Mecánica Computacional Vol. 14, páginas 512-519
compilado por Sergio Idelsohn y Victorio Sonzogni
Editado por AMCA, Santa Fe, 1994

FINITE ELEMENT 3-D MESH GENERATION USING THE ADVANCING FRONT TECHNIQUE

Enzo A. Dari, Marcelo J. Vénere
División Mecánica Computacional
Centro Atómico Bariloche - C.N.E.A. - Argentina

Raul Feijóo
Laboratório Nacional de Computación Científica - Brasil

RESUMEN

En el presente trabajo se describe una implementación eficiente del método Frontal para la generación de mallas tridimensionales de elementos finitos. Se presta especial atención a las estructuras de datos que deben utilizarse para obtener un costo computacional *O(N)*. También presentamos algunos casos que no pueden ser resueltos con el algoritmo original propuesto por Peraire y colaboradores, y a continuación describimos una serie de tests adicionales que es preciso realizar para evitar estos problemas. Finalmente incluímos una serie de ejemplos que permiten evaluar las posibilidades y costo computacional del algoritmo modificado.

ABSTRACT

In the present work we describe an efficient implementation of the Frontal technique for three dimensional finite element mesh generation. Special attention is given to the data structures needed to obtain an algorithm with *O(N)* computational cost. We also present some cases that can not be solved with the original algorithm, presented by Peraire et al, and we introduce the additional tests necessary to avoid these problems. Finally we present some examples to show the capabilities and computational cost of the modified algorithm.

1. INTRODUCCION

Cuando se desea utilizar el método de elementos finitos para el modelado numérico de problemas tridimensionales, la discretización de la geometría se presenta como uno de los principales obstáculos. Es por ello que en la actualidad se invierten importantes esfuerzos en el desarrollo de herramientas robustas para la generación automática de mallas en 3D. Hasta el momento, las técnicas que han mostrado un cierto grado de éxito en este objetivo, están restringidas a elementos tipo tetraedros y requieren como única información de entrada, la especificación del tamaño de elemento deseado en todo punto del espacio y una triangulación de la superficie de la pieza (ésta debe estar formada por triángulos de buena calidad y que se ajusten al tamaño de elemento especificado). Sin duda una de las técnicas que más se han destacado en este área es el método Frontal, inicialmente propuesto por Lo [1] para problemas bidimensionales, y posteriormente mejorado y extendido a 3D por Peraire et al. [2], [3]. Este algoritmo es descripto con cierto detalle en el punto 3 de este trabajo.

Cuando se compara esta técnica con el método de Delaunay (seguramente el más utilizado en la actualidad), aparecen ventajas y desventajas que hacen difícil una conclusión. Entre las ventajas, las más importantes son quizás la forma natural de generar lo nodos interiores de la malla y el hecho de no requerir una recuperación de frontera. Esto permite, en el promedio, obtener mejores discretizaciones. Por otro lado, la mayor desventaja es sin duda el elevado costo computacional. La generación de un nuevo tetraedro requiere una serie de tests para evitar que se produzcan superposición de volumenes y para ello es preciso verificar que las nuevas caras y aristas no intersectan al frente de generación. Dado que este frente puede contener varios

miles de caras, se hace imprescindible recurrir a una clasificación del mismo, reduciendo asi el número de tests y manteniendo un costo computacional $O(N)$. En el punto 4 se describen las estructuras de datos necesarias para tal fin.

En la práctica el algoritmo tal como está propuesto en el trabajo de Peraire et al, permite la generación de elementos con volumen prácticamente nulo. En el punto 5 mostramos algunos casos muy simples en que se da esta situación y a continuación presentamos algunas modificaciones que pueden hacerse al algoritmo a efectos de obtener mallas sin este tipo de elementos. Básicamente las mismas consisten en ampliar el test de superposición de volumen de forma de que al generar un nuevo tetrahedro no se deja una situación que posteriormente solo puede ser resuelta con un elemento distorsionado.

Finalmente en el punto 6 incluímos algunos ejemplos que permitirán evaluar las posibilidades del método, como asi también tener una idea de su costo computacional.

2. ESPECIFICACION DEL TAMAÑO DE LOS ELEMENTOS

Al generar una malla de elementos finitos, ya sea de superficie o de volumen, es preciso especificar el tamaño de los elementos en cada punto del espacio. Una de las metodologías más aceptadas para ello, y que hemos adoptado, es definir una función *diámetro del elemento* $h(x,y,z)$ en un número finito de puntos y extenderla a todo el dominio mediante interpolación.

Para realizar esta interpolación utilizamos el método sugerido en [5]. La idea es en primer lugar generar la malla Delaunay del conjunto de puntos en que se especificó la función $h(x,y,z)$, la que deberá incluir estrictamente al dominio de interés (normalmente será necesario agregar ocho puntos que sean vértices de una caja que englobe al dominio). Una vez disponible esta *malla base*, para calcular el valor de la función h en un punto arbitrário se busca cual es el tetraedro que contiene al punto en cuestión y se interpola linealmente dentro del mismo.

Esta metodología resulta especialmente adecuada para hacer re-mallados adaptivos. En estos casos en base a un indicador del error en cada elemento se puede calcular un tamaño deseado h_e y adjudicarlo al centroide del elemento; la *malla base* que se genera con estos puntos tendrá entonces tantos nodos como elementos tenga la red original. En estas situaciones es importante que la busqueda del tetraedro que contiene al punto sea muy eficiente, siendo inaceptable una búsqueda secuencial sobre todos los elementos de la malla (costo $O(N)$ por búsqueda). Utilizando la propiedad de que la malla base es siempre convexa, es posible realizar una búsqueda direccional (ver [6]), con lo que el costo de una búsqueda cae a $O(N^{1/3})$. En nuestra implementación obtuvimos tiempos de busqueda prácticamente constantes (rigurosamente el costo computacional es $O(log(N))$), independientemente del tamaño de la malla base, gracias a una clasificación con un *oc-tree* de dicha malla. Este *oc-tree* es construido de forma que cada terminal contenga como máximo un nodo de la malla base. Luego para cada terminal se crea la lista de elementos que tienen intersección con el mismo. Una vez montada esta estructura de datos, la búsqueda del elemento que contiene un dado punto se realiza en dos pasos (ver [7]): primro se busca el términal del *oc-tree* en que cae el punto (costo $O(log(N))$), y posteriormente se busca al tetraedro entre la lista de elementos asociados al terminal (costo constante).

3. DESCRIPCION DEL ALGORITMO ORIGINAL

Como se mencionó en la introducción, el algoritmo requiere como punto de partida una triangulación de la superficie del dominio. Considerando a ésta como el frente de generación inicial, se van tomando los triángulos de a uno y se genera un tetraedro en la siguiente forma:

1. Seleccionar según el criterio elegido una cara ABC del frente de generación. Existe total libertad para elegir un criterio y dada la gran influencia que tiene en la malla final obtenida resulta interesante tener la

posibilidad de cambiarlo. En [3] se sugiere seleccionar primero los elementos más pequeños para evitar pasar por alto zonas localmente densificadas.

2. Calcular la posición P donde sería ideal colocar el cuarto nodo para formar el nuevo tetraedro. Para ello se busca el tamaño δ deseado en el centroide del triángulo ABC y se calcula h (altura del tetrahedro) de forma que el tamaño promedio de las tres nuevas aristas sea δ . Para evitar cambios de densidad muy bruscos resulta conveniente acotar h por arriba y por debajo, con el tamaño promedio de las aristas de ABC.
3. Buscar los nodos Q_i ya existentes en el frente y que están en el radio de influencia de P. Esta zona se define como la esfera de radio $c.h$ y centro en P. En [3] se propone c=1.5.
4. Ordenar los nodos Q_i en forma creciente de distancia O_i-P donde O_i es el centro de una esfera que pasa por los puntos Q_i y D, siendo D el punto A, B o C que está a mayor distancia de M. Para terminar de determinar la esfera se agrega la restricción de que O_i se encuentra sobre la recta P-M.
5. Se forma el nuevo tetraedro con el primer nodo Q_i que pasa el test de no-superposición de volumen. Si ninguno lo pasa se toma el nodo P. Por último si este tampoco pasa el test, se generan nuevos nodos sobre la recta M-P, pero a menor altura que h (nodos P_i), hasta que alguno consiga pasar el test.
6. Se actualiza el frente agregando las nuevas caras y borrando las que ya existían.

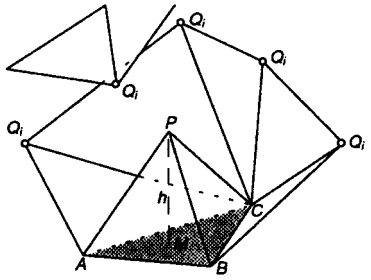

Figura 1: Generación de un nuevo tetrahedro.

ABC: Cara seleccionada del frente
M: Centroide del triángulo ABC
h: Altura de elemento deseada en el punto M
P: Punto ideal donde colocar el cuarto nodo del tetraedro
Qi: Nodos existentes en el frente que son candidatos para reemplazar a P

Este proceso se repite mientras existan triángulos en el frente. En la figura 2 se muestra un ejemplo de como evoluciona la malla generada por un lado y el frente de generación por el otro, en un caso muy sencillo.

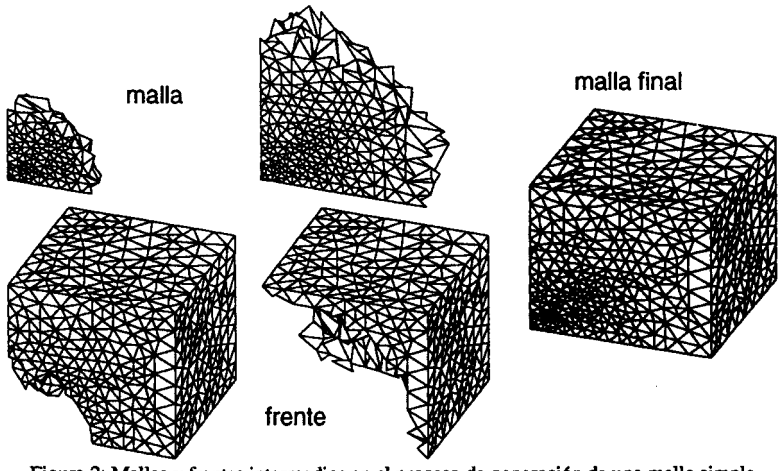

Figura 2: Mallas y frentes intermedios en el proceso de generación de una malla simple

Es conveniente mencionar que en [3], la posición del punto P (paso 2) se calcula teniendo en cuenta no sólo el tamaño de elemento deseado como en nuestro caso, sino también un cierto aplastamiento (*stretching*) en una dirección, para lo cual hay que suministrar además el grado y la dirección de *stretching*.

4. ESTRUCTURAS DE DATOS PARA OBTENER UN COSTO $O(N)$

En el área de generación de mallas tridimensionales, para que un algoritmo sea aplicable a casos de gran envergadura (cientos de miles o millones de elementos), resulta imprescindible que su costo computacional sea $O(N)$ u $O(N.logN)$. Lamentablemente, en general estos algoritmos muestran una tendencia natural a tener un costo $O(N^2)$ que resulta inaceptable, aun en casos de mediano tamaño. Para poder reducir este orden es preciso recurrir a estructuras de datos y algoritmos bastante más complejos.

En el caso particular del método frontal aparecen dos puntos que pueden introducir un costo $O(N^2)$: Por un lado, para cada tetraedro que se genera hay que buscar cuales son los nodos del frente que están dentro de la esfera de influencia de P y verificar que el nuevo elemento no intersecta a ninguna cara del frente. Y por el otro, cuando se toma una cara del frente, se lo hace según un criterio (por ejemplo el triángulo con menor altura), por lo que habrá que buscar cuál de todas las caras del frente es la que cumple con el mismo.

Para solucionar el primer problema, la busqueda de nodos y caras del frente debe realizarse en un entorno del punto P y no sobre todo el frente. Para ello utilizamos una estructura tipo *oc-tree*, de forma que en cada terminal del mismo se almacenan las caras que lo intersecten. El entorno de P se define entonces como todos los terminales del *oc-tree* que intersectan a la esfera de radio $c.h$ y centro en P. Notese que la información almacenada en este *oc-tree* debe ser actualizada continuamente, ya que con la generación de un nuevo tetraedro van a crearse y borrarse caras en el frente.

El segundo problema puede resolverse manteniendo ordenado el frente según el criterio elegido, evitando asi la búsqueda. Debido a que el frente es una estructura dinámica que crece y se achica, no alcanza con ordenarlo inicialmente, sino que cada vez que se inserta o borra una cara, el mismo debe ser reordenado. Para hacer esto en forma eficiente utilizamos una estructura tipo árbol binario. En determinados casos, puede ocurrir que este árbol se desbalancee y el costo $O(logN)$ de la búsqueda se transforme en $O(N)$. Para evitar esta situación es conveniente realizar un re-balance periódico del árbol.

5. CASOS PATOLOGICOS Y MODIFICACIONES PROPUESTAS

Lamentablemente el algoritmo tal como ha sido descripto en el punto 3, permite la generación de tetraedros de volumen prácticamente nulo. La causa de ello es que cuando se genera un nuevo elemento sólo se presta atención a que el mismo sea válido (no intersecte al frente) y no se está analizando que situación queda cuando las nuevas caras generadas son agregadas al frente de generación. En la figura 3 se muestra un ejemplo muy sencillo donde a causa de ello puede ser necesario crear un elemento con volumen casi nulo:

Supongamos que se va a generar el tetraedro correspondiente a la cara ABC. En el frente existen dos nodos Q_0 y Q_1, posibles candidatos para formar el nuevo elemento donde Q_1 es el que está primero en la lista de nodos Q (paso 4). Si los nodos B, C, Q_0 y Q_1 son exactamente coplanares, la única posibilidad es la de la situación a) ya que el tetraedro $ABCQ_1$ intersectaría la arista BQ_0 y no sería admisible. Sin embargo basta perturbar la posición de Q_0 o Q_1 en forma adecuada para que esta intersección no se detecte, y en ese caso se llega a la situación b), que posteriormente sólo podrá resolverse generando el tetraedro BCQ_0Q_1 con volumen casi nulo.

Para evitar estas situaciones hemos introducido modificaciones al test de validez utilizado en el punto 5 del algoritmo original: en lugar de detectar intersecciones entre aristas y caras, se calculan distancias entre

las mismas y si estas superan cierta cota se considera válido al tetraedro a generar. Los casos en que la arista y la cara tengan un vértice en común requieren un tratamiento especial.

De esta manera se solucionan los problemas como el de la figura 3, pero se ha introducido un nuevo parámetro (la cota). Un valor elevado de esta cota garantiza una buena calidad de elementos, pero en contrapartida, es probable que la generación no pueda terminarse a causa de caras en el frente con las que no es posible generar un tetraedro que pase el test de validez. En estos casos es posible completar la generación con una cota inferior.

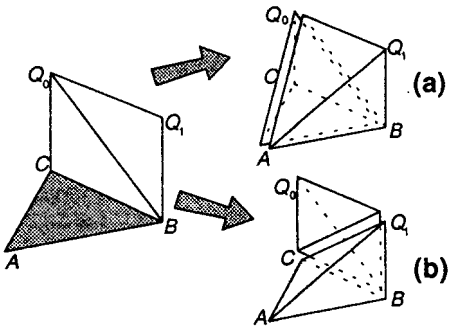

Figura 3: Posible generación de un elemento con volumen nulo

6. EJEMPLOS

Ejemplo 1: Boquilla de salida de un recipiente de presión

Este ejemplo consiste en la generación de una malla tridimensional para el cálculo del estado de tensiones en los alrededores de la boquilla de salida de un recipiente de presión perteneciente a la planta industrial de agua pesada de Arroyito (C.N.E.A.). La forma del dominio y la triangulación de la superficie se muestran en las figuras 4 y 5. La malla de volumen generada contiene 69105 tetraedros y 18049 nodos.

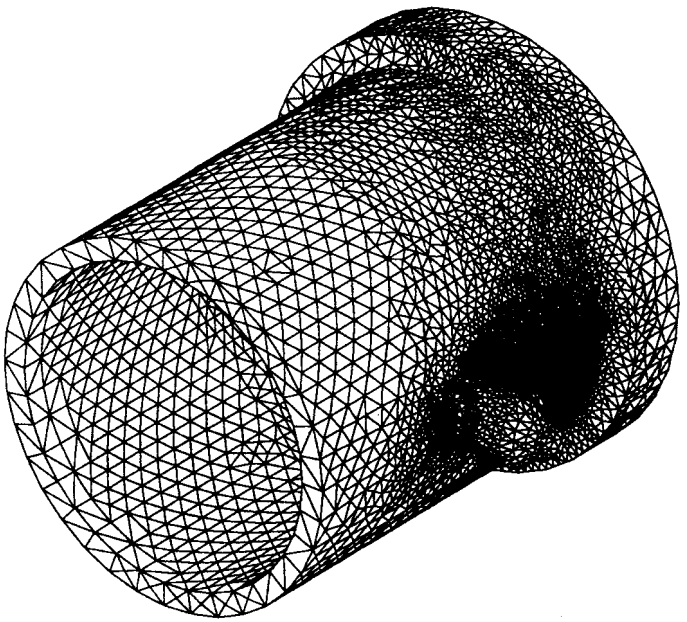

Figura 4: Malla para recipiente de presión.

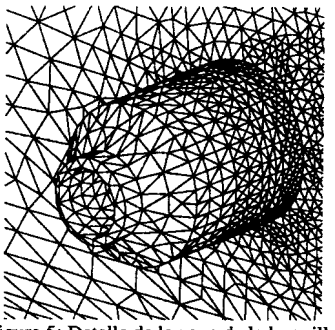

Figura 5: Detalle de la zona de la boquilla

Ejemplo 2: Tapa de un recipiente de Presión

Este es otro ejemplo de una malla para realizar un análisis tridimensional del estado de tensiones en un recipiente de presión. En este caso se trata de la tapa de una torre que tiene una serie de orificios no simétricos y también pertenece a la planta industrial de agua pesada de Arroyito (C.N.E.A.). En la figura 6 se muestra la superficie de la malla generada, y en la figura 7 se puede ver la discretización en el interior del dominio (se realizó un corte de la malla de volumen). Esta malla esta formada por 120401 tetraedros y 25522 nodos.

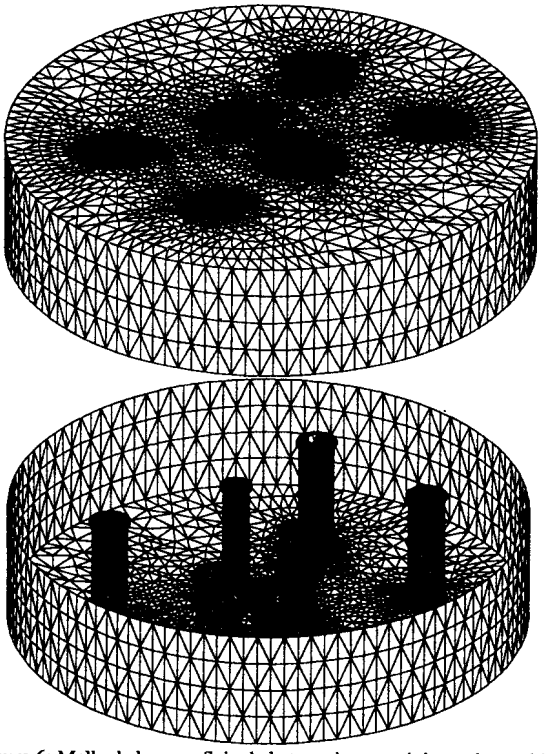

Figura 6: Malla de la superficie de la tapa de un recipiente de presión

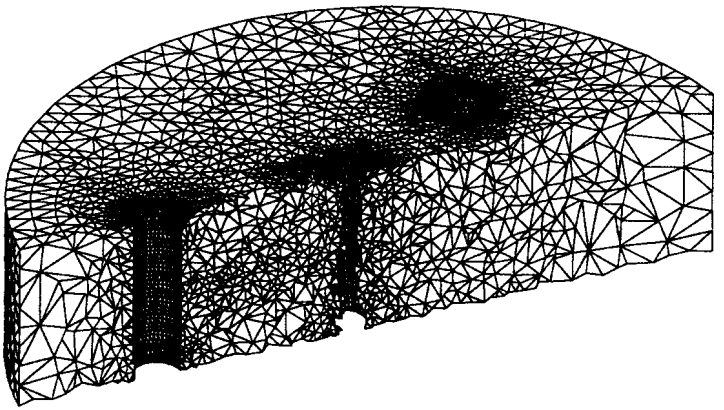

Figura 7: Corte de la malla de volumen

7. CONCLUSIONES

En nuestra experiencia, el método Frontal se presenta como una alternativa válida para la generación de mallas de volumen, siempre que se tengan en cuenta las consideraciones adicionales que mencionamos en el punto 5. Su costo computacional es al menos un orden de magnitud superior al de la técnica Delaunay, pero en nuestra estadística sigue siendo $O(N)$ y no representa un serio obstáculo para la performance de los computadores actuales. En contrapartida observamos que este método no genera elementos distorsionados cerca de la superficie del dominio, como es el caso del método Delaunay.

El principal defecto de nuestra implementación es que en algunos casos es necesario realizar la generación en varios pasos, ajustando la cota que impide la generación de elementos distorsionados.

8. AGRADECIMIENTOS

Los autores desean agradecer el soporte recibido de los siguientes organismos:

- CONICET-Argentina, a través del PID-BID N° 40.
- CNPq-Brasil y CONICET-Argentina, a traves del "Proyecto Argentino-Brasileño en Mecánica Computacional".
- RAHE-Brasil, a traves del proyecto "Rede Integrada de Procedimentos para o Projeto de Componentes Mecánicos. Sistema RIP-PROCOM".

9. BIBLIOGRAFIA

1. Lo, S.H., "A new Mesh Generation Scheme for Arbitrary planar domains", International Journal for Numerical Methods in Engineering, Vol. 21, 1985, págs. 1403-1426.
2. Peraire J., Vahdati M., Morgan K., Zienkiewicz O.C., "Adaptive remeshing for compressible flow computations.", Journal of Comp. Physics, Vol.72, 1987, págs. 449-466.

3. Peraire J., Peiró J., "Adaptive Remeshing for Three-DImensional Compressible Flow Computations", Journal of Comp. Physics, Vol 103, 1991.

4. Peiró J., "A finite element procedure for the solution of the Euler equations on unstructured meshes", Ph.D.Thesis, Dept. Civil Eng., Univ. College of Swansea, 1989.

5. Dari E.A., Vénere M.J., "Visualización de campos en 2D y 3D a partir de su valor en un número finito de puntos". Congreso de la AATN, Buenos Aires, 1990.

6. Vénere M.J., Dari E.A., "Análisis comparativo de algoritmos para obtener triangulaciones Delaunay". Mecánica Computacional, Vol.10, 1990, pags. 491-506.

7. Dari E.A., Vénere M.J., "Algoritmos eficientes para la búsqueda del elemento de una red que contiene un punto dado". Mecánica Computacional, 1990, Vol.10, pags. 455-464.

8. Jin H., Tanner R.I., "Generation of unstructured tetrahedral meshes by advancing front technique", International Journal for Numerical Methods in Engineering, Vol. 36, 1993, págs. 1805-1823.