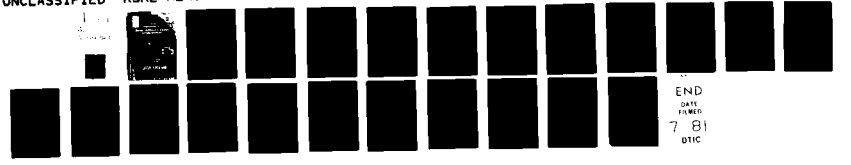
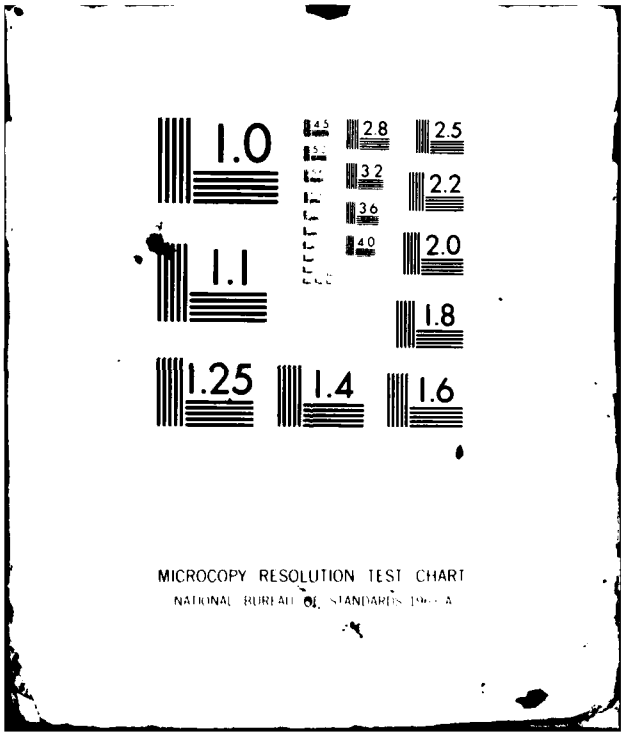
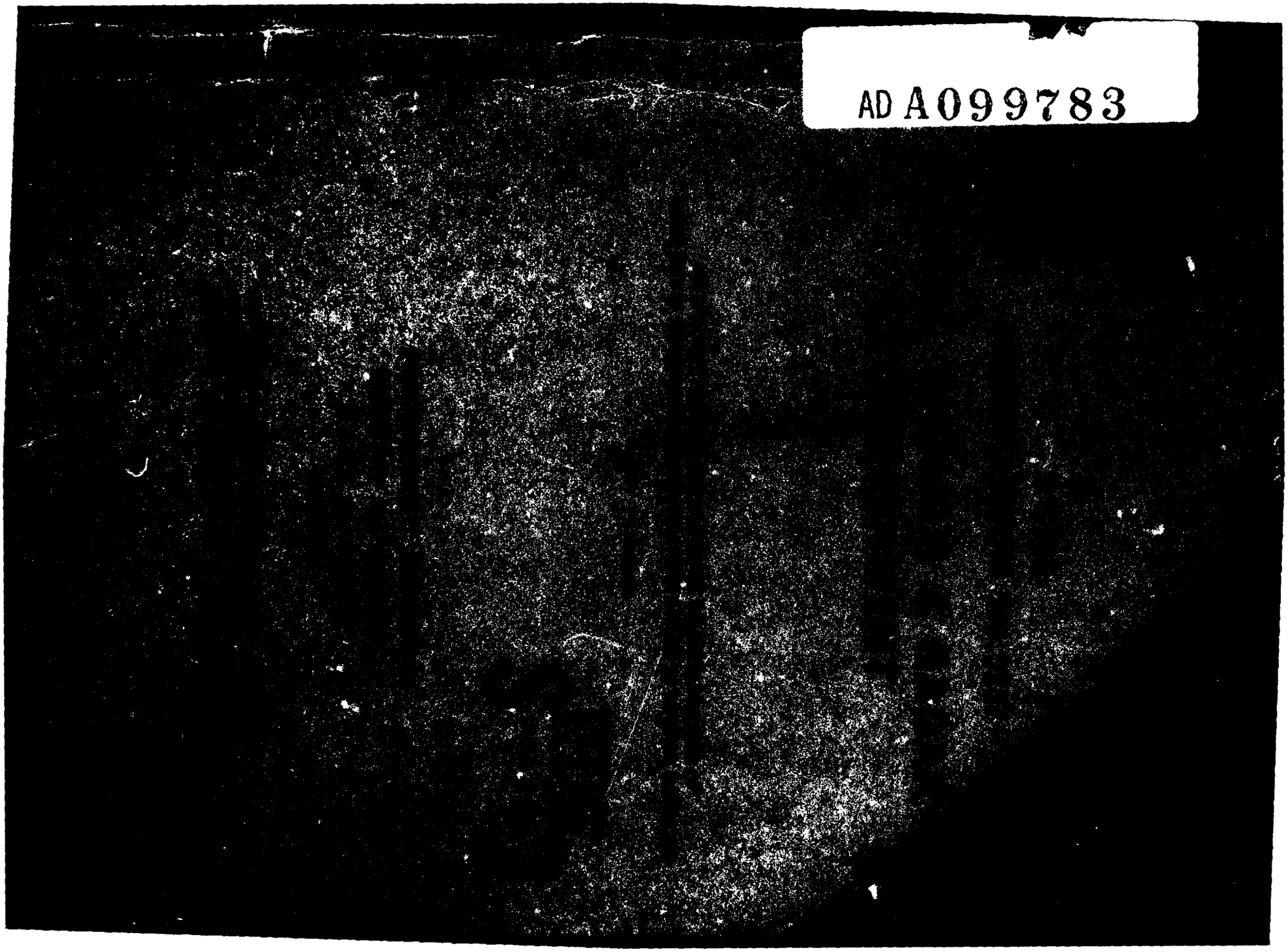
AD-A099 783 ROYAL SIGNALS AND RADAR ESTABLISHMENT MALVERN (ENGLAND) F/G 12/1
AN EFFICIENT ITERATIVE METHOD FOR THE COMPUTER SOLUTION OF LARG--ETC(U)
NOV 79 T E HODGETTS
UNCLASSIFIED RSRE-MEMO-3237 DRIC-BR-75641 NL



END
DATE FILMED
7 81
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS



(12) 23 (18) DRIC (19) BR-75642

ROYAL SIGNALS AND RADAR ESTABLISHMENT

Memorandum No 3237

(6)

Title: AN EFFICIENT ITERATIVE METHOD FOR THE COMPUTER SOLUTION OF LARGE SETS OF SIMULTANEOUS LINEAR EQUATIONS APPLIED TO ANTENNA MODELLING

Author: (10) T. E. Hodgetts

Date: (11) Nov[redacted]979

(14) RSRE-MEMO-3237

SUMMARY

Very large sets of simultaneous linear equations can usually be solved to reasonable accuracy more quickly by an iterative method than by a direct one, but most iterative methods only converge if the elements of the principal diagonal of the coefficient matrix dominate all the other elements. This paper describes an iterative method which depends for its convergence only on dominance by the elements of an arbitrarily wide band of diagonals centred on the principal diagonal. The theory of the method is fully worked out, and a computer program implementing it in Fortran IV is presented.

RSRE MEMORANDUM 3237

AN EFFICIENT ITERATIVE METHOD FOR THE COMPUTER SOLUTION OF LARGE SETS OF SIMULTANEOUS LINEAR EQUATIONS APPLIED TO ANTENNA MODELLING

T E Hodgetts

LIST OF CONTENTS

1 INTRODUCTION

Boundary value problems in field theory, including those of the electromagnetic radiation from antennas, are conveniently expressible in the form of linear integral equations. It has become customary to solve these equations using the so-called "method of moments" (Ref 1). This method consists of expanding the unknown field function as a linear combination of suitably chosen functions and taking the integral average of the resulting form of the integral equation with each member of the set of functions (or sometimes with each member of another set of functions). The integral equation then reduces to a fully-determined set of linear simultaneous equations in as many unknown coefficients as there are functions in the set (if 2 sets of functions are used there should be the same number of functions in each set).

In principle, the solution of a fully-determined set of linear simultaneous equations is an elementary problem. It is known, however, that the number of arithmetic operations required to solve exactly N linear simultaneous equations in N unknowns is proportional to N^3 (for large N), and if N is allowed to increase substantially beyond about 100 the solution time rapidly becomes intolerably long except on very fast computers. Moreover, the accuracy of such a calculation is limited only by the precision with which the coefficients are represented, so that the solution

may be given to 10 or 20 significant figures although all but the first 3 or 4 of these are invalidated by the inaccuracy of the physical model and the mathematical approximations employed.

The use of an iterative method of solution addresses both these objections. As will be shown later, the number of arithmetic operations required to perform one iteration on an N x N matrix of coefficients is approximately N^2, and so the solution time can in principle be substantially reduced if the number of iterations required is at least several times smaller than the order N of the matrix. This is generally possible whenever the solution is not required to attain great accuracy, which is precisely the case here. This paper describes an iterative method using banded matrices which, unlike the methods commonly used, does not rely on diagonal dominance in the coefficient matrix, but only on dominance by an arbitrarily wide band of diagonals centred on the principal diagonal.

2 THE ITERATIVE PRINCIPLE

The application of iteration to the problem of solving N linear simultaneous equations in N unknowns is most conveniently described in the notation of matrix algebra. Let the system be represented by the matrix equation

$$Ax = f \tag{1}$$

where A is an N x N matrix of coefficients and x and f are N x 1 matrices (N - dimensional column vectors) representing the unknowns and the right-hand sides of the equations, respectively. The iterative principle is applied to this system by separating the matrix A into 2 parts, one containing most of the large elements of A so that it will dominate the other part. If we refer to the parts as the large and the small part, denoted by A_l and A_s, we then have

$$A = A_l + A_s \text{ or } (A_l + A_s)\, x = f \quad \text{from equation (1), so} \tag{2}$$

we may iterate using the equation

$$A_l x^{(i)} = f - A_s x^{(i-1)} \tag{3}$$

where $x^{(i)}$ denotes the i^{th} approximation to the exact solution x of equation (1).

We begin the mathematical analysis of this process by deriving the condition for it to converge. Let the error of the i^{th} iteration be denoted by the vector $e^{(i)}$, thus

$$e^{(i)} = x^{(i)} - x \tag{4}$$

Then we have

$$e^{(i)} = x^{(i)} - x = A_1^{-1}(A_1x^{(i)} - A_1x) = A_1^{-1}(f - A_sx^{(i-1)} - A_1x)$$

from equation (3)

$$= A_1^{-1}((A_1 + A_s)x - A_sx^{(i-1)} - A_1x)$$

from equation (2)

$$= A_1^{-1}(A_s(x - x^{(i-1)}))$$

$$= -A_1^{-1}A_se^{(i-1)}$$

from equation (4), and so

$$e^{(i)} = (-A_1^{-1}A_s)^ie^{(o)} \tag{5}$$

It then follows from standard results in matrix theory (eg Ref 2, pp 111-112) that the necessary and sufficient condition for $e^{(i)}$ to tend to the null vector as i tends to infinity is that the moduli of the latent roots of the matrix $(-A_1^{-1}A_s)$ should all be less than unity, and that the convergence is ultimately of the type normally called linear (the number of correct significant figures in $x^{(i)}$ increases linearly with i) since $e^{(i)}$ and $e^{(i-1)}$ satisfy asymptotically the equation

$$e^{(i)} = \lambda e^{(i-1)} \tag{6}$$

where the scalar λ is the latent root largest in modulus of the matrix $(-A_1^{-1}A_s)$. This dependence on the latent roots of the matrix $(-A_1^{-1}A_s)$ shows why the parts of A should be "large" and "small" to bring about convergence. We can also see from equation (5) that the convergence or otherwise of the sequence of vectors $e^{(i)}$ does not depend on $e^{(o)}$, so if an iterative process of the kind represented by equation (2) converges at all it will do so from any starting point.

3 BANDED MATRICES

If the iterative method is to be practical, equation (3) must be economically soluble at each iteration, so the implied operation of inverting A_1 must be arithmetically much quicker than the implied operation of inverting A to solve equation (1) directly. This is commonly ensured by making A triangular or diagonal, as in the Gauss-Seidel and Jacobi methods (see Reference 2, p 179 et seq, for a description of these), and the algorithm for solving equation (3) is then a very simple one. Unfortunately, both these methods diverge in general unless the matrix A is diagonally dominant, and, although matrices arising out of antenna modelling problems usually have their largest elements on their principal diagonals, the other elements are not usually sufficiently relatively small.

The technique described in this paper addresses this problem by making the matrix A_1 a banded matrix; that is, a matrix which has null elements everywhere except on the principal diagonal and on a number of neighbouring sub-diagonals and super-diagonals. (This idea is presented in the paper cited as Ref 3, on which this work is based.) It is known (Ref 2, pp 17-18) that any square matrix may be resolved into a product of a lower-triangular matrix and an upper-triangular one, provided that the square matrix and all its principal sub-matrices are non-singular. In antenna modelling it is generally taken as an article of faith that the matrix of coefficients corresponding to an antenna will be non-singular if the exciting frequency does not correspond to any of the antenna's free resonances, and that at the free resonances no unique solution is possible *anyway* (ignoring the difference between the free resonances of the antenna and those of the model of the antenna). It is then argued that, since all the principal sub-matrices are the matrices associated with "sub-problems" (problems obtained by omitting some of the parts of the model), they will also be non-singular; and that the banded matrix A_1 and its principal sub-matrices are non-singular, since they are obtained from the full matrix A and its principal sub-matrices by simply ignoring some of the smaller interactions between parts of the model. This argument is patently unreliable, but it appears to be justified in practice. Now, when the banded matrix A_1 is so chosen that the number of sub-diagonals is equal to the number of super-diagonals (the common value being M, say), it turns out, as remarked in Ref 3, that the triangular matrices into which A_1 may be resolved are also banded and have the same bandwidth; that is, the lower-triangular matrix has nulls everywhere but on

the principal diagonal and on the M neighbouring sub-diagonals, and the upper-triangular matrix has nulls everywhere but on the principal diagonal and on the M neighbouring super-diagonals. As the matrix of coefficients A is roughly symmetrical (in a good model, the action of part i on part j will be approximately equal to that of part j on part i, on account of reciprocity) it is acceptable and, indeed, logical to choose A_1 such that the numbers of sub-diagonals and super-diagonals are equal. Consequently, resolving A_1 into a product of banded triangular matrices enables equation (3) to be solved quickly at each iteration; the resolution only needs to be done once and is arithmetically much quicker than resolution of a full matrix of the same order (as will be shown), and the solution of equation (3) becomes then merely a matter of forward and backward substitution, made even quicker than usual by the presence of extra nulls in the triangular matrices.

4 THE TRIANGULAR DECOMPOSITION

The discussion in reference 2 (pp 17-22) on triangular decomposition shows that it has a certain arbitrariness; there is, essentially, one degree of freedom in each row (or each column) of one of the triangular matrices, or the degrees of freedom may be shared between them. This is a consequence of the fact that the 2 triangular matrices between them contain $(N^2 + N)$ non-null elements constrained only by the (N^2) elements in the original N x N matrix, so that, in principle, we may choose N elements of the triangular matrices to have any values we please. In practice, it is found that the algorithm for determining the elements of the triangular matrices "grows" outwards from the principal diagonals of these matrices, and it is convenient to assign values to all the elements on one of the principal diagonals; usually they are all made unity. In the present application this is not very satisfactory, because the near-symmetry of the matrices A and A_1 means that the triangular matrices into which A_1 is resolved will have much the same internal proportionalities (the discussion in Reference 2, pp 142-147, shows that for any symmetric matrix there exists a decomposition into essentially equal triangular matrices, transposes of each other) but will be markedly different in their actual values, which may lead to rounding errors. This difficulty has been removed by adopting a triple decomposition, setting every element of the principal diagonals of the triangular matrices to unity and inserting a diagonal matrix between the 2 triangular matrices; that is,

$$A_1 = \begin{pmatrix}\text{lower triangular with}\\ \text{unit principal diagonal}\end{pmatrix}\begin{pmatrix}\text{diagonal}\end{pmatrix}\begin{pmatrix}\text{upper triangular with}\\ \text{unit principal diagonal}\end{pmatrix} \quad (7)$$

In this decomposition, the 2 triangular matrices are now comparable in size whenever A_1 is near-symmetric, they are still banded and of the same bandwidth as A_1 whenever that matrix is banded, and every element in the decomposition that has not already been specified as unity or null is uniquely determinable from the law of matrix multiplication. The uniqueness of this triple decomposition is proved in Appendix A.

5 THE DETERMINATION OF THE ELEMENTS

We will write the decomposition we intend to make, following equation (7), in the form

$$B = LDU \tag{8}$$

and particular elements of these matrices will be denoted by the corresponding small letter with suffixes indicating the row and column, eg b_{ij} is the element in the i^{th} row and j^{th} column of B. (A_1 has been replaced by B merely to make this convenient notation possible.) We now apply the law of matrix multiplication to derive an expression for each element in the product LDU, which equation (8) demands shall be equal to the corresponding element in B. We have

$$(ld)_{ij} = \sum_{k=1}^{N} l_{ik}d_{kj} = l_{ij}d_{jj},$$

since the matrices are of order N x N and D is a diagonal matrix: similarly

$$(du)_{ij} = \sum_{k=1}^{N} d_{ik}u_{kj} = d_{ii}u_{ij}$$

Hence we may write

$$(ldu)_{rs} = \sum_{i=1}^{N} (ld)_{ri}u_{is} = \sum_{i=1}^{N} l_{ri}d_{ii}u_{is}$$

$$= \sum_{i=1}^{N} l_{ri}(du)_{is} = \sum_{i=1}^{N} l_{ri}d_{ii}u_{is},$$

verifying the associative law of matrix multiplication.

Consequently we have

$$b_{rs} = \sum_{i=1}^{N} l_{ri} d_{ii} u_{is} \tag{9}$$

We now verify that banded L and U matrices give rise to a banded B matrix of the same bandwidth. L is a lower triangular matrix with only M non-null sub-diagonals, and U is an upper triangular matrix with only M non-null super-diagonals; so we have

$$l_{rs} = 0 \text{ for } r < s,\ l_{rs} = 0 \text{ for } r > s + M$$

$$u_{rs} = 0 \text{ for } r > s,\ u_{rs} = 0 \text{ for } r < s - M. \tag{10}$$

From conditions (10) we obtain $l_{ri} = 0$ for $i < r - M$ and $u_{is} = 0$ for $i > s$; consequently all terms in the sum $\sum_{i=1}^{N} l_{ri} d_{ii} u_{is}$ will contain a null factor and vanish, making b_{rs} vanish by equation (9), unless $r - M = i_{min} \leqslant i_{max} = s$ or $r \leqslant s + M$. Similarly we have $l_{ri} = 0$ for $i > r$ and $u_{is} = 0$ for $i < s - M$; so all terms in the sum in equation (9) will contain a null factor and vanish, making b_{rs} vanish, unless $s - M = i_{min} \leqslant i_{max} = r$ or $r \geqslant s - M$. Hence B must have null elements everywhere except on its principal diagonal and on the M sub-diagonals and M super-diagonals closest to its principal diagonal; that is, it must be a banded matrix of bandwidth M (or less). This does not, of course, prove that every such banded matrix possesses such a banded decomposition; indeed, no such proof can exist, since it is perfectly possible for a banded matrix to be singular or to have singular principal sub-matrices. It can however be proved that, if the banded matrix and its principal sub-matrices are all non-singular, a banded decomposition exists, which must then be the unique decomposition; for the algorithm we are about to derive will always produce a banded decomposition provided that note of the diagonal elements of D vanishes, and it is shown in Appendix A (using a proof based on that of the triangular decomposition theorem in ref 2, pp 17-20) that this condition is satisfied whenever the banded matrix and its principal sub-matrices are all non-singular.

It remains only to obtain the unspecified elements of L, D and U, which is done by reversion of the set of equations represented by equation (9).

6 THE REVERSION ALGORITHM

We begin the determination of an algorithm by expressing somewhat more compactly the information contained in equations (9) and (10) together with the conditions for the bandedness of B; thus:

Known	Unknown
$b_{rs}(r - s > M) = 0$	$l_{rs}(M \geqslant r - s > 0)$
$b_{rs}(r - s < -M) = 0$	$u_{rs}(-M \leqslant r - s < 0)$
$l_{rs}(r - s > M) = 0$	$d_{rs}(r - s = 0)$
$l_{rs}(r - s < 0) = 0$	
$l_{rs}(r - s = 0) = 1$	
$u_{rs}(r - s > 0) = 0$	
$u_{rs}(r - s < -M) = 0$	
$u_{rs}(r - s = 0) = 1$	
$d_{rs}(r - s \neq 0) = 0$	

Determining Equations

$$b_{rs}(M \geqslant r - s \geqslant -M) = \sum_{i=1}^{N} l_{ri} d_{ii} u_{is}$$

(note that the number of determining equations is the same as the number of unknown elements)

We introduce the two dyadic arithmetic operators $\uparrow$ and $\downarrow$, which are defined as follows:

$x \uparrow y = x$ if $x \geqslant y$ or y if $x \leqslant y$, ie the greater of x and y

$x \downarrow y = y$ if $x \geqslant y$ or x if $x \leqslant y$, ie the lesser of x and y.

The summary just given then becomes, on incorporating the known nulls into the equations for the unknown elements,

$$b_{rs}(M \geqslant r - s \geqslant -M) = \sum_{i=1 \uparrow (r-M) \uparrow (s-M)}^{r \downarrow s \downarrow N} (l_{ri} d_{ii} u_{is}) \qquad (11)$$

(Note that the operators ↑ and ↓ are associative and commutative.) These equations are most simply reversed by introducing the auxiliary quantities f_{rs} defined by

$$f_{rs} = l_{rs} d_{ss};$$

then we obtain, after some simple algebraic manipulations,

$$\left.\begin{aligned} f_{rs} &= b_{rs} - \sum_{i=1\uparrow(r-M)}^{s-1} (f_{ri} u_{is}) \quad (s = 1\uparrow(r-M), \overset{(+1)}{\ldots\ldots}, r-1) \\ d_{rr} &= b_{rr} - \sum_{i=1\uparrow(r-M)}^{r-1} (f_{ri} u_{ir}) \\ u_{rs} &= \left[b_{rs} - \sum_{i=1\uparrow(s-M)}^{r-1} (f_{ri} u_{is}) \right] / d_{rr} \quad (s = r+1, \overset{(+1)}{\ldots\ldots}, N\downarrow(r+M)) \\ l_{rs} &= f_{rs}/d_{ss} \quad (s = 1\uparrow(r-M), \overset{(+1)}{\ldots\ldots}, r-1) \end{aligned}\right\} r = 1, \overset{(+1)}{\ldots\ldots}, N \qquad (12)$$

where the value of $\sum_{i=a}^{b} (\sim) = 0$ whenever $b < a$. It will be seen that this algorithm is complete (every one of equations (11) is used, and every unknown element is calculated), consistent (no element or auxiliary quantity is calculated more than once, or referred to before it has been calculated), and will always succeed (the divisions in it are always permissible because none of the diagonal elements of D can vanish under the usual conditions, and all the other operations are unconditionally permissible). It can also be seen that the number of arithmetic operations required to perform the decomposition is approximately MN^2 for $M << N$; this result, quoted in Reference 3, was confirmed after the algorithm had been programmed, by putting counting variables in the loops of the program.

7 THE IMPLEMENTATION

Equations (8), (7), (3) and (12) now enable us to write

$$L_1 D_1 U_1 x^{(i)} = f - A_s x^{(i-1)} \qquad (13)$$

where the elements of L_1, D_1, U_1 and A_s are all known. If we now determine by forward substitution the column vector $z^{(i)}$ such that

$$L_1 z^{(i)} = f - A_s x^{(i-1)} \tag{14}$$

and then determine by division the column vector $y^{(i)}$ such that

$$D_1 y^{(i)} = z^{(i)} \tag{15}$$

we then obtain by taking equations (13), (14), and (15) together the result

$$U_1 x^{(i)} = y^{(i)} \tag{16}$$

which may be solved for $x^{(i)}$ by back-substitution. The existence and uniqueness of solutions to equations (14), (15) and (16) is guaranteed by the fact that the 3 coefficient matrices L_1, D_1 and U_1 are non-singular whenever the sufficient conditions for the triple decomposition of A_1 are satisfied.

From the standpoint of pure mathematics, this completes the problem; we have specified an algorithm and determined the conditions under which it can be applied and will converge. To complete the practical solution, however, it is necessary to provide a convergence criterion, which is usually based on the value of the residuals in the equations as expressed by the vector $Ax^{(i)} - f$; from equations (1) and (4) this vector may be written as $Ae^{(i)}$, and we can then use the ratio of the magnitudes of the vectors $Ae^{(i)}$ and Ax (= f) as an estimate of the ratio of the magnitudes of the vectors $e^{(i)}$ (= $x^{(i)} - x$) and x. Unfortunately this estimate is not necessarily a good one if the matrix A is ill-conditioned (ref 2, pp 103 et seq). Accordingly, our implementation of the method works by iterating until the error estimate is less than some chosen small number (eg 10^{-2} or 10^{-3}) and then taking 2 more iterations and using the 3 sets of results so obtained to obtain a final refined result, as follows. If the error in the 3 sets of results is determined solely by the latent root largest in modulus, the differences between them can be manipulated to eliminate the error altogether, since the errors will be decreasing geometrically. This assumption is, of course, never rigorously true, and will fail completely if there happen to exist several latent roots with the same largest modulus but with different phases; but in practice it is found to give a substantial increase in accuracy, up to an order of magnitude in some cases. The

calculation which obtains the final result by "eliminating" the error also provides us with the value of the convergence rate, and this value will in general be different for each component of the solution vector; hence the consistency (or otherwise) of the complete set of convergence rates affords a useful test of the assumption of geometric convergence, and if a particular run fails to converge we have information from which to estimate the increase in bandwidth required for convergence.

8 PROGRAMMING DETAILS

The algorithm has been implemented in Fortran IV, following preliminary experiments ing Basic, and the Fortran program listing with explanation is presented in Appendix B. It is assumed that the elements of A_s will be held in backing store, but the elements of A_1 must be kept in main store because the decomposition algorithm involves reading A_1 both by rows and by columns. This restriction has prevented full testing of the program to date, as practical problems with vehicle antennas involve A_1 matrices which are too large for 32000 words of main store (the maximum allocated to user programs on many mainframe computers, eg the ICL 1900 series under George III); however, tests with artificially constructed matrices, and with simple but genuine problems, have shown that the method does work and has been implemented correctly.

REFERENCES

1 Harrington, R F, "Field Computation by Moment Methods", Macmillan, New York, 1968.

2 Faddeev, D K & Faddeeva, V N, "Computational Methods of Linear Algebra" (translated from the Russian text), W H Freeman & Co, San Francisco, 1963.

3 Ferguson, T R et al, "Efficient Solution of Large Moments Problems", IEEE Transactions on Antennas and Propagation (March 1976), pp 230-235.

APPENDIX A

PROOF OF THE TRIPLE DECOMPOSITION THEOREM

It is desired to prove that, if a square banded matrix and all its principal sub-matrices are non-singular, that matrix may be written as the product, in order, of a lower-triangular matrix of the same bandwidth and with unit diagonal elements, a diagonal matrix with no null diagonal elements, and an upper-triangular matrix of the same bandwidth and with unit diagonal elements; and that this decomposition is unique.

From reference 2, pp 17-18, we can see that an arbitrary square matrix which is non-singular and has all its principal sub-matrices non-singular can be written as the product, in order, of a lower-triangular matrix with unit diagonal elements, a diagonal matrix, and an upper-triangular matrix with unit diagonal elements; that this representation is unique; and that the diagonal elements of the diagonal matrix are given by the equations

$$d_{11} = |A_1|\,,\quad d_{ii}\ (1 < i < N) = \frac{|A_i|}{|A_{i-1}|}\,,\quad d_{NN} = \frac{|A|}{|A_{N-1}|}\,,$$

where A is the matrix being decomposed, N is its order (strictly N x N), A_i is its i^{th} principal sub-matrix and d_{ii} is the element in the i^{th} row and i^{th} column of the diagonal matrix in the triple decomposition of A. Since A and all the A_i are non-singular, it follows from these equations that all the d_{ii} are non-zero. It therefore remains only to prove that the unique decomposition of a banded matrix A involves triangular matrices of the same bandwidth, and an implicit proof of this is furnished by the algorithm in the main text. Since this provides a triangular decomposition (subject to the condition that all the d_{ii} are non-zero, which we have just verified), it provides the (unique) decomposition, and the decomposition involves banded triangular matrices of the appropriate bandwidth, which completes the proof.

APPENDIX B

THE FORTRAN PROGRAM LISTING, AND ITS EXPLANATION

The program listing presented here divides into 4 sections: A is concerned with declarations, initialisation and input of the coefficients and righthand sides; B consists of the algorithm for decomposing the banded matrix into its lower-triangular, diagonal and upper-triangular components; C contains the start of the iteration, and its main loop; and D contains the iteration control code, the final refinement of the results and their output. These sections will now be explained in detail (we shall use the convention that, eg line A-31 refers to the 31st line in section A).

In line A-1, the workspace arrays are declared. The sizes given allow for a maximum matrix order of 69 and a maximum bandwidth of 20 (corresponding to 41 non-null diagonals). A is a scratch array, which normally holds the row of the main matrix currently being operated on; B is the array of righthand sides; Q holds the non-null coefficients of the banded matrix, suitably sheared in column index so as to convert a parallelogram into a rectangle. The variables QE, QE1 and QE2 in line A-2 are scratch variables, as are QR, QI and QQ in line A-3; QR and QI are generally used to hold the real and imaginary parts of QE. The variable SUMBB in line A-3 is used to hold the square of the magnitude of the vector of righthand sides B, which is computed later. The arrays FMTIN and FMTOUT in line A-3 are used to hold the object-time formats for input and output, respectively; the variable ERROR in this line holds the small number against which the error estimate is compared after each iteration. The variables N, MWIDTH, ITER and MAX in line A-4 hold the order of the matrix (strictly this is N x N), the bandwidth, the current iteration count and the maximum number of iterations allowed. The variables I, J, K and L in this line are used as indices for loops, and the 14 variables declared in line A-5 are scratch variables (used mostly for holding variable upper and lower limits for loops). The variable DONE in line A-6 becomes true when the iteration is complete, and the variable LARGER remains true as long as the current error estimate exceeds the value held in ERROR (as will be explained in context, these 2 conditions are not complementary).

In lines A-7 to A-12, the object-time formats, the matrix order, the permitted error, the permitted number of iterations and the bandwidth are read

in, and the tape or disc used for out-of-core storage is initialised. In lines A-13 to A-26, the coefficients and the righthand sides are read in, and the coefficients which comprise the non-null elements of the banded matrix are written to the array Q (the indices are adjusted so that the first non-null element in each row of the banded matrix is placed in the first element of the corresponding row of Q); the coefficients are then written out to a scratch storage device. In lines A-27 to A-37, the scratch storage file is closed and reset (for the first iteration), the square of the magnitude of the vector of righthand sides is computed and stored in SUMBB, and the vector of righthand sides B is copied into the arrays X and XB for later use.

Section B consists essentially of one set of nested loops, in which the algorithm described in the section "The reversion algorithm" is executed. The number of elements in the matrices of the triple decomposition is the same as the number of elements in the original matrix, if predetermined unit and null elements are left out of account, and the algorithm is so arranged that each element generated in the decomposition can over-write an element of the original band matrix stored in Q; in most cases this over-writing is performed more than once, the intermediate values being used in intermediate calculations. Each of the non-predetermined elements in the triangular and diagonal matrices is sheared in column index so as to preserve rectangular nature of the array Q; the result of this is that the diagonal elements of the diagonal matrix lie in the $(\text{MWIDTH} + 1)^{\text{th}}$ column of Q, and the non-predetermined elements of the triangular matrices are positioned touching on either side (lower-triangular on the left, upper-triangular on the right). With these facts borne in mind, it can be seen that section B is simply a translation of the reversion algorithm into Fortran.

Section C begins by initialising several variables (lines C-1 to C-5) and then jumps from line C-6 to C-31, because the first part of the iterative process is omitted on the first iteration (this is equivalent to taking $x^{(o)}$ and $x^{(-1)}$ as null vectors). In lines C-7 to C-27 we compute the elements of $(f - A_s x^{(i-1)})$ and store them in the array X (lines C-9 to C-21), and using the array XB which contains $(f - A_s x^{(i-2)})$ or $(A_1 x^{(i-1)})$ (see equation (3)) we compute, in lines C-22 to C-26, the square of the magnitude of $(Ax^{(i-1)} - f)$ (using the consequence of equation (2) that $(Ax^{(i-1)} - f) = (A_1 x^{(i-1)}) - (f - A_s x^{(i-1)})$) and copy array X into array XB, which is the

reason why XB contains $(f - A_s x^{(i-2)})$ when it is used the next time round. In lines C-28 to C-30 we compute the ratio of the magnitudes of the vectors $(Ax^{(i-1)} - f)$ and f, compare the ratio with the permitted error held in ERROR, set the logical variable LARGER according to the result of the comparison, and reset the file of coefficients for the next iteration.

The first and subsequent iterations now join together, with array X holding the vector $(f - A_s x^{(i-1)})$; and the rest of section C is simply the solution of equation (13), using the derived equations (14), (15) and (16). In lines C-31 to C-41 we solve equation (14) by forward substitution to find the vector $z^{(i)}$; in lines C-42 to C-44 we solve equation (15) by division to find the vector $y^{(i)}$; and in lines C-45 to C-56 we solve equation (16) by back-substitution to find the result of the current iteration, the vector $x^{(i)}$. We then update the iteration count and output its value and the values of 2 elements of $x^{(i)}$, as a check on the progress of the iteration.

Lines D-1 and D-2 contain tests for the completion of the iteration. Normally both these tests will fail, and control will pass to line D-3; in lines D-3 to D-6, the solution just computed will be copied into the array reserved for the previous solution and control will be returned to line C-7 where the next iteration begins. However, if on the previous iteration (not the one just completed but the one before that) the error estimate has achieved or improved on the permitted error, the test in line D-2 will succeed and control will pass to line D-7; this will also happen if the iteration just completed has brought the iteration count up to the maximum allowed number. In lines D-7 to D-12, the previous solution and the new solution are both moved one level to become the previous-but-one solution and the previous solution (respectively), the logic variable DONE is set to indicate that the coming iteration is to be the last one, and control is returned to line C-7 for this last iteration. When it is complete, control arrives at line D-1 as usual, and this time the test in this line is successful and control passes to line D-13. Assuming that by this stage the error in each of the unknowns is determined wholly or mainly by the latent root largest in modulus of the iteration matrix $(- A_1^{-1} A_s)$, we may write corresponding elements of the 3 solutions we now have as $(\alpha + \beta)$ (previous-but-one), $(\alpha + \beta\lambda)$ previous) and $(\alpha + \beta\lambda^2)$ (present), where α is the true value and λ is the latent root largest in modulus. In lines D-15 and D-16 the values of $(\beta\lambda^2 - \beta\lambda)$ and $(\beta\lambda - \beta)$ are computed and assigned to the

variables QE1 and QE2 respectively. If $(\beta\lambda - \beta)$ is found to be zero within the resolution of the computer, indicating that β is essentially zero or λ is essentially unity, the appropriate element of array A is zeroed (as a marker) and the present solution in array X is left unchanged (lines D-17 to D-19); this is consistent with the fact that if $\beta = 0$ or $\lambda = 1$ the sequence of solutions $(\alpha + \beta)$, $(\alpha + \beta\lambda)$, $(\alpha + \beta\lambda^2)$ cannot be improved on. If $(\beta\lambda - \beta)$ is not essentially zero, control passes from line D-17 to D-20, and in lines D-20 and D-21 we then compute λ and α and assign them to the appropriate elements of arrays A and X, using the identities $\lambda = (\beta\lambda^2 - \beta\lambda)/(\beta\lambda - \beta)$ and $\alpha = (\alpha + \beta\lambda^2) - ((\beta\lambda^2 - \beta\lambda)^2/((\beta\lambda^2 - \beta\lambda) - (\beta\lambda - \beta)))$. Finally, in lines D-23 to D-26 we write out the results and terminate the program run.

SUPPLEMENT TO APPENDIX B THE FORTRAN PROGRAM LISTING PART A

```
      COMPLEX A(69),B(69),X(69),XLAST(69),XLAST2(69),XB(69),Q(69,41)
      COMPLEX QE,QE1,QE2
      REAL FMTIN(8),FMTOUT(8),ERROR,SUMBB,QR,QI,QQ
      INTEGER N,MWIDTH,ITER,MAX,I,J,K,L
      INTEGER I1,J1,J2,J3,K1,K2,M1,M2,M3,IM1,JM1,KM,KM1,KM2
      LOGICAL DONE,LARGER
      READ (500,1000) FMTIN
      READ (500,1000) FMTOUT
 1000 FORMAT ((6X,8A8))
      READ (500,1010) N,ERROR,MAX,MWIDTH
 1010 FORMAT (I4,F10.0,2I4)
      REWIND 25
      DO 20 I=1,N
      READ (550) (A(L), L=1,N)
      READ (500,FMTIN) (B(L), L=1,N)
      J1=I-MWIDTH
      IM1=1-J1
      IF (J1 .LT. 1) J1=1
      J2=I+MWIDTH
      IF (J2 .GT. N) J2=N
      DO 10 J=J1,J2
         JM1=IM1+J
         Q(I,JM1)=A(J)
   10 CONTINUE
      WRITE (25) (A(L), L=1,N)
   20 CONTINUE
      ENDFILE 25
      REWIND 25
      SUMBB=0.0
      DO 30 I=1,N
      QE=B(I)
      QR=REAL(QE)
      QI=AIMAG(QE)
      SUMBB=SUMBB+QR*QR+QI*QI
      X(I)=QE
      XB(I)=QE
   30 CONTINUE
```

```
1.          M1=MWIDTH+1
2.          DO 170 I=1,N
3.          J1=I-MWIDTH
4.          IF (J1 .LT. 1) J1=1
5.          IM1=M1-I
6.          DO 110 J=J1,I
7.             K2=J-1
8.             IF (J1 .GT. K2) GOTO 110
9.             JM1=IM1+J
10.            KM=M1+J
11.            QE=Q(I,JM1)
12.            DO 100 K=J1,K2
13.               KM1=IM1+K
14.               KM2=KM-K
15.               QE=QE-Q(I,KM1)*Q(K,KM2)
16.   100      CONTINUE
17.            Q(I,JM1)=QE
18.   110   CONTINUE
19.         J3=I+MWIDTH
20.         IF (J3 .GT. N) J3=N
21.         J2=I+1
22.         K2=I-1
23.         IF (J2 .GT. J3) GOTO 150
24.         DO 140 J=J2,J3
25.            K1=J-MWIDTH
26.            IF (K1 .LT. 1) K1=1
27.            JM1=IM1+J
28.            KM=M1+J
29.            QE=Q(I,JM1)
30.            IF (K1 .GT. K2) GOTO 130
31.            DO 120 K=K1,K2
32.               KM1=IM1+K
33.               KM2=KM-K
34.               QE=QE-Q(I,KM1)*Q(K,KM2)
35.   120      CONTINUE
36.   130      Q(I,JM1)=QE/Q(I,M1)
37.   140   CONTINUE
38.   150   IF (J1 .GT. K2) GOTO 170
39.         DO 160 J=J1,K2
40.            JM1=IM1+J
41.            Q(I,JM1)=Q(I,JM1)/Q(J,M1)
42.   160   CONTINUE
43.   170   CONTINUE
```

```
      ITER=0
      LARGER=.TRUE.
      DONE=.FALSE.
      M2=M1+1
      M3=M1+MWIDTH
      GOTO 260
  200 QQ=0.0
      DO 250 I=1,N
      QE=B(I)
      READ (25) (A(L), L=1,N)
      J1=I-M1
      IF (J1 .LT. 1) GOTO 220
      DO 210 J=1,J1
         QE=QE-A(J)*XLAST(J)
  210 CONTINUE
  220 J2=I+M1
      IF (J2 .GT. N) GOTO 240
      DO 230 J=J2,N
         QE=QE-A(J)*XLAST(J)
  230 CONTINUE
  240 X(I)=QE
      QE=XB(I)-QE
      QR=REAL(QE)
      QI=AIMAG(QE)
      QQ=QQ+QR*QR+QI*QI
      XB(I)=X(I)
  250 CONTINUE
      QQ=SQRT(QQ/SUMSQ)
      LARGER = QQ .GT. ERROR
      REWIND 25
  260 DO 280 I=2,N
      QE=X(I)
      J1=M2-I
      IF (J1 .LT. 1) J1=1
      IM1=I-M1
      DO 270 J=J1,MWIDTH
         JM1=IM1+J
         QE=QE-Q(I,J)*X(JM1)
  270 CONTINUE
      X(I)=QE
  280 CONTINUE
      DO 290 I=1,N
      X(I)=X(I)/Q(I,M1)
  290 CONTINUE
      DO 310 I=2,N
      I1=N+1-I
      QE=X(I1)
      J3=I+MWIDTH
      IF (J3 .GT. M3) J3=M3
      IM1=I1-M1
      DO 300 J=M2,J3
         JM1=IM1+J
         QE=QE-Q(I1,J)*X(JM1)
  300 CONTINUE
      X(I1)=QE
  310 CONTINUE
      ITER=ITER+1
      WRITE (600,FMTOUT) ITER,X(1),X(N)
```

```
1.          IF (DONE) GOTO 430
2.          IF ((.NOT. LARGER) .OR. (ITER .EQ. MAX)) GOTO 410
3.          DO 400 I=1,N
4.          XLAST(I)=X(I)
5.      400 CONTINUE
6.          GOTO 200
7.      410 DO 420 I=1,N
8.          XLAST2(I)=XLAST(I)
9.          XLAST(I)=X(I)
10.     420 CONTINUE
11.         DONE=.TRUE.
12.         GOTO 200
13.     430 DO 460 I=1,N
14.         QE=X(I)
15.         QE1=QE-XLAST(I)
16.         QE2=QE-QE1-XLAST2(I)
17.         IF (CABS(QE2)) 450,440,450
18.     440 A(I)=(0.0,0.0)
19.         GOTO 460
20.     450 A(I)=QE1/QE2
21.         X(I) = QE - QE1*(QE1/(QE1-QE2))
22.     460 CONTINUE
23.         WRITE (600,1020) MWIDTH,ITER,MAX,ERROR,QQ
24.  1020 FORMAT (1H1,3I5,2X,2(2X,E14.7))
25.         WRITE (600,FMTOUT) ((L,A(L),X(L)), L=1,N)
26.         STOP
27.         END
```

DOCUMENT CONTROL SHEET

Overall security classification of sheet U/C

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the box concerned must be marked to indicate the classification eg (R) (C) or (S))

1. DRIC Reference (if known)	2. Originator's Reference Memorandum 3237	3. Agency Reference	4. Report Security Classification U/C
5. Originator's Code (if known)	6. Originator (Corporate Author) Name and Location RSRE		
5a. Sponsoring Agency's Code (if known)	6a. Sponsoring Agency (Contract Authority) Name and Location		

7. Title
AN EFFICIENT ITERATIVE METHOD FOR THE COMPUTER SOLUTION OF LARGE SETS OF SIMULTANEOUS LINEAR EQUATIONS APPLIED TO ANTENNA MODELLING

7a. Title in Foreign Language (in the case of translations)

7b. Presented at (for conference papers) Title, place and date of conference

8. Author 1 Surname, initials	9(a) Author 2	9(b) Authors 3,4...	10. Date	pp. ref.
HODGETTS, T E				

11. Contract Number	12. Period	13. Project	14. Other Reference

15. Distribution statement
HEAD COMMS GROUP, RSRE

Descriptors (or keywords)

continue on separate piece of paper

Abstract

Very large sets of simultaneous linear equations can usually be solved to reasonable accuracy more quickly by an iterative method than by a direct one, but most iterative methods only converge if the elements of the principal diagonal of the coefficient matrix dominate all the other elements. This paper describes an iterative method which depends for its convergence only on dominance by the elements of an arbitrarily wide band of diagonals centred on the principal diagonal. The theory of the method is fully worked out, and a computer program implementing it in Fortran IV is presented.