A compact storage scheme for the solution of symmetric linear simultaneous equations

By Alan Jennings*

A method is presented for the computer solution of symmetric linear simultaneous equations which takes advantage of the presence of zero elements away from the leading diagonal but which is more flexible than diagonal band storage. The equations are solved by a form of compact elimination.

A routine based on the method has been programmed in Atlas Autocode and used in some analyses of building frameworks. It is found to be economical both in use of store and computing time.

In the solution of many types of problem, linear simultaneous equations arise which are both symmetric and sparce. In particular, network and field problems can give rise to equations of this sort.

Several alternative methods of storage which make use of low densities of non-zero left-hand side coefficients have been used; they can be broadly classified into diagonal band, submatrix and sparse matrix storage. The most commonly used methods for the analysis of large structural systems are submatrix storage schemes, which can either involve large submatrices in a predetermined pattern, for instance tridiagonal (Wilson, 1959), or have smaller submatrices in a pattern controlled dynamically by means of layout parameters (Livesley, 1960). The use of large submatrices may result in a large number of zero elements being stored within the submatrices and a restriction on the equations to the predetermined layout, whereas the more general approach with smaller submatrices requires complex programming.

Submatrix methods are useful where magnetic tapes are needed as a backing store; however, with the advent of computers with large fast-access stores, the need to subdivide matrices for the purpose of storage transfer is not significant for a large range of problems. If submatrix methods are not used, then it is noted that diagonal band storage is very simple to program for direct solution, but the restriction imposed by having a constant bandwidth could in some cases be very inconvenient.

If the equations are solved by iterative methods then the basic operation is one of multiplication. It is fairly easy to adapt this to any method of storage including forms of sparse matrix storage (Khabaza, 1963). Where comparisons have been made between direct and iterative methods for banded matrices (Wilson, 1959; Clough, Wilson and King, 1963) it appears that iterative methods will only tend to be more efficient when the bandwidth is large and the number of right-hand sides small. However, the number of iterations required for reasonable accuracy is dependent on the method of iteration chosen, the order of variables, the nature of the initial trial vector and the convergence characteristics of the problem, so that any comparison between direct and iterative methods is always open to revision under different circumstances. An increase in the efficiency of the programming technique used for either method is also a factor which would affect the comparison.

The method adopted in this paper is a direct solution method using a non-standard form of storage for the matrix of left-hand side coefficients. The method is almost as versatile as a sparce matrix storage scheme in dealing with arbitrary patterns of non-zero elements, while at the same time retaining most of the simplicity inherent in the method of direct solution when using a diagonal band storage scheme.

Matrix formulation of equations

A set of symmetric linear simultaneous equations may be represented in matrix notation by

$$\mathbf{AX} = \mathbf{B} \qquad (1)$$

where **A** is an $n \times n$ symmetric array of L.H.S. coefficients and each column of the $n \times m$ matrix **B** represents a set of R.H.S. coefficients. The variables which relate to each set of R.H.S.'s are represented by columns in the $n \times m$ matrix **X**. When considered in a matrix context the L.H.S. and R.H.S. coefficients of the equations become elements of the matrices **A** and **B** respectively.

Methods of direct solution perform linear operations on the rows of the matrices **A** and **B** so that **B** is transformed into **X** (which normally has no zero elements). In the method to be presented the storage for matrix **B** is a conventional two-dimensional array, but the storage for **A** is of non-standard type.

The method has been applied to the solution of positive definite systems of equations in which row and column interchange facilities are not required for direct solution.

Method of storage

The method adopted for matrix **A** is to store all the elements below the leading diagonal in sequence by rows, but with all elements preceding the first non-zero element in each row left out.

* *Civil Engineering Dept., Queen's University, Belfast, 9.*

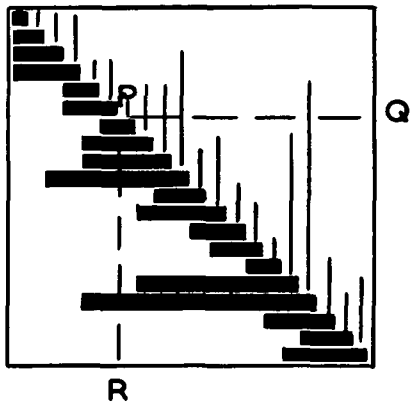

Fig. 1

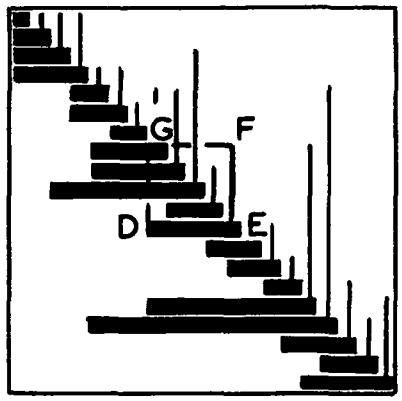

Fig. 2

The following set of L.H.S. coefficients

$$\begin{bmatrix} 1{\cdot}5 & 0{\cdot}2 & -1{\cdot}1 & 0 & 0 & 0 \\ 0{\cdot}2 & 1{\cdot}2 & 0 & 0 & 0 & 0 \\ -1{\cdot}1 & 0 & 2{\cdot}2 & 5{\cdot}1 & 0 & -1{\cdot}2 \\ 0 & 0 & 5{\cdot}1 & 10{\cdot}6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2{\cdot}6 & 0 \\ 0 & 0 & -1{\cdot}2 & 0 & 0 & 6{\cdot}1 \end{bmatrix}$$

would be stored in the computer in what will be called the *main sequence* as follows:

1·5 0·2 1·2 −1·1 0 2·2 5·1 10·6 2·6 −1·2 0 0 6·1.

In addition an *address sequence* is used to locate the positions of the leading diagonal elements within the main sequence. For the coefficients shown the address sequence would be

1 3 6 8 9 13.

It will be noted that, when solved by elimination and backsubstitution without row or column interchange, all the build up of non-zero elements below the leading diagonal will occur within the elements stored in the main sequence.

The compact elimination method of solution

Fig. 1 shows diagrammatically the elements stored for a particular elimination. Consider first the standard Gaussian elimination without row or column interchange. At some stage the coefficients of column PR will be eliminated by using equation PQ. In order to discover which rows are affected the address sequence for the whole of the latter part of the matrix will have to be inspected and the appropriate rows operated on. However in the compact elimination method the elimination of one row (say DE of **Fig. 2**) is performed by referring only to the elements within the square DEFG. Thus the compact elimination method is not only more straightforward to program but also involves less jumping about within the main sequence store.

The reduction process

Consider the reduction of a set of four simultaneous equations with two right-hand sides. Denote L.H.S. coeffs by a and R.H.S. coeffs by b. After the first two equations have been reduced so that their leading diagonal coefficients are unity the equations appear as follows:

$$\begin{array}{cccccc} 1 & a_{12}^{**} & a_{13}^{**} & a_{14}^{**} & b_{11}^{**} & b_{12}^{**} \\ & 1 & a_{23}^{**} & a_{24}^{**} & b_{21}^{**} & b_{22}^{**} \\ a_{31} & a_{32} & a_{33} & a_{34} & b_{31} & b_{32} \\ a_{41} & a_{42} & a_{43} & a_{44} & b_{41} & b_{42} \end{array}$$

The third equation is then reduced by making

$$\left.\begin{aligned} a_{31}^{*} &= a_{31} \\ a_{32}^{*} &= a_{32} - a_{31}^{*}a_{12}^{**} \\ a_{33}^{*} &= a_{33} - a_{31}^{*}a_{13}^{**} - a_{32}^{*}a_{23}^{**} \\ a_{34}^{*} &= a_{34} - a_{31}^{*}a_{14}^{**} - a_{32}^{*}a_{24}^{**} \\ b_{31}^{*} &= b_{31} - a_{31}^{*}b_{11}^{**} - a_{32}^{*}b_{21}^{**} \\ b_{32}^{*} &= b_{32} - a_{31}^{*}b_{12}^{**} - a_{32}^{*}b_{22}^{**} \end{aligned}\right\} \quad (2)$$

and then putting

$$\left.\begin{aligned} a_{34}^{**} &= a_{34}^{*}/a_{33}^{*} \\ b_{31}^{**} &= b_{31}^{*}/a_{33}^{*} \\ \text{and} \quad b_{32}^{**} &= b_{32}^{*}/a_{33}^{*} \end{aligned}\right\} \quad (3)$$

The modified coefficients are stored in a transposed form in the storage locations vacated by the eliminated coefficients, which means that a_{34}^{**} cannot be stored until the reduction of the fourth equation is partly completed. This difficulty is overcome by making use of symmetry so that

$$a_{34}^{**} = a_{43}^{*}/a_{33}^{*} \quad (4)$$

this element being formed on the reduction of the fourth equation. On the reduction of the third equation the modified L.H.S. coefficients evaluated are those of the third column, i.e.

$$\left.\begin{aligned} a_{13}^{**} &= a_{31}^{*}/a_{11}^{*} \\ \text{and} \quad a_{23}^{**} &= a_{32}^{*}/a_{22}^{*} \end{aligned}\right\} \quad (5)$$

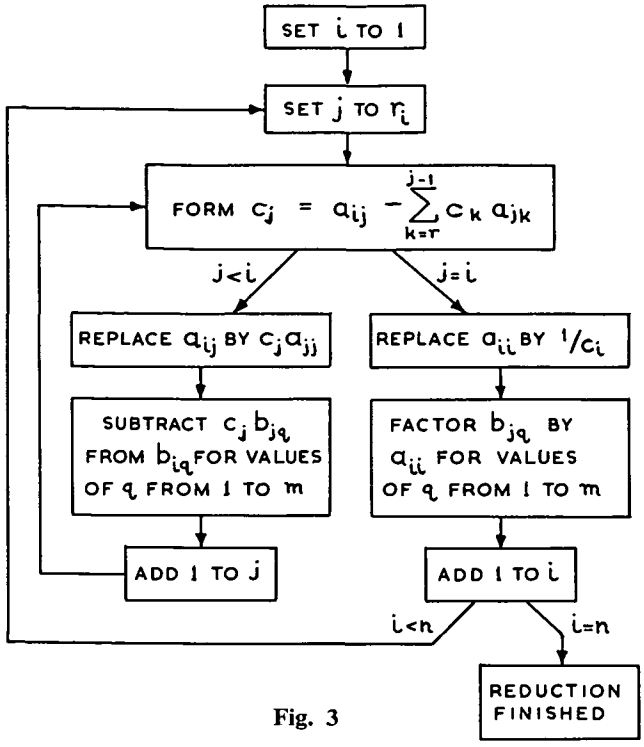

Fig. 3

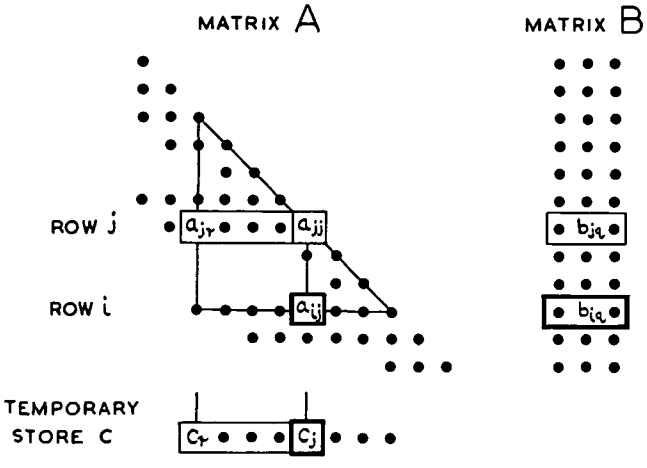

Fig. 4

As the factors $1/a_{11}^{*}$, $1/a_{22}^{*}$, etc., will be needed after the reduction has taken place on their respective rows then they are entered into the storage locations vacated by a_{11}, a_{22}, etc.

This process can be generalized and applied to the case where the storage is of the compact form already described. The sequence of arithmetical operations can be summarized by the flow diagram shown in **Fig. 3**. The column number of the first element in row i has been designated r_i, and r is the greatest of r_i and r_j where r_j is the column number of the first element in row j.

A set of temporary stores are used which are designated c. The maximum number of these stores required equals the largest number of elements stored for any row of matrix **A**. **Fig. 4** illustrates the sequence of events

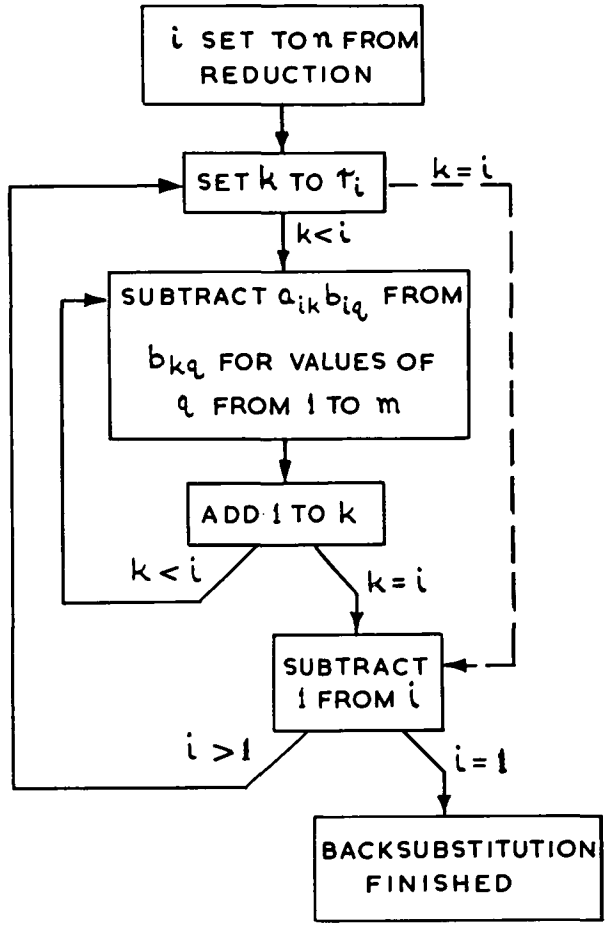

Fig. 5

contained within one cycle of the inner loop of the flow diagram, which has the effect of modifying one element of the **A** matrix and performing parallel operations on the **B** matrix. A possible pattern of stored elements is shown by dots, the elements used are enclosed by boxes and those modified are enclosed by heavier boxes.

Backsubstitution

Having reduced the L.H.S. to an upper triangular array of coefficients with unit terms on the leading diagonal, the process of backsubstitution eliminates all the elements above the leading diagonal.

Using the notation of equations 2 and 3 then element a_{ij}^{**} is eliminated by subtracting $a_{ij}^{**} b_{jq}^{**}$ from b_{iq}^{**} for all values of q from l to m.

The elements a_{ij}^{**} are most easily eliminated by columns. Because a_{ij}^{**} is stored in the location vacated by a_{ji} then the elimination will proceed by rows in the main sequence store. The flow diagram associated with backsubstitution is shown in **Fig. 5**.

Comparison with other methods of solution

The method presented enables sparse symmetric linear simultaneous equations to be solved by elimination using less storage space than with diagonal band and sub-

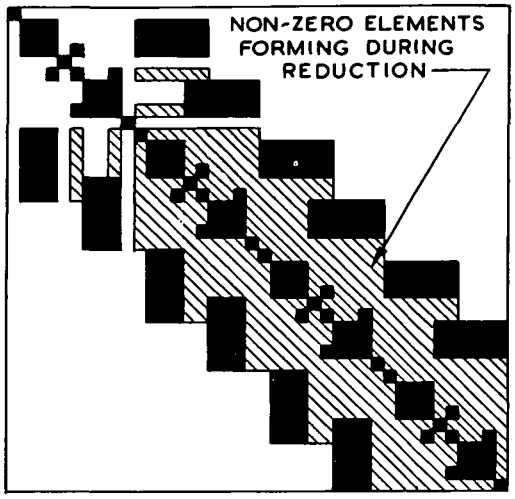

Fig. 6

matrix methods. Equations of the form shown in **Fig. 6** (where $n = 40$) lend themselves easily to diagonal band storage. However, even in this case the compact storage system entails a saving in storage space of over 30% as shown in **Table 1**, and a saving in total number of arithmetical operations of over 30% as shown in **Table 2**. Also shown in Table 2 is a comparison of machine times for various Atlas Autocode programs when used to solve a set of equations whose form is shown in Fig. 6. Only the compact storage method program made use of symmetry, the Gaussian elimination programs were written without row and column interchange and one program had a facility for missing out a row of operations if the element to be eliminated was already zero. The sparse matrix program which uses a direct method of solution, was found to give optimum computer time when the variables were rearranged so that the **A** matrix appeared in the form shown in **Fig. 7**.

Whereas it is advantageous when using the compact storage scheme that the equations be so arranged that the average bandwidth is small, some elements away from the leading diagonal can be accommodated without a large increase in storage requirement or computing time. It is not necessary or beneficial to introduce extra variables in order to draw all the elements inside a fixed bandwidth as advocated for the submatrix method of Livesley (1960). It is possible in certain types of structural problems to obtain a set of simultaneous equations whose non-zero L.H.S. coefficients lie within the shaded area in **Fig. 8**. This arises when a few only of the variables have an influence on the whole of the structure, the rest having only a local influence. The compact storage system is extremely suitable in such cases.

Formation of equations

Because it is not possible to allocate the storage space to the individual elements until the position of the first non-zero elements in each row is known, the construction of the equations could give rise to difficulty under certain circumstances.

Table 1

Comparison of storage requirements for matrix shown in Fig. 6

STORAGE TYPE	REQUIREMENT
Total matrix	1,600
Diagonal band (symmetric) storage	600
Tridiagonal submatrix (symmetric) storage	700
Compact storage (including c store of 15)	409

Table 2

Comparison of arithmetical operations and computer time for the solution of equations of the form shown in Fig. 6

METHOD	NUMBER OF MULTIPLICATIONS AND DIVISIONS	COMPUTER TIME SECS.
Gaussian elimination without bypass	21,300	1·8
Gaussian elimination with zero bypass	—	0·68
Diagonal band (symmetric)	4,630	—
Sparse matrix (regular joint sequence)	4,000 approx.	2·02
Sparse matrix (improved joint sequence)	2,000 approx.	1·04
Compact storage (symmetric)	3,169	0·37

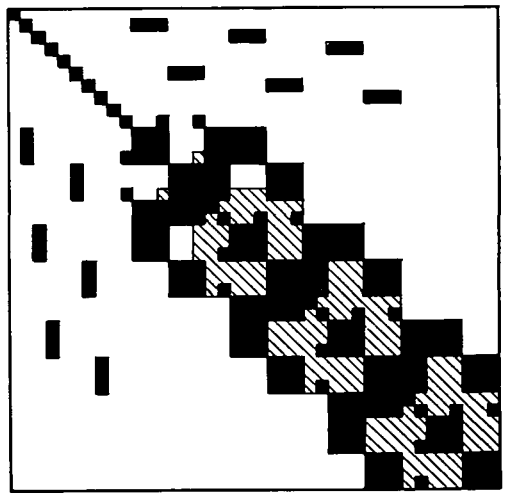

Fig. 7

If the equations are to be constructed by hand and read into the computer as data or if they can easily be produced row at a time then no difficulty arises. How-

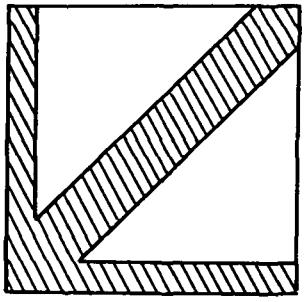

Fig. 8

ever, if the elements are to be entered in a more random fashion then it will be first necessary, by some means, to determine the positions of the first non-zero elements of all the rows before the matrix can be constructed.

Conclusion

A method of computer storage and method of solution has been developed for symmetric linear simultaneous equations which have zero elements away from the leading diagonal in the matrix of L.H.S. coefficients. The method is versatile and economical as regards use of storage space and computer time for solution, while at the same time not being unduly complex.

Acknowledgement

This paper arises from work carried out at the Civil Engineering Department and the Computing Laboratory at Manchester University, and which has been continued in the Department of Civil Engineering, Queen's University of Belfast.

References

CLOUGH, R. W., WILSON, E. L., and KING, I. P. (1963). "Large Capacity Multistory Frame Analysis Programs", *Journal of the Structural Division, A.S.C.E.*, Vol. 89, No. ST4, p. 179.

KHABAZA, I. M. (1963). "An Iterative Least Squares Method Suitable for Solving Large Sparse Matrices", *The Computer Journal*, Vol. 6, p. 202.

LIVESLEY, R. K. (1960). "The Analysis of Large Structural Systems", *The Computer Journal*, Vol. 3, p.34.

WILSON, L. B. (1959). "Solution of Certain Large Sets of Equations on Pegasus Using Matrix Methods", *The Computer Journal* Vol. 2, p. 130.

Book Review

Approximation of Functions, edited by H. Garabedian, 1965; 215 pages. (Barking: *Elsevier Publishing Company Ltd.*, 70s.)

The literature on approximation, a subject which has always been of interest to mathematicians, has of recent years experienced an explosive increase. For example, it is noted by P. J. Davis in this volume, in a brief and very readable article, that he found in just two years of *Mathematical Reviews* over 120 papers and a dozen books on topics of approximation theory that seemed relevant to practical numerical analysis; these are listed in an appendix to his paper. Also in this volume, in an account of recent Russian literature on approximation, G. E. Lorentz lists over 150 publications since 1958, and notes that these are only the more important and original papers, some 45–50% of the total output. In the light of these figures, it is clear that the mathematician interested in approximation is now faced with a formidable task in merely attempting to keep abreast of new developments.

In such a situation, the value of a book of this kind is very great. Here we have thirteen papers which were presented at the *Symposium on the Approximation of Functions* held at the *General Motors Research Laboratories* in 1964, each by an expert of international reputation. The list of contibutors alone will arouse the interest of anyone acquainted with the subject: F. L. Bauer, G. Birkhoff and C. de Boor, R. C. Buck, E. W. Cheney, L. Collatz, P. J. Davis, A. A. Goldstein, M. Golomb, G. G. Lorentz, J. R. Rice, A. Sard, E. L. Stiefel, and J. L. Walsh. Although such a collection of papers cannot possess the coherence of a book by a single author, it can and does present a general, if incomplete, picture of the frontiers of the subject in 1964.

Apart from the surveys by Davis and Lorentz already mentioned, the subject-matter of the papers ranges over those approximation topics which are of greatest interest to numerical analysts; there are papers on approximation by polynomials, by rational functions and by splines, approximation to continuous functions and approximation on finite point sets, as well as papers giving general theoretical background. There is still evidence of a difference of viewpoint between those who are interested in solving practical problems and those who are concerned with proving theorems in functional analysis (or, as Davis puts it, between the "Earth men" and the "Space men") but perhaps the distinction is less sharp than it was some four or five years earlier. Certainly there are theoretical papers here which are concerned with the development of algorithms, and this trend is surely to be welcomed.

It is difficult and perhaps unfair to pick out any one paper for special mention, but in the reviewer's opinion that by Birkhoff and de Boor on "Piecewise polynomial interpolation and approximation" should be particularly noted for its eminently readable presentation of a topic which is of considerable current interest. The interest here is heightened by expressions of the authors' opinions and recommendations; controversial though these may sometimes be, they are always stimulating and relevant.

The general standard of the papers is very high, and the book is well printed. No numerical analyst should be without it.

C. W. CLENSHAW