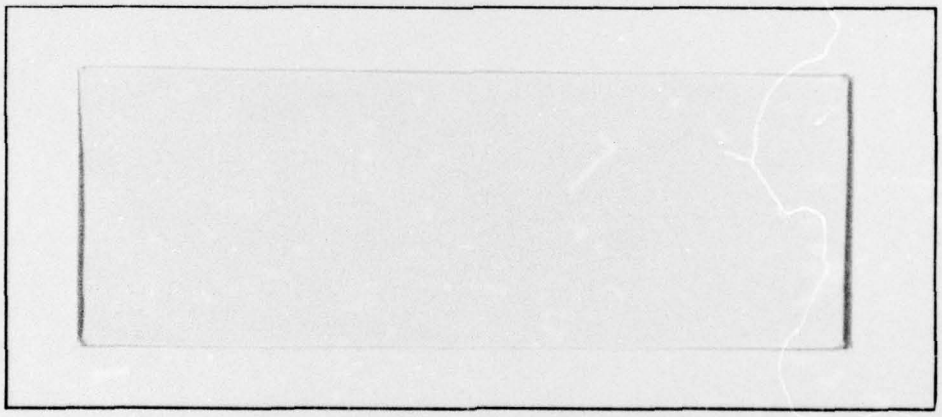
AD-A033 485 YALE UNIV NEW HAVEN CONN DEPT OF COMPUTER SCIENCE F/G 9/2
APPLICATIONS OF AN ELEMENT MODEL FOR GAUSSIAN ELIMINATION.(U)
1975 S C EISENSTAT, M H SCHULTZ N00014-67-A-0097-0016
UNCLASSIFIED RR-54 NL

1 OF 1
AD
A033485

END

DATE
FILMED
2-77

(12)



YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

(12)

A. H. Sherman is at the Department of Computer Science, University of Illinois, Urbana, Illinois 61801.

(9) Research (Rept.)

(15) N00014-67-A-0097-0016,
NSF-GJ-43157

(11) 1975

(6) Applications of an Element Model for Gaussian Elimination,

(10) S. C. Eisenstat, M. H. Schultz
A. H. Sherman

Research Report #54

(14) RR-54

(12) 14 p.

This work was supported in part by NSF Grant GJ-43157 and ONR Grant N00014-67-A-0097-0016.

407051

1. *Introduction*

Consider the system of linear equations

(1.1) $$A x = b$$

where A is an N×N sparse symmetric positive definite matrix such as those that arise in finite difference and finite element approximations to elliptic boundary value problems in two and three dimensions. A classic method for solving such systems is Gaussian elimination: We use the k*th* equation to eliminate the k*th* variable from the remaining N-k equations for k = 1,2,...,N-1 and then back-solve the resulting upper triangular system for the unknown vector x. Equivalently, we form the U^TDU decomposition of A and successively solve the triangular systems

(1.2) $$U^T z = b,\ D y = z,\ U x = y$$

Unfortunately, as the elimination proceeds, coefficients that were zero in the original system of equations become nonzero (or fill-in), increasing the work and storage required. The purpose of this paper is to introduce a new graph-theoretic model of such fill-in; to establish lower bounds for the work and storage associated with Gaussian elimination; and to present a minimal storage sparse elimination algorithm that significantly reduces the storage required.

In section 2, we review the graph-theoretic elimination model of Parter and Rose and introduce a new element model of the elimination process. In section 3, we use this model to give simple proofs of inherent lower bounds for the work and storage associated with Gaussian elimination, generalizing similar results of George and Hoffman, Martin, and Rose. Last, in section 4, we show how the element model, combined with the rather unusual idea of recomputing rather than saving the factorization, leads to a minimal storage sparse elimination algorithm that requires significantly less storage than regular sparse elimination (e.g. $O(n^2)$ vs. $O(n^2 \log n)$ for the five- or nine-point operator on an n×n mesh).

2. *An Element Model for Gaussian Elimination*

In this section, we shall review the graph-theoretic elimination model of Gaussian elimination suggested by Parter [6] and extensively developed by Rose [7] and introduce a new element model; cf. Eisenstat [1], Sherman [9].

Given an irreducible N×N symmetric positive definite matrix $A = (a_{ij})$, we can represent the zero-nonzero structure of A by a graph $G(A) = (V,E)$ as follows: The vertex v_i of G(A) corresponds to the *ith* row/column of A; the edge (v_i,v_j) is an edge of G(A) if and only if $a_{ij} \neq 0$. We shall model Gaussian elimination on the matrix A as a sequence $G^{(k)}$ of such graphs.

Let $G^{(1)} = G(A)$. Then the graph $G^{(k+1)}$ is derived from $G^{(k)}$ as follows: Corresponding to using the k*th* equation to eliminate the k*th* variable from the remaining N-k equations, we add any edges necessary to make all vertices adjacent to the k*th* vertex v_k pairwise adjacent and then delete v_k and all edges incident to it. Thus the graph $G^{(k)}$ represents the nonzero structure of the lower N-k+1 × N-k+1 submatrix of A *just before the* k*th* variable is eliminated. From this observation, the following operation and storage counts are immediate:

Theorem (Rose [7]): Let d_k denote the degree of vertex v_k in the elimination graph $G^{(k)}$, i.e. at the time it was eliminated. Then the number of multiplies required to form the U^TDU decomposition of A is given by

$$W = \sum_{k=1}^{N} \frac{1}{2} d_k(d_k+3)$$

and the number of off-diagonal nonzero entries in U is given by

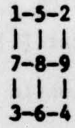
$$S = \sum_{k=1}^{N} d_k.$$

The element model emulates Gaussian elimination as a sequence of transformations on the graph G(A) and the collection E of maximal cliques of vertices in G(A) which we

shall refer to as *elements*. Initially, $\tilde{G}^{(1)} = G(A)$, $E^{(1)} = E$, and all vertices are marked uneliminated. Then $\tilde{G}^{(k+1)}$ and $E^{(k+1)}$ are derived from $\tilde{G}^{(k)}$ and $E^{(k)}$ as follows: Corresponding to using the k*th* equation to eliminate the k*th* variable from the remaining N-k equations, we mark th k*th* vertex v_k as eliminated and add any edges necessary to make all vertices adjacent to v_k pairwise adjacent (no vertices or edges are deleted); all the elements of $E^{(k)}$ containing v_k are merged into a new element and then deleted from $E^{(k)}$. Note that the subgraph of $\tilde{G}^{(k)}$ induced by the uneliminated vertices is just $G^{(k)}$ so that the number of uneliminated vertices adjacent to v_k at the time it is eliminated is d_k as above. Also, elements in $E^{(k)}$ are cliques in $\tilde{G}^{(k)}$ (though not necessarily maximal cliques). A vertex will be said to be an *exterior vertex* in $\tilde{G}^{(k)}$ if it belongs to more than one element; otherwise it belongs to exactly one element and is said to be an *interior vertex*.

As an example, consider the elimination process for the nine-point operator on a 3×3 grid:

Figure 2.1



Initially, the elements correspond to the individual mesh squares or elements, since these form the only maximal cliques in G(A). Hence the name "element model" (cf. George [4]). When we eliminate vertex v_1, there is no change in the element graph (except to mark v_1 eliminated) since it is an interior vertex. The same is true when we eliminate v_2, v_3, and v_4. When we eliminate v_5, however, the two elements containing v_5 are merged, as also happens when we eliminate v_6. Now eliminating v_7 causes the two remaining elements to be merged into the final element which consists of all the vertices. Eliminating vertices v_8 and v_9 has no further effect since they are interior vertices.

We note the following properties of the element model for future use. Throughout the elimination process, the number of elements that contain a given vertex never increases. Thus an eliminated vertex is necessarily an interior vertex, and, correspondingly, an

exterior vertex is uneliminated. (However, an interior vertex need not have been eliminated.) Initially, the largest element contains at most d_{max} vertices, where d_{max} is the maximum degree of any vertex in G(A) and no vertex belongs to more than $2^{d_{max}}$ elements. (See Eisenstat [1] for an example where a vertex actually belongs to $2^{\frac{1}{2}(d_{max}-1)}$ elements.) At the end of the elimination, the only element consists of all the vertices.

3. Lower Bounds for Gaussian Elimination

In recent years, there has been a great deal of interest in computational complexity, particularly in analyzing the number of fundamental operations inherent in a computaion. Along these lines, George [4] and Hoffman, Martin, and Rose [5] have used the elimination graph model to prove lower bounds for the work and storage associated with Gaussian elimination on symmetric positive definite matrices whose graphs are certain regular planar grids. In this section, we shall describe how the element model can be used to give simpler proofs of these results that extend easily to some irregular grids in two and three dimensions. For further details, see Eisenstat [1].

First we shall define the model of computation in which these lower bounds are valid. Given an irreducible N×N symmetric positive definite matrix A, we seek to solve the system of linear equations (1.1) using Gaussian elimination. The work associated with the elimination process will be taken as the number of multiplies to factor A; the storage as the number of nonzero entries in the matrix U.

The work and storage required for Gaussian elimination is directly related to the order in which the variables are eliminated. Thus, instead of solving the original system (1.1), we might prefer to solve the permuted system.

$$P A P^T y = P b, \quad P^T y = x \tag{3.1}$$

for some given permutation matrix P so as to reduce the associated work and storage. Indeed, a great deal of research has been done toward discovering good ordering strategies (e.g. minimum degree [7] and nested dissection [4]). Note that the permutation P does not

change the structure of the graph G(A), other than relabelling the vertices and thus changing the order of elimination. Since the lower bounds we shall derive will be independent of such considerations, they will be valid for all possible orderings.

As in section 2, let $G(A) = (V,E)$ denote the graph associated with A, and let d_{max} denote the maximum degree of any vertex in G(A). We shall now make the following additional assumption about the matrix A or, equivalently, the graph G(A):

"Isoperimetric" Inequality: There exists constants $K > 0$ and $0 < \alpha, \beta \leq 1$ such that, given any subset S of V with $|S| \leq \beta N$, we have

$$|\partial S| \geq K |S|^{\alpha}$$

where

$$\partial S = \{v \in S : \exists w \in V-S \text{ such that } v \text{ and } w \text{ are adjacent in } G(A)\}$$

is the *boundary* of S.

This assumption is closely related to the classical isoperimetric inequality, which relates the area A and perimeter P of a plane figure

$$P^2 \geq \frac{\pi A}{4},$$

and the isovolumetric inequality, which relates the volume V and surface area S of a three-dimensional region

$$S^3 \geq 36\pi V^2.$$

Indeed, proofs of this property for particular grids follow the classical proofs. As an example, for the five- or nine-point operator on an n×n grid, we have

$$|\partial S| \geq |S|^{1/2} \quad \text{if } |S| \leq \frac{n^2}{2},$$

while for the seven- or twenty-seven-point operator on an n×n×n grid we have

$$|\partial S| \geq |S|^{2/3} \quad \text{if } |S| \leq \frac{n^3}{6},$$

cf. Eisenstat [1].

Lemma (Eisenstat [1]); Assume that the graph G(A) satisfies an isoperimetric inequality. Then there exists a $K(\beta N/2^{d_{max}})^{\alpha}$ clique in some elimination graph $G^{(k)}$.

Proof: At the start of the elimination process, the maximum size of any element is d_{max}, the maximimum degree of any vertex in G(A). At the end of the elimination process, there is exactly one element of size N, namely the set of all vertices. Therefore, at some point during the elimination, the first element of size > βN was created. This element was created by merging all the elements containing some vertex v_k. As we saw in section 2, v_k could belong to at most $2^{d_{max}}$ elements at this point and thus at least one such, say e*, must contain more than $\beta N/2^{d_{max}}$ vertices. Consider the boundary vertices of e*. There are at least $K(\beta N/2^{d_{max}})^{\alpha}$ such vertices by the isoperimetric inequality, all are unelimi-nated vertices at this stage, and they are all pairwise adjacent. Thus they form a clique in $G^{(k)}$. QED

Corollary: The bandwidth of the matrix A is at least

$$K(\beta/2^{d_{max}})^{\alpha} N^{\alpha}$$

Corollary: The total work required to factor the matrix A using band elimination is at least

$$\frac{1}{2} K^2(\beta/2^{d_{max}})^{2\alpha} N^{(2\alpha+1)};$$

the number of nonzeroes in the band of A is at least

$$K(\beta/2^{d_{max}})^{\alpha} N^{(\alpha+1)}.$$

The proof follows directly from the standard operation and storage counts for band elimination in terms of the number of equations and the bandwidth.

Corollary: The total work required to factor the matrix A is at least

$$\frac{1}{6} K^3(\beta/2^{d_{max}})^{3\alpha} N^{3\alpha};$$

the number of nonzeroes in U is at least

$$\frac{1}{2} K^2 (\beta/2^{d_{max}})^{2\alpha} N^{2\alpha}.$$

The proof follows from the Lemma as in Hoffman, Martin, and Rose [5].

We shall now examine the consequences of these simple results. For planar grids with $\sim n^2$ vertices and bounded degree, such as the five- or nine-point operator on an n×n grid:

(1) the bandwidth is at least $0(n)$

(2) the work and storage are at least $0(n^4)$ and $0(n^3)$ respectively for band elimination

(3) the work and storage are at least $0(n^3)$ and $0(n^2)$ respectively for sparse elimination (a more careful analysis gives $0(n^2 \log n)$ for the storage; see Eisenstat [1]).

For three dimensional grids with $\sim n^3$ vertices and bounded degree, such as the seven- or twenty-seven-point operator on an n×n×n grid:

(4) the bandwidth is at least $0(n^2)$

(5) the work and storage are at least $0(n^7)$ and $0(n^6)$ respectively for band elimination

(6) the work and storage are at least $0(n^6)$ and $0(n^4)$ respectively for sparse elimination.

4. *Minimal Storage Sparse Elimination*

One of the major disadvantages of Gaussian elimination for solving sparse systems of linear equations is the amount of storage required. For example, to solve the nine-point finite difference operator on an n×n grid requires at least $0(n^2 \log n)$ storage whereas an iterative method would require only $0(n^2)$ storage. In this section, we present a variation of sparse Gaussian elimination that trades a significant reduction in storage for a modest increase in work. For ease of exposition, we shall restrict attention to the nine-point operator on an n×n grid with $n = 2^t-1$, but the results are valid for more general finite

element grids, cf. Eisenstat, Schultz, and Sherman [3], Sherman [9]. For similar results on band elimination, see Eisenstat, Schultz, and Sherman [2], Sherman [9].

Two basic concepts are used in achieving this reduction: First, rather than save the entire factorization, we shall throw most of the nonzero entries of U away and recompute them as needed during the back-solution; second, we use an element merge tree to specify a divide-and-conquer elimination ordering and to keep track of those entries of U that are being saved during each step of the calculation.

Suppose we were to perform Gaussian elimination in such a way that the last 2n+1 variables to be eliminated correspond to the vertices on a dividing cross as shown in Figure 4.1. At the end of the elimination process, we would have generated all the coefficients in the upper triangular system of equations that remains, and we have merely to solve this system for the vector of unknowns x. Yet suppose we had saved only those coefficients in the last 2n+1 rows. Then we could only solve for the last 2n+1 variables, i.e. the values of the unknowns on the dividing cross. This is enough, however, to split our original n×n problem into four smaller $\sim n/2 \times \sim n/2$ problems of the same form, which we can now solve in the same fashion.

Figure 4.1.

Of course, we will have to do more work than we would have had we saved the entire factorization. But how much more? The nested dissection ordering of George [4] orders the variables on the dividing cross last and requires approximately Cn^3 multiplies for some fixed constant C. Thus the cost of the whole procedure is just Cn^3 plus the cost of solving four $\sim n/2 \times \sim n/2$ problems. Letting $\theta(n)$ denote this cost, we have that

$$\theta(n) = Cn^3 + 4\theta(n/2)$$

and that $\theta(n) = Cn^3$ for n sufficiently small. (After all, when the amount of storage for

the factorization gets small enough, we may as well save all of it.) The solution to this recurrence relation is

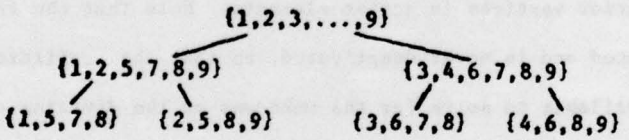
$$\theta(n) = 2Cn^3.$$

Thus we are doing twice as much work but, as we shall see, the savings in storage will be much more significant.

The *element merge tree* is based on the element model introduced in section 2. Recall that initially there were a number of elements, or maximal cliques of vertices, and that as the elimination progressed these elements were merged into larger and larger elements until only a single element containing the entire set of vertices remained. Given an elimination ordering, we construct the corresponding element merge tree as follows: The nodes in the tree represent elements that were created during the elimination process; the root of the tree is the final element consisting of the entire set of vertices; the node (i.e. element) e_1 is a son of another node e_2 if and only if e_1 was merged into e_2 when some vertex v_k was eliminated. The merge tree for the 3×3 nine-point operator of Figure 4.2a is given in Figure 4.2b.

Figure 4.2a.

Figure 4.2b.



We will now specify the ordering of vertices and the corresponding element tree which we will use to make the storage required for our procedure $O(n^2)$ as opposed to the $O(n^2 \log n)$ for the factorization. Beginning with the entire grid, we break it into four equal-size elements using a dividing cross as in Figure 4.1. (Note that the vertices on the dividing cross belong to more than one element since they are exterior vertices.) The last vertices to be eliminated will be the center vertex followed by the other vertices on

the dividing cross. We then order the interior vertices in element I, followed by those of element II, element III, and element IV. The vertices within each of these are eliminated in an analogous fashion: The last vertices to be eliminated are the center vertex followed by the other vertices on a dividing cross; we then order the vertices in each of the four subelements, and so on.

The ordering that results can be shown to be equivalent to the nested dissection ordering in terms of the work and storage required (cf. Eisenstat, Schultz, and Sherman [3], Rose and Whitten [8], Sherman [9]), and the element merge trees are identical, although the actual orderings are completely different. Thus the divide-and conquer order produces an $O(n^3)$ elimination scheme. We now show that the elimination can be carried out in such a way that the coefficients in the last 2n+1 rows can be computed using only $O(n^2)$ storage.

Consider again the element merge tree. The nodes at the bottom or zeroth level correspond to elements in the original graph G(A), all of which contain exactly $4 = (2^0+1)^2$ vertices. At the first level, the elements were formed by merging four bottom-level elements and thus each contains exactly $9 = (2^1+1)^2$ vertices. In general, at the k*th* level the elements were formed by merging four elements on the (k-1)*th* level and, by induction, contain precisely $(2^k+1)^2$ vertices.

We shall say that a particular element in the element tree is active at a particular point in the elimination if it has not yet been merged into another element. Then the only entries in the triangular factor U that we shall be saving at that point are those corresponding to pairs of exterior vertices in active elements. Note that the final element is active once it is created and is never deactivated, so that the coefficients in the last 2n+1 rows of U will be available to solve for the unknowns on the dividing cross at the end of the elimination.

Since there are at most four active elements at any level in the element merge tree (other than the zeroth) at any stage of the elimination by virtue of the recursive definition of the ordering, we can easily bound the total storage required: At the k*th* level from the bottom, there are at most $\ell = 4 \cdot 2^k$ exterior vertices per element, which require at most $\ell(\ell+1)/2$ nonzero entries of U to be saved; thus the total storage is at most

$$\sum_{k=0}^{t-1} 4 \cdot \frac{1}{2} (4 \cdot 2^k)\ (4 \cdot 2^k + 1) \approx \frac{32}{3} n^2.$$

A more careful analysis using two-way dissection show that the total storage can be reduced to $9n^2$, cf. Eisenstat, Schultz, and Sherman [3], Sherman [9].

Note that this surprising result does not really violate the results of section 3; the storage for U still is $O(n^2 \log n)$ -- we just aren't saving all the entries. Another point worth noting is that the algorithm can be implemented to run in time $O(n^{\log_2 7})$ if the elimination of variables on dividing crosses is done in blocks using Strassen's algorithm for matrix multiplication and inversion [10]. Again, this does not violate the results of section 3 since we are doing block rather than point elimination.

References

[1] S. C. Eisenstat. Complexity bounds for Gaussian elimination. To appear.

[2] S. C. Eisenstat, M. H. Schultz, and A. H. Sherman. Minimal storage band elimination. To appear

[3] S. C. Eisenstat, M. H. Schultz, and A. H. Sherman. Minimal storage sparse elimination. To appear.

[4] J. A. George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis* 10:345-363, 1973.

[5] A. J. Hoffman, M. S. Martin, and D. J. Rose. Complexity bounds for regular finite difference and finite element grids. *SIAM Journal on Numerical Analysis* 10:364-369. 1973.

[6] S. V. Parter. The use of linear graphs in Gaussian elimination. *SIAM Review* 3:119-130, 1961.

[7] D. J. Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In R. Read, editor, *Graph Theory and Computing*, 183-217. Academic Press, 1972.

[8] D. J. Rose and G. F. Whitten. Automatic nested dissection. *Proceedings of the ACM National Conference*, 82-88, 1974.

[9] A. H. Sherman. On the efficient solution of sparse systems of linear and nonlinear equations. PhD dissertation, Yale University, 1975.

[10] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik* 13:354-356, 1969.