AI and NN Tools for the Finite Element Method: Self-Organizing Mesh Creation and Heuristic Node Numbering in Meshes

by

Larry Manevitz*, **Dan Givoli****, **Micha Margi*** and **Malik Yousef***

* Department of Mathematics and Computer Science, Haifa University, Haifa 31905, Israel.
** Department of Aerospace Engineering, Technion — Israel Institute of Technology, Haifa 32000, Israel.

Abstract

We report on two on-going inter-related projects that tie together tools of artificial intelligence and the finite element method.

I. A Hybrid Neural Network- Expert System for the generation of "balanced" meshes for use in the finite element method. II. An Expert System for the efficient numbering of finite element nodes.

In part I we describe the design of the system that assigns the geometry to a topological mesh in a way appropriate for efficient use of the finite element method. This method uses a hybrid expert system-neural network. The system first decides based on geometric and physical considerations on an appropriate density function that corresponds to "regions of interest" in the body to be analyzed by the FEM and to decide on the kinds, number and topology of the finite elements and their mesh. (This is currently done manually in our system.) Then the actual placement of the mesh is performed by a Kohonen-style self organizing neural network by identifying the physical geometry with the weight space and using the density function to generate sampled input. The constraints of the problem require a novel mixing of different dimensional self-organizing maps. This mechanism allows for arbitrary topology of the meshes (e.g. quadrilateral as well as triangular). Comparison studies with a popular mesh generating package, PLTMG, shows an advantage to the NN approach.

In part II we describe the early version of an expert system, employed in numbering the nodes of a given finite element mesh. The numbering is performed in such a way as to minimize the average bandwidth of the finite element stiffness matrix. The numbering procedure is heuristic and is designed to mimic the methods of a human expert numberer. The performance of the expert system is demonstrated by means of a number of two-dimensional examples.

Introduction

The finite element method (FEM) has become during the last three decades a major and very widely used tool for the numerical solution of partial differential equations. This is especially (but not exclusively) true in structural engineering and solid continuum mechanics. The finite element procedure has been proved most effective when applied to linear boundary value problems in spatial domains which are complicated geometrically. (See e.g. Hughes [1].)

It is well appreciated among finite element users in industrial and scientific communities that the successful application of this technique requires a fair amount of experience and expertise in order to make the computation feasible and the results accurate at the same time. This is true with regard to any of the large commercial finite element packages available currently, although various codes may have a different amount of flexibility. The user expertise is needed because applying the FEM to a problem requires making choices, such as the choice of mesh and the type of elements to be used, for which there are no known effective algorithms that can replace a human user in all cases.

Here we report on our approach to two of the processes in this category: (i) an effective generation of a mesh over a given body (ii) the optimal numbering of nodes.

When applied to linear boundary value problems, the finite element discretization of the governing partial differential equations leads finally to a linear system of algebraic equations (see Hughes [1]),

$$\mathbf{Kd} = \mathbf{F}\ . \tag{1}$$

Here $\mathbf{K}$ is the so-called *stiffness matrix*, $\mathbf{F}$ is the load vector, and $\mathbf{d}$ is the vector of unknowns. The dimension of the system (1) in typical industrial applications is of order 100 or 1000 or even 10000. The computational effort involved in solving such a large system is reduced considerably owing to an important property which the matrix $\mathbf{K}$ possesses: it is *sparse*, i.e. most entries K_{ij} are zero.

The degree of sparseness of $\mathbf{K}$ depends on the specific finite element discretization of the spatial domain, or more explicitly on the *finite element mesh.* The spatial domain is divided into *elements*, which are connected to each other by *nodes*. In two dimensions, the elements may be quadrilateral or triangular, and the nodes are their vertices. Points on the edges and in the interior of the elements may also be defined as nodes.

Two nodes are said to *interact* if they both belong to a common element in the mesh. If there is a total number of N nodes in the mesh and they are numbered from 1 to N, then $K_{ij} \neq 0$ only if nodes i and j interact. Thus the structure (or skyline; see below) of the stiffness matrix is determined by the node interaction. Since the mesh is typically imposed on a physical domain and thus can be thought of as a planar or spatial graph, most K_{ij} are zero, i.e. $\mathbf{K}$ is sparse.

PART I

Given a physical body, one has to place a mesh on it. This requires defining a triangulation or quadrilateralization of the surface. It is known that in theory for many applications quadrilaterization should give better approximations when running the FEM; however, the methods of placing the mesh are more developed for the triangularization method. Accordingly, we decided to concentrate on the quadrilateralization and on methods which will allow for the use of a mesh containing a mixture of both quadrilaterals and triangles.

In addition when looking at the physical body, there are points and regions which one wants to cover with a finer mesh than other areas. These are regions or points of "interest" where the linear approximation is intrinsically worse. Essentially it is an optimization problem; given a fixed amount of computation that one wants to expend, which translates into a fixed amount of elements; how should one best distribute these elements so as to obtain the best approximation using the finite element method. A better mesh results in a better approximation.

There are several requirements for the quality of a mesh. For example, one wants the proportions of the elements to be as close as possible to those with good aspect ratios (the aspect ratio is the ratio of the radii of circumscribed circle to that of the inscribed circle; "good" means here close to one). For a quadrilateral then, for the best results from the finite element method one wants quadrilaterals close to squares and triangles close to equilaterals. In addition, one wants the change in size between elements determined by the mesh to be gradual. Quantitatively, one wants to keep the ratio of radii of circumscribed circles of adjacent elements to be close to one; globally that means that the maximum and minimum of such ratios of all pairs of adjacent elements should be as close to one as possible.

All in all, then this is a rather complicated global optimization problem.

Our approach was to split this problem into several parts. On the one hand, there is the decision of the size of the mesh and the appropriate densities in different regions of the body; while on the other hand, one has to realize the mesh by specific geometric coordinates on the body. The first part should be accomplished by an expert system which will, based on geometric and physical considerations, decide on the regions of interest and density distribution needed of the elements (this has not been implemented in our work); while the second part is realized by a self-organizing neural network.

In a typical finite element mesh, the density of the nodes is taken to vary from being very high near certain critical regions or points to more sparse where the approximation can be taken more grossly as linear approximation. This can be handled using the neural network approach by setting the probability density function of the input to correspond to the desired density of the network. The "self-organizing" feature of the Kohonen map then places the highest density of nodes according to the sampling of data according to this density function. Thus the meshing is handled in principle by (i) using an expert system (or other method) to extract the needed information to design the appropriate density function (ii) deciding on the number and kind of finite elements desired as well as the topology of the corresponding mesh, (iii) then allow the elements to find their own coordinates via the Kohonen self organizing network by identifying the block with weight space and selecting input data according to the density function selected in step (i).

We point out that what the Kohonen network does, is in essence form a model for the density function of the weights of the training data [19] [20]. We identify the nodes of a mesh with the neurons of a neural network with the connectivity of the network corresponding to the edges of the mesh. Then, by identifying the physical body with the weight space, and then using sampling data from the original body according to a probability density corresponding to the "degree of interest" of a region, then the Kohonen competitive learning mechanism automatically places the mesh in a positions matching the density function as best it can, when constrained by the interconnectivity. This should in principle result in a mesh of good quality.

However, it turns out that such a straight-forward adaptation of the Kohonen algorithm has some difficulties in this context which required some sophistications to the algorithm. That is: (i) a finite element mesh has to fit exactly inside the domain and reach the boundaries; (ii) computational requirements are somewhat high; (iii) non-convex domains require special techniques.

Our implementation dealt with these problems with the following techniques: (i) The algorithm actually uses an interweaving of a two-dimensional Kohonen algorithm on the mesh with a one-dimensional Kohonen algorithm on each connnected component of the boundary. Note that this suggests a generalization to higher dimensions. (ii) We used an adaptation to the Kohonen algorithm suggested by Tzvi and Iaakov [2] which resulted in an increase in speed of 75% without degradation of performance. (iii) A special modification to the algorithm was made to handel non-convex domains.

To test the results, we solved a battery of partial differential equations with boundary conditions on a variety of domains. Each problem had an analytic solution so that the exact errors could be calculated. The problems were run, for comparison, on a popular professional mesh generator PLTMG. Results showed a substantial advantage to the neural network approach. See [16] for full details.

PART II

The order in which the nodes are numbered is crucial, since this numbering determines how sparse **K** is. This part of our work attacks the problem of finding a good numbering system.

Note first that optimal numbering of a given mesh is an NP complete problem (see [18]). Thus for a mesh containing a large number of nodes and elements it is of course impractical to find an exact solution for the optimization problem by performing a full enumeration. However, there are so-called "algorithmic" solutions which have been incorporated in commercial software. Anecdotal evidence about this software reports that: (i) a human expert can often outperform it (ii) The best such algorithms typically result in rather complicated numberings involving substantial "windings" and for some applications there is an advantage in the simplicity of the numbering. (These algorithms include the methods described by Cuthill and McKee [6], King [7], Collins [8], Akin and Pardue [9], Gibbs [10], Razzaque [11], Pina [12], Sloan and Randolph [5], Fenves and Law [13] and Sloan [14].)

According, we decided to develop an expert system to try and mimic the heuristic mechanisms of a proficient human numberer. So far, only some of the heuristics have been implemented but already the expert system competes favorably with the human user on the examples we have tested.

In order to state the optimization problem mathematically we introduce some notation. Let the dimension of the matrix $\mathbf{K}$ be $N \times N$. Let b_i be defined as

$$b_i = i + 1 - \max_{K_{ji} \neq 0} j \ . \tag{2}$$

In other words, b_i is the height of column i starting from the diagonal and up to the skyline. Now, let the *average half bandwidth* AHB and the *root-mean-square bandwidth* RMSB be defined as

$$\mathrm{AHB} = \frac{1}{N} \sum_{i=1}^{N} b_i \ , \tag{3}$$

$$\mathrm{RMSB} = \left(\frac{1}{N} \sum_{i=1}^{N} b_i^2 \right)^{1/2} \ . \tag{4}$$

It is apparent that the bandwidth depends on the nodal numbering system. This dependence becomes strong for large meshes. The optimization problem under consideration can thus be stated as follows: Find a numbering system for the nodes (from 1 to N) such that AHB (or RMSB) will be *minimal.*

Description of the Expert System Characteristics

One of the hardest steps in producing an expert system is to construct the heuristics according to which the expert system will perform, and which will mimic the considerations of a human expert. To this end, we note the following functions typically performed when numbering the nodes of a finite element mesh manually and in this abstract, briefly comment on our system's approach to these tasks.

0. Preparation of numbering strategies for "paradigm simple blocks" with different geometries and topologies. Varying parameters of freedom are associated with each strategy. These strategies are achieved by experience, by trial and error, and sometimes by full analysis. This step is not "performed" for each mesh separately, but is rather a data base of knowledge an expert has accumulated. Examples of simple blocks are rectangles, annuli, discs.
1. Subdivision of the mesh into a disjoint union of simple blocks. Our system does not perform this, but receives it as given. (See Part I for further discussion on this.)
2. Choosing the order in which these blocks are picked for numbering.

In this step the order of the simple blocks is determined. To optimize the numbering a user tries to keep the node numbers as continuous as possible. However when passing from block to block this is often impossible. Thus the goal is to keep the "jumps" across block interfaces as small as possible. In the discussion that follows, a *component* is a topological component, i.e. a maximal connected sub-body.

In the implemented system the procedure for block ordering is as follows:

- Choose the first block to be numbered as the largest simple block, based on the number of nodes.
- Remove this simple block from the mesh. (This may cause the remaining mesh to have several disconnected components.)
- Place all resulting separate components in increasing order in a stack.
- Until the stack is empty do the following:
  - Remove the smallest component from the stack.
  - From the chosen component, choose the next simple block to be numbered in the following way:

    - Consider all simple blocks in the component with an interface with the previously numbered block (there are always such).
    - Choose the one with the largest interface.
  - Remove the chosen block from the component. (This may divide the remaining blocks into several connected components.)
  - Place the new components on the stack in increasing order.
3. For each block, choosing a numbering strategy, depending on the topology and the geometry of the block. That is, choosing the best match to a paradigm simple block. Essentially this has two parts: (1) solution of a pattern recognition problem; i.e. given a simple block find which of the paradigm simple blocks it is closest to; (2) after identification choosing a numbering method for the block form amongst the possible methods mentioned in step 0.

In the implemented version, all the blocks are assumed to be pseudo-rectangles and there is only one numbering strategy for numbering rectangles, so this step is to a large extent vacuous.

Notwithstanding, it remains for the system to perform the following analysis: (a) to decide on the orientation of the block, regarded as having a rectangular shape; (b) to be able to ignore small perturbations. The first is done using an algorithm involving the convex hull of the block and regression; the second is accomplished using a heuristic (also involving the convex hull) that defines a "grain size".

4. For each block, determining the free parameters associated with the strategy chosen for that block, such as the node from which to start the numbering. For the rectangular strategy which is handled in the implemented version, the only free parameter is the node from which to start the numbering, and there are four possibilities, corresponding to the four corners of the pseudo-rectangle. We call this starting point n^*, and it is selected by a heuristic that is based on both topological and geometrical considerations.

It is interesting and crucial to note that while the mesh is a purely topological object as far as node numbering is concerned, the user decides on the division based not only on topological criteria, but also on *geometric* criteria. This is an obvious logical fallacy and it is easy to devise artificial mesh examples where the results are extremely bad. Nonetheless for real meshes the results seem acceptable.

The reason why human users can be so successful in numbering meshes although basing their decisions on geometric considerations is that they intuitively rely on the fact that *actual meshes are designed with implicit geometric constraints*. For example, a good practice in mesh design is to use elements which have aspect-ratios close to unity, namely quadrilaterals and triangles which are nearly equilateral. Another good practice is to pass from a crude region (large elements) to a refined region (small elements) in a gradual manner. (See e.g. Irons and Ahmad [4].)

Our expert system tries to mimic a human user; therefore it will make geometric considerations as well as topological ones. This is in contrast to the algorithmic approach, where only the topological properties of the mesh play a part in the node reordering procedure.

The system was tested on a number of meshes taken from the literature; here we mention the largest mesh tested to date similar to one that appears in [3] which has 359 nodes and 559 triangular elements. Here the average half band width of the system was 17.6; that of the human expert was 15.9 while for comparison a random numbering resulted in a value of 210.2.

Some other meshes were taken from applications in large deformation continuum mechanics discussed in [17] and solved by the finite element code NIKE2D. The results of the system are quite comparable to that of the human expert. (In one case the expert system actually outperformed the human expert.) See [15] for details.

The results show that although the human expert performs better than the expert system, the differences between the two AHBs are reasonably small. Other examples have been tested as well and the results compared favorably with those of a human expert.

References:

1. T.J.R. Hughes, *The Finite Element Method*, Prentice-Hall, Englewood Cliffs, New Jersey, 1987.
2. T. Tzvi and E. Iaakov. Towards real-time self-organizing network sith parallel and noisy inputs. Preprint, Hebrew University, 1994.

3. O.C. Zienkiewicz, *The Finite Element Method*, 3rd ed., McGraw-Hill, New York, 1977.
4. B. Irons and S. Ahmad, *Techniques of Finite Elements*, Ellis Horwood and John Wiley, Chichester, England, 1980.
5. Sloan and Randolph, "Automatic Element Reordering for Finite Element Analysis with Frontal Schemes," *Int. J. Numer. Meth. Eng.*, 19,pp. 1153–1181, 1983.
6. E. Cuthill and J. McKee, "Reducing the Bandwidth of Sparse Symmetric Matrices," *Proc. ACM National Conf.*, Assoc. for Computing Machinery, New York, 1969.
7. I.P. King, "An Automatic Reordering Scheme for Simultaneous Equations Derived from Network Systems," *Int. J. Numer. Meth. Eng.*, 2, pp. 523–533, 1970.
8. R.J. Collins, "Bandwidth Reduction by Automatic Renumbering," *Int. J. Numer. Meth. Eng.*, 6, pp. 345–356, 1973.
9. J.E. Akin and R.M. Pardue, "Element Resequencing for Frontal Solutions," in J.R. Whiteman (ed.), *Mathematics of Finite Elements and Applications*, Academic Press, New York, 1975.
10. N.E. Gibbs, "A Hybrid Profile Reduction Algorithm," *ACM Trans. Math. Software*, 2, pp. 378–387, 1976.
11. A. Razzaque, "Automatic Reduction of Frontwidth for Finite Element Analysis," *Int. J. Numer. Meth. Eng.*, 15, pp. 1315–1324, 1980.
12. H.L. Pina, "An Algorithm for Frontwidth Reduction," *Int. J. Numer. Meth. Eng.*, 17, pp. 1539–1546, 1981.
13. S.J. Fenves and K.H. Law, "A Two-Step Approach to Finite Element Ordering," *Int. J. Numer. Meth. Eng.*, 19, pp. 891–911, 1983.
14. S.W. Sloan, "An Algorithm for Profile and Wavefront Reduction of Sparse Matrices," *Int. J. Numer. Meth. Eng.*, 23, pp. 239–251, 1986.
15. L. Manevitz, D. Givoli and M. Margi, "Heuristic finite element node numbering," *Computing Systems in Engineering*, 4, pp. 159-168, 1991.
16. L. Manevitz, M. Yousef and D. Givoli, "Finite element mesh generation using self-organizing neural networks", *Microcomputers in Civil Engineering*, to appear.
17. J.O. Hallquist, "NIKE2D — a vectorized, implicit, finite deformation, finite element code for analyzing the static and dynamic response of 2-D solids," University of California, UCID-19677, Rev. 1.
18. M.R. Garey, D.S. Johnson, *Computers and Intractability*, W.H.Freeman Press, San Francisco, 1979.
19. T. Kohonen, *Self-Organization and Associative Memory*, Second Edition, Spriger-Verlag, Berlin, 1988.
20. Robert Hecht-Nielsen, *Neurocomputing*, Addison-Wesley, New York,1990.