THE SOLUTION OF MESH EQUATIONS ON A PARALLEL COMPUTER*

Joseph W.H. Liu

Department of Computer Science
University of Waterloo, Waterloo, Ontario, Canada

CS-74-19

October 1974

* Work supported in part by a University of Waterloo Graduate Bursary.

THE SOLUTION OF MESH EQUATIONS ON A PARALLEL COMPUTER*

by

Joseph W.H. Liu

Department of Applied Analysis & Computer Science

University of Waterloo

Waterloo, Ontario, Canada

* Work supported in part by a University of Waterloo Graduate Bursary

ABSTRACT

This paper discusses the applicability of "nested dissection" (due to George) in the parallel solution of mesh systems by symmetric elimination. For systems associated with an n by n regular mesh, it is known that elimination by nested ordering, which requires $O(n^3)$ multiplications and $O(n^2 \log_2 n)$ storage locations on sequential computers, is optimal. In this paper, it is shown that the dissection technique coupled with parallel elimination allows such mesh systems to be solved in $O(n)$ arithmetic operations. The doubly nested dissection, a scheme more convenient for parallel computation, is also discussed.

§1 Introduction

The use of internal parallelism has a significant influence on the development of recent computing systems. The availability of parallel array computers makes it important to consider how such a capability can be exploited in various classes of computational problems.

In this paper, the method of symmetric elimination for solving a system of mesh equations is analyzed from the viewpoint of its adaptability for parallel computation. Our aim is to study the inherent parallelism contained in a particular problem formulation and to determine by how much we can increase the speed if we use the parallel mode of computation.

The computer model that we shall use in the analysis is assumed to have an unlimited number of processors. They are identical arithmetic processors, each of which can perform any of the binary operations $+$, $-$, $\times$, $\div$ in unit time. All processors obtain their instructions from a single instruction stream, so that they execute this same instruction, but on different operand pairs.

For our study, we ignore all problems of memory access, data communication, and programming. Efficiency losses due to book-keeping, scheduling and overhead are neglected. Our main concern is the parallel content of the symmetric elimination algorithm for a system of mesh equations.

Let M be the mesh consisting of n^2 squares (elements), formed by subdividing the unit square with a mesh spacing of $\frac{1}{n}$. An unknown is associated with each of the $N = (n+1)^2$ grid points, called nodes in M. Following George [4], we define a finite element system or mesh system associated with M to be any N by N symmetric, positive definite system

(1.1) $Ax = b$

with the property that entry A_{ij} is nonzero only if unknowns x_i and x_j are associated with nodes of the same mesh element of M.

The numerical solution of (1.1) using the LDL^T decomposition of A on a serial computer has been studied by George [3,4], and Hoffman, Martin and Rose [6]. With the assumption that we avoid operating on and storing zeros in the decomposition, Hoffman, Martin and Rose use a graph-theoretical approach to obtain lower bounds on the number of nonzero entries in L and the number of multiplicative operations required to effect the LDL^T decomposition. They show that at least $O(n^3)$ multiplications are required and at least $O(n^2 \log_2 n)$ storage is needed for <u>any</u> ordering of the system. The <u>nested dissection</u> scheme developed by George [3,4] is found to attain these lower bounds. Thus, the nested ordering is optimal in the order of magnitude sense with respect to computational complexity and storage, provided that we use the symmetric factorization algorithm.

In this context, Birkhoff and George [1] point out that there is enormous potential parallelism in the computations associated with nested dissection. In this paper, we consider the computational complexity bound for the parallel solution of the finite element system (1.1) using symmetric elimination by dissection techniques. Our measure of computation is taken to be the number of parallel arithmetic operations required to factor A into LDL^T and to do the back substitution. In section 3, the nested dissection scheme is used to show that the factorization of A can be done in $O(n)$ parallel operations. In section 4, we show that the final solution

can be obtained again in O(n) operations. The doubly nested dissection, a scheme more suited for parallel computation, is discussed in section 5. Section 6 contains our concluding remarks.

§2 Symmetric Elimination

In this section, we review the elimination process and discuss its adaptability for parallel computation. For our purposes, we find it convenient to describe the LDL^T factorization of A by the outer product formulation (Rose [10]).

Let $D_0 = A$. We have

$$D_0 = \begin{pmatrix} d_1 & v_1^T \\ v_1 & B_1' \end{pmatrix} = \begin{pmatrix} 1 & \\ \frac{v_1}{d_1} & I_{N-1} \end{pmatrix} \begin{pmatrix} d_1 & \\ & B_1' - \frac{v_1 v_1^T}{d_1} \end{pmatrix} \begin{pmatrix} 1 & v_1^T/d_1 \\ & I_{N-1} \end{pmatrix}$$

$$= L_1 \begin{pmatrix} d_1 & \\ & B_1 \end{pmatrix} L_1^T = L_1 D_1 L_1^T,$$

$$D_1 = \begin{pmatrix} d_1 & \\ & B_1 \end{pmatrix} = \begin{pmatrix} d_1 & & \\ & d_2 & v_2^T \\ & v_2 & B_2' \end{pmatrix}.$$

The process is repeated recursively on B_1, and at the k-th step, we have

$$D_{k-1} = \begin{pmatrix} d_1 & & & \\ & \ddots & & \\ & & d_{k-1} & \\ & & & B_{k-1} \end{pmatrix} = \begin{pmatrix} d_1 & & & \\ & \ddots & & \\ & & d_{k-1} & \\ & & & \begin{matrix} d_k & v_k \\ v_k & B_k' \end{matrix} \end{pmatrix}$$

$$= \begin{pmatrix} I_{k-1} & & \\ & 1 & \\ & \frac{v_k}{d_k} & I_{N-k} \end{pmatrix} \begin{pmatrix} d_1 & & & \\ & \ddots & & \\ & & d_{k-1} & \\ & & & \begin{matrix} d_k & \\ & B_k' - \frac{v_k v_k^T}{d_k} \end{matrix} \end{pmatrix} \begin{pmatrix} I_{k-1} & & \\ & 1 & v_k^T/d_k \\ & & I_{N-k} \end{pmatrix} \tag{2.1}$$

$$= L_k D_k L_k^T$$

where d_k is a positive scalar,

v_k is a vector of length (N-k),

B_k is an (N-k) by (N-k) symmetric positive definite matrix.

The procedure eventually stops after the (N-1)st step with

$$D_{N-1} = \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_N \end{pmatrix} .$$

Hence, A can be written as

$$L_1 L_2 \cdots L_{N-1} D_{N-1} L_{N-1}^T \cdots L_2^T L_1^T = LDL^T,$$

where $L = L_1 L_2 \ldots L_{N-1}$ is unit lower triangular

and $D = D_{N-1}$ is diagonal.

In the sequel, we shall refer to performing the k-th step of the factorization as <u>eliminating variable x_k</u>.

In [9], Pease studies the method of Gaussian elimination for solving general N by N linear systems from the viewpoint of parallel processing. For parallel machines having the capability of replacing a row by a linear combination of two rows simultaneously, he shows that general linear systems can be solved in $O(N^2)$ parallel operations. For the computer model used in our analysis, more parallel computations can be exploited so that we have

<u>Lemma 2.1</u> The symmetric factorization of an N by N symmetric positive definite matrix can be performed in 3(N-1) parallel arithmetic operations.

Proof Consider the k-th step in the factorization algorithm (2.1). One parallel division gives v_k/d_k. The outer product $(v_k/d_k)v_k^T$ can then be formed using one multiplication. Another parallel subtraction $B_k = B_k'-(v_k/d_k)v_k^T$ completes the k-th step. Summing over the (N-1) steps, we need at most 3(N-1) operations for the whole factorization process.

In lemma 2.1, if ν_k is the number of nonzero components in v_k, it is straightforward to see that p-fold parallelism is necessary, where $p = \max_k \{\frac{1}{2}\nu_k(\nu_k+1)\}$.

We note that the factorization algorithm given by (2.1) is itself essentially sequential in nature. The computation involved in the elimination of variable x_k depends on the results of the previous eliminations. Thus, we need to investigate the zero-nonzero structure of the finite element matrix A as a source of parallelism.

Let X denote the set of variables in the linear systems. Let

$$R = \{x_{p_1}, x_{p_2}, \ldots, x_{p_r}\} \quad \text{and}$$

(2.2)

$$S = \{x_{q_1}, x_{q_2}, \ldots, x_{q_s}\}$$

be two disjoint subsets of X.

Definition Subsets R and S are said to be independent if the off-diagonal component $A_{p_i q_j}$ is zero for $i = 1,\ldots,r$ and $j = 1,\ldots,s$. Otherwise, they are said to be connected.

Equivalently, R and S are independent if and only if there exists a permutation P on A such that

$$(2.3) \qquad PAP^T = \begin{pmatrix} A_R & & C_R^T \\ & A_S & C_S^T \\ C_R & C_S & B \end{pmatrix}$$

where A_R and A_S and $r \times r$ and $s \times s$ submatrices respectively and

$$Px = \begin{pmatrix} x_{p_1} \\ \vdots \\ x_{p_r} \\ x_{q_1} \\ \vdots \\ x_{q_s} \\ \vdots \end{pmatrix} .$$

For a subset S of variables, the boundary $\partial(S)$ of S is defined to be the set of all variables not in S that are connected to S. If $S = \{x\}$, we shall write $\partial(x)$.

Lemma 2.2 R and S are independent if and only if $R \cap S = \phi$ and $R \cap \partial(S) = \phi$.

By the submatrix in A corresponding to S, we mean the submatrix obtained by deleting all the rows and columns i of A, where $i \notin \{q_1, q_2, \ldots, q_s\}$. The following lemma is a restatement of a result due to Parter [8].

Lemma 2.3 The elimination of variables in S only affects the submatrix in A corresponding to the subset $S \cup \partial(S)$.

We now discuss the solution of systems with independent subsets. Let R and S be independent subsets (2.2) and P be as defined in (2.3). The equivalent permuted system

$$(PAP^T)\ (Px) = Pb$$

has a desirable block structure (2.3), where parallel elimination is possible. Symmetric factorization of PAP^T can be performed as follows:

Step 1a Perform the first r steps of the factorization of

$$\begin{pmatrix} A_R & C_R^T \\ C_R & 0 \end{pmatrix} \text{ into } \begin{pmatrix} L_R & \\ C_R L_R^{-T} D_R^{-1} & I_{N-r-s} \end{pmatrix} \begin{pmatrix} D_R & \\ & -C_R A_R^{-1} C_R^T \end{pmatrix} \begin{pmatrix} L_R^T & D_R^{-1} L_R^{-1} C_R^T \\ & I_{N-r-s} \end{pmatrix},$$

where $A_R = L_R D_R L_R^T$.

Step 1b Perform the first s steps of the factorization of

$$\begin{pmatrix} A_S & C_S^T \\ C_S & 0 \end{pmatrix} \text{into} \begin{pmatrix} L_S & \\ C_S L_S^{-T} D_S^{-1} & I_{N-r-s} \end{pmatrix} \begin{pmatrix} D_S & \\ & -C_S A_S^{-1} C_S^T \end{pmatrix} \begin{pmatrix} L_S^T & D_S^{-1} L_S^{-1} C_S^T \\ & I_{N-r-s} \end{pmatrix},$$

where $A_S = L_S D_S L_S^T$.

Step 2 Compute $\bar{B} = B+(-C_R A_R^{-1} C_R^T)+(-C_S A_S^{-1} C_S^T)$, and factor $\bar{B}$ into $L_{\bar{B}} D_{\bar{B}} L_{\bar{B}}^T$.

It should be clear that steps 1a and 1b can be executed simultaneously due to the independence of R and S (George [5]). We shall refer to this process as parallel block factorization (or elimination) with respect to R and S.

In anticipation of what follows in the next section, we establish some preliminary results. The result in the following lemma is implicit in the graph-theoretic treatment of elimination by Rose [10].

Lemma 2.4 If A is irreducible, v_k is a nonzero vector of length (N-k) for k = 1,...,N-1.

Lemma 2.5 Let the matrix A be partitioned as $\begin{pmatrix} A_1 & C^T \\ C & B \end{pmatrix}$, where A_1 is an M by M irreducible submatrix. If parallel elimination of (2.1) is used, the number of parallel operations required to perform the first (M-1) steps of the factorization algorithm for A is the same as that required to factor A_1.

Proof Lemma 2.4 implies that the outer products of all $v_k v_k^T$ for A_1 have to be computed. Thus, in carrying out the first (M-1) factorization step for A, the same number of outer products is required, although vectors of longer length are involved. This does not increase the parallel operation count.

§3 Nested Dissection and Parallel Factorization

In parallelism exploitation, recursive doubling (Stone [12]), a technique generalized from the log product rule for the product of 2^m numbers, is often used. The basic technique involves the splitting of the given task into two smaller, similar and independent subtasks. The subtasks, being independent, can be executed simultaneously in different processors. The intermediate results of the subtasks are then combined to complete the computation. It is important to realize that each subtask can in turn be performed by successive splitting so that the computations will be spread over more processors.

The important idea of nested dissection, developed by George [3,4] in solving finite element problems, has precisely the characteristics of recursive doubling. In this section, the applicability of dissection techniques in the solution of the mesh system (1.1) by parallel elimination will be discussed. We shall follow the approach of Birkhoff and George [1], where nested dissection is treated as partial orderings of partitions of the mesh into disconnected components.

Consider the linear system (1.1) associated with the regular n by n grid on the unit square. Without loss of generality, we assume that $n = 2^{\ell}-2$ for some integer $\ell \geq 2$. The case for $n = 6$ is shown in Figure 3.1 below.

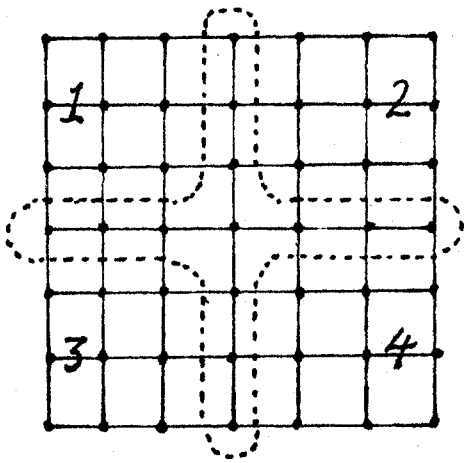

Figure 3.1 Dissection of 6 × 6 regular mesh

By using a "+" shaped set of separating vertices (enclosed by dotted line in Figure 3.1), we can disconnect the mesh M into four regular $(\frac{n}{2}-1)$ by $(\frac{n}{2}-1)$ submeshes. From the definition of a finite element system, it is easy to see that variables associated with the four components are mutually independent. The partition, hence, induces the following block structure on (1.1):

$$(3.1)\qquad \begin{pmatrix} A_1 & & & & C_1^T \\ & A_2 & & & C_2^T \\ & & A_3 & & C_3^T \\ & & & A_4 & C_4^T \\ C_1 & C_2 & C_3 & C_4 & B \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_B \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_B \end{pmatrix},$$

where A_1, A_2, A_3, A_4 are irreducible finite element matrices associated with the four submeshes. The matrix of (3.1) is in a desirable form in which the parallel block factorization of section 2 can be applied. Let $F(n)$ denote the number of parallel operations required to factor a finite element matrix associated with the n by n grid by parallel block elimination.

Lemma 3.1 $\quad F(n) = F(\frac{n}{2}-1)+O(n)$.

Proof On performing the parallel block factorization for

$$\begin{pmatrix} A_i & C_i^T \\ C_i & 0 \end{pmatrix} \qquad i = 1,2,3,4,$$

it follows from lemma 2.5 that $F(\frac{n}{2}-1)+O(1)$ parallel operations are required. The entire factorization can then be completed by forming

$$\bar{B} = B - \sum_{i=1}^{4} C_i A_i^{-1} C_i^T$$

and factoring $\bar{B}$. Since $\bar{B}$ is an 2n+1 by 2n+1 matrix, by lemma 2.1, 4+6n parallel operations are sufficient. Combining, we have
$F(n) = F(\frac{n}{2} - 1)+O(1)+4+6n = F(\frac{n}{2} - 1)+O(n)$.

Successive application of splitting is implicit in the recursive relation of lemma 3.1. Thus, each independent component of the dissected mesh is itself dissected, so that with infinite parallelism, we have

Theorem 3.2 $F(n) = O(n)$.

Proof The theorem follows directly from lemma 3.1 and the inequality

$$1+ \frac{1}{2} + \frac{1}{2^2} +\ldots+ \frac{1}{2^\ell} < \sum_{i=0}^{\infty} \frac{1}{2^i} = 2.$$

Theorem 3.2 shows that the dissection idea is well suited for parallel computation. In fact, we may reinterpret the idea as that of ordering the vertices in the mesh M. From the matrix point of view, elimination by nested dissection may be regarded as the elimination of a maximal set of pairwise independent variables at each stage. It should be clear that elimination of independent variables can be treated as independent tasks and hence can be performed in parallel. This helps to explain the effective use of parallel capability by nested dissection from another point of view.

For completeness sake, a nested dissected ordering for n = 6 is given in Figure 3.2. We have adopted the notation that variables that will be eliminated in parallel are given the same number in the nested ordering.

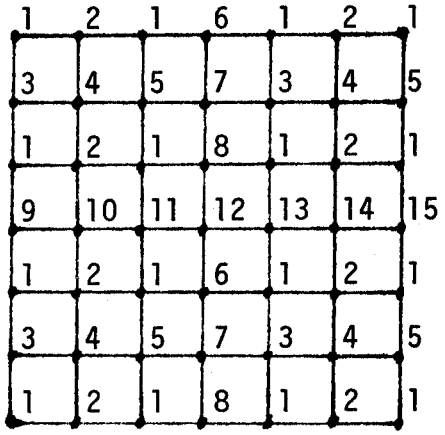

Figure 3.2 Nested Ordering for n = 6

§4 Final Solution of the Mesh System

After completing the symmetric factorization of the finite matrix A into LDL^T, the solution of the n by n mesh system (1.1) can then be found by solving the systems

$$\begin{aligned} Ly &= b, \\ Dz &= y, \\ \text{and} \quad L^T x &= z. \end{aligned} \tag{4.1}$$

Recall that the lower triangular matrix L has $O(n^2 \log_2 n)$ nonzero entries. It is then easy to see that on a serial computer, (4.1) can be done in $O(n^2 \log_2 n)$ operations. In this section, we shall see how this substitution step can be speeded up by parallelism.

We begin with a careful study of the solution of an N by N unit lower triangular system:

$$Lu = b. \tag{4.2}$$

Let L be expressed in the following form:

$$\begin{pmatrix} 1 & & & & \\ & 1 & & & \\ \ell_1 & & \ell_2 & & \\ & & & 1 & \\ & & & & 1 \\ & & & \ell_{N-1} & \end{pmatrix},$$

where each ℓ_k is a vector of length (N-k). The system (4.2) can be solved as follows:

Step 0: Initialise $u_i = b_i$ for $i = 1,\ldots,N$.

Step k: Compute $u_k \ell_k$ and form

$$(4.3) \quad \begin{pmatrix} u_{k+1} \\ \vdots \\ u_N \end{pmatrix} \leftarrow \begin{pmatrix} u_{k+1} \\ \vdots \\ u_N \end{pmatrix} - u_k \ell_k .$$

After (N-1) steps, the vector u contains the solution of (4.2). A direct exploitation of parallelism in (4.3) gives

Lemma 4.1 The solution of an N by N unit lower triangular linear system can be obtained in parallel using 2(N-1) operations.

For those L's arising from the LDL^T factorization of mesh systems, the operation count in lemma 4.1 can be significantly improved. Let R and S be two independent sets of variables. The symmetric factorization of

$$\begin{pmatrix} A_R & & C_R^T \\ & A_S & C_S^T \\ C_R & C_S & B \end{pmatrix}$$

yields an L with the block form (see section 2):

$$\begin{pmatrix} L_R & & \\ & L_S & \\ \hat{\tilde{C}}_R & \tilde{C}_S & L_{\tilde{B}} \end{pmatrix}$$

where $A_R = L_R D_R L_R^T$,

$A_S = L_S D_S L_S^T$,

$\bar{C}_R = C_R^{-1} L_R^{-T} D_R^{-1}$,

$\bar{C}_S = C_S^{-1} L_S^{-T} D_S^{-1}$,

and $\bar{B} = B - C_R A_R^{-1} C_R^T - C_S A_S^{-1} C_S^T = L_{\bar{B}} D_{\bar{B}} L_{\bar{B}}^T$.

The forward substitution in

$$\begin{pmatrix} L_R & & \\ & L_S & \\ \bar{C}_R & \bar{C}_S & L_{\bar{B}} \end{pmatrix} \begin{pmatrix} y_R \\ y_S \\ y_B \end{pmatrix} = \begin{pmatrix} b_R \\ b_S \\ b_B \end{pmatrix}$$

can be performed by the following parallel block substitution scheme:

Step 1a Solve

$$\begin{pmatrix} L_R & \\ \bar{C}_R & I_{N-r-s} \end{pmatrix} \begin{pmatrix} y_R \\ b'_R \end{pmatrix} = \begin{pmatrix} b_R \\ 0 \end{pmatrix},$$

where I_{N-r-s} is the identity matrix of order (N-r-s).

Step 1b Solve

$$\begin{pmatrix} L_S & \\ \bar{C}_S & I_{N-r-s} \end{pmatrix} \begin{pmatrix} y_S \\ b'_S \end{pmatrix} = \begin{pmatrix} b_S \\ 0 \end{pmatrix}.$$

Step 2 Compute $b'_B = b_B + b'_R + b'_S$, and solve

$$L_{\bar{B}}\, y_B = b'_B.$$

Again, steps 1a and 1b can be executed independently. We need the following lemma:

Lemma 4.2 Let $A = \begin{pmatrix} A_1 & C^T \\ C & B \end{pmatrix}$, where $A_1 = L_1 D_1 L_1^T$ is an M by M irreducible submatrix. If parallel substitution of (4.3) is used, the number of parallel operations required to perform the first (M-1) steps of the back substitution for

$$\begin{pmatrix} L_1 & \\ \bar{C} & L_{\bar{B}} \end{pmatrix} \begin{pmatrix} y_1 \\ y_B \end{pmatrix} = \begin{pmatrix} b_1 \\ b_B \end{pmatrix}$$

is the same as that required to solve $L_1 y_1 = b_1$.

Proof By lemma 2.2, the columns below the diagonal in L_1 are nonzero. Thus the two tasks require the same amount of parallel operations although the first one involves scalar multiplication and subtraction of longer vectors.

Similar scheme and results can be obtained for upper triangular systems. We are now ready to improve the complexity bound for the parallel solution of (4.1). Recall that the mesh system (1.1) associated with the n by n grid can be reordered by the dissection technique into the block form (3.1). If $A = LDL^T$, the corresponding unit lower triangular matrix L takes the form:

$$\begin{pmatrix} L_1 & & & & \\ & L_2 & & & \\ & & L_3 & & \\ & & & L_4 & \\ \bar{C}_1 & \bar{C}_2 & \bar{C}_3 & \bar{C}_4 & L_{\bar{B}} \end{pmatrix},$$

where $A_i = L_i D_i L_i^T$,

$\bar{C}_i = C_i L_i^{-T} D_i^{-1}$, for i = 1,2,3,4.

Let S(n) be the number of parallel operations required to perform the forward and backward substitution (4.1) by parallel block substitution. With lemmas 4.1 and 4.2, it is not difficult to obtain:

Lemma 4.3 $S(n) = S(\frac{n}{2} - 1) + O(n)$.

Theorem 4.4 $S(n) = O(n)$.

§5 Doubly Nested Dissection

A careful study of the parallel block factorization algorithm with respect to independent subsets R and S (section 2) shows that the modifications of B by $-C_R A_R^{-1} C_R^T$ and $-C_S A_S^{-1} C_S^T$ are performed sequentially in step 2. If we were to factor the apparently more appropriate submatrices

$$\begin{pmatrix} A_R & C_R \\ C_R & B \end{pmatrix} \text{ and } \begin{pmatrix} A_S & C_S^T \\ C_S & B \end{pmatrix}$$

in step 1, that is, to compute $B-C_R A_R^{-1} C_R^T$ and $B-C_S A_S^{-1} C_S^T$ synchronously, we encounter the problem of memory interference. This arises because some entries in B may need modifications from the contributions of both $-C_R A_R^{-1} C_R^T$ and $-C_S A_S^{-1} C_S^T$.

Problems of similar nature occur in the parallel block substitution scheme (section 4), if

$$\begin{pmatrix} L_R & \\ \bar{C}_R & I_{N-r-s} \end{pmatrix} \begin{pmatrix} y_R \\ b'_B \end{pmatrix} = \begin{pmatrix} b_R \\ b_B \end{pmatrix}$$

and

$$\begin{pmatrix} L_S & \\ \bar{C}_S & I_{N-r-s} \end{pmatrix} \begin{pmatrix} y_S \\ b'_B \end{pmatrix} = \begin{pmatrix} b_S \\ b_B \end{pmatrix}$$

were to be solved simultaneously. The treatment in sections 3 and 4 offers a solution, but it requires some additional computations and extra storage. In this section, we give an alternate solution to this problem.

Recall from lemma 2.3 that the elimination of variables in S does not change the submatrix corresponding to variables in $X\backslash(S \cup \partial S)$. Bearing this in mind, we introduce the concept of complete independence.

Definition Variable subsets R and S are said to be completely independent if they are independent and $\partial R \cap \partial S = \phi$.

In other words, complete independence implies a permutation P on A such that

$$PAP^T = \begin{pmatrix} A_R & & & \text{symmetric} \\ & A_S & & \\ C_R & & A_{\partial R} & \\ & C_S & C_{RS} & A_{\partial S} \\ & & C_{\partial R} & C_{\partial S} \quad B \end{pmatrix}.$$

It is obviously desirable to have B = 0, the zero submatrix. This shall be assumed in the remaining part of this section. The parallel factorization algorithm can now be reformulated as follows:

Step 1 Perform in parallel the first r steps and s steps of the factorization of

$$\begin{pmatrix} A_R & C_R^T \\ C_R & A_{\partial R} \end{pmatrix} \text{ into } \begin{pmatrix} L_R & \\ C_R L_R^{-T} D_R^{-1} & I_{N-r-s} \end{pmatrix} \begin{pmatrix} D_R & \\ & \bar{A}_{\partial R} \end{pmatrix} \begin{pmatrix} L_R^T & D_R^{-1} L_R^{-1} C_R^T \\ & I_{N-r-s} \end{pmatrix}$$

$$\begin{pmatrix} A_S & C_S^T \\ C_S & A_{\partial S} \end{pmatrix} \text{ into } \begin{pmatrix} L_S & \\ C_S L_S^{-T} D_S^{-1} & I_{N-r-s} \end{pmatrix} \begin{pmatrix} D_S & \\ & \bar{A}_{\partial S} \end{pmatrix} \begin{pmatrix} L_S^T & D_S^{-1} L_S^{-1} C_S^T \\ & I_{N-r-s} \end{pmatrix}$$

respectively, where $\bar{A}_{\partial R} = A_{\partial R} - C_R A_R^{-1} C_R^T$ and $\bar{A}_{\partial S} = A_{\partial S} - C_S A_S^{-1} C_S^T$.

Step 2 Factor $\bar{B} = \begin{pmatrix} \bar{A}_{\partial R} & C_{RS}^T \\ C_{RS} & \bar{A}_{\partial S} \end{pmatrix}$ into $L_{\bar{B}} D_{\bar{B}} L_{\bar{B}}^T$,

where

$$L_{\bar{B}} = \begin{pmatrix} L_{\partial R} & \\ \bar{C}_{RS} & L_{\partial S} \end{pmatrix}.$$

The corresponding lower triangular system

$$\begin{pmatrix} L_R & & & \\ & L_S & & \\ \bar{C}_R & & L_{\partial R} & \\ & \bar{C}_S & \bar{C}_{RS} & L_{\partial S} \end{pmatrix} \begin{pmatrix} y_R \\ y_S \\ y_{\partial R} \\ y_{\partial S} \end{pmatrix} = \begin{pmatrix} b_R \\ b_S \\ b_{\partial R} \\ b_{\partial S} \end{pmatrix}$$

can then be solved by

Step 1 Solve

$$\begin{pmatrix} L_R & \\ \bar{C}_R & I_{\partial R} \end{pmatrix} \begin{pmatrix} y_R \\ b'_{\partial R} \end{pmatrix} = \begin{pmatrix} b_R \\ b_{\partial R} \end{pmatrix}$$

and

$$\begin{pmatrix} L_S & \\ \bar{C}_S & I_{\partial S} \end{pmatrix} \begin{pmatrix} y_S \\ b'_{\partial S} \end{pmatrix} = \begin{pmatrix} b_S \\ b_{\partial S} \end{pmatrix}$$

in parallel.

Step 2 Solve

$$\begin{pmatrix} L_{\partial R} & \\ \bar{C}_{RS} & L_{\partial S} \end{pmatrix} \begin{pmatrix} y_{\partial R} \\ y_{\partial S} \end{pmatrix} = \begin{pmatrix} b'_{\partial R} \\ b'_{\partial S} \end{pmatrix}.$$

The algorithm for solving the corresponding upper triangular system can be similarly formulated. We note that complete independence enables modifications to be performed in parallel. To incorporate the idea of complete independence into our model problem, we modify our nested scheme given in section 3 to the so-called doubly nested dissection.

The mesh M is divided into four regular submeshes by a hollow "+" shaped set of separating vertices. The doubly dissection of the 13 × 13 regular mesh is given in Figure 5.1.

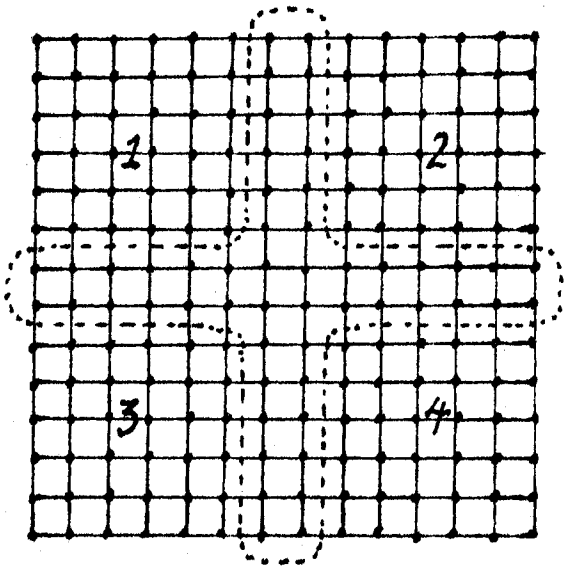

Figure 5.1 Doubly Dissection of 13 × 13 mesh

Variables associated with the four components are completely independent due to the double screening effect of the separator. Recursive application of the dissection technique yields a doubly nested ordering. We shall end this section by stating the following theorem and providing a complete doubly nested ordering for n = 5 in Figure 5.2.

Theorem 5.1 Any finite element system (1.1) associated with M can be solved using the doubly nested dissection in O(n) parallel operations.

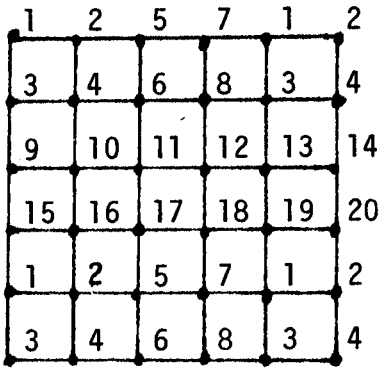

Figure 5.2 A Doubly Nested Scheme for n = 5

6 Concluding Remarks

The nested dissection technique, which is due to George [3], is found to have the essential features of recursive doubling. In this context, we have shown that the finite element system derived from a regular n by n mesh can be solved in O(n) parallel operations, provided that we have a large degree of parallelism. Following Stone [12], we define the speed-up ratio as a measure of efficiency evaluation

$$\text{speed-up ratio} = \frac{\text{computation time on a serial computer}}{\text{computer time on the parallel computer}} .$$

It is known that the direct solution of the linear system (1.1) must take a time proportional to $O(n^3)$ on a serial computer (Hoffman, Martin and Rose [6]). Thus, the speed-up ratio for parallel elimination by nested dissection is proportional to n^2, which indicates that the technique is well suited for parallel computation.

We have also discussed the doubly nested dissection scheme, a technique developed so that parallel eliminations and parallel substitutions of blocks can be carried out in an absolutely independent manner. The corresponding algorithms are simpler and better suited for parallel computation. It is believed to have practical significance.

The same kind of dissection technique can be applied to problems with less regular domains. Computation can be speeded up by eliminating and back substituting (completely) independent sets of variables. However, it would be difficult to analyze the speed-up ratio precisely.

An important assumption in our analysis is the availability of any number of parallel processors. It is not hard to see from the remark after lemma 2.1 that to achieve a complexity bound of $O(n)$, $O(n^2)$-fold parallelism is required in carrying out the symmetric factorization of matrix A and the back solution of $LDL^T x = b$. $O(n^2)$ is discouragingly large when compared with the maximum degree of parallelism allowed in existing parallel computers. For example, the ILLIAC IV computer can perform at most 512 simultaneous computations. Yet our analysis shows how the logical independence of subtasks in solving mesh equations may be exploited to increase the speed of computation. As a matter of fact, the same ideas can be used if we do not have as many as $O(n^2)$ parallel processors. Suppose k processors are available, where k is less than $O(n^2)$. Elimination by nested dissection can still be employed using $O(n^3/k)$ number of parallel operations.

ACKNOWLEDGEMENTS

The author expresses his gratitude to Professor Alan George for suggesting the research topic and for his helpful advice and valuable criticisms.

REFERENCES

[1] G. Birkhoff and A. George, "Elimination by nested dissection", in Complexity of Sequential and Parallel Numerical Algorithms, edited by J.F. Traub, Academic Press, New York and London, 1973.

[2] F.W. Dorr, "The direct solution of the discrete Poisson equation on the rectangle", SIAM Review 12 (1970), p.248-263.

[3] Alan George, "Block elimination on finite element systems of equations", in Sparse Matrices and their Applications, edited by D.J. Rose and R.A. Willoughby, Plenum Press, New York, 1972.

[4] Alan George, "Nested dissection of a regular finite element mesh", SIAM J. Numer. Anal. 10 (1973), p.345-363.

[5] Alan George, "A survey of sparse matrix methods in the direct solution of finite element equations", Proc. Summer Computer Simulation Conference, Montreal, Canada, 1973, p.15-20.

[6] A.J. Hoffman, M.S. Martin and D.J. Rose, "Complexity bounds for regular finite difference and finite element grids", SIAM J. Numer. Anal. 10 (1973), p.364-369.

[7] W.L. Miranker, "A survey of parallelism in numerical analysis", IBM Report RC 2871 (May 1970).

[8] S.V. Parter, "The use of linear graphs in Gauss elimination", SIAM Review 3 (1961), p.119-130.

[9] M.C. Pease, "Matrix inversion using parallel processing", JACM 14 (1967), p.757-764.

[10] D.J. Rose, "A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations", in Graph Theory and Computing, edited by R.C. Read, Academic Press, New York, 1971.

[11] K.L. Stewart and J. Baty, "Dissection of structures", J. Struct. Div., ASCE, Proc. Paper No.5502, (1967), p.217-232.

[12] H.S. Stone, "Problems of parallel computation", in Complexity of Sequential and Parallel Numerical Algorithms, edited by J.F. Traub, Academic Press, New York and London, 1973.