COMMUNICATIONS IN NUMERICAL METHODS IN ENGINEERING, Vol. 10, 403–412 (1994)

EFFICIENT REFINEMENT/DEREFINEMENT ALGORITHM OF NESTED MESHES TO SOLVE EVOLUTION PROBLEMS

L. FERRAGUT

Department of Applied Mathematics and Informatic Methods, Polytechnic University of Madrid, c/Rios Rosas 21, 28003-Madrid, Spain

AND

R. MONTENEGRO AND A. PLAZA

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

SUMMARY

An adaptive refinement/derefinement algorithm of nested meshes is presented. Some definitions are introduced. The main properties of the derefinement algorithm are remarked upon and its efficiency is shown through two numerical examples: a time-dependent convection–diffusion problem with dominant convection and a quasistationary problem.

1. INTRODUCTION

The traditional approach to the solution of evolution problems using adaptive finite-element methods has proved to be very useful in this kind of problem.[1–3] However, in the topic of nested meshes, using local refinement, if the areas to be refined change with time, the appearance of a large number of nodes creates a difficulty. Many of these nodes, although once necessary, are useless at the present moment.

An increment in the number of nodes in the mesh implies an increase in the number of equations of the system to be solved. Therefore it seems necessary to develop a derefinement algorithm capable of removing dupe nodes, in order to get a good approximation of the numerical solution obtained in the previous time step and to allow for combination with a local refinement. We have used triangular elements with three nodes and the 4-T algorithm of Rivara[4–5] at the refining. With this combination (refinement and derefinement), we obtain families of sequences of nested meshes which are more flexible than those obtained by local refinement only and the number of equations does not increase excessively during the whole evolutive process.[6–7]

Besides, the fact of using nested grids enables us to use the multigrid method easily, to solve the system of equations associated with the finite-element method.[8]

2. THE DEREFINEMENT ALGORITHM

2.1. *Definitions and properties*

Let $T = \{\tau_1 < \tau_2 < \cdots < \tau_n\}$ be a sequence of nested triangular grids and τ_j any triangulation

CCC 0748–8025/94/050403–10
Received 25 January 1993
Revised 20 September 1993

of T. One node N of τ_j will be called a proper node of τ_j if it does not belong to any previous mesh. In other cases, N will be called an inherited node in τ_j. Similarly, the edges and elements are named at each level. Proper nodes of τ_j are called j-new nodes by Rivara.[5]

If an edge is divided in two at refining, it is called the father edge of these two, and these are the son edges of the former. Similarly the father elements and son elements are defined. As we are using the 4-T algorithm of Rivara at refining, an element has four sons or less.

When an element is refined, some edges appear inside it. These edges are called internal edges; these edges are called j-new edges by Rivara.[5] On the boundary of the element some edges appear as well. Now these edges are called external edges. In Figure 1, nodes N_1, N_2 and N_3 are proper ones in τ_j; edges f_1 and f_2 are external in τ_j, f_3 is internal and c_1 is inherited.

In this context, some properties are:

(i) Any proper element of some triangulation is either inherited in the following triangulation or has its sons there.

(ii) If an element has no sons, it belongs to the finest mesh of the sequence of nested triangulations.

These definitions and properties are important because in contrast to the refinement algorithm in which only the last mesh is created, and the new one that is being created are involved, in the derefinement algorithm all levels of meshes are involved.

The fundamental property of the derefinement algorithm is the following:

(iii) only those elements without successors, i.e. elements that belong to the finest mesh, can be eliminated.

As the finest mesh is changing at derefining, an element will be able to be eliminated if, at derefining, its successors have been previously eliminated.

2.2. *Data structure*

Each element, face or node in any level $\tau_j \in T$ has only one global number. Let *NUMN*, *NUMF* and *NUMEL* be, respectively, the total numbers of different nodes, faces and elements

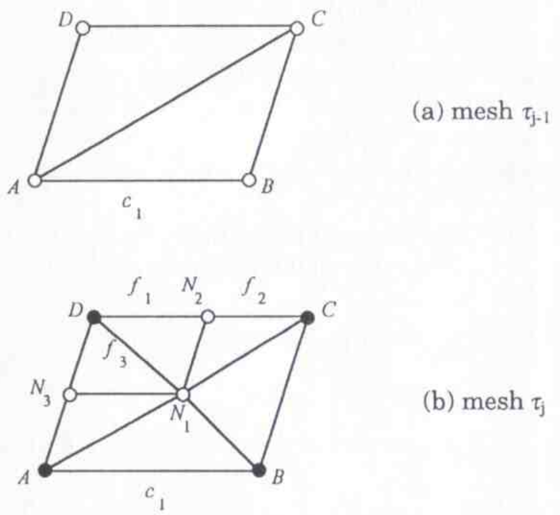


Figure 1. Example of refinement

in the sequence T plus the ones introduced in the sack vectors (these vectors are defined later). Let $NUMP$ be the total number of nodes in T. For each level τ_j the actual numbers of nodes, edges and elements are known, named $NN(j)$, $NF(j)$ and $NE(j)$, where $1 \leqslant j \leqslant n$, n being the number of levels in T.

The data structure can be summarized as follows:

(a) Structure vectors: **IMNODE**(1:*NUMP*), **IMFACE**(1:*ISPF*) and **IMELEM**(1:*ISPE*) for the nodes, faces and elements, respectively. In **IMNODE** only the proper nodes of each level are kept because if one node belongs to a particular mesh, it belongs to the following meshes as well, so $NUMP = NN(n)$. **IMFACE** and **IMELEM** keep the global numbers of all faces and elements, respectively, of each level of T:

$$ISPF = \sum_{j=1}^{n} NF(j) \text{ and } ISPE = \sum_{j=1}^{n} NE(j)$$

With this data structure the implementation of the multigrid method is relatively simple.

(b) Genealogy vectors: **IR**[1 : 3, 1 : *NUMF*] and **IXH**[1 : 6, 1 : *NUMEL*]. For each edge, **IR** reports the numbers of its son edges and its father edge. Similarly, for each element, **IXH** gives us the number of its son elements, its father element and the local number of its longest side.

(c) Derefinement vectors: **NODES**(1 : *NUMN*), **NFACES**(1 : *NUMF*) and **NELES** (1 : *NUMEL*). For each node, edge or element, these vectors give us the level at which it is proper and the sign of the vectors is used to control the derefinement procedure.

(d) Sack vectors: **NNSAC**, **NFSAC** and **NESAC**. In these vectors, the global number of nodes, edges and elements that have been eliminated are kept to be used in future refinements.

(e) Surrounding edge: **IEX**(1 : *NUMN*) . For each node **IEX** reports the number of the edge at which that node is at the middle point.

This data structure allows an easy implementation of the refinement/derefinement algorithm in standard finite-element codes.

2.3. *The derefinement procedure*

Let $T = \{\tau_1 < \tau_2 < \cdots < \tau_n\}$ be a sequence of nested triangular grids, where τ_1 represents the initial mesh and τ_n the finest mesh. Our goal is to obtain another sequence after derefining T, called T^d, i.e. $T^d = \{\tau_1 < \tau_2^d < \cdots < \tau_m^d\}$ where $m \leqslant n$.

The derefinement algorithm can be described briefly in this form:

INPUT: Sequence $T = \{\tau_1 < \tau_2 < \cdots < \tau_n\}$

Loop in levels of T; for $j = n$ to 2, do:

1. For each proper node of τ_j the derefinement condition, which is defined in Section 2.4, is evaluated and the nodes and edges which can be eliminated are pointed out with the derefinement vectors.
2. Conformity of the arising new level j is ensured.
3. (a) If some proper node of τ_j remains new, nodal connections are defined for the new level j. Genealogy vectors of τ_j and τ_{j-1} are modified.

3. (b) In other cases, the current level j is deleted in the structure vectors. Genealogy vectors of τ_{j-1} are modified.
4. The changes in the mesh are passed on to the following meshes. The structure vectors are compressed.

OUTPUT: Sequence $T^d = \{\tau_1 < \tau_2^d < \cdots < \tau_m^d\}$

The application of the derefinement algorithm to a sequence of five levels can be observed in Figure 2. There, the first line represents the sequence T. The second line shows the derefined sequence T^d. The white nodes are the proper nodes of each level capable of being selected for evaluation; the black nodes are the proper nodes that will have to remain to ensure conformity.

We must take into account that only proper nodes are eligible in each mesh level, and out of these, only those suitable to be cancelled out are taken for evaluation. This is due to the fact that if N is a particular node that cannot be cancelled and c is its surrounding edge, then any node of the elements in which c is an edge cannot be cancelled either. In this way we can ensure that the meshes of the new sequence are nested. This allows us to evaluate the derefinement condition in a minimum number of nodes, and only once for each of those.

Once the derefinement condition has been checked in all the eligible proper nodes of a particular mesh, the conformity of the arising new level is ensured. Therefore, the conformity of all meshes in each sequence is ensured, maintaining some nodes that, otherwise, given the derefinement condition, might have been cancelled. For instance, in Figure 1, if N_2 remains, N_1 remains too.

Once T^d is obtained, the equation number associated with each degree of freedom/node must be redefined to apply again the finite-element solver, preserving the global number of each node and the previous numerical solution.

2.4. Derefinement indicator

A proper node may be removed if the absolute difference between the values in this node of the numerical solution and its corresponding interpolated function is less than a sufficiently small parameter $\varepsilon > 0$. That is, if u_h is the numerical solution for a given mesh and u_h^d is the interpolated function of u_h in the derefined mesh, we will get

$$\| u_h - u_h^d \|_\infty = \sup_x | u_h(x) - u_h^d(x) | < \varepsilon$$

Obviously, this derefinement indicator does not allow us to control the discretization error; in an adaptive algorithm this control is usually performed by an error indicator in the refinement process. It could be argued that the same error indicator should be used as a derefinement indicator. However, when we use a time-step integration scheme, a good approximation of the solution at time t_n must be kept to calculate the approximation at time $t_{n+1} = t_n + \Delta t_n$. Then the described derefinement indicator can be considered optimal in the sense that a given solution is approximated with a minimum number of nodes after derefining. If $\delta > 0$ is a given tolerance for the error in the maximum norm, a practical criterion to choose ε would be to take a value sufficiently smaller than δ, for example $\varepsilon \approx 0\cdot 1\ \delta$. Alternatively, one may choose for ε a small fraction of $\| u \|_\infty$ or, depending on the problem, another characteristic value of the range of the solution.

2.5. Comparison with Rivara's algorithm

We should point out that our derefinement procedure differs in some ways from the one

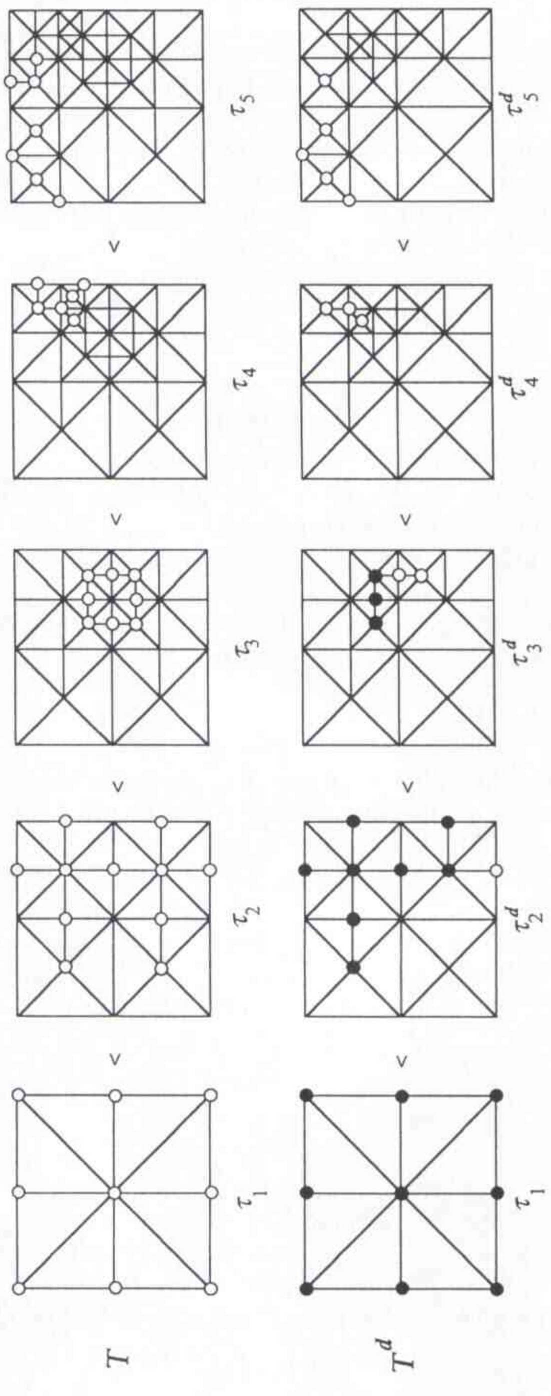


Figure 2. Example of application of derefinement procedure

proposed by Rivara:[5]

- We introduce a simple derefinement condition for the nodes of the mesh and not for the elements.
- The molecular structure defined by Rivara is slightly more economical. However, our data structure allows for an easy adaptation of the algorithm to existing standard finite-element codes and can be more generally applied in the sense that different kinds of problems and finite elements can be implemented without modification of the data structure.
- We do not need to distinguish if a particular node has been created as a consequence of the creation of another at some previous level.
- Our derefinement algorithm can be combined with the 2-T or 4-T refinement algorithms and, in both cases, the derefinement area can be very local.

3. NUMERICAL RESULTS

To verify the validity of the described algorithm we present two numerical examples. The results confirm that the refinement/derefinement combination is very useful to solve time-dependent problems in which moving refinement areas are required. The additional computational time required by this algorithm amounts to less than 1 per cent of the total execution time.

In neither of the two examples have we included a rigorous control of the error in the maximum norm, so the second criterion indicated in Section 2.4 has been adopted.

3.1. A convection–diffusion problem

We consider the convection–diffusion linear problem defined in a two-dimensional domain Ω, a unit square domain centred at the point $(0\cdot5, 0\cdot5)$ of boundary Γ:

$$\frac{\partial u}{\partial t} + \mathbf{v} \cdot \nabla u - \nabla \cdot (k \nabla u) = f$$

where $\mathbf{v} = \mathbf{v}(\mathbf{x})$ is the fluid velocity. The semi-implicit formulation used to approximate the evolution convection–diffusion process and a discussion of the stability and consistency can be found in Reference 9. There, the numerical integration scheme used is

$$u^{n+1} - \Delta t\, \nabla \cdot [k\, \nabla u^{n+1}] = \Delta t f^{n+1} + u^n - \Delta t \mathbf{v} \cdot \nabla u^n + \frac{\Delta t^2}{2} \sum_i (\mathbf{v} \cdot \nabla v_i) \frac{\partial u^n}{\partial x_i} + \frac{\Delta t^2}{2} \sum_{i,j} v_i v_j \frac{\partial^2 u^n}{\partial x_i\, \partial x_j}$$

We suppose null Neumann conditions on Γ and a rotating velocity field, with $v_1 = \omega x_1(1 - x_1)(x_2 - 0\cdot5)$ and $v_2 = \omega x_2(0\cdot5 - x_1)(1 - x_2)$. In the present application $\omega = 2\cdot5 \times 10^4$, $k = 1$ and $f = 0$. That is, a significant Peclet number is $3\cdot125 \times 10^3$.

Here we have used the following refinement indicator, η_i, for an element Ω_i:

$$\eta_i = h_i^2 \mid \nabla u_h \mid$$

h_i being the diameter of Ω_i and u_h the linear numerical solution in the triangular element.

The initial solution is a given function that is approximated in Figure 3(a) using the

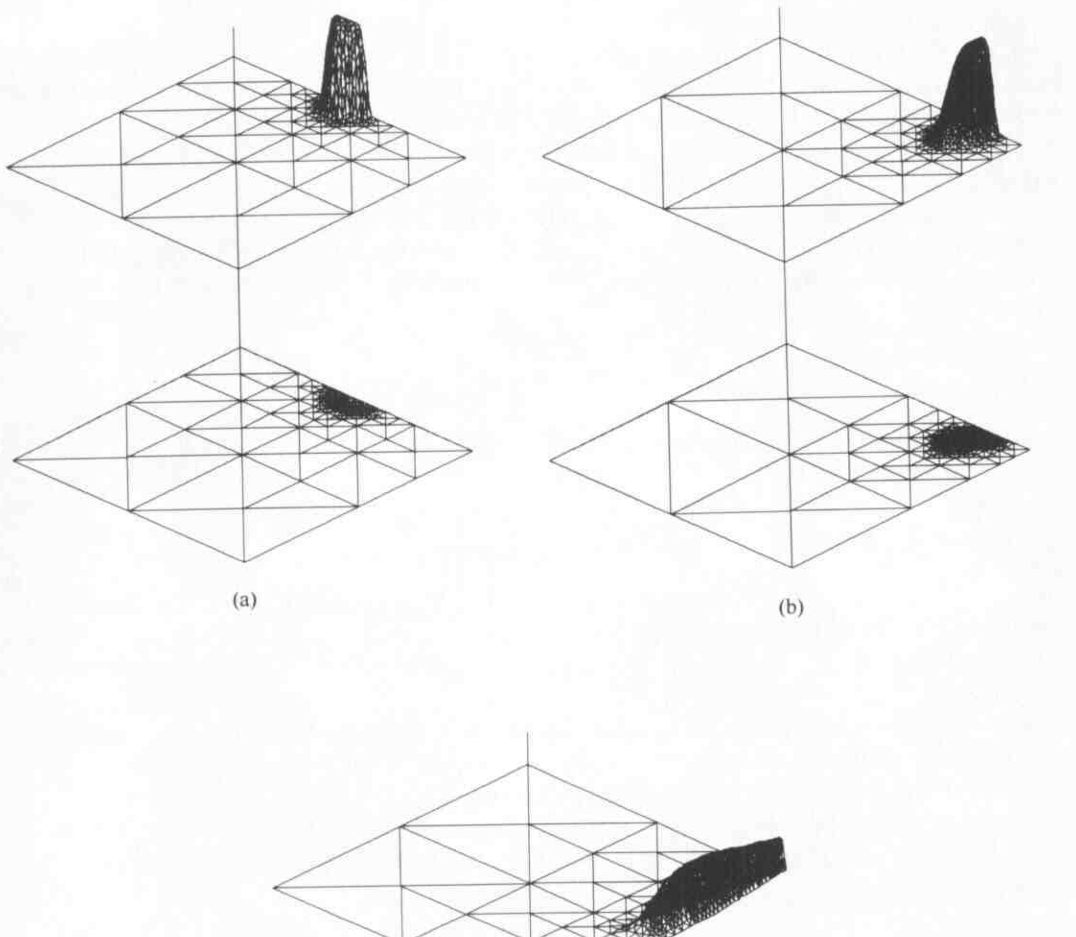


Figure 3. Meshes and solutions of an evolution problem: (a) initial solution, $t = 0{\cdot}0$, 234 nodes; (b) $t = 0{\cdot}00014$, 864 nodes; (c) $t = 0{\cdot}00032$, 919 nodes

refinement/derefinement algorithm. This combination enables us to get a good approximation with a minimum number of nodes.

In order to obtain the stationary solution (constant in this example), 2892 time steps have been calculated. The adaptive strategy was: three refinements followed by the derefinement procedure. The exact solution verifies $\max_{x,t} |u(x,t)| = 1$, and taking $\varepsilon = 0\cdot 001$ we are sure that the error introduced by derefinement is less than 1 per cent of this maximum value. The time increases after each refinement. To evaluate the time increment, the stability conditions have been considered.[9] In each time step, one multigrid iteration is enough to solve the system of equations. Figure 3 shows some meshes and the respective solutions for different time steps. Once the stationary state is approached, the derefinement algorithm reduces the sequence of meshes to the first coarse level.

3.2. A quasistationary problem

Figure 4 shows the domain for a Poisson problem in which the function f of the second member is time-dependent:

$$f(x_1, x_2) = \begin{cases} 40 \text{ if } d(t) < r \\ \ 0 \text{ if } d(t) > r \end{cases}$$

where r is the ratio of the circular source (shaded in Figure) and $d(t)$ the Euclidean distance between (x_1, x_2) and the centre of the source that moves with angular velocity $\omega = \pi/2$ rad/seg counter-clockwise. We assume the Dirichlet condition ($u = 1$) on Γ_1 and the null Neumann condition on Γ_2.

With respect to the derefinement condition we have taken $\varepsilon = 0\cdot 009$ using the argument of the previous example. At each time step, the adaptive strategy used has been one global refinement followed by the derefinement procedure. Figure 5 shows several results.

A multigrid method with conjugate gradient as smoothing has been applied to solve at each step the linear system of equations producing convergence with a stabilized value of four multigrid iterations.

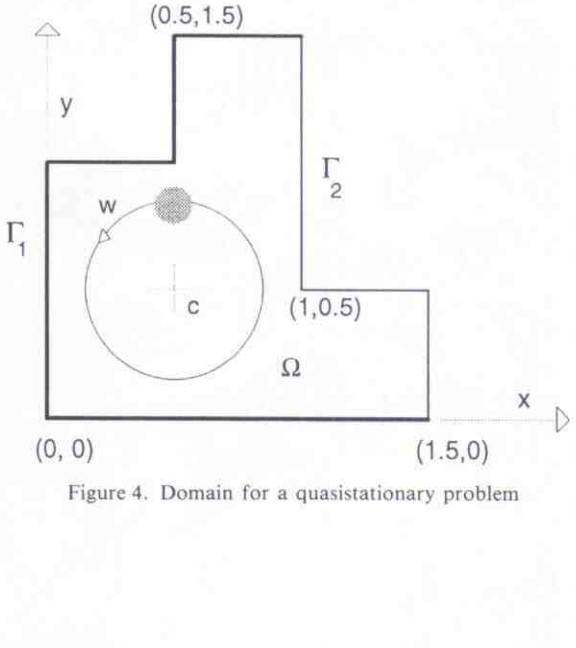
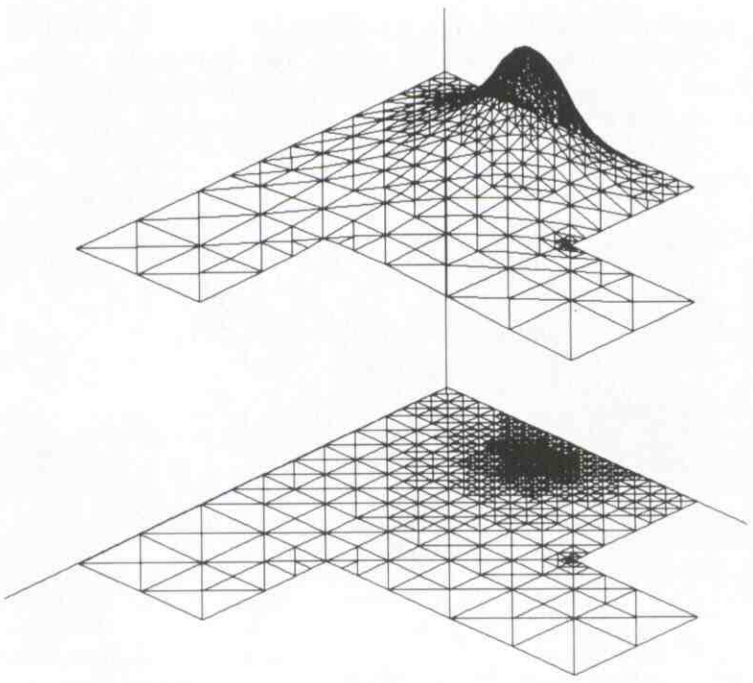
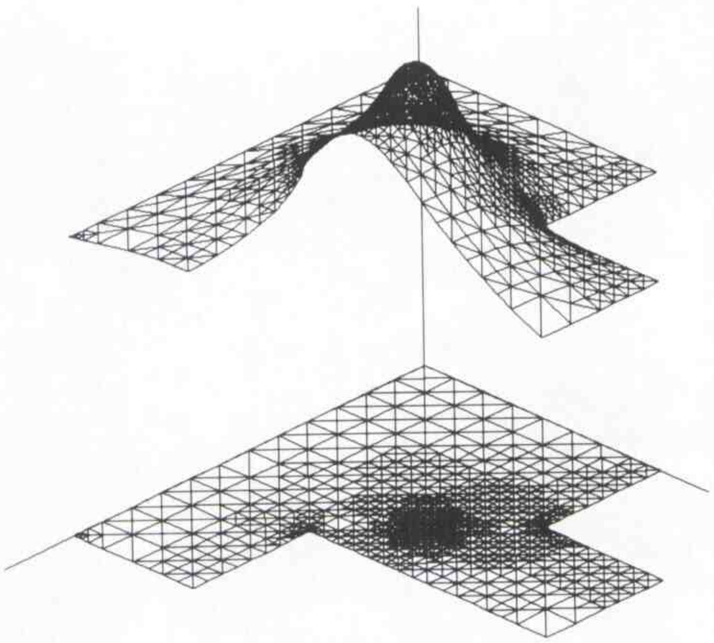


Figure 4. Domain for a quasistationary problem



(a)



(b)

Figure 5. Meshes and solutions of a quasistationary problem: (a) $t = 1$, 838 nodes; (b) $t = 3 \cdot 5$, 959 nodes

Remark. This example suggests another application of the derefinement algorithm. It can be used as a local refinement procedure. Derefining after global refinement is thereby equivalent to a local refinement. In this case the information obtained through two computed solutions on successively refined meshes can be used to estimate the error by extrapolation.

REFERENCES

1. D. W. Kelly, J. P. de S. R. Gago, O. C. Zienkiewicz and I. Babuska, 'A posteriory error analysis and adaptive processes in the finite element method: Part I – Error analysis', *Int. j. numer. methods eng.*, **19**, 1593–1619 (1983).
2. J. P. de S. R. Gago, D. W. Kelly, O. C. Zienkiewicz and I. Babuska, 'A posteriory error analysis and adaptive processes in the finite element method: Part II – Adaptive mesh refinement', *Int. j. numer. methods eng.* **19**, 1621–1656 (1983).
3. J. Z. Zhu and O. C. Zienkiewicz, 'Adaptive techniques in the finite element method', *J. Comput. Appl. Numer. Methods*, **4**, 197–204 (1988).
4. M. C. Rivara, 'A grid generator based on 4-triangles conforming. Mesh-refinement algorithms', *Int. j. numer. methods eng.*, **24**, 1343–1354 (1987).
5. M. C. Rivara, 'Selective refinement/derefinement algorithms for sequences nested triangulations', *Int. j. numer. methods eng.*, **28**, 2889–2906 (1989).
6. A. Plaza, R. Montenegro and L. Ferragut, 'An adaptive refinement/derefinement algorithm of structured grids for solving time-dependent problems', in Ch. Hirsch *et al.* (Eds.), *Numerical Methods in Engineering*, Elsevier Science Publishers B.V., Amsterdam, 1992, pp. 225–232.
7. A. Plaza, *Derefinement Algorithms of Nested Bidimensional Grids* (in Spanish), Doctoral Thesis, University of Las Palmas de Gran Canaria, 1993.
8. W. Hackbush, *Multigrid Methods and Applications*, Springer-Verlag, Berlin, 1985.
9. R. Montenegro, G. Montero, L. Ferragut and G. Winter, 'Application of adaptive finite elements methods to convection-diffusion problems in 2-D' (in Spanish), *Rev. Int. Met. Numer. Cal. Dis. Ing.*, **5**, 535–560 (1989).