Three-Dimensional Mesh Improvement Using Self Organizing Feature Maps

Derek Dyck*, David A.Lowther*, Wolfgang Mai**, Gerhardt Henneberger**
*Infolytica Corporation, Montreal, Quebec, Canada
**Institut fur Elektrische Maschinen, RWTH Aachen, D-52056 Aachen, Germany

Abstract— **A method of adaptation in the solution of three dimensional electromagnetic field problems is described. The approach is based on using a self organizing feature map to redistribute the nodes rather than generating new ones on each adaptive pass.**

***Index Terms*—Self-organizing feature maps, Finite element meshes, Neural networks, Adaptive systems.**

I. Introduction

Over the last decade, the state-of-the-art in the simulation of electromagnetic devices has moved rapidly from simple two-dimensional models to a point at which full three-dimensional analysis is becoming common place. Much of this transition has occurred because of the developments in computer technology resulting in a great amount of computational power being available at a relatively low price. As a consequence, it is now not unreasonable to consider obtaining the solutions of problems that may need several hundred megabytes of main memory on a desktop personal computer and, given current processor speeds, these analyses can be performed in times of the orders of minutes or hours. Probably the most common method now being applied to the solution of electromagnetic field problems is that based on the finite element approach. Such a technique requires that the entire problem space be divided into elementary blocks, usually tetrahedra or hexahedra in three-dimensions.

The accuracy of the solution obtained by this approach is highly dependent on the distribution and size of the elements - in general, a fine mesh is required wherever the field has a large variation and a coarser mesh is used elsewhere. The problem facing users of these methods is that the correct mesh for a particular analysis depends on the field variation through the device. However, the field variation is not known a priori and thus the ideal mesh cannot be determined.

There are two commonly used approaches to solving this problem. The first is to over-discretize the entire solution domain. This attempts to guarantee that the mesh is fine enough in the critical areas but has the disadvantage that there are a large number of elements in the areas where the field is hardly varying. Thus, while an accurate solution may be obtained, the cost in terms of cpu time and memory is extremely high. The second approach is to use an adaptive system in which the solution domain is initially discretized with as coarse a mesh as can reasonably be used. From the initial solution, an error map is derived and used to refine the mesh in parts of the problem to improve the solution accuracy. This process is repeated until the overall error is reduced to some pre-specified level. In this case, the final mesh is kept as small as possible thus minimizing the amount of memory needed. However, each adaptive steps adds more degrees of freedom, thus increasing the solution time. It should be noted that elements are never removed from the mesh in this system.

Both of these systems can result in the use of more degrees of freedom than a really necessary for the accurate solution of a problem. Basically, the nodes suporting the solution are not necessarily placed in the most useful positions. The ideal goal of a mesh generator should be to provide the best solution possible for a given number of nodes and this means that the position of each node should be optimized in some way. It is the intention of this paper to suggest one approach to achieving this goal of optimal node positioning. The approach to be used is based on the properties of *self organizing feature maps.*

II Self Organizing Feature Maps and Mesh Generation

Self organizing feature maps (SOFMs or Kohonen networks) [1] have been discussed in earlier papers as an approach to finite element mesh generation [2], [3], [4]. In reality, the method is not a true mesh generator - it is rather a *mesh re-organizer* and, when used in this way, belongs to the standard class of mapping mesh generators. In such an approach, an initial regular mesh is created based on a three-dimensional array of points. This forms a topological structure which ensures the correctness of the mesh. A mapping is then applied to distort the topological structure onto the desired geometric structure. In the case of a SOFM, this mapping varies from node to node. In the general case, there is no reason why the initial mesh should be a regular array of nodes and elements - it merely has to set up a valid topology. Thus the first step in the algorithm to be described is a conventional three-dimensional mesh generator.

The SOFM then attempts to reposition the nodes in the mesh based on a local estimate of the error in the solution. This error is used to generate a set of training points the density of which is directly related to the final mesh density. Note that these points are not nodes of the final mesh, they are purely guidance for the mesh training. In the process of

Manuscript received June 3, 1998
Email: lowther@infolytica.qc.ca

0018-9464/99$10.00 © 1999 IEEE

Authorized licensed use limited to: Universitaetsbibliothek der RWTH Aachen. Downloaded on August 07,2020 at 11:07:27 UTC from IEEE Xplore. Restrictions apply.

training, the nodes of the mesh are moved from an initial distribution while the interconnections between nodes are maintained. Thus the final mesh can be guaranteed topologically correct. The terms training and learning are often used with SOFMs because of their close relationship with conventional neural networks. The method by which the nodes are moved in space is referred to as *the learning algorithm.*

A. The Learning Algorithm

In a previous paper [4], the basic learning algorithm for a two-dimensional SOFM has been described. The approach in three-dimensions is the logical extension of the same algorithm. Essentially, a set of training points is generated based on the desired mesh density distribution. Each training point is presented to the mesh and the node closest to the training point is identified. This node is moved towards the training point. In addition, all the nodes connected to the nearest node, the "neighbourhood" are moved towards the training point. The size of the neighbourhood is defined based on the topological radius from the nearest node. Generally, only one or two radii are considered. The process is repeated for each training point and results in the mesh being "dragged" towards the highest density of training points.

The updating rule is:

$$w_{ik} = w_{ik}(old) + \alpha(e).(x_{tk} - w_{ik}(old)) \qquad (1)$$

where x_{tk} is the coordinate vector of the training point,t, w_{ik} the weights of the considered node (i.e. the current position of the node) and $\alpha(e)$ is a deceleration factor; k is the component of the coordinate vector. The period during which all the training points are presented to the mesh is known as an "epoch". The deceleration factor is reduced with each epoch.

III An Adaptive System

As stated above, the basic SOFM approach in three-dimensions is basically that previously described for two-dimensional situations. However, whereas the previous paper described a two-dimensional system where a single distribution of training points was provided to the mesh to cause the mesh re-organization, the intention here is to use the re-organization as a form of adaptation. In this case, the training point distribution being used is not based on any pre-conceived ideal mesh density but rather on the current error levels in the solution. Thus the approach is seen to be a useful complement to a conventional advancing front or Delaunay based mesh generator.

The node moving feature of the SOFM means that nodes will be moved into those areas where a more refined mesh is needed and, at the same time, removed from areas where the mesh could be made coarser. The overall result is to adapt the mesh without the addition of extra nodes. Of course, it may not be possible to achieve the required accuracy without the addition of any new nodes and thus new nodes can be added if needed. In fact, a complete system alternates between the two techniques leading to an improved mesh with a minimum number of degrees of freedom for a required accuracy. The overall result is a reduction in both the memory required and the cpu time taken.

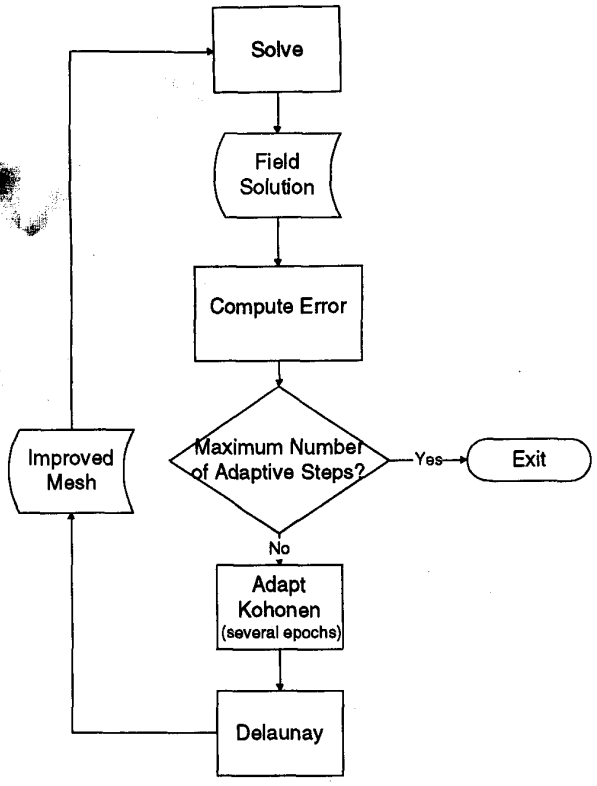

Fig. 1. Flow chart for mesh re-organization.

A. Error Calculations

The set of training points to be used for moving nodes is based on a local error criterion. In the system described below, the local error was determined from the error in the divergence of B within an element and this is manifested by a charge on the element faces.

The process is then one of computing the total error within the mesh by summing over all the tetrahedra. Next, the total number of training points to be used is set and a training point "density" found by dividing by the total error. Within any one tetrahedron, the global density is multiplied by the error in the tetrahedron and the volume of the element. This gives a total number of training points in a particular tetrahedron and these are placed randomly within the element.

Authorized licensed use limited to: Universitaetsbibliothek der RWTH Aachen. Downloaded on August 07,2020 at 11:07:27 UTC from IEEE Xplore. Restrictions apply.

B. Method

Once a set of training points have been established, the learning algorithm is applied. The neighbourhood radius to be considered for each training point is limited to 1 since the size can grow rapidly in three-dimensions - at R=2 the neighbourhood can consist of over 100 nodes. Within the context of an adaptive system, there is little point in running very many epochs. Instead, only two epochs are used with α being reduced from 0.01 to 0.005 from the first to the second epoch. It should be noted at this point that the node moving algorithm requires that nodes on the surfaces of bodies remain on those surfaces and similarly for edges. Thus there is a constraint on the movement of those nodes.

Once two epochs have been considered, a Delaunay pass is applied to the mesh to improve the quality of the elements and a new solution is calculated.

The process is repeated until the maximum number of steps is reached. The complete process is shown in Fig. 1.

IV. Results

The method described above has been implemented in an experimental three-dimensional finite element analysis code intended to solve bounded high frequency problems. Two different geometries have been considered. The first is a simple resonator system, the second a miter bend waveguide.

A. The Resonator

The resonator consists of a metallic cylinder inside a metallic box, Fig. 2.. The goal of the analysis is to compute the resonance frequency of the structure.

The initial mesh was created using an extrusion process and resulted in a very uniform element distribution throughout the device. This is shown in Fig. 3. The adaptive

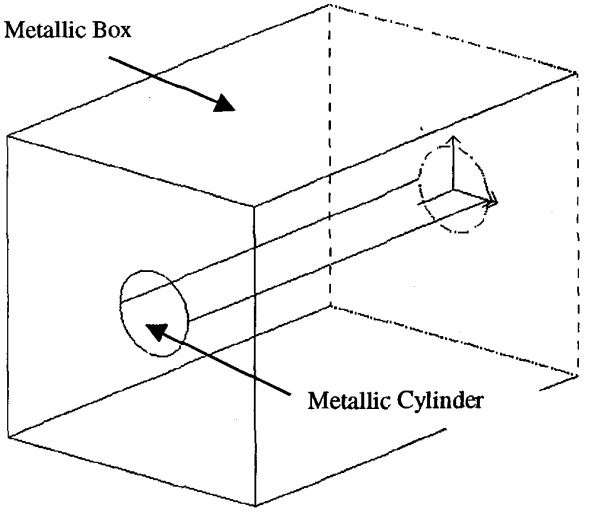

Fig. 2. Wireframe drawing of the resonator.

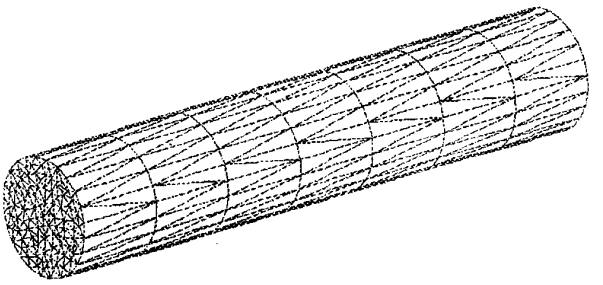

Fig. 3. Resonator initial mesh.

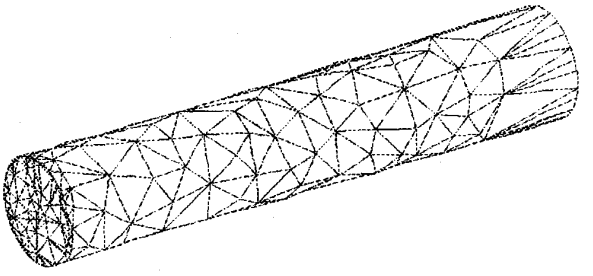

Fig. 4. Resonator final mesh.

process was applied for seven steps using two epochs with the SOFM on each step. The number of training points generated for the SOFM was set to be 10 times the number of nodes in the mesh. A neighbourhood of R=1 was used in the first epoch and R=0 for the second. The final mesh is shown in Fig. 4.

The figures show only the surface meshes in both cases and the number of nodes on the surface in both cases is the same. The internal re-organization of the mesh follows the surface change. As can be seen, the system has moved the nodes towards the end of the cylinder and, on the end face, the nodes (and elements) have been moved towards the edge. Note that the total number of nodes in the two meshes is identical. The improvement in the field solution is demonstrated in Table I where the maximum, minimum and average errors in the field are given. The error values are normalized to the average flux density value in the solution and the system has considerably reduced the maximum error (by moving the nodes to the edge of the cylinder). However, the average error has reduced only by a small amount - this

TABLE I.
ERROR IMPROVEMENT FOR RESONATOR

	Initial Mesh	Final Mesh
Maximum Error	269%	206%
Minimum Error	0.0036%	0.0006%
Average Error	7.53%	7.15%

Authorized licensed use limited to: Universitaetsbibliothek der RWTH Aachen. Downloaded on August 07,2020 at 11:07:27 UTC from IEEE Xplore. Restrictions apply.

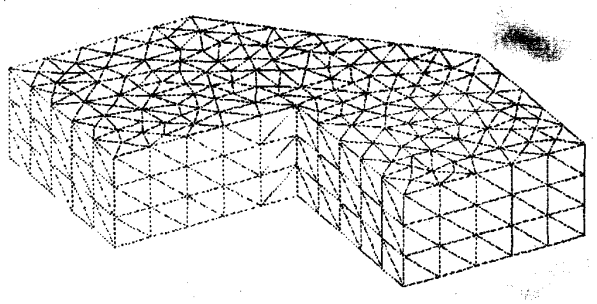

Fig. 5. Initial mesh for the miter bend.

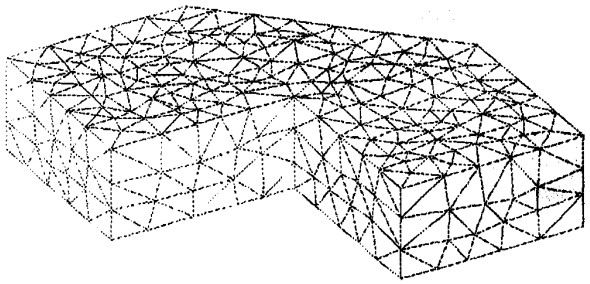

Fig. 6. Final mesh for the miter bend.

is due to the fact that the edge is only a small proportion of the total volume of the problem.

As a comparison, the resonator was also solved using a conventional mesh refinement (h-type) adaptive system using the same error criteria. In each refinement step, the 5% worst tetrahedra (in terms of the error) were refined. The results are shown in Table II in terms of the error in the resonant frequency. As can be seen, the mesh re-organization process provides a major gain in accuracy for a given computational cost. While the mesh refinement system can give better results, it does so at a massive increase in computational cost.

B. The Miter Bend

The second geometry is a basic miter bend in a waveguide. The initial and final meshes are shown in Fig. 5 and Fig.6. The same values were used for the SOFM parameters as in the resonator case. In this problem, the desired outputs were the S parameters for the model. In total 4 passes through the adaptive system were made. Although

TABLE II.
COMPARISON OF PERFORMANCE BETWEEN SOFM AND REFINEMENT FOR RESONATOR

Method	No. of tets	Freq (GHz)	Error
Initial	14860	1.8204	2.166%
SOFM	14860	1.8449	0.848%
5% refined	20852	1.8565	0.227%
10% refined	32313	1.8607	0.000%

TABLE III
ERROR IMPROVEMENT FOR MITER BEND.

	Initial Mesh	Final Mesh
Maximum Error	105%	60%
Minimum Error	0.34%	0.24%
Average Error	13.7%	10.7%

the change in the mesh structure is much less than in the resonator case, the small movements in the node positions actually had a major effect on the errors as can be seen in Table III.

V. CONCLUSIONS

The use of a self-organizing feature map as a method of mesh adaptation has been described. The intention is to produce an adaptive system which can improve the accuracy of a solution without an increase in the number of nodes being used. This leads to a reduction in both the amount of memory required and the cpu times taken to achieve a required error level. Even with the explosive growth in computational capabilities, this is important especially when the solutions of three-dimensional structures either in the high frequency domain or for low frequency induced current problems. In both cases, full vector solutions become necessary and the number of unknowns expands rapidly with the number of nodes.

Of course, for a given number of nodes in the mesh, there is a limit to the minimum error which may be achieved and, at some point, if a further reduction in error is required, there is little choice but to refine the mesh with the addition of new nodes or an increase in the polynomial order. Thus a realistic system would probably use a combination of conventional h or p adaptation coupled with the scheme described here.

The main disadvantage of the SOFM approach is the time taken to "train" the mesh. Depending on the number of nodes inserted in a typical h refinement step, the cpu costs of re-organizing compared to node insertion plus solution may be very similar. However, the gain in memory requirements may result in an overall faster solution time if the entire problem can be fitted into main memory, thus removing the requirement to use virtual memory.

REFERENCES

[1] T.Kohonen, *Self-organizing and associative memory*, 3rd ed., Springer-Verlag, Berlin, 1989.

[2] C.H.Ahn, S.S.Lee, H.J.Lee, S.Y.Lee, "A self-organizing neural network approach for automatic mesh generation," *IEEE Transactions on Magnetics*, Vol. 27, pp. 4201-4204, September 1991.

[3] J.Pechoux, D.A.Lowther, "Kohonen Maps and Automatic Mesh Generation in Two and Three Dimensions," *Proceedings of the 7th International IGTE Symposium*, Graz, Austria, pp. 206-211, September 1996.

[4] W.Mai, D.A.Lowther, "On Automatic Mesh Generation using Kohonen Maps," *IEEE Transactions on Magnetics*, Vol. 34, pp. 3391-3394, September 1998.

Authorized licensed use limited to: Universitaetsbibliothek der RWTH Aachen. Downloaded on August 07,2020 at 11:07:27 UTC from IEEE Xplore. Restrictions apply.