions in this manner, we obtain the numbering of the vertices shown in Fig. 4.

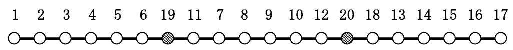

Figure 4. *Reordering of the nodes by the proposed method.*

The new matrix corresponding this renumbering is shown in Fig. 5. Because the leftmost vertex in each subregion is given the second largest number in the subregion, the element that caused the dependence of elimination operation is moved to the second last column in the subregion.

3.2 Parallelism in the elimination operation

In Fig. 7, we show the block structure of the nonzero pattern of the matrix shown in Fig. 5. Here, the columns of the matrix are divided into sets that correspond to purely inner vertices (A, B and C), inner vertices that are connected to boundary vertices (the thin column sets right after the sets A, B and C) and the boundary vertices (the last set). Likewise, the rows are divided into sets that correspond to three subregions and the boundary vertices. Each block of the matrix is shaded if there are at least one nonzero element in the block, and is white otherwise.

Now we focus on the column set B. Among the four blocks in B, only the second one contains nonzero elements and the blocks left to this block contain no nonzero elements. As a result, the columns in B are not modified by the elimination of columns left to B, even if partial pivoting is introduced. This means that we can start elimination of columns in B before elimination of columns in A has been completed. Similarly, because the only nonzero block in C is the third one and all the blocks left to this block is zero, the columns in C is not modified by the elimination of columns left to C. So we can start eliminating columns in C without waiting for the completion of the elimination of columns in A and B.

Fig. 6 illustrates the nonzero structure of the matrix shown in Fig. 5 after elimination by the 6th column. As we have explained now, the columns in B (the 7th to the 10th column) have not been modified by the elimination of the 1st to the 6th columns. By using this new renumbering, we can eliminate the columns in A, B and C in parallel using 3 processors.

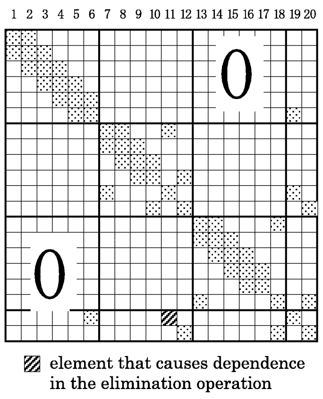
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
0
0
▨ element that causes dependence in the elimination operation

Figure 5. *A tridiagonal matrix reordered by the proposed method.*

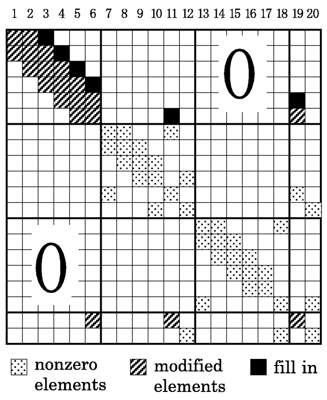

Figure 6. *Nonzero structure after elimination by the 6th column.*

In Fig. 8, we show the nonzero structure of the second block in B during elimination. Here, we consider the situation where there are 12 columns in the set

B and the elimination by the 5th column in the set have been completed. When we apply Gaussian elimination with partial pivoting to a tridiagonal matrix, it can be easily seen that fill-ins appear at positions two elements above the diagonal. In our method, in addition to these, we also have fill-ins in the second last rows and columns in the block. As a result, the number of elements involved in the elimination in each step increases from 6 to 12 and the number of pivot candidates increases from two to three. This almost doubles the number of arithmetic operations and it is the price we have to pay for parallelization.

So far, we have described our algorithm for the case of $p = 3$. However, it can be easily generalized to use any number of processors. Our algorithm needs only one interprocessor synchronization, which occurs when all the "purely inner" columns allocated to each processor have been eliminated. The remaining columns, which correspond to the boundary vertices and vertices adjacent to them, needs to be eliminated sequentially, but the number of such columns is $3(p-1)$ and is independent of the matrix size N.

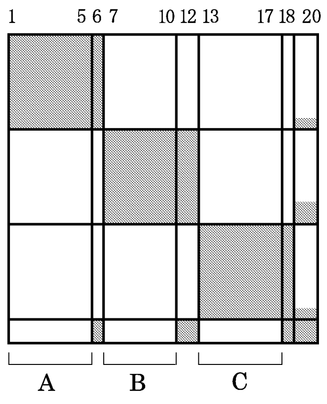

Figure 7. *Block structure of nonzero elements.*

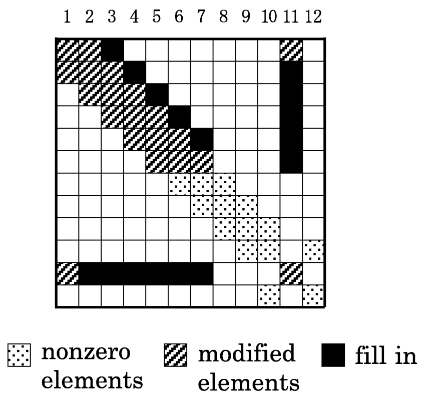

Figure 8. *Nonzero structure of block B during elimination.*

4 Numerical Results

We implemented our method on the Hitachi SR8000/F1, a shared memory parallel machine with 8 processors, and compared its performance and numerical accuracy with that of the conventional methods.

4.1 Parallel performance

To evaluate the parallel performance, we used random tridiagonal matrices whose elements were extracted from uniform random numbers in [0,1] (matrices of type (a)). The matrix size N was varied from 500 to 8,000 and the number of processors p was varied from 1 to 8. We used a sequential tridiagonal solver based on Gaussian elimination with partial pivoting when $p = 1$ and used our method when $p \geq 2$.

The execution times for the LU decomposition part are shown in Table 1 and Fig. 9. As can be seen from the table, our method achieves speedup of 5.5 times compared with the sequential method when $N = 8000$ and $p = 8$. It is also faster than the sequential method when $p = 2$ or $p = 4$.

Fig. 10 shows the details of the execution time for the case of $p = 8$. The white area and the shaded area denote the execution time for the parallel part (elimination of the purely inner columns) and the sequential part (elimination of the remaining columns), respectively. We can see from the graph that the execution time for the latter is almost constant and its percentage decreases as N increases. This is in consistent with the observation we made at the end of subsection 3.2 and means that the parallel efficiency of our method increases with N.

We also measured the execution times for matrices of type (b) and (c) which we will define in the next subsection. But here we omit the results because they were almost the same as those for matrices of type (a).

Table 1. *Execution time of the LU decomposition.*

Matrix size N	Sequential (1PU)	Ours (2PU)	Ours (4PU)	Ours (8PU)	Speedup (8PU)
500	3.26E-4	2.04E-4	1.85E-4	2.58E-4	1.26
1000	6.24E-4	3.49E-4	2.66E-4	3.05E-4	2.04
2000	1.21E-3	6.63E-4	4.32E-4	3.84E-4	3.15
4000	2.44E-3	1.32E-3	7.42E-4	5.48E-4	4.45
8000	4.79E-3	2.59E-3	1.38E-3	8.75E-4	5.47

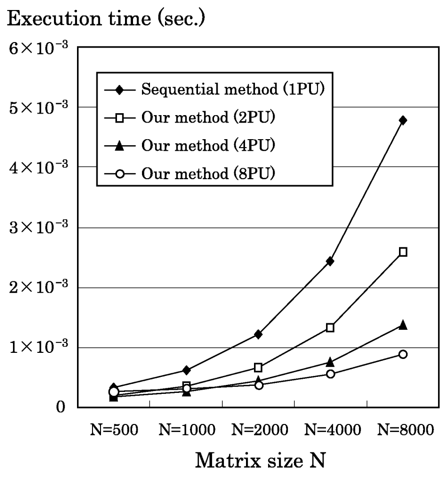

Figure 9. *Execution time of the LU decomposition.*

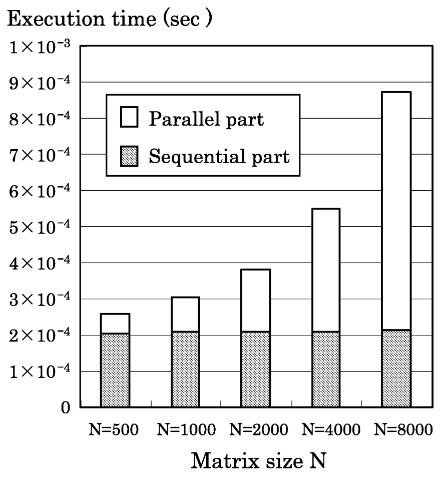

Figure 10. *Details of the execution time.*

4.2 Numerical accuracy

Next we compared the accuracy of our method with that of the sequential tridiagonal solver with partial pivoting and that of the dissection method without pivoting. The problems we used are (a) the random matrices which we used in the previous subsection, (b) random matrices which are the same as (a), except that the diagonal elements are multiplied by 10^{-4}, and (c) matrices obtained by tridiagonalizing the Frank matrices $A_{ij} = \min(i, j)$ and subtracting their smallest eigenvalue from the diagonal elements. Matrices of type (c) arise in the computation of eigenvectors using the inverse iteration method. The matrix size N was varied from 500 to 8000 as in the previous subsection, and the accuracy was measured in terms of the residual $\| T\mathbf{x} - \mathbf{b} \|_\infty$.

The residual for the matrices (a), (b) and (c) are shown in Figs. 11, 12 and 13, respectively. It can be seen from the figures that our method achieves higher accuracy than the dissection method in all cases except for one, which is the $N = 2000$ case for matrix (c). The accuracy of our method is up to two orders of magnitude higher than the dissection method for matrices of type (b). This suggests that pivoting is indispensable for matrices which do not have diagonal dominance. When compared with the sequential Gaussian elimination with partial pivoting, our method attains much the same accuracy. The results shown in figures 11 and 12 are for a specific seed of the random number generator, but the difference of the accuracy of the three methods showed almost the same tendency for other values of the seed. In Fig. 14, we show how the residual changes when the seed of the random number generator is changed for the case of $N = 2000$ and matrix type (b). As can be seen from the graph, the accuracy of our method is almost the same as that of the sequential method and about two orders of magnitude better than that of the dissection method. This agrees with the results shown in Fig. 12.

From these numerical results, we can say that our method is a good choice when one wants to solve general nonsymmetric tridiagonal matrices on parallel computers efficiently and accurately.

5 Conclusion

In this paper, we proposed a new parallel direct solver for nonsymmetric tridiagonal matrices that can incorporate partial pivoting. We implemented our algorithm on one node of the Hitachi SR8000/F1 and obtained speedup of 5.5 times compared with the sequential tridiagonal solver with partial pivoting when the matrix size is 8000 and the number of processor is 8. The accuracy of our method is almost the same as that of the sequential solver and is up to two orders of magnitude better than that of the parallel solver based on the dissection method without pivoting. Our future work will include implementation of this algorithm on distributed memory parallel machines and incorporation of this algorithm into real applications such as the inverse iteration method for eigenvalue computation.

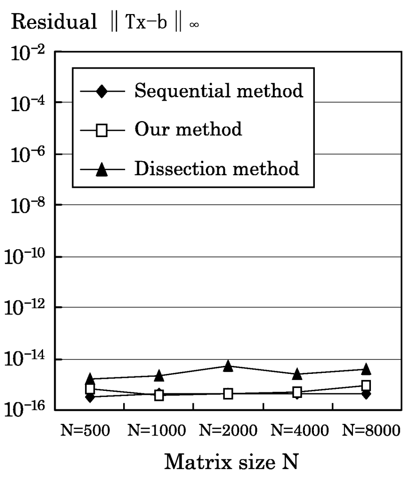

Figure 11. *Residual of the three methods for random matrices.*

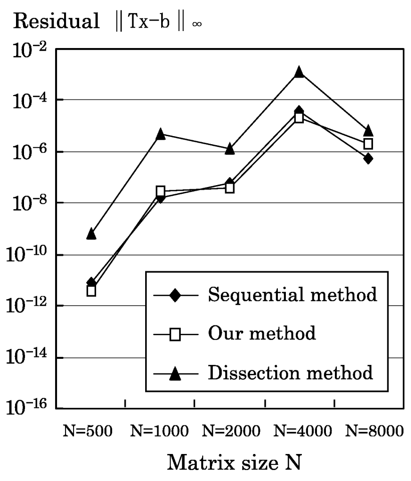

Figure 12. *Residual of the three methods for random matrices with diagonal elements multiplied by* 10^{-4}.

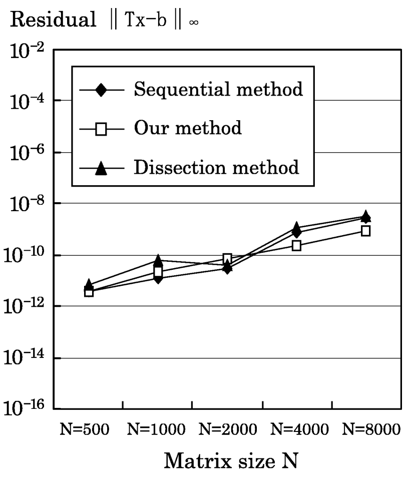

Figure 13. *Residual of the three methods for matrices obtained by tridiagonalizing the Frank matrices.*

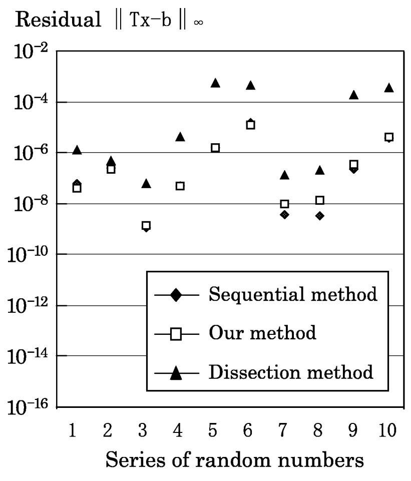

Figure 14. *Residual of the three methods when the seed of the random number generator is changed.*

Bibliography

[1] J. H. Wilkinson and C. Reinsch (eds.), *Linear Algebra*, Springer Verlag (1971).

[2] R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall (1962).

[3] C. H. Reinsch, *Smoothing by Spline Functions*, Numerische Mathematik, Vol. 10 (1967), pp. 177–183.

[4] M. T. Heath et al, *Parallel Algorithms for Sparse Linear Systems*, in Parallel Algorithms for Matrix Computations, SIAM (1990).

[5] G. H. Golub and C. F. van Loan, *Matrix Computations*, The Johns Hopkins University Press (1989).

[6] H. S. Stone, *An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations*, J. Assoc. Comput. Mach., Vol. 20 (1973), pp. 27–38.

[7] K. Sumiyoshi and T. Ebisuzaki, *Performance of Parallel Solution of a Block Tridiagonal Linear System on Fujitsu VPP 500*, Parallel Computing, Vol. 24 (1998), pp. 287–304.

[8] D. Heller, *Some Aspects of the Cyclic Reduction Algorithm for Block Tridiagonal Linear Systems*, SIAM J. Numer. Anal., Vol. 13, No. 4 (1976), pp. 484–496.

[9] M. Hegland, *On the Parallel Solution of Tridiagonal Systems by Wraparound Partitioning and Incomplete LU Factorization*, Numerische Mathematik, Vol. 59, No. 5 (1991), pp. 453–472.

[10] P. Amodio and L. Brugnano, *The Parallel QR Factorization Algorithm for Tridiagonal Linear Systems*, Parallel Computing, Vol. 21 (1995), pp. 1097–1110.

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

[12] P. Dubois and G. Rodrigue, *An Analysis of the Recursive Doubling Algorithm*, In Kuck, D. J. and Sameh, A. H., eds., High Speed Computer and Algorithm Organization, Academic Press, New York (1977).