A PARALLEL ALGORITHM FOR FINITE ELEMENT COMPUTATION

P. Subramaniam

Picker International
595 Miner rd.
Highland Hights, OH. 44143

N. Ida

Electrical Engineering Department
The University of Akron
Akron, OH. 44325

ABSTRACT

The work presented here deals with the parallel implementation of finite element analysis algorithms for computation of electromagnetic fields. The methods apply equally well to other areas. The choice of a parallel implementation is based on the fact that many of the operations and algorithms used for finite element analysis (FEM) are essentially parallel or can be parallelized with a moderate level of effort. The solution of electromagnetic field problems is particularly appropriate into the context of parallel machines because of the open boundary nature of the problem and the size of the finite element matrices.

INTRODUCTION

The basic stages of the finite element method have been parallelized and a working implementation has been tested. The first stage in the solution is the definition of elemental matrices. The sizes of these matrices range from as little as 3x3 for a simple 2-D element to 60x60 or more for 3-D elements. The solution of the system of equations is handled by a parallelized Gaussian elimination algorithm. Postprocessing consists essentially of calculating field intensities and flux densities as well as coil impedances.

The essential steps in FEM analysis are:

1. Discretization of the solution domain
2. Calculation of elemental matrices
3. Assembly of a global matrix
4. Solution of the system of equations
5. Post-processing of the results

This work concentrates on steps 2 and 3. Step 1 is a geometrical problem that includes definition of a geometry, decision on a discretization level, input of a variety of geometrical data, material properties etc. This is usually handled through a graphics preprocessor and is not suited for parallel machines. The interaction of the designer with the computer at this stage is essential and therefore, the best approach is to use a graphics workstation. The input for the FEM program is a geometric and problem dependent data file. This is assumed to have been generated for the purpose of this work.

The solution of the system of equations generated in the FEM process has been treated elsewhere [2]. The parallel solution routines developed are used in conjunction with the programs described here.

Finally, the postprocessing step has been left out because of its highly specialized nature. This may include calculation of electromagnetic fields everywhere or, perhaps a single scalar value like the total energy in the system or the impedance of a coil.

THE FINITE ELEMENT METHOD

A brief outline of the FEM is given below with special reference to the following boundary-value problem:

$$\nu_x \frac{\partial^2 A}{\partial x^2} + \nu_y \frac{\partial^2 A}{\partial y^2} + J = 0 \quad (x,y) \in G \qquad (1a)$$

$$A = A_0 \quad \text{on boundaries of } G \qquad (1b)$$

where ν_x, ν_y are material properties associated with the solution domain and J is the current in the domain. In the solution domain, G, the magnetic vector potential (MVP) A satisfies Poisson's equation, while on the boundary B of the solution region, the MVP or its first derivatives are known. Here, and throughout this work, reference is made to the magnetic vector potential. Equation 1 however, applies to a variety of physical quantities, vector or scalar. Time dependent problems may also be considered but for simplicity, these are not discussed here. Also, a 2-D equation is used. 3-D equations are treated similarly with a somewhat lengthier process and with larger arrays.

The boundary-value problem in Eq.(1) can be stated by the following variational problem:

$$F(A) = \int_G \left[\frac{1}{2}\left[\nu_x \left(\frac{\partial A}{\partial x}\right)^2 + \nu_y \left(\frac{\partial A}{\partial y}\right)^2\right] + J.A\right] dx\,dy \quad (x,y) \in G \qquad (2a)$$

$$A = A_0 \quad \text{on boundaries of } G \qquad (2b)$$

Minimization of F(A) yields the "best solution" to the original equation.

The case of known normal derivatives is not included in Eq. (2). This is a natural boundary conditions for the functional and need not be specified.

The solution region is divided into a number of "finite elements". Here, each element is a rectangle or a triangle. The elements are assumed to be interconnected at a number of nodal points

situated on their boundaries. The MVPs at these nodal points are the unknowns.

The MVP within each element can be defined using various approximations. There are two basic methods of defining these approximations. One is based on products of polynomials in a local system of coordinates while the other is based on a polynomial over the element, defined in the system of coordinates in which the problem is solved [1].

The two methods are quite different and will be outlined briefly below as this is necessary for their implementation. To illustrate the process we use the two simple elements in Fig. 1.

ISOPARAMETRIC FINITE ELEMENTS

The approximation used for the function A in the interior of the finite element in Fig. 1a is:

$$A(x,y)=\sum_{i=1}^{4} N_i A_i \qquad (3)$$

where N_i are a set of shape functions defined at the four nodes of the elemet and A_i are the values of the unknown function at the same nodes. Similarly, the derivatives of A can be taken as:

$$\frac{\partial A(x,y)}{\partial x}=\sum_{i=1}^{4} \frac{\partial N_i}{\partial x} A_i \qquad \frac{\partial A(x,y)}{\partial y}=\sum_{i=1}^{4} \frac{\partial N_i}{\partial y} A_i \qquad (4)$$

The functions N_i as well as their derivatives $\partial N_i/\partial x$, $\partial N_i/\partial y$ need to be known. The standard method for their calculation is to find them in a local system of coordinates where they are extremely simple and then to map them into the global system of coordinates. The derivatives with respect to x and y are calculated as:

$$\begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{bmatrix} = [J]^{-1} \begin{bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{bmatrix} \qquad [J] = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \qquad (5)$$

where [J] is the Jacobian matrix.

These are now substituted in Eq. (2). In order to find a solution to the problem, the functional in (2) is minimized by setting it's first derivatives with respect to each unknown to zero. This produces an algebraic equation for each unknown. Rather than doing this for the whole solution domain, it is done for each element separately and the contributions from separate elements are summed up in a global system of equations.

To produce the elemental contributions, the terms in Eq. (2) need to be evaluated. This is done numerically by Gaussian quadrature.

$$\sum_{i=1}^{2}\sum_{j=1}^{2} W_i . W_j . f'(\xi_i, \eta_j) \qquad (6)$$

where W_i, W_j are weights and ξ_i, η_j are quadrature points. This integration is done locally and then, using the shape functions, mapped to the global system of coordinates. For a simple element like this two points in each spatial direction are sufficient. For more complex elements, more points may be required.

Implementation on the MPP

Implementation begins by creating three arrays: SF1,SF2 and SF3 where SF1 holds the shape functions of the four nodes as four rows of four entries each. Each column holds the shape function for one quadrature point. Similarly, SF2 holds $\partial N_i/\partial \xi$ in four columns, each column corresponding to a quadrature point and each row to a nodal point of the element. SF3 has the same structure for $\partial N_j/\partial \eta$.

The first step in the computation of the elemental matrices is the calculation of the Jacobian (Eq. 5). It is stored in four arrays: RJAC1, RJAC2, RJAC3 and RJAC4. RJAC1 holds the first coefficient (RJAC(1,1) for 128 elements calculated in parallel. RJAC2 contains the second coefficient (RJAC(1,2)), RJAC3 contains RJAC(2,1) and RJAC4 contains RJAC(2,2). The determinant of the Jacobian is calculated and stored in array DETJ for all elements in parallel. The inverse of the Jacobian needed in Eq. (5) is now found for all 128 elements by a single divide of the RJAC arrays by the DETJ array. The results are placed back into the RJAC(i) arrays.

The derivatives of the shape functions with respect to x (Eq. (5)) are calculated by multiplying RJAC1 by the SF2 array, RJAC2 by the SF3 array and adding the results together. The calculation of the derivatives with respect to Y is found as the sum of RJAC3*SF3 and RJAC4*SF3.

The contributions to the elemental matrix in Eq. (6) are found by parallel array multiplication of each of the four DNDX(i) array by DNDX(j), including itself and adding these to the product of the DNDY(i)*DNDY(j) arrays. The material properties RXI (ν_x), RYI (ν_y) are taken from array MAT by shifting operations. From this, the 16 contributions to the elemental matrices are found and entered in a single array (RRR). Each element takes the first 16 rows on a single column.

The global equation assembly proceeds by moving each value in the elemental matrix into the global matrix.

To create the right hand side (RHS) of the system of equations, the same process is used. As is clear from Eq. (2), The current density J is multiplied by the shape functions in SF1 and integrated using Eq. (6). An elemental column vector is created and this is later put in a global RHS vector.

The method outlined can handle up to 128 elements in parallel. It is possible to extend this to any number but it requires handling multiple arrays. Since in a FEM analysis, most of the time is spent in the solution process this was not considered worth while. There are two problems associated with this method. One is the need for sequential insertion in the global matrix. The

second, is in the number of arrays required. Since the number of arrays is relatively large (about 20 for a four node element), the only elements that can be handled in this fashion are elements with a small number of nodes (3 to 6 nodes). Other types of elements need special considerations. This may seem limiting but most FEM programs use these elements. For more complex elements, elemental matrices can be combined in arrays, with fewer parallel matrices at each step. Alternatively, the method of storage in the following section is used.

DIRECT DEFINITION OF ELEMENTAL MATRICES

Another method for the definition of the elemental contributions is to assume that the distribution within the finite element is of the following form [1]

$$A(x,y)=ax+by+c \tag{7}$$

If this polynomial is written at the three nodes of the element in Fig. 1b, we get three equations in the unknowns a, b and c. By solving these equations we get:

$$A(x,y)=N_iA_i+N_jA_j+N_mA_m=\sum_{i=1}^{4}N_iA_i \tag{8}$$

where N_i, N_j, N_m are the shape functions given by

$$N_i(x,y)=(a_ix+b_iy+ci)/2D \tag{9a}$$
$$N_j(x,y)=(a_jx+b_jy+cj)/2D \tag{9b}$$
$$N_m(x,y)=(a_mx+b_my+cm)/2D \tag{9c}$$

where:

$$a_i=y_j-y_m \qquad a_j=y_m-y_i \qquad a_m=y_i-y_j \tag{10a}$$
$$b_i=x_m-x_j \qquad b_j=x_i-x_m \qquad b_m=x_j-x_i \tag{10b}$$
$$c_i=x_jy_m-x_my_j \qquad c_j=x_my_i-x_iy_m \qquad c_m=x_iy_j-x_jy_i \tag{10c}$$

and x_k, y_k are the x and y coordinates of node k, k=i,j,m. D is the area of element ijm.

The expressions in Eq.(9) are substituted into the functional F(A). Minimization of F(A) yields a characteristic equation for triangle ijm:

$$[K]_e\{A\}_e=\{F\}_e \qquad [K]_e=\begin{bmatrix}k_{ii} & k_{ij} & k_{im}\\ k_{ji} & k_{jj} & k_{jm}\\ k_{mi} & k_{mj} & k_{mm}\end{bmatrix} \qquad \{F\}_e=\begin{bmatrix}F_i\\ F_j\\ F_m\end{bmatrix} \tag{11}$$

where:

$$k_{ii}=(a_ia_i+b_ib_i)/4D \quad k_{ij}=(a_ia_j+b_ib_j)/4D$$
$$k_{im}=(a_ia_m+b_ib_m)/4D \quad k_{ji}=(a_ja_i+b_jb_i)/4D$$
$$k_{jj}=(a_ja_j+b_jb_j)/4D \quad k_{jm}=(a_ja_m+b_jb_m)/4D \tag{12}$$
$$k_{mi}=(a_ma_i+b_mb_i)/4D \quad k_{mj}=(a_ma_j+b_mb_j)/4D$$
$$k_{mm}=(a_ma_m+b_mb_m)/4D \quad F_i=F_j=F_m=D*J/3$$

A global equation in the form [K]{A}={F} is assembled by accumulating the contributions of all elemental matrices (Eq. (11).

Implementation on the MPP

The parallelization process can be divided into two phases. In the first phase, the elemental matrix is computed where the 9 coefficients are calculated in parallel. In the second phase, a number of elemental matrices are placed in the global matrix concurrently.

First, the elemental matrix $[K]_e$ in Eq.(11) is rearranged into the following form:

$$[K]_e=([K]_{e1}+[K]_{e2})/4D \tag{13}$$

where

$$[K]_{e1}=\begin{bmatrix}a_ia_i & a_ia_j & a_ia_m\\ a_ja_i & a_ja_j & a_ja_m\\ a_ma_i & a_ma_j & a_ma_m\end{bmatrix} \quad [K]_{e2}=\begin{bmatrix}b_ib_i & b_ib_j & b_ib_m\\ b_jb_i & b_jb_j & b_jb_m\\ b_mb_i & b_mb_j & b_mb_m\end{bmatrix} \tag{14}$$

The two matrices in (14) can be further represented as

$$[k]_{e1}=[P]([P])^t+[Q]([Q])^t \tag{15}$$

where

$$[P]=\begin{bmatrix}a_i & a_i & a_i\\ a_j & a_j & a_j\\ a_m & a_m & a_m\end{bmatrix} \qquad [Q]=\begin{bmatrix}b_i & b_i & b_i\\ b_j & b_j & b_j\\ b_m & b_m & b_m\end{bmatrix} \tag{16}$$

Based on the expression in Eq.(10), [P] and [Q] are now represented as:

$$[P]=[P1]-[P2] \qquad [Q]=[Q1]-[Q2] \tag{17}$$

where [P1], [P2], [Q1] and [Q2] are the two parts of the expressions in Eq. 10a and 10b.

Special consideration is given to the calculation of D, the area of triangle ijm. In order to perform the divide operation in Eq. (13), an array, [D] is created

$$[D]=2([R]-[S]) \tag{18}$$

where

$$[R]=\begin{bmatrix}a_ib_j & a_ib_j & a_ib_j\\ a_ib_j & a_ib_j & a_ib_j\\ a_ib_j & a_ib_j & a_ib_j\end{bmatrix} \qquad [S]=\begin{bmatrix}a_jb_i & a_jb_i & a_jb_i\\ a_jb_i & a_jb_i & a_jb_i\\ a_jb_i & a_jb_i & a_jb_i\end{bmatrix} \tag{19}$$

These two arrays can be further represented in terms of the coordinates of the three nodal points:

$$[R]=([R2]-[R3])([S1]-[S3]) \tag{20a}$$
$$[S]=([R3]-[R1])([S3]-[S2]) \tag{20b}$$

where [S1] has x_i propagated in all nine locations and [R1] has y_i propagated throughout. Similarly, [S2] and [R2] contain x_j and y_j respectively and [S3] and [R3] contain x_m and y_m propagated throughout.

Finally, the elemental matrix of triangle ijm is obtained by performing one array divide operation:

$$[K]_e=([K]_{e1}+[K]_{e2})/D \tag{21}$$

At this point, the calculation of the elemental matrix has been parallelized. However, for the element used here, each array operation involves only 9 coefficients. A number of elemental matrices

can be computed concurrently by creating a whole plane or nearly a whole plane of data before any array operation is carried out. As an example a rectangular mesh with 210 triangular elements is used (Fig. 2). The mesh can be partitioned into 8 sets of elements that have mutually exclusive nodes as shown. All nodal coordinates of elements in this set are placed in [P1], [P2], [Q1], [Q2] in Eq.(17), and [S1], [S2], [S3], [R1], [R2], [R3] in Eq.(20) to form nearly a whole plane of data. These elements can be assembled in parallel. Once the elements in a set have been assembled, a new set is treated until all elements have been assembled.

A Parallel Pascal code has been developed based on the above parallel algorithm and applied to perform global equation assembly on the MPP for the finite element mesh shown in Fig. 2. The total processing time is 114.47 ms including the time needed for local nodal numbering. A larger portion of the total processing time has been spent on forming whole planes of data. Since the x and y coordinates are stored in two arrays, considerable use of fast row and column propagation routines has been made. Thus, the assembly routines are not particularly efficient for elements with few nodal points.

There are some important elements, such as 3D solid and shell elements, etc., which are extremely complex and require considerable computer resources. For these elements, parallel assembly of the equations is a significant step towards improving solution times and, in some cases (boundary integral elements) may be more significant than the solution of the system of equations. For these elements, the method outlined becomes more efficient as the number of nodes per element approaches 128.

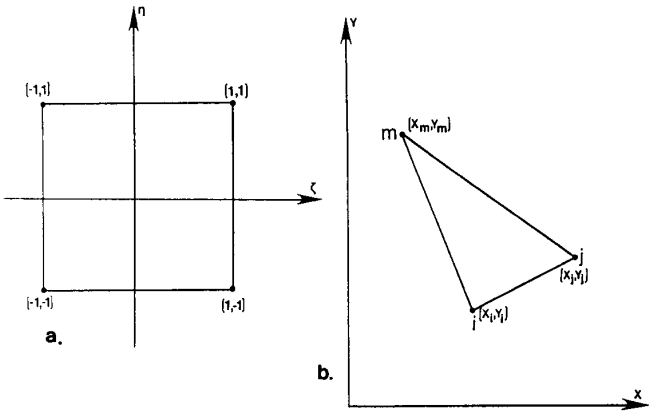

Figure 1. Two finite elements. a. Isoparametric element defined in a local system. b. Triangular element defined in a global system of coordinates.

Table 1. Processing time on the MPP for matrix assembly for different finite element meshes. Time is in seconds

No. of Equat.	No. of Elements	Band-width	Time on the MPP
128	93	63	0.183
256	189	127	1.296
512	441	127	3.488
1024	889	255	14.363

CONCLUSIONS

Two methods for assembly of systems of equations arising from finite element analysis have been presented. One is particularly suited for elements with few nodal points while the other is for directly defined finite elements. The time involved in assembly is not considered to be significant compared with the time needed for solution.

REFERENCES

[1] O. C. Zienkiewicz, The Finite Element Method in Engineering, third edition, McGraw-Hill Book Co., London, 1977.

[2] J. S. Wang and N. Ida, "Parallel algorithms for direct solution of large systems of equations", these proceedings.

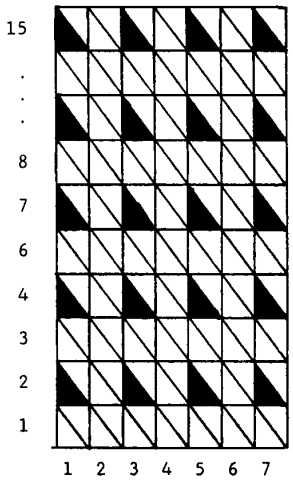

Figure 2. Elements in a mesh with mutually exclussive nodes.