Vol. **15**, **4**, 471–483 (1999)

Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería

Un algoritmo de desrefinamiento en 3D para mallas de tetraedros basado en el esqueleto

Ángel Plaza y Miguel A. Padrón

Departamento de Matemáticas
Edificio de Matemáticas e Informática
Universidad de Las Palmas de Gran Canaria
Campus de Tarifa
35017 Las Palmas, España
Tel.: 34-928-458827, Fax: 34-928-458711
e-mail: email: angel@aries.dma.ulpgc.es

Resumen

En este trabajo, se presenta un nuevo algoritmo de desrefinamiento para mallas de tetraedros no-estructuradas. El algoritmo es el inverso del algoritmo de refinamiento de Plaza y Carey[14]. Ambos algoritmos funcionan de forma automática. El algoritmo de refinamiento se puede aplicar a cualquier malla inicial de tetraedros. De igual manera el desrefinamiento se puede usar para conseguir una malla más grosera a partir de una secuencia de mallas encajadas obtenidas por medio de la aplicación reiterada del algoritmo de refinamiento. La combinación de ambos algoritmos se puede usar para resolver problemas dependientes del tiempo en dimensión tres, de forma similar a como ya se han usado algoritmos análogos en dimensiones inferiores Ferragut *et al.*[5].

A SKELETON-BASED DEREFINEMENT ALGORITHM FOR TETRAEDRAL GRIDS

Summary

In the present study, a novel three dimensional derefinement algorithm for nested tetrahedral grids based on bisection is presented. This algorithm is the inverse algorithm scheme first developed by Plaza and Carey[14]. Both schemes are fully automatic. The refinement algorithm can be applied to any initial tetrahedral mesh without any preprocessing. Similarly the derefinement scheme can be used to get a coarser mesh from a sequence of nested tetrahedral meshes obtained by successive aplication of the refinement algorithm. The refinement and derefinement schemes can be easily combined to deal with time dependent problems. These combinations depend only on a few parameters that are fixed into the input data for the user, in the same manner in which anologous algorithms for lower dimensions have been already used Ferragut *et al.*[5].

INTRODUCCIÓN

En los últimos años se han desarrollado diferentes técnicas de refinamiento para triangulaciones bidimensionales y que posteriormente han sido extendidas a 3D. Incluso se han usado conceptos de geometría fractal y sistemas de funciones iteradas[3,13]. Para un tratamiento detallado sobre refinamiento local de mallas véase por ejemplo[4].

Un algoritmo en 2D basado en la bisección es el 4T de Rivara[18,19,21]. La Figura 1 muestra la aplicación de este algoritmo a un solo triángulo. Nótese que el primer paso, (Figura 1a), consiste en la bisección del triángulo por el lado mayor.

 ISSN: 0213–1315 Recibido: Enero 1999

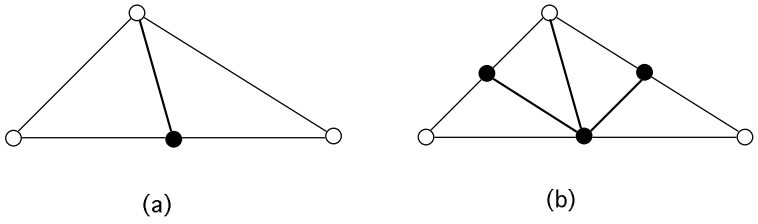


Figura 1. El algoritmo 4T de Rivara

También en dimensión tres se han desarrollado algunas técnicas en los últimos años tanto para el refinamiento como para el desrefinamiento de mallas de tetraedros. Algoritmos basados en la bisección por la arista mayor son los de Rivara y Levin[23] y el de Muthukrishnan *et al.*[12]. Los algoritmos de Bänsch[2] y Liu y Joe[8,9] dividen cada tetraedro en ocho subtetraedros mediante bisección reiterada de las aristas. Otros algoritmos semejantes se encuentran en[1,7,10,11]. Sin embargo estos algoritmos no se pueden aplicar en general a cualquier malla inicial y necesitan algún tipo de pre-proceso.

Recientemente Plaza y Carey[14,15,24] han presentado una generalización del algoritmo 4-T de Rivara a dimensión tres. El algoritmo de Plaza y Carey opera en primer lugar en el esqueleto de la malla (esto es, el conjunto de las caras triangulares de los tetraedros) para dividir después el interior de cada tetraedro de forma congruente con la subdivisión de sus caras. Del mismo modo, el algoritmo de desrefinamiento aquí presentado trabaja en primer lugar en el esqueleto de la triangulación 3D, asegurando la conformidad de la malla que aparecerá, para finalmente, reconstruir el interior de cada tetraedro. En esta segunda tarea (la redefinición del interior de cada tetraedro) se utiliza la parte correspondiente del algoritmo de refinamiento, lo que simplifica el proceso. Ambos algoritmos se pueden aplicar a cualquier triangulación inicial y no necesitan ningún tipo de pre-proceso.

El artículo está organizado como sigue: en primer lugar se recogen algunas definiciones. En la segunda sección se resume el algoritmo de refinamiento en 3D. Después se presentan los algoritmos de desrefinamiento en 2D y 3D basados en el esqueleto. A continuación se muestra el esquema de la combinación de ambos algoritmos para la resolución de problemas evolutivos. Por último se ofrecen algunos ejemplos numéricos.

Definiciones

Sea $V = \{X_0, X_1, \ldots, X_m\}$ un conjunto de $m+1$ puntos de R^n $(0 \leq m \leq n)$ tales que $\{\overrightarrow{X_0 X_i} : 1 \leq i \leq m\}$ es un conjunto de vectores linealmente independientes. Entonces la envoltura cerrada convexa de V, denotada por $S =< V >=< X_0, X_1, \ldots, X_m >$ se llama m-simplex en R^n, y los puntos $X_0, \ldots, X_m$ son llamados *vértices de S*. El número m se dice que es la *dimensión* de S. De esta manera un tetraedro (cerrado) se define por el conjunto (no ordenado) de sus vértices $< X_0, X_1, X_2, X_3 >$, y una cara por sus tres vértices $(< X_i, X_j, X_k >$. Una arista queda definida por un par de vértices distintos $< X_i, X_j >$. Vértices, arista, caras y tetraedros son, respectivamente $0-, 1-, 2-$, y $3-$símplices. Cualquier $i-$simplex contenido en un $n-$simplex S (con $i < n$) será denominado $i-$símplex ó $i-$cara de S.

Sea Ω un conjunto acotado de R^n $(n = 3$, aunque la definición es válida en general) con interior no vacío $\overset{\circ}{\Omega} \neq \emptyset$, y frontera $\partial\Omega$ poligonal. Una partición de Ω en $n-$símplices $\tau = \{t_1, \ldots, t_n\}$ se denominará una triangulación de Ω. Dos símplices t_i, t_j de τ se dicen *adyacentes* si $t_i \bigcap t_j \neq \emptyset$. Si τ es tal que símplices adyacentes comparten una cara entera, ó una arista entera ó un vértice, se dice que τ es una triangulación conforme. El conjunto $skt(\tau) = \{f : f$ es una $(n-1)$-cara de algún t_i, con $t_i \in \tau\}$ se denominará $(n-1)$-esqueleto de τ, también denotado por $(n-1) - skt(\tau)$. Por ejemplo, el esqueleto de una triangulación en dimensión tres está compuesto por las caras triangulares de los tetraedros. Nótese que

el concepto de una malla de símplices conforme es aplicable al $skt(\tau)$ por estar compuesto por $(n-1)$-símplices. Por tanto, si τ es conforme, $skt(\tau)$ es también conforme.

Dos triangulaciones (conformes) τ y τ^* de un mismo conjunto acotado Ω se llamarán *encajadas* ó *anidadas*, y escribiremos $\tau < \tau^*$ si se cumple la siguiente condición: $\forall\ t \in \tau,\ \exists\ t_1,\ \ldots\ , t_p \in \tau^*$ tal que $t = t_1 \cup \ldots \cup t_p$. Diremos también que τ es más grosera que τ^*, ó que τ^* es más fina que τ.

A partir de una malla inicial τ en la que se realizan sucesivos refinamientos por bisección, se obtiene una secuencia de triangulaciones anidadas. En una secuencia de triangulaciones encajadas, los conjuntos de nodos también son encajados. De hecho podemos hablar de nodos *propios* y nodos *heredados* como se define a continuación.

Sea $T = \{\tau_1 < \tau_2 < \ldots < \tau_n\}$ una secuencia de mallas encajadas, donde τ_1 representa la malla inicial, y sea τ_j alguna triangulación de T. Un nodo N de τ_j se llamará *nodo propio* de τ_j si ese nodo no pertenece a ningún nivel anterior de la malla. De lo contrario, N se denominará *nodo heredado* en τ_j. Las aristas, caras y elementos son nombrados de la misma forma[16]. Los nodos propios de τ_j también se denominan nodos *j-nuevos* por Rivara [20].

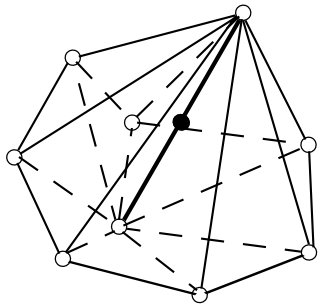


Figura 2. Envoltura de una arista relativa a la teselación

Al aplicar el refinamiento por medio de la bisección de los elementos, denominamos *arista entorno* de un nodo N a la arista en la cual N es el punto medio. Para cada arista E llamamos *envoltura de* E, y la denotamos por $h(E)$, al conjunto de elementos que están compartiendo la arista E. Este conjunto en general es no convexo. En la Figura 2 se dibujan la arista entorno de un nodo (en negro) y la envoltura de su arista entorno. Si la arista tiene un nodo en su punto medio, como en el caso de la figura anterior, el conjunto de *nodos extremos* de la envoltura (que son los puntos blancos en la figura) se denominará *molécula* para el nodo negro de la figura. Obsérvese que esta definición puede ser extendida a cualquier dimensión.

EL PROCESO DE REFINAMIENTO

El algoritmo de refinamiento en 3D se basa en la aplicación del algoritmo de refinamiento en 2D al esqueleto de una malla de tetraedros. Por eso, primero mostramos cómo trabaja la versión bidimensional. La Figura 3a representa una triangulación inicial en la cual el triángulo t va a ser subdividido. El primer paso del algoritmo es la subdivisión de las aristas de t (Figura 3b). Después el triángulo adyacente t^* es examinado para asegurar de esta forma la conformidad. Este proceso termina cuando ya no quedan más triángulos vecinos en los que se tenga que asegurar la conformidad (Figura 3c). Finalmente se divide cada triángulo en dos, tres ó cuatro triángulos-hijos, dependiendo del número de aristas divididas que contenga (Figura 3d).

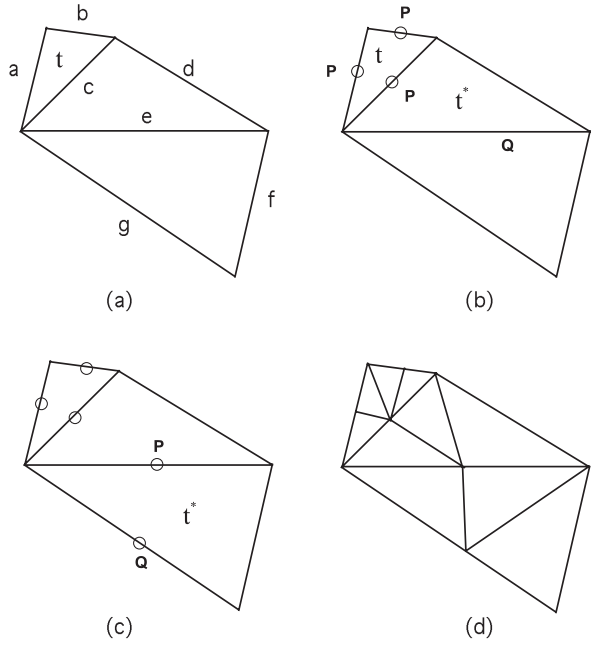


Figura 3. Algoritmo de refinamiento en 2D

El pseudo-código del procedimiento que asegura la conformidad de la malla en 2D se expone a continuación:

Para cada arista e de t, **hace**
Marca el punto medio de e.
/* sea t^* el triángulo vecino de t por la arista e,
y $l(t^*)$ la arista mayor de t^* */
Mientras $l(t) \neq l(t^*)$, **hace**
Añade t^* a la lista L.
Marca el punto medio de la arista $l(t^*)$.
$l(t) = l(t^*)$; $t = t^*$
/* sea t^* el vecino de t a través de la arista $l(t)$,
$l(t^*)$ la arista mayor de t^* */
Fin del mientras
Fin del para

En la Figura 4 se muestra el itinerario seguido para conseguir la conformidad. Las flechas indican qué triángulos vecinos han de examinarse.

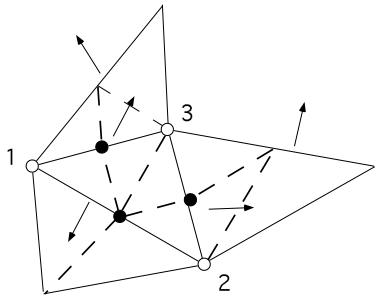


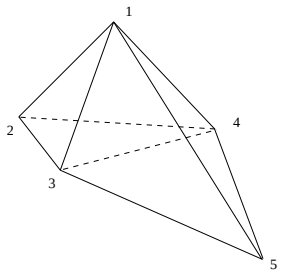
Figura 4. Comprobación de la conformidad local al refinar

Se expone una versión del algoritmo en dimensión tres, en la que un sólo tetraedro va a ser dividido atendiendo al error local de la solución[14,15,24]. Sea este tetraedro $t_0 \in \tau$.

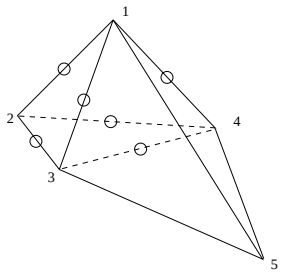
ENTRADA: $\{t_0, \tau\}$
$L = \{\emptyset\}$
Sea L un vector de nuevos-nodos */
/* *1. Subdivisión de las aristas* */
Para cada arista $e \in t_0$, **hace**
Subdivide e, siendo N_e el nuevo nodo en el punto medio.
$L = L \cup \{N_e\}$
Fin del para
/* *2. Se asegura la conformidad* */
Mientras haya un nuevo nodo $N \in L$, con arista entorno E, **hace**
Para cada tetraedro $t \in h(E)$, **hace**
Si t es no-conforme, **entonces**
Se divide la arista mayor de cada cara no-conforme f de t.
/* Sea N_f el nuevo nodo introducido */
$L = L \cup \{N_f\}$
Si es el caso, e divide la arista de referencia de t.
/* Sea N_t el nuevo nodo introducido */
$L = L \cup \{N_t\}$
Fin del si
Fin del para
$L = L - \{N\}$
Fin del mientras
/* *3. Subdivisión de las caras* */
Para cada $f \in skt(\tau)$ que se ha de dividir, **hace**
Subdivide f.
Fin del para
/* *4. Subdivisión de los tetraedros* */
Para cada $t \in \tau$ que se ha de dividir, **hace**
Subdivide internamente t.
Fin del para
SALIDA: Nueva malla τ^*.

La arista de referencia de cada tetraedro es la de mayor longitud, salvo en el caso en el que haya dos aristas comunes a dos caras triangulares y que sean las mayores para esas dos caras. Este caso se corresponde con un tetraedro tipo 2 de los citados en la referencia[14], y puede tomarse cualquiera de esas dos aristas como la de referencia del tetraedro.

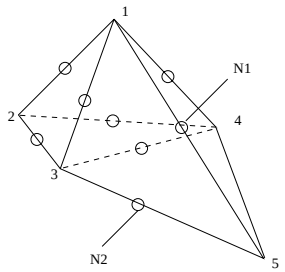
La Figura 5 muestra la aplicación del algoritmo de refinamiento a una malla inicial muy simple compuesta por 2 tetraedros, (Figura 5a). El tetraedro 1-2-3 -4 va a ser refinado de acuerdo con un indicador de error (Figura 5b), y a continuación el tetraedro vecino es refinado para asegurar la conformidad de la malla (ver Figura 5c en la que se introducen los nodos $N1$ y $N2$). Finalmente, la Figura 5d presenta la división del 2-esqueleto y la Figura 5e la malla final donde se han definido los nuevos elementos. En esta última figura se han resaltado las aristas internas que aparecen en los tetraedros al subdividirlos.



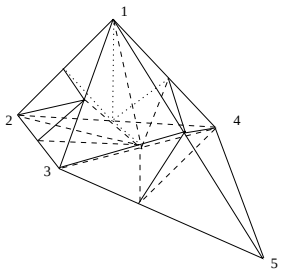
(a) Malla inicial



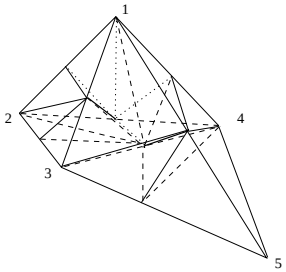
(b) Paso 1



(c) Paso 2: Conformidad



(d) Paso 3: División del esqueleto



(e) Malla final

Figura 5. Ejemplo de aplicación del algoritmo de refinamiento en 3D

EL ALGORITMO DE DESREFINAMIENTO

Para desrefinar una malla refinada, se construye un algoritmo inverso del de refinamiento. Sin embargo, aunque este nuevo algoritmo es el inverso del de refinamiento, es importante hacer notar su mayor complejidad en el sentido de que ahora todos los niveles de la malla están implicados.

Un punto importante al desrefinar es que se necesita conocer la *genealogía* de las aristas, caras y elementos. Es decir, es necesario saber las *aristas-hijas* y la *arista-padre* para cada arista, e igualmente para las caras y los elementos.

Para los nodos, aristas, caras y elementos asignamos una serie de vectores, llamados vectores de desrefinamiento o indicadores de desrefinamiento. Estos vectores nos dan información sobre el nivel en el que cada elemento es propio, además de indicar si ese elemento geométrico debe permanecer ó ha de ser eliminado.

La condición de desrefinamiento es aquella que un nodo N debe verificar para ser quitado. Hemos usado en 3D la misma condición que se ha usado en 2D[5], es decir, la diferencia relativa (y otras veces la absoluta) entre la solución numérica en un nodo particular y la solución numérica promedio en los extremos de los nodos de la arista que contiene a dicho nodo. Si esta diferencia es menor que un pequeño parámetro $\epsilon > 0$, el nodo puede eliminarse. Esto es, si u_h es la solución numérica para una malla dada, y u_h^i es la función interpolada de u_h en la malla desrefinada, obtendremos

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

Ciertamente, el indicador de desrefinamiento no nos permite controlar el error de discretización. En los algoritmos adaptativos el control lo realiza usualmente el indicador de error del proceso de refinamiento. La idea es que al usar un paso de tiempo de integración una buena aproximación de la solución en el tiempo $t_{n+1} = t_n + \Delta t_n$, es la solución en el paso de tiempo anterior t_n. Así que el indicador de desrefinamiento descrito se puede considerar óptimo en el sentido de que una solución dada es aproximada con un número mínimo de nodos después de desrefinar. Si $\delta > 0$ es una tolerancia dada, para el error con la norma del máximo, un criterio práctico sería elegir por ejemplo, $\epsilon \approx 0.1\delta$. De igual forma, se puede elegir ϵ como una pequeña parte de $\|u\|_\infty$ o usar algún otro valor característico de acuerdo con el problema que se estudia ó el rango esperado de la solución.

Sin embargo se debe hacer notar que los algoritmos son independientes del criterio de refinamiento ó de la condición de desrefinamiento utilizada. También se pueden usar fórmulas más complicadas que involucren aproximaciones hasta la segunda derivada de la solución, o incluso sobre los tetraedros en lugar de los nodos.

Por último llamaremos a un nodo propio N *candidato* a ser eliminado, si N puede ser quitado de la malla. Esto quiere decir que la conformidad se asegura manteniendo algunos nodos que de otra forma, dada la condición de desrefinamiento, podrían ser eliminados.

Las ideas y definiciones anteriores son aplicables tanto al caso bidimensional como al caso tridimensional. Antes de exponer el algoritmo de desrefinamiento en 3D, se resume a continuación el caso bidimensional.

El algoritmo de desrefinamiento en 2D

Sea $T = \{\tau_1 < \tau_2 < \ldots < \tau_n\}$ una secuencia de mallas triangulares encajadas, donde τ_1 representa la malla inicial y τ_n la malla más fina de la secuencia. Desrefinar la secuencia, significa obtener una nueva secuencia de mallas $T^m = \{\tau_1 < \tau_2' < \ldots < \tau_m'\}$, donde $m \leq n$. Un esquema de este algoritmo fue propuesto en [17] como sigue:

ENTRADA: Secuencia $T = \{\tau_1 < \tau_2 < \ldots < \tau_n\}$
/* Bucle en niveles de T */
Para $j = n$ **hasta** 2, **hace**
Para cada nodo *propio susceptible de ser eliminado* $N \in \tau_j$, **hace**
Se evalúa la condición de desrefinamiento.
Se marcan los nodos y las aristas.
/* *Se asegura la conformidad localmente* */
/* Sea c la arista entorno de N */
Para cada triángulo vecino t de c, **hace**
Si t es no-conforme, **entonces**
Cambia el indicador de desrefinamiento del nodo P de la arista mayor, y los nodos de la molécula de P deben permanecer.
Fin del si
Fin del para

Fin del para
/* *Redefinición de la malla* */
Si algún nodo propio de τ_j permanece, **entonces**
Se definen nuevas conexiones nodales.
Los vectores de genealogía son modificados.
En otro caso
El nivel en curso j es borrado de la estructura de datos.
Fin del si
Los cambios son heredados para las siguentes mallas.
/* Se obtiene una nueva secuencia de mallas encajadas */
Fin del para
SALIDA: Secuencia $T^m = \{\tau_1 < \tau_2' < \ldots < \tau_m'\}$

Nótese que la conformidad se asegura localmente recorriendo los elementos vecinos de la arista-entorno de cada nodo propio. La Figura 6 muestra una secuencia de cinco niveles de malla. La columna de la izquierda representa la secuencia de entrada $T = \{\tau_1 < \tau_2 < \ldots < \tau_5\}$ del algoritmo. Las tres siguientes columnas representan el resultado del proceso al desrefinar desde el nivel más fino (τ_5) hasta el tercero (τ_3). El nivel desrefinado en cada paso es señalado por una línea punteada en la Figura 6. Los puntos negros significan que esos nodos deben permanecer en la malla por razón de la conformidad, y los blancos son nodos propios candidatos a ser eliminados en cada paso.

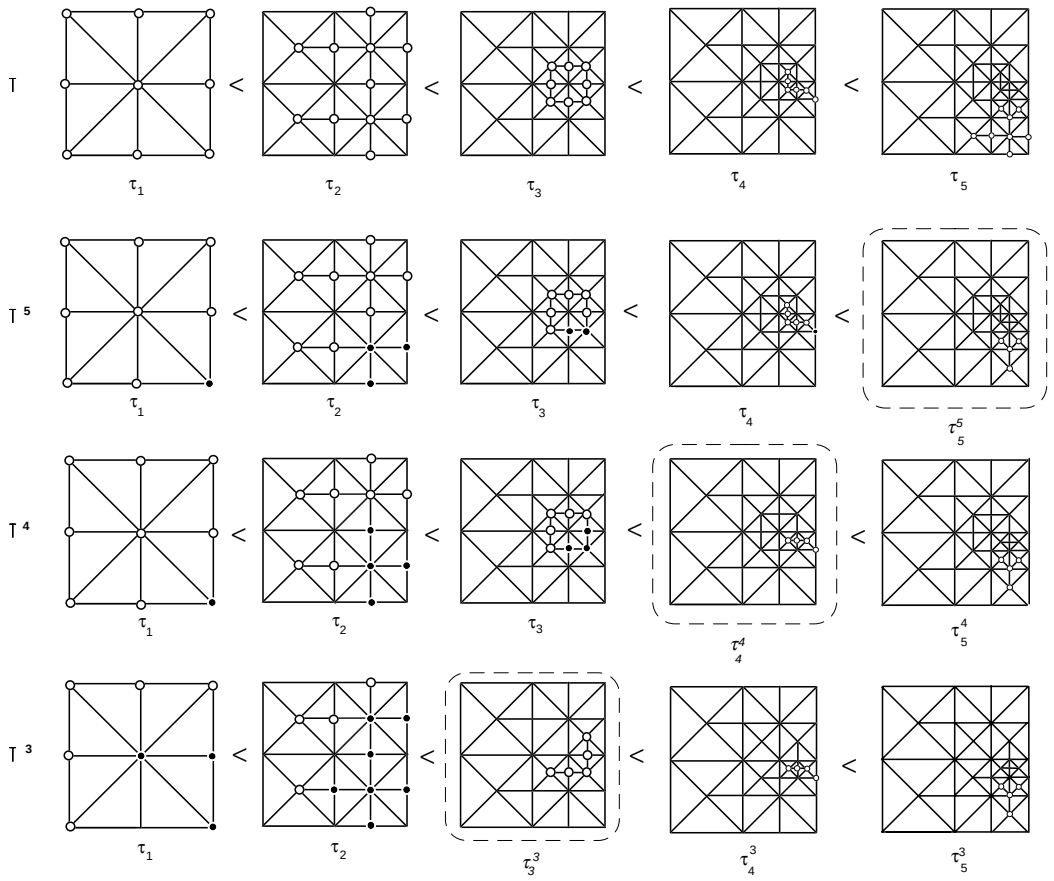


Figura 6. Proceso de desrefinamiento en 2 dimensiones

El algoritmo de desrefinamiento en 3D

El algoritmo comprende principalmente dos pasos: la aplicación del algoritmo de desrefinamiento al esqueleto, y la reconstrucción interna de los tetraedros. Como en el caso bidimensional, el concepto de adyacencia es la idea central en el algoritmo en 3D. Para recorrer los elementos que pertenecen a la envoltura de una arista cualquiera E, usamos la información de los elementos vecinos a cada cara, y la llamada *cara base* de la arista E. Entendemos por *cara base* de una arista E, una cara que tenga por arista la arista E. La defínción de cara base no es unívoca, es decir, adjudicamos a cada arista como cara base una cualquiera de las caras triangulares que la tienen por arista.

ENTRADA: Secuencia $T = \{\tau_1 < \tau_2 < \ldots < \tau_n\}$
/* Buclē en niveles de T */
Para $j = n$ **hasta** 2, **hace**
Para cada nodo *propio elegible* $N \in \tau_j$, **hace**
/* 1. *Se evalúa la condición de desrefinamiento* */
Se examina la condición de desrefinamiento.
Se marcan los nodos y las aristas.
/* 2. *Se asegura la conformidad localmente* */
/* Sea E la arista-entorno de N, y $h(E)$ la *envoltura* de E */
Para cada tetraedro t en $h(E)$, **hace**
Conforma el tetraedro t.
Fin del para
Fin del para
/* 3. *Se re-define el* $skt(\tau_{j-1})$ */
Para cada $f \in skt(\tau_{j-1})$, **hace**
Subdivide f por el algoritmo 4-T de Rivara.
Fin del para
/* 4. *Se redefine el interior de los tetraedros* */
Para cada $t \in \tau_{j-1}$, **hace**
Define una nueva subdivisión de t.
Fin del para
Fin del para
/* Fin del bucle en niveles de malla */
SALIDA: Secuencia $T^m = \{\tau_1 < \tau_2' < \ldots < \tau_m'\}$.

LA COMBINACIÓN REFINAMIENTO/DESREFINAMIENTO

Los algoritmos de refinamiento y desrefinamiento se pueden combinar para obtener una estrategia adaptativa. Este procedimiento se esquematiza en lo que sigue:

Generación de la malla inicial y definición de los parámetros para el refinamiento y el desrefinamiento
Para $npasos = 1$, **hasta** N_{max}, **hace**
Para cada valor $npasos$, **hace**
Para $i = 1$, hasta N_r, **hace**
1. Cálculo del nuevo paso de tiempo $\Delta(t)$.
2. Solución del correspondiente sistema de ecuaciones.
3. Refinamiento Local.
Fin del para
Fin del para
4. Desrefinamiento de la malla, con tolerancia ϵ.
Fin del para

Otra alternativa es utilizar el refinamiento y el desrefinamiento (por elementos triangulares) en el mismo punto del proceso anterior (paso 3), según el indicador de error.

Observaciones:

i. La combinación refinamiento/desrefinamiento depende de pocos parámetros: N_{max}, N_r, $\Delta(t)$, γ, y ϵ.
ii. Tanto el refinamiento como el desrefinamiento muestran una complejidad lineal: $O(N)$.
iii. La condición de desrefinamiento se evalúa en un número *mínimo* de nodos.
iv. Por la propia naturaleza de las mallas encajadas, es relativamente fácil utilizar un método multimalla[5,6].
v. El algoritmo de desrefinamiento se puede combinar con uno de refinamiento global para un proceso de refinamiento local. En este caso, las soluciones obtenidas por sucesivos refinamientos de las mallas, determinan un indicador de error para desrefinamientos locales, aunque el coste computacional sea mucho más alto. Sin embargo el coste computacional de esta estrategia en 3D es extremadamente grande.
vi. Al eliminar nodos menos significativos, el número de ecuaciones involucradas en un proceso evolutivo permanece acotado. Esto es una importante propiedad porque la mayor parte del tiempo de CPU se emplea en la resolución del sistema de ecuaciones.

EJEMPLOS NUMÉRICOS

Un ejemplo de refinamiento

Presentamos un ejemplo numérico para mostrar el comportamiento del algoritmo de refinamiento. La Figura 7a muestra una malla inicial que es localmente refinada para obtener la malla final de la Figura 7b, que contiene 251 nodos y 1016 tetraedros.

(a) Malla inicial (b) Malla localmente refinada

Figura 7. Ejemplo de refinamiento local

Se pueden encontrar muchos ejemplos de la aplicación de la combinación del refinamiento y desrefinamiento en problemas de dos dimensiones[5]. Aquí presentamos únicamente la simulación de un problema evolutivo en 3D. Este consiste en seis refinamientos locales para capturar la singularidad, seguidos de un desrefinamiento. La singularidad corresponde a un eje horizontal en la figura, que se ha ido desplazando durante el proceso. Nótese cómo el número de nodos permanece acotado en un cierto rango durante el proceso.

CONCLUSIONES

Los algoritmos de refinamiento y de desrefinamiento aquí presentados son una herramienta muy útil para el tratamiento de problemas no-estacionarios en dimensión tres. La adaptatividad de la malla es particularmente importante en problemas tridimensionales porque el tamaño del problema y el coste computacional crecen muy rápidamente cuando la dimensión del dominio es mayor y el diámetro de la malla se reduce. Con la combinación refinamiento/desrefinamiento se obtienen secuencias de mallas encajadas. Esto, facilita el uso del método multimalla para la resolución del sistema de ecuaciones asociado al método de los elementos finitos[6].

Estas ideas son importantes, no sólo para desarrollar métodos eficientes para resolver ecuaciones en derivadas parciales, sino que claramente pueden ser interesantes en otras áreas, tales como aproximación de superficies, visualización, compresión de datos, o modelado de sólidos.

Todavía hay algunas cuestiones abiertas relacionadas con la prueba matemática de la estabilidad de las mallas obtenidas, y la existencia de un número finito de clases de tetraedros semejantes (que quizá dependa sólo de la geometría de la malla inicial). Aunque estas propiedades han sido demostradas recientemente en dimensión dos[22], su extensión a dimensión tres no está aún resuelta.

AGRADECIMIENTOS

Este trabajo ha sido financiado en parte por la DGICYES beca número PR95-280 y por la Dirección General de Universidades del Gobierno de Canarias.

REFERENCIAS

1 D.N.. Arnold, A. Mukherjee y L. Pouly, "Locally adapted tetrahedral meshes using bisection", *SIAM J. Sci. Comput*, (1997).

2 E. Bänssch, "Local mesh refinement in 2 and 3 dimensions", *IMPACT Com. Sci. Engng.*, Vol. **3**, pp. 181–191, (1991).

3 S.W. Bova y G.F. Carey, "Mesh generation/refinement fractal concepts and iterated fuction systems", *Int. J. Num. Meth. Engng.*, Vol. **33**, pp. 287–305, (1992).

4 G.F. Carey, "*Computational grids: Generation, refinement and solution strategies*", Taylor and Francis, (1997).

5 L. Ferragut, R. Montenegro y A. Plaza, "Efficient refinement/derefinement algorithm of nested meshes to solve evolution problems", *Comm. Num. Meth. Engng.*, Vol. **10**, pp. 403–412, (1994).

6 W. Hackbush, "*Multigrid methods and applictions* ", Springer Verlag, (1985).

7 I. Kossaczký, "A recursive approach to lacal mesh refinement in two and three dimensions", *J. Comp. App. Math*, Vol. **55**, pp. 275–288, (1994).

8 A. Liu y B. Joe, "On the shape of tetrahedra from bisection", *Math. Comp.*, Vol. **63**, pp. 141–154, (1994).

9 A. Liu y B. Joe, "Quality local refinement of tetrahedral meshes based on bisection", *SIAM J. Sci. Comput.*, Vol. **16**, pp. 1269–1291, (1995).

10 J.M. Maubach, "Local bisection refinement for n-simplicial grids generated by reflection", *SIAM J. Sci. Stat. Comput.*, Vol. **16**, pp. 210–227, (1995).

11 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).

12 E. Muthukrishnan, P.S. Shiakolas, R.V. Nambiar y K.L. Lawrence, "Simple algorithm for adaptive refinement of three-dimensional finite element tetrahedral meshes", *AIAA Journal*, Vol. **33**, pp. 928–932, (1995).

13 A. Plaza, "The fractal behaviour of triangular refined/derefined meshes, *Comm. Num. Meth. Engng.*, Vol. **12**, pp. 295–302, (1996).

14 A. Plaza y G.F. Carey, *About local refinement of tetrahedral grids based on bisection*, 5th Int. Mesh Roundtable, pp. 123-136, Sandia Corporation, (1996).

15 A. Plaza y G.F. Carey, *A new refinement algorithm for tetrahedral grids based on bisection*, Technical Report **54**, T.I.C.A.M., (1996).

16 A. Plaza, L. Ferragut y R. Montenegro, "Derefinement algorithms of nested meshes", Ed. J. van Leeuwen, *Algorithms, software, architecture*, Elsevier Science Publishers B.V., pp. 409–415, North Holland, (1992).

17 A. Plaza, R. Montenegro y L. Ferragut, "Derefinement algorithms of nested meshes", Ed. M. Papadrakakis, *Advances in post and preprocessing for finite element technology*, Civil-Comp Ltd, (1994).

18 M.C. Rivara, "Mesh refinement on the generalized bisection of simplices", *SIAM J. Num. Anal.*, Vol. **2**, pp. 604-613, (1984).

19 M.C. Rivara, "A grid generator based on 4-triangles conforming mesh refinement algorithms", *Int. J. Num. Meth. Engng.*, Vol. **24**, pp. 1343–1354, (1987).

20 M.C. Rivara, "Selective refinement/derefinement algorithms for sequences nested triangulations", *Int. J. Num. Meth. Engng.*, Vol. **28**, pp. 2889–2906, (1989).

21 M.C. Rivara, "Local modification meshes for adaptive and/or multigrid finite-element methods", *J. Comp. and Appl. Math.*, Vol. **36**, pp. 79–89, (1991).

22 M.C. Rivara y G. Iribarren, "The 4-triangles longest-side partition of triangles and linear refinement algorithms", *Math. Comp.*, (1997).

23 M.C. Rivara y C. levin, "A 3-d refinement algorithm suitable for adaptive and multigrid techniques", *J. Comp. and Appl. Num. Math.*, Vol. **8**, pp. 281–290, (1992).

24 A. Plaza y G.F. Carey, "Local refinement of simplicial grids based on the skeleton", *Appl. Numer. Math.*, Vol. **32**, N° 2, pp. 195–218, (2000).