# Simulation of 3D evolution problems via refinement and derefinement tetrahedral algorithms

Angel Plaza[1] Miguel A. Padrón[2] Graham F. Carey[3]

**Abstract**

In this paper we present the simulation of 3D evolution problems by the use of a three-dimensional refinement/derefinement algorithm for nested tetrahedral grids. The algorithm is based on an adaptive refinement scheme and on an inverse algorithm introduced by the authors. These algorithms work first on the *skeleton* of the 3D triangulation, the set of the triangular faces. Both schemes are fully automatic.

## Introduction

In the area of finite element methods the efficient approximate solution of partial differential equations local refinement is critical. 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 in any dimension are triangles in two dimensions and tetrahedra in three dimensions (simplices). Many different refinements and improvement techniques for two- and three-dimensional triangulations are now available. For a discussion of different techniques for local grid refinement see [4].

In 2D our refinement algorithm is equivalent to the 4T algorithm of Rivara [17, 18, 19]. Figure 1 shows the partition of a triangle into four (a) and the patterns used for local refinement (b) and (c).

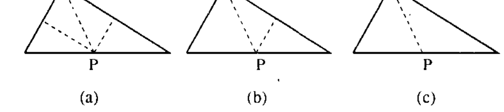

Figure 1: 4-Triangles-refinement patterns

In three dimensions several techniques have been developed in the last five years for refining (and coarsening) tetrahedral meshes by means of bisection of tetrahedra [2], [8, 9], [7], [10, 1]. However, these algorithms are not applicable to any initial mesh and need some kind of pre-processing.

Recently Plaza and Carey [12, 13] have presented a generalization of the 4-T Rivara algorithm to three dimensions. 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 derefinement algorithm is based on the skeleton where the conformity of the mesh in assured, and then the interior of the tetrahedra is reconstructed.

Here, we present the main ideas for a suitable combination of refinement and derefinement.

## Definitions

Let $V = \{X_0, X_1, \ldots, X_m\}$ be a set of $m+1$ points in $R^n$ $(1 \leq m \leq n)$ such that $\{\overrightarrow{X_0X_i}: 1 \leq i \leq m\}$ is a linearly independent set of vectors. Then the closed convex hull of $V$ denoted by $S =< V >=< X_0, X_1, \ldots, X_m >$ is called an $m$-simplex in $R^n$, while the points $X_0, \ldots, X_m$ are called *vertices of* $S$, and the number $m$ is said to be the *dimension* of $S$.

Let $\Omega$ be a bounded set in $R^n$ with non-empty interior, $\overset{\circ}{\Omega} \neq \emptyset$, and poligonal boundary $\partial\Omega$, and consider a partition of $\Omega$ into a set $\tau = \{t_1, \ldots, t_i\}$ of $n$-simplices, such that any adjacent simplex elements share an entire face or edge or a common vertex, i.e. there are no *non-conforming* nodes in $\tau$. Then we can say that $\tau$ is a conforming simplex mesh or a conforming triangulation for $\Omega$.

Let $\tau$ be an $n$-simplicial mesh. The set $skt(\tau) = \{f \;:\; f$ is an $(n-1)$-face of some $t \in \tau\}$ will be called *the skeleton* or the $(n-1)$-skeleton of $\tau$ [3]. 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.

Two (conforming) triangulations $\tau$ and $\tau^*$ of the same bounded set $\Omega$ are said to be *nested*, and we write $\tau < \tau^*$ if the following condition holds: $\forall t \in \tau,\ \exists\ t_1, \ \ldots\ , t_p \in \tau^*$ such that $t = t_1 \cup \ldots \cup t_p$. We also say that $\tau$ is coarser than $\tau^*$ or that $\tau^*$ is finer than $\tau$.

Since bisection of the elements is used, all the new nodes will appear at the midpoints of edges of previous levels of mesh. We call the *surrounding edge* of a node $N$ the edge in which $N$ is at the midpoint. Besides, for each edge $e$ we call the *hull of* $e$, denoted by $h(e)$, the set of elements sharing edge $e$. Note that this set is not convex in general.

## The Refinement Algorithm

### The refinement algorithm in 2D

Let $\tau$ be the initial triangulation, let $t$ be a triangle to be subdivided, and let $L$ be the list of triangles to be subdivided. The first step of the algorithm is the subdivision of the edges of $t$, Figure 2(b). Then the adjacent triangle $t^*$ is checked to make it conforming. The proccess ends when no further edge in any adjacent triangle is divided for conformity.

```
/* Input variables: $t$, triangle to be refined, and $\tau$, 2D triangular mesh
   Output variables: $L$, list of triangles to be refined
   Internal variables: $t^*$, triangle; $l(t)$, $l(t^*)$ longest-edges of triangles $t$ and $t^*$ respectively */
$L = \{\emptyset\}$
Add $t$ to the list $L$: $L = L \cup t$
For each edge $e$ of $t$, do
    Subdivide $e$
    /* let $t^*$ be the neighbouring triangle of $t$ by the edge $e$,
    and let $l(t^*)$ be the longest-edge of $t^*$ */
    While $l(t) \neq l(t^*)$, do
            Add $t^*$ to the list $L$: $L = L \cup t^*$
            Subdivide edge $l(t^*)$
            $l(t) := l(t^*)$; $t := t^*$
            /* let $t^*$ be the neighboring triangle of $t$ by the edge $l(t)$
            and $l(t^*)$ be the longest-edge of $t^*$ */
    End While
End For
```

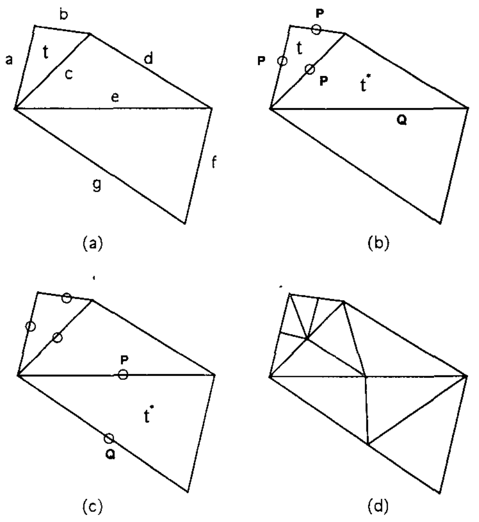

Figure 2: 2D refinement algorithm

## The refinement algorithm in 3D

Let $\tau$ be a three-dimensional tetrahedral grid and $t_0$ be a tetrahedron in the grid to be refined. The 3D algorithm as proposed by Plaza and Carey [15] is:

```
/* Input variables: $t_0$, tetrahedron to be refined, and $\tau$, 3D triangular mesh
   Output variables: new mesh $\tau$
   Internal variables: $L$, set of nodes; $N_e, P, P_{e^\star}$, nodes; $e, e^\star$, edges; $f$, face; $t$, tetrahedron */
/* 1. Edge subdivision */
$L = \{\emptyset\}$
For each edge $e \in t_0$ do
    Bisect $e$ producing new-node $N_e$
    $L = L \cup N_e$
End For
/* 2. The conformity is ensured */
While $L \neq \emptyset$ do
        /* Let $P$ be a node from $L$ with surrounding edge $e_P$ */
        For each tetrahedron $t \in h(e_P)$ do
            For each non-conforming face $f$ of $t$ do
                Bisect the longest-edge $e^\star$ of $f$ producing new-node $P_{e^\star}$ at the midpoint of $e^\star$
                $L = L \cup P_{e^\star}$
            End For
        End For
        $L = L - P$
End While
/* 3. The subdivision of the skeleton is performed */
For each triangular face $f \in skt(\tau)$ to be subdivided do
    Subdivide $f$
End For
```

```
/* 4. The subdivision of the tetrahedra is performed */
For each tetrahedron $t \in \tau$ to be subdivided do
    Subdivide $t$
End For
End.
```

Some of the properties of the previous algorithm are sumarized at the end of the paper. The complexity of the algorithm has been estimated by $O(N_a) + O(N_f) + O(N_t)$ [16], 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 also indicate that in practice the algorithm performs like a linear complexity algorithm [12].

## The Derefinement Algorithm in 3D

For the 2D version of the derefinement algorithm see References [5] [14]. For coarsening a refined mesh we may construct an inverse algorithm based on the previous refinement scheme. Note that all levels of mesh are involved at derefining. 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 number. This number gives us the level in which the corresponding element han been created. Besides we use the sign of these numbers to control the derefinement 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 in which for each type of topological elements these numbers are kept is called, *derefinement vector*.

A proper node $N$ is called an *eligible* proper node for derefining if $N$ can be removed from the mesh while attending to the conformity condition. This means, that a particular node $N$ is eligible if $N$ can be removed from the mesh and the mesh remains conforming, and $N$ 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 derefinement condition, might have been removed.

The derefinement condition we are using in 3D is the same as that which has been used in 2D [5]. A proper node may be removed if the absolute difference between the values in this node of the numerical solution an its corresponding interpolated function is less than a sufficiently small parameter $\epsilon > 0$. That is, if $u_h$ is the numerical solution for a given mesh and $u_h^i$ is the interpolated function of $u_h$ in the derefined mesh, we will get

$$||u_h - u_h^i|| = \; sup_x \mid u_h(x) - u_h^i(x) \mid < \epsilon$$

This error indicator at derefining 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. The subjacent idea is that when a time-step integration scheme is used, a good approximation of the solution when time is $t_{n+1} = t_n + \Delta t_n$ is the previous solution when time is $t_n$. So the described error indicator at derefining 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 $\epsilon$ would be to take a value sufficiently smaller than $\delta$, for example $\epsilon \approx 0.1\delta$. Similarly, one may choose for $\epsilon$ a small fraction of $||u||_\infty$ or, another characteristic value according to the problem or to the range of the expected solution.

However, it should be noted that the algorithms are independent from the refinement or derefinement criteria involved. So the same error indicator used at refining could be used as a derefinement error indicator as well.

The outline of the 3D derefinement algorithm is as follows:

```
/* Input variables: Refined sequence of meshes $T = \{\tau_1 < \tau_2 < \ldots < \tau_k\}$
   Output variables: New derefined sequence $T^m = \{\tau_1 < \tau'_2 < \ldots s < \tau'_m\}$
   Internal variables: $N$, node; $e$, edge; $h(e)$, hull of $e$; $f$; triangular face; $t$, tetrahedron;
   derefinememt indicators */
/* Loop in levels of $T$ */
For $j = k$ to 2, do
    For each eligible proper node $N \in \tau_j$, do
        /* 1. The derefinement condition is evaluated */
        1.1. The derefinement condition is checked
        1.2. The nodes and edges are pointed out
        /* 2. The conformity is ensured locally */
        /* let $e$ be surrounding edge of $N$, and $h(e)$ the hull of $e$ */
        For each tetrahedron $t \in h(e)$, do
            make $N$ conforming in $t$
        End For
    End For
    /* 3. The sequence of meshes is re-defined */
    For each $f \in skt(\tau_{j-1})$, do
        3.1 Subdivide $f$ by the 4-T partition of Rivara
    End For
    For each $t \in \tau_{j-1}$, do
        3.2 Perform the new subdivision of $t$
    End For
End For.
```

As in the 2D case, the concept of adjacency is the central idea in the algorithm in 3D. In order to check the elements belonging to the *hull* of the edge $e$ we go from that edge to a face $f$, called the *supporting face* for $e$, in which $e$ is an edge, and from this face to the neighboring elements of the face. See [16] for details.

## The Refinement/Derefinement Combination and Properties

By combining the above two strategies we obtain the composite refinement/derefinement scheme. This combination can be outlined as follows:

```
*Initial mesh generation and set up of the parameters for refining and derefining*
For $nsteps = 1$, to $N_{max}$, do
    For $i = 1$, to $N_r$, do
        1. Computation of the new timestep $\Delta_i(t)$
        2. Solution of the corresponding system of equations
        3. Computation of the refinement error indicator
        4. Local refinement
    End For
    5. Derefinement of the mesh, with tolerance $\epsilon$
End For.
```

Several properties of the algorithms and their combination are the following: 1) Both refinement and derefinement procedures can be applied to any initial triangulation, even to strong non-convex regions, without any preprocessing [11]. Both procedures are finite and exhibit linear complexity $O(N)$, where $N$ is the number of nodes [13] [16]. 2) The refinement procedure can be applied to any initial triangulation, even to strong non-convex regions, without any preprocessing [11]. 3) The derefinement condition is evaluated in a *minimum* number of nodes: the *elegible proper nodes.* 4) The nested nature of the grids make the use of multigrid methods relatively easy [6, 5]. 5) In the resolution of an evolutinary process, removing dupe nodes keeps the number of equations bounded. This is an important feature because in a finite element code most of the CPU time is employed in the resolution of the associated system of equations [5]. 6) The idea of using the skeleton, could be applied to obtain similar algorithms in higher dimensions.

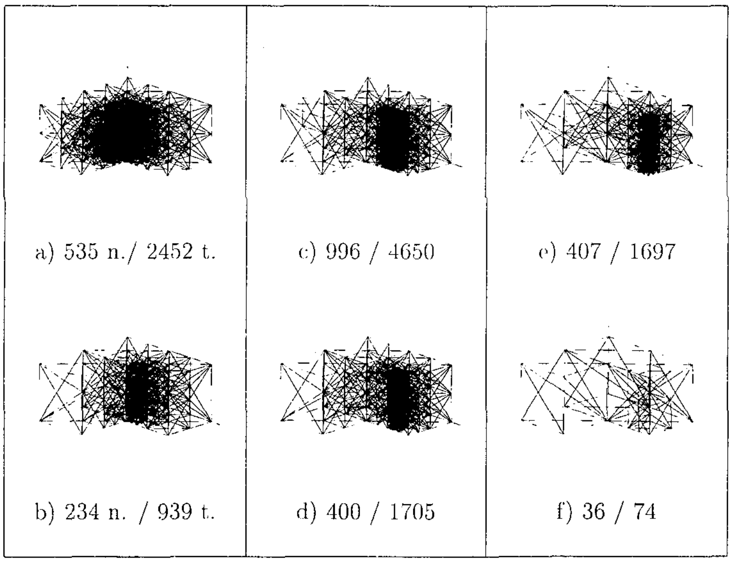

Figure 3: Simulation example in 3 dimensions.
After approximating the singularity the mesh is derefined.
The sequence must be read from a) to f). The number of nodes & tets involved are indicated

## Numerical Examples and Concluding Remarks

We present here only one simulation example in 3D. Figure 3 shows the evolution of meshes when a combination of (local) refinement followed by a derefinement algorithm is applied to get the moving refinement area. Note how the refinement area changes as the singularity moves.

The refinement and coarsening algorithms presented here provide a very useful tool for the treatment of unsteady problems in three dimensions. Adaptivity of the mesh is particularly important in three-dimensional problems because problem size and computational cost grow very rapidly as the mesh size is reduced. With the refinement/derefinement combination families of sequences of nested meshes are obtained, and the multigrid method can be used in an easy way to solve the system of equations associated with the finite-element method [6]. These ideas are clearly relevant in other areas such as approximation of surfaces, visualization, data compression and solid modelling.

## References

[1] D. N. Arnold, A. Mukherjee, and L. Pouly. Locally adapted tetrahedral meshes using bisection. *SIAM J. Sci. Comput*, 1997.

[2] E. Bänsch. Local mesh refinement in 2 and 3 dimensions. *IMPACT Com. Sci. Eng.*, 3:181–191, 1991.

[3] M. Berger. *Geometry.* Springer-Verlag, 1987.

[4] G. F. Carey. *Computational Grids: Generation, Refinement and Solution Strategies.* Taylor and Francis, 1997.

[5] L. Ferragut, R. Montenegro, and A. Plaza. Efficient refinement/derefinement algorithm of nested meshes to solve evolution problems. *Comm. Num. Meth. Eng.*, 10:403–412, 1994.

[6] W. Hackbush. *Multigrid Methods and Applications.* Springer Verlag, 1985.

[7] I. Kossaczký. A recursive approach to local mesh refinement in two and three dimensions. *J. Comp. App. Math.*, 55:275–288, 1994.

[8] A. Liu and B. Joe. On the shape of tetrahedra from bisection. *Math. Comp.*, 63:141–154, 1994.

[9] A. Liu and B. Joe. Quality local refinement of tetrahedral meshes based on bisection. *SIAM J. Sci. Comput.*, 16:1269-1291, 1995.

[10] A. Mukherjee. *An adaptive finite element code for elliptic boundary value problems in three dimensions with applications in numerical relativity.* PhD thesis, Penn. State University, 1996.

[11] M. A. Padrón. *A 3D derefinement algorithm for tetrahedral nested meshes based on the squeleton.* PhD thesis, University of Las Palmas de Gran Canaria, 1999. In Spanish.

[12] A. Plaza and G. F. Carey. About local refinement of tetrahedral grids based on bisection. In *5th Inter. Mesh. Roundtable*, pages 123–136. Sandia Corporation, 1996.

[13] A. Plaza and G. F. Carey. Refinement of simplicial grids based on the skeleton. *App. Num. Math.*, 32(2):195–218, 2000.

[14] A. Plaza, R. Montenegro, and L. Ferragut. An improved derefinement algorithm of nested meshes. In M. Papadrakakis, editor, *Advances in Post and Preprocessing for Finite Element Technology*, pages 175–180. Civil-Comp Ltd., 1994.

[15] A. Plaza, M. A. Padrón, and G. F. Carey. A 3d derefinement algorithm for tetrahedral grids. In *McNU'97, Trends in unstructured mesh generation*, pages 17–23, 1997.

[16] A. Plaza, M. A Padrón, and G. F. Carey. A 3d refinement/derefinement combination to solve evolution problems. *App. Num. Math.*, 32(4):401–418, 2000.

[17] M. C. Rivara. Mesh refinement based on the generalized bisection of simplices. *SIAM J. Numer. Anal.*, 2:604–613, 1984.

[18] M. C. Rivara. A grid generator based on 4-triangles conforming mesh refinement algorithms. *Int. J. Num. Meth. Eng.*, 24:1343–1354, 1987.

[19] M. C. Rivara. Local modification of meshes for adaptive and/or multigrid finite-element methods. *J. Comp. and Appl. Math.*, 36:79–89, 1991.

1 Department of Mathematics. U.L.P.G.C., 35017-Las Palmas de Gran Canaria. Spain. Supported in part by Project PI1999/146 from Gobierno de Canarias. E-mail: aplaza@dma.ulpgc

2 Department of Civil Engineering. U.L.P.G.C.

3 Texas Institute for Computational and Applied Mathematics (TICAM), ASE/EM Dept., University of Texas at Austin. U.S.A. E-mail: carey@cfdlab.ae.utexas.edu Supported by ARPA grant number DABT63-96-C-0061.