An efficient band-oriented scheme for solving n by n grid problems

by ALAN GEORGE

University of Waterloo
Waterloo, Ontario, Canada

INTRODUCTION

Let $N=n^2$ for some positive integer n, and consider a square n by n grid consisting of $(n-1)^2$ small squares and having a node at each of the n^2 grid points. In this paper we consider the problem of directly solving the class of N by N symmetric positive definite linear systems of equations

$$Ax=b, \tag{1}$$

where each x_i is associated with a grid point and A has the property that $A_{ij}\neq 0$ only if x_i and x_j are associated with nodes belonging to the same small square. We must specify how the unknowns are to be numbered if the above remark is to precisely determine the structure of A.

Our method of solution is to factor A into LL^T, where L is lower triangular, and then to solve $Ly=b$ followed by the solution of $L^Tx=y$.* The algorithm is essentially the well-known Cholesky (or square root) method, which has the agreeable property that it is numerically stable when applied to PAP^T, where P is any N by N permutation matrix.[14] Thus, we are free to permute the rows and columns of A to achieve other objectives, such as reduced computation and/or storage requirements, or convenient storage management. These objectives often compete with each other, and their relative importance depends upon the problem being solved, the characteristics of the computing system available, and programming expertise. The "best" way to number the equations depends upon which of the objectives we consider to be most important.

Certain members of our problem class which arise in connection with difference discretizations of Poisson's equation on a rectangular domain can be solved using special fast direct methods which require only $0(n^2 \log_2 n)$ arithmetic operations and $0(n^2)$ storage locations.[1,3]

For *any* system in our class, it is possible to number the equations (1) so that A can be factored in $0(n^3)$ arithmetic operations, and the number of non-zero components in L is only $0(n^2 \log_2 n)$.[5] Unfortunately, these latter numbering schemes are somewhat complicated, and yield L's having their non-zero components scattered throughout the lower triangle. In order to achieve the above bounds on storage and computation, general sparse matrix techniques must be used. Their programming is relatively complicated, and their performance and efficiency is sensitive to hardware and software characteristics. See Gustavson[6] for a careful discussion of these methods.

On the other hand, if we number the equations in the natural row by row fashion, and employ a standard band linear equation solver,[8] the programming and data management are straightforward and convenient. Unfortunately, the computation and storage requirements are $0(n^4)$ and $0(n^3)$ respectively.

In this paper we describe a computational scheme for solving (1) which requires no sparse matrix techniques; only dense or band linear systems must be solved. Furthermore, we show that the computation and storage requirements for our scheme are respectively $0(n^3\sqrt{n})$ and $0(n^2\sqrt{n})$.

Our method and results apply with little modification to more general situations where there are nodes on the sides of the small squares, and where there is more than one unknown associated with each node.[4,15] Our scheme also applies to matrix problems which arise in connection with the use of spline bases to solve elliptic boundary value problems.[12] In this case unknowns associated with nonadjacent squares may be connected. We discuss these generalizations in our concluding remarks, but since the extensions are straightforward, for clarity

* For sparse matrix calculations, the $\bar{L}D\bar{L}^T$ factorization may be more efficient, where $\bar{L}$ is unit lower triangular and D is a positive diagonal matrix. The scheme we propose works equally well for either factorization.

we present the simplest case. We shall not, therefore, distinguish between "node" and "unknown" in the sequel.

DESCRIPTION OF THE COMPUTATIONAL SCHEME

For some positive integer $\alpha \ll n$, choose $\alpha-1$ horizontal lines of our n by n grid which divide the mesh into α approximately equal parts, each part containing approximately n^2/α nodes. These sets of nodes (unknowns) are independent in the sense that if x_i and x_j lie in different sets, then $A_{ij}=0$.

Consider Figure 1 below, where for definiteness we choose $\alpha=3$.

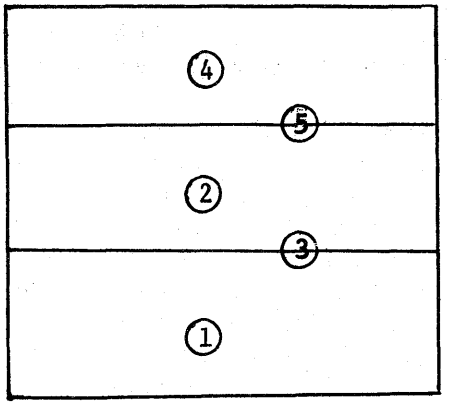

Figure 1—The n by n mesh divided into 3 parts. Circled numbers indicate the order in which node sets are to be numbered.

The lines ③ and ⑤ each consist of n nodes, and following Rose,[10] we refer to them as *separators*. The node sets designated by ①, ② and ④ will be referred to as the *independent blocks*. In general, we have α independent blocks separated by $\alpha-1$ separators.

We number the unknowns in ① followed by those in ② *column by column*, followed by the unknowns in ③ in any order. We then number the unknowns in ④ column by column followed finally by those in ⑤ in any order.

In block form, the coefficient matrix A has the structure below

$$A=\begin{pmatrix} A_1 & & C_1^T & & \\ & A_2 & B_2^T & & C_2^T \\ C_1 & B_2 & A_3 & & \\ & & & A_4 & B_4^T \\ & C_2 & & B_4 & A_5 \end{pmatrix}.$$

Factoring A into LL^T, we obtain

$$L=\begin{pmatrix} L_1 & & & & \\ & L_2 & & & \\ C_1L_1^{-T} & B_2L_2^{-T} & L_3 & & \\ & & & L_4 & \\ & C_2L_2^{-T} & F_3 & B_4L_4^{-T} & L_5 \end{pmatrix}$$

where

$$F_3^T=-L_3^{-1}B_2A_2^{-1}C_2^T,$$

$$A_i=L_iL_i^T,\ i=1,2,4,$$

$$\hat{A}_3=A_3-B_2A_2^{-1}B_2^T-C_1A_1^{-1}C_1^T=L_3L_3^T,$$

and

$$\hat{A}_5=A_5-B_4A_4^{-1}B_4^T-F_3F_3^T-C_2A_2^{-1}C_2^T=L_5L_5^T. \quad (2)$$

Note that for $\alpha>3$, the structure of the last two columns and rows of A and L would simply repeat.

Using the important observation of Rose and Bunch[11] that the matrices $B_iL_i^{-T}$ and $C_iL_i^{-T}$ will be fuller than B_i or C_i, we propose only to store the L_i's and F_i's. Having them available, the solution of our example would proceed as follows, where x and y are partitioned corresponding to A. Parentheses indicate the order in which computations are performed. Note that only triangular systems of equations are solved. For example, in 1(c) below the vector $L_4^{-T}y_4$ is obtained by solving a triangular system.

1. (a) Solve $L_iy_i=b_i$, $i=1, 2, 4$. These can be solved in any order, or simultaneously.
   (b) Compute $b_3'=b_3-B_2(L_2^{-T}y_2)-C_1(L_1^{-T}y_1)$ and then solve $L_3y_3=b_3'$.
   (c) Compute
   $$b_5'=b_5-B_4(L_4^{-T}y_4)-F_3y_3-C_2(L_2^{-T}y_2)$$
   and then solve $L_5y_5=b_5'$.
2. (a) Solve $L_5^Tx_5=y_5$.
   (b) Compute $y_4'=y_4-L_4^{-1}(B_4^Tx_5)$ and solve $L_4^Tx_4=y_4'$.
   (c) Compute $y_3'=y_3-F_3^Tx_5$ and solve $L_3^Tx_3=y_3'$.
   (d) Compute $y_2'=y_2-L_2^{-1}(C_2^Tx_5)-L_2^{-1}(B_2^Tx_3)$ and solve $L_2^Tx_2=y_2'$.
   (e) Compute $y_1'=y_1-L_1^{-1}(C_1^Tx_3)$ and solve $L_1^Tx_1=y_1'$.

The scheme for $\alpha>3$ is obvious.

Before proceeding to the next section, the reader should verify that our storage needs in the example above are only $n^3/3+0(n^2)$, rather than the $n^3+0(n^2)$ required for the usual row by row numbering scheme.

We assume throughout that the so-called diagonal storage scheme[8] is used to store the band matrices.

Storage requirements and operation counts

Following Cuthill and McKee,[2] we define the bandwidth m of a symmetric matrix W by

$$m = \max_{W_{ij}\neq 0} |\, i-j \,|$$

Using the notation $\approx$ to mean "approximately," and assuming $\alpha \ll n$, we observe that the dimension of the A_i's and L_i's corresponding to the independent blocks will be $\approx n^2/\alpha$. The A's corresponding to the separators are of dimension n, and although they are sparse, their corresponding L_i's are in general full lower triangular matrices. The F_i's are in general full matrices. Recalling that there are α independent blocks, $\alpha-1$ separators, and $\alpha-2$ F_i's, and adding several more n^2 words of storage for x, b and temporary space, we obtain the following estimate $S(\alpha)$ of our storage requirements:

$$S(\alpha) = n^3/\alpha + 3\alpha n^2/2.$$

We ignore the storage required for the B_i and C_i, since their requirements are only $0(\alpha n)$.

The above is minimized when $\alpha = \hat{\alpha} = \sqrt{2n/3}$, and

$$S(\hat{\alpha}) = \sqrt{6} n^{5/2}.$$

Table 1 demonstrates the rather significant reduction in storage requirements over the usual row by row ordering scheme.

TABLE 1—Storage Requirements for our Block Scheme Compared to the Row by Row Ordering

n	N	$\sqrt{6}\, n^{5/2}$	n^3
10	100	775	1,000
20	400	4,380	8,000
30	900	12,075	27,000
40	1,600	24,800	64,000
50	2,500	43,250	125,000
100	10,000	244,950	1,000,000

We now obtain a crude estimate for the number of multiplicative operations required for our scheme. We use $\frac{1}{2}m^2N$ as an estimate for the cost of factoring an N by N symmetric positive definite matrix having bandwidth m.

First observe that the calculation of the L_i's corresponding to the independent blocks requires about $\alpha(\frac{1}{2})(n/\alpha)^2(n^2/\alpha) = (n^4/\alpha^2)/2$ multiplicative operations. The calculation of the L_i's corresponding to the separators (once we have computed the $\hat{A}_i$'s) requires approximately $(\alpha-2)n^3/6 \approx \alpha n^3/6$ operations.

Now consider the calculation of the $\hat{A}_i$'s from formulas of the type (2). An example is

$$\hat{A}_i = A_i - B_{i-1}A_{i-1}^{-1}B_{i-1}^T - F_{i-2}F_{i-2}^T - C_{i-3}A_{i-3}^{-1}C_{i-3}^T.$$

Normally, we would compute $B_{i-1}A_{i-1}^{-1}B_{i-1}^T$ by first calculating $W = L_{i-1}^{-1}B_{i-1}^T$ and then computing WW^T. However, using the structure of L_{i-1} and assuming exact numerical cancellation does not occur,** it is easy to show that W is about half full. We require at least $n^4/(3\alpha)$ multiplications to compute WW^T, and about $n^3/(2\alpha)$ auxiliary storage locations are required for W.

However, if we instead compute $\hat{W} = A_{i-1}^{-1}B_{i-1}^T$ and then utilize the fact that B has fewer than $3n$ nonzero components when computing $B_{i-1}\hat{W}$, we perform only about $2n^4/\alpha^2$ operations. Furthermore, we can compute $B_{i-1}A_{i-1}^{-1}B_{i-1}^T$ column by column, and only n^2/α temporary storage locations are required.

Using this crucial observation, the calculation of the $\hat{A}_i$'s requires about $(\alpha-1)(4n^4/\alpha^2+n^3) \approx 4n^4/\alpha+\alpha n^3$ multiplications. Utilizing the sparsity of the B_i's and C_i's in the same way, the calculation of the F_i's requires about $(\alpha-2)(2n^4/\alpha^2+n^3/2) \approx 2n^4/\alpha+\alpha n^3/2$ operations.

Collecting terms, we obtain the estimate $M(\alpha)$ for the number of multiplications required to produce L:

$$M(\alpha) = 6n^4/\alpha + 5\alpha n^3/3 + n^4/(2\alpha^2).$$

Assuming $\alpha \gg 1$, $M(\alpha)$ is approximately minimized for $\alpha = \tilde{\alpha} = \sqrt{18n/5}$, yielding

$$M(\tilde{\alpha}) = 2\sqrt{10}\, n^3\sqrt{n} + 5n^3/36.$$

Thus, $M(\tilde{\alpha}) > n^4/2$ unless N is very large indeed ($\approx 25{,}000$), although $M(\tilde{\alpha}) < n^4$ if N is larger than about 1,600. Thus, unless N is very large we will pay a modest premium in arithmetic operations if we use our block scheme. In exchange we obtain a substantial decrease in storage requirements. It is interesting to note that if we do not make use of our observation in computing $\hat{A}_i$'s, the operation count is $0(n^4)$.

CONCLUDING REMARKS

1. The procedure represents a considerable improvement over the standard row by row scheme; whether its comparative simplicity renders it competitive or superior to the use of more sophisticated orderings depend upon our particular computing environment and programming expertise. An attempt to at least partially answer this question is a topic of further research.

** This is a reasonable assumption in the presence of rounding error.

2. As we stated in the introduction, similar results hold for more general grid problems. When edge or interior nodes occur, one should use "profile" or "envelope" methods[4,7,9] rather than band schemes for best results. Problems arising through the use of splines have the property that unknowns associated with grid points p and q are connected provided the maximum difference in their x or y grid coordinates is bounded by some number d, which depends upon the degree of the spline. To apply our scheme we simply choose sets of d adjacent parallel grid lines as separators and proceed as before.
3. For matrix problems arising from the 3-dimensional unit cube grid having n^3 nodes, we can choose separators consisting of planes of grid points and again apply the same techniques. The computation and storage estimates achieved are respectively $0(n^6\sqrt{n})$ and $0(n^4\sqrt{n})$, compared to $0(n^7)$ and $0(n^5)$ for the standard plane-by-plane numbering scheme.
4. It seems fairly obvious that similar ideas can be applied to less regular problems, but it is difficult to obtain quantitative estimates of how much might be gained. Intuitively, we want to choose small separators, yielding independent blocks of nodes which can be numbered so as to have a small band or profile. The study of automatic schemes for doing this is another topic for future research.

ACKNOWLEDGMENT

Part of this research was performed while the author was visiting the IBM Watson Research Center, Yorktown Heights, New York. Several helpful conversations with Dr. Fred Gustavson of the research center are gratefully acknowledged. This work was supported in part by Canadian National Research Grant A8111.

REFERENCES

1 B L BUZBEE G H GOLUB C W NIELSON
On direct methods for solving Poisson's equations
SIAM Journal for Numerical Analysis 1 1970 pp 627-656

2 E CUTHILL J McKEE
Reducing the bandwidth of sparse symmetric matrices
Proc 24th National Conference Association Computer Machinery ACM Publication P 69 1122 Avenue of the Americas New York New York 1969

3 F W DORR
The direct solution of the discrete Poisson equation on a rectangle
SIAM Review 12 1970 pp 248-263

4 J A GEORGE
Computer implementation of the finite element method
Stanford Computer Science Department Technical Report STAN-CS-71-208 Stanford California 1971

5 J A GEORGE
Nested dissection of a regular finite element mesh
To appear in SIAM Journal for Numerical Analysis

6 F G GUSTAVSON
Some basic techniques for solving sparse systems of linear equations in *Sparse Matrices and Their Applications*
(D J Rose and R A Willoughby Editors) Plenum Press 1972

7 A JENNINGS
A compact storage scheme for the solution of symmetric linear simultaneous equations
Computer Journal 9 (1966) pp 281-285

8 R S MARTIN J H WILKINSON
Symmetric decomposition of positive definite band matrices
Handbook series Linear Algebra Numerische Mathematik 7 (1965) pp 355-361

9 R J MELOSH R M BAMFORD
Efficient solution of load-deflection equations
Journal of the American Society of Civil Engineers Structural Division 95 (proc paper #6510) pp 661-676

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 (R C Read Editor) Academic Press New York 1972

11 D J ROSE J R BUNCH
The role of partitioning in the numerical solution of sparse systems
In *Sparse Matrices and Their Applications* (D J Rose and R A Willoughby Editors) Plenum Press New York 1972

12 M H SCHULTZ
Elliptic spline functions and the Rayleigh-Ritz-Galerkin
Mathematics of Computation 24 1970 pp 65-80

13 K L STEWART J BATY
Dissection of structures
Journal of the American Society of Civil Engineers Structural Division (Proc Paper #6502) 93 pp 217-232

14 J H WILKINSON
The algebraic eigenvalue problem
Clarendon Press Oxford England 1965

15 O C ZIENKIEWICZ
The finite element method in engineering science
McGraw-Hill London 1971