A REVIEW OF SOME RECENT DEVELOPMENTS IN THE SOLUTION OF LARGE SYSTEMS OF EQUATIONS

D.A.H. Jacobs
Central Electricity Generating Board, Leatherhead, Surrey, England

ABSTRACT

The numerical solution of engineering and scientific problems often involves solving large systems of algebraic equations. These systems may have a very sparsely populated coefficient matrix, or an almost full matrix, dependent on the method of discretization used. The methods of solving linear systems of such equations are broadly classified as either direct or iterative. The solution of non-linear systems, such as often arise when modelling saturable materials, for example, almost inevitably necessitates iterative solution, and for each iterative step a linear system is solved. In this paper some of the recent developments, particularly those that were not mentioned in the last COMPUMAG Conference, are reviewed. Both direct and iterative methods are covered. In addition some methods are included which are slightly older, but which warrant further exploitation.

1. INTRODUCTION

The increasing availability of computers in the past 20 years has promoted the extensive development of numerical methods for solving engineering and scientific problems. A number of general purpose methods have been widely used. Methods employing finite differences or finite elements have been used extensively for solving problems characterized by partial differential equations. An alternative method of solution is to formulate the problem in terms of an equivalent integral equation. Much research, particularly recently, has been directed towards the development of numerical methods for solving such integral equations.

Almost without exception, all these methods use a discretized model of the problem to characterize the continuous dependent variable. This consists of a finite set, often large, of variables which may represent function values directly or be coefficients of, for example, polynomials which represent the function over a finite subregion. The governing equations and boundary conditions are replaced by a system of simultaneous algebraic equations involving these discrete values. Frequently the system of equations is large. The coefficient matrix may be very sparsely populated, or be virtually full, dependent on the method being employed. Non-linear problems will in general produce a non-linear algebraic system to be solved. Frequently this is achieved by solving a sequence of linear problems.

Given a linear system of algebraic equations, two general classes of methods exist for their solution: direct and iterative. Although the basic form of these methods is now part of classical mathematics, there have been extensive developments during recent years.

Section 2 covers the principal developments in the direct solution of large systems of equations. In general these involve the exploitation of the sparsity or structure of the coefficient matrix. For systems of equations with full matrices, in general only features such as symmetry or positive definiteness of the coefficient matrix are available for improving solution methods and algorithm design. For matrices which are sparse considerable reductions in the computational work necessary to solve the associated system of equations have been achieved. Five topics are covered in the section. First the recent developments in the determination of optimal node orderings when using a band solution routine are outlined. Then Section 2.2 briefly reviews the use of the minimal degree algorithm for general sparse systems of equations. In Section 2.3 the method of nested dissection is described. It is based on the repeated geometrical dissection of the mesh used in finite difference or finite element models. Provided computation with known zero elements is avoided, the method is likely to be much more efficient than methods employing, for example, the frontal solution approach, particularly for problems in two dimensions. Section 2.4 is a brief description of some of the other highly efficient direct methods which are only applicable to special classes of problems, in particular those governed by separable equations. Fast Fourier transform techniques and cyclic reduction have been available for some time, but have lacked extensive use. In addition several of these techniques and methods can be employed in an iterative fashion for solving equations which do not satisfy the rather restrictive conditions. In Section 2.5 a novel method which has not yet received very much attention is briefly outlined. It is based on a marching technique for solving systems of equations obtained from elliptic differential equations. It exploits the extra precision over and above that required by the user that is available on many modern computers.

The major developments and advances in direct solution methods have been achieved in general by exploiting special features of the problem. For example the sparsity and patterned layout of the nodes is used by nested dissection, and the separability of the equation and the resultant coefficient pattern is exploited by methods based on fast Fourier transforms and cyclic reduction.

Analogously the majority of advances in iterative methods are based on the exploitation of certain features of the problem. However first, in Section 3.1, the determination of the optimal relaxation parameter for successive over relaxation is surveyed since this is still a much used method. Then in Section 3.2 the basic principle of the strongly implicit procedure is outlined, and the experience of its use over the last six years is summarized. Then in Section 3.3 a class of iterative methods which use the conjugate gradient method applied to a 'preconditioned' system of equations is described. They are likely to be the most interesting development of the decade in this field of iterative methods. Deferred correction is a method which can be employed to extend the application of many solution methods, both direct and iterative, to problems which do not satisfy all the criteria necessary for the particular method. Its use, particularly in association with some of the recent developments is outlined in Section 3.4.

Being a review paper, extensive mathematical detail has been omitted. References to the important papers in the various topics are given in the text. Additionally the paper by Reid[1] (and to a lesser extent Reid[2]) contain a wealth of detail on the direct solution methods for systems of equations. The paper by Fox[3] is a good review of iterative methods written before the publication of the recent papers on preconditioned conjugate gradient and marching methods. It does, however, include mention of both classes of methods.

2. DIRECT SOLUTION METHODS

2.1 Band Matrix Methods

Finite element models generally produce systems of algebraic equations which are positive definite. This means[4] that during a direct solution employing, for example, Gaussian elimination, LU decomposition or Cholesky factorization pivoting is not required. The coefficient matrix is generally very sparse because of the nature of the elements. Each node is associated with only a small number of elements, for each of which an algebraic equation is obtained. The avoidance of pivoting means that the sparsity pattern is not permuted during the elimination.

Band matrix techniques exploit the avoidance of pivoting and use the 'outline' sparsity structure. It can easily be shown that during the solution of a system of 'n' equations satisfying the above conditions, and for which all the non-zeros of the coefficient matrix lie within 'm' rows or columns of the main diagonal, the creation of non-zeros (termed fill-in) only occurs within the semi-bandwidth m about the main diagonal. However within that semi-bandwidth, fill-in tends to be almost complete. There is therefore no requirement to use a sophisticated form of compact sparse matrix storage, only the elements within the semi-bandwidth need be stored and operated on. Jennings[5] extended this result to the case of a variable bandwidth matrix. For a matrix of order n and semi-bandwidth m, Gaussian elimination employing band matrix techniques requires $O(\frac{1}{2} nm^2)$ computations contrasted with $O(1/3\ n^3)$ used by Gaussian elimination on a full matrix of order n. The storage is also reduced from n^2 to 2 nm. For typical engineering applications with n = 2000, m might be 200, and the computation is therefore reduced by about two orders of magnitude.

Irons[6] showed that in addition to reducing the computation, band matrix techniques can be used to reduce still further the in-core storage requirements during the elimination. With a semi-bandwidth of m, only a square matrix of order m need reside in core at any time since the elimination of a single variable can involve variables (and therefore the corresponding equation) at most m rows (or columns) away. The term 'frontal' approach is used because the elimination proceeds along a 'front' which traverses across the problem. The in-core storage required is then only m^2. All of the above figures can be halved, or almost halved in the case of storage, for symmetric matrices.

The importance of reducing the semi-bandwidth m is evident from the dependence of the amount of computation on m^2. Moreover the in-core storage requirement is also proportional to m^2. When using a solution procedure employing variable bandwidth, one seeks to reduce both the bandwidth of the matrix and also the profile, the profile of the matrix being defined as the sum of the semi-bandwidths of each row of the matrix.

Sometimes it is easy to order the variables by inspection so as to maintain a small bandwidth (and profile), but for complex mesh problems it is not always straightforward. The algorithm devised by Cuthill and McKee[7] has been very widely used. It is based on the comparison of several orderings each of which starts at a vertex of the graph which can be thought of as representing the matrix of coefficients. George[8] observed that the profile of the matrix may be further reduced by reversing the order of the nodes produced by the original algorithm. Space precludes a thorough mathematical treatment, for which readers are referred to the original paper[7] or to reference[9].

More recently Gibbs, Poole and Stockmeyer[9] have devised a new algorithm which has already achieved considerable use, and can provide substantial reductions, about 90% in some typical cases, in the computational effort required to obtain the node ordering. The bandwidths and profiles they obtained on an extensive series of test problems are comparable with those obtained using the Cuthill-McKee algorithm.

Reid[2] states that when solving systems of equations which do not have a positive definite matrix so that pivoting is necessary to maintain stability, then partial pivoting by rows or columns produces only about 50% more non-zeros than obtained by pivoting on the main diagonal. The amount of computation and in-core storage therefore increases by about 125%.

2.2 The Minimal Degree Algorithm

The frontal solution methods only attempt to minimize computation by reducing the bandwidth. Other algorithms have been developed which aim to minimize the computation directly. These more generally applicable approaches[1] to solving sparse systems of algebraic equations make use of the sparsity by using a data structure that stores only the non-zeros of the matrix and associated indicators giving their positions, or relative positions, in the matrix. In this way during the solution of a system of equations, full advantage can be taken of the sparsity structure and only those elimination operations involving non-zero elements are performed; all operations on known zero elements are avoided. The data structure must enable non-zero elements to be inserted and eliminated during the solution procedure as fill-ins and eliminations occur. A possible data structure could, for example, store for each row of the matrix a pair of vectors, one with real values storing the non-zero elements of the row, and the second with integer values storing the indices of the column in which the elements occur. In addition a third vector storing the integers which define the nearest non-zero by columns is

often specified to aid computational efficiency. Note that this is one of only several possible schemes for packed storage. Corresponding to such storage schemes, very efficient means have been developed for combining rows or columns of matrices stored in this way which only operate on non-zero elements. Clearly the overheads in terms of logical checking are relatively large, and such techniques are only beneficial, if the matrix is relatively sparse, typically less than 1/20 of the elements non-zero during the elimination. This result may need to be qualified for certain extreme types of problems such as those with a very small number of nodes.

For a general unsymmetric matrix a strategy for choosing the pivots is required and must consider not only the stability of the procedure but also the fill-in that is generated. Generally for stability a criterion based on the relative size of the pivot compared to the largest element in its row or column is used. Ordinary partial pivoting corresponds to the choice of one of the largest elements in the row or column and does not allow enough freedom for the exploitation of sparsity. Curtis and Reid[10] have found that the choice of criterion for the pivot determined by stability is not critical, and recommend, based on experiment, the selection of a pivot element not less than a quarter of the maximum element in the row or column. With this weaker selection of pivot elements based on stability, a greater emphasis can be directed towards reducing fill-in.

The Markowitz[11] criterion is frequently used to limit the fill-in during the solution of a system of equations when sparse storage methods are used. The LU or LDL^T factorization can be divided into n steps, where n is the order of the matrix. Each step comprises the elimination of one variable (or equivalently the factorization of one row or one column of the matrix). At the kth step of the elimination, a submatrix, A^k, of order $n - k + 1$ is to be factored. The Markowitz strategy is to find that element which satisfies the chosen stability criterion and which also has the smallest product of the number of other non-zeros in its row and number of other non-zeros in its column, and then by column and row interchanges to bring this non-zero element to the pivotal position. This criterion minimizes the number of multiplications for that stage of the elimination within the stability bounds.

For solving general systems of equations which have a symmetric and positive definite matrix, the Markowitz criterion constrained to diagonal elements reduces to selecting at each pivotal step the diagonal element with least non-zeros in its row as pivot element.

Reid[1,2] fully describes these types of solution procedure and their efficient implementation in computer software. Such software tends to be divided into three stages. The 'analyze' stage determines the form of the factorization. The factorization stage in which the factorization with the matrix values is actually performed using the procedure determined in the first stage. Finally the operate stage which uses the factorization obtained to solve the system of equations with given right hand side vector. Of these stages, the

first is generally the most expensive. However Reid[12] believes that there is potential for drastically reducing the computation in this phase particularly for symmetric positive definite matrices. (See also George and McIntyre[13].)

The less common case of solving non-definite symmetric matrices has also been resolved, and stable factorizations can be obtained in about the same computational effort as that taken by existing codes (that is without the probable savings mentioned immediately above) for the positive definite case. The strategy and the choice of pivots is however quite different. (See Duff, Munksgaard, Nielsen and Reid[14].)

2.3 Nested Dissection

George[15] has recently devised another method of solving systems of algebraic equations which aims to use the minimum of core storage and operations during the Gaussian Elimination. It is a method of ordering the nodes of a mesh so that the resulting set of associated algebraic equations can be solved with the near minimum of computation. The benefits are obtained by using a sparse matrix storage and computation system which stores and operates only on those elements of a matrix which are non-zero, and omits all operations on known zero elements.

Consider a rectangular array mesh of N by N nodes. For the commonly used five point molecule, each nodal value only occurs in the equation for that value itself and the four neighbouring values. For this problem George[15] was able to reduce the operation count to $O(N^3)$ and the storage to $O(N^2 \log_2 N)$, both of which are close to their minimum values. For typical two dimensional meshes these values are significantly lower than those derived using the frontal solution technique and not very much greater than those used by iterative techniques for each iteration.

As an illustrative example of the method, consider a rectangular region. In this each group of four nodes forming a basic rectangle combine to form an algebraic equation so that the 'connections' between nodes form the sides and diagonals of rectangles. This is the type of connection one has when using finite element models. The nested dissection ordering is obtained by first dissecting the region of interest into sub-regions of approximately the same size, each with the same shape as the basic elements. We therefore dissect the whole rectangle into four rectangles of approximately equal size. All the nodes on the common dissecting lines make up the set S_1. Each of the four regions is now itself dissected into four approximately equal sized rectangles, and the nodes on these lines of dissection which do not belong to set S_1 make up S_2. This process is continued until no further dissection is possible, i.e. in the case of a rectangle until single elements or blocks of two by one, or two by two elements remain. All these 'loose' nodes form the final set, S_q, say. The ordering of the nodes for assembling the system of equations then follows by taking first all the nodes in the set S_q, and then all the nodes in set S_{q-1}, and so on. Within each set the ordering of the nodes is arbitrary.

The analysis of the method is given by George[15], and alternative calculations of the operation counts are given in Allen[16] and Jacobs[17]. However the pertinent result is that, for example, for a topologically square region and mesh with N nodes in each coordinate direction, the nested dissection ordering requires only $O(N^3)$ operations to solve the system of equations compared to $O(N^4)$ operations used by the banded matrix techniques such as the Frontal solution. With a very small problem, say 10 by 10 nodes (N=10) the saving is only minimal, but with an average sized problem, say 32 by 32, the saving in computation is 96% and with a large problem, say 100 by 100 nodes (N=100) the saving is 99%, i.e. the amount of computation using nested dissection ordering and sparse matrix techniques is only 1% of the computation used by a Frontal solution method. It is interesting to note that the bandwidth of the matrix obtained using nested dissection orderings is in general close to, indeed often equal to, the order of the matrix.

In addition George[15] was able to prove a very valuable result for the two dimensional nested dissection ordering. In essence he showed that no ordering of the nodes could reduce the computation necessary in using Gaussian Elimination or related methods to a figure less than $O(N^3)$ for a square region. Thus to reduce the computation still further, an alternative method of solving the equations must be sought. The methods employing Fourier analysis and cyclic reduction for solving problems with separable equations (see Section 2.4) are an example of such an alternative method, as are the methods based on marching (see Section 2.5).

The core storage required can be calculated and is $O(N^2 \log_2 N)$ in place of $O(N^3)$ for banded methods. Like banded methods, not all this needs to be resident at one time, the amount depends on the sparse matrix equation solver routine used.

The method of nested dissection is readily extended to non-square regions and to regions using elements other than rectangles. It can also be extended to problems in more than two dimensions. However unlike the case for two dimensional meshes, George[15] was unable to prove that nested dissection provided the otpimal ordering for three dimensional meshes, and so there is possibly still scope for considerable improvement. The operation count when solving a problem in three dimensions, with say N nodes in each of the three coordinate directions, using the nested dissection ordering, is $O(N^6)$ which is to be contrasted with the $O(N^7)$ operations required if, for example, the Frontal solution procedure is used. Since typically the N used in three dimensional problems is smaller than that used for two dimensional regions, the saving is also generally less large. Even with such savings in computation iterative methods will remain in many cases the only feasible means of solving very large three dimensional problems. Where direct methods are used, however, fairly substantial savings in computation could be obtained by using the nested dissection ordering with sparse matrix techniques in place of band matrix techniques.

George[18] is currently developing methods which will be equivalent to nested dissection, and will therefore use close to the minimal computation and core store, but which employ the matrix structure to determine the order of elimination and not the geometrical configuration of the finite difference or finite element grid.

2.4 Fourier Analysis and Cyclic Reduction

The paper by Hockney[19] (see also Hockney[20], Buzbee, Golub and Neilson[21] and Fox[3] for general reviews) was one of the first to give details of a very efficient direct solution algorithm for solving the system of algebraic equations arising from a discretization of Poisson's equation on a uniform mesh. It is based on a numerical analogue of the method of separation of variables. The component of the solution dependent on each dependent variable is found separately. Hockney's[19,20] method consisted of applying a discrete Fourier analysis using the fast methods of Cooley and Tukey[22] in one of the coordinate directions. The Poisson equation

$$\nabla^2 u = g(x,y)$$

with, for example, the boundary conditions

$$u = 0 \quad \text{on} \quad x = 0 \quad \text{and} \quad x = \ell$$

$$\text{and on} \quad y = 0 \quad \text{and} \quad y = m$$

then becomes after substituting

$$u(x,y) = \sum_k \bar{u}_k(y) \sin(\pi k x/\ell)$$

$$g(x,y) = \sum_k \bar{g}_k(y) \sin(\pi k x/\ell)$$

the system of second order two point boundary value ordinary differential equations in the second dependent variable y

$$\frac{d^2\bar{u}_k}{dy^2} - \left(\frac{\pi k}{\ell}\right)^2 \bar{u}_k = \bar{g}_k(y) ,$$

with $\bar{u}_k = 0$ at $y = 0$ and $y = m$.

If the analogous discrete analysis is applied to the five point difference equations, then the range of k is equal to the number of mesh points in the x direction, and the solution of the ordinary differential equation for the Fourier coefficients requires the solution of a tridiagonal system.

In Hockney's algorithm, use is made of the method of odd/even reduction which exploits the identical form of the coefficients in all the difference equations that are obtained when solving Poisson's equation on a uniform rectangular mesh. By a simple process of

elimination, the equations for all the nodes on odd rows are used to eliminate these nodal values from the remaining equations for the even rows. This leaves a system of only half the size involving nodes on the even rows only. If the number of nodes in the co-ordinate direction is an integer power of 2, then this procedure can be repeated until only one set of equations remains. The number of odd-even reductions to be used can be optimized for the particular algorithms chosen (Hockney[19]) and the optimal number he finds to be approximately $\log_2 \log_2 N$ which is close to 2 for typical problems. The fast discrete Fourier analysis of Cooley and Tukey[22] can then be applied on the 'remaining' rows which may consist of all the even rows, or even just one row if the number of nodes along the rows is also an integer power of 2.

The resultant system of equations for the Fourier coefficients of the dependent variable on the 'remaining' rows are tridiagonal and can be solved by recursive cyclic reduction which consists of the recursive use of odd-even reduction. Hockney[20] advocates cyclic reduction in preference to the Gaussian elimination procedure for solving the tridiagonal matrices because, for the case that the boundary conditions are periodic, it avoids the use of an auxiliary vector; for the case of given boundary values (Dirichlet conditions) or normal gradient (Neumann conditions) there is little to indicate a preference for one method or the other. By using the equations for the Fourier coefficients obtained during the recursive cyclic reduction, all the Fourier coefficients for the 'remaining' rows can be built up. A Fourier synthesis is then used to generate the nodal values on all the 'remaining' rows from which the other nodal values can be obtained by using the equations used in the odd-even reduction of the original system.

The method depends on two specific features:

(i) the form of all the difference equations is the same, such as those obtained from Poisson's equation,

and (ii) the numbers of grid nodes in each coordinate direction are integer powers of 2.

However with such restrictions the computation on an N by N grid is only an astonishing $O(N^2 \log_2 N)$ operations, which contrasts with the $O(N^4)$ for a frontal solution method, or $O(N^3)$ for nested dissection.

The cyclic reduction algorithm requires care in implementation to avoid numerical instability. Buneman[23] developed an equivalent method which preserves stability. He developed a solution procedure which used double cyclic reduction for solving the full system of equations. However this has proved to be slightly less efficient than Hockney's algorithm.

A large number of advances have been made built around the basic method. Amongst these are the cyclic methods developed by Sweet[24] for general numbers of nodes, although powers of 2 remain the optimal choice. Several people, including Swarztrauber and Sweet[25] extended the application of the method to polar coordinate systems.

Perhaps the most important extension has been that due to Hockney[20], of the capacity matrix technique for solving Poisson's equation in non-rectangular regions. In outline the method first solves Poisson's equation within a circumscribing rectangle around the region of interest with suitably guessed boundary conditions. This solution will not in general satisfy the specified boundary conditions on the actual boundary and so an error vector composed of the errors on the nodes of this boundary can be determined. By using the capacity matrix for all the nodes on this inner boundary which relates the potentials to the charges on these nodes alone, the charges required to be placed on the real boundary nodes to compensate for the errors can be determined. A second Poisson equation is now solved with these additional charges added to the source term. The new potential obtained is that required. Hence the solution time is approximately double that for the circumscribing rectangular region. The capacity matrix has been generalized in a paper by Martin[26] to problems with general boundary conditions on non-rectangular regions - Hockney's original method was only applicable to Dirichlet problems.

Swarztrauber[27] has produced a generalization of the cyclic reduction algorithm which is applicable to general separable equations, and work is continuing on further extensions of the method to an increasingly wide class of problems. (Fox[3] gives a brief survey of some of the recent endeavours.)

For equations which are non-separable, Concus and Golub[28] developed a transformation which produces an iterative solution based on a separable system in a form of deferred correction (see Section 3.4). Consider the problem of solving

$$-\nabla.(D\nabla\phi) = s(x,y)$$

where $D = D(x,y)$.

Then we transform the dependent variable to

$$w(x,y) = [D(x,y)]^{\frac{1}{2}}\,\phi(x,y)$$

for which the governing equation becomes

$$-\nabla^2 w + pw = q$$

where $p = p(x,y) = D^{-\frac{1}{2}}\,\nabla^2(D^{\frac{1}{2}})$

and $q = q(x,y) = D^{-\frac{1}{2}}\,s(x,y)$.

This system of equations can be solved iteratively, using a form of deferred correction. Concus and Golub suggest the form

$$(-\nabla^2 + P)w^{k+1} = (P - p(x,y))w^k + q(x,y)$$

where $P = \frac{1}{2}(\min p + \max p)$.

However many families of iterative solution procedures could be used dependent on the type of problem and the form of the coefficient D(x,y).

Currently the only contenders for fast two dimensional Poisson solvers are the methods covered in this section and those termed 'marching' methods to be discussed in Section 2.5. If the problem fits the rather severe restrictions of the method, the only 'good' reason for not using these methods is the problem of developing the software, although packages are available.

The very substantial computational savings obtained by using these fast direct solution methods encourages their use in an iterative scheme, such as that described above for non-separable equations. They could also be used profitably for the inner iteration of other more general problems where previously an iterative or general direct procedure might have been used. Thus if a general problem can be reformed to be the limiting solution of say a succession of Poisson equations, then these powerful rapid elliptic solvers can be used for each step. For problems involving mild non-linearity this can sometimes be achieved quite easily. However electromagnetic problems involving highly saturable materials produce severely non-linear systems of equations for which there is I believe no general method currently available for obtaining the solution based on a Poisson equation solver.

Little attention seems to have been directed towards the methods covered in this section to three dimensional problems. Martin[29] is one reference to such implementations. It would appear that the computation required for solving large three dimensional problems may still be larger than that used by iterative techniques. However if a three dimensional problem can be split into layers of two dimensional ones, with relatively weak linking between the 'separate' two dimensional solutions, then a fast two dimensional solution routine can be used within an outer iteration in the third dimension. For certain problems, such schemes could be very efficient.

2.5 Marching Methods

In the class of direct equation solvers, perhaps the most 'novel' approach which has been developed over the last few years are the methods based on 'marching' (see Roache[30] and his forthcoming papers[31], Lorentz[32], Bank and Rose[33,34] and Bank[35]). They are basically extensions of the older shooting methods used for solving evolutionary problems governed by parabolic partial differential equations or for one dimensional boundary-value ordinary differential equations. In their unmodified form, marching methods when used to solve systems of algebraic equations obtained from elliptic differential operators have been known to be unsuitable because of the instability and associated rapid growth of errors. The modification of the method is founded on the assumption that many modern computers provide far greater precision than is required in the final result. The marching procedure can therefore be employed provided it is only used for a small number of successive steps in which the error growth could be kept sufficiently small.

First the basic marching procedure will be described. Consider a standard five point difference replacement of the Laplacian for a Poisson type differential equation resulting in a set of algebraic equations, each of the form

$$\phi_{i,j-1} + \phi_{i-1,j} - 4\phi_{i,j} + \phi_{i+1,j} + \phi_{i,j+1} = q_{i,j}$$

If the values of $\phi_{i,j}$ are known on two adjacent columns of the mesh, say j=1 and j=2, then the difference equation can be used to determine the value on j=3 by 'marching' forward. The procedure can be continued across the whole of the grid.

Two problems arise. The first is that when solving a Poisson type elliptic equation the boundary conditions do not provide values of ϕ_{ij} on two adjacent columns (or rows). Thus one is forced to 'guess' the values on say column j=2. The calculated values on j=3 will then be incorrect. However the three columns of values on j=1,2 and 3 can be substituted into the difference equations for the j=2 column which they will in general not satisfy. However the discrepancy will completely determine the difference between the correct values of $\phi_{i,2}$ and the incorrect guessed values. For a problem in which derivative or periodic boundary conditions are specified, the values on both the j=1 and j=2 columns would have to be guessed at the start. Both would then have to be corrected simultaneously after the first column march to j=3 by substituting three sets of values then available in the difference equation for the columns j=1 and j=2. In either case having corrected the 'guessed' values a second 'march' forward is performed and subsequently corrected.

In practice as is well known this procedure is unstable. Even after applying the correction procedure, the new values of $\phi_{i,j}$ will not be entirely correct. At the very least they will include round-off errors which will be accumulated during the marching forward. In the evaluation of $\phi_{i,j+1}$, the round-off error in $\phi_{i,j}$ is multiplied by 4 and added to other round-off errors. The resulting large error will in turn be multiplied by 4 when the value at $\phi_{i,j+2}$ is calculated, and so on. Thus for each additional column to which one marches, the accuracy will fall by a factor which approaches 5.83 because of the bit nature of storage. Eventually if this process is continued, the error swamps the solution. However if the computer possesses far greater precision than is actually required, and if the number of marching steps used is sufficiently small, then the procedure is acceptable. For example if the computer has 48 bit precision, and not more than 8 marching steps are used, then there should be a satisfactory growth in the error provided the final solution is not required to an accuracy greater than 24 bits (about

one part in 10^7). However if more than 8 marching steps are used, the growth in error would be unacceptable.

Clearly the marching can be performed both forwards and backwards, and in this way a problem with up to 18 mesh divisions could be solved satisfactorily to the above accuracy. For larger problems, Lorentz[32] has devised a partitioning method in which the marching is performed not just from the boundaries but also from internal lines generated within the region not more than 18 nodes apart. Thus suppose that the number of nodes in one direction is $2k\ell$ where k is not greater than 8. Then the procedure recommended by Lorentz is to set two lines of nodes to zero separated by $2k - 2$ nodes. Forward and backward marching is then used starting from the pairs of zero nodes spread throughout the region. The values are then corrected by the procedure outlined above employing a Fourier analysis to solve the systems of equations for the errors incurred.

The method described by Lorentz[32] is only applicable to problems in which the boundary conditions are periodic, and the total computation is about $\frac{4}{D} N^2 \log_2 N$ for a system of N^2 equations for an $N \times N$ region with D decimals of precision lost. The method is therefore, potentially, the fastest direct solver available. The program performing best on grids with the number of nodes in both directions are a power of 2.

Based on the test problem of solving $\nabla^2\phi = q$ in a 128 by 128 meshed rectangular region set for the GAMM conference held at Karlsruhe in March 1977, a program based on the Lorenz marching procedure would take about one half the time of the nearest 'rival'.

Lorentz[32] optimized his marching method which restricts its use to solving Poisson's equation with periodic boundary conditions. However the method can be extended to other classes of boundary conditions by using methods which are not quite so fast. Bank and Rose[33,34] describe a similar method which is applicable to a general class of block tridiagonal matrices which is only slightly less efficient than the highly optimized Lorenz procedure. In their two papers, they unify the description of fast direct solution methods. Bank[35] describes marching methods suitable for certain non-separable elliptic equations with non-constant coefficients. Roache[31] promises to describe methods, and experience of their use, he has developed for solving non-separable equations.

Undoubtedly with the wide use of computers such as the CDC machines offering substantial greater aritlmetic precision than is usually required in the solution, the use of marching methods is I believe an area offering significantly more efficient solution procedures for linear elliptic partial differential equations.

For non-linear problems, such as electromagnetic problems with non-uniform permeability, linearization will certainly have to be used if these methods are to be employed. I believe that deferred correction could be employed to considerable advantage for classes of problems for which a Poisson type equation is a good approximation.

The remarks made above are principally directed to two dimensional problems. For problems in three dimensions, a marching procedure in one direction would result in the requirement to solve a relatively full system of equations in the other two dimensions. Thus for an N by N by N problem, the computation is likely to be $O(N^6)$, which is of the same order as nested dissection. However this is an area where there is current research, and major progress may be made. The identification of the major structure in the algebraic system for the two-dimensional sub-problem might offer an iterative means of solution with considerably lower operation count.

3. ITERATIVE SOLUTION METHODS

3.1 Relaxation Parameters

Successive over-relaxation[3] (SOR) remains one of the most used iterative methods. Although there are undoubtedly more efficient iterative methods available for most problems, it remains one of the methods which is the most easy to code. The block line version is useful when solving problems with periodic boundary conditions, and the block multi-line methods[36] can be used to solve, for example, second order differential equations with a mixed derivative term, or a fourth order equation such as the Biharmonic. Much of the classical theoretical analyses and associated results are then applicable to such cases provided certain conditions are satisfied.

However the greatest difficulty remains the determination of the relaxation parameter. For simple problems the optimal value can be obtained analytically. But for most real problems, no such analysis is possible. Instead an algorithm is frequently used in which the value of the relaxation parameter is updated as the iteration proceeds. Virtually all such methods which are theoretically based depend on the coefficient matrix having Young's Property A and being consistently ordered. In addition a symmetric and positive definite matrix is often a pre-requisite.

It is well known (see Young[37] or Fox[3]) that

(i) it is better to overestimate the choice of relaxation parameter close to the optimal than to underestimate it by the same amount; and yet

(ii) a successive monotonic sequence of relaxation parameters converging to the optimal value can only be found adaptively by approaching from below.

In general because of the wide spectrum of eigenvector components in the error when commencing the iteration, it is generally best to use a range of relaxation parameters in the early stages of the iteration. The parameter value should then approach the optimal value after a reasonable number of iterations. Such a procedure can be used if the optimal value is known beforehand. If the optimal value is not known an adaptive algorithm can be used. Perhaps the best known is that due to Carré[38]. However the procedure cannot guarantee that overestimates of the optimal relaxation would not be obtained, an undesirable feature.

Reid[39] gives an alternative procedure which appears to have lacked the attention it deserves. Again he uses the ratio of the norms of successive displacement vectors to estimate the largest eigenvalue of the matrix. However he then uses the displacement vector as an estimate of the dominant eigenvector of the iteration matrix, and hence by dividing by the eigenvalue one has an estimate of the dominant eigenvector of the Jacobi matrix (see Reid[39] for full details). The Rayleigh quotient[4] is then formed to give a good estimate of the maximum eigenvalue of the Jacobi matrix. It can be shown to be an underestimate, as is desired. In the experimental results reported by Reid, his method did sometimes require a few more iterations to obtain a solution of specified accuracy than the method of Carré or Kulsrud[40]. However the guarantee of never exceeding the optimal parameter and secondly of approaching it very closely was illustrated in the most difficult problem Reid considered, a problem of the order of difficulty that arises in 'real' problems. Reid's method is undoubtedly one worthy of consideration if SOR is being used.

Regrettably, the matrices obtained with many 'real' problems do not always have Young's Property A. They are also often non-linear so that the eigenvalues and eigenvectors change as the solution is approached. The best advice is probably to use a method such as Reid's with a relatively slow approach to the optimal parameter.

If a sequence of linear problems is to be solved, all governed by the same coefficient matrix, the optimal relaxation parameter will be the same for all the problems. It is then often worthwhile deploying some effort to obtain the optimal parameter value. If a hypothetical problem modelled by the same matrix can be generated which has a known solution as can sometimes be done, then the actual error is known during the iteration so the maximum eigenvalue of the iteration matrix can be obtained more accurately and hence the optimal relaxation parameter obtained. Furthermore such a procedure has other uses. The residual at each stage of the iteration can be related to the actual error when solving the problem with known solution. This relationship can then be used when solving the 'real' problems to obtain a measure of the error. (This method of error determination is closely analogous to that proposed by Zadunaisky[41] for use when solving ordinary differential equations.)

Despite this rather lengthy section on relaxation parameters, it is my view that the methods of the succeeding two subsections are generally vastly superior to SOR.

3.2 The Strongly Implicit Procedure

The Strongly Implicit Procedure (SIP) developed by Stone[42] is an implicit method which derives a new solution of a system by algebraic equations simultaneously, that is each new value obtained depends on all the other new values. This is in marked contrast to the classical iterative procedures in which new values are obtained at single points as in SOR or at small sets of nodes, for example lines as in alternating direction implicit methods[3]. In this respect the implicit nature of SIP closely resembles a fully direct solution procedure. Indeed the method is best described and formulated as an 'approximate' direct method, and because it is only 'approximately' direct, it has to be used repeatedly to obtain convergence, i.e. iteratively.

The basic form of most iterative methods is a factorization of the coefficient matrix A into the sum of two matrices B and C so that

$$A = B + C .$$

The factorization is chosen so that the matrix B is 'easily' inverted, typically in O(n) computations where n is the order of the matrix A. The iteration procedure is then based on the repeated use of the up-date equation

$$B\underline{w}^{k+1} = \underline{b} - C\underline{w}^{k}$$

where k is the iteration index. Since B is easily inverted, this system can be solved very economically. By subtracting $B\underline{w}^{k}$ from both sides one obtains

$$B\,\underline{\delta w}^{k} = \underline{r}^{k}$$

where $\underline{\delta w}^{k} = \underline{w}^{k+1} - \underline{w}^{k}$ is the change vector

and $\underline{r}^{k} = \underline{b} - A\underline{w}^{k}$ is the residual vector.

It is clear from the form of the iteration that if it converges, which implies $\underline{\delta w}^{k} \to \underline{0}$, then $\underline{r}^{k} \to \underline{0}$ and hence $\underline{w}^{k}$ converges to the solution of the original equation.

The SIP employs an approximate LU factorization of the coefficient matrix A. The exact LU factorization normally takes $O(1/3\ n^3)$ computations for a full matrix, or $O(1/2\ nm^2)$ computations for matrix of order n and semi-bandwidth m. The SIP performs the LU factorization on a matrix B which is 'close' to the matrix A and has the important feature that the factorization and approximate equation solution requires only O(n) computations. This is achieved by restricting the matrices L and U to have non-zero elements on the main diagonal and on those other diagonals which occupy the same positions as such elements of the coefficient matrix A in its lower and upper triangles respectively. The product of two such matrices

$$LU = B$$

has non-zero diagonals in the same positions as those of the matrix A, together with a number of additional ones.

For example consider the case of Poisson's equation using a five point finite difference molecule replacement in a rectangular region. The matrices L and U will each have only three non-zero diagonals.

The product of two such matrices has seven non-zero diagonals, five in positions coincident with those of A and two additional ones adjacent to the outermost diagonals of A, as illustrated in Fig. 1. In determining the coefficients of the matrices L and U there are five values to be obtained on each row, one value is redundant and an efficient exploitation of this sets the main diagonal of the upper triangular matrix U to unity. With the five values on each row of the matrix A given, L and U are uniquely determined if the matrix B is made identical to the matrix A on the non-zero diagonals of the latter.

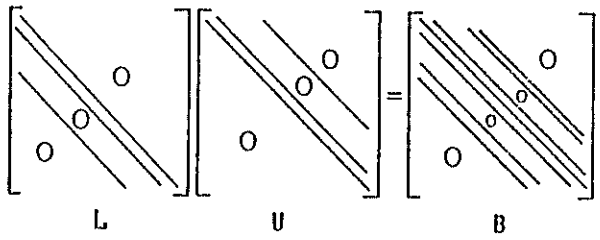


FIG.1 THE PRODUCT OF TWO THREE DIAGONAL MATRICES

However, non-zero elements also appear on two other diagonals of the matrix B. In fact the matrix B represents a seven point molecule involving the five of Fig. 2 together with the two additional nodes (i+1, j-1) and (i-1, j+1), with coefficients $b_{i,j}e_{i,j-1}$ and $c_{i,j}f_{i-1,j}$ respectively.

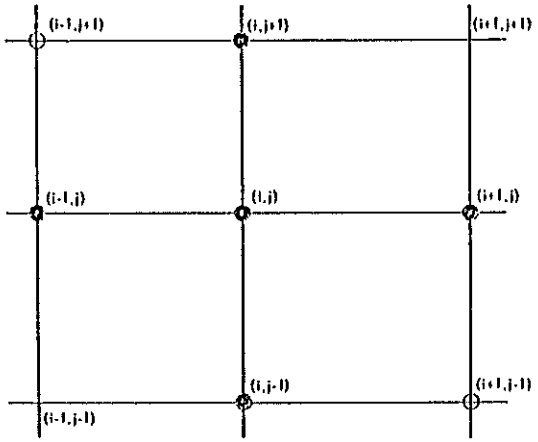


FIG. 2 THE GRID POINTS IN THE VICINITY OF THE (i,j) NODE

To reduce the effect of these additional terms, Taylor series approximations are used to relate the two additional values in the seven point molecule to the five 'central' point values. At this point the iteration parameter α is introduced. This parameter is used to improve the convergence rate of the scheme by 'implicitly' cancelling α times the extra values introduced.

Stone suggests a set of 9 values for the parameter α. The parameters are used cyclically and each is used for two iterations; for the first iteration the equations are scanned in one direction, and for the second the order is reversed. This alternating order means that the extra nodes introduced by the approximate LU factorization are those already detailed on one iteration and the other two diagonal terms, namely (i+1, j+1) and (i-1, j-1), on the second; this introduces some symmetry into the scheme. The particular cyclic use of parameters means that over each set of 6 iterations a fairly wide spread of the parameter α is employed. For many problems this assists efficiency and frequently the solution converges after only a few iterations.

Extensions to the method originally proposed by Stone for the five point molecule are possible for solving systems of equations derived from other difference molecules. Of these the seven point scheme[43] for three dimensional Poisson type problems is perhaps the most used.

Within the C.E.G.B. we have had the use over the last six years of a number of subroutines based on SIP for solving general five diagonal (e.g. two dimensional Poisson type equations), seven diagonal (e.g. three dimensional Poisson type) and thirteen diagonal[44] (e.g. two dimensional fourth order equations such as the Biharmonic) systems of equations which frequently arise in finite difference modelling. They have been used on a diverse range of problems, including a number involving the solution of non-linear equations. The significant power in terms of the efficiency, and the ease of use assisted by the alleviation of the need to optimize acceleration parameters for the vast majority of problems have been the principal attributes. However for certain large problems, for example a Poisson's equation on a 70 by 70 mesh, SIP has produced very slow convergence. In general the method is quite satisfactory for most two dimensional problems with upto about 3000 nodes. In three dimensions, where a seven diagonal system of equations is obtained, the extended method is just as satisfactory; indeed the computational work to obtain a solution of reasonable accuracy is proportionately very much smaller compared to general direct solution methods for two dimensional problems. This is because for an $N \times N$ problem in two dimensions, even the method of nested dissection would solve the system in $O(N^3)$ computations, and each iteration of SIP takes $O(N^2)$ computations. In three dimensions, the comparable figures for an $N \times N \times N$ problem are $O(N^6)$ for nested dissection and $O(N^3)$ per iteration for SIP. A significant feature of the SIP method is the weak dependence of the rate of convergence on the number of nodes of the problem, and even weaker dependence on the dimension of the problem.

The major disadvantages of SIP compared to classical iterative techniques are

(i) the increased number of arrays needed to store some of the matrix factors, although the storage is still much less that that required by most general equation solvers, including those using sparse storage techniques,

(ii) that to preserve the structured diagonal form of the coefficient matrix necessary for SIP, it is necessary to circumscribe non-rectangular regions by a rectangle of grid lines and to include all nodes within this circumscribing rectangle. Although this increases storage, much of the additional computation can be avoided, and

(iii) that the mesh must be topologically rectangular, to maintain the structure of the coefficient matrix, thus local mesh refinement must be achieved, for example, by embedding sub-problems with fine meshes within the actual problem using a coarse mesh.

However these penalties are generally overshadowed by the very much faster convergence achieved by SIP than by classical iterative methods. Fast convergence means that not only is a solution obtained with the expense of less computation, but also the error in the solution for a given convergence criterion is generally less for a fast convergent method.

Even today after ten years, the Strongly Implicit Procedure remains one of the most powerful iterative techniques for solving banded systems of equations such as those that arise from finite difference models of elliptic partial differential equations, both linear and non-linear. For many three dimensional problems involving a large number of nodes, iterative methods such as SIP are currently the only viable solution technique. It is still too early to decide whether the use of the conjugate gradient method on a preconditioned matrix described in the next section will totally supersede SIP.

3.3 Conjugate Gradients with a Preconditioned Matrix

The conjugate-gradient method (Reid[45]) is basically a direct method in conception because using exact computation a finite number of computations provides the solution. However with calculations performed on a digital computer with finite precision arithmetic, the method behaves much more like an iterative procedure. For a system of algebraic equations of order n, the method using exact arithmetic would involve at most n steps. At each step the solution is progressively refined so that at the completion of the final step the exact solution is attained. With finite precision arithmetic, each step progressively improves the solution, but the exact solution attainable to the computer's precision is obtained only after an infinite or at least very large number of steps.

However two points in real applications are of significant relevance. First, generally one only requires a solution to a finite precision, say three or four decimal places. Thus for a linear problem, if a reasonable rate of convergence can be achieved, only a small number of iterations of an iterative method need be used. For a non-linear problem in which the solution is obtained by successive steps of linearization, the solution of each 'linear' step generally starts from a good approximation, namely the last approximate solution. In addition only an approximate solution is required since the equations being solved are only linearized approximations of the real set. Secondly, when using the conjugate gradient method as mentioned above, each of the n steps successively refines the solution. Combining these two features means that the conjugate gradient method can be seen when used for solving real problems as an iterative method.

The rate at which the solution is refined when using the conjugate gradient method depends on the matrix of coefficients and in particular on the number of distinct eigenvalues. Consider, for example, the unit matrix. Convergence, in fact the exact solution, is obtained in only one step, namely one iteration. Indeed the smaller the number of distinct eigenvalues, in theory, the faster the solution will be obtained. When solving a 'real' problem, one has little control over the eigenvalues of the coefficient matrix of the algebraic equations, apart from, for example, changing the grid. Thus, in particular, the number of distinct eigenvalues is not controllable.

However, Meijerink and van der Vorst[46] have shown how a preconditioning of the coefficient matrix of the equations to be solved can be used to produce a matrix with all, or at least most of the eigenvalues contained in a narrow range. They used an 'approximate' Cholesky factorization of the coefficient matrix, similar in concept to the approximate LU factorization used by Stone in the SIP. If A is a positive definite and symmetrix matrix, both of which are necessary conditions for the conjugate gradient method, then the approximate Cholesky factorization of the coefficient matrix A is given by

$$A \doteq L'DL'^{T} \qquad \text{or} \qquad LL^{T}$$

where L' (and L) is lower triangular. In the method proposed the matrix L' (and L) is constrained to have the same (or closely related) sparsity pattern to that of the matrix A. They then use the conjugate gradient method to solve not the original system

$$A\underline{x} = \underline{b} ,$$

but the preconditioned system

$$(L^{-1} A L^{-T})\, L^{T}\underline{x} = L^{-1}\underline{b}$$

i.e. $$B\underline{y} = \underline{x}$$

where $$B = L^{-1} AL^{-T}$$

$$\underline{y} = L^{T}\underline{x}$$

$$\underline{c} = L^{-1}\underline{b} .$$

With the particular choice of the matrix L, we know that

$$B \doteq I .$$

Indeed if L had been the exact Cholesky factor of A, then B would have been the identity matrix and the conjugate gradient method would yield the exact solution in just one step. However since the matrix L is only an approximate factor of A, several steps are likely to be required. However the computation involved in determining L and using it in matrix-vector multiplications is very small because of its forced sparsity.

Meijerink and van der Vorst developed the method for a coefficient matrix A which has the property of being a symmetric M-matrix. The latter property is characterized by a non-singular matrix with all coefficients off the main diagonal less than or equal to zero, and with $A^{-1} \geq 0$. The coefficient matrices arising from many finite difference and finite element models are M-matrices. They proved a number of results, including ones on numerical stability for classes of preconditioning. The two methods they proposed for solving five diagonal systems of equations were based on

(i) selecting the factorization matrix L to have the same sparsity pattern as the matrix A in the lower triangle, i.e. to be three diagonal, and

(ii) selecting the factorization matrix L to have six non-zero diagonals, namely those of the matrix A and one additional adjacent to the two along the centre of the matrix, and two additional on the main diagonal side adjacent to the outermost diagonal.

Their test results indicated that both methods provided substantial improvements in efficiency over existing iterative methods, and that of the two, the second choice of sparsity was the better.

Kershaw[47] describes very favourable experience of using the method on symmetric positive definite matrices obtained from models used in laser fusion simulation. In addition he proposed extensions to the method for the cases of a general positive definite symmetric matrix and of a general non-singular matrix. The latter extension makes use of an approximate LU decomposition and the version of the conjugate gradient method applied to the normalized equations which have a symmetric coefficient matrix.

Our own recent work (to be published[48]) is a development of the preconditioned conjugate gradient method specifically for solving the systems of equations derived from finite difference models of partial differential equations. The coefficient matrix need not be symmetric and is often non-linear. The preconditioning uses an approximate LU factorization derived along the lines of that used in the strongly implicit procedure. Namely the banded sparsity structures of L and U are the same as that of the matrix A and an implicit cancellation of extra terms introduced is used. Such a factorization makes great use of the finite difference molecule from which the system of equations is derived. Indeed for the system of equations derived from a five point molecule approximation, by using the cancellation of the extra terms introduced in the approximate LU factorization (see Section 3.2) we know that

$$A = LU + O(\Delta x^3, \Delta y^3, \Delta x \Delta y)$$

and hence the 'approximate' factorization is relatively very accurate. The preconditioning of the system of equations using these factors L and U therefore produces a coefficient matrix which differs little from the identity matrix, and hence the conjugate gradient method should converge very rapidly.

Our preliminary test results show that for two dimensional problems with less than about 3000 nodes, the new method is directly comparable in performance to the strongly implicit procedure. For problems with excess of 3000 nodes the new method provides substantially faster convergence rather than the strongly implicit procedure. Even with about 22 500 nodes the rate of convergence is only about ½ that of a problem with 3000 nodes. We are developing general purpose solution routines based on the new method for various different multi-diagonal systems (e.g. 5 diagonal for a two dimensional problem and 7 diagonal for a three dimensional problem involving a second order partial differential equation) and also endeavouring to evaluate the method on a range of problems.

There is little doubt, even at this relatively early stage, that preconditioned conjugate gradient methods will offer a powerful alternative iterative solution method to the classical iterative techniques. They would appear also to be substantially more efficient and reliable than the strongly implicit procedure if only for two dimensional problems with a large number of nodal points.

3.4 Deferred Correction

The method of deferred correction (representing developments of the original ideas of Fox[49], see also Fox[3]) is a general technique which can be used in many forms to assist in the solution of systems of equations which do not precisely fit the criteria necessary for a particular specialized solution procedure. Already in Sections 2.4 and 2.5 mention has been made of using the methods developed for solving separable equations in an iterative mode for solving certain non-separable equations. This is a form of deferred correction (or perhaps more descriptively termed defect correction).

Basically deferred correction is a solution procedure which has a similar form in matrix terms to most of the iterative procedures. In place of solving the specified system of equations

$$A\underline{x} = \underline{b}$$

we use an iterative procedure characterized by the equation

$$A' \, \delta\underline{x}^k = \underline{b} - A\underline{x}^k$$

where $\underline{x}^{k+1} = \underline{x}^{k} + \underline{\delta x}^{k}$.

The matrix A' is selected to approximate closely the matrix A but to be of a form suitable for solution by the selected method which might be direct or iterative. For example suppose that nearly all the non-zero elements of the matrix A lie on five diagonals. A few non-zero elements, perhaps introduced by incorporation of derivative boundary conditions on a curved boundary, might lie off these five diagonals. The solution of such a system using a technique, for example the strongly implicit procedure, which depends on the banded structure of the matrix is therefore made more complex by these extra non-zero terms. However a form of deferred correction can be used in which the solution procedure for the calculation of the correction term uses only the coefficients conforming to the strict diagonal pattern. Thus the 'iterative' scheme becomes

$$A_5 \, \underline{\delta x}^{k} = \underline{b} - A \, \underline{x}^{k}$$

where A_5 is the matrix consisting of only the five main non-diagonals of the matrix A.

We have used this method[50] to obtain high accuracy solutions to problems using a low accuracy difference replacement to generate the solution procedure, but calculating the residuals using the high accuracy form. For a number of trial problems for which the strongly implicit procedure was used, high accuracy solutions were obtained at little extra computational cost to that required to obtain solutions to the less accurate system used as the basis of the solution procedure.

With the development of very efficient solution procedures, such as the fast Fourier transform and marching methods, and the fast iterative methods, SIP and preconditioned conjugate gradients, the use of deferred correction is a powerful means of extending the use of these new methods to a wide range of problems. For example Hockney's procedure of Section 2.4 is only applicable to the solution of problems governed by a constant coefficient differential equation. When solving a problem with mild non-linearity, his procedure could be used in an iteration using deferred correction. The matrix A' would be chosen as that formed from the linearized constant coefficient problem. At each iterative step the direct solution procedure would be used to correct the dependent variable values in accordance with the residuals determined from the full non-linear equation. Although I have seen no mention of the use of such methods, as the computation of increasingly complex problems is pursued they might well provide a powerful additional solution tool.

Another form of deferred correction can be applied to the solution of systems of equations with a virtually full coefficient matrix. Such systems arise with many of the integral methods and for which little of the foregoing sections is directly applicable. However the dominant, or at least relatively important, features of the problem may be characterized by just a few of the coefficients of the matrix. Frequently the dominant features will be those characterized by the largest elements on each row of the coefficient matrix. If these are of the form or in the pattern that makes them amenable to solution by one of the methods already described, e.g. five diagonal form for the SIP or preconditioned conjugate gradient method, then the vast computational savings they offer can be partially exploited by using an iterative scheme based on deferred correction in which the inner solution uses an update procedure based on the specially structured matrix of coefficients.

Regrettably such algorithms are likely to be very problem and geometry dependent, and therefore do not lead themselves to easy incorporation in general purpose packages. However for problems involving very large systems of equations (n very large), where the direct solution usually takes $O(n^3)$ computations, the use of such an iterative procedure could offer very substantial savings. If the system of equations is non-linear, an outer iteration is probably being used, and can be combined with an inner iteration based on a suitable matrix reduction. However for highly non-linear problems it may prove very difficult to obtain a satisfactory matrix reduction since the elements of the coefficient matrix may change very rapidly.

4. CONCLUSIONS

The majority of the developments in the solution of large systems of equations have exploited particular features of the coefficient matrix. The method of nested dissection compared to say methods using banded matrix techniques can provide fairly large reductions in the computation required to solve systems of equations directly, particularly for large two dimensional problems derived from finite element or finite difference models. Although the method is based on the geometrical features of the grid, George is developing equivalent methods which it is believed will produce the same effect but will use only the matrix structure. The marching methods, on which considerable development has taken place recently, now join the methods based on fast Fourier transform and cyclic reduction, as being very rapid equation solvers for Poisson and separable linear equations. Both groups of methods are restricted to solving certain classes of systems of equations, although extensions have been made and are under development to enable the methods to be applied to more general types of equations. For two dimensional problems the methods are highly efficient and use little more computation than a single iteration of iterative methods. For three dimensional problems the savings in computation are large, but for large problems in three dimensions most forms of such methods are still prohibitively expensive.

The development of iterative methods has advanced. The new procedures generally employ some of the features of direct methods, and the 'difference' between direct and iterative, at least in conception, is being narrowed. Like the advances made in direct solvers the iterative procedures tend to use features of the structure of the matrix. The Strongly Implicit Procedure exploits the difference molecule from which the system of algebraic equations is derived. The more recent development of preconditioned conjugate gradient

methods use the structure of the matrix and therefore are more widely applicable than SIP. Preliminary results also indicate that some of the latter class of methods provide substantial improvements in convergence rates for large two dimensional problems compared with SIP, for example. Such iterative procedures are permitting the economic solution of increasingly large three dimensional problems.

5. ACKNOWLEDGEMENT

The work was carried out at the Central Electricity Research Laboratories and is published by permission of the Central Electricity Generating Board.

6. REFERENCES

1. Reid, J.K., 1977, Sparse Matrices. In 'The State of the Art in Numerical Analysis', edited by D.A.H. Jacobs, Academic Press, London.

2. Reid, J.K., 1978, Software for Sparse Matrices. In 'Numerical Software - Needs and Availability', edited by D.A.H. Jacobs, Academic Press, London.

3. Fox, L., 1977, Finite-difference Methods for Elliptic Boundary-value Problems. In 'The State of the Art in Numerical Analysis', edited by D.A.H. Jacobs, Academic Press, London.

4. Wilkinson, J.H., 1965, 'The Algebraic Eigenvalue Problem', Oxford University Press.

5. Jennings, A., 1977, 'Matrix Computations for Engineers and Scientists', John Wiley, London.

6. Irons, B.M., 1970, A frontal solution program for finite element analysis, Int. J. Numer. Meth. Engng., Vol. 2, 5-32.

7. Cuthill, E. and McKee, J., 1969, Reducing the Bandwidth of Sparse Symmetric Matrices. Proc. 24th Nat. Conf. of the ACM, ACM Publ. P-69, 'Association for Computing Machinery', Brandon Systems Press, New Jersey, 157-172.

8. George, A., 1971, Computer implementation of the finite element method, STAN-CS-71-208, Computer Science Dept., Stanford Univ., Stanford, California.

9. Gibbs, N.E., Poole, W.G. and Stockmeyer, P.K., 1976, An algorithm for reducing the bandwidth and profile of a sparse matrix, SIAM J. Numer. Anal., Vol. 13, No. 2, 236-250.

10. Curtis, A.R. and Reid, J.K., 1971, The solution of large sparse unsymmetric systems of linear equations, J. Inst. Math. Appl., Vol. 8, 344-353.

11. Markowitz, H.M., 1957, The elimination form of the inverse and its application to linear programming, Management Science, Vol. 3, 255-269.

12. Reid, J.K., 1978, Private communication, June.

13. George, J.A. and McIntyre, D.R., 1976, On the application of the minimum degree algorithm to finite element systems, CS-76-16, University of Waterloo.

14. Duff, I.S., Munksgaard, N., Nielsen, C.W. and Reid, J.K., 1978, Direct solution of sets of linear equations whose matrix is sparse, symmetric and indefinite, Submitted to J. Inst. Maths. Applics.

15. George, J.A., 1973, Nested disscetion of a regular finite element mesh, SIAM J. Numer. Anal., Vol. 10, 345-363.

16. Allen, K., 1976, The direct solution of boundary value problems in partial differential equations, M.Sc. Thesis, Oxford.

17. Jacobs, D.A.H., 1977, Nested dissection - an improved ordering for finite difference and finite element methods, C.E.R.L. Note No. RD/L/N 108/77.

18. George, J.A., 1977, Solution of Linear Systems of Equations: Direct Methods for Finite Element Problems. In 'Sparse Matrix Techniques', edited by V.A. Barker, Springer-Verlag, Berlin, Heidelberg and New York.

19. Hockney, R.W., 1965, A fast direct solution of Poisson's equation using Fourier analysis, J. Assoc. Comp. Mach., Vol. 12, 95-113.

20. Hockney, R.W., 1970, The potential calculation and some applications, Meth. Comp. Phys., Vol. 9, 135-211.

21. Buzbee, B.L., Golub, G.H. and Nielson, C.W., 1970, On direct methods for solving Poisson's equations, SIAM J. Numer. Anal., Vol. 7, 627-656.

22. Cooley, J.W. and Tukey, J.W., 1965, An algorithm for the machine calculation of Fourier series, Maths. Comput., Vol. 19, 297-301.

23. Buneman, O., 1969, A compact non-iterative Poisson solver, Report 294, Stanford University For Plasma Physics, Stanford, California.

24. Sweet, R.A., 1974, A generalized cyclic reduction algorithm, SIAM J. Numer. Anal., Vol. 11, 506-520.

25. Swarztrauber, P.N. and Sweet, R.A., 1973, The direct solution of the discrete Poisson equation on a disk, SIAM J. Numer. Anal., Vol. 10, 900-907.

26. Martin, E.D., 1974, A generalized capacity matrix technique for computing aerodynamic flows, Computers and Fluids, Vol. 2, 79-97.

27. Swarztrauber, P.N., 1974, A direct method for the discrete solution of separable elliptic equations, SIAM J. Numer. Anal., Vol. 11, 1136-1150.

28. Concus, P. and Golub, G.H., 1973, Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations, SIAM J. Numer. Anal., Vol. 10, 1103-1120.

29. Martin, E.D., 1973, Direct numerical solution of three-dimensional equations containing elliptic operators, Int. J. Num. Meth. Eng., Vol. 6, 201-212.

30. Roache, P.J., 1971, A New Direct Method for the Discretized Poisson Equation, In 'Lecture Notes in Physics', Vol. 8, Edited by M. Holt, Springer-Verlag, New York.

31. Roache, P.J., 1978, Marching Methods For Elliptic Problems: Parts I, II and III, Num. Heat Transfer, Part I in Vol. 1, 1-25, Parts II and III to be published.

32. Lorentz, E.N., 1976, A rapid procedure for inverting del-square with certain computers, Monthly Weather Review, Vol. 104, 961-966.

33. Bank, R.E. and Rose, D.J., 1975, An $O(n^2)$ method for solving constant coefficient boundary value problems in two dimensions, SIAM J. Numer. Anal., Vol. 12, 529-550.

34. Bank, R.E. and Rose, D.J., 1977, Marching algorithms for elliptic boundary value problems. I: The constant coefficient case, SIAM J. Numer. Anal., Vol. 14, 792-829.

35. Bank, R.E., 1977, Marching algorithms for elliptic boundary value problems. II: The variable coefficient case, SIAM J. Numer. Anal., Vol. 14, 950-970.

36. Parter, S.V., 1959, On two line iterative methods for the Laplace and biharmonic difference equations, Num. Math. Vol, 240-252.

37. Young, D., 1954, Iterative methods for solving partial difference equations of elliptic type, Trans. Amer. Math. Soc., Vol. 76, 92-111.

38. Carré, B.A., 1961, The determination of the optimum accelerating factor for successive over-relaxation, The Computer Journal Vol. 4, 43-78.

39. Reid, J.K., 1966, A method for finding the optimum successive over-relaxation parameter, Computer Journal, Vol. 9, 200-204.

40. Kulsrud, H.E., 1961, A practical technique for the determination of the optimum relaxation factor of the successive over-relaxation method, Comm. ACM Vol. 4, 184-187.

41. Zadunaisky, P.E., 1976, On the estimation of errors propagated in the numerical integration of ordinary differential equations, Num. Math., Vol. 27, 21-39.

42. Stone, H.L., 1968, Iterative solution of implicit approximations of multi-dimensional partial differential equations, SIAM J. Numer. Anal., Vol. 5, 530-558.

43. Weinstein, H.G., Stone, H.L. and Kwan, T.V., 1969, Iterative procedure for solution of systems of parabolic and elliptic equations in three dimensions, Ind. and Eng. Chem. Fund., Vol. 8, 281-287.

44. Jacobs, D.A.H., 1974, The strongly implicit procedure for Biharmonic problems, J. Comp. Phys., Vol. 13, 303-315.

45. Reid, J.K., 1970, On the Method of Conjugate Gradients for the Solution of Large Sparse Systems of Equations. In 'Large Sparse Sets of Linear Equations', Edited by J.K. Reid, Academic Press, London.

46. Meijerink, J.A. and van der Vorst, H.A., 1977, An iterative solution method for systems of which the coefficient matrix is a symmetric M-matrix, Maths. Comp., Vol. 31, 148-162.

47. Kershaw, D.S., 1978, The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations, J. Comp. Phys., Vol. 26, 43-65.

48. Jacobs, D.A.H. and Pritchard, J., 1978, A preconditioned conjugate gradient method for elliptic difference equations, to be published.

49. Fox, L., 1947, Some improvements in the use of relaxation methods for the solution of ordinary and partial differential equations, Proc. Roy. Soc. A, Vol. 190, 31-59.

50. Jacobs, D.A.H., 1975, Iterative replacement schemes for complicated banded difference equations, C.E.R.L. Note No. RD/L/N 169/75.

AN ISOPARAMETRIC HERMITE FINITE ELEMENT PROGRAM FOR TWO DIMENSIONAL, NON-LINEAR, MAGNETOSTATIC FIELDS

Viggo Hoppe
Burmeister & Wain A/S, Engineering Division
2 Torvegade, DK 1449 Copenhagen K, Denmark

ABSTRACT

The Burmeister & Wain FEM-system B&W-FINEL is based on family of isoparametric, Hermitean finite elements presented in ref. 2. The reasons for the choice of these higher order elements is explained and their performance as compared with linear elements are discussed.

B&W-FINEL has been used since 1970 for the computation of solutions to the poisson differential equation e.g. instationary, non-linear heat conduction and compressible fluid flow. Also elastostatic problems have been solved in two and three dimensions and in axisymmetric structures with non-symmetric loads. In ref. 1 a solution of magnetostatic problems was announced and an actual result was presented at the symposium.

Since then some practical experiences have been gained with the program especially with regard to the non-linear behaviour.

The program handles magnetic fields caused by electric currents and polarised regions. Symmetry- and antimetry conditions may be applied, but open circuits with a high dispersion field requires the modelling of large parts of the space. A method for connecting up finite elements with regions with a series of exact solutions is presented at the MAFELAP 78 conference in April 1978 at Brunel University, Uxbridge. A post-processor computes magnetic forces by means of line integrals of the Maxwell stress tensor.

In the paper much attention is paid to the requirements which a material model has to fulfil to be consistent with the physical behaviour of the material and the numeric analysis of the program. Two models are proposed: one for reversible materials which has been implemented in the program and a new one developed through the last month before the conference. The latter handles the case of fully three-dimensional, anisotropic, non-linear fields with hysteresis. Examples of three-dimensional loops with non-parallel $\underline{B}$- and $\underline{H}$-vector are presented.

INTRODUCTION

The Burmeister & Wain finite element system B&W-FINEL is based on a family of isoparametric quadrilateral Hermitean elements in R_n presented in ref. 2. Compared with Hermitean elements as presented in standard text books the FINEL elements differ in the respect that the degrees of freedom are the function value and its derivatives up to order q in the four corners, whereas conventional elements contain some additional degrees of freedom which are of order up to 2q. The reasons for this difference becomes evident when isoparametric elements of arbitrary shape are called for as it is simply not possible to transform a m'th order derivative ($q < m \leq 2q$) to derivatives in another co-ordinate system without knowing the complete set of derivatives up to order m of the function as well as the isoparametric mapping.

Other attempts have been made to construct polynomials for this purpose but it may be shown (ref. 2) that there exists a unique family of interpolation polynomials based on a complete polynomial $\xi^i \eta^j \zeta^k$, $i + j + k \leq 2q + l = p$, plus a minimum of extra terms of degree $i + j + k \leq 2q + n$ namely those having at least two uneven exponents. The polynomials for 0' and 1' order represent the same terms as the so-called serendipity elements.

The interpolation polynomials of B&W-FINEL are expressed in the local curvilinear (ξ, η, ζ)-co-ordinates of the isoparametric mapping. The matrices are then subjected to a linear transformation so that degrees of freedom become expressed in global cartesian co-ordinates (x, y, z).

In points where all the joining elements are of the same material, the global (but not the local) derivatives are identical and by chosing these as degrees of freedom we have obtained 4 advantages:

1) The total number of unknowns in the system has been reduced without affecting the accuracy.

2) The field is almost continuous with continuous derivatives up to order q in the neighbourhood of the corner nodes between elements.

3) For second order problems (e.g. the electrostatic and magnetostatic) and $q \geq 2$ a number of unknowns may be eliminated by introducing the equilibrium equations (ref. 2) which further reduces the number of unknowns.

4) It is possible to construct an extension of this element family in which the order is attributed to the individual nodes rather than the elements. By this method which is not yet implemented in any program it will be possible to make a cheap low order trial run to find the interesting parts of the structure and then make a final

run where the accuracy is increased sufficiently in these areas. As the error is $O(h^{p+1})$ then an increase in p gives exponential convergence whereas a decrease of h only gives polynomial convergence. The simple term p- and h-convergence is often used.

In practically all electrotechnic finite element programs in literature linear elements are used. This is sufficient from a mathematical point of view as second order problems only require C_o-continuity. However, apart from the advantages mentioned above there are several other effects to be considered:

5) Few high-order elements are necessary for the modelling of a given structure i.e. saving in manpower.

6) Higher order isoparametric elements can model curved surfaces accurately. An error is introduced where curves are represented by broken lines (See ref. 6 , p 196).

7) Graphical representation becomes increasingly popular. Results obtained by higher order elements fit the geometry exactly, are more accurate, give smooth field lines and are therefore appealing to the eye and need less intellectual effort to be understood.

There is a minimum number of elements necessary to model a given structure. If the order q (and the degree p) is raised then the computing time raises drastically. But if the task is to solve a problem with an error not greater than a given small figure, then the high order elements will often be cheaper to use than the linear ones in terms of computing time.

The B&W-FINEL elasticity programs comprise the plane 2-dimensional cases, the axi-symmetric case with non-symmetric loads and the general 3-dimensional case. All using cubic Hermite elements. Furthermore, there exists a program which covers the plane and the fully axi-symmetric case with quintic elements which exploit the ideas expressed in point 3 and which give extremely accurate results. Fig. 1 shows the isostress lines in a strip with a semi-circular cut-out subjected to in-plane bending. To appreciate the accuracy of the result it should be considered that the curves correspond to lines through points with equal gradient of the field. With linear elements these curves cannot be drawn as the gradient is constant within each element.

B&W-FINEL furthermore covers axi-symmetric plus 2- and 3-dimensional transient non-linear heat conduction, plane and axi-symmetric compressible fluid flow and plane and axi-symmetric electrostatics and development of periodic linear electromagnetic and acoustic programs is in progress.

APPLICATION OF B&W-FINEL TO MAGNETOSTATICS

The program which is the subject of this paper works in plane and axi-symmetric geometry with cubic, isoparametric Hermite elements and non-linear material properties. In the program we have chosen to represent the $\underline{B}$-vector as the curl of a vector potential

$$\underline{A}_r = \{0,0,\frac{A}{2\pi r}\} \qquad \underline{A}_t = \{0,0,\frac{A}{t}\}$$

$$\begin{Bmatrix} B_r \\ B_z \end{Bmatrix} = \begin{Bmatrix} \frac{1}{2\pi r} \cdot \frac{\partial A}{\partial z} \\ \frac{-1}{2\pi r} \cdot \frac{\partial A}{\partial r} \end{Bmatrix} \quad (1a) \qquad \begin{Bmatrix} B_x \\ B_y \end{Bmatrix} = \begin{Bmatrix} \frac{1}{t} \cdot \frac{\partial A}{\partial y} \\ \frac{-1}{t} \cdot \frac{\partial A}{\partial x} \end{Bmatrix} \quad (1b)$$

for axi-symmetric and plane geometry respectively.

Much attention has been paid to the modelling of the material properties. It was found that the model proposed in ref. 1 for reversible materials led to a zero saturation magnetisation and as Newton-iteration is more effective with a magnetisation curve which is smooth (has a continuous derivative) piecewise linear models were also ruled out and a continuous model has been suggested which is now included in the program. Quite recently a general three-dimensional model including hysteresis has been developed but it has not yet been implemented in the program.

Any solution of a non-linear P.D.E. must be based on successive iterations and linearisations, i.e. on a relationship of the form:

$$\underline{H}=\nu_o\cdot(\underline{B}-\underline{J})=\nu_o\underline{\underline{\nu}}(\underline{B})\cdot\underline{B} \quad (2a); \qquad \underline{B}=\mu_o\cdot(\underline{H}+\underline{M})=\mu_o\underline{\underline{\mu}}(\underline{H})\cdot\underline{H} \quad (2b);$$

in which single underlining denotes a vector, double underlining a matrix or a tensor. Thus $\underline{\underline{\nu}}(\underline{B})$ and $\underline{\underline{\mu}}(\underline{H})$ are tensors with dimensionless elements and $\nu_o=1/\mu_o$; $\underline{\underline{\nu}} = \underline{\underline{\mu}}^{-1}$ and $\underline{J} = \mu_o \underline{M}$. We do not assume isotropy and parallel $\underline{B}$ and $\underline{H}$ vectors.

We wish to minimise the energy functional:

$$\pi = \int_\Omega \left(\int_o^{\underline{B}} \underline{H}^T \cdot d\underline{B} \quad - \underline{B}^T \cdot \underline{M}_o - \underline{A}^T \cdot \underline{i} \right) d\Omega = \min. \qquad (3)$$

the internal integral is eliminated by the linearisation

$$\pi = \int_\Omega \left(\tfrac{1}{2}\, \nu_o \cdot \underline{B}^T \cdot \underline{\underline{\nu}}(\underline{B}) \cdot \underline{B} - \underline{B}^T \cdot \underline{M}_o - \underline{A}^T \cdot \underline{i} \right) d\Omega = \min. \qquad (4)$$

and the process is repeated until the difference between successive iterations $A_{i+1}-A_i$ is less than $10^{-7} \cdot (A_{max}-A_{min})$ in any one node and A_{max} (A_{min}) is the largest (smallest) A-value in the entire structure.

At interfaces between dissimilar materials two possibilities exist: Either to use the relations $H_{s+} = H_{s-}$ and $B_{n+} = B_{n-}$ where n and s are directions normal and tangential to the interface or to disconnect the derivatives of $\underline{A}$ in the normal direction at nodes on the interface. The results shown have been obtained with the first procedure but the latter has advantages when Newton-methods and hysteresis are involved.

At r = o in axi-symmetric problems $A = \frac{\partial A}{\partial r} = \frac{\partial A}{\partial z} = 0$ and $B = \begin{Bmatrix} 0 \\ \frac{-1}{2\pi} \cdot \frac{\partial^2 A}{\partial r^2} \end{Bmatrix}$

At external boundaries and planes with antimetry A = 0, and at planes of symmetry the natural boundary condition satisfies $\frac{\partial A}{\partial n} = 0$ in a least squares sense. A principle has been developed in ref. 4 for connecting domains with exact solutions containing singularities and domains extending towards infinity to the finite elements, but so far this has only been used to connect axi-symmetric structures with non-symmetric loads to general 3-dimensional structures in elasticity.

The program accepts magnetisation $\underline{M_0}$ and current density $\underline{i}$ which are constant within elements. Line- and point-loads like magnetic double layers, current sheets, poles, etc. may be included, but we have preferred to leave them out as they lead to singularities.

Computation of magnetic forces can be made by means of line integration along element boundaries of the Maxwell stress tensor. As there is a small discontinuity in H_n normal to the interface, slight differences can also be observed in the force if computed on the basis of the elements exterior or interior to the path of integration. This can be used as an estimate of the accuracy of the result.

There are two key-problems in achieving efficient computations: material models and iteration strategies, and the rest of the paper will be devoted to these subjects and some computational results:

MAGNETIC MATERIAL MODELS

Any model should comply with the following rules:

1. The model should be point symmetric.
2. The model must exhibit a monotonous behaviour with saturation.
3. The saturated material must be isotropic even though the unsaturated material is not.

For materials with hysteresis there are two additional rules:

4. Laws of thermodynamics imply that all loops absorb energy, (and turn it into heat)
5. Internal small loops close themselves and are forgotten (ref. 7).

For isotropic reversible materials $\underline{B}$ and $\underline{H}$ are parallel and the tensor $\underline{\underline{\mu}}(\underline{H})$ reduces to a scalar $\mu(H)$, $H = |\underline{H}|$. In the program we have preferred to express the relative susceptibility $\chi = \mu - 1$:

$$\underline{M} = \chi(H) \cdot \underline{H}$$

For reasons of computational efficiency we have chosen an expression of the form

$$M = \sum_{i=1}^{N} M_i \cdot \rho\left(\frac{H}{H_i}, \kappa_i\right) \qquad (6)$$

in which $H_i = M_i/\chi_i$, M_i is the saturation magnetisation for the i'th component, χ_i its susceptibility for H = 0 and $0 \leq \kappa_i \leq 1$ is a dimensionless shape parameter. With $\xi = \frac{H}{H_i}$ we have chosen

$$\rho(\xi, \kappa) = \frac{1}{2\kappa}\left[\sqrt{(\xi+\kappa)^2 + 1 - \kappa^2} - \sqrt{(\xi-\kappa)^2 + 1 - \kappa^2}\right] = \frac{2\xi}{\sqrt{(\xi+\kappa)^2 + 1 - \kappa^2} + \sqrt{(\xi-\kappa)^2 + 1 - \kappa^2}} \qquad (7)$$

This is a smoothed ramp function, which satisfies

$\rho(\xi, \kappa) = \pm 1$ for $\xi \to \pm\infty$; $\rho(\xi, \kappa) = -\rho(-\xi, \kappa)$

$\rho(0, \kappa) = 0$; $\frac{\partial}{\partial \xi}(\rho(0,\kappa)) = 1$

$\kappa = 0$ gives $\rho(\xi, 0) = \frac{\xi}{\sqrt{\xi^2 + 1}}$

$\kappa = 1$ gives the broken line $\rho(\xi, 1) = \begin{cases} -1 & \text{for } \xi \leq -1 \\ \xi & \text{for } -1 \leq \xi \leq 1 \\ 1 & \text{for } \xi \geq 1 \end{cases}$

Even though $\frac{B}{\mu_o H} = \mu_s$ is a scalar, the matrix $\left\{\frac{\partial B}{\mu_o \partial H}\right\}$

is not a scalar but a tensor which must be used for Newton iteration:

$$d(H) = \frac{d(\sqrt{H_x^2 + H_y^2 + H_z^2})}{dH} \cdot d\underline{H} = \frac{H_x \cdot dH_x + H_y \cdot dH_y + H_z \cdot dH_z}{H} = \frac{\underline{H} \cdot d\underline{H}}{H} =$$

$$\alpha dH_x + \beta dH_y + \gamma dH_z \qquad (8)$$

in which $\alpha = \frac{H_x}{H}$, etc. are the directional cosines of the H vector. Differentiation of $\underline{B} = \mu_o \mu_s (H) \cdot \underline{H}$ gives $d\underline{B} = \mu_o \cdot [d\mu_s \cdot \underline{H} + \mu_s \, d\underline{H}]$

$$dB = \mu_o \begin{Bmatrix} \mu_s + (\mu_t - \mu_s)\alpha^2 & (\mu_t - \mu_s)\alpha\beta & (\mu_t - \mu_s)\alpha\gamma \\ & \mu_s + (\mu_t - \mu_s)\beta^2 & (\mu_t - \mu_s)\beta\gamma \\ \text{symmetric} & & \mu_s + (\mu_t - \mu_s)\gamma^2 \end{Bmatrix} \cdot d\underline{H} = \underline{D} \cdot d\underline{H} \qquad (9)$$

$\mu_s = \frac{B}{\mu_o H}$ may be termed the relative secant permeability and

$\mu_t = \frac{1}{\mu_o} \cdot \frac{dB}{dH}$ the relative tangent permeability.

This is consistent with the reciprocal relation

$$d\underline{H} = \nu_o \begin{Bmatrix} \nu_s + (\nu_t - \nu_s)\alpha^2 & (\nu_t - \nu_s)\alpha\beta & (\nu_t - \nu_s)\alpha\gamma \\ & \nu_s + (\nu_t - \nu_s)\beta^2 & (\nu_t - \nu_s\ \beta\gamma \\ \text{symmetric} & & \nu_s + (\nu_t - \nu_s)\gamma^2 \end{Bmatrix} d\underline{B} = \underline{D}^{-1} \cdot d\underline{B} \qquad (10)$$

in which $\nu_s = \frac{1}{\mu_s}$ and $\nu_t = \frac{1}{\mu_t}$

Next we consider a material which is anisotropic because it consists of lamination with airgaps in between. The base material is isotropic and the proportion of airgap is δ. The saturation magnetisation M_s is reduced by the factor $1-\delta$. With $\chi_s = \mu_s - 1$ we can write the relative permeability parallel to the laminations.

$$\mu_p = \chi_p + 1 = (1 - \delta)\mu_s + \delta; \quad \chi_p = (1 - \delta)\chi_s \qquad (11)$$

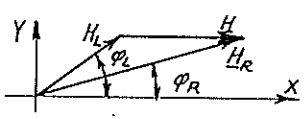
and normal to the laminations

$$\nu_n = (1 - \delta)\nu_s + \delta; \qquad \chi_n = \frac{(1 - \delta)}{1 + \delta\chi_s} \chi_s \qquad (12)$$

Now $\chi_s \cong \frac{M_s}{H} \to 0$ for $H \to \infty$, when the saturation magnetisation M_s is approached. We see that $\chi_n/\chi_p \to 1$. These are therefore the formulas to use with anisotropic material.

In case of crystalline anisotropy a similar behaviour would be expected.

At the previous conference the question of a fully 3-dimensional material model exhibiting hysteresis was touched upon and found as yet unsolved. In the last month before this conference I decided to investigate if it were possible to construct a model consistent with the rules quoted earlier (p.) on the basis of the classical concept of a magnetic material as a matrix of small magnetic spheres which initially were randomly orientated and which subsequently were aligned towards the same direction under the influence of an external field H. The assumptions made was firstly that each magnet could move independently and that each of them was kept in position by an individual strictly local field H_L. For an isotropic material H_L is constant and unformly distributed in all directions in space and we can now calculate the magnetisation curve for such a material analytically. The spheres orientate in the direction of the resultant $\underline{H}_R$ of the local field $\underline{H}_L$ and the external H-field. The magnetic moment within the interval ϕ_L | $\phi_L + d\phi_L$ which covers the solid angle $2\pi \cdot \sin\phi_L \cdot d\phi_L$ is:



$$M_s \cdot \cos\phi_R \cdot \frac{2\pi}{4\pi} \cdot \sin\phi_L \cdot d\phi_L \qquad (13)$$

If we introduce $\xi = \frac{H}{H_L}$ we can write

$$\cos\phi_R = \frac{\cos\phi_L + \xi}{\sqrt{(\cos\phi_L + \xi)^2 + \sin^2\phi_L}} \qquad (14)$$

The magnetic moment integrated over all solid angles is

$$M = \frac{M_s}{2} \int_0^{\pi} \frac{(\cos\phi_L + \xi)\sin\phi_L}{\sqrt{(\cos\phi_L + \xi)^2 + \sin^2\phi_L}} \cdot d\phi_L$$

$$= \frac{M_s}{2} \int_{-1}^{1} \frac{(t + \xi)\,dt}{\sqrt{(t + \xi)^2 + 1 - t^2}} = M_s \cdot \begin{cases} \frac{2}{3}\xi & \text{for } |\xi| \leq 1 \\ \pm\left(1 - \frac{1}{3\xi^2}\right) & \text{for } |\xi| \geq 1 \end{cases} \qquad (15)$$

The asymptotic behaviour of the previous model is consistent with this expression.

The model (which probably is not new) is continuous in its concept but leads to a magnetisation curve with a discontinuous second derivative. Non-isotropic distributions of H_L and M_S are conceivable.

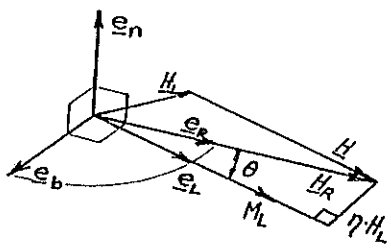
If a Coulomb-type friction moment $\eta \cdot H_L \cdot M_L$ independent of $\underline{H}_R$ is introduced the component of $\underline{H}_R$ at right angle to the direction of the magnetisation $\underline{M}_L$ of a sphere should be greater than $\eta \cdot H_L$ to be able to turn it. Any rotation will take place in the common plane of $\underline{M}_L$ and $\underline{H}_R$ until the friction and the moment balances. An analytic integration over the solid angles is no longer possible but a numerical integration can be used and in ref. 5 several formulas for integration over the surface of a unit sphere can be found. The integration points are placed as the vertices of regular polyhedra plus certain intermediate points.

To make the model reproduce real hysteresis loops it was necessary to assume that in the same direction in space there are a number of magnets which have friction factors which were different and in the results presented in the figures I have chosen 12 integration points with 5 friction levels $\frac{1}{5}\eta$, $\frac{2}{5}\eta$, $\frac{3}{5}\eta$, $\frac{4}{5}\eta$ and η and an equal proportion of the magnets at each level, so that the magnetisation at each level is $M_S/60$.

The procedure loops over the directions in space and compute the resultant field $\underline{H}_R$. Then it loops over the friction levels: Let $\underline{e}_L$ be a unit vector in the direction of the magnetisation vector $\underline{M}_L$ ($= \underline{e}_L \cdot M_S/60$) left over from the previous iteration. Then $\underline{S}_n = \underline{e}_L \times \underline{H}_R$ is a vector normal to the plane common to $\underline{M}_L$ and $\underline{H}_R$, its numeric value S_n is the moment acting on a unit magnet and $\underline{e}_n = \underline{S}_n/S_n$ is a unit normal vector to the plane. If S_n is less than ηH_L the magnet does not move, otherwise it moves towards $\underline{H}_R$ until $S_n = \eta H_L$. To do this we find a unit vector $\underline{e}_R = \underline{H}_R/H_R$ and form $\underline{e}_b = \underline{e}_R \times \underline{e}_n$. The three vectors are orthonormal and the new $\underline{e}_L$ is found as

$$\underline{e}_L = \underline{e}_b \cdot s + \underline{e}_R \cdot \sqrt{1-s^2} \qquad (16)$$

in which $s = \sin\theta = \eta H_L/H_R$.



The results obtained with the model are in good accordance with the behaviour of actual magnetic materials and without making any special effort to do so the loops close and in the demagnetisation example on fig. 6 the end points of each loop follow the virgin curve quite closely.

This suggests that in spite that the model neglects the existence of the Weiss-domains and other known physical phenomena pertinent to the magnetic behaviour it contains some physical truth.

Non-isotropic materials can be modeled by assuming that H_L is a function of the direction in space, and also M_S and ηH_L may vary between the individual integration points.

A number of computer plots show the performance of the model; Fig. 2 to 6 have $\underline{B}$ and $\underline{H}$ parallel.

Fig. 2 shows the magnetisation curve for a reversible material obtained with 12 integration points. It is very close to the theoretical value as given by (15).

Fig. 3 shows the magnetisation loop obtained with a single level of friction. The model gives internal loops which are lines parallel to the H-axis. It also exhibits a phenomenon equivalent with Barkhausen jumps when a magnet which is locked in a position almost parallel to the field but orientated in the opposite direction suddenly is torn loose and flops over to be nearly parallel to $\underline{H}$. This phenomenon cannot be avoided but it is diminished when the number of integration points and friction levels are increased.

Fig. 4 shows a loop drawn with 5 friction levels. The virgin curve is now in accordance with experiments, but for very intense H fields the jumps are still appreciable.

Fig. 5 shows a number of loops inside each other and it seems in good accordance with fig. 1 of ref. 7.

Fig. 6 shows a magnetisation followed by a series of demagnetizing loops with decreasing amplitude of H.

Fig. 7 is a sequence in which H_x is increased from 0 to $2H_L$. Thereafter the $\underline{H}$-vector is rotated around the z-axis. $\eta = 1.0$ has been chosen. The $\underline{M}$-vector is app. 19° behind the $\underline{H}$-vector. M_x versus H_x is plotted.

Fig. 8 is a similar sequence in which $\underline{H}$ not only rotates but is also decreased at the same time.

Fig. 9 is a sequence in which the starting $\underline{H} = (2,0,0)$. $\underline{H}$ is then rotated around the direction (1,1,1).

One other important question is computing time spent on the hysteresis model related to the total computing time and it is obvious that a model which requires the storage of 3x5x12=180 quantities for each integration point in the structure may be costly.

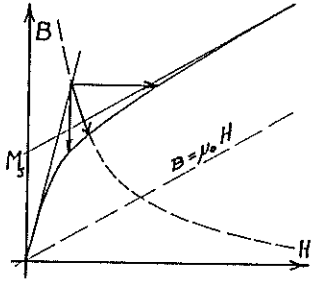
In case of 2-dimensional and axi-symmetric problems there is a symmetry around the XY-plane in the hysteresis model and therefore only half the integration points are necessary. Numerical experi-

ments show that 8 integration points and 5 friction levels give acceptable results whereas a large number of integration points do not give reasonable results with only 1 or 2 friction levels. Often only a small region has to be treated as non-linear and the linear regions may be eliminated from the computations by means of static condensation which is now a well known procedure within finite element analysis. A typical job with a complexity as fig. 11 may contain 140 elements of which only 4-8 need to be treated non-linearly. Even though each of these uses 10 times as much computing time as the rest, the problem is still managable as the computing time for the linear case is only 2.2 SUP-min. on a UNIVAC 1106.

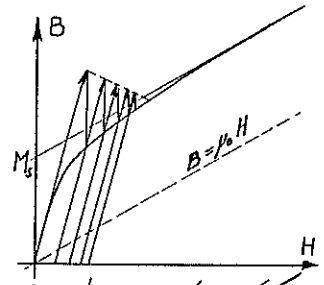
ITERATION STRATEGIES

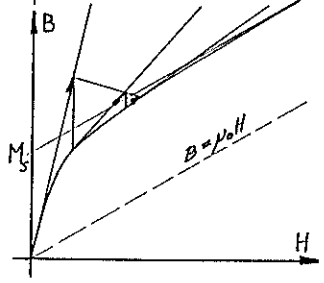
The above statements naturally lead to considerations of the optimal strategy in computations and we may immediately divide the problems in two classes which are either reversible (conservative) or non-reversible (with hysteresis, non-conservative). In the first case the result is independent of the sequence in which the fields are changed but in the latter this is not so, therefore the optimal strategy in the first case is the one which gives the correct answer in as few steps as possible, whereas when hysteresis is involved the minimum number of steps is determined also by the sequence of changes of the field.

In the reversible case the mathematical principle behind the convergence is that of a contraction mapping and the smaller the Lipschitz-constant the fewer iterative steps are needed. However, the time spent on each step may be considerably reduced if the same matrix could be used several times. Newton-iteration is more complicated as it requires a new matrix each time, but it is very efficient as the number of digits corrected is doubled from step to step. The iteration strategies for the reversible case may be summarised as follows:



1. When a point $(\underline{B},\underline{H})$ which is not on the magnetization curve is found, we must extrapolate to a corresponding point on the curve in which we compute the permeability for the next iteration. For this we may move along lines with either B or H or B·H constant.





2. As explained above, we need not update the left hand side for each iteration step. The matrix need only be inverted after each updating.

3. We may base our iterations on the conventional (secant) or the differential (tangent) permeability. The latter which is identical to Newton iteration is shown in the diagram.

The experiences gained so far are not conclusive but the indications are:

ad 1.
Extrapolation along H-constant should always be used. In case of an infinitely long homogeneous bar in a long solenoid this is the exact solution. Extrapolation along B-constant is nearly always divergent.

ad 2.
This method is sometimes divergent if only the absolute values and not the vector components of $(\underline{B},\underline{H})$ are carried over to the next iteration. It may be possible to optimize the number of iterations between updating the matrix.

ad 3.
For $\underline{B}=\mathrm{curl}(A)$ proper Newton iterations imply extrapolations with B=constant. However, H=constant is a better guess from the physical point of view. The non-linearity always implies anisotropy because the permeability perpendicular to the field vector is unaffected by the field strength.

From (10) we can deduce

$$d\underline{H} \simeq \underline{H}-\underline{H}_i = \underline{\underline{D}}^{-1}\cdot d\underline{B} \simeq \underline{\underline{D}}^{-1}\cdot(\underline{B}-\underline{B}_i);\ \underline{H} \simeq \underline{\underline{D}}^{-1}\cdot(\underline{B}-\underline{B}_i) + \underline{H}_i \qquad (17)$$

where $(\underline{H}_i,\underline{B}_i)$ are the field vectors from the previous step (4) now can be written:

$$\Pi_{i+1} = \int_\Omega \left(\tfrac{1}{2}\ \nu_0\cdot\underline{B}^T\cdot\underline{\underline{D}}^{-1}\cdot\underline{B}-\underline{B}^T\cdot(\underline{M}_0-\underline{H}_i+\underline{\underline{D}}^{-1}\cdot\underline{B}_i)-\underline{A}^T\cdot\underline{i}\right) d\Omega \qquad (18)$$

which is the version of the Newton iteration scheme included in the program.

For the irreversible case with hysteresis no experience with regard to computational strategy is as yet available but the methods developed for plasticity which exhibits several seemingly similar features can probably be used.

CONCLUSION

The use of higher order elements in magnetostatics can give economic, accurate and appealing results as demonstrated by the results presented in fig. 10.a,b,c, 11.a,b and 12. The problems of representing the material properties correctly seem solved but some simplifications in the model could be desired.

REFERENCES

Ref. 1 Viggo Hoppe: "A Versatile field program based on the Finite Element Method". Int. High-voltage Symp. Zürich, 1975.

Ref. 2 Viggo Hoppe: "Parametric Hermitean Elements with Reduced sets of Unknowns", "The mathematics of Finite Elements and Applications" MAFELAP 75, proc. Brunel Univ. conf. of the I.M.A. April 1975, J.R. Whiteman ed., Acad. Press 1977.

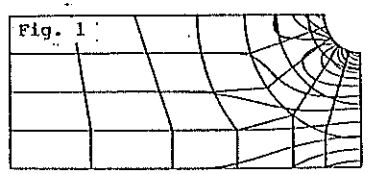
Ref. 3 R. Glowinski - A. Marrocco - Analyse numérique du champ magnétique d'un alternateur par éléments finis et surrelaxation ponctuelle non linéaire. Computer methods in applied mechanics and engineering, 3, 1, (1974).

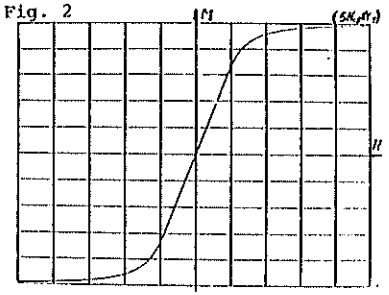
Ref. 4 Hans H. Sørensen & Viggo Hoppe: "Interface Problems Between Finite-Element Regions With Dissimilar Function Spaces". The math. of F.E. and app. MAFELAP 78, proc. Brunel Univ. conf. April 1978.

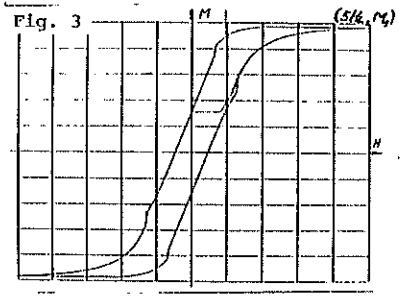
Ref. 5 A.H. Stroud: Approximate Calculation of Multiple Integrals Prentice Hall, 1971

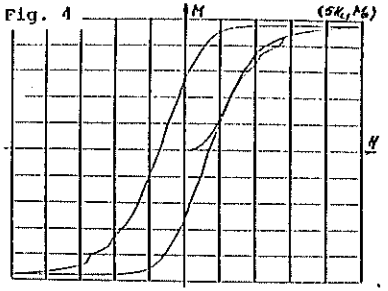
Ref. 6 Strang & Fix: An analysis of the Finite Element Method. Prentice Hall 1973.

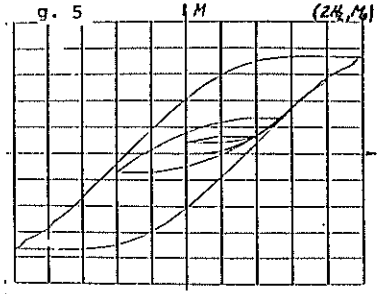
Ref. 7 N. Janssen: Mathematical Modelling of Magnetic Hysteresis. Proc. COMPUMAG conf. Oxford 1976.

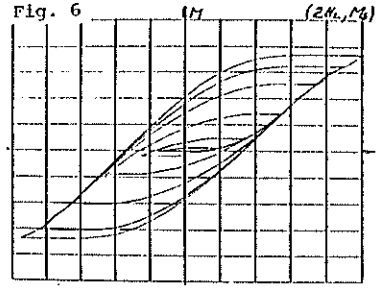


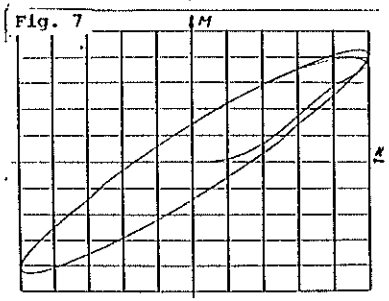


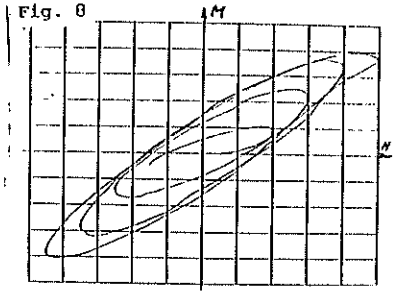


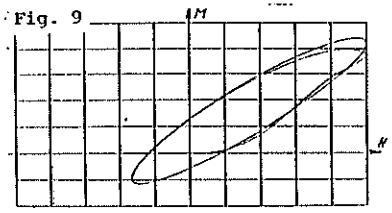












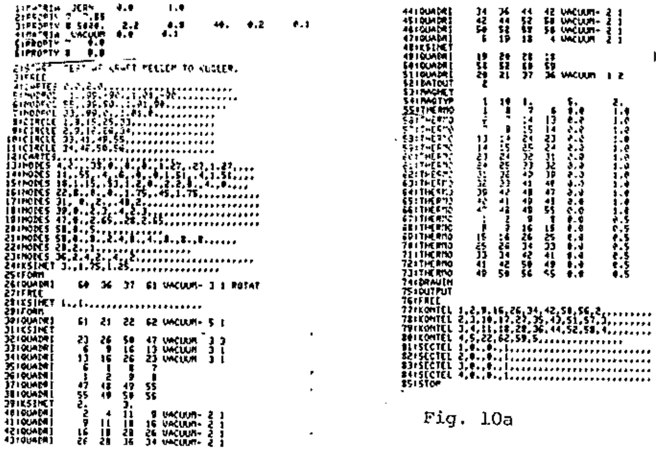


Fig. 10a

Sample problem of two identical spheres magnetized parallel to their common centerline. Fig. 10a shows the complete input to the B&W-FINEL program. The four KONTEL cards represent four integration paths for the computation of magnetic forces. These coincide reasonably well with eachother and the theoretical value.

Fig. 10b

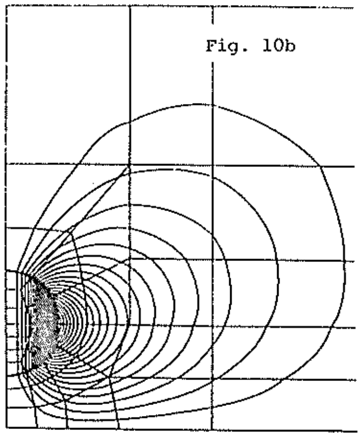
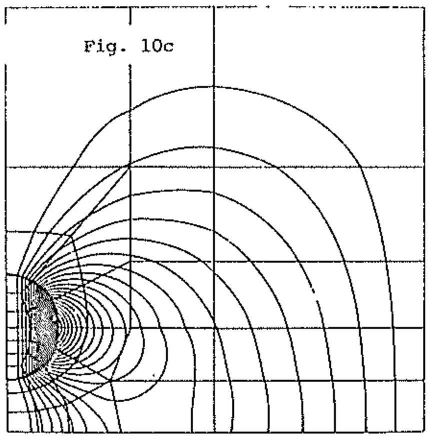


Fig. 10c



In fig. 10b the magnetization is opposed in the spheres (case of antimetry).

In fig. 10c the magnetization is in the same direction (case of symmetry). All drawings are computer-plots produced on a Tektronix 4014-1.

Fig. 11a

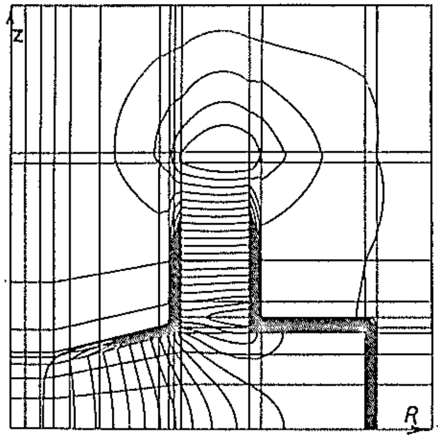


Fig. 11b

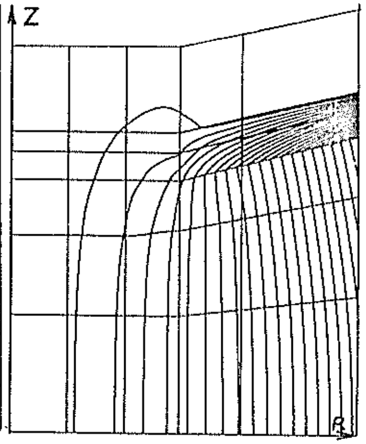


Fig. 11a and b. Magnetic field in a magnetron. The field is created by two radially magnetized ring-shaped permanent magnets. The central field has been drawn in more detail to the right.

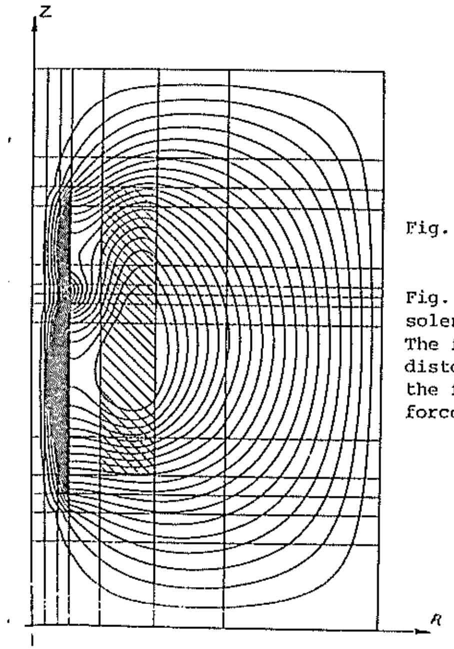


Fig. 12

Fig. 12. Magnetic actuator with solenoid and and open ciruit. The fact that the far field is distorted at the boundaries of the finite region affects the force computation very little.

A CLASS OF METHODS FOR SOLVING TWO-DIMENSIONAL LINEAR AND NON-LINEAR STEADY STATE AND TRANSIENT PROBLEMS

A G Jack
C A Parsons & Co Ltd.
Heaton Works
Newcastle upon Tyne

R L Stoll
Department of Electrical Engineering
The University
Southampton

ABSTRACT

The paper describes a class of methods for solving two-dimensional steady-state or transient eddy-current problems. The technique employed is to treat the problem as an initial boundary value problem the solution at a particular time being obtained by time stepping to the desired point. The method of time stepping is explicit and is based upon the Dufort-Frankel algorithm. The performance of the original Dufort-Frankel algorithm for linear problems is tested against models capable of analytical solution. Following this the performance of the nonlinear modification of the method in one dimensional form is examined making use of an established one dimensional nonlinear solution. A method of dealing with nonconducting zones in two-dimensional models is detailed. This allows the use of numerical/analytical hybrids which are advantageous in terms of computation time in certain circumstances. Lastly an example of the use of the method for a representative two-dimensional problem is described.

1. INTRODUCTION

Many practical devices have regions in which significant eddy-currents are produced. Such eddy-currents may or may not be desireable but a capability to predict them in the design phase is essential. The presence of ferromagnetic material is a further complication making the representation of nonlinearity important. Examples will often be large, geometrically complicated and contain many different materials.

In this paper a numerical method is described which can deal with such problems as may be adequately modelled using two spatial dimensions.

2. A BASIC APPROACH TO THE PROBLEM

Two-dimensional eddy-current problems may be usefully described in terms of the magnetic vector potential. We may be interested in either transient or steady-state solutions. Both may be gained by a solution of

$$\frac{\partial}{\partial x}\left(\nu\frac{\partial A_z}{\partial x}\right)+\frac{\partial}{\partial y}\left(\nu\frac{\partial A_z}{\partial y}\right)=J_z+\sigma\frac{\partial A_z}{\partial t}\quad,\qquad(1)$$

where ν, the reluctivity, is a function of the curl of the vector potential. Steady-state problems can be solved by treatment as an initial boundary value problem, the steady-state solution arising after the decay of an initial transient. A suitable method of solution for either transient or steady-state problems is to follow the solution in a step-by-step manner through time. This is the type of method described here. Such a method is capable of dealing with saturation and the harmonics it produces without drastic assumptions.

To set up a time stepping process the time differential term of Equation 1 may be replaced by a first order difference approximation. The choice of this difference replacement has implications for the resulting time stepping method. If, for instance, a backward difference approximation is used an implicit method results, and upon substitution of a numerical approximation for the left hand side of Equation 1 a set of simultaneous equations has to be solved at each time step[1]. If, on the other hand, a forward difference approximation is used an explicit method results and a set of independent linear equations must be solved at each time step. Intuitively we might expect that an explicit method will take less time per time step to compute but need more time steps than an implicit method to achieve results of the same accuracy.

Which of these methods will be the better approach? A useful indication is given by one dimensional problems in which both approaches have been used. For instance, Stoll (1974)[2] page 86 gives the information that from numerical tests on one dimensional problems an implicit method in the form of the Crank-Nicolson scheme is marginally faster (in computation time) than an explicit method in the form of the method known as Hopscotch[3]. In a one dimensional problem the matrix resulting from the 'replacement' of the spatial derivative is tridiagonal. In the two-dimensional case the matrix is banded, the size of the band being dependent on the shape of the mesh. This is of no consequence for an explicit method. However an implicit method requires computation time which is strongly related to this band width. As the number of nodes increases the band width also increases and an implicit method suffers increasingly in comparison with an explicit method. In conclusion, it appears that an explicit method will be advantageous for two-dimensional problems and increasingly so as the number of nodes increases.

3. DESCRIPTION OF THE METHOD

For one dimensional problems it has been demonstrated that the Hopscotch method is efficient[4]. For two-dimensional problems the advantages of the Hopscotch method over the closely related Dufort-Frankel algorithm are mostly lost and in any event the special splitting of the mesh required for the Hopscotch method is not necessarily possible for a finite element mesh. For a linear region of single permeability the Dufort-Frankel replacement of Equation 1 may be written in finite difference form as

$$A_{ij}^{k+1} = A_{ij}^{k-1} + \frac{2r}{4r+1}\left(A_{i+1j}^{k} + A_{i-1j}^{k} + A_{ij+1}^{k} + A_{ij-1}^{k} - 4A_{ij}^{k-1}\right) \quad (2)$$

where ij refers to a nodal position in x,y,t given by x=ih, y=jh, t=kΔt, and $r = \Delta t/\sigma\mu h^2$. The exclusion of the node A_{ij}^{k} from the algorithm is to avoid instability. Two-dimensional nonlinear vector potential versions of the Dufort-Frankel algorithm implicitly involve A_{ij}^{k} via the calculation of the reluctivity. A further disadvantage of a two-dimensional formulation over a one dimensional formulation in terms of the magnetic field strength is that the nonlinear parameter in the one dimensional case $(\partial B/\partial H)$ is a function of the magnitude of the independent variable (the magnetic field strength) whereas in two-dimensional problems the nonlinear parameter (the reluctivity ν) is a function of the slope of the potential (i.e. on curl A). These two disadvantages necessitate the use of reluctivity damping, where the reluctivity at any time step k is based upon previous time step reluctivities by

$$\nu^{k} = \beta(\text{function } (\text{curl } A^{k})) + (1 - \beta)\,\nu^{k-1} \quad (3)$$

In this equation β is the 'damping factor' and values of β in the range 0.02 and 0.35 have been found to produce stable results of acceptable accuracy over a large range of excitations given suitable values for Δt and h. The accuracy of the method described above (and its stability for nonlinear problems) is dependent upon the choice of space and time step. A linear one dimensional example, for which an analytical solution is available, provides a suitable vehicle for establishing suitable values for space and time steps (i.e. Δy and Δt) for linear problems. It has been found that such information provides valuable insight into the choice of these parameters for nonlinear problems. The model used is that of a thick solid conducting block with a specified sinusoidal tangential magnetic field strength at the surface and an arbitrary boundary of A=0 many skin depths from the surface. Figure 1 shows a plot of lines of constant maximum error in the current density occurring at any depth within the block. From this it can be seen that the choice of a large space step, a large time step, or a large ratio of time to space step all result in large errors. The last mentioned virtually obscures errors due to too large a time step. The space step limit is in line with that found by Carpenter[5] for finite element analysis of fundamental harmonic problems. It is simple to show using the analytical solution that for this problem to minimise the errors we require

$$1 \gg \frac{\omega\,\Delta t}{\sqrt{2}\;\Delta y/\delta} \qquad (\delta = \text{skin depth}, \ \omega = \text{circular frequency}) \quad (4)$$

$$1 \gg \frac{\omega\,\Delta t}{\sqrt{6}} \quad (5)$$

$$1 \gg \frac{\Delta y}{\sqrt{6}\,\delta} \quad (6)$$

From this and Figure 1 two ideas can be drawn:- (i) the space step, time step and ratio of the two must be chosen in relation to the solution and (ii) a more accurate and yet quicker to compute solution can be gained by judicious choice of space and time step.

Extending the problem to allow saturation of the block makes necessary the use of another numerical solution for test purposes An available established method is that due to Lim[3]. In a nonlinear problem harmonics are produced by saturation and the analysis of the effect of space and time steps becomes far more complex. Another factor is the effect of the damping factor β. Simple rules are much harder to define as Figures 2 and 3, showing differences between a single Lim type solution and the described method using various space and time steps, demonstrate. However in these figures the ground rules established in the linear study apply in that large space steps result in large errors (showing first, perhaps not surprisingly, in the harmonics); large time steps are not particularly harmful, unless accompanied by small space steps, in which case instability results. The relationship between space and time steps can be traced back to the replacement of the node A_i^k in the spatial derivative by $(A_i^{k-1} + A_i^{k+1})\frac{1}{2}$ (to avoid instability). Nonlinearity reinstroduces that node via the calculation of the reluctivity hence the presence of instability and the inducement of it by using large ratios of time to space step. Experiments have shown that a good choice for the space step is equal to the unsaturated skin depth and for the time step about 1/300th of the periodic time. An example of the comparison between a one dimensional solution posed in vector potential form is described here and Lims method of solution which uses the magnetic field strength is given in Figure 4. The comparison is good and yet the vector potential solution is twelve times

faster than the Lim solution because much larger space and time steps are possible.

A linear two-dimensional example is that of a conducting block subjected to a tangential magnetic field strength in the form of a travelling wave on its surface. Using the analytical solution for this problem it can be shown that for small errors, conditions can be placed upon the two space steps, the time step and the ratio of time to space steps as follows:-

for Δx, (where x is tangential to the surface of the block)

$$1 \gg \frac{\pi}{\sqrt{12}} \frac{\Delta x}{g} \qquad (g = \text{pole pitch of the travelling wave}) \qquad (7)$$

for Δy, if $g \gg \frac{\pi \delta}{\sqrt{2}}$

$$1 \gg \frac{\Delta y}{\sqrt{6}\,\delta} \qquad (8)$$

or if $g \ll \frac{\pi \delta}{\sqrt{2}}$

$$1 \gg \frac{\pi}{\sqrt{12}} \frac{\Delta y}{g} \qquad (9)$$

for Δt,

$$1 \gg \frac{\omega \Delta t}{\sqrt{6}} \qquad (10)$$

and for the ratios $\Delta t/\Delta x$ and $\Delta t/\Delta y$

$$1 \gg \frac{\omega \Delta t}{\sqrt{2}\ \Delta y/\delta} + \frac{\omega \Delta t}{\sqrt{2}\ \Delta x/\delta} \qquad (11)$$

Again the behaviour of the solution is seen to dictate the required value of the space and time steps in the same way as the one dimensional solution.

Experience with two dimensional nonlinear models indicates that the results of the one dimensional experiments hold for two dimensional problems with the possible exception that two dimensional problems tend to be more stable.

If near to optimal space and time steps are used the method is

quite attractive in terms of computational time and it is viable to time step 6000 nodes through 3 or 4 cycles on moderately fast computers. This seems to be in excess of the capability of currently available implicit time stepping programs.

4. METHODS FOR ZONES IN WHICH THE CONDUCTIVITY IS ZERO

Two-dimensional problems often have zones in which the conductivity is zero, such as air gaps, laminated cores, etc. For areas such as this the equation being solved is either Laplaces or Poissons equation. There seems to be no reason why the Dufort-Frankel difference equation should not be used for this area. However using the Fourier method it can be shown that the method is marginally stable/unstable and numerical tests have shown that with certain boundary conditions instability results. We can get over this by exploiting the fact that the method is explicit. The areas in which the conductivity is non zero require only values at the time steps k and k-1 to compute k+1 potentials. From the areas where the conductivity is zero only values at time k are required. Once all the nonconducting areas have been moved forward to k+1, a solution at k+1 can be sought in the conducting areas using the previously calculated values of potential on the boundary between the regions as a Dirichlet boundary. Having gained this solution by some means (e.g. SOR) it can be used along with k+1 and k values in the conducting areas to gain a solution at k+2 in those areas and so on.

This technique is not limited to the Dufort-Frankel method, any explicit method is open to the same technique. One important point to note is that at each time step a complete solution for the nonconducting areas is not necessary; only a boundary condition for the conducting areas is required. Any method for solving a boundary value problem can be used for the Laplacian or Poissonian regions and in many practical cases an analytical solution can be used. The use of an analytic magnetostatic solution combined with the fact that only a boundary condition is required can result in a very considerable saving in computation time. An example of a case where an analytical solution is appropriate is the air gap of an electrical machine (where the stator boundary is considered smooth).

5. A FINITE ELEMENT VERSION OF THE METHOD

Equation 2 defined the Dufort-Frankel algorithm for a linear single permeability region represented by a finite difference mesh with equal mesh squares of side h. It is a simple matter to extend that algorithm to nonlinear multiple permeability regions repres-

ented by an uneven rectangular finite difference mesh. The finite element method can also be used but the classically derived equation using a linear variation of potential in each element requires some modification. This classically derived equation can be written

$$[S].[A] + [M].\frac{\partial}{\partial t}[A] + [F] = 0 \qquad (12)$$

To make an explicit method possible the structure of (M), which is such that off-diagonal terms exist, must be altered so that only diagonal terms are non zero. To accomplish this a technique known as 'lumping' is applied in which the induced current in each element is split evenly between the three vertex nodes. Also the equivalent term in (S) corresponding to the contribution from the node A^{k}_{ij} in the finite difference method must be applied to the equivalent in the finite element method of $(A^{k+1}_{ij} + A^{k-1}_{ij})/2$. This is necessary to achieve an equivalent form of the Dufort-Frankel algorithm rather than an equivalent of Richardsons (unstable) method. This is achieved by splitting (S) into two matrices, (T) with only off-diagonal terms and (U) with only diagonal terms. The finite element algorithm now becomes

$$[T][A]^{k} + \tfrac{1}{2}[U]\left\{[A]^{k+1} + [A]^{k-1}\right\} + [Mmod]\left\{[A]^{k+1} - [A]^{k-1}\right\}\frac{1}{2\Delta t} + [F] = 0 \qquad (13)$$

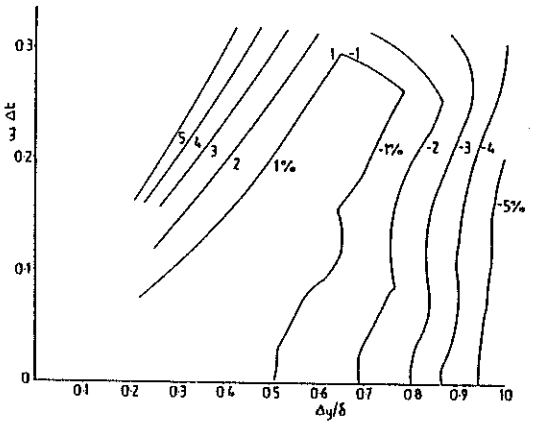
Numerical tests[7] have shown the algorithm to behave similarly to the finite difference method on the same problems.

6. A PRACTICAL EXAMPLE

Turbogenerators rotors are built from large solid ferromagnetic forgings. The field winding is restrained within slots in the forging by conducting wedges. During certain abnormal operating conditions eddy currents are induced in the various components of the rotor. Calculation of these currents is an important part of the design of the machine. The method described here has been applied to a mathematical model of such a machine during unbalanced stator current operating conditions. A flux distribution resulting from that is shown in Figure 5. The model contained 4369 nodes in the rotor and in the air gap an analytical solution was used with 50 spatial harmonics. Following switch on of the stator excitation the periodic solution was dominant after about 3 cycles, to reach which 1200 time steps were used. To talk of computational time is difficult since it depends so much upon the individual machine used, suffice to say the solution, although very large, represents a quite moderate amount of execution time.

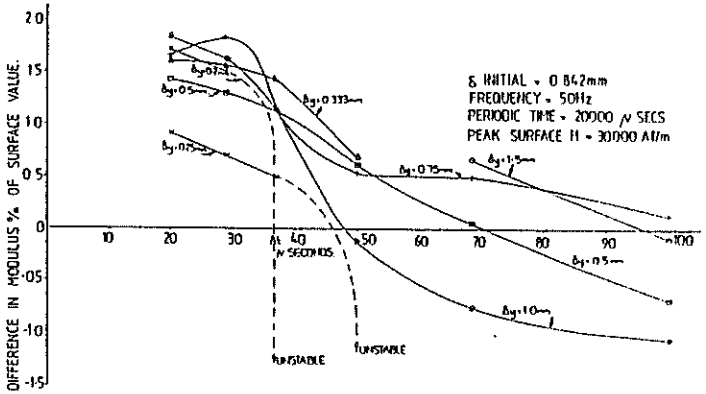
7. CONCLUSIONS

A class of methods has been defined in which finite element or finite difference methods can be used to model the spatial magnetic vector potential variation in conducting areas and any suitable numerical or analytical method can be used in the nonconducting areas. Large problems are capable of viable solution; problems with over 6000 nodes have been treated successfully. The solutions gained make no drastic approximation about saturation or harmonics.



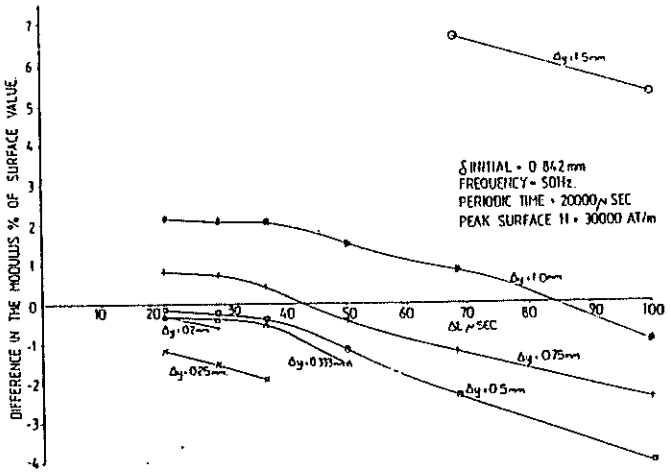
PERCENTAGE ERROR IN THE MODULUS OF THE CURRENT DENSITY WITH RESPECT TO THE SURFACE VALUE.

FIG.1



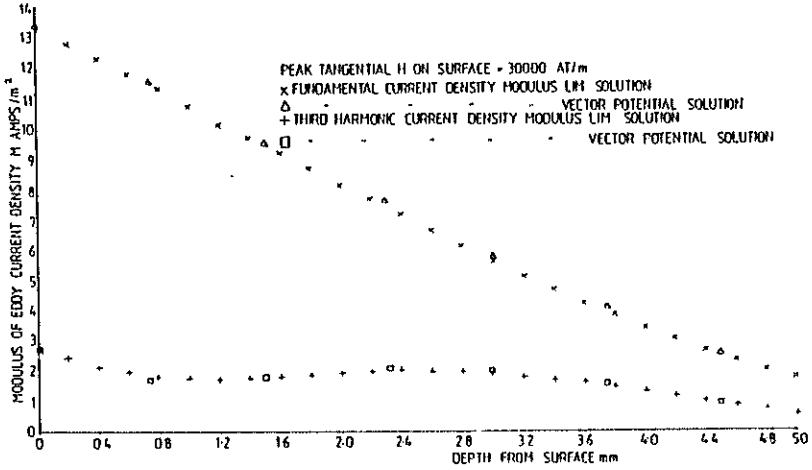
DIFFERENCE BETWEEN THE VECTOR POTENTIAL & LIM SOLUTIONS IN THE MODULUS OF THE FIRST HARMONIC OF THE CURRENT DENSITY ON THE SURFACE

FIG.2



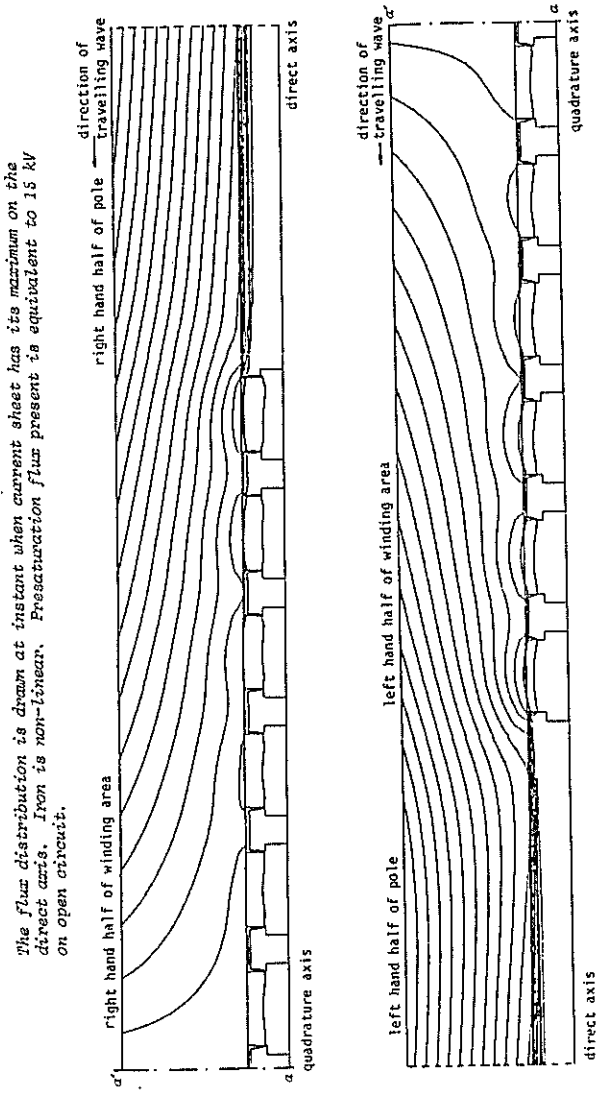
DIFFERENCE BETWEEN THE VECTOR POTENTIAL & LIM SOLUTIONS IN THE MODULUS OF THE THIRD HARMONIC OF THE CURRENT DENSITY ON THE SURFACE.

FIG. 3



COMPARISON OF VECTOR POTENTIAL SOLUTION & LIM SOLUTION OF THE SAME ONE DIMENSIONAL NONLINEAR PROBLEM

FIG.4



The flux distribution is drawn at instant when current sheet has its maximum on the direct axis. Iron is non-linear. Presaturation flux present is equivalent to 15 kV on open circuit.

FLUX LINES AT AN INSTANT OF TIME IN THE ROTOR SURFACE FOR 50% NEGATIVE SEQUENCE CURRENT

FIG. 5.

1. ALDEFELD B (1978) Electromagnetic field diffusion in ferromagnetic materials. Proc. Inst. Elect. Engrs. 125, 278-82

2. STOLL, R L (1974) The analysis of eddy currents, Oxford University Press, London

3. GOURLAY, A R (1970) Hopscotch: a fast second-order partial differential equation solver. J. Inst. Math. Appl. 6, 375-90.

4. YU WAI MAN, Y K L (1975) Small scale mathematical and experimental modelling of eddy currents in the solid rotor of an electrical machine. PhD Thesis, Southampton University.

5. CARPENTER, C J (1975) Finite element network models and their application to eddy-current problems, PROC IEE, Vol. 122 No. 4

6. LIM, K K and HAMMOND P (1972) Numerical method for determining the electromagnetic field in saturated steel plates. Proc. Instn. Elect. Engrs. 119, 1667-74.

7. JACK, A G (1977) A two-dimensional study of some eddy-current problems occuring in large electrical machine rotors PhD Thesis, Southampton University 1977.

SUR LA RESOLUTION DE SYSTEMES LINEAIRES, PROVENANT DE L'UTILISATION DE LA METHODE DES ELEMENTS FINIS, PAR LA METHODE SEMI-DIRECTE DE STONE

Hubert Froidevaux
Ecole Polytechnique Fédérale, Lausanne

RESUME

On applique une méthode de Stone modifiée à la résolution de systèmes linéaires, provenant par exemple, de la discrétisation par éléments finis de problèmes aux limites de type elliptique. On énonce des conditions de convergence du processus itératif. On présente des expériences numériques.

1. INTRODUCTION

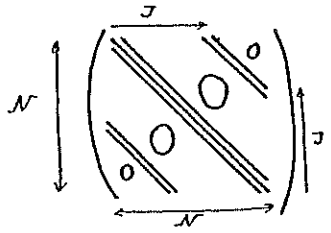
H.L. Stone, dans un article de 1968 (SIAM J. Numer. Anal. Vol. 5, No 3, septembre 1968) a introduit une méthode élégante de résolution de systèmes linéaires provenant de la discrétisation par différences finies de problèmes aux limites et il la compare à d'autres méthodes classiques. En s'inspirant de son idée on introduit toute une classe de méthodes de résolution.
Soit le système linéaire

$$A\vec{x} = \vec{f} \qquad (1)$$

que l'on veut résoudre. On suppose que la matrice A est symétrique définie positive, c'est-à-dire qu'elle satisfait la condition $(A\vec{x},\vec{x})>0$ quelque soit $\vec{x} \neq \vec{0}$. Dans ce cas une méthode de solution de (1) est donnée par Choleski. La matrice A se factorise en $A = \tilde{L}\tilde{L}^T$, $\tilde{L}$ étant une matrice triangulaire inférieure. Alors le système (1) devient

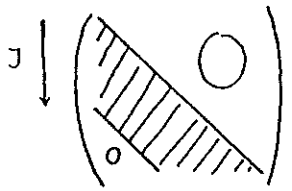
$$\tilde{L}\tilde{L}^T\vec{x} = \vec{f} \qquad (2)$$

On obtient facilement sa solution en deux étapes. On résout $\tilde{L}\vec{y} = \vec{f}$ puis $\tilde{L}^T\vec{x} = \vec{y}$. Si A est une matrice creuse de la forme



(3)

alors $\tilde{L}$ ne sera pas creuse.



(4)

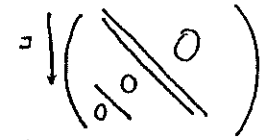
Ce fait présente le grave inconvénient d'utiliser beaucoup de place. Alors que le stockage de A demande environ 4N places, celui de L en demande environ JxN.

L'idée de Stone est d'utiliser une méthode itérative conservant les zéros de A. On décompose A en une somme

$$A = LL^T + B \qquad (5)$$

Avec L de la forme

$$L = \qquad (6)$$



Alors (1)devient

$$LL^T\vec{x} + B\vec{x} = \vec{f} \qquad (7)$$

système que l'on resout par itérations

$$LL^T\vec{x}_{n+1} = \vec{f} - B\vec{x}_n \qquad (8)$$

On introduit un facteur de relaxation comme suit :

$x_{n+\frac{1}{2}}$ est calculé par

$$LL^T\vec{x}_{n+\frac{1}{2}} = \vec{f} - B\vec{x}_n \qquad (9)$$

et $\vec{x}_n$ est calculé par

$$\vec{x}_{n+1} = \vec{x}_n + \alpha[\vec{x}_{n+\frac{1}{2}} - \vec{x}_n] \qquad (10)$$

α nombre positif, est le facteur de relaxation.

En éliminant $\vec{x}_{n+\frac{1}{2}}$ entre (9) et (10), on obtient :

$$LL^T\vec{x}_{n+1} = LL^T\vec{x}_n - \alpha[A\vec{x}_n - f] \qquad (11)$$

2. CONVERGENCE DE PROCEDE ITERATIF (11)

On s'intéresse évidemment à la convergence du procédé itératif (11) vers la solution de (1) quelque soit le vecteur de départ $\vec{x}_0$.

Proposition 1

Si A est symétrique définie positive, $\alpha>0$ et si la matrice $2A^{-1}LL^T-\alpha I$ est définie positive, alors le procédé itératif (11) converge quelque soit $\vec{x}_0$.

Démonstration

(11) s'écrit $\vec{x}_{n+1} = [I-\alpha(LL^T)^{-1}A]\vec{x}_n + \alpha(LL^T)^{-1}f$

I étant la matrice identité.

On sait que la condition de convergence est que $\|I-\alpha(LL^T)^{-1}A\| < 1$ pour une norme matricielle quelconque.

On introduit une norme matricielle particulière définie par le produit scalaire

$[x,y] = (Ax,y)$ et $\|x\| = \sqrt{[x,x]}$

$(.,.)$: produit scalaire habituel

Calculons, en posant $M = (LL^T)^{-1}$,

$\|(I-\alpha MA)\vec{x}\|^2 = \|\vec{x}\|^2 + \alpha^2\|MA\vec{x}\|^2 - 2\alpha[MAx,x]$

d'où

$\|(I-\alpha MA)\vec{x}\|^2 - \|\vec{x}\|^2 = \alpha(\alpha\|MA\vec{x}\|^2 - 2[MAx,x])$

$= \alpha[MA\vec{x},\alpha MA\vec{x}-2I\vec{x}] = \alpha[\vec{y},\alpha I\vec{y}-2A^{-1}LL^Ty]$ (avec $y = MA\vec{x}$)

Le dernier terme est négatif quelque soit $y \neq 0$, donc quelque soit $\vec{x} \neq 0$,
d'où

$\|(I-\alpha MA)x\|^2 < \|\vec{x}\|^2$ quelque soit $\vec{x}$, ce qui permet de conclure.

Examinons la signification de la condition $2A^{-1}LL^T-\alpha I$ définie positive

$[(2A^{-1}LL^T - \alpha I)\vec{x},\vec{x}]>0$

$[2A^{-1}LL^T\vec{x},\vec{x}]>\alpha[\vec{x},\vec{x}]$

et en vertu de la définition de $[\cdot,\cdot]$ on a

$(2LL^T\vec{x},\vec{x})>\alpha(A\vec{x},x)$

d'où l'on tire une condition nécessaire pour la convergence du processus (11)

$$\frac{(LL^T\vec{x},\vec{x})}{(A\vec{x},\vec{x})} > \frac{\alpha}{2} \qquad \text{quelque soit } \vec{x} \neq 0 \qquad (12)$$

On peut aussi traduire la relation (12) en terme de valeurs propres des matrices A et LL^T . Ces deux matrices étant définies positives ont toutes leurs valeurs propres positives et on a les relations

$$\underline{\lambda}_A\|\vec{x}\|^2 \leq (A\vec{x},\vec{x}) \leq \overline{\lambda}_A\|\vec{x}\|^2$$
$$\underline{\lambda}\,\|\vec{x}\|^2 \leq (LL^T\vec{x},\vec{x}) \leq \overline{\lambda}\,\|\vec{x}\|^2 \qquad (13)$$

$\underline{\lambda}_A(\underline{\lambda})$: plus petite valeur propre de $A(LL^T)$

$\overline{\lambda}_A(\overline{\lambda})$: plus grande valeur propre de $A(LL^T)$

(12) et (13) donnent une relation suffisante pour assurer la convergence du processus itératif (11).

$$\underline{\lambda} > \frac{\alpha}{2}\,\overline{\lambda}_A\,;\ \alpha>0 \qquad (14)$$

Cette relation montre que pour un système linéaire (1) donné, ($\overline{\lambda}_A$ est fixé) , il est toujours possible de trouver une décomposition LL^T et un α tels que les itérations convergentes. Théoriquement, du moins, LL^T peut être quelconque.

L'erreur à l'itération n est donnée par la différence

$$\vec{r}_n = \vec{x}_n - \vec{x} \qquad (15)$$

où $\vec{x}$ est la solution exacte du système (1). Pour le procédé itératif (11) on obtient facilement la relation

$$\vec{r}_n = [I-(LL^T)^{-1}A]^n\vec{r}_0 \qquad (16)$$

Un autre vecteur intéressant est

$$\vec{R}_n = A\vec{x}_n - \vec{f} \qquad (17)$$

Il est relié à $\vec{r}_n$ par

$$\vec{R}_n = A\vec{r}_n \qquad (18)$$

et on a

$$\vec{R}_n = A[I-\alpha(LL^T)^{-1}A]^n A^{-1}\vec{R}_0 \qquad (19)$$

et pour une norme quelconque

$$\|\vec{R}_n\| = \|A[I-\alpha(LL^T)^{-1}A]A^{-1}\|^n \|\vec{R}_0\| = C^n\|\vec{R}_0\| \qquad (20)$$

Alors le nombre

$$C = \|I-\alpha(LL^T)^{-1}A\| \qquad (21)$$

mesure la qualité du processus itératif. On sait déjà que pour avoir convergence, il faut que $C<1$. La vitesse de convergence augmente lorsque C diminue. C est lié au rayon spectral γ de la matrice $I-\alpha(LL^T)^{-1}A$.

$$\gamma = \rho(I-\alpha(LL^T)^{-1}A) \qquad (22)$$

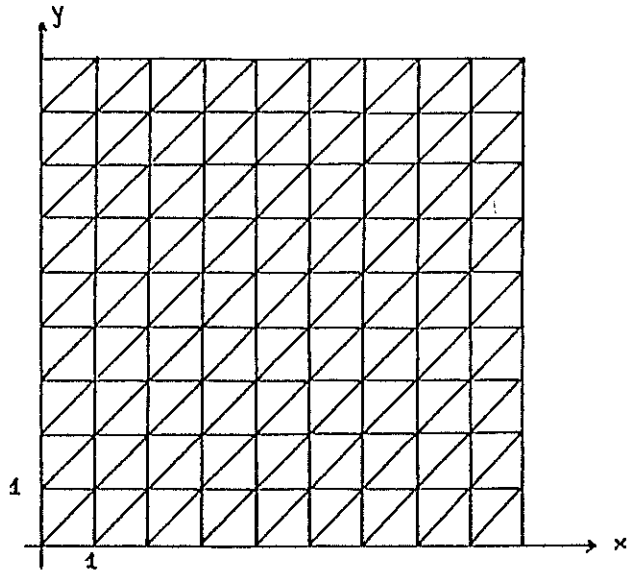
3. CHOIX DE LA MATRICE A

La matrice A doit être définie positive et avoir la structure indiquée en (3). Le plus simple pour réaliser ces deux conditions est de discrétiser, par éléments finis triangulaires du premier ordre, le problème (elliptique) aux limites

$$\begin{aligned} &\mathrm{div}(a(x,y)\,\mathrm{grad}\,) = 0 \quad \text{dans} \quad \Omega \\ &u\big|_{\partial\Omega} = g \\ &a(x,y)>0 \quad \text{dans} \quad \Omega \end{aligned} \qquad (23)$$

Pour obtenir la structure creuse de (3) il faut trianguler et numéroter convenablement les noeuds.

Le domaine $\Omega(0)$ est le carré de la figure suivante :



Le domaine $\Omega(a)$ est l'image de $\Omega(0)$ par la transformation $T(a)$:

$$\Omega(0) \qquad\qquad \Omega(a)$$

$$\begin{pmatrix} x \\ y \end{pmatrix} \xrightarrow{T(a)} \begin{pmatrix} x+ay \\ y \end{pmatrix}$$

et tous les domaines ont la même triangulation à la transformation $T(a)$ près.

Les différents problèmes que l'on résout sont :

P1 : $\begin{cases} \Delta u = 0 & \text{dans } \Omega(0) \\ u = x+y & \text{sur } \partial\Omega(0) \end{cases}$

P2 : $\begin{cases} \Delta u = 0 & \text{dans } \Omega(3) \\ u = x+y & \text{sur } \partial\Omega(3) \end{cases}$

P3 : $\begin{cases} \Delta u = 0 & \text{dans } \Omega(5) \\ u = x+y & \text{sur } \partial\Omega(5) \end{cases}$

P4 : $\begin{cases} \operatorname{div}(r^2 \operatorname{grad} u) = 0 & \text{dans } \Omega(5) \\ u = x+y & \text{sur } \partial\Omega(5) \end{cases}$

P5 : $\begin{cases} \operatorname{div}(r^4 \operatorname{grad} u) = 0 & \text{dans } \Omega(5) \\ u = x+y & \text{sur } \partial\Omega(5) \end{cases}$

$r^2 = x^2+y^2$

4. CHOIX DE LL^T

A étant donné, il faut choisir LL^T de telle sorte que la condition (12) ou bien (14) soit satisfaite et que C (21) soit le plus petit possible. Ces deux conditions sont pratiquement impossible à vérifier à priori, c'est pourquoi on fait des décompositions $A = LL^T+B$ de manière heuristique. Le choix est toutefois guidé par la

Remarque 1 : Si LL^T est exactement égal à A, le processus itératif (11) donne la solution en une itération. On a donc avantage à choisir LL^T "proche de A" et facilement inversible.

Pour préciser, introduisions quelques notations. La matrice A est décrite par les vecteurs a(N), b(N-1), c(N-J+2), d(N-J+1) et la matrice L par les vecteurs r(N), s(N-1), t(N-J+1).

On veut construire LL^T satisfaisant à la remarque 1. Il semble naturel d'identifier les diagonales et les surdiagonales de A et LL^T respectivement. L'algorithme de calcul est :

$$
\begin{aligned}
r(1) &= [a(1)]^{\frac{1}{2}} \\
s(1) &= b(1)/r(1) \\
t(1) &= [uc(i)+(1-u)d(i)]/r(1)
\end{aligned}
$$

$$
\left.\begin{aligned}
r(i) &= [a(i)-(s(i-1))^2]^{\frac{1}{2}} \\
s(i) &= b(i)/r(i) \\
t(i) &= [uc(i)+(1-u)d(i)]/r(i)
\end{aligned}\right\} \quad 2 \leq i \leq J-1
$$

$$
\left.\begin{aligned}
r(i) &= [a(i)-(s(i-1))^2-(t(i-J+1))^2]^{\frac{1}{2}} \\
s(i) &= b(i)/r(i) \\
t(i) &= [uc(i)+(1-u)d(i)]/r(i)
\end{aligned}\right\} \quad J \leq i \leq N-J+1
$$

$$
\left.\begin{aligned}
r(i) &= [a(i)-(s(i-1)) -(t(i-J+1))^2]^{\frac{1}{2}} \\
s(i) &= b(i)/r(i)
\end{aligned}\right\} \quad N-J+2 \leq i \leq N-1
$$

$$
r(N) = [a(N)-(s(N-1))^2-(t(N-J+1))^2]^{\frac{1}{2}}
$$

Dans l'algorithme on introduit un paramètre u qui permet plusieurs décompositions de A. On note L_u la matrice obtenue pour un u fixé.

5. ESSAIS NUMERIQUES

On itère

$$
\left.\begin{aligned}
L_u L_u \vec{x}_{n+1} &= L_u L_u \vec{x}_n - \alpha\left[A\vec{x}_n - \vec{f}\right] \\
\vec{x}_0 &= 0
\end{aligned}\right\} \qquad (24)
$$

Le processus dépend des deux paramètres α, u. Les propriétés de convergences dépendent de la matrice A donc dans notre cas du problème (P1, P2, P3, P4, P5) et de u et α. La vitesse de convergence est définie par

$$
v = \log C \qquad (25)
$$

Cette définition provient de (20) qui indique que

$$
\log\|\vec{R}_n\| = n \log C + \log\|\vec{R}_0\| \qquad (26)
$$

La vitesse de convergence caractérise la convergence vers zéro de la suite $\{\|\vec{R}_n\|\}$. La figure (27) montre que (26) est effectivement juste. Pour cela on définit les deux normes vectorielles.

$$
\|\vec{x}\|_1 = \|(x_1, x_2, \ldots, x_N)\|_1 = \max_{1 \leq i \leq N} \{|x_i|\} \qquad (28)
$$

$$
\|\vec{x}\|_2 = \|(x_1, x_2, \ldots, x_N)\|_1 = \left[\sum_{i=1}^{N} (x_i)^2\right]^{\frac{1}{2}} \qquad (29)
$$

La figure (30) montre les domaines du plan (α,u) où il y a convergence de (24). Ces images montrent la dépendance de la convergence en P1, P2, P3, P4, P5.

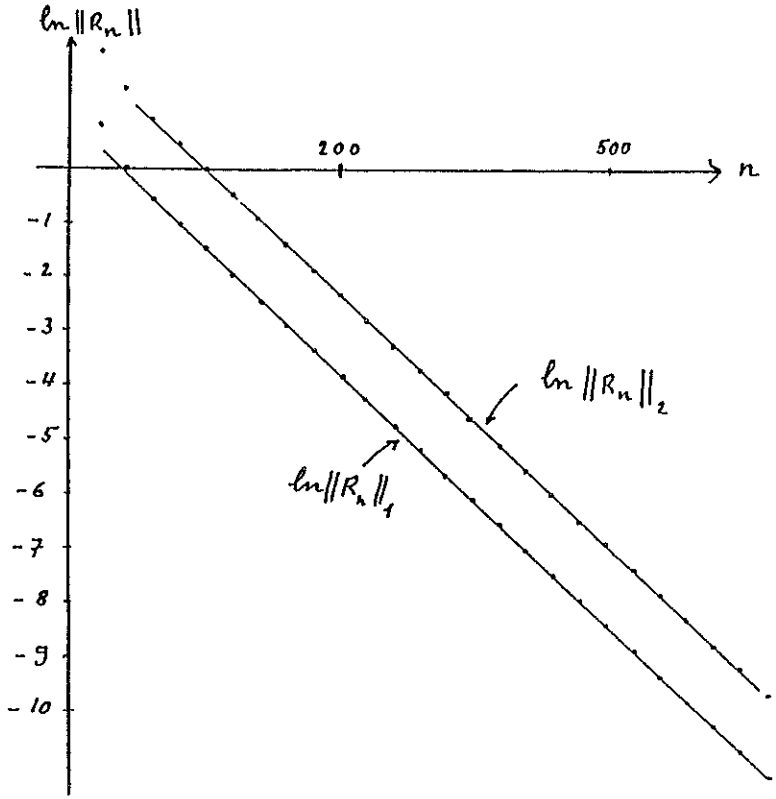


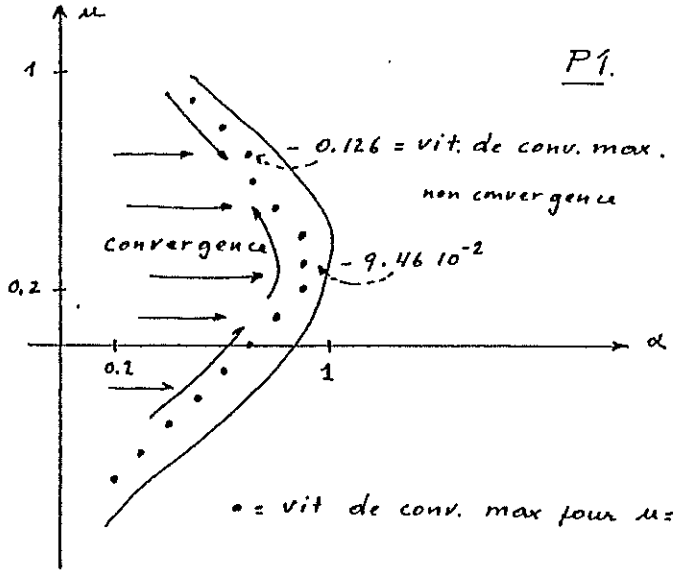
Figure 27

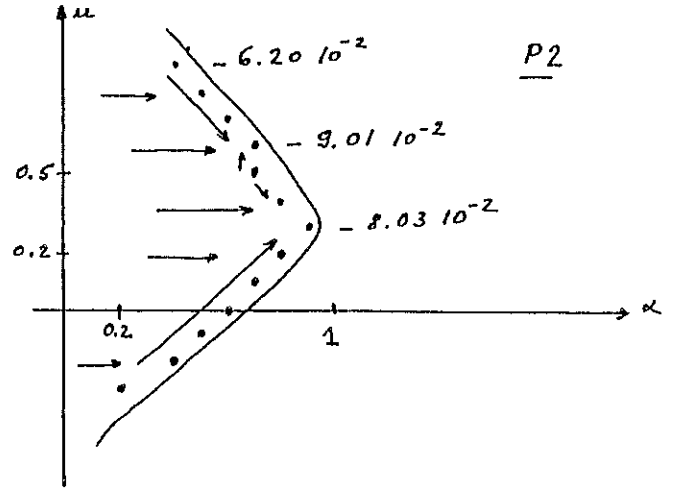
Courbe $\ln\|R_n\| = n \ln c + \ln\|R_0\|$

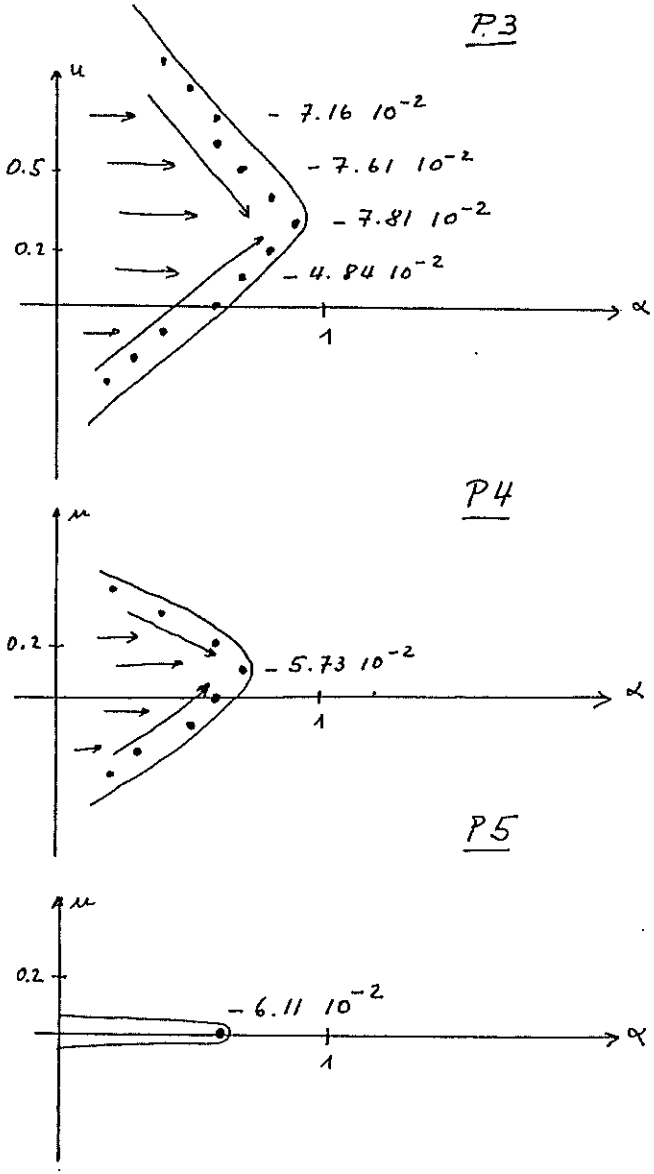
Problème P1 avec $u = 0$, $\alpha = 0.3$

la vitesse de convergence est mesurée par $\ln c$

Figure 30 : Domaines de convergence pour les problèmes P1, P2, P3, P4, P5
les flèches indiquent la direction de croissance de la vitesse de convergence.







6. REMARQUES ET CONCLUSION

1. Les valeurs des paramètres (α,u), pour lesquelles la vitesse de convergence est la plus grande, se situent au bord du domaine de convergence.

2. La vitesse de convergence dépend, comme on pouvait s'y attendre, de la matrice A, donc du problème à résoudre. Cependant dans le cas P5, elle est encore très convenable.

3. La méthode se programme très simplement et elle économise (c'est son but) la mémoire.

4. On a tenté d'améliorer la vitesse de convergence en modifiant légèrement la matrice L , par l'introduction d'un troisième paramètre multipliant la diagonale de L. Les résultats ne sont pas intéressants.

5. Il me semble qu'uneamélioration possible pourrait être obtenue. Mon projet est de construire la matrice L en introduisant plusieurs paramètres u . Soit {u} cet ensemble de paramètres, $L_{\{u\}}$ la matrice L correspondante. La matrice $L_{\{u\}}$ pourrait satisfaire

$$\| L_{\{u\}} L^T_{\{u\}} - A \| = \text{minimum sur } \{u\}$$

pour une norme matricielle.

6. La méthode peut évidemment s'appliquer à des matrices quelconques en effectuant une décomposition A=LU+B .

7. On peut aussi appliquer les mêmes idées à d'autres types de matrices A.

MAGGY2 AND PADDY, PROGRAM PACKAGES FOR 2 AND 3 DIMENSIONAL MAGNETOSTATIC PROBLEMS

S.J. Polak, A. Wachters
A. de Beer, J.van Welij

ABSTRACT

In this paper the program packages MAGGY2 and PADDY are discussed. Special attention is paid to the mesh and geometry definition and the linear algebra.

I. INTRODUCTION

A great deal of experience has been obtained with algorithms for 2 and 3 dimensional magnetostatic problems [1] [2]. The essential difficulties in such problems now are

- complicated configurations
- high accuracies.

In an industrial environment the second plays a minor role whereas the first is of major importance. Because our background is industrial we mostly pay attention to difficulties from problems with complicated configurations. Such problems involve long

- data preparation and
- computing

times.
For obvious reasons the relative importance of the first is increasing w.r.t. the second. One way to make the data preparation easier is the construction of a program-package with a problem oriented language, POL, see e.g. [3] and [4].
This problem oriented language should allow the description of the problem in a terminology close to the usual technical notation.
The program packages MAGGY2 and PADDY are constructed for resp. 2 and 3 dimensional magnetostatic problems. MAGGY2 has MAGLAN as a POL and PADDY has PARDEL. Both offer similar problem description facilities.
In this paper we pay attention to the mesh and geometry specifications in the problem oriented languages and to the linear algebra in these packages. We will do this against the background of both packages as a whole. However a separate paper on MAGGY2 has been submitted such that we concentrate here on the aspects in PADDY. For the sake of comparison we do treat the MAGGY2 features as well.
In section 2 we give some information on our ideas w.r.t. program packages. In section 3 problem oriented languages are discussed and in section 4 the mesh and geometry specifications are considered. In section 5 the algorithms in MAGGY2 and PADDY are further described to form a background for section 6 where we consider the organisation of the linear algebra.

2. PROGRAM PACKAGES

A program package consists, in our terminology, of two items

- a problem oriented language(POL) and
- a set of programs.

The POL is discussed further in section 3.
The set of programs contains

- a syntax checker
- a semantics checker-expander
- the principal algorithm
- output programs.

The basic structure of both packages PADDY and MAGGY2 is the same and shown in fig. 2.1.

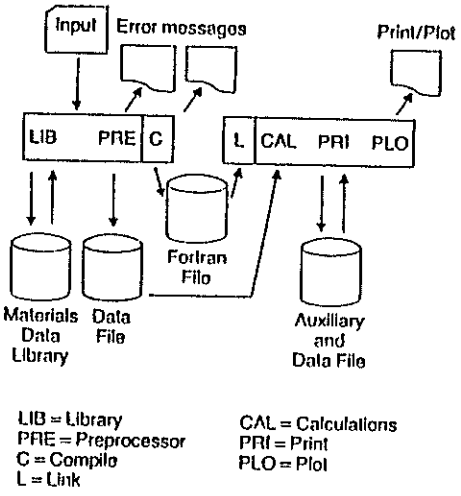


fig. 2.1.

The structure is discussed in detail in [4].
The only point which needs some attention for the rest of this article is the compile-link step. This is done

to enable the inclusion of both generated (by the interpreter) and user specified FORTRAN programs. This way also the FORTRAN fixed dimension problem is solved. The dimension statements are generated.

3. PROBLEM ORIENTED LANGUAGES

A problem oriented language should enable the specification of the following data in a well separated way

- problem description
- algorithm information
- I/O requirements

To enable a specification of the different information types the input is subdivided in blocks. These are

- region block(s),
- algorithm block,
- print,
- plot,
- curve and
- result

blocks

A thorough description of our basic philosophy w.r.t. this topic can be found in [3] and [4]. Because it is essential for the following to understand the basic principles a short survey follows here.
The problem description is given in the region blocks. In those the geometrical and material data are specified per physical part of the problem. We try to keep the terminology here as independent of informatics as possible. The basic rule being that the problem exists without a computer.
Once more, examples are given in [6] and [4].
In practice we find in the region blocks of MAGGY2 and PADDY material property statement such as

BN = row of number pairs
or BN = name
or I = number

and geometry descriptions as discussed in section 4.
The algorithm block contains mesh definition statements and accuracy control statements such as parameters for the termination of iterative processes etc.
Of the further blocks only the curve and result blocks are not selfexplanatory w.r.t. to their usage (see[5]).

4. MESH AND GEOMETRY SPECIFICATION

(4.1) The mesh for MAGGY2 consists of the quadrilaterals formed by two sets of broken intersecting lines as shown in fig. 4.1. Thus the mesh is topologically equivalent to a square mesh.

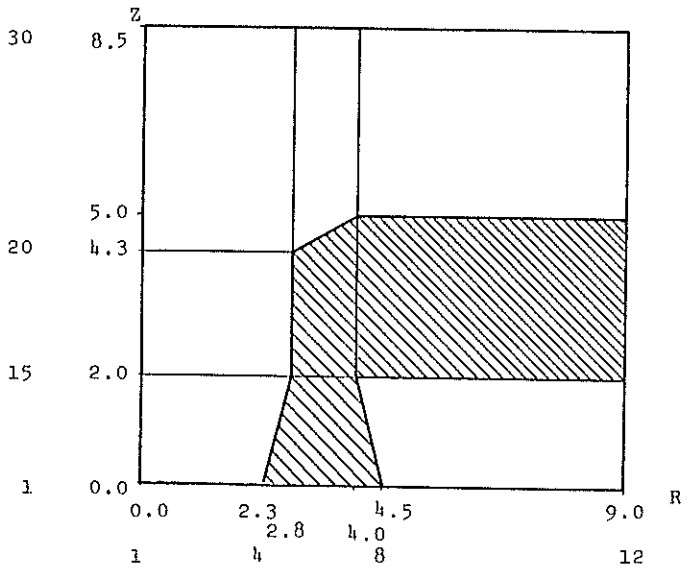


fig. 4.1.

Neighbouring lines must have consecutive numbers. In the algorithm block a "coarse mesh" is defined by giving only those lines that are strictly necessary. The mesh is then completed by linear interpolation.
The meshlines are followed in the mesh statements and their coordinates given.
Thus the mesh in fig. 4.1 is given by
R(1) = 4*0,R(4) = 2.3,3*2.8,R(8) etc.
and
Z(1) = 4*0,...
Here the * symbol is used to indicate that a number of consecutive points in the coarse mesh have the same coordinate along a mesh line.
In general the mesh statement has the form

co(integer) = row of finters.

Where co (coordinate) may be

X,Y,Z,R or PHI

a finter is either a real or a name.
The name can be a function name which corresponds to a user given fortran function see ([3]).
The disadvantages of this mesh are

- restriction of the geometrical flexibility
- usually too many unknowns.

The advantages are
- a perfect bandstructure in the matrix
- easy to specify.

We think that the advantages of this mesh far outweigh the disadvantages for the following reasons. The restriction of the flexibility is not serious in praxis (see [6]), but the mesh is very easy to specify. The algorithms for banded matrices (e.g. [7]) are such that they compensate for the second disadvantage.

(4.2) The mesh for PADDY is a logical 3 dimensional extension fo the MAGGY mesh. It should be stressed that the advantages of this type of mesh in 3D are even larger because thinking in bricks is the only practical way of constructing such a mesh.

However the disadvantage of too many unknowns is also larger. Therefore we first construct a complete mesh, like the MAGGY2 mesh. Then the mesh may be locally refined or points may be deleted with easy special facilities as explained in the following.

(4.3) The mesh is specified by following meshlines in three directions now, instead of two, but further analogous with the MAGGY2 mesh.

A mesh statement has the form.

$$co({}^{i[:i]}_{\star})\,[,({}^{i[:i]}_{\star}),({}^{i[:i]}_{\star})\,]) = n \star R\ldots$$

co = X,Y,Z,R,PHI,THETA
i = integer
n = natural number.

The left hand side of the mesh statement contains at most one $\star$.

With the mesh statement the values of a certain coordinate on one or more meshlines may be given. If for one complete plane a coordinate value is the same this may be given by

co(i) = real

e.g. X(10)= 5.1

If in a meshplane the coordinate value varies, a number of mesh lines in the $\star$ direction can be given in a mesh statement like

X(10,$\star$,3:7) = 4.0,2$\star$5.1,6.0

This statement indicates that the Y mesh lines in the 10th X plane for the Z plane numbers 3 to 7 all have four coarse mesh points with the indicated coordinates. In one meshplane we can have a number of bundles of this type, e.g.

X(10,$\star$,1:2) = 4$\star$4.0
X(10,$\star$,3:7) = 4.0,2$\star$5.1,6.0
X(10,$\star$,8:10) = 3.0,2$\star$5.1,7.0

The number of coordinate values in the r.h.s. must be the same as the number of mesh planes given for the corresponding coordinate. Thus we must have four planes given in the Y-direction for this example.

(4.4) The refine and delete options must be used with the help of the volume statement. With the volume statement

we can indicate a topological hexahedron in the mesh. The volume statement has the form

VOLUME name = co(i:i),co(i:i),co(i:i)

where co is defined as before and the six integers are possibly different integers. Volumes may degenerate into planes, lines or points if the appropriate line numbers are equal.

(4.5) The refine and delete facilities are given with

$$\text{REFINE}[\text{INSIDE}]\text{namelist}({}^{co\ \star\ n}_{n\ \star\ co})[,({}^{co\ \star\ n}_{n\ \star\ co})[,({}^{co\ \star\ n}_{n\ \star\ co})]]$$

$$\text{DELETE}\ \{{}^{[\text{INSIDE}]}_{\text{OUTSIDE}}\}\text{namelist co(i)[co(i)]}\ldots$$

With the first statement we may refine the mesh in volumes given in the namelist. These volumes must correspond to names in volume statements. For each coordinate a refine factor, n, may be given. If different refinements are given for a part of the mesh the union of the points defined in the different refinements will give the total mesh.

With the delete statement we can remove points from the mesh. This is done by using the volume statement either to remove points inside or to remove points outside the volume. The names of the volumes are specified in the namelist and the intersections of the planes following the namelist and in or outsides of the volumes will be omitted.

(4.6) In MAGGY2 a region occupied by one material type is given with the help of area statements of the form

AREA = row of line identifiers

Thus the shaded area in fig 4.1. is given by

AREA = Z(1),R(8),Z(15),R(12),Z(20),R(4)

(4.7) In PADDY the material regions are not given in terms of the mesh but in terms of the actual coordinates. The statements to be used have the following form.

$$\text{LINE name} = {}^{\text{string}}_{\text{namelist}}\ \text{repeat}\ \text{shift}$$

$$\text{SURFACE name} = {}^{\text{string}}_{\text{namelist}}\ \text{repeat}\ \text{shift}\ \text{from to}$$

$$\text{string} = (R,R,R)\{\{{}^{\text{STR}}_{\text{CIR}}\}(R,R,R)\}\ldots\{\{{}^{\text{STR}}_{\text{CIR}}\}[(R,R,R)]\}$$

$$\text{namelist} = \text{name}[,\text{name}]\ldots$$

$$\text{repeat} = \text{REPEAT n}\ \star\ [\text{OVER}]\ (R,R,R)\ \text{OF}[{}^{\text{string}}_{\text{namelist}}]$$

shift = SHIFT[OVER](R,R,R) OF {string / namelist}

from to = FROM{string / namelist} TO {string / namelist}

R = real.

A string is a series of points connected by straights (STR) or circles (CIR). A part of a circle is given by three consecutive points connected by (twice) CIR.
If the last point of the string is equal to the first the last may be omitted and the string terminates with STR or CIR.
In a line statement a string is considered to be a line. In a surface statement a string must be closed to specify a surface. The names in a namelist refer to other lines or surfaces. In this way a number of lines or surfaces may be concatenated to form a larger entity.
With a repeat lines or surfaces can be copied any number of times over a vector (R,R,R). The name in the left hand side of the repeat statement is associated with both the original and the new surfaces or lines.
With a shift a surface or line may be translated once. The name in the left hand side of this statement is only associated with the new line or surface.
With from to, surfaces between two sets of lines can be given. The number of points used in specifying each of the sets of lines must be equal.
(4.8) A region block contains a number of statements of the type described in 4.7. The surfaces specified this way must enclose a volume. Of course there are ample possibilities for user errors here. Therefore the volume described in one region block is rigorously checked on completeness.
Points used in the description of the volumes also must occur in the mesh. All the intersections of the surfaces and the meshlines must be meshpoints. The last still holds after refinements and deletes !

5. ALGORITHMS

(5.1) For the computation of B and H in MAGGY2 we have chosen the usual vector potential formulation combined with Newton-Raphson and on choice of the user, a block-frontal [8] method or an ICCG method [7]. Both are shortly described here. For PADDY we have chosen the composite scalar potential as proposed recently in [9]. The nonlinear equations obtained by using a variational formulation for the approximation of this potential are again solved with Newton-Raphson and an ICCG method is used for the solution of the linear equations. We shall not discuss the variational formulation of the problem or the Newton-Raphson but we will concentrate here on the linear equation solving. In principle we have to solve :
(5.1.1) $Ax = b$
Where A is a positive definite matrix.

(5.2) In MAGGY2 A has the usual nine point band structure. Then we may use a Choleski decomposition organised in the following way.
Say $A = LDL^T$
Then, when constructing a row of L^T we only need information in and above that row of L. Thus only a triangle of coefficients is needed.
In frontal methods (e.g.[11]) assembling and decomposing is always done simultaneously. Suppose we have decomposed one block with a length equal to the band width, then we assemble the following block. Contributions for the third block which are found while doing this are temporarily stored in a work array. When the second block is completely assembled it is decomposed. Then the first block can be stored on disk etc. This is a reasonable combination of I/O and core space. A matrix of length 2500 and band width 50 will be decomposed in some 15 secs on IBM 370/168. This way, occupying 25K bytes.
(5.3) The ICCG method available in MAGGY2 and PADDY is briefly discussed here. A detailed discussion of its implementation in PADDY is given in section 6.
First we construct a cheap approximate decomposition

$$A = LL^T + R$$

Then we solve
(5.3.I) $Mq = v$

with $M = L^{-1}AL^{-T}$, $q = L^T x$ and $v = L^{-1} b$

This means that we have transformed $Ax = b$

into $L^{-1}A(L^T)^{-1} L^T x = L^{-1} b$.
For the solution of 5.3.1 we use the conjugate gradient method. This method is, because M usually has a cluster of eigenvalues with possibly a few exceptions, particularly suitable.
The practical experience reported e.g. in [10] and [12] and our own experience indicates that this indeed is an excellent algorithm for sets of linear equations with a sparse positive definite matrix.
(5.4) The approximate decomposition LL^T of A is constructed by leaving elements of L zero where they are zero in A. This is termed ICCG(0) in [10], whereas ICCG(I) is an algorithm in which I extra elements in each row of L are calculated in positions where A has zeroes.
In the case of a banded matrix, as always found in MAGGY2, the positions of these extra elements are logical (see [10]) but for arbitrary sparse matrices this is not the case. Therefore we have ICCG(2) in MAGGY2 and ICCG(0) in PADDY.
The approximate decomposition of A into LL^T in PADDY is further discussed in section 6.

6. DECOMPOSITION OF A INTO LL^T

(6.1) When no refine or delete statements are used the mesh will stay topologically equivalent to a square mesh (say a finite difference mesh). Then we find a band matrix with, in each row 27 possibly non zero elements. Only those elements a_{ij} are non zero for which

$$|i-J| \in \{0,1,n-1,n,n+1,mn-n-1,mn-n,mn-n+1,mn-1,mn,mn+1,\ mn+n-1,\ mn+n,mn+n+1\}$$

where m and n are the number of meshplanes in resp. X (or R) and Y (or PHI) direction. The number of planes in the Z direction, k, gives the number of unknowns kmn.
However if refines or deletes are used the band structure is disturbed. This even can be such that no band structure can be recognised at all.
In general we do not yet have much knowledge about the practical consequences of this fact. It should be realised here that algorithms for perfect band matrices can be very fast because there is no bookkeeping to speak of.
In case of an arbitrary sparse matrix the bookkeeping will take a not negligible part of the total computing time.
The algorithms therefore have been structured to take a maximum advantage of possible substructures in the matrix.
(6.2) We think the matrix subdivided in logical blocks.
There are three types of logical blocks.

- crystal band,
- band and
- non

structured blocks.
The crystal band structure is generated by a regular mesh.
Band structure is taken in the usual sense and non structure is, as the contradictory name shows, found in a block without band structure.
In fig. 6.2.1 there are three different logical blocks.
These have been subdivided according to the previous logical blocks needed for the decomposition. Typically for block 2 we decompose a block-2 type structure with a block-1 type structure.
From this it may be seen that a decomposition algorithm depends on a pair of block types. There are therefore a number of different decomposition algorithms.

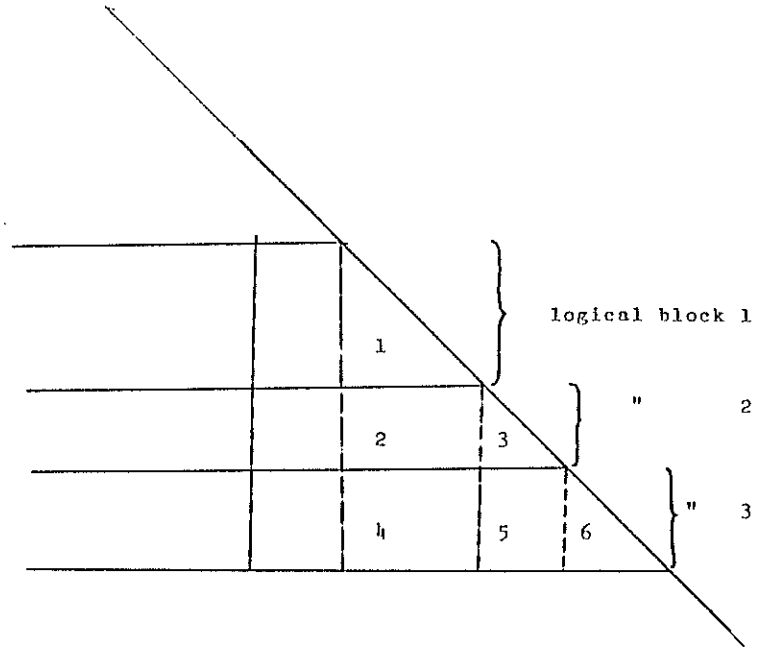


fig. 6.2.1

(6.3) The information needed during the decomposition is found in three in-core arrays and on three disk files.
The array ILGTYP contains a block type indicator for each logical block.
In case of band or crystal band structure the information w.r.t. the column indices is contained in the arrays KLGINF and LGINFO. KLGINF contains the starting adresses of the information in LGINFO per logical block. In case of non-structure the column indices are stored row-wise on a file called K-file. The structure of a record on the K-file is shown in fig. 6.3.1.

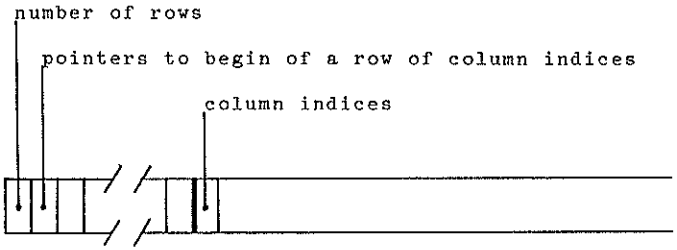


fig. 6.3.1

The nonzero elements of the left lower triangle of A are stored on the A-file and the matrix L is stored on the L-file.
A record of these files has the form shown in fig. 6.3.2

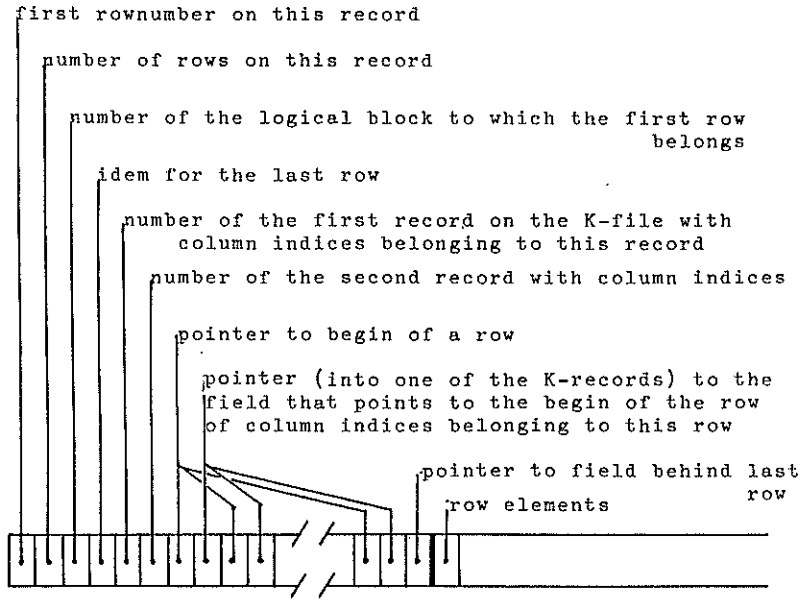


fig. 6.3.2

(6.4) In general only a part of A and L will be in core. Such a part is called a physical block. A physical A-block consists of a number of records. The number of records read from the A-file into HA may be different each time HA is filled, depending on the structure of A. These numbers (of A-file records) are contained in an array NRECA. This array is filled during the assembly of A. A physical L-block consists of one record and is called HL.
As is seen in fig. 6.4.1 HL determines a window in HA. This window can be decomposed with HL.

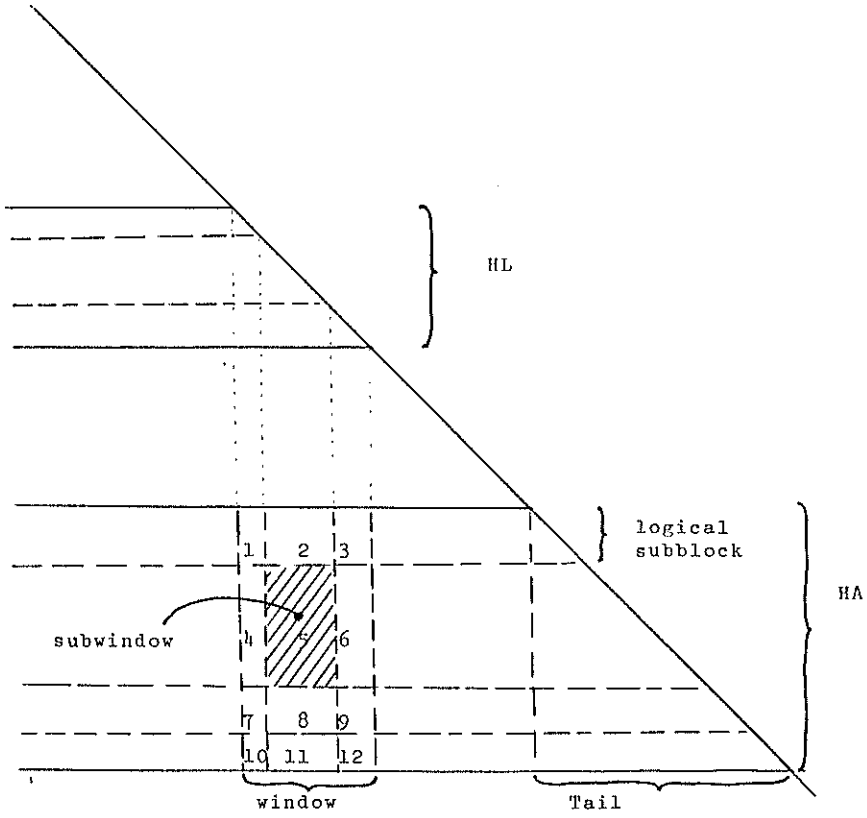


fig. 6.4.1

The intersection of a physical and a logical block is called a logical subblock. The logical subblocks of HA and HL cause a partitioning of a window in subwindows.
The decomposition is performed in the order indicated in fig. 6.4.1. That is subwindow 1,2,3 etc.
Each subwindow is decomposed columnwise.
Thus decomposition is performed
per window, per logical subblock, per subwindow, per column.

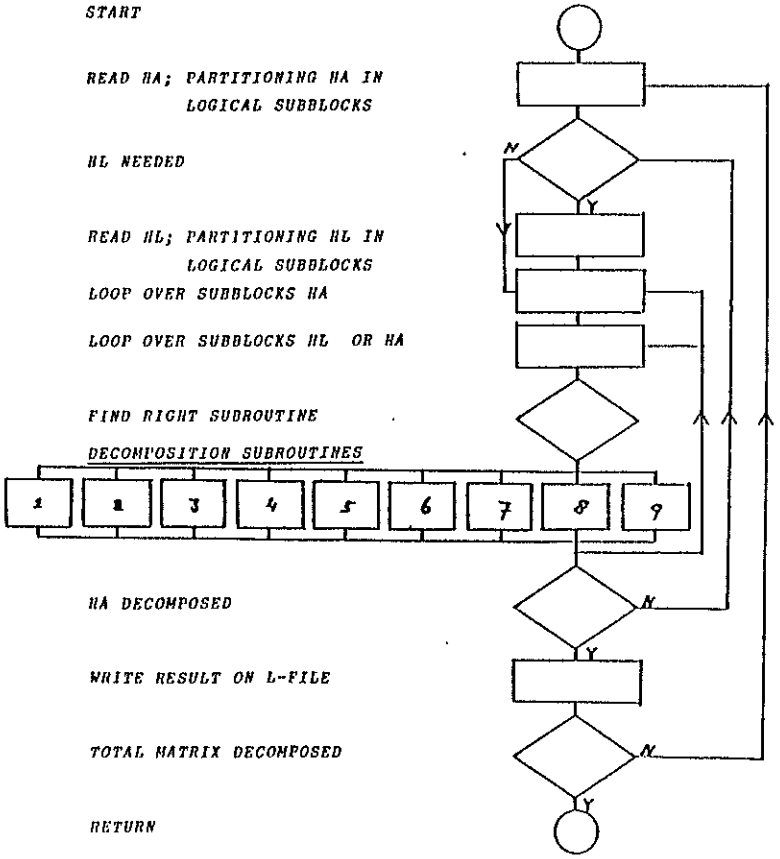


fig. 6.5.1

Once the windows in a physical subblock are decomposed there is a tail left (see fig. 6.4.1).
This tail is decomposed column wise (using HA).
(6.5) The flow chart in fig. 6.5.1. now is selfexplanatory. The HPCO routines perform the actual decomposition and some of them are described in the following.
(6.6) When decomposing the element a_{ij} one has to evaluate the inner product of two rows of L.
This inner product is given by

$$\sum_{k=1}^{j-1} l_{ik} l_{jk} \quad \text{if } j \le i.$$

A fast evaluation of these inner products is the main problem of the decomposition algorithm.
The nonzero elements of row p have distances to the diagonal given by a function

$$f_p : \{1,\ldots,n_p\} \to N$$

Here n_p is the number of nonzero elements in row p and $f_p(k)$ is the column number of the k-th nonzero element in row p, counted from the diagonal.
In the case of crystal band or band structure k counts the subdiagonals of A. We shall also use f_p in that case to index nonzero elements in a column.
So in row i only those elements are decomposed for which

(6.6.1) $i-j = f_i(t)$, $t \in \{1,\ldots,n_i\}$

The only contributions to the innerproducts are given by the triplets (i,j,k) with

(6.6.2) $i-k = f_i(s)$, $s \in \{1,\ldots,n_i\}$ and

(6.6.3) $j-k = f_j(r)$, $r \in \{1,\ldots,n_j\}$

From 6.6.2 and 6.6.3 follows

(6.6.4) $i-j = f_i(s) - f_j(r)$

and from 6.6.4 and 6.6.1

(6.6.5) $f_j(r) = f_i(s) - f_i(t) \qquad s>t$

In turn when 6.6.5 holds then

(6.6.6) $l_{j+f_i(t),j+f_i(t)-f_i(s)} * l_{j,j-f_j(r)} \neq 0$

So the problem is to find triplets (r,s,t) for which 6.6.5 holds.
In case of bandstructures of both logical blocks involved we can use these relations to evaluate the innerproducts in a very fast way.
As mentioned in 6.4 we will be decomposing a column in a subwindow. In that column, j,we will have in the subwindow in question, say, $3 \le t \le 8$.
Then we will find for each t a rownumber $j + f_i(t)$.
(It should be realised that f_i now is used to index the nonzero elements of a column of the structure of row i).

Then we must evaluate the innerproduct involving terms of the type 6.6.6. So we have to establish all pairs r,s for which 6.6.5 holds. In case both blocks have crystal band structure and $f_i = f_j$ there is one table which gives this relation.
This table is given in fig. 6.6.7

t	pairs (r,s)
1	(2,3)(3,4)(5,6)(6,7)(8,9)(9,10)(11,12)(12,13)
2	(1,3)(6,8)(7,9)(9,11)(10,12)
3	(1,4)(5,8)(6,9)(7,10)(8,11)(9,12)(10,13)
4	(5,9)(6,10)(8,12)(9,13)
5	(1,6)(3,8)(4,9)
6	(1,7)(2,8)(3,9)(4,10)
7	(2,9)(3,10)
8	(1,9)(3,11)(4,12)
9	(1,10)(2,11)(3,12)(4,13)
10	(2,12)(3,13)
11	(1,12)
12	(1,13)
13	-

fig. 6.6.7

In case HA has a crystal band structure but $f_i \neq f_j$ there is an auxiliary table.
With that table we can construct a table like 6.6.7 in case HL has band or crystal band structure.
Again suppose we decompose column j in a subwindow. We can find $f_i(t)$ as before. To use relation 6.6.5 we can use the table in fig. 6.6.8. In this table we find the right hand sides of 6.6.5, i.e. $f_i(s) - f_i(t)$ for all the pairs s,t (n.b. s,t only relate to the subwindow structure).

$f_i(s) - f_i(t)$	(s,t)
1	(3,2)(4,3)(6,5)(7,6)(9,8)(10,9)(12,11)(13,12)
2	(4,2)(7,5)(10,8)(13,11)
n-2	(2,1)(8,7)(11,10)
n-1	(3,1)(8,6)(9,7)(11,9)(12,10)
n	(4,1)(8,5)(9,6)(10,7)(11,8)(12,9)(13,10)
n+1	(9,5)(10,6)(12,8)(13,9)
n+2	(10,5)(13,8)
2n-2	(11,7)
2n-1	(11,6)(12,7)
2n	(11,5)(12,6)(13,7)
2n+1	(12,5)(13,6)
2n+2	(13,5)
mn-2n-2	(5,4)
mn-2n-1	(5,3)(6,4)
mn-2n	(5,2)(6,3)(7,4)
mn-2n+1	(6,2)(7,3)
mn-2n+2	(7,2)
mn-n-2	(5,1)(8,4)
mn-n-1	(6,1)(8,3)(9,4)
mn-n	(7,1)(8,2)(9,3)(10,4)
mn-n+1	(9,2)(10,3)
mn-n+2	(10,2)
mn-2	(8,1)(11,4)
mn-1	(9,1)(11,3)(12,4)
mn	(10,1)(11,2)(12,3)(13,4)
mn+1	(12,2)(13,3)
mn+2	(13,2)
mn+n-2	(11,1)
mn+n-1	(12,1)
mn+n	(13,1)

fig. 6.6.8

There are two more possibilities. We may generate a table like 6.6.7 without the help of a table like 6.6.8 or we may find the triples i,j,k without using r,s,t. In the last case a binary search is used to find the nonzero contributions for the innerproduct. The different possibilities can be found in fig. 6.6.9.

HA

HL	crystal	band	non
crystal band $f_i=f_j$	6.6.7	like 6.6.7	binary search
crystal band $f_i \neq f_j$	6.6.8 table like 6.6.7	like 6.6.7	binary search
band	6.6.8 table like 6.6.7	like 6.6.7	binary search
non	binary search	binary search	binary search

fig. 6.6.9.

7. CONCLUSION

In this paper we still miss a display of results obtained. For the programpackage MAGGY2 these results can be found in [4],[6] and [8]. For PADDY we only have some evidence w.r.t. the linear algebra set up. This evidence is still too restricted to draw general conclusions from it. However it already indicates that this is a fast implementation. A further account will be given in the future ,of course.

STRUCTURE OF AN ARRAY-PROCESSOR FOR PARALLEL COMPUTATION OF MAGNETIC FIELDS

A.U.Luccio and G.M.Piacentino
Università di Pisa and INFN-Frascati, Italy

ABSTRACT

In many computer codes to calculate three-dimensional magnetic fields the magnetization of the iron is found by solving non-linear systems of algebraic equations. This is accomplished by iterative methods (Newton-Raphson or the like) in which large linear systems are solved at each step.

The process is long and costly on sequential elaborators, since the complexity of calculation is high. A very large saving can be obtained by suitable partition of the problem, such that the calculations can be accomplished by independent microprocessors working in parallel, and connected to independent memories.

This will lead eventually to the construction of an inexpensive dedicated computer to calculate magnetic fields. The actual structure of this computer is described, consisting of three processors with private memories connected to common memories through a cross-bar switch that settles conflicts of access.

1. INTRODUCTION

The calculation of magnetic fields produced by current-carrying coils in magnetic materials is of the highest interest in the design of electromagnets. The calculations are classically performed by means of large sequential systems on which several codes are run. These codes have reached nowadays a very high level of sofistication, they are however generally very costly, since they need large memory areas and long computing times.

The use of parallel computation on dedicated systems looks attractive, which make use of smaller and cheaper processing units, since at least the times needed for performing the various parts of the calculation should be substantially reduced. Such dedicated computers could become standard equipment of laboratories.

The idea of performing parallel computation is suggested by the intrinsic parallel structure of the calculations done with some of the most popular codes.

In the present work, one of these codes is studied, together with its possible implementation on a parallel computer. As an example, we have chosen the code SNOW[1], written for the calculation of magnetic fields with cylindrical symmetry. It is our aim to show that this code can be conveniently shaped for being run on an array-processor consisting of three CPUs with suitable memories and connections. It is clear that the job can be done even more conveniently by systems made of a larger number of units, and that the present discussion is to be considered only as an example of application, more than a study of a real prototype of a dedicated computer.

The present stress on the parallel rather than on the sequential computation implies, from the side of the programmer, a deeper knowledge of the hardware structure of the system. It is our feeling that this is indeed the path towards wider perspectives on the application of computers to the solution of specific problems that my come in applied physics.

We believe that the "hardware programmer" will take the place of the classical "software programmer" more and more, as the rapid growing field of microprocessors will put at our disposal cheaper and cheaper small units, that will allow us to build complex systems.

2. MATHEMATICAL STRUCTURE OF SNOW

The calculation of the magnetic field generated by current-carrying coils and pieces of magnetic material can be accomplished, in problems with cylindrical symmetry, by means of the code SNOW. We want to discuss the implementation of this code on a parallel computer.

SNOW sets the problem as follows: let us think of a system made of coils and some pieces of magnetic material. The magnetic field in each point is given by the superposition of a field B_c generated by the currents alone plus a field B_m due to the magnetization of the material

$$B = B_c + B_m \qquad (1)$$

At each point P in space, the field $B_m(P)$ can be expressed in its turn as a function of the total field B in each point Q of the material, in the following way

$$B_m(P) = \int g(P,Q)\, \chi(Q)\, B(Q)\, dQ \qquad (2)$$

where g is a function of the coordinates only, which represents the magnetic coupling between each point Q (source) and each point P (field), and χ the magnetic susceptivity at Q. χ is in its turn a

function of $\mathbf{B}$. The integral extends over the volume of the magnetic material.

An explicit expression of the coupling function is the following

$$g(P,Q) = -\frac{1}{4\pi} \nabla_p\left[\frac{\vec{r}}{r^3}\right] \tag{3}$$

where $\vec{r}$ represents the vector P-Q. Eq. (3) is readily derived from the expression of $\mathbf{B}_m$ as a gradient of a scalar magnetic potential.

Once known the functions g and χ , the problem reduces to the solution of the integral equation

$$\mathbf{B}(P) = \mathbf{B}_c(P) + \int g(P,Q)\,\chi(Q)\,\mathbf{B}(Q)\,dQ \tag{4}$$

obtained by combining eqs. (1) and (2).

Let us now subdivide the magnetic material in small parts, which can be considered as homogeneously magnetized. Eq. (4) can be now discretized. If we limit ourselves to problems with cylindrical symmetry, the system of algebraic equations which is equivalent to Eq.(4) can be written in the following way

$$\mathbf{F}(\mathbf{B}) \equiv \left[I - \Gamma X(\mathbf{B})\right]\mathbf{B} - \mathbf{B}_c = 0 \ . \tag{5}$$

Here, Γ is the matrix of the magnetizing coefficients g_{ij} which in the discrete case replace the function g(P,Q). I is the unit matrix. $X(\mathbf{B})$ is the susceptivity matrix, considered as known for the material at hand. $\mathbf{B}$ and $\mathbf{B}_c$ are vectors.

It is important to note that: i) the problem is solved once the field $\mathbf{B}$ is calculated everywhere in the material. To start with, the points P coincide with the points Q and the matrix Γ is square. ii) The system (5) is non linear, since X depends on $\mathbf{B}$. We assume here that the medium is isotropic and hence that χ is a function of the modulus of $\mathbf{B}$ only. iii) The matrix X is diagonal.

In the present case, the subdivision of the magnetic material is made of rings with small cross section, coaxial with the symmetry axis z. The problem is hence a two-dimensional one and the coefficients g_{ij} can be grouped in four categories , according to their coupling the z- or r-component of the magnetization of a ring with the z- or r-component, respectively, of the field in another ring.

Let us group the elements of the Γ matrix four by four as follows

$$\begin{aligned} g_{2i-1,2j-1} &= g^{rr}_{ij} \\ g_{2i,2j} &= g^{zz}_{ij} \\ g_{2i,2j-1} &= g^{zr}_{ij} \\ g_{2i-1,2j} &= g^{rz}_{ij} \end{aligned} \qquad i,j = 1, \ldots n \tag{6}$$

Let us call self-magnetizing coefficients the g's with i=j.

With the following notations

R, Z, ϑ cylindrical coordinates of a source point Q,
r, z, ϑ cylindrical coordinates of a field point P,
s = Z − z, p = R + r, q = R − r,

explicit expressions for the g coefficients (6) are found in Tables I and II.

To solve the non linear algebraic system (5) let us use the iterative Newton-Raphson method, based on the expression[2]

$$\mathbf{B}_{k+1} = \mathbf{B}_k - \mathbf{F}'(\mathbf{B}_k)^{-1}\,\mathbf{F}(\mathbf{B}_k) \ , \tag{7}$$

where $\mathbf{F}'$ is the jacobian matrix of $\mathbf{F}$, built with the derivatives of the components of $\mathbf{F}$ with respect to the components of $\mathbf{B}$. the index k means k-th iteration.

Explicitly, if f' are the components of $\mathbf{F}'$, it is

$$f'_{ij} = \delta_{ij} - g_{ij}\left[\chi(\mathbf{B}_j) + \mathbf{B}_j \frac{\partial \chi(\mathbf{B}_j)}{\partial \mathbf{B}_j}\right] \ . \tag{8}$$

Eq.(8) shows that, with the function $\chi(\mathbf{B})$, also the function

$$\chi' = \chi(\mathbf{B}) + \mathbf{B}\frac{\partial \chi}{\partial \mathbf{B}} \tag{9}$$

can be considered as known.

Eq.(7) is always converging if the starting "working point" is not too far from the "true" solution.

3. CODIFYING THE PROBLEM FOR ITS COMPUTER HANDLING

The first problem is that of computing the coefficients g given in the Tables I and II, which are functions solely of the geometry of the subdivisions of the magnetic material (rings in this case).

This calculation can be performed sequentially, i.e. by computing a coefficient next to the other, or in parallel.

The parallel computation can be made by subdividing the intero job in smaller tasks, some of which are not in conflict, since they start from the same conditions. These tasks are then executed in parallel by several central units, which access to dynamically independent memories.

As an example, as we will show in more detail in the following, by making use of three CPUs and of three memories with adequate connections and synchronization, it is possible to calculate at the same time the four coefficients shown in Table I with evident saving of time.

The optimization of this procedure depends on the length of each first-level task assigned to each unit. We can think of the structure of the flow diagram as drawn on several sheets, and this stratifying such that the execution of each task in parallel requires only the use of values calculated in parallel on the preceding sheet.

The second problem is that of the solution by successive iterations of the algebraic non linear system (5) by the Newton-Raphson method. Using Eqs.(7) is equivalent to linearize the system, step by step, since Eqs. (7) are linear. The iteration is an intrinsically sequential process and it cannot be dealt with in parallel. The calculation of (7) at each step instead,can be parallelized with large saving, analogously to what done for the g coefficients.

In this calculation, the parallelization applies to the evaluation of the elements of the F' matrix, as well as to the inversion of F'. It applies also to the multiplication of F'^{-1} by F. In particular, we will show on an example how convenient it is to invert a matrix in parallel.

The third problem is that of the calculation of the matrix Γ, not square in the most general case, that connects the points inside the magnetic material to points in the external field, where we like to know the field, according to Eq.(1). This problem coincides with the first one.

4. HARDWARE STRUCTURE OF THE PROPOSED SYSTEM

It is beyond the purposes of the present paper to discuss the general problems of the design of a multi-processor computer. We will limit ourselves to the presentation of a possible architecture. This structure is shown in Fig.1. It is composed by three CPUs, called α, β, and γ; by a memory subdivided in three independent areas A, B, and C; by three ROMs, denoted by 1, 2, 3, needed to contain the programs for α, β, and γ, respectively; by a cross-bar switch CS which connects all the described elements, and which allows the simultaneous access of each processor to at least a different memory; and finally by a clock CK.

The three identical CPUs α, β, and γ are capable of performing logical and arithmetic operations, other than operations as load, shift, fetch, etc.

The memories A, B, and C contain the data at the beginning and during the elaboration, as well as the results. In A, B, and C there are three areas, called work areas, designed to allow the communications between couples of processors during the elaboration.

The memories ROM1, ROM2, and ROM3, in the present case, are special-purpose, read-only.

For our project we have chosen the single-chip Zilog Z-80 microprocessors, also if other processors could also be considered. The main reason was that in our Computer Science Department in Pisa there were already those processors, together with a Zilog developing system useful for testing, debugging and assembling programs. Moreover the Z-80 are very efficient and inexpensive, and finally the manufacturer offers several ancillary equipments, useful in the design of the remaining parts of our system[3]

As it is shown by Fig. 1, each ROM is not directly connected to the corresponding processor, but via the CS. This particular structure is due to the organization of the Z-80. The bus structure on the CS is shown in Fig. 2. There, the capital letters mean 16-bit buses used for address transfer: L, M, N are output buses from CS, which fulfill the task of selecting the addresses in the ROM memories. G, H, I are also output buses from the CS, which feed the three general memories A, B, C of Fig. 1. D, E, F are instead output buses from the processors α, β, and γ and are used to feed into the CS the address of the chosen word.

Small letters denote 8-bit buses. These are used to transfer data and instructions from and to processors and memories: a, b, c are one-way buses from the read-only memories, while d, e, f, g, h, i are two-way buses connected with the three processors and with the general memories.

The CS performs also other operations, other than transferring data and instructions. It sequentializes the requests for access to a memory in the case of conflicts, and contains special structures that allow the correct order of write-read access to the same unit.

5. A PARALLEL ALGORITHM FOR THE GENERATION OF THE X-MATRIX. AN EXAMPLE

A possible algorithm for the generation of the X-matrix by means of the three-microprocessor system will be described partially here. For evident reasons of space, it is not useful to describe the whole algorithm. We will limit ourselves to show how it is possible to share specific tasks to the three CPUs, to perform the calculation of the arguments in the integrals of Table I, for a given value of the variables R, r, Z, z. This procedure employs simultaneously each processor and each memory. The timing is given by the clock CK.

The example is described diagrammatically in Tables III, IV and V. Table I shows the sequence of the instructions assigned by the ROMs to the three processors α, β, and γ. Table IV shows the content of the registers of these processors. Since these registers are twelve in each unit, some over-writing will occur. Table V shows the content of the work areas in A, B, C at each time.

By reading down through Table III, it appears clear that it is an explicit parallel program that requires some optimization in the sharing of the tasks among the units. Here indeed, at variance with a sequential program, the time used is not directly proportional to the total number of instructions.

Special operations encountered are the square root and the calculation of the elliptic integrals. These are also performed in parallel and constitute the longest tasks to be performed by the central units.

For the square root, we suggest to use an iterative algorithm based on the formula

$$r_{n+1} = \frac{1}{2}\left[r_n + \frac{R}{r_n} \right] \quad , \tag{10}$$

where r_n represents the root at the n-th iteration, and R is the given radicand. As a starting value, we suggest to take $r_o = R$.

To evaluate the elliptic integrals of first and second kind K(m) and E(m) of the modulus m, it is possible to use a polynomial development with seventeen coefficients . These coefficients are tabulated as data inside the ROMs, under the form of "operation between register and ready value".

6. A PARALLEL ALGORITHM FOR NEWTON-RAPHSON ITERATIONS

The second problem stated above consists in the iterative solution of an algebraic non linear system with the Newton-Raphson formula (7). At each iteration it is necessary to invert the jacobian matrix F', whose components are given by (8). Our problem is hence that of inverting a matrix by a parallel algorithm.

It can be shown that a square matrix of rank n can be inverted in parallel, if we express it in terms of partial matrices and operate on these smaller-rank matrices[5]. In particular, if the original matrix is of rank multiple of three, such process can be made advantageously with a three-microprocessor structure, similar to the present one. Our problem can be solved therefore in this way, by ordering in the memories A, B, and C the elements of the Γ-matrix, and then of the F'-matrix, as to prepare the latter to its subdivision in 3×3 matrices.

The problem of the inversion by blocks can be discussed in the following way. Let us write a n×n matrix as follows

$$A_{n+1} = \begin{pmatrix} A_n & U_n \\ V_n & a_{n+1,n+1} \end{pmatrix}, \tag{11}$$

where A_n is also a matrix, U_n and V_n are vectors, and $a_{n+1,n+1}$ is a scalar quantity. The inverse matrix can be written

$$A_{n+1}^{-1} = \begin{pmatrix} B_n & R_n \\ Q_n & \alpha_{n+1,n+1}^{-1} \end{pmatrix} \quad , \tag{12}$$

where

$$\begin{aligned} R_n &= - A_n^{-1} U_n \alpha_{n+1,n+1}^{-1} \\ \alpha_{n+1,n+1} &= a_{n+1,n+1} - V_n A_n^{-1} U_n \\ B_n &= A_n^{-1} - A_n^{-1} U_n Q_n \\ Q_n &= - V_n A_n^{-1} \alpha_{n+1,n+1}^{-1} \end{aligned} \tag{13}$$

are the elements of A_{n+1}^{-1} . A_n^{-1} is the inverse matrix of A_n, R_n and

B_n are vectors, while $\alpha_{n+1,n+1}$ is a scalar.

Actually, Eqs.(13) hold also if the components of the vectors and of the matrices and the scalars are replaced by matrix of rank m. In particular, in our case, we can choose m=3.

By bordering iteratively a rank-3 matrix, it is possible to build the inverse of a matrix of whatever rank n multiple of 3, if we perform at each iteration only products and sums of 3×3 matrices plus a single inversion of a 3×3 matrix.

It is therefore important to show the algorithm for the parallel inversion of a 3×3 matrix.

Let us consider the matrix of numbers

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} . \qquad (14)$$

The explicit algorithm to invert this matrix, the register contents in the processors and the work areas in the memories A, B, C are shown in the Tables VI, VII, VIII.

The bordering algorithm to invert a matrix, starting from simpler matrices, is much faster than the classic algorithms to be performed on sequential systems. An analysis of the computational complexity is made in Ref.[5]

To perform matrix multiplications, finally, an algorithm similar to the Winograd's[6] can be employed, if we observe that each element of the product matrix C = A·B can be written as follows

$$c_{ij} = \sum_{k=1}^{3} a_{ik} b_{kj} = (a_{i1}+b_{2j})(a_{i2}+b_{1j}) - a_{i1}b_{j2} - b_{1j}b_{2j} + a_{i3}b_{3j} .$$

This formula is justified by the fact that sums are much faster operations than products, and here are there less products and more sums than in classical sequential algorithms.

7. REFERENCES

1. Luccio, A.U. On the Calculation of Magnetic Fields in Magnets with Cylindrical Symmetry. University of Maryland. Tech. Rept. No. 74-104, July 1974.

   Luccio, A.U. Proceedings Fifth Intern. Conf. on Magnet Technology (MT-5). Roma, April 21, 1975. p.183.

2. Ortega, J.M. and Rheinboldt, W.C. Iterative Solution of Nonlinear Equations in Several Variables. Academic Press 1970.

3. Faggin, F. Introduction to Microprocessors and Systems Zilog Z-80. Lecture Notes at the University of Milan, Italy. November 1976.

   Zilog, Z-80-CPU. Technical Manual. 1977.

4. Abramowitz, M. and Stegun, I.A. Handbook of Mathematical Functions. National Bureau of Standards, 1964.

5. Bini, D., Luccio, A.U., and Piacentino, G.M. Parallel Matrix Inversion with a Three-processor Computer. Frascati Tech. Report LNF-78/14(R). 7 March 1978.

6. Csanky, L. Ph. D. Thesis University of California at Berkeley. Dept. of Engineering. 1973.

Table I. Magnetizing mutual coefficients for rings

$$g_{zz} = \frac{1}{4\pi}\int ds \int \sigma\{c_1 E + c_2 (K-E)\} R\, dR$$

$$g_{zr} = \frac{1}{4\pi}\int ds \int \sigma\{c_3 E + c_4 (K-E)\} R\, dR$$

$$g_{rz} = \frac{1}{4\pi}\int ds \int \sigma\{c_5 E + c_6 (K-E)\} R\, dR$$

$$g_{rr} = \frac{1}{4\pi}\int ds \int \sigma\{c_7 E + c_8 (K-E)\} R\, dR$$

$c_1 = p^2/(p^2+s^2) - 2\,s^2/(q^2+s^2)$, $c_2 = s^2/(p^2+s^2)$,

$c_3 = -s\{p/(p^2+s^2) + 2\,q/(q^2+s^2)\}$, $c_4 = \frac{s}{p+q}\cdot(pq-s^2)/(p^2+s^2)$,

$c_5 = s\{p/(p^2+s^2) - 2\,q/(q^2+s^2)\}$, $c_6 = \frac{s}{p-q}\cdot(pq+s^2)/(p^2+s^2)$,

$c_7 = p^2/(p^2+s^2) - 2\,q^2/(q^2+s^2)$, $c_8 = \left[q^2 + p^2(q^2+s^2)/(p^2+s^2)\right]/(p^2-q^2)$.

$p=R+r$, $q=R-r$, $s=Z-z$, $\sigma = (q^2+s^2)^{-1}(p^2+s^2)^{-1/2}$,

K and E are elliptic integrals with modulus $m = (p^2-q^2)/(p^2+s^2)$.

Table II. Magnetizing self coefficients for rings

$$g_z = -\frac{2b}{\pi}\int_{r-a}^{r+a} \left\{(R+r)^2+b^2\right\}^{-1/2} \left\{(R-r)^2+b^2\right\}^{-1} E(m)\, R\, dR$$

$$g_r = -\frac{2b}{\pi a^2}\left[(r+a)(1-\frac{a}{r})\, I_{10} + (r-a)(1+\frac{a}{r})\, I_{11} - \frac{2R}{a}(r+a)^2(1-\frac{a}{r})\, I_{20} + \frac{2R}{a}(r-a)^2(1+\frac{a}{r})\, I_{21}\right]$$

$m = 4Rr/\left\{(R+r)^2+b^2\right\}$, where a and b are the minor dimensions, radial and axial, of the ring.

$$I_{10} = \int_0^{\pi/2} (1+k_o^2\sin^2\varphi)^{-3/2} d\varphi \quad , \quad I_{11} = \int_0^{\pi/2} (1+k_1^2\sin^2\varphi)^{-3/2} d\varphi \quad ,$$

$$I_{20} = \int_0^{\pi/2} (1+k_o^2\sin^2\varphi)^{-3/2}\sin^2\varphi\, d\varphi \quad , \quad I_{21} = \int_0^{\pi/2} (1+k_1^2\sin^2\varphi)^{-3/2}\sin^2\varphi\, d\varphi,$$

$k_o^2 = 4r(r+a)/a^2$, $k_1^2 = 4r(r-a)/a^2$.

Table III. Parallel algorithm to calculate the arguments for the integrals of Table I. Instructions given to the three CPUs.

CK times	α	β	γ
t_1, t_2	$p=R+r \rightarrow A$	$q=R-r \rightarrow B$	$s=Z-z \rightarrow C$
t_3, t_4	$p^2 \rightarrow A$	$q^2 \rightarrow B$	$s^2 \rightarrow C$
t_5	$B \rightarrow q$	$C \rightarrow s$	$A \rightarrow p$
t_6	$B \rightarrow q^2$	$C \rightarrow s^2$	$A \rightarrow p^2$
t_7	$C \rightarrow s^2$	$A \rightarrow p^2$	$B \rightarrow q^2$
t_8, t_9	$p^2+s^2 \rightarrow A$	$q^2+s^2 \rightarrow B$	$p^2-q^2 \rightarrow C$
t_{10}, t_{11}	$p^2/(p^2+s^2) \rightarrow A$	$q^2/(q^2+s^2) \rightarrow B$	$q^2/(p^2-q^2) \rightarrow C$
t_{12}	$B \rightarrow q+s^2$	$C \rightarrow p-q^2$	$A \rightarrow p+s^2$
t_{13}, t_{14}	$s^2/(q^2+s^2) \rightarrow B$	$p^2/(p^2-q^2) \rightarrow C$	$c_2 = s^2/(p^2+s^2) \rightarrow A$
t_{15}	pq	qs	ps
t_{16}, t_{17}	$pq/(p+s^2) \rightarrow A$	$qs/(q+s^2) \rightarrow B$	$ps/(p+s^2) \rightarrow C$
t_{18}, t_{19}	$q^2/(p^2+s^2) \rightarrow B$	$qs/(p^2-q^2) \rightarrow C$	$ps/(p^2-q^2) \rightarrow A$
t_{20}	$B \rightarrow q^2/(q^2+s^2)$	$A \rightarrow p^2/(p^2+s^2)$	$C \rightarrow qs/(p^2-q^2)$
t_{21}	$B \rightarrow qs/(q^2+s^2)$	$C \rightarrow ps/(p^2+s^2)$	$A \rightarrow pq/(p^2+s^2)$
t_{22}	$2s^2/(q^2+s^2)$	$2qs/(q^2+s^2)$	$s/(p+q)$
t_{23}, t_{24}	$c_1 \rightarrow B$	$-c_3 \rightarrow C$	$s/(p-q)$
			$(pq-s^2)/(p^2+s^2)$
t_{25}, t_{26}, t_{27}	$2q^2/(q^2+s^2)$	$c_5 \rightarrow A$	$c_4 \rightarrow B$
	$c_7 \rightarrow B$	$C \rightarrow q^2/(p^2-q^2)$	$(pq+s^2)/(p^2+s^2)$
t_{28}, t_{29}	$m \rightarrow C$	$T = p^2/(p^2+s^2)+1$	$c_6 \rightarrow B$
		$A \rightarrow c_2$	
t_{30}	$A \rightarrow p$	$U = c_2 p^2/(p^2-q^2)$	$A \rightarrow p^2+s^2$
.....	$B \rightarrow q$	$V = Tq^2/(p^2-q^2)$	$B \rightarrow q^2+s^2$
.....	$p+q$	$c_8 = U+V \rightarrow C$	
t_{34}	$R=(p+q)/2 \rightarrow A$		
iterations	K(m)	E(m)	$\sqrt{p^2+s^2}$

Table III. Continuation.

CK times	α	β	γ
iterations	K(m) $\longrightarrow$ A	E(m) $\longrightarrow$ B	$\sigma=(q^2+s^2)^{-1}(p^2+s^2)^{-\frac{1}{2}}$
.....			A $\longrightarrow$ R
			σ B $\longrightarrow$ A
t_1	B $\longrightarrow$ E(m)	C $\longrightarrow$ $-c_3$	A $\longrightarrow$ c_5
t_2	K-E	$-c_3$E	B $\longrightarrow$ E
t_3	K-E $\longrightarrow$ A	B $\longrightarrow$ c_1	c_5E
t_4, t_5	A $\longrightarrow$ c_2	c_1E $\longrightarrow$ C	c_5E $\longrightarrow$ B
t_6	c_2(K-E)	A $\longrightarrow$ K-E	B $\longrightarrow$ c_7
t_7	C $\longrightarrow$ c_1E	B $\longrightarrow$ c_4	A $\longrightarrow$ K-E
t_8	c_1E+c_2(K-E)	c_4(K-E)	c_7E
t_9	A $\longrightarrow$ σR	c_4(K-E)+c_3E	C $\longrightarrow$ c_8
t_{10}	Arg(g_{zz})	$\longrightarrow$ C	c_8(K-E)
t_{11}	$\longrightarrow$ A	B $\longrightarrow$ c_5E	c_7E+c_8(K-E)
t_{12}	C $\longrightarrow$ c_3E+c_4(K-E)	B $\longrightarrow$ c_6	Arg(g_{rr})
t_{13}	Arg(g_{zr})	c_6(K-E)	$\longrightarrow$ A
t_{14}	$\longrightarrow$ C	c_5E+c_6(K-E)	
t_{15}		A $\longrightarrow$ σR	
t_{16}		Arg(g_{rz})	
t_{17}		$\longrightarrow$ B	

Table IV. Parallel algorithm to calculate the arguments for the integrals of Table I. Register content of the three processors.

	α		β		γ	
1	R	$pq/(p^2+s^2)$	r	$qs/(q^2+s^2)$	Z	$ps/(p^2+s^2)$
2	r	$q^2/(p^2+s^2)$	R	$qs/(p^2-q^2)$	z	$ps/(p^2-q^2)$
3	p	$q^2/(q^2+s^2)$	q	$p^2/(p^2+s^2)$	s	$qs/(p^2-q^2)$
4	p^2	$qs/(q^2+s^2)$	q^2	$ps/(p^2+s^2)$	s^2	$pq/(p^2+s^2)$
5	q	$2s^2/(q^2+s^2)$	s	$2qs/(q^2+s^2)$	p	$s/(p+q)$
6	q^2	c_1	s^2	$-c_3$	p^2	$s/(p-q)$
7	s^2	$2q^2/(q^2+s^2)$	p^2	c_5	q^2	$(pq-s^2)/(p^2+s^2)$
8	p^2+s^2	c_7	q^2+s^2	$q^2/(p^2-q^2)$	p^2-q^2	c_4
9	$p^2/(p^2+s^2)$	m	$q^2/(q^2+s^2)$	T	$q^2/(p^2-q^2)$	$(pq+s^2)/(p^2+s^2)$
10	q^2+s^2	p	p^2-q^2	c_2	p^2+s^2	c_6
11	$s^2/(q^2+s^2)$	q	$p^2/(p^2-q^2)$	U	c_2	p^2+s^2
12	pq	p+q	qs	V ...	ps	q^2+s^2 ....

Table V. Parallel algorithm to calculate the argoments for the integrals of Table I. Work area content of the three memories.

CK	A	B	C
t_2	p	q	s
t_4	p^2	q^2	s^2
t_9	$p^2/(p+s^2)$	$q^2/(q+s^2)$	$p^2/(p-q^2)$
t_{11}		$s^2/(q+s^2)$	$q^2/(p-q^2)$
t_{14}	c_2		$p^2/(p-q^2)$
t_{17}	$pq/(p+s^2)$	$qs/(q+s^2)$	$ps/(p+s^2)$
t_{19}	$ps/(p-q^2)$	$q^2/(p+s^2)$	$qs/(p-q^2)$
t_{23}		c_1	$-c_3$
t_{25}	c_5	c_4	
t_{27}		c_7	
t_{28}		c_6	m
t_{34}	R		c_8
---	K	E	σR
t_3	$K-E$		
t_5		c_5E	c_1E
t_{10}			$c_4(K-E)+c_3E$
t_{11}	$Arg(g_{zz})$		
t_{13}	$Arg(g_{rr})$		
t_{14}			$Arg(g_{zr})$
t_{17}		$Arg(g_{rz})$	

Table VI. Parallel algorithm to invert the 3×3 matrix of Eq.(14). Sequence of instructions given to the three CPUs by the ROMs.

CK times	α	β	γ	
t_1	$C \longrightarrow a_{13}$	$A \longrightarrow a_{21}$	$B \longrightarrow a_{32}$	
t_2	$A \longrightarrow a_{21}$	$B \longrightarrow a_{32}$	$C \longrightarrow a_{13}$	
t_3, t_4	$a_{13}a_{21} \longrightarrow A$	$a_{21}a_{32} \longrightarrow B$	$a_{32}a_{13} \longrightarrow C$	
t_5	$B \longrightarrow a_{12}$	$C \longrightarrow a_{23}$	$A \longrightarrow a_{31}$	
t_6	$C \longrightarrow a_{23}$	$A \longrightarrow a_{31}$	$B \longrightarrow a_{12}$	
t_7	$a_{12}a_{23} \longrightarrow A$	$a_{23}a_{31} \longrightarrow B$	$a_{31}a_{12} \longrightarrow C$	
t_8	$A \longrightarrow a_{11}$	$B \longrightarrow a_{22}$	$C \longrightarrow a_{33}$	
t_9	$B \longrightarrow a_{22}$	$C \longrightarrow a_{33}$	$A \longrightarrow a_{11}$	
t_{10}, t_{11}	$a_{11}a_{22} \longrightarrow A$	$a_{22}a_{33} \longrightarrow B$	$a_{33}a_{11} \longrightarrow C$	
t_{12}, t_{13}	$a_{12}a_{21} \longrightarrow A$	$a_{23}a_{32} \longrightarrow B$	$a_{31}a_{13} \longrightarrow C$	
t_{14}, t_{15}	$a_{13}a_{22} \longrightarrow A$	$a_{21}a_{33} \longrightarrow B$	$a_{32}a_{11} \longrightarrow C$	
t_{16}, t_{17}	$a_{11}a_{23} \longrightarrow A$	$a_{22}a_{31} \longrightarrow B$	$a_{33}a_{12} \longrightarrow C$	
t_{18}	$a_{13}a_{21}-a_{11}a_{23}$	$a_{21}a_{32}-a_{22}a_{31}$	$a_{13}a_{32}-a_{12}a_{33}$	
t_{19}, t_{20}	$B \longrightarrow a_{32}^*$	$C \longrightarrow a_{13}^*$	$A \longrightarrow a_{21}^*$	(§)
t_{21}	$a_{12}a_{23}-a_{13}a_{22}$	$a_{22}a_{33}-a_{23}a_{32}$	$-a_{31}a_{12}+a_{32}a_{11}$	
t_{22}, t_{23}	$A \longrightarrow a_{31}^*$	$B \longrightarrow a_{12}^*$	$C \longrightarrow a_{23}^*$	(§)
t_{24}	$a_{11}a_{22}-a_{12}a_{21}$	$a_{23}a_{31}-a_{21}a_{33}$	$a_{33}a_{11}-a_{31}a_{13}$	
t_{25}, t_{26}	$C \longrightarrow a_{33}^*$	$A \longrightarrow a_{11}^*$	$B \longrightarrow a_{22}^*$	(§)
t_{27}, t_{28}	Δ	Δ	Δ	($)
t_{29}	$\frac{a_{11}a_{22}-a_{12}a_{21}}{\Delta}$	$\frac{a_{22}a_{33}-a_{23}a_{32}}{\Delta}$	$\frac{a_{11}a_{33}-a_{31}a_{13}}{\Delta}$	
t_{30}	$=a_{33}^{-1} \longrightarrow C$	$=a_{11}^{-1} \longrightarrow A$	$=a_{22}^{-1} \longrightarrow B$	
t_{31}	$\frac{a_{12}a_{23}-a_{13}a_{22}}{\Delta}$	$\frac{a_{23}a_{31}-a_{21}a_{33}}{\Delta}$	$\frac{a_{31}a_{12}-a_{32}a_{11}}{\Delta}$	
t_{32}	$=a_{31}^{-1} \longrightarrow A$	$=a_{12}^{-1} \longrightarrow B$	$=a_{23}^{-1} \longrightarrow C$	
t_{33}	$\frac{a_{13}a_{21}-a_{11}a_{23}}{\Delta}$	$\frac{a_{21}a_{32}-a_{22}a_{31}}{\Delta}$	$\frac{a_{13}a_{32}-a_{12}a_{33}}{\Delta}$	
t_{34}	$=a_{32}^{-1} \longrightarrow B$	$=a_{13}^{-1} \longrightarrow C$	$=a_{21}^{-1} \longrightarrow A$	

Table VII. Parallel algorithm to invert the 3×3 matrix of Eq.(14). Register content of the three processors.

	α		β		γ		
1	a_{13}	$a_{13}a_{21}-a_{11}a_{23}$	a_{21}	$a_{21}a_{32}-a_{22}a_{31}$	a_{32}	$a_{13}a_{32}-a_{12}a_{33}$	
2	a_{21}	a_{32} *	a_{32}	a_{13} *	a_{13}	a_{21} *	(§)
3	$a_{13}a_{21}$	$a_{12}a_{23}-a_{13}a_{22}$	$a_{21}a_{32}$	$a_{22}a_{33}-a_{23}a_{32}$	$a_{32}a_{13}$	$a_{32}a_{11}-a_{31}a_{12}$	
4	a_{12}	a_{31} *	a_{23}	a_{12} *	a_{31}	a_{23} *	(§)
5	a_{23}	$a_{11}a_{22}-a_{12}a_{21}$	a_{31}	$a_{23}a_{31}-a_{21}a_{33}$	a_{12}	$a_{33}a_{11}-a_{31}a_{13}$	
6	$a_{12}a_{23}$	a_{33} *	$a_{23}a_{31}$	a_{11} *	$a_{31}a_{12}$	a_{22} *	
7	a_{11}	Δ'	a_{22}	Δ'	a_{33}	Δ'	($)
8	a_{22}	Δ	a_{33}	Δ	a_{11}	Δ	($)
9	$a_{11}a_{22}$	a_{33}^{-1}	$a_{22}a_{33}$	a_{11}^{-1}	$a_{33}a_{11}$	a_{22}^{-1}	
10	$a_{12}a_{21}$	a_{31}^{-1}	$a_{23}a_{32}$	a_{12}^{-1}	$a_{31}a_{13}$	a_{23}^{-1}	
11	$a_{13}a_{22}$	a_{32}^{-1}	$a_{21}a_{33}$	a_{13}^{-1}	$a_{32}a_{11}$	a_{21}^{-1}	
12	$a_{11}a_{23}$		$a_{22}a_{31}$		$a_{33}a_{12}$		

(§) * means: multiply by the preceding value.

($) The determinant Δ is calculated in two steps.

Table VIII. Parallel algorithm to invert the 3×3 matrix of Eq.(14). Work area content of the three memories.

CK	A	B	C
	a_{11}	a_{12}	a_{13}
	a_{21}	a_{22}	a_{23}
	a_{31}	a_{32}	a_{33}
t_4	$a_{13}a_{21}$	$a_{21}a_{32}$	$a_{32}a_{13}$
t_7	$a_{12}a_{23}$	$a_{23}a_{31}$	$a_{31}a_{12}$
t_{11}	$a_{11}a_{22}$	$a_{22}a_{33}$	$a_{33}a_{11}$
t_{12}	$a_{12}a_{21}$	$a_{23}a_{32}$	$a_{31}a_{13}$
t_{15}	$a_{13}a_{22}$	$a_{21}a_{33}$	$a_{32}a_{11}$
t_{17}	$a_{11}a_{23}$	$a_{22}a_{31}$	$a_{33}a_{12}$
t_{30}	a_{11}^{-1}	a_{22}^{-1}	a_{33}^{-1}
t_{32}	a_{31}^{-1}	a_{12}^{-1}	a_{23}^{-1}
t_{34}	a_{21}^{-1}	a_{32}^{-1}	a_{13}^{-1}

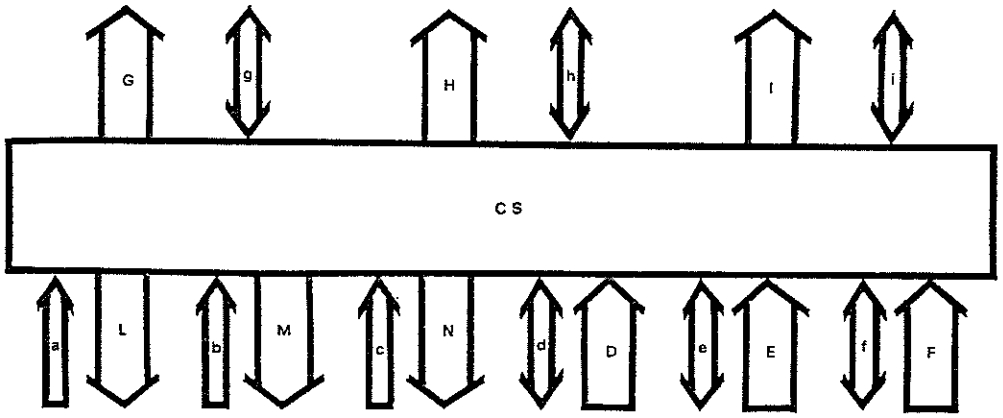


Fig.2 - Bus structure of the cross-bar switch CS for the three processor computer of Fig. 1. Capital letters denote 16-bit buses; small letters 8-bit buses.

Fig.1- Structure of a three-microprocessor computer. α , β , and γ are the CPUs; A, B, and C are memories; CS is a cross-bar switch; CK a clock. The ROMs contain instructions for α, β , γ .