Implementation of Nested Dissection Method Using Block Elimination

Hashim Saber (*)[1]
Department of Mathematics
College of Science and Mathematics
University of North Georgia, USA

Charles Bryan
Department of Mathematics
The University of Montana, USA

Abstract

In this paper, we consider the problem of solving an $n^2 \times n^2$ sparse positive definite system $AU = b$, arising from the use of finite difference methods to solve an elliptic boundary value problem on an $n \times n$ mesh where $n = 2^k - 1$ where k is a positive integer. The large sparse linear system can be solved directly in an efficient way using nested dissection method, originally proposed by Alan George. This paper demonstrates two algorithms for finding orderings using a version of the nested dissection method which leads to block Gaussian elimination of the matrix A. Implementation of these algorithms is pursued and the issue of storage and execution time tradeoffs is discussed.

Keywords: Nested Dissection, Direct Method, Elliptic Boundary Value Problem.

1. Introduction

1.1 Problem formulation.

This paper presents a fast solver for non-homogeneous variable coefficient boundary value problem (BVP) in which the elliptic operator is self-adjoint and linear:

$$\frac{\partial u}{\partial x}\left[F(x,y)\frac{\partial u}{\partial x}\right] + \frac{\partial u}{\partial y}\left[G(x,y)\frac{\partial u}{\partial y}\right] = -B(x,y) \qquad \text{in } R = (0,1)\times(0,1) \qquad (1a)$$

$$u(x,y) = g(x,y) \qquad \text{on } \partial R \qquad (1b)$$

1.2 Discretization

The method we describe is applicable to a variety of geometries and discretization schemes (finite elements, finite differences, etc.). In this paper, we restrict our attention to the model problem where a square domain is discretized via a finite difference scheme on a regular $n \times n$ square mesh. The resulting system of $N(=n^2)$linear equations takes the matrix form

$$A\,\mathrm{U} = \mathrm{b} \qquad (2)$$

Where A is $N \times N$ matrix and the vector U consists of the unknown quantities $U_{i,j} \sim u(ih, jk)$. The quantities $h^2 B_{i,j}$ together with the values of u on the boundary are the entries of b.

The main goal of this paper is a detailed study of the solution of problem (1) when a particular "nested dissection" method is used to order the unknown variables. Nested dissection is a technique due to George [5] for systematically partitioning the graph associated with a matrix using separators (A ***separator*** is a set of vertices, the removal of which, together with their incident edges, disconnects an otherwise connected graph component).

[1] **(*)**This paper is dedicated to my advisor Professor Charles Bryan whom I am especiallygrateful for his patience and encouragement at the time I needed them most. Dr. Bryan, you have set an example of excellence as a researcher, mentor, and role model.

When a separator is found, its vertices are segregated; the graph spanned by the remaining vertices has two or more components. Separators for these components are then found and segregated.

The procedure continues forming smaller and smaller components until separators can no longer be found. Then all vertices are numbered, beginning with the last identified ones and proceeding "backward". The matrix canthen be permuted accordingly.

A brief survey of nested dissection methods is presented in this section. Section 2 presents a version of nested dissection ordering and the structure of the matrix *A* in equation (2) if the unknowns are ordered accordingly. An ordering algorithm for that version of nested dissection is explicitly given. Our approach in factoring the matrix *A* and solving the system in equation (2) is given in section 3. Section 4 deals with the number of arithmetic operations and storage requirements in implementation of the method described in section 3. In section 5, we give some concluding remarks.

The concept of Georg's nested dissection ordering first appeared in George [5]. He presented an ordering for the $n \times n$ grid in the plane that leads to a system of *N* linear equations in equation (2), where $N = n^2$. Many articles followed which generalized the effectiveness of his idea, combined his method with other well-known methods, applied it to three-dimensional problems, and introduced similar but more applicable ideas. A brief description of some of the articles is given below.

For regular $n \times n$ grids, George [5] introduced the nested dissection ordering and gave a precise recommendation for the $n \times n$ grids, with $n = 2^k + 1$ (*k* is positive integer), which resulted in $O(N^{\frac{3}{2}})$ arithmetic operations and $O(N \log N)$ storage [compared with $O(N^3)$ and $O(N^2)$ for operation counts and storage, respectively, for the usual row (band) numbering scheme] where $N = n^2$ is the number of the unknowns. Also he proved that all orderings of the mesh must yield an operation count of at least $O(N^{\frac{3}{2}})$, if the standard factorization algorithm has been used.

Birkhoff and George [1] generalized George nested dissection [which is typically concerned with a planar mesh consisting of either triangles or quadrilaterals treated as finite elements of a plane] to any grid. Their analysis is applied more generally to any linear system whose K^{th} equation refers to the K^{th} unknown; for example, a set of difference equations. They considered nested dissection in terms of ***partial orderings*** of partitions of the corresponding system's graph into disconnected components. Also, they gave a new and more general definition for the process of elimination by nested dissection in terms of a sequence of partitions of the graph of the matrix to be factored. They provided some simple examples showing that the order of magnitude of the operation-count is the same for all plane domains. Moreover, they gave (incomplete) arguments which indicate that nested dissection is numerically stable.

Examining the original work of the nested dissection method, Duff, Erisman and Reid [2] considered the case when *n* is not a power of 2. They introduced an extension which appears to remove any advantage of requiring *n* to be a power of 2. Also, they considered the use of line-shaped dissection sets, instead of cross shaped sets and found that they do have a slight disadvantage. They evaluated storage and operation counts for George's ordering of three dimensional grids of the side $n = 2^k$ and concluded that the iterative techniques are superior in this case. Also, they compared nested dissection on a square grid with the minimum degree algorithm. And for the 9 point finite element case they concluded that nested dissection shows a significant advantage over the minimum degree algorithm.

George [6] described how the ordering of the $n \times n$ grid problem can be implemented in an efficient manner for the two orderings given in [4] and [5], and provided numerical experiments which show that the execution times of these programs properly reflects the arithmetic operation counts. Also, he compared the performance of these programs with respect to the execution time and storage. The two ordering schemes, nested dissection and one-way nested dissection, were discussed.

George, Poole, and Voigt [10] studied the consequences of terminating the mesh subdivision before completion; that is, at some stage one does not subdivide the quadrants further, but simple uses row by row, band-oriented ordering for the nodes in each quadrant.

This idea is related to the idea of sub-structuring in structural engineering applications [12]. Analysis of the arithmetic and storage requirements for this technique is given and the ordering is shown to be competitive with nested dissection with regard to arithmetic operations and superior to that ordering in storage requirements. Also, it was shown that there is no practical advantage in carrying the dissection to completion. The analysis was carried out for a square region and for a general region. The authors proposed that a general region may be subdivided (or sub-structured) into a union of smaller regions, many of which will be squares if the subdivision is chosen appropriately.

George [7] provided an automatic scheme for finding orderings analogous to the one way dissection ordering. The storage requirements for these orderings appear to grow as $N^{\frac{5}{2}}$, and such orderings are inferior to nested dissection orderings, whose storage requirements only grow as $O(N \log N)$ $\left[N = n^2 \text{ for the } n \times n \text{ grid}\right]$.These estimates are asymptotic, and unless N is very large indeed, the one-way dissection ordering appears to require considerable less storage than the nested dissection orderings. In exchange for lower storage requirements, this method performs more arithmetic than the nested dissection, so the automatic determination of such one-way orderings for irregular problems is important when storage is limited. Also, a heuristic algorithm is described for finding one-way dissection orderings for sparse matrix problems, and some numerical experiments describing its application to some finite element problems were provided.

George and Liu [9] provided an automatic algorithm for producing nested dissection orderings for ***irregular***finite element problems. A heuristic algorithm is describe for finding a nested dissection ordering for an undirected graph, along with an appropriate data structure and a storage allocation scheme for a linear equation solver to such orderings. Numerical experiments are provided which indicate that their combinations of orderings and solution schemes, is superior to standard band or envelope schemes as long as the problems are moderately large.

Finally, Lipton, Rose, and Tarjan [11] introduced a generalization of the nested dissection algorithm (automatic nested dissection). In factoring A to LL^T the algorithm executes in $O(|A| \log |A|)$ time and $O(|A|)$ space, where $|A|$ is the number of non-zero entries in the matrix A, and produce an ordering for which $|A|$ is $O(|A| \log |A|)$ and they claimed all of these bounds optimal. This algorithm is described in George [8], "unfortunately, this algorithm is fairly complicated, and apparently its implementation has not yet been developed. This algorithm is also restricted to planar problems since it makes explicit use of planarity." Their generalization was achieved without degrading the time and space bound so that it applies to any system of equations defined on planar or almost-planar graphs. The method uses the fact that planar graphs have good separators (as small as possible). They also showed that Gaussian Elimination is efficient for any class of graphs which have good separators and, conversely, that graphs without good separators (including "almost all" sparse graphs) are not amenable to sparse Gaussian Elimination.

2. Ordering Scheme and the Matrix Structure

The numbering procedure we use is a version of nested dissection which can be considered as a method for systematically partitioning the graph associated with a matrix using separators.

Consider numbering the region R_{k+1} of size $n_{k+1} \times n_{k+1}$ where $n_k = 2^k - 1$. We call such region a ***square region of type k+1*** (Figure 1). The separator we choose is a cross (called ***cross of type k***), as shown in the figure, consists of four arms indicated by $A_i^K, i \in I_4$ and a center node C_k. The four arms of size $n_k \times 1 \text{ or } 1 \times n_k$ will be referred to as ***arms of type k***. If the vertices of this cross are virtually removed from the graph, they leave it partitioned into four components, namely, R_k^1, R_k^2, R_k^3, and R_k^4. Each of these four components is a square region of type k R_k of size $n_k \times n_k$ where its separator which is a cross can be found and the procedure continue until each of the last components is a simple single node 1×1.

Numbering the region R_{k+1} consists of assigning a number to each node (x_i, y_j), with x-coordinate x_i and y-coordinate y_j, we simply refer to a node (x_i, y_j) by (i, j) and define $\{M_{k+1}(i, j) : \text{where } 1 \le i, j \le n_{k+1}\}$ to be the set of numbers associated with the nodes (x_i, y_j) in R_{k+1}. First we define an array $a^k(p)$ with $1 \le p \le n_k$ as follows:

Given $a^{k-1}(p),\ 1 \le p \le n_{k-1}$ with $a^1(1) = 1$, then:

$$a^k(p) = \begin{cases} a^{k-1}(p) & 1 \le p \le n_{k-1} \\ a^k(n_k - p + 1) + n_{k-1} & n_{k-1} + 2 \le p \le n_k \\ n_k & p = n_{k-1} + 1 \end{cases}$$

2.1 The numbering algorithm

We define the numbering $\{M_{k+1}(i,j)$: where $1 \le i, j \le n_{k+1}\}$ of a square region R_{k+1} of size $n_{k+1} \times n_{k+1}$ as follows: Given the numbering $\{M_k(i,j)$: where $1 \le i, j \le n_k\}$ mesh with $M_1(i,j) = 1$, and $a^k(p)\ \ 1 \le p \le n_k$, (where $a^k(p)$ is defined above). Then the numbering $\{M_{k+1}(i,j)$: where $1 \le i, j \le n_{k+1}\}$ of the region R_{k+1} can be described in the following steps:

Step 1: Numbering the four square regions R_k^1, R_k^2, R_k^3, and R_k^4 **:**

$$M_{k+1}(i,j) = M_k(i,j);\ 1 \le i \le n_k,\quad 1 \le j \le n_k$$
$$M_{k+1}(i,j) = M_k(n_{k+1} - i + 1, j) + n_k^2;\ n_k + 2 \le i \le n_{k+1},\quad 1 \le j \le n_k$$
$$M_{k+1}(i,j) = M_k(i, n_{k+1} - j + 1) + 2n_k^2;\ 1 \le i \le n_k,\ n_k + 2 \le j \le n_{k+1}$$
$$M_{k+1}(i,j) = M_k(n_{k+1} - i + 1, n_{k+1} - j + 1) + 3n_k^2;\ \ n_k + 2 \le i, j \le n_{k+1}$$

Step 2: Numbering the four arm regions A_k^1, A_k^2, A_k^3, and A_k^4 **:**

$$M_{k+1}(i,j) = a^{k+1}(j) + 4n_k^2;\quad i = n_k + 1,\quad 1 \le j \le n_k$$
$$M_{k+1}(i,j) = a^{k+1}(j) + 4n_k^2 + n_k;\quad i = n_k + 1,\quad n_k + 2 \le j \le n_{k+1}$$
$$M_{k+1}(i,j) = a^{k+1}(j) + 4n_k^2 + 2n_k;\quad 1 \le i \le n_k,\quad j = n_k + 1$$
$$M_{k+1}(i,j) = a^{k+1}(j) + 4n_k^2 + 3n_k;\quad n_k + 2 \le i \le n_{k+1},\ j = n_k + 1$$
$$M_{k+1}(n_k, n_k) = n_{k+1}^2$$

In Figure 2, we give the region numbering of two examples where in the first the region is R_2 divided into 9 sub-regions and the second is R_3 which is divided into 49 sub-regions.

The zero-non-zero pattern of the matrices induced by numbering the regions in Figures 2-a andFigure 2-b are shown in Figures 3 and 4, respectively. The areas in which the non- zero entries are confined are shaded with cross hatching. This arrangement of the matrix has the important property that the fill-in caused by Gaussian Elimination with diagonal pivoting is also confined to the cross hatched areas plus the shaded squares.

3. The LL^T Factorization

If we number the nodes as described in section 2, the collection of equations can be written as the system:

$$A^{k+1}U = b$$

Where A^{k+1} is an $N_{k+1} \times N_{k+1}$ symmetric, positive definite matrix with $N_{k+1} = n^2{}_{k+1}$. $[N_{k+1} = 4N_k + 4n_k + 1]$. We partition A^{k+1} into 9 blocks by 9 blocks arising from the partition $N_{k+1} = 4N_k + 4n_k + 1$. The various blocks may be named A_i^k, V_i, H_i, etc. , as indicated in Figure 1 where the blank blocks are zero blocks. In this partitioning, each A_i^k is an $N_k \times N_k$ matrix and again can be partitioned into 9 blocks by blocks as in the definition above. Each C_i is $n_k \times n_k$, and s is a 1×1 matrix or scalar. The matrix A^{k+1} can be factored as L_{k+1} where $A^{k+1} = L_{k+1}L_{k+1}^T$ is an $N_{k+1} \times N_{k+1}$ lower triangular matrix with positive diagonal elements, and L_{k+1}^T is its transpose.

This is Cholesky's factorization and the process of finding L_{k+1} by an efficient algorithm using the structure of A^{k+1} is the concern of this section. For $1 \le j \le k$, we refer to factoring A^{k-j+1} as level *j* factorization.

Upon examining the block structure of A^{k+1}, we obtain the analogous block structure of L_{k+1} (Figure 1). Note that:

$$K_i = V_i^T,\ J_i = H_i^T,\ F_i = W_i^T,\ G_i = Z_i^T,\ N_i = M_i^T\ 1 \le i \le 4,$$
$$P_i = X_i^T\ 1 \le i \le 2,\ Q_i = Y_i^T\ 1 \le i \le 3,$$

With this block structure and assuming we have completed the factorization process for levels *j* where $j = k, k-1, \ldots, 2$ we obtain:

$$W_i = L_i^{-1} V_i\ 1 \le i \le 4$$
$$Z_i = L_i^{-1} H_i\ 1 \le i \le 4$$
$$E_5 E_5^T = C_1 - W_1^T W_1 - W_2^T W_2$$
$$E_6 E_6^T = C_2 - W_3^T W_3 - W_4^T W_4$$
$$P_1 = -E_5^{-1} W_1^T Z_1,\ Q_1 = -E_5^{-1} W_2^T Z_2,\ N_1 = E_5^{-1} R_1^T$$
$$P_2 = -E_6^{-1} W_3^T Z_3,\ Q_2 = -E_6^{-1} W_4^T Z_4,\ N_2 = E_6^{-1} R_2^T$$
$$E_7 E_7^T = C_3 - Z_2^T Z_2 - Z_4^T Z_4 - P_1^T P_1 - P_3^T P_3$$
$$Q_3 = -E_7^{-1} W_1 \left(Q_1^T Q_1 - Q_2^T Q_2 \right)$$
$$E_8 E_8^T = C_4 - Z_2^T Z_2 - Z_4^T Z_4 - Q_1^T Q_1 - Q_2^T Q_2 - Q_3^T Q_3$$
$$N_3 = E_7^{-1} \left(R_3^T - Q_1^T N_1 - Q_2^T N_2 \right)$$
$$N_4 = E_8^{-1} \left(R_4^T - Q_1^T N_1 - Q_2^T N_2 - Q_3^T N_3 \right)$$
$$r = s - M_1 N_1 - M_2 N_2 - M_3 N_3 - M_4 N_4,$$

and E_i is an $n_k \times n_k$ lower triangular matrix for $1 \le i \le 8$.

Continuing with the Cholesky method, we solve the system $L_{k+1} S = b$ (forward solution), then $L^T{}_{k+1} U = S$ (backward solution) with $b, U,$ and S partitioned in a manner consistent with L_{k+1},

$$S_i = L_i^{-1} b_i\ 1 \le i \le 4$$
$$S_5 = E_5^{-1} \left(b_5 - W_1^T S_1 - W_2^T S_2 \right)$$
$$S_6 = E_6^{-1} \left(b_6 - W_3^T S_3 - W_4^T S_4 \right)$$
$$S_7 = E_7^{-1} \left(b_7 - Z_1^T S_1 - Z_3^T S_3 - P_1^T S_5 - P_2^T S_6 \right)$$
$$S_8 = E_8^{-1} \left(b_8 - Z_2^T S_2 - Z_4^T S_4 - Q_1^T S_5 - Q_2^T S_6 - Q_3^T S_7 \right)$$
$$S_9 = \frac{1}{r} \left(b_9 - M_1 S_5 - M_2 S_6 - M_3 S_7 - M_4 S_8 \right); \text{ then}$$
$$U_9 = S_9 / r\ U_8 = E_8^{-T} \left(S_8 - N_4 U_9 \right)$$
$$U_7 = E_7^{-T} \left(S_7 - Q_3 U_8 - N_3 U_9 \right)$$
$$U_6 = E_6^{-T} \left(S_6 - P_2 U_7 - Q_2 U_8 - N_2 U_9 \right)$$
$$U_5 = E_5^{-T} \left(S_5 - P_1 U_7 - Q_1 U_8 - N_1 U_9 \right)$$
$$U_4 = L_4^{-T} \left(S_4 - W_4 U_6 - Z_4 U_8 \right)$$
$$U_3 = L_3^{-T} \left(S_3 - W_3 U_6 - Z_3 U_7 \right)$$

$$U_2 = L_2^{-T}\left(S_2 - W_2U_5 - Z_2U_8\right)$$

$$U_1 = L_1^{-T}\left(S_1 - W_1U_5 - Z_1U_7\right)$$

Implementing our LL^T –factorization requires three major steps.

Step (1)

Generating WZ-matrices, that is generating by W_i and Z_i, $1 \le i \le 4$ and

Step (2)

A multiplication step which involves multiplying $Z_i^T Z_i, W_i^T W_i,$ and $W_i^T Z_i$; $1 \le i \le 4$.

Step (3)

Using the result of step (2) to generate the LL^T factorization; this is a straightforward step which involves LL^T factorization of a full matrix, multiplying two full matrices and adding/subtraction matrices.

Remark:

There are two different approaches of factoring the matrix A which we refer to Algorithm (I) and Algorithm (II). In Algorithm (I), we retain all the WZ-matrices corresponding to level *k* through *k+2-j* and use them to generate the WZ-matrices for the k+1-j level. While in Algorithm (II), we discard the generated WZ-matrices as soon as we have used them to generate the other blocks of *L* corresponding to the current level. In this case, we need to reproduce all necessary entries in the WZ-matrices for levels *k* through *k+2-j* and use them when calculating the WZ-matrices of the k+1-j level.

4. Operation Count and Storage Requirements

Our measure of operations count consists of two parts, the number of multiplicative operations (multiplication and division) required to factor *A* into LL^T and that of solving $LL^T x = b$. We regard the multiplicative operations as a reasonable measure since the required numbers of additions and subtractions is about the same; moreover, the factorization is typically the major portion of the computation. In the following, "operation" will mean multiplicative operation. The measure of storage requirements is the number of non-zero off-diagonal and diagonal entries of the matrix *L* which have to be stored in order to carry out the solution of equation (1) as described in section 3.

Consider generating the LL^T factorization of *A* in $Ax = b$ where A is an $N_k \times N_k$ matrix (this is the 2nd level factorization according to the definition given in section 3 with $n_k = 2^k - 1$.We assume $n_k \cong \frac{n_{k+1}}{2}$). The block structures of *A* and *L* were given in Figure 4. It is important to recall from section 3 that rather than saving the entire factorization, we shall store only $E_i, M_i, 1 \le i \le 4$, $X_i, 1 \le i \le 2$, $Y_i, 1 \le i \le 3$(referred to as RF-blocks), and discard most of the non-zero entries of *L*, namely L_iV_i and L_iH_i $1 \le i \le 4$. We recomputed these as needed during the forward-backward final solution or generate the next level's factorization blocks if desired.

Now suppose we were to perform the *k-j+1* level, $1 \le j \le k$, factorization of (1) in such a way that we have calculated and saved the RF-blocks corresponding to all previous levels $(\text{levels } k, k-1, \ldots, k-j)$; then we first need to generate the matrices L_iV_i and L_iH_i $1 \le i \le 4$ in order to generate E_i and the other RF-blocks. Let*F (n)* = Number of operations to factor the $n^2 \times n^2$ matrix *A* with $n_k = n$.

4.1 Calculating *F (n)*:

The following table indicates the types of operations, number of times they occurred, and their operations count. These results were derived from the factorization equations given in the beginning of section 3.

From the table below, we derive the following recursive equation for Algorithm (II):

$$F(n) = 4F\left(\frac{n}{2}\right) + \frac{3341}{336}n^3 + \frac{1358}{252}n^{\log_2 6} - \frac{99}{16}n^2 + O(n\log_2 n),$$

and with $F(1)=1$, the solution of this recurrence relation is

$$F(n)=\frac{3341}{504}n^3+\frac{1358}{84}n^{\log_2 6}-\frac{3173}{504}n^2-\frac{99}{16}n^2\log_2 n+O(n\log_2 n)$$

Operation	No. of Times Occurred	Operations Count
Factor lower level case	4	$4F(n/2)$
Solve LQ = B where B is a full n/2 x n/2 matrix	5	$5(n/2)[\frac{1}{2}\{(n/2)^2+(n/2)\}]=(5/16)n^3+(5/8)n^2$
Factor a full n/2 x n/2 matrix into LL^T	4	$4[(1/6)(n/2)^3+\frac{1}{2}(n/2)^2+O(n)]=n^3/12+n^2/2+O(n)$
Special multiplications of type y_2, y_3	4	$[4y_2(n/2)+4y_3(n/2)]=(17/12)n^3+5n^2+O(n\log_2 n)$
Special multiplications of type y_6	4	$4[(10/7)(n/1)^3+(8/3)(n/2)^2+10(n/2)+0(1)=(5/7)n^3+(8/3)n^2+0(n)$
Generate a W or Z-matrix	4	$4[W(n/2)+Z(n/2)]=\frac{7476}{1008}n^3+\frac{5432}{1008}n^{\log_2 6}-\frac{275}{12}n^2+O(n\log_2 n)$
Solve Lx=v where L is n/2 x n/2 full	4	$4\frac{1}{2}(n/2)^2$
Multiply GX where G is n/2 x n/2 matrix and X is n/2 -vector	5	$5(n/2)^2$
Other operations		$O(n)$

As a result, we have the following theorem

Theorem:

The number of operations to factor an $n^2\times n^2$ matrix *A* using the method explained in sections 2 & 3 is given by

$$F(n)=\frac{3341}{504}n^3+\frac{1358}{84}n^{\log_2 6}-\frac{3173}{504}n^2-\frac{99}{16}n^2\log_2 n+O(n\log_2 n)$$

4.2 Storage requirements

Let $\Theta(n)$ be the storage requirement to store the necessary entries in an $n^2\times n^2$ matrix L. If we choose to store the RF-matrices relative to all levels as explained in section 3, then we have the following recursive relation:

$$\Theta(n)=4\Theta\left(\frac{n}{2}\right)+7\left(\frac{n}{2}\right)^2+6\left(\frac{n}{2}\right)+1$$

$$\Theta(1)=1$$

The solution of this recursive relation is

$$\Theta(n) = \frac{7}{4}n^2 \log_2 n - \frac{5}{4}n^2 - \frac{3}{2}n.$$

Theorem

The required number of non-zero entries in the lower triangular factor L of the matrix A factored as described in section 3 is given by:

$$\Theta(n) = \frac{7}{4}n^2 \log_2 n - \frac{5}{4}n^2 - \frac{3}{2}n$$

4.3 Operations count to solve $LL^TU = b$:

Let $L(n)$ be the number of operations to solve $LS = b$ (forward solution given in section 3) where L is $n \times n$ lower triangular matrix, S and b are n-vectors. Let $U(n)$be the number of operations to solve $L^TU = S$ (backward solution given in Chapter 3) where L^T is the transpose of L and U is n-vector. Let $S(n)$ be the number of operations to solve $LL^TU = b$. So$S(n) = L(n) + U(n)$.

Before stating the recurrence relation to generate $S(n)$, we need to explain how we generate $W_4U_6 + Z_4U_8$ and $W_1^T S_1$ in the forward and backward formulas in section 3 and then find their required number of operations where U_i, S_j are $\frac{n}{2} \times \frac{n}{2}$ matrices and W1 is an $\left(\frac{n}{2}\right)^2 \times \frac{n}{2}$ matrix as described in section 3.

Operation	No. of Times Occurred	Operations Count
Generate $W_i^T S_i$	4	4[n/2+U(n/2)]
Solve $L_i^{-1}B$	4	4 L(n/2)
Generate $Q_2 * S$ where Q is n/2 n/2 and S is n/2-vector	5	5 $(n/2)^2$
Solve $E^{-1} * B$ where E is n/2 x n/2 lower triangle matrix	4	4[½$(n/2)^2$3+½(n/2)]
Multiply M_i * X_i where M,S are n/2-vectors	4	4(n/2)

As a result we have

$$L(n) = 4L\left(\frac{n}{2}\right) + 4U\left(\frac{n}{2}\right) + \frac{7}{4}n^2 + 11n + 1$$

Similarly we have

$$U(n) = 4L\left(\frac{n}{2}\right) + 4U\left(\frac{n}{2}\right) + \frac{7}{4}n^2 + 11n + 1$$

With

$S(n) = L(n) + U(n)$, the recursive relation for $S(n)$is

$$S(n) = 8S\left(\frac{n}{2}\right) + \frac{7}{4}n^2 + 22n + 2$$

The solution to the recurrence relation is:

$$S(n) = \frac{397}{42}n^3 - \frac{7}{2}n^2 - \frac{22}{3}n - \frac{2}{7}$$

Theorem:

The number of operations to solve $LL^TU = b$, where L is $n^2 \times n^2$ lower triangular matrix is given by

$$S(n) = \frac{397}{42}n^3 - \frac{7}{2}n^2 - \frac{22}{3}n - \frac{2}{7}$$

4.4 Remark:

If we implement Algorithm (I) where we choose to store the WZ-matrices corresponding to all levels, then we have the following results:

1. Operations count for LL^T factorization.
$$F(n) = \frac{5039}{504}n^3 + \frac{875}{24}n^2 \log_2 n - \frac{4535}{504}n^2 + O\left(n(\log_2 n)^2\right)$$
2. Storage requirements.
$$\Theta(n) = \frac{31}{4}n^2 \log_2 n - 7n^2 - \frac{5}{8}n \log_2 n$$
3. Operations count to solve $LL^TU = b$.
$$S(n) = \frac{7}{42}n^2 \log_2 n - \frac{3}{2}n^2 - +O(n)$$

5. Conclusion

We have described a version of the nested dissection method to approximate the solution of an elliptic boundary value problem with variable coefficients (equation 1). The region is divided into $n \times n$ mesh where $n = 2^k - 1$. We also designed an algorithm to implement the method by breaking the computations up into three steps: setting up the problem which includes numbering the mesh and generating the input matrix, matrix factorization, and forward-backward solution. We found that the choice of $n = 2^k - 1$ (compared with $n = 2^k + 1$ in George [4]) leads to techniques which can be defined recursively and hence systematically. This rewarded us with two main advantages. The first is a well-structured partitioning of the matrix which allows us to proceed with block elimination. The use of the partitioned matrix in the solution of (2) allowed us to design the efficient factorization and solution procedures. The second advantage is that it allows us to designate the location of the non-zero entries of L in advance. This is an important feature in designing an efficient way to store L.

Theoretical results in Section 5 are summarized in Table 1. As we see from these results, Algorithm II requires more operations to perform than Algorithm I while Algorithm I requires more storage. In deciding which one to use, we also need to know whether there is more than one right hand side involved to the matrix problem. Based on these limitations and the results in Table 1, one can decide which algorithm to use.

Table (1)

Algorithm	Operations Count: Factorization	Operations Count: Solution	Storage Requirements
Algorithm I where we store all entries of L	$\frac{5039}{504}n^3 + \frac{875}{34}n^2\log_2 n - \frac{4535}{504}n^2 + 0(n)$	$(7/2)n^2\log_2 n - (3/2)n^2 + 0(n)$	$(31/4)n^2\log_2 n - 7n^2 - (5/8)n\log_2 n$
Algorithm II where we throw away some of the entries of L and recompute them when needed	$\frac{3341}{168}n^3 + \frac{1358\log_2 6}{168}n - \frac{3173}{168}n^2 - \frac{99}{16}n^2\log_2 n + 0(n\log_2 n)$	$\frac{397}{42}n^3 - (7/2)n^2 - \frac{18}{3}n + 0(n)$	$(7/4)n^2\log_2 n - (5/4)n^2 - (3/2)n$

$$A^{k+1} = \begin{bmatrix} A_1^k & & & & V_1 & & H_1 & \\ & A_2^k & & & V_2 & & & H_2 \\ & & A_3^k & & & V_3 & H_3 & \\ & & & A_4^k & & V_4 & & H_4 \\ K_1 & K_2 & & & C_1 & & & R_1^T \\ & & K_3 & K_4 & & C_2 & & R_2^T \\ J_1 & & J_3 & & & & C_3 & R_3^T \\ & J_2 & & J_4 & & & C_4 & R_4^T \\ & & & & R_1 & R_2 & R_3 & R_4 \quad s \end{bmatrix}$$

$$L_{k+1} = \begin{bmatrix} L_1 & & & & & & & \\ & L_2 & & & & & & \\ & & L_3 & & & & & \\ & & & L_4 & & & & \\ F_1 & F_2 & & & E_5 & & & \\ & & F_3 & F_4 & & E_6 & & \\ G_1 & & G_3 & & X_1 & X_2 & E_7 & \\ & G_2 & & G_4 & Y_1 & Y_2 & Y_3 & E_8 \\ & & & & M_1 & M_2 & M_3 & M_4 \quad r \end{bmatrix} \qquad L_{k+1}^T = \begin{bmatrix} L_1^T & & & & W_1 & & Z_1 & \\ & L_2^T & & & W_2 & & & Z_2 \\ & & L_3^T & & & W_3 & Z_3 & \\ & & & L_4^T & & W_4 & & Z_4 \\ & & & & E_5^T & & P_1 & Q_1 \quad N_1 \\ & & & & & E_6^T & P_2 & Q_2 \quad N_2 \\ & & & & & & E_7^T & Q_3 \quad N_3 \\ & & & & & & & E_8^T \quad N_4 \\ & & & & & & & r \end{bmatrix}$$

Figure 1

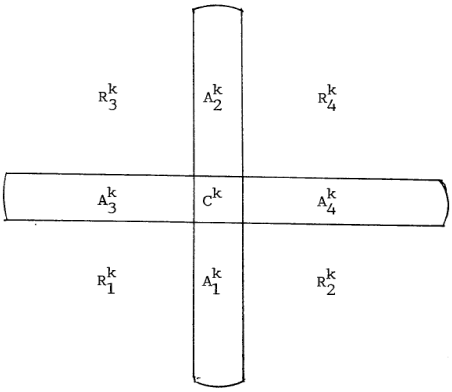

Figure 1
The Region R_{k+1} of Size n_{k+1} X n_{k+1}

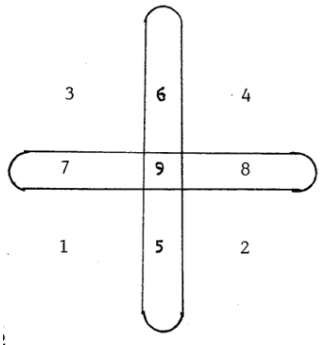

Figure 2b
Numbering the 3x3 Region R_2

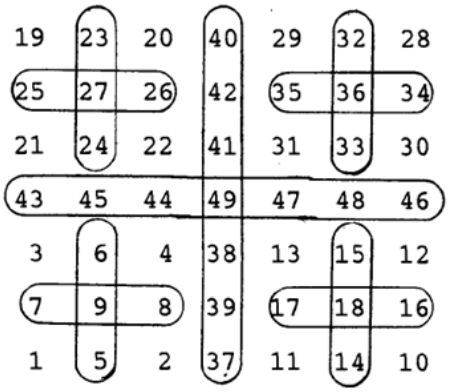

Figure 2a
Numbering the 7x7 Region R_3

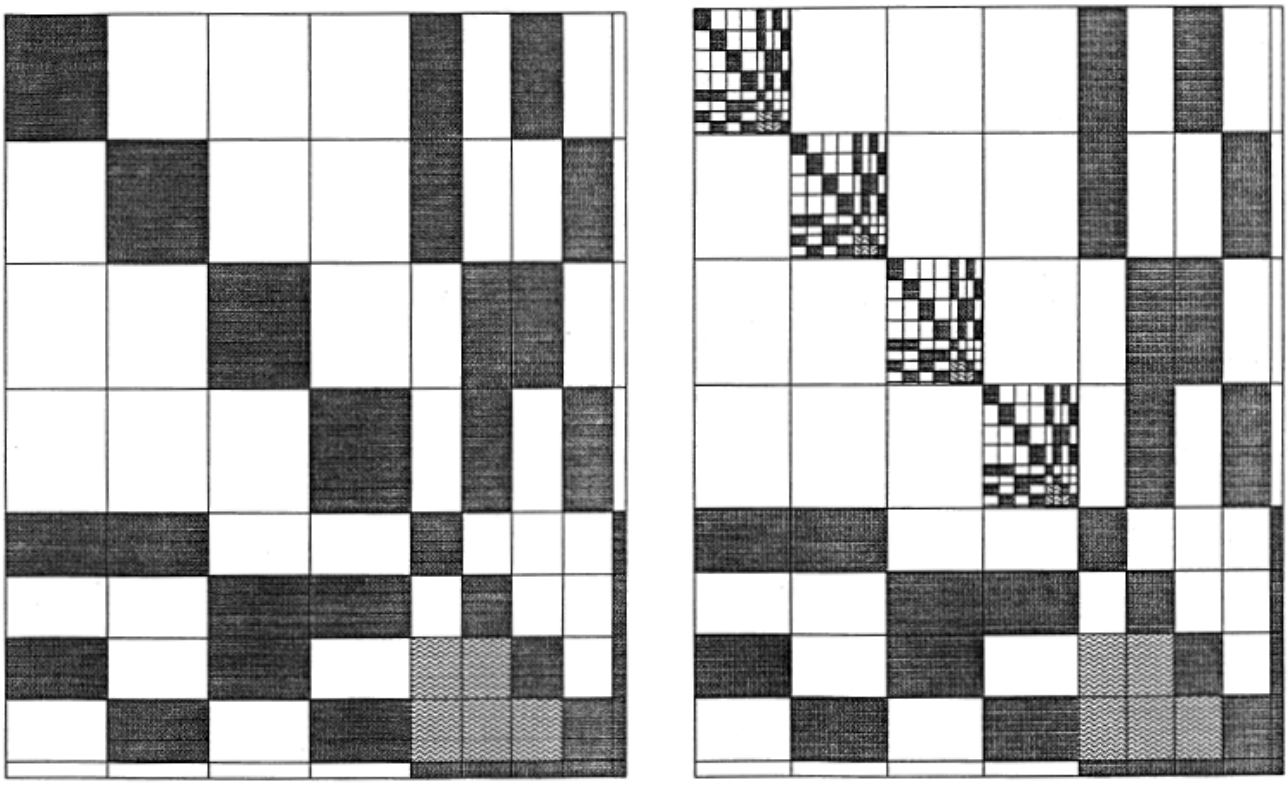

Figure 4

References

Birkhoff, G. and George, Alan. (1973) *"Elimination by Nested Dissection"*. InTraub (1973).

Duff, I.S., Erisman, A.M. and Ried, J.K. (1976) *"On George's Nested Dissection Method"*. SIAM J.Num.Anal.,Vol.13, No.5, October 1976.

Duff, I.S., (1981) *"Sparse Matrices and Their Uses"*. Proc. IMA Numer. Anal. Cont., University of Reading (U.K.), July 1980. Academic Press - London

George, A. (1972) *"An Efficient Band-Oriented Scheme for Solving $n \times n$ Grid Problems"*. Proc. 1972 Fall Joint Computer Conference, AEIPS Press, Montrale, N.J., pp. 1317-1321.

George, A. (1973) *"Nested Dissection of a Regular Finite Element Mesh"*. SIAM J. Numer. Anal. 10, pp. 345-363.

George, A. (1977) *"Numerical Experiments Using Dissection Methods to Solve $n \times n$ Grid Problems"*. SIAM J. Number. Anal., Vol. 14, No. 2, April 1977

George, A. (1977) *"Automatic One-Way Dissection. Algorithm for Irregular Finite Element Problems"*. Proc. 7th Biennial Conf., Univ. of Dundee (1977)

George, A. (1981) *"Direct Solution of Sparse Positive Definite Systems: Some Basic Ideas and Open problems"*.In Duff (1981).

George, A. and Liu, J.W. (1978) *"An Automatic Nested Dissection Algorithm for Irregular Finite Element Problems"*. SIAM J. Numer. Anal., Vol. 15, No. 5, August 1978

George, A., Poole, W.G. and Voigt, R.G. (1978) *"Incomplete Nested Dissection for Solving $n \times n$ Grid Problems"*. SIAM J. Numer. Anal., Vol. 15, No. 4, August 1978

Lipton, R.J., Rose, D.J. and Tarjan, R.E. (1979) *"Generalized Nested Dissection"*. SIAM J. Numer. Anal. 16, pp. 346-358.

Noor, A.K., Kamal, H.A. and Fulton, R.E. (1978) *"Substructuring Projections. J. Computers and Structures"*.

Traub, J.F. (1973) *"Complexity of Sequential and Parallel Numerical Algorithms"*. Proc. Of Conf. in Pittsburgh (Carnegie Mellon); Academic Press.