ELSEVIER

Advances in Engineering Software 27 (1996) 51–57

S0965-9978(96)00005-1

0965-9978/96/$15.00

An improved derefinement algorithm of nested meshes

A. Plaza,* R. Montenegro* & L. Ferragut†

**Department of Mathematics, University of Las Palmas de Gran Canaria, 35017-Las Palmas de Gran Canaria, Spain*
†*Department of Pure and Applied Mathematics, Faculty of Science, University of Salamanca, Spain*

In this paper we present a new version of the derefinement algorithm developed by Plaza *et al.* (in *Numerical Methods in Engineering*, Elsevier Science, Amsterdam, 1992, pp. 225–232; in *Algorithms, Software, Architecture*, Elsevier Science, Amsterdam, 1992, pp. 409–415; A. Plaza, PhD thesis, University of Las Palmas de Gran Canaria, 1993; *Commun. Numer. Meth. Engng*, 1994, **10**, 403–412).[1–4] The purpose is to achieve a better derefinement algorithm with a lesser degree of complexity. We present the theoretical study of this improved derefinement algorithm and of the inverse one for refinement. Firstly, our initial version of the derefinement algorithm is summarized. Then we present the refinement algorithm associated with the improved derefinement one. Finally, automatic control of the sequences of irregular nested triangulations is shown by means of the resolution of an unsteady problem. In this problem the initial mesh has only nine nodes and a combination of refinements and derefinements have been applied to approach both the circular domain and the initial solution.

1 INTRODUCTION

It is well known that numerical grid generation and the ability to control automatically and adaptively discretizations in the numerical solution of partial differential equations is critical to the reliable application of numerical analysis techniques, especially by means of adaptive finite-element methods and multigrid algorithms. We can say that a good discretization of the domain in which a problem in differential equations must be solved is, at least, as important as the numerical formulation of that problem.[5,6] On the other hand, we usually need efficient and robust algorithms to achieve a good discretization. This is particularly important in time-dependent problems, in which the changing of refined areas is required.

Our algorithms[1–4,7,8] are based on the work of Rivara.[9–12] In fact they can be seen as alternative versions of her algorithms. These algorithms build and manage automatically sequences of nested irregular discretizations of the domain. This automatic control is very important to applying practical multigrid procedures.

In this paper, we introduce our first version of the derefinement algorithm. It can be found in detail in Refs 1–4. Then we study briefly its complexity in the sense of number of operations by means of an upper bound for the worst case. Subsequently, and based on the previous study, we introduce a new vector in the data structure and some modifications in the algorithm to obtain a new improved version. Furthermore, as a derefinement algorithm can be understood as the inverse of a refinement algorithm, a new version of the derefinement algorithm naturally leads us to consider whether a new version of the refinement algorithm is possible. To start this study, the first version of the refine-ment algorithm developed by Ferragut,[7,8] which is also a version of the 4-T algorithm of Rivara,[9,10] is analyzed.

Finally, we compare in some detail our final version of the derefinement algorithm with the one proposed by Rivara.

2 THE FORMER DEREFINEMENT ALGORITHM

Regarding the derefinement algorithm we can distinguish two aspects: the geometric problem and the algebraic problem. The former involves eliminating some nodes of the sequence in such a way that the nestedness of the sequence is assured. The latter consists of the renumbering of the equations that remain after derefining the mesh geometrically. This renumbering, in our algorithm, works only with the number of equations; the global number of each node remains the same before and after derefining. The geometrical

problem of our derefinement algorithm can be summarized as follows:

2.1 A scheme of the algorithm

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 in the sequence. Derefining the sequence means to obtain a new sequence:

$$T' = \{\tau_1 < \tau_2' < \cdots < \tau_m'\}$$

where $m \leq n$.

Our former version of the derefinement algorithm[1-4] works in the following manner:

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 mesh τ_j, the derefinement condition is evaluated and the nodes and edges suitable to be eliminated are marked.
2. Conformity of the new level j is assured, decreasing the derefined area.
3. The following question is presented:

3(a) If some *proper node* of τ_j is eliminated, then:

3(a1) If some *proper node* of τ_j stays, then:
New nodal connections, and new families of edges and elements are defined for the level τ_{j-1} and for the derefined level of τ_j, τ_j^* say.

In the other case,

3(a2) The current level j is deleted in the structure vectors. Genealogy vectors of τ_{j-1} are modified.

End if.

In the other case,

3(b) All *proper nodes* of τ_j must remain and level τ_j is not modified.

End if.

4. The changes in the current level j are inherited by the following ones.
5. A new sequence of nested meshes, T^j, is obtained, which is the new input for the next iteration of the loop in levels.

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

Reference 4 can be consulted for detail on the data structure. It can be noticed here that the additional necessary data for the application of the derefinement algorithm is equal to about the number of nodes in the domain, although for each level of mesh the global number of all edges and elements are kept sequentially. Therefore, an edge or an element appears as many times as levels of meshes it belongs to. Hence, we can say that the required memory is about the number of levels n, multiplied by the number of nodes, NN: $O(nNN)$. As outlined in Ref. 2, this data structure can be optimized, but it is very convenient for an easy and efficient implementation of the multigrid method.

With regard to the derefinement condition, the following has been used: the absolute difference between the numerical solution at node N and the interpolated solution of the ends of its *surrounding 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. Hence this imposed constant will be called epsilon. The relative difference can also be used for the same purpose.

2.2 Conforming procedure

It is worth noting that the derefinement condition is checked in a minimum number of nodes, by using derefinement vectors for nodes, edges and elements: only *proper nodes* are eligible in each mesh-level, and out of these, only those suitable to be cancelled are taken for evaluation. If one node cannot be eliminated no neighbouring nodes can be cancelled either. More details can be found in Refs 2–4.

Once the derefinement condition has been checked in all the eligible *proper nodes* of a particular mesh (inside the loop in levels), the conformity of the arising new level is assured. The conformity of all meshes in each sequence is assured by minimizing the derefinement area, that is, maintaining some nodes that otherwise and concerning the derefinement condition, could be eliminated. In fact, if a node P belongs to the longest edge of an element in which there is another node on any other edge, the node P must remain.

The procedure assuring the conformity of the mesh τ_j is summarized as follows: the concept of the *1/2-non-conforming triangle* of Rivara is used:[9] a triangle t will be called a *1/2-non-conforming triangle* if there exists at least one hanging non-conforming midpoint P over one of its sides. Point P will be called a *1/2-non-conforming point*. For instance, in Fig. 1 the shaded triangles are *1/2-non-conforming*.

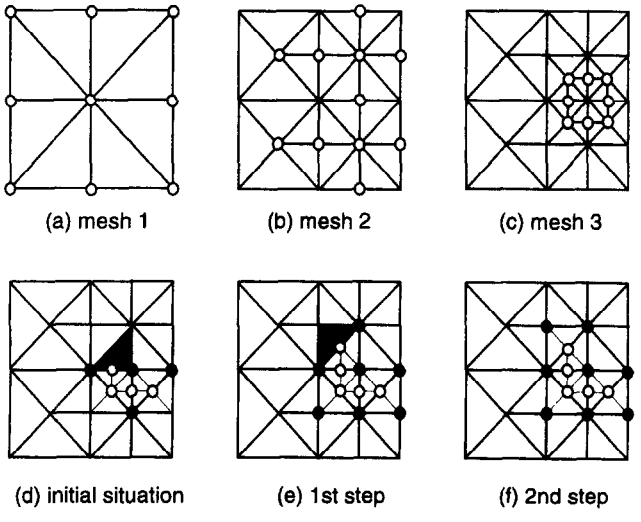

Fig. 1. Conformity of the arising new level.

INPUT (τ_j, τ_{j-1}, Derefinement Vectors)
While Conformity must be assured:
  For each $t \in \tau_{j-1}$:
    If t is 1/2-non-conforming:
      Local conformity of triangle t is assured.
      Assure conformity.
    End if.
  End for.
End while.
OUTPUT (τ_j, τ_{j-1}, Derefinement Vectors)

Local conformity is assured by changing, if necessary, the derefinement vector of the middle point node of the longest edge. But if there is a change in the derefinement vector of a node, conformity must be assured again by a new loop in the elements of the mesh τ_{j-1}.

An example of how the conformity of the arising new level was assured by this version of the algorithm can be seen in Fig. 1. There, the first line represents a sequence of three nested meshes in which the *proper nodes* are shown in white. The second line shows the evolution for the derefinement vectors, when the third level is derefined. In the second line of the figure, the white nodes mean the *proper nodes* of the third level that will stay according to the derefinement condition (Fig. 1d) or conformity of the mesh (in Fig. 1e,f); the black nodes are inherited nodes in this level and these are marked because they are not suitable to be cancelled in order to assure the nestedness of the sequence of meshes. The shaded area is the non-conforming area at each step of this process.

2.3 Efficiency and complexity of the algorithm

The order of operations required by the derefinement algorithm has been estimated by two parameters: the number of nodes in the mesh, NN, and the number of levels of mesh in the sequence, n. We will give here an upper bound of the number of operations.

If we follow the previous outline of the algorithm and if $NNP(j)$ is the number of *proper nodes* of the level τ_j, we have: Step 1 of the algorithm implies a complexity of $O(NNP(j))$; to assure the conformity $O(NNNNP(j))$; Steps 3(a1) and 3(a2): $O(NN)$; and finally, to inherit the changes by the following levels of mesh: $O(nNN)$.

If we think that the number of elements of the mesh τ_{j-1} can be considered about the order of NN, this implies that assuring the conformity costs less than $O(NNNNP(j))$.

Since

$$\sum_{j=2}^{n} NNP(j) \leq NN \tag{1}$$

although Steps 1 and 2 depend upon n, their complexity can be computed by:

$$\begin{aligned} O(nNNP(j)) &= O(NN) \\ O(nNNNNP(j)) &= O(NN^2) \end{aligned} \tag{2}$$

Therefore our former version of the derefinement algorithm gets a complexity of the order:

$$O(NN^2 + NN + n^2NN) = O(NN^2 + n^2NN) \tag{3}$$

However, the conformity can be assured *at the same time* that *proper nodes* are taken for evaluation of the derefinement condition. In this manner the loops in the elements of the previous level of mesh are avoided and this implies (except for a constant) as many operations as *proper nodes* in each level. That means that Steps 1 and 2 have a complexity of $O(NN)$. Then, in this case, if we can assure that the number of levels is bounded, we have a linear complexity for the algorithm: $O(NN)$. Actually, this analysis gives us an upper bound for the complexity of the algorithm; experience tells us that additional computation time required by the algorithm amounts to less than 1% of total execution time.

3 THE NEW 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 in the sequence. Our goal is to obtain another sequence after derefining T, T' say: $T' = \{\tau_1 < \tau_2' < \cdots < \tau_m'\}$ where $m \leq n$.

The new derefinement algorithm can be shortly described 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 is evaluated and the nodes and edges suitable to be eliminated are pointed out. Conformity is assured *in a new manner*.
2(a) If some *proper node* of τ_j is eliminated, then:
  2(a1) If some *proper node* of τ_j stays, then:
    New nodal connections are defined for the new level j. Genealogy vectors are modified.
  In the other case:
  2(a2) The current level j is deleted in the data structure. Genealogy vectors are modified.
  End if.
In the other case:
2(b) Level τ_j is not modified.
End if.
3. The changes in the mesh are inherited by the following meshes.
4. A new sequence of nested meshes T^j is obtained.
OUTPUT: Sequence $T' = T^2 = \{\tau_1 < \tau_2' < \cdots < \tau_m'\}$

3.1 The new conforming procedure

The way in which the conformity of the new level of mesh is assured enables us to achieve a better, less complex, algorithm. This is carried out by means of a new vector of *neighbouring elements* of grid-edges.

In the new version of the derefinement algorithm, the conformity of the arising new sequence is assured at the same time as the derefinement condition is checked. This implies clear advantages with regard to the complexity of the new version of the algorithm.

The procedure assuring the conformity of the mesh τ_j is summarized in the following.

INPUT (τ_j, τ_{j-1}, Derefinement Vectors)
For each *proper node* $N \in \tau_j$ and if node N must remain for derefinement condition: let c be the *surrounding edge* of N, for each neighbouring element $t \in \tau_{j-1}$ of c do:
If t is 1/2-non-conforming:
- The derefinement vector for the node P of its longest side is changed.
- The derefinement vector for some nodes of t is changed as well.

End if.
End for.
OUTPUT (τ_j, τ_{j-1}, Derefinement Vectors)

The management of the derefinement vectors for the nodes of the same example shown in Fig. 1, using the new version of the algorithm, can be seen in Fig. 2.

The second line shows the evolution of the derefinement vectors, when the third level is derefined. The white nodes in the second line of the figure mean the *proper nodes* of the third level that will stay according to the derefinement condition. Shaded nodes mean *proper nodes* that will stay according to the conformity of the mesh (in Fig. 1e,f); black nodes are *inherited nodes* in this level and these are marked because they are not suitable to be cancelled in order to assure the nestedness of the sequence of meshes. It is worth noting that in this new form, some *proper nodes* of each current level at derefining are not eligible for checking the derefinement condition.

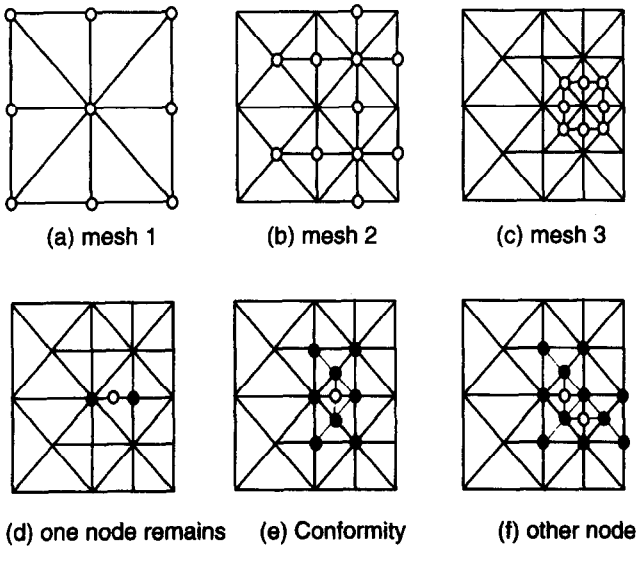

Fig. 2. Conformity of the arising new level by means of the new version of the algorithm.

Finally, we remark that there are some differences between our derefinement procedure and the one proposed by Rivara. Briefly, we can say that, although the definitions of *j-new node* and *j-new edge* used by Rivara are equivalent to our definitions of *proper node* and *internal edge*, we neither use the molecular structure defined by Rivara nor need to distinguish if a particular node has been created as a consequence of the creation of another in some preceding level; our algorithm can be combined with the 2-T or 4-T refinement algorithms and in both cases the size of the derefinement area is determined by the physical problem, not by the algorithm itself.

On the other hand, the derefinement algorithm of Rivara works at the same time as the refinement one, while our algorithms are independent modules that are fixed by the user choosing the particular adaptive strategy. Also, we have introduced a simple criterion for derefining and this derefinement condition shows how better the previous numerical solution was with respect to the solution over the derefinement mesh.

4 ON THE REFINEMENT ALGORITHMS

4.1 A scheme of our first algorithm

Let $T^n = \{\tau_1 < \tau_2 < \cdots < \tau_n\}$ be a sequence of nested triangular grids, where τ_1 represents the initial mesh and τ_n the finest mesh in the sequence. To refine the sequence, or the level n, means to obtain a new sequence:

$$T^{n+1} = \{\tau_1 < \tau_2 < \cdots < \tau_n < \tau_{n+1}\}$$

The version of the refinement algorithm due to Ferragut[7,8] works in the following manner:

INPUT: Sequence $T^n = \{\tau_1 < \tau_2 < \cdots < \tau_n\}$
For each $t \in \tau_n$, do:
1. The refinement condition is evaluated.
2. If t must be refined, then: its edges are marked. Conformity of the new level $n+1$ must be assured.

End for.
3. Conformity of the new level is assured, increasing the refined area.

OUTPUT: Sequence $T^{n+1} = \{\tau_1 < \tau_2 < \cdots < \tau_n < \tau_{n+1}\}$

The way in which the conformity of the arising new level is assured is analogous to that in our former derefinement algorithm, see Section 2.2. Now, for each non-conforming element of the level n, one node must be

introduced in the middle of the largest edge and conformity must be assured again. This implies as many loops in elements of the level at refining as nodes introduced for conformity.

4.2 A scheme based upon the new version

This version is obtained by changing the way in which the conformity of the new level of mesh is assured.

INPUT: Sequence $T^n = \{\tau_1 < \tau_2 < \cdots < \tau_n\}$
For each $t \in \tau_n$, do:
1. The refinement condition is evaluated.
2. If t must be refined, then:
   2(a) Its edges are marked. Conformity must be assured.
   2(b) For each edge of t, F, say, do:
   While conformity must be assured:
   Assure local conformity of the neighbouring element of suitable edge F.
   End while.
   End for.
   End if.

End for.
OUTPUT: Sequence $T^{n+1} = \{\tau_1 < \tau_2 < \cdots \tau_n < \tau_{n+1}\}$

It is worth noting here that the conformity can be assured at the same time as the elements of the level τ_n are taken for the evaluation of the refinement condition. That means that this new version of the refinement algorithm can be understood as the inverse of the derefinement algorithm that is presented in this paper. However, this fact does not imply that this version of the refinement algorithm is better — in the sense of the required number of operations — than the previous one. Both algorithms show a similar computational behaviour.

Note that when one element t must be refined the (local) conformity is assured taking the *neighbouring element* of each edge in which one node has been introduced. If this *neighbouring element* is non-conforming it can be made conforming by adding another node. This procedure ends when no further nodes are put.

5 APPLICATIONS

We present some test applications to show the behaviour of the derefinement algorithm. We consider the convection–diffusion problem defined in a two-dimensional domain Ω, of boundary Γ:

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

where $u = u(\mathbf{x}, t)$ is the solution in a fluid element placed in $\mathbf{x} = x_1\mathbf{i} + x_2\mathbf{j}$ at time t; fluid velocity is given by $\mathbf{v} = \mathbf{v}(\mathbf{x})$; we study the lineal model, $k = k(\mathbf{x}); f = f(\mathbf{x}, t)$ are the external sources. We suppose boundary and initial conditions such that the existence and uniqueness of solution are assured. Some meshes and the respective solutions in different time steps of the following unsteady convection–diffusion problem can be observed in Fig. 3.

Let Ω be a circular domain centred at the point $(0{\cdot}5, 0{\cdot}5)$ of diameter $\sqrt{2}$, with null Neumann conditions on its boundary Γ. We study the problem with a rotating velocity field, with $v_1 = \omega(x_2 - 0{\cdot}5)$ and $v_2 = -\omega(x_1 - 0{\cdot}5)$ such that $\nabla \cdot \mathbf{v} = 0$ and tangent to Γ. The maximum value of the velocity field is $\sqrt{2}\omega/2$ and it is taken in the boundary Γ of Ω, and the minimum is taken in the centre of the domain. In the present application we have chosen $\omega = 10^3$. The value of the diffusion coefficient is $k = 1$. That is, a significance Peclet number for this problem is about 10^3. We suppose no external sources, $f = 0$.

The initial solution is a given function that is automatically approximated in Fig. 3(a) using an adaptive strategy. This strategy combines four global refinement followed by a derefinement with $\epsilon = 0{\cdot}005$, twice and enables us to get both a good approximation of the initial solution and of the geometry of the domain with a minimum number of nodes, 1573 in this case.

We have used as error indicator $\eta_i = h_i^2 |\nabla u_h|$ for an element Ω_i, where h_i is the diameter of Ω_i and u_h the linear numerical solution in the triangular element. If we denote by $E_{\max}$ the maximum value of the error indicators, then, an element Ω_i will be refined if and only if $\eta_i \geq \gamma E_{\max}$, where γ is called the parameter of refinement. In this example we allow values of γ between $0{\cdot}5$ and $0{\cdot}9$. The exact value is automatically calculated using the following expression:

$$\gamma = \min(\max(0{\cdot}5, NN/NOPT), 0{\cdot}9)$$

where NN is the number of nodes in the current level of mesh and $NOPT$ the optimum number of nodes fixed by the user. In this application we have considered $NOPT = 2000$ and $\epsilon = 0{\cdot}001$ at derefining.

It is worth noting that the number of nodes and their location in the mesh is automatically controlled by a small number of parameters and by the numerical solution. The value of these parameters is very important in solving a problem. Some quick trials may be needed before these parameters can be fixed by the user.

To get the stationary solution (constant in this example), a total number of 1468 time steps have been calculated. The adaptive strategy was three refinements followed by a derefinement procedure with increment of time after each refinement or derefinement. The stability conditions obtained in Ref. 13 have been considered to evaluate the time increment after each refinement or derefinement. In each time step, one multigrid iteration is enough to solve the system of equations.

Once the stationary state is reached, the derefinement algorithm reduces the sequence of nested meshes to

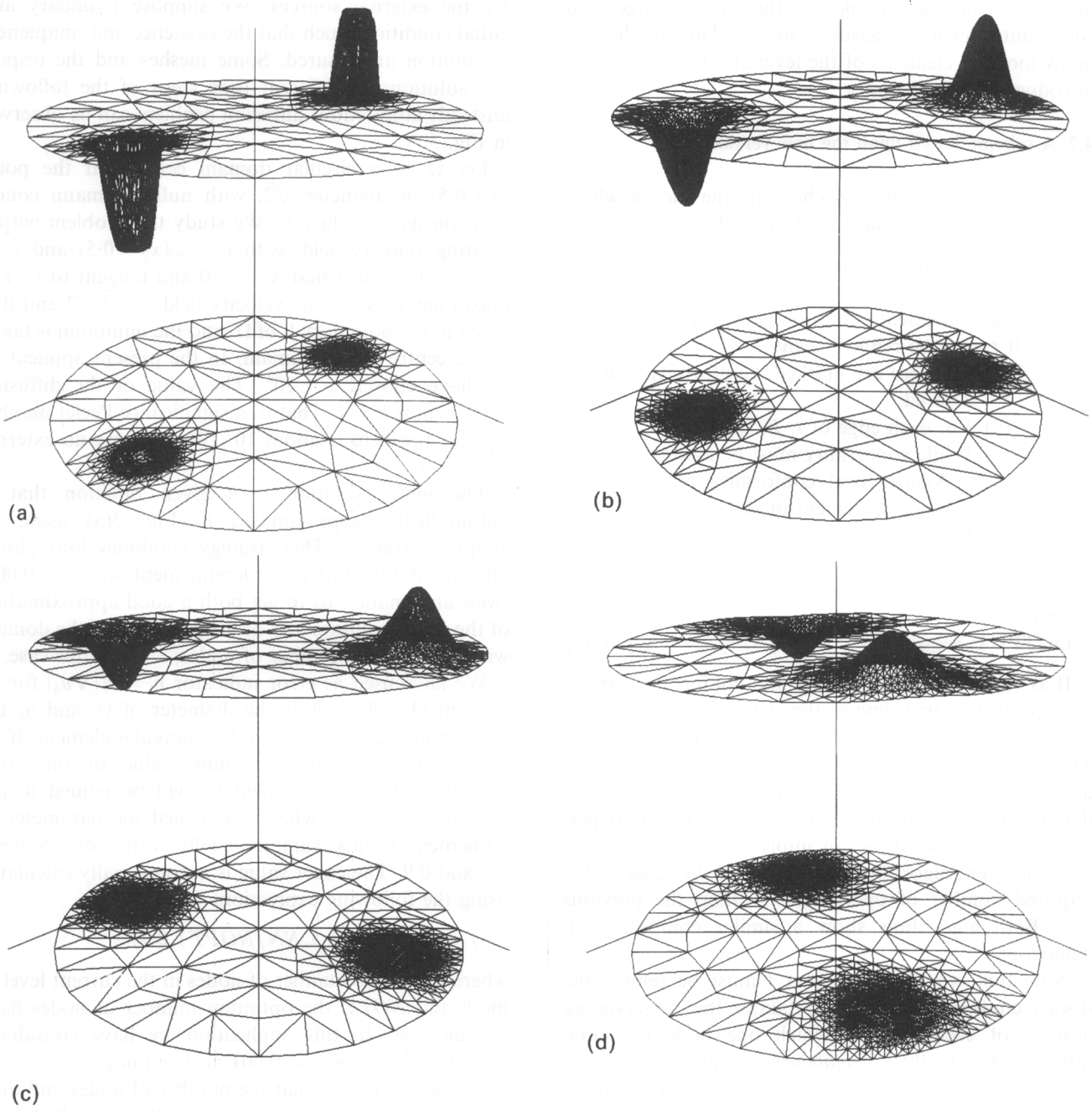

Fig. 3. Evolution of the solution for an unsteady convection–diffusion problem: (a) initial solution, $t = 0{\cdot}0$ s, 1573 nodes; (b) 240 time steps, $t = 0{\cdot}00047$ s, 2115 nodes; (c) 480 time steps, $t = 0{\cdot}00116$ s, 1923 nodes; (d) 720 time steps, $t = 0{\cdot}0021$ s, 1873 nodes; (e) 960 time steps, $t = 0{\cdot}0035$ s, 1383 nodes; (f) 1200 time steps, $t = 0{\cdot}0055$ s, 449 nodes.

just one coarse level and the program run stops automatically.

6 CONCLUSIONS

The study of the complexity of an algorithm is very useful in improving versions. In this paper this improvement is shown for a particular algorithm: a derefinement algorithm of two-dimensional nested grids.

The use of a new vector of neighbouring elements of grid-edges reduces the algorithm complexity. Complexity is further reduced by simultaneous checking of both the derefinement condition and conformity of the arising new sequence. Our derefinement procedure departs significantly from the one proposed by Rivara, and features improve the computational behaviour of our former algorithm.

This study will be very important in more difficult algorithms, for example, analogous ones to refine and derefine in three dimensions. There are already some

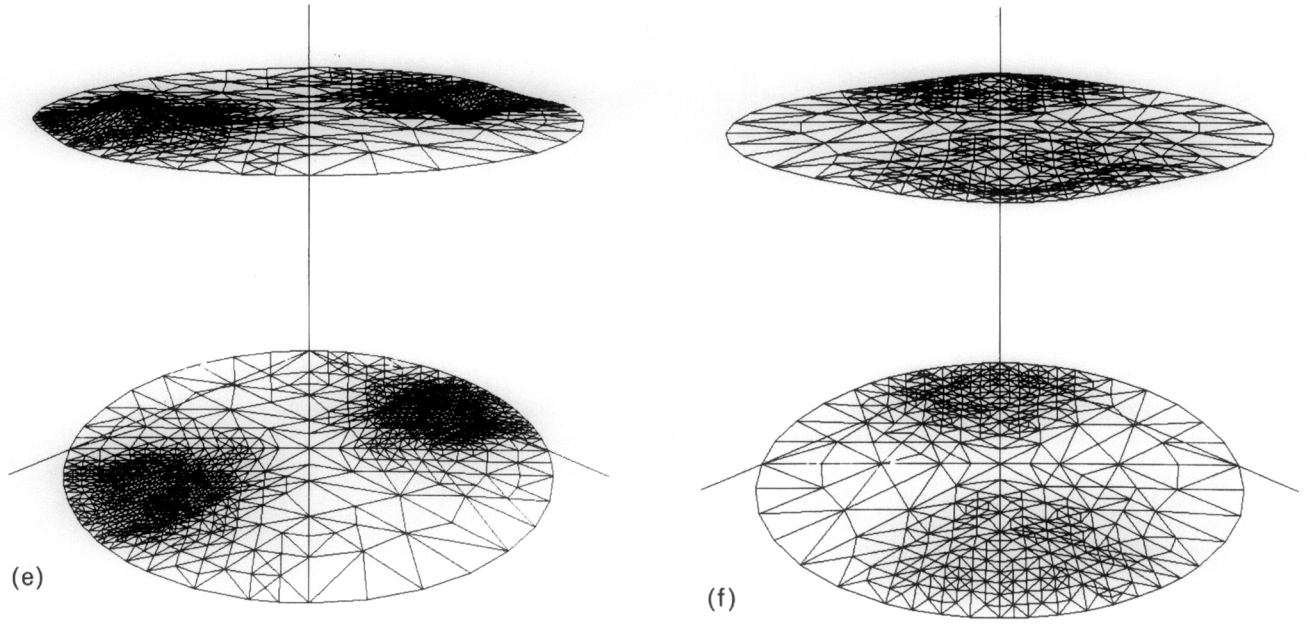

Fig. 3. Continued.

papers,[14,15] in this direction of research, but it is not yet clear whether they are simple generalizations of Rivara's algorithms.[12]

ACKNOWLEDGEMENT

We would like to express our acknowledgements to M. C. Rivara for her kind comments and suggestions.

REFERENCES

1. Plaza, A., Montenegro, R. & Ferragut, L., An adaptive refinement/derefinement algorithm of structured grids for solving time-dependent problems. In *Numerical Methods in Engineering*, eds Ch. Hirsch *et al.* Elsevier Science, Amsterdam, 1992, pp. 225–232.
2. Plaza, A., Ferragut, L. & Montenegro, R., Derefinement algorithms of nested meshes. In *Algorithms, Software, Architecture*, ed. J. van Leeuwen. IFIP, Elsevier Science, Amsterdam, 1992, pp. 409–415.
3. Plaza, A., Derefinement algorithms of nested meshes (in Spanish). PhD thesis, University of Las Palmas de Gran Canaria, 1993.
4. Ferragut, L., Montenegro, R. & Plaza, A., Efficient refinement/derefinement algorithm of nested meshes to solve evolution problems. *Commun. Numer. Meth. Engng*, 1994, **10,** 403–412.
5. Kashiyama, K. & Okada, T., Automatic mesh generation method for shallow water flow analysis. *Int. J. Numer. Meth. Fluids*, 1992, **15,** 1037–1057.
6. Zienkiewicz, O. C. & Zhu, J. Z., Adaptivity and mesh generation. *Int. J. Numer. Meth. Engng*, 1991, **32,** 783–810.
7. Ferragut, L., Una solución al problema de la programación de métodos de elementos finitos autoadaptativos. *Anales Ing. Mec.* 1987, **5,** 201–206.
8. Ferragut, L., Neptuno, un sistema de elementos finitos autoadaptativo. Depto. Matemáticas Aplic. y Métodos Informáticos, Madrid, 1987.
9. Rivara, M. C., A grid generator based on 4-triangles conforming. Mesh-refinement algorithms. *Int. J. Numer. Meth. Engng*, 1987, **24,** 1343–1354.
10. Rivara, M. C., Selective refinement/derefinement algorithms for sequences nested triangulations. *Int. J. Numer. Meth. Engng*, 1989, **28,** 2889–2906.
11. Rivara, M. C., Local modification of meshes for adaptive and/or multigrid finite-element methods. *J. Comput. Appl. Math.*, 1991, **36,** 79–89.
12. Rivara, M. C. & Levin, C., A 3-D refinement algorithm suitable for adaptive and multigrid techniques. *Commun. Appl. Numer. Meth.*, 1992, **8,** 281–290.
13. Montenegro, R. *et al.*, Application of adaptive finite elements methods to convection–diffusion problems in 2-D (in Spanish). *Rev. Int. Met. Num. Cal. Dis. Ing.*, 1989, **5,** 535–560.
14. Bänsch, E., Local mesh refinement in 2 and 3 dimensions. *IMPACT Commun. Sci. Engng*, 1991, **3,** 181–191.
15. Bornemann, F. *et al.*, Adaptive multilevel methods in three space dimensions. *Int. J. Numer. Meth. Engng*, 1993, **36,** 3187–3203.