A-12

ALGORITHMS, SOFTWARE, ARCHITECTURE

Information Processing 92

Proceedings of the IFIP 12th World Computer Congress
Madrid, Spain, 7-11 September 1992

VOLUME I

Edited by

JAN VAN LEEUWEN
Department of Computer Science
University of Utrecht
Utrecht, The Netherlands

1992

NORTH-HOLLAND
AMSTERDAM • LONDON • NEW YORK • TOKYO

Algorithms, Software, Architecture / J. van Leeuwen (Editor)
Information Processing 92, Volume I
Elsevier Science Publishers B.V. (North-Holland)

Derefinement algorithms of nested meshes

A. Plaza[a], L. Ferragut[b] and R. Montenegro[a]

[a]Department of Mathematics, University of Las Palmas de Gran Canaria, Campus Universitario de Tafira, 35017-Las Palmas de Gran Canaria, Spain

[b]Department of Applied Mathematics and Informatic Methods, Politechnic University of Madrid, c / Rios Rosas 21, Madrid, Spain

Abstract

In this paper a new derefinement algorithm of nested triangular meshes is presented and discussed. This algorithm is combined with a local refinement. The refinement / derefinement combination, that we call *readaptive process*, is very useful to solve time-dependent problems in which moving refinement areas are required. The fact of using nested meshes enables us to use the multigrid method in order to solve the equation system associated to the finite element method. Moreover, a readaptive process can be used to obtain the best piece-wise triangular support to interpolate a given two dimensional function.

Keyword Codes: G.1.8; G.4; I.1.2
Keywords: Partial Differential Equations; Mathematical Software; Algorithms

1. INTRODUCTION

In the last years, the efficiency of the adaptive finite element method has been proved, particularly to solve problems in which the solution presents a singularity [1-4]. However, when studying a time-dependent problem, if this dependency implies a change of the top gradient area, the existence of a large number of points creates a serious difficulty. Many of these nodes -though necessary in any past time- are *useless* in the present moment. This fact is very important because an increment in the number of nodes in the mesh implies an increase in the number of equations of the associated algebraic system to be solved, and in the necessary memory space. If we think that, using element by element (EBE) methods, more than 90% of CPU time is employed in solving this equation system, it seems necessary to study a derefinement algorithm able to remove *dupe* nodes and to be combined with a local refinement in this kind of problems.

On the other hand, the multigrid method presents a very convenient feature i: number of operations required for solving the equation system [5,6]. Hence, th suitability of using nested meshes both at the refining and at the derefinin stages. A new derefinement algorithm of nested meshes able to be combined wit a refinement algorithm and be implemented with the multigrid method : presented in this work. Now, the meshes generated by a readaptive process ar more flexible than those obtained by local refinement only, and the number nodes remains bounded during the whole process.

Logically, for each refinement algorithm, a parallel derefinement algorith can be thought, if the structure of nested meshes is to be maintained. We wa point out here that for every refinement algorithm not always an inver derefinement algorithm can be found. We have used a version of the 4- refinement algorithm of M.C. Rivara [7,8], that is implemented in our co Neptuno [1].

2. PREVIOUS DEFINITIONS. PROPERTIES.

Let $T = \{ \tau_1 < \tau_2 < ... < \tau_n \}$ be a sequence of nested meshes and τ_j a triangulation of T. One node N of τ_j will be called a *proper node* of τ_j if it do not belong to any previous mesh, i.e. $N \in \tau_j$ and $N \notin \tau_i$, $\forall i < j$. In other case, will be called *inherited node* in τ_j.

Similarly, any edge or element is *proper* or *inherited* in each mesh. Thus, one edge is divided in two, in the refinement process, we shall say that t former edge is the *father edge* and the latter are the *son edges*. *Father elemer* and *son elements* are defined in the same manner. Moreover, if an edge is pro in a mesh, said τ_j, we shall distinguish between *external* and *internal* edges proper edge is external if it has a father edge, and it is internal if it has not. figure 1 an example of these definitions is presented. Nodes N_1 and N_2 are pro nodes in τ_j; edges f_1 and f_2 are external in τ_j, f_3 is internal and c_3 is inherit finally, element e_1 has three sons and e_2 has two.

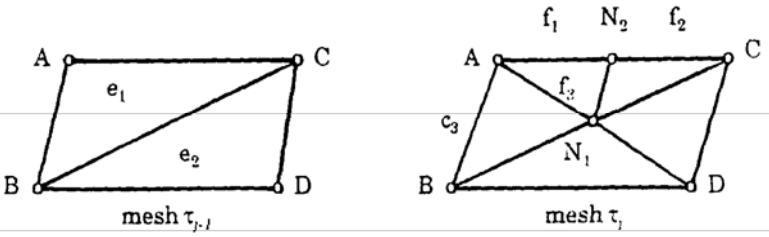


Figure 1. Example of refinement.

It is easy to observe that:

i) any proper element of some triangulations either it is inherited in the following triangulations or it has its sons there.

ii) if an element has not sons, it belongs -as proper or as inherited- to the finest mesh of the irregular nested triangulations.

These definitions and properties are important because in contrast to the refinement algorithm in which only two levels of meshes are involved the last mesh created and the new one that is being created at that moment- in the derefinement algorithm we obtain, in general, a new sequence of irregular nested triangulations: all level of meshes are involved in this process. For this reason, the *family* or *genealogy* of each element must be considered.

The fundamental property of the derefinement algorithm is the following:

iii) *Only those elements without successors, that is those which belong to the current finest mesh, can be eliminated.*

3. DATA STRUCTURE

The structure of nested meshes is defined in the code Neptuno by three vectors, that will be called *intermediate vectors:* IMNODE, IMFACE and IMELEM. In these vectors the numbers of nodes, edges and elements of each level of mesh are kept. It is sufficient to keep the proper nodes of each level because if one node belongs to a particular mesh, it belongs to the following meshes as well. This is false with respect to edges and elements. Therefore, an edge or an element appears as many times as levels of meshes it belongs to.

The family of each edge is given by the matrix IR[1:3,1:NUMF] where NUMF is the total number of edges in the structure of meshes. For each edge, IR reports the numbers of the son edges and of the father edge.

The family of each element is defined by the matrix IXH[1:6,1:NUMEL] being NUMEL the global number of elements in our structure of meshes. For each element, IXH gives us its sons, the local number of the longest side and, finally, the number of its father.

Furthermore, we need three vectors that will be called *derefinement indicators* or *level vectors* and that will find out when a node, edge or element has to be cancelled out, and at the same time, the level of the mesh in which it is proper. We point out here that this level is well defined. These vectors are: NODES, for the nodes, NFACES, for the faces and NELES for the elements.

The numbers of nodes, edges and elements eliminated by derefine procedure will be kept in other three vectors that will be called *sack vectors:* NNSAC, NFSAC and NESAC. These numbers will be used on next refinements.

Finally, and in order to facilitate the computation of the derefinement condition, we introduce a new vector, said IEX[1:NUMP] where NUMP is the number of nodes in the mesh. For each node IEX reports the *surounding edge*, that is the number of the edge in which that node is in the middle point.

We should stress that the size of the vectors and matrices used by Neptuno is

variable. When refining, the size grows; whereas at derefining the size decrease Moreover the space of memory used can be optimized, e. g. the intermediat vectors are not neccesary, but they are very convenient for ease and efficiency the multigrid method implementation.

4. THE DEREFINEMENT ALGORITHM

In general, we have a sequence of nested meshes $T = \{ \tau_1 < \tau_2 < ... < \tau_n \}$ and w want to obtain another sequence after derefining T, $T' = \{ \tau_1 < \tau_2' < ... < \tau_m' \}$, i.e.,

i) $m \prec n$

ii) $\forall\ \tau_k' \in T', \exists\ \tau_i,\ \tau_j \in T$, such as $\tau_i < \tau_k' < \tau_j$

In this case, we will say that the secuence T' is coarser than T, or T finer tha T', and write $T' < T$. This relation is a partial order relation in the family sequences of triangular meshes over a given bidimensional domain.

Schematically, the derefinement algorithm can be described as follows:

INPUT: Secuence $T = \{ \tau_1 < \tau_2 < ... < \tau_n \}$

For j=N to 2, do:

1. For each proper node of mesh τ_j, the derefinement condition is evaluate and the nodes and edges able to be eliminated are pointing out.

2. Conformity of the new level *j* is assured, minimizing the derefined area.

3. New nodal conections, and new families of edges and elements are define for the level τ_{j-1} and for the derefined level of τ_j, said τ_j^*.

4. The changes in the current level *j* are inherited to the following ones.

5. It is obtained a new sequence of nested meshes, T_j, that is the new input i the next iteration of the loop on meshes.

OUTPUT: Sequence $T' = \{ \tau_1 < \tau_2' < ... < \tau_m' \}$

5. DISCUSSION OF THE ALGORITHM

A new structure of meshes coarser than the preceding one, is obtained in eac iteration of the loop. Thus, the number of different sequences that appear alon the derefine procedure is the number of level of meshes minus one. This can b expresed as follows: $T > T_n > \ldots > T_2 = T'$. So, the mesh τ_j^* belongs, in general t an intermediate sequence; only at the last iteration of the loop (j=2) τ_2^* is in th derefined sequence T', and then τ_2^* is equal to τ_2'.

It is worth to be noted that only proper nodes are elegible in each mesh-level and out of these, only those suitable to be cancelled out are taken for evaluation This means that if one node cannot be cancelled this implies that an neighbouring nodes cannot be cancelled either. In this manner the nestedness o the new sequence is assured. An example of this question can be observed i figure 1: if N_1 has to stay, nodes A.B.C and D will also stay. This allows us t

evaluate the derefinement condiction in a number of nodes minimal, and only once for each of those.

Regarding the derefinement condition, the following one has been used: the absolute difference between the numerical solution at node N and the interpolated solution of the ends of its surounding edge, is checked. If this absolute difference is less than a constant -that can be fixed for each program run-, the node N can be cancelled. This imposed constant will be called epsilon. However, the relative difference can also be used.

The conformity of all meshes in each sequence is assured to maintaining some nodes that otherwise, and concerning the derefinement condition, could have been cancelled. In fact, if a node N belongs to the longest edge of an element in which in other edge there is another node, the node N remains. For instance, in figure 1, if N_2 remains, N_1 remains too.

In the derefinement process some new elements may arise; these elements did not exist in the input sequence. For instance, in figure 1, if node N_2 is cancelled, the element AN_1C was not in τ_j . These elements will be called half-sons. At derefining, an element that had four elements might turn out to have three, two, or none sons. The definition of the new nodal connections is obtained considering all of different possibilities and their old nodal connections.

The algorithm cheks whether all proper nodes of every mesh are going to be eliminated. In this case, it is not necessary to define new nodal connections, but to take the intermediate vectors in order to compress them. That mesh level is then eliminated. For this reason, the number of meshes in the output structure can be made lesser than in the input sequence.

At the end of each iteration, the modifications of the new nodal connections have to be passed on to the following levels of meshes. This is obtained by changing the intermediate vectors. Therefore it is necessary to manage the derefinement indicators. A new intermediate sequence is obtained and taken out as input in the next iteration.

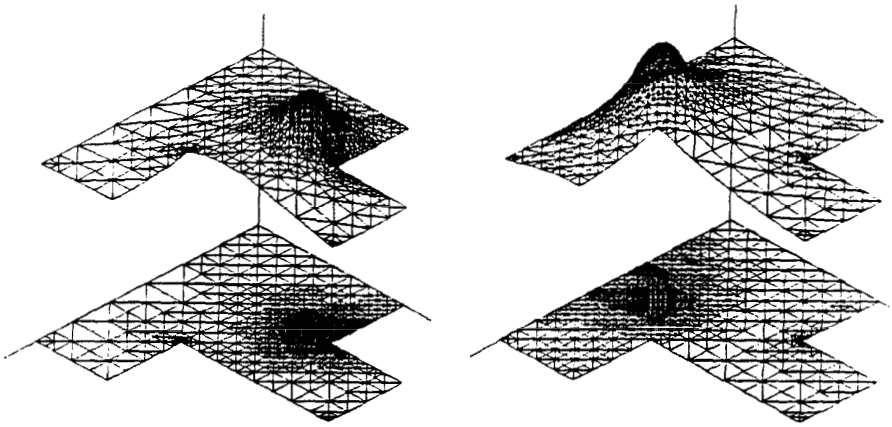
6. NUMERICAL RESULTS

Some numerical examples are presented in this section.

Figure 2 shows some meshes and the solution in different time of events of a Poisson problem in which the function of the second member is time-dependet. The domain geometry is non symetrical and we have Dirichlet and Neumann conditions in the boundary. Maximun number of nodes managed was 3397, and maximun number of mesh-level was 24. The maximum number of nodes eliminated by one application of the derefinement algorithm was 2748 and the minumun 252. Parameter epsilon used was 0.0009 with relative difference as derefinement condition.

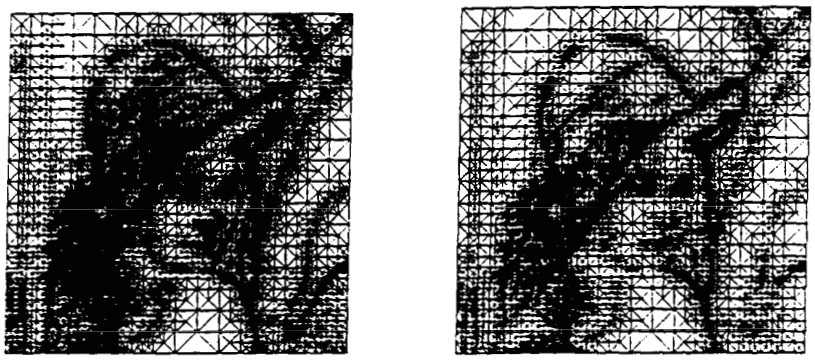
Figure 3 shows the efficiency of the derefinement algorithm to obtain the best piece-wise support to interpolate a given two-dimensional function. The function

selected here is a classical image of 256x256 pixels, see for example [9]. sequence of eight nested meshes has been generated automatically. In each noc we allocate the corresponding grey level value, from 0 to 255. After, th derefinement algorithm has been applicated using different values for epsilc and absolute difference as derefinement condition.



(a) (b)

Figure 2. Derefined meshes and solution of a quasievolutive Poisson proble (a) t=0 s, 674 nodes on the mesh, (b) t=6 s, 823 nodes.



(a) (b)

Figure 3. Different derefined meshes to approach a black-and-white imag (a) ε=15; 13892 nodes; (b) ε=25; 9112 nodes.

7. CONCLUSIONS

The refinement/derefinement combination is very useful to solve time-dependent problems in which moving refinement areas are required. The structured derefinement algorithm enables us to use the multigrid method in order to solve the equation system associated to the finite-element method. The aditional computation time required by this algorithm amounts less than 1% of total execution time. Only a scalar optimization at compilation time has been done on a Stardent 3000 computer.

As another application, this derefinement algorithm can be used for local refining. Derefining after global refinement is thereby equivalent to a local refinement procedure, not requiring the use of error indicators. This implies clear advantages in the class of problems for which error indicators cannot be found or their computation in too costly.

Finally, this algorithm can also be used in order to obtain the best piece-wise support to interpolate a given two-dimensional function, or to approach the initial solution in an evolutive problem.

8. REFERENCES

1 L. Ferragut, A solution to the programming problem of self-adaptive finite element methods (in spanish), Anales de Ing. Mecánica, Year 5, n. 1, pp. 201-206 (1987).

2 J.P. de S.R. Gago, D.W. Kelly, O.C. Zienkiewicz and I. Babuska, A Posteriori Error Analisys and Adaptive Processes in the Finite Element Method; Part II: Adaptive Mesh Refinement, Int. Jour. Num. Meth. in Eng., Vol. 19, pp. 1621- 1656 (1983).

3 D.W. Kelly, J.P. de S.R. Gago, O.C. Zienkiewicz and I. Babuska, A Posteriori Error Analisys and Adaptive Processes in the Finite Element Method; Part I: Error Analisys, Int. Jour. Num. Meth. in Eng., Vol. 19, pp. 1593-1619 (1983).

4 J.Z. Zhu and O.C. Zienkiewicz, Adaptive Techniques in the Finite Element Method, Comp. in App. Num. Meth., Vol. 4, pp. 197-204 (1988).

5 W. Hackbush and V. Trottengurg (eds.), Multigrid Methods, Lectures Notes in Mathematics, Springer-Verlag, Berlin, 1982.

6 W. Hackbush and V. Trottengurg (eds.), Multigrid Methods II, Lectures Notes in Mathematics, Springer-Verlag, Berlin, 1982.

7 M.C. Rivara, A Grid Generator based on 4-triangles conforming mesh refinement algorithms, Int. Jour. Num. Meth. in Eng., Vol. 24, 1343-1354 (1987).

8 M.C. Rivara, Selective refinement / derefinement algorithms for sequences of nested triangulations, Int. Jour. Num. in Eng., Vol. 28, pp. 2889-2906 (1987).

9 R. Aravind and Allen Gersho, Image Compression Based on Vector Quantization with Finite Memory, Opt. Eng. Vol. 26, No. 7, pp. 570-580, July (1987).