Available online at www.sciencedirect.com
SCIENCE DIRECT®
Applied Numerical Mathematics 51 (2004) 565–579
APPLIED NUMERICAL MATHEMATICS
www.elsevier.com/locate/apnum

Adaptive techniques for unstructured nested meshes

Miguel A. Padrón [a], José P. Suárez [b], Ángel Plaza [c,*]

[a] *Department of Civil Engineering, University of Las Palmas de Gran Canaria, 35017, Spain*
[b] *Department of Cartography and Graphic Engineering, University of Las Palmas de Gran Canaria, 35017, Spain*
[c] *Department of Mathematics, University of Las Palmas de Gran Canaria, 35017, Spain*

Available online 12 August 2004

Abstract

The purpose of this paper is twofold. First we introduce improved versions of our algorithms for refining and coarsening 2D and 3D nested triangular and tetrahedral grids, and secondly the application of these algorithms in the simulation of 2D and 3D problems, is demonstrated. A key idea of the algorithms is the use of the topological concept of the skeleton of a triangulation in two or three dimensions in order to reduce the dimension of the refinement problem in a natural hierarchic manner.

Improved skeleton based refinement (SBR) algorithms and their counterpart, the skeleton based derefinement (SBD) algorithms are described in this study. The algorithms are fully automatic and are applied here to a 2D boundary value problem, a 3D approximation problem with a large gradient, a geometric shape modeling problem and a simulation evolution problem in 3D.

Keywords: Refinement; Coarsening; Grid generation; Skeleton graph

1. Introduction

Adaptive mesh refinement strategies are essential to accurate efficient approximation of geometrical domains, approximation of functions and approximate solution of partial differential equations. Adaptivity of the mesh is particularly important in three-dimensional problems because the problem size and computational cost grow very rapidly as the mesh size is reduced. The elements that offer the simplest choice and greatest flexibility are simplices: triangles in two dimensions and tetrahedra in three dimensions. Many different refinements strategies and improvement techniques for two- and

* Corresponding author.
E-mail address: aplaza@dmat.ulpgc.es (Á. Plaza).

0168-9274/$30.00
doi:10.1016/j.apnum.2004.06.010

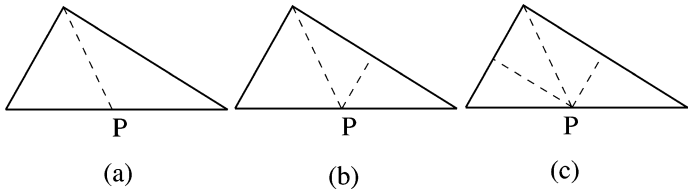


Fig. 1. Patterns for dividing a triangle: (a) Longest-edge bisection; (b) Division in three triangles; (c) 4T-LE partition.

three-dimensional triangulations have been investigated. For example, see the extensive bibliographies in [6,12,13].

Even fractal concepts and iterated function systems have been employed [5,23]. Refinement and coarsening algorithms are also relevant in automatic mesh generation, linear approximation of functions, and data mining for computer visualization [7,9].

In 2D, the longest-edge bisection of a triangle t is the partition of the triangle by the midpoint of its longest-edge and the opposite vertex. The 4-triangles longest-edge (4T-LE) partition of a triangle t is defined by joining the midpoint of the longest-edge of t to the opposite vertex, and then, joining that midpoint to the midpoints of the other two edges of t by segments parallel to the edges (Fig. 1(c)). The longest-edge neighbor of t is the neighboring triangle t^* which shares with t the longest-edge of t. The longest-edge propagation path (LEPP) of a triangle t as defined by Rivara et al. [31], will be the ordered list of all the triangles $t_0, t_1, t_2, \ldots, t_{n-1}, t_n$ such that t_i is the neighbor triangle of t_{i-1} by the longest edge of t_{i-1}, for $i = 1, 2, \ldots, n$.

Our refinement algorithm is equivalent to the 4T longest-edge algorithm of Rivara [28–30]. One of the issues of these algorithms is the conformity of the mesh. Let Ω be a bounded set in R^d, $d = 2, 3$ with non-empty interior, $\mathring{\Omega} \neq \emptyset$, and polygonal boundary $\partial\Omega$, and consider a partition of Ω into a set $\tau = \{t_1, \ldots, t_n\}$ of triangles or tetrahedra, such that any adjacent elements share an entire face or edge or a common vertex, i.e. there are no *non-conforming* nodes in τ. Then we can say that τ is a conforming triangular or tetrahedral mesh or a conforming triangulation for Ω. In order to assure the conformity of the meshes additional nodes are included, based on the LEPP.

Figs. 1(a) and (b) show the two patterns used for the conformity at local refinement. Note that in the conformity step the algorithms always put a node at the midpoint of the longest-edge of the non-conforming triangle. In this way a kind of non-desirable *'domino effect'* could be presented and the division of a unique triangle could lead us to divide the entire triangulation [16]. This is an issue which is not addressed in this paper, but fortunately it has been solved in the sense that when the 4T-LE algorithm is iteratively applied to an initial triangulation, with N the number of new nodes introduced in the refined area, then the number of nodes located outside the refinement area is of the order $N^{1/2}$, while the number of nodes in the refinement area is of order the N [32,35].

In 3D, Plaza and Carey [24,25] have presented a generalization of the 4-T longest-edge Rivara algorithm, called skeleton based refinement (SBR) algorithm. The algorithm works first on the triangular faces of the tetrahedra, the skeleton of the 3D-triangulation, and then subdivides the interior of each tetrahedron in a *consistent* manner with the subdivision of the skeleton. As in the refinement case, the coarsening algorithm is based on the skeleton where the conformity of the mesh is assured and then the interior of the tetrahedra is reconstructed [22,27].

It should be noted that there are other similar algorithms for local tetrahedral refinement, like those by Bänsch [2], Kossaczký [19], Maubach [20], Arnold and Mukherjee [1] or Mukherjee [21]. They are not

however, purely longest-edge based algorithms, and hence they are not generalization to 3D of the 4T-LE refinement algorithm. For a more detailed comparison among these algorithms see [25].

Here, we are using the skeleton based data structure [34], and we also show the applicability of a suitable combination of refinement and coarsening for solving different problems: Solution of a non-linear 2D Dirichlet problem, approximation of a scalar function with a large gradient, automatic mesh generation for a curve domain, and finally a simulation evolution problem in 3D.

2. Basic definitions and preliminaries

The skeleton concept from computational geometry [4] is useful for the definition of a suitable data structure for the refinement and coarsening algorithms.

Definition 1 (*Skeleton of a triangulation*). Let τ be an n-simplicial mesh. The set $\mathrm{skt}(\tau) = \{f\colon\ f$ is an $(n-1)$-face of some $t \in \tau\}$ will be called *the skeleton* or the $(n-1)$-skeleton of τ.

For instance, the skeleton of a triangulation in three dimensions is comprised of the faces of the tetrahedra, and in two dimensions the skeleton is the set of the edges of the triangles.

A natural graph based data structure [33] follows from the skeleton of a mesh.

Definition 2 (*k-skeleton graph $G^k(P, L)$*). Let τ be an n-simplicial mesh with the corresponding hierarchy of k-skeleton sets, $k < n$. The k-skeleton undirected graph $G^k(P, L)$ of τ is built as follows: Let $P = \{p_0, p_1, \ldots, p_r\}$ so that p_i is a k-face of τ, and $L = \{l_0, l_1, \ldots, l_s\}$ so that l_j represents the topological adjacency relationship **R** between two different nodes of P, that is between two k-faces of τ.

Since the $G^k(P, L)$ data structure has been designed efficiently to manage and store the skeleton and the longest-edge paths in a mesh, it provides a suitable data structure to be implemented in the 2D and 3D refinement and derefinement algorithms developed here. This $G^k(P, L)$ data structure is compared in Table 1 to three other general purpose 2D data structures reviewed in [18,33,36]: winged-edge [3], half-edge and quad-edge. The comparison considers five critical features of data structures. The first feature is the topological adjacency in the structure and is denoted by edge to vertex (EV), edge to edge (EE), edge to face (EF) and face to face (FF). The higher dimension feature indicates if the data structure extends immediately to higher dimension meshes (3D, 4D, ...). The storage cost is estimated in terms of the number of vertices, faces and elements, V, F, E respectively, that are used in the adjacencies. Finally the computational costs in accessing any particular adjacency are estimated from the data structures.

Table 1
Comparison to other data structures

Data structure	Adjacencies	Higher dim.	Storage cost	Time cost
Winged-edge	EV, EE, EF	Yes	$\mathcal{O}(V+F+E)$	K
Half-edge	EV, EE, EF	Yes	$\mathcal{O}(V+F+E)$	K
Quad-edge	EE, EF	No	$\mathcal{O}(V+F+E)$	K
G^1-G^2	EE, FF, EF	Yes	$\mathcal{O}(V+E+F)$	K

Although the $G^k(P, L)$ data structure has been designed for the skeleton based algorithms, Table 1 shows that the proposed data structure is competitive compared to the reported data structures.

3. The 2D-SBR algorithm

In two dimensions our refinement algorithm can be viewed as a version of the 4T longest-edge (4T-LE) refinement algorithm of Rivara. The algorithm works first on the skeleton of the 2D triangulation, (the wire frame comprised by the edges of the triangulation) and then it subdivides the interior of the triangles to obtain finally the new refined mesh. We classify the edges of each triangle into two types: The longest edge named Type 1, and the remaining two edges, named Type 2, and this idea can be expressed in terms of the 1-skeleton graph introduced previously by considering the relation $\mathbf{R} =$ *"length of edge x in the triangle is less than length of the longest edge"*. Because **R** is an order relation between edges, it is possible to add an orientation to the links in the graph G^1 that represent this ordering. Hence, the links in the directed G^1 graph in Fig. 2 indicate the dependency of the edges when refining to enforce conformity.

Remark 1. Updating the 1-skeleton directed graph data structure by applying 4T-LE bisection affects only up to three graph nodes in each refinement step.

Remark 2. Let τ be a two-dimensional conforming triangulation with N triangles and t an arbitrary triangle of τ. Let e be the longest-edge of t. Then the LEPP of t can be obtained from a depth-first

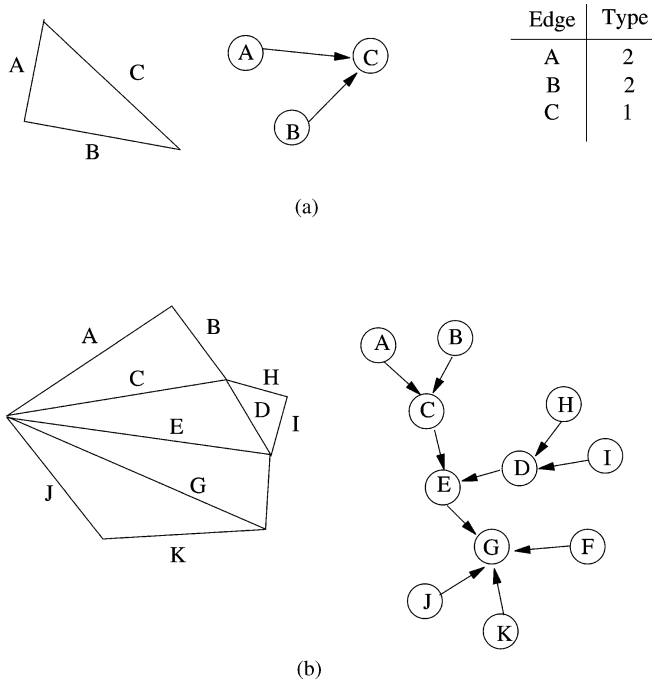


Fig. 2. 1-skeleton oriented graph in 2D: (a) Triangle, its 1-skeleton graph, and the classification of its edges; (b) Triangulation τ and its 1-skeleton graph $G^1(\tau)$.

traversal of the 1-skeleton directed graph of τ starting from the node e of the graph, with a maximum cost of order $\mathcal{O}(N-1)$.

Numerical experiments carried out in [34] reveal that asymptotically the average cost per triangle of traversing the LEPP approaches a small constant. This result is important since it is very relevant to the utility of the longest edge bisection schemes and their efficient use of memory. The 2D-SBR algorithm when used to uniformly refine a mesh is of linear complexity with respect to the number of nodes.

Algorithm 2D-SBR(τ, t_0).
/* Input variables: τ, 2D triangular mesh, t_0 triangle to be refined
/* Output variables: new mesh τ
/* *Edge subdivision*
$L =$ 1-skeleton(t_0)
1: For each edge $e \in L$ **do**
Subdivision(e)
End For
/* *The conformity is ensured*
2: For each edge $e_i \in L$ **do**
$S_i = LEPP(e_i, G^1(\tau))$
For each triangle $t_j \in S_i$ **do**
Let e_j be the longest-edge of t_j
Subdivision (e_j)
End For
End For
/* *The subdivision of the triangles is performed*
3: For each triangle $t_j \in \tau$ to be subdivided **do**
Subdivision(t_j)
End For

An example of how the 2D-SBR algorithm works is shown in Fig. 3 in which (a) is the original 2D triangulation, (b) shows the subdivision of the edges of the chosen triangle t_0, in other words, the edges in the 1-skeleton of t_0; (c) is the extension of conformity by subdivision of edges. Note that the triangles, noted as t^* in (b) and (c), belong to the LEPP.

Finally, Fig. 3(d) shows the internal subdivision of the involved triangles.

4. The 3D-SBR algorithm

The 3D-SBR algorithm to refine a unique tetrahedron t_0 belonging to some initial 3D triangulation τ can be described as follows [34]:

Algorithm 3D-SBR(τ, t_0).
/* Input variables: τ, 3D triangular mesh, t_0 tetrahedron to be refined
/* Output variables: new mesh τ
/* *Edge subdivision*

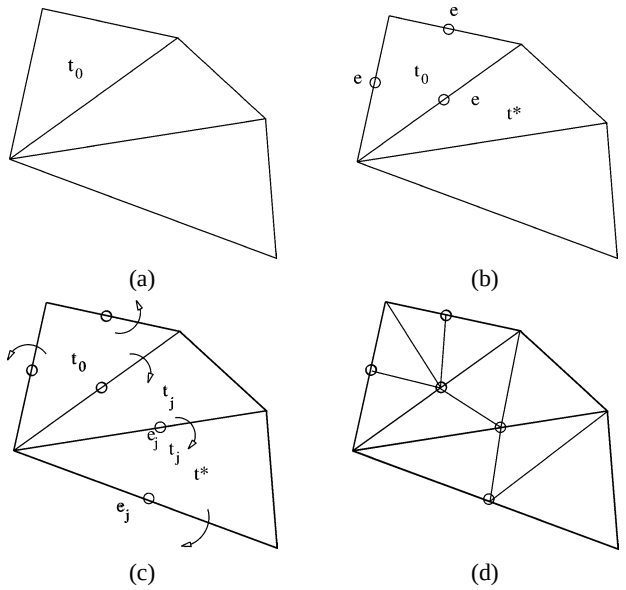


Fig. 3. 2D refinement algorithm: (a) Input mesh and objective triangle t_0; (b) Subdivision of $1 - \text{skt}(t_0)$; (c) Assuring conformity through the edges; (d) Triangle subdivision and output mesh.

$L = 1\text{-skeleton}(t_0)$
1: For each edge $e \in L$ **do**
Subdivision(e)
End For
/* *The conformity is ensured*
2: While $L \neq \emptyset$ **do**
Let e_k be an element $\in L$
For each tetrahedron $t \in \text{hull}(e_k)$ **do**
For each non-conforming face $f \in G^2(t)$ **do**
Let e_i be the longest-edge of f
Subdivision(e_i)
$L = L \cup P_{e_i}$
End For
End For
$L = L - e_k$
End While
/* *The subdivision of the skeleton is performed*
3: For each triangular face $f \in G^2(\tau)$ to be subdivided **do**
Subdivision(f)
End For
/* *The subdivision of the tetrahedra is performed*
4: For each tetrahedron $t \in \tau$ to be subdivided **do**
Subdivision(t)
End For
End

It should be pointed out here that the procedure of *subdivision* subdivides the successive skeletons in reverse order: Steps 1 and 2 perform the subdivision of the edges, step 3 performs the subdivision of the faces, and step 4 performs the interior subdivision to sub-tetrahedra.

The data structure used in the 3D-SBR algorithm follows from Definition 2 and is a face based data structure G^2 that defines the triangular planar faces and their adjacency relations in the current mesh [33]. The points where the 2-skeleton graph $G^2(\tau)$ is used are clearly seen ină the previous algorithm. First, the inner loop of step 2 accesses the non-conforming faces in the mesh and in step 3 the subdivision of the 2-skeleton is performed. In both steps, the 2-skeleton graph provides access to the faces taking part in the refinement in a local and efficient way. In step 2, the procedure hull(e) provides the set of simplices $S \in \tau$ such that $e \in S$. That is, hull(e) is comprised of all the simplices that share the same edge e.

Some of the properties of the previous algorithm, as well as the derefinement algorithms, are summarized later. The computational complexity of the algorithm is simply $O(N_a) + O(N_f) + O(N_t)$ [27], where N_a is the number of added nodes, N_f is the number of involved faces, and N_t the number of involved tetrahedra. Numerical experiments confirm that in practice the algorithm performs with linear complexity [24].

5. The 3D-SBD algorithm

For the 2D version of the coarsening algorithm see Refs. [9,26]. For coarsening a refined mesh we may construct an inverse algorithm based on the previous refinement scheme. We have to take into account at derefining the *genealogy* of the edges, faces or elements. To each topological element of the mesh (node, edge, face or tetrahedron) we assign an integer. This number gives us the level in which the corresponding element has been created. Furthermore, we use the sign of these integers to control the coarsening procedure. That is, if the number is positive, the corresponding topological element must remain. On the other hand, if the number is negative it implies that this element must be removed. The vector of numbers for each type is called a *coarsening vector*. A proper node P is called an *eligible* proper node for derefining if P can be removed from the mesh and the mesh remains conforming. P is *not-eligible* if its removal makes the mesh non-conforming. Hence, the conformity of each level is assured by maintaining some nodes that otherwise, given the coarsening condition, might have been removed. The coarsening condition we are using in 3D is the same used in 2D [9]. A proper node may be removed if the absolute difference between the values at this node of the numerical solution and its corresponding interpolated value is less than a tolerance $\varepsilon > 0$. That is, if u_h is the numerical solution for a given mesh and u_h^i is the interpolated value of u_h in the derefined mesh, we will get

$$\left|u_h(P) - u_h^i(P)\right| < \varepsilon \implies \left\|u_h - u_h^i\right\|_\infty = \sup_x \left|u_h(x) - u_h^i(x)\right| < \varepsilon.$$

It should be noted that the algorithms are independent of the actual choice of refinement or coarsening criteria.

As in the 2D case, the concept of adjacency is a central idea for the algorithm in 3D and also, as in the refinement schemes, hull(e) and G^2 graph store the adjacency relations used by the derefinement algorithm. See [27] for details. The outline of the 3D coarsening algorithm is as follows:

Algorithm 3D-SBD(T, T^m).
/* Input variables: refined sequence of meshes $T = \{\tau_1 < \tau_2 < \cdots < \tau_k\}$
/* Output variables: new derefined sequence $T^m = \{\tau_1 < \tau_2' < \cdots < \tau_m'\}$
/* Loop in levels of T
For $j = k$ to 2 **do**
  For each *eligible proper* node $N \in \tau_j$ **do**
    /* *The coarsening condition is evaluated*
    1: The coarsening condition is checked and nodes and edges are marked
    If node N must remain **then**
      /* *The conformity is ensured locally*
      Let e be surrounding edge of N, and hull(e)
      2: For each tetrahedron $t \in$ hull(e) **do**
        Make N conforming in t
      End For
    End If
  End For
  /* *The sequence of meshes is re-defined*
  3: For each $f \in G^2(\tau_{j-1})$ **do**
    Subdivision(f)
  End For
  4: For each $t \in \tau_{j-1}$ **do**
    Subdivision(t)
  End For
End For

Therefore:

(1) Both refinement and coarsening procedures can be applied, even to strong non-convex regions, without any preprocessing [22].
(2) The coarsening condition is evaluated in a *minimum* number of nodes: the *eligible proper nodes*.
(3) The nested nature of the grids make the use of multigrid methods relatively easy [9,14].
(4) The idea of using the skeleton might be of interest in relativistic applications and other problems involving even higher dimensions.

6. An example of a non-linear Dirichlet problem in 2D

In the first example the refinement algorithm in 2D is used to illustrate an adapted mesh for solving a nonlinear partial differential equation. The 2D-SBR algorithm and the graph based data structure have been implemented in Matlab^{TM}. The boundary value problem is as follows:

$$-\Delta u + u \cdot u_x = f \quad \text{in } \Omega, \qquad u = u_{\text{ex}} \quad \text{in } \Omega \tag{1}$$

on $[0, 1] \times [0, 1]$ with $f(x, y)$ such that $u_{\text{ex}} = 10x^9$ is the exact solution of the problem.

We use the error indicator;

$$E(K) = \alpha \left\| h(f - au) \right\|_K + \beta \left(\frac{1}{2} \sum_{e \in \delta_K} h_e^2 [\mathbf{n_e} \cdot c \nabla u_h]^2 \right)^{1/2} \tag{2}$$

for element K where $\mathbf{n_e}$ is the unit normal vector to edge e and coefficients α and β are independent of the current triangulation. The final term in brackets is the jump in flux across the element edge and the L_2 norm of the interior residual is computed over the element K. Triangles are selected to be refined in each intermediate mesh when $E(K) > \frac{1}{2} \max_K (E(K))$.

In this example we include to the refinement algorithm a simple node movement strategy which improves the quality of the refined meshes. It consists in moving each interior point toward the center of mass of the polygon formed by the adjacent triangles. Fig. 4(a) shows the evolution of the maximum error for each intermediate stage of the solution. Fig. 4(b) shows a comparison of CPU time of the solver and the respective time cost to refine each intermediate mesh. The initial mesh Fig. 4(c), is a Delaunay type mesh with 309 triangles and 175 nodes. In Fig. 4(d) it is depicted a local refined mesh at stage 3 with 7315 triangles and 3779 points. The final mesh with 27676 triangles is completed after six refinement stages. Using a relative tolerance of 5×10^{-7} the adaptation finishes after six iterations.

Additional test problems in 2D can be found for example in [9,26].

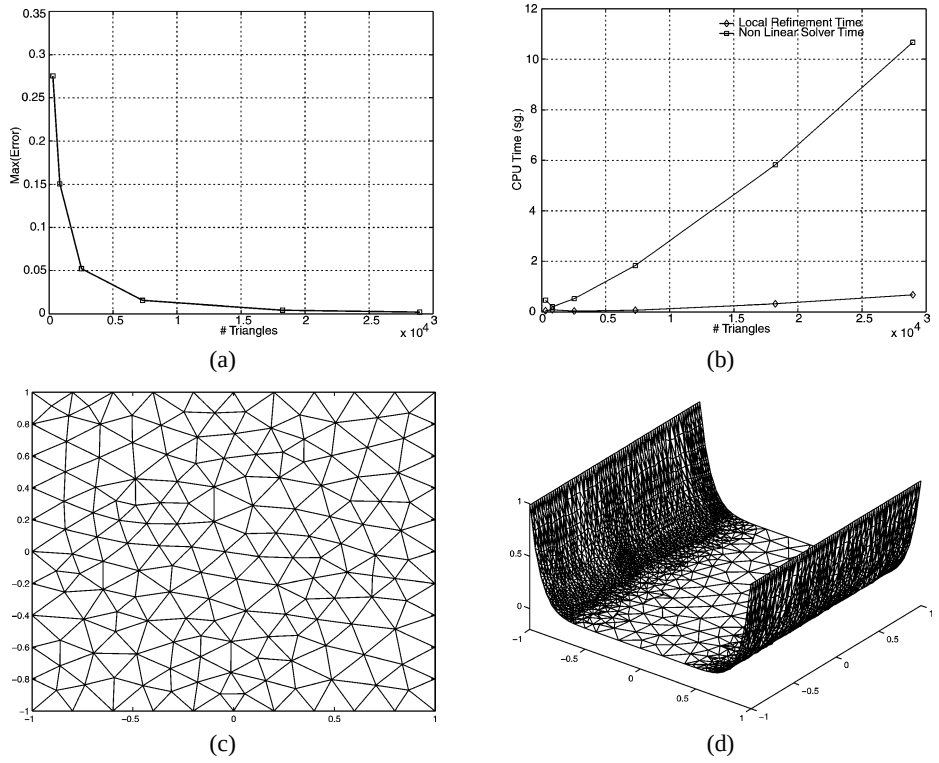
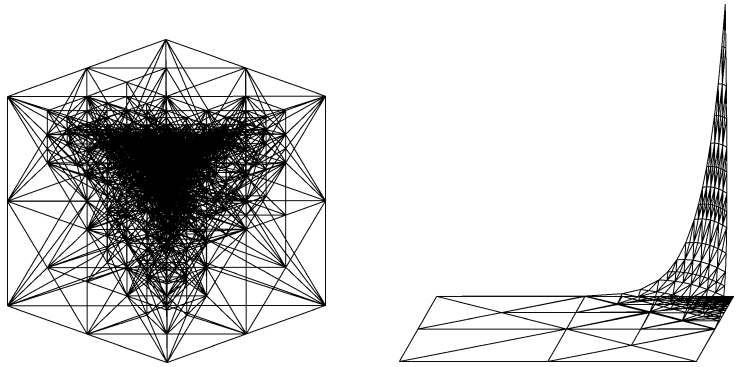


Fig. 4. 2D approximated solution of a non-linear Dirichlet problem: (a) Evolution of error maximum; (b) Solver and refinement time costs; (c) Initial mesh: 309 triangles; (d) Local refinement at stage 3.



(a) (b)

Fig. 5. Approximation of function $u(x, y, z) = (xyz)^{10}$: (a) Mesh with 13723 tetrahedra; (b) Values of the approximation on a square face.

7. A scalar function approximated by 3D-SBR/3D-SBD algorithms

This example shows how the refinement and coarsening algorithms can be used to obtain an "optimal" mesh in 3D for approximating a function with large gradients. For this purpose we have chosen the following function:

$$u(x, y, z) = (xyz)^{10}. \tag{3}$$

Function $u(x, y, z) = (xyz)^{10}$ and its derivatives are smooth, and the highest variation of u is presented in the neighborhood of $(1, 1, 1)$. Fig. 5(a) shows the final mesh obtained after performing 3 local refinements with parameter $\gamma = 0.75$ following by a coarsening with parameter $\varepsilon = 4 \times 10^{-4}$ and absolute difference as coarsening criteria. We repeat that process three times. We defined γ as a parameter meaning that all the tetrahedra with an error greater than 75% must be refined. As an error function distribution we are supposing that per node P the error function is $e(P) = u(P)$, then the error per tetrahedral element is calculated by integrating the linear approximation of the function u, u_i, over each element, by the formula

$$e(T) = \int_T u_i \, \mathrm{d}v = u(\bar{x}_T) \cdot \mathrm{vol}(T),$$

where $\bar{x}_T$ is the baricentric point of T. Note that the large gradient at the corner is well approximated by an optimal tetrahedral mesh. Fig. 5(b) shows the values of the function u on a side of the cube.

8. Approximation of 3D curve domains

In this example we show the applicability of the refinement and coarsening algorithms in order to automatically generate a mesh for a domain with curved boundary contour in 2D or surface in 3D. This is an issue of importance in the area of CAD/CAM modeling of geometry. Here we do not deal with the measurement of the error [8,11,15,17]. Note, however, that the use of nested meshes gives us a naturally hierarchical structure suitable for surface approximation in a similar way to that in [10].

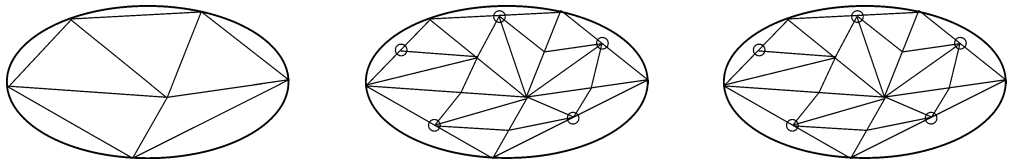


Fig. 6. Approximation of a curve domain in 2D: (a) Initial mesh; (b) Refined mesh; (c) Projection of new.

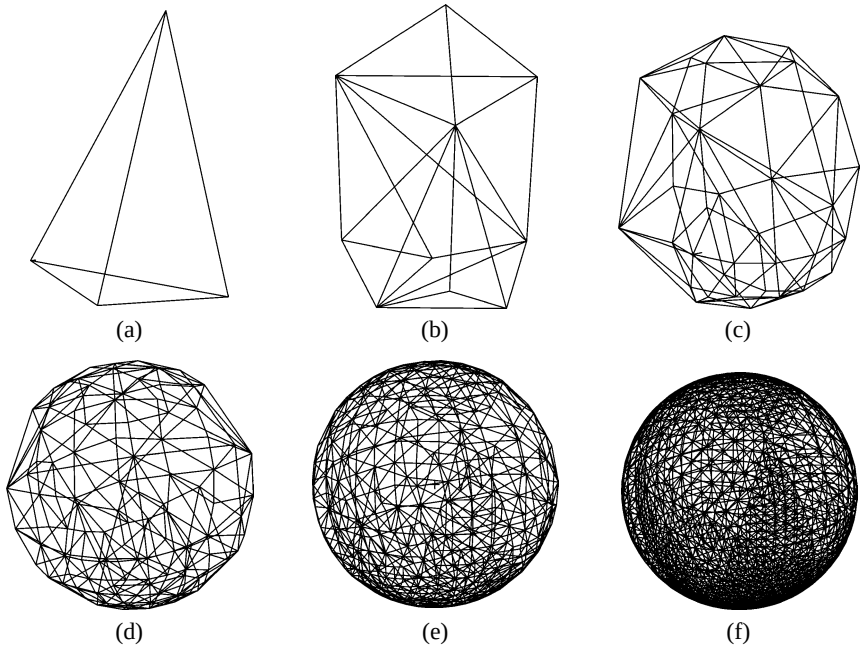


Fig. 7. An example of approximation of surfaces.

The idea of extending (local) refinement to curve approximation is well known and is illustrated in Fig. 6. It involves projection of new nodes to the curved boundary. In Fig. 7 we present a similar example for 3D approximation of a sphere surface, where our initial mesh has only the four triangular faces of a tetrahedron, and the refinement/projection scheme progressively improves the surface approximation. Basically, the points that previously were at the midpoint of an edge of a triangle, have been projected on the surface, in this case a sphere. For this specific problem, we have used global refinement in every one of the five steps up to arrive to the final figure, see Fig. 7(f) with 2050 nodes and 4096 triangles.

9. Simulation of a plane moving front

We present here the simulation of a 3D front-propagation in order to demonstrate and test the performance of the combined refinement and coarsening algorithms for this class of behavior. As the front moves across the domain, the tetrahedral grid is adaptively refined in the zone the front is invading, and it is derefined in the zone the front is vacating.

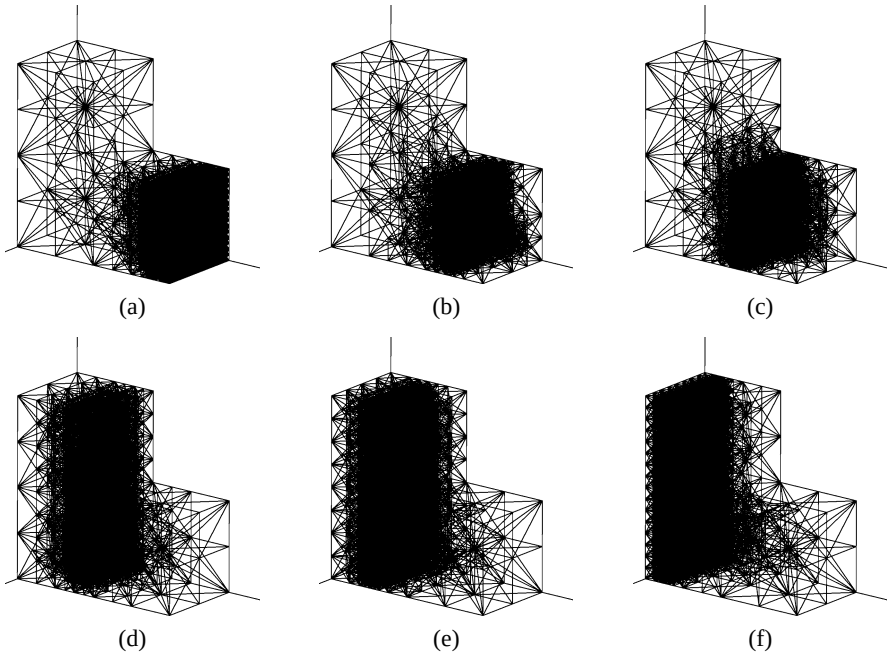


Fig. 8. Simulation example in 3D dimensions of a plane moving singularity: (a) 6372 n. & 32302 t.; (b) 1499 n. & 7206 t.; (c) 3669 n. & 18837 t.;' (d) 1678 n. & 8284 t.; (e) 3926 n. & 20443 t.; (f) 6306 n. & 32659 t.

It is worth noting that we are not solving a PDE problem in this test. In this simulation example the front is defined *a priori* by a horizontal plane which is moving across the domain. At refining, we have used as an error function, the error distribution per node in the domain, $E(P) = \frac{1}{\delta+\|\vec{x}-\vec{x}_0(t)\|^2}$ with $\vec{x}_0(0) = \vec{0}$ in order to better localize the tolerance achieved. Once the front is defined *a priori*, the error per tetrahedral element is computed by integrating the linear interpolant E_I, over each element. For each tetrahedron T we have

$$e(T) = \int_T E_I \, \mathrm{d}v = E(\bar{x}_T) \cdot \mathrm{vol}(T)$$

where $\bar{x}_T$ is the baricentric point of T.

The mesh is locally refined by selecting for further subdivision those elements with error greater or equal than the 48 per cent of the maximum error per element in the mesh. At derefining, we have used as numerical solution the same function as the previous error distribution per node in the domain.

At the beginning, the front is at the plane $y = 2$ of the L-shaped domain Fig. 8(a), and we make three global refinements and two local refinements in our test, in order to capture the front. Then, the location of the front is moved, that is we suppose that the front is now at the plane $y = 1.75$ and then, a derefinement procedure is applied in order to remove nodes. In this way we simulate the time dependency behavior of a problem, see Fig. 8. Once the front has moved we apply again three local refinements with parameter $\gamma = 0.48$ and the front is moved to the plane $y(n) = y(n-1) - 0.25$, followed by a derefinement procedure with parameter $\varepsilon = 0.6$ is applied. This process is repeated a number of "time-steps" until the front arrives to the plane $y = 0$, see Fig. 8(f).

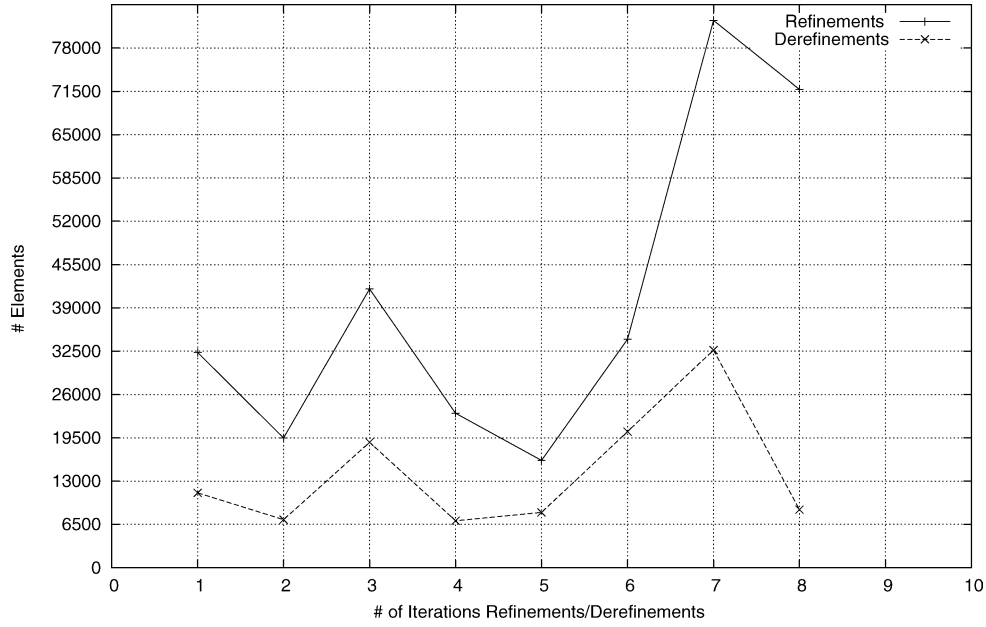


Fig. 9. Tetrahedra evolution for the 3D evolution problem.

With this combination the refinement region moves as the front does, while the number of nodes and tetrahedra remain bounded through the entire process, see Fig. 9.

10. Conclusions

In this paper the skeleton based refinement (SBR) algorithms and their counterpart, the skeleton based derefinement algorithms have been presented and the main properties have been pointed out. The algorithms are fully automatic and they exhibit linear complexity in the number of nodes and elements involved. The skeleton have been also very useful in the derivation of a graph based data structure employed in the implementation of the algorithms.

The refinement and coarsening algorithms presented here provide a useful tool for the simulation of 2D and 3D problems using adaptive triangular and tetrahedral grids. With the refinement/coarsening combination, sequences of nested meshes are obtained, and the multigrid method may be used conveniently to solve associated systems of equations efficiently. As an example, we have solved a nonlinear Dirichlet 2D problem of a square domain. In 3D the simulation of an evolution problem with a moving front is demonstrated. These ideas are clearly relevant in other areas such as visualization, data compression and solid modeling. In this context a simple example of surface approximation has been shown.

Acknowledgements

This research has been partially supported by Gobierno de Canarias grant PI2003/35, and lay Universidade de Las Palmas de Gran Canaria grant UN-2003/16.

References

[1] D.N. Arnold, A. Mukherjee, L. Pouly, Locally adapted tetrahedral meshes using bisection, SIAM J. Sci. Comput. (2) 22 (1997) 443–448.
[2] E. Bänsch, Local mesh refinement in 2 and 3 dimensions, IMPACT Comput. Sci. Engrg. 3 (1991) 181–191.
[3] B.G. Baumgart, A polyhedron representation for computer vision, in: AFIPS Natl. Computer Conf., Anaheim, CA, 1975, pp. 589–596.
[4] M. Berger, Geometry, Springer, Berlin, 1987.
[5] S.W. Bova, G.F. Carey, Mesh generation/refinement using fractal concepts and iterated function systems, Internat. J. Numer. Methods Engrg. 33 (1992) 287–305.
[6] G.F. Carey, Computational Grids: Generation, Refinement and Solution Strategies, Taylor and Francis, London, 1997.
[7] G.F. Carey, A.I. Pehlivanov, R.C. Mastroleo, R. McLay, Y. Shen, V. Carey, R. Dutton, Hierarchic visualization and parallel computation, in: Proceedings High Performance Computing '98, 1998, Boston, MA.
[8] J.C. Cuillière, An adaptive method for the automatic triangulation of 3D parametric surfaces, Comput. Aided Design 30 (1998) 139–149.
[9] L. Ferragut, R. Montenegro, A. Plaza, Efficient refinement/derefinement algorithm of nested meshes to solve evolution problems, Comm. Numer. Methods Engrg. 10 (1994) 403–412.
[10] L. de Floriani, G. Falcidieno, G. Nacy, C. Pieno, A hierarchical structure for surface approximation, Comput. & Graph. 31 (2) (1984) 183–193.
[11] V. François, J.C. Cuillière, Automatic mesh pre-optimization based on the geometric discretization error, Adv. Engrg. Software 31 (2000) 763–774.
[12] P.J. Frey, P.L. George, Maillages. Application aux Éléments Finis, Hermes, Paris, 1999.
[13] P.L. George, Automatic Mesh Generation, Willey, New York, 1991.
[14] W. Hackbush, Multigrid Methods and Applications, Springer, Berlin, 1985.
[15] E. Hartmann, Numerical parametrization of curves and surfaces, Comput. Aided Geom. Design 17 (2000) 251–266.
[16] M.T. Jones, P.E. Plassmann, Parallel algorithms for adaptive mesh refinement, SIAM J. Sci. Comput. 18 (1997) 686–708.
[17] K. Kase, A. Makinouchi, T. Nakagawa, H. Suzuki, F. Kimura, Shape error evaluation method of free-form surfaces, Comput. Aided Design 31 (1999) 495–505.
[18] L. Kettner, Using generic programming for designing a data structure for polyhedral surfaces, Comput. Geom. Theory Appl. 13 (1999) 65–90.
[19] I. Kossaczký, A recursive approach to local mesh refinement in two and three dimensions, J. Comput. Appl. Math. 55 (1994) 275–288.
[20] J.M. Maubach, Local bisection refinement for n-simplicial grids generated by reflection, SIAM J. Sci. Statist. Comput. 16 (1995) 210–227.
[21] A. Mukherjee, An Adaptive Finite Element Code for Elliptic Boundary Value Problems in Three Dimensions with Applications in Numerical Relativity, Ph.D. Thesis, Penn. State University, 1996.
[22] M. A. Padrón, A 3D Derefinement Algorithm for Tetrahedral Nested Meshes Based on the Skeleton, Ph.D. Thesis, University of Las Palmas de Gran Canaria, 1999 (in Spanish).
[23] A. Plaza, The fractal behaviour of triangular refined/derefined meshes, Comm. Numer. Methods Engrg. 12 (1996) 295–302.
[24] A. Plaza, G.F. Carey, About local refinement of tetrahedral grids based on bisection, in: 5th Inter. Mesh. Roundtable, Sandia Corporation, 1996, pp. 123–136.
[25] A. Plaza, G.F. Carey, Local refinement of simplicial grids based on the skeleton, Appl. Numer. Math. 32 (2) (2000) 195–218.
[26] A. Plaza, R. Montenegro, L. Ferragut, An improved derefinement algorithm of nested meshes, in: M. Papadrakakis, B.H.V. Topping (Eds.), Advances in Post and Preprocessing for Finite Element Technology, Saxe-Coburg, Edinburgh, 1994, pp. 175–180.
[27] A. Plaza, M.A. Padrón, G.F. Carey, A 3d refinement/derefinement combination for solving evolution problems, Appl. Numer. Math. 32 (4) (2000) 401–418.
[28] M.C. Rivara, Mesh refinement based on the generalized bisection of simplices, SIAM J. Numer. Anal. 2 (1984) 604–613.
[29] M.C. Rivara, A grid generator based on 4-triangles conforming mesh refinement algorithms, Internat. J. Numer. Methods Engrg. 24 (1987) 1343–1354.

[30] M.C. Rivara, Local modification of meshes for adaptive and/or multigrid finite-element methods, J. Comput. Appl. Math. 36 (1991) 79–89.
[31] M.C. Rivara, N. Hitschfeld, B. Simpson, Terminal-edges Delaunay (small-angle based) algorithm for the quality triangulation problem, Comput. Aided Design 33 (2001) 263–277.
[32] M.C. Rivara, M. Venere, Cost analysis of the longest-side (triangle bisection) refinement algorithm for triangulations, Engrg. Comput. 12 (1996) 224–234.
[33] J.P. Suárez, Graph Based Data Structures for Refinement/Derefinement Algorithms Based on the Longest Edge Bisection, Applications, Ph.D. Thesis, University of Las Palmas de Gran Canaria, 2001 (in Spanish).
[34] J.P. Suárez, G.F. Carey, A. Plaza, Graph based data structures for skeleton based refinement algorithms, Comm. Numer. Methods Engrg. 17 (2001) 903–910.
[35] J.P. Suárez, A. Plaza, G.F. Carey, Propagation path properties in iterative longest-edge refinement, in: 12th Inter. Mesh. Roundtable, Sandia Corp., 2003, pp. 79–90.
[36] K. Weiler, Edge-based data structures and concepts for solid modeling in curved-surface environments, IEEE Comput. Graphics Appl. 5 (1) (1985) 21–40.