Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXVIII, págs. 1269-1281 (artículo completo)
Cristian García Bauza, Pablo Lotito, Lisandro Parente, Marcelo Vénere (Eds.)
Tandil, Argentina, 3-6 Noviembre 2009

GENERACIÓN DE MALLAS DE HEXAEDROS PARA PROBLEMAS DE SOLDADURA

Pablo J. Novara[a], **Nestor A. Calvo**[a,b] **y Alberto Cardona**[a,b]

[a]*Facultad de Ingeniería y Ciencias Hídricas, Universidad Nacional del Litoral, Ciudad Universitaria, Santa Fe, Argentina, zaskar_84@yahoo.com.ar*

[b]*Centro Internacional de Métodos Computacionales en Ingeniería - CIMEC (INTEC). Parque Tecnológico Litoral Centro, Santa Fe, Argentina.*

Palabras Clave: Generación de mallas, soldadura, hexaedros, mallas semiestructuradas

Resumen. Si bien no existe un generador automático y universal para mallas de hexaedros, algunos problemas presentan dominios suficientemente simples como para permitir el mallado con técnicas diseñadas ad-hoc. En este trabajo se presentan los métodos utilizados para mallar con hexaedros cordones y sustratos metálicos en problemas de soldadura, utilizando técnicas 2.5 dimensionales, a las que se agrega un refinamiento 3D guiado por templates.

1. INTRODUCCIÓN

La exactitud de los resultados de una simulación mediante FEM depende en gran medida de las características de la malla utilizada. El tipo y la calidad de los elementos generados afectan directamente a la estabilidad numérica del cálculo y al error de aproximación de la solución. Además, de acuerdo al tipo de análisis que se requiere realizar y al método que se pretende emplear, puede ser deseable o necesario utilizar un tipo de elemento en particular. Por ejemplo, en análisis elastoplástico que involucra deformaciones a volumen constante en régimen plástico, se prefiere usar hexaedros, mientras que para análisis térmico con cambio de fase, se encontró que los tetraedros con interpolación lineal poseen muy buenas propiedades de convergencia (Benzley et al., 1995; Pakal, 1998). Por otra parte, la cantidad de elementos generados también determina el tiempo requerido para llevar a cabo la simulación y su consumo de recursos. En general, es deseable que la malla contenga un mayor número de elementos en las zonas de especial interés para el análisis, y un menor número de elementos en las zonas más alejadas. De esta forma, se busca controlar el error de aproximación a valores aceptables en cada zona de acuerdo a su relevancia para el análisis.

Se encuentran desarrolladas en la actualidad numerosas técnicas generales para la generación automática de mallas estructuradas y no estructuradas. Las mallas más comúnmente utilizadas son las mallas constituidas por tetraedros. Para este tipo de elementos, existen algoritmos ampliamente aceptados y de probada eficacia (Owen, 1998, 2001). Sin embargo, para el caso de elementos hexaédricos, la generación automática no es un problema resuelto. Si bien existen también numerosos enfoques al respecto (Carbonera y Shepherd (2006); Kraft (1999); Schneiders et al. (1996); Schneiders (2000), etc), ninguno de ellos ofrece la generalidad suficiente que le permita desempeñarse correctamente para cualquier dominio geométrico. Por otro lado, este tipo de métodos presenta una elevada complejidad que se traduce en un alto coste computacional. Los métodos para la generación de mallas estructuradas o semiestructuradas permiten ahorrar tiempo de cálculo, pero presentan un gran número de restricciones respecto a los dominios sobre los cuales pueden aplicarse. Un bajo coste computacional no sólo permite reducir el tiempo de proceso, sino que también posibilita la regeneración de la malla en determinados pasos del análisis, permitiendo adaptar los niveles de refinamiento conforme avance el tiempo en la simulación. Cabe aclarar que, aunque no es un problema trivial, resulta posible generar una malla de tetraedros a partir de una malla de hexaedros, lo cual se logra manteniendo constante el conjunto de nodos y cambiando adecuadamente las conectividades entre los mismos. El proceso inverso, en cambio, generalmente no es posible, y en caso de realizarse, modificando para ello el conjunto de nodos, resulta más complejo y costoso, y la calidad de la malla de hexaedros resultante será generalmente inferior.

Debido a esto, en este trabajo se presenta un algoritmo de generación automática de mallas de hexaedros semiestructuradas, que aprovecha las características particulares de la geometría específica del problema de soldadura combinando variaciones de diferentes técnicas de mallado más restrictivas. De esta forma, el algoritmo propuesto presenta un costo computacional comparativamente reducido y una calidad de malla aceptable de acuerdo a las exigencias del problema. Dado que la malla debe modificarse conforme avance el tiempo en la simulación debido a la incorporación de material en los diferentes cordones de soldadura, en realidad se construye un conjunto de mallas semi-independientes, que luego se combinan compatibilizando las interfaces entre las mismas y se utiliza una bandera de activación por elemento para simular la deposición del material de soldadura conforme avanza el tiempo en la simulación.

2. DESCRIPCIÓN DEL PROBLEMA

El problema que aquí se plantea es la generación de mallas de hexaedros para el análisis por el método de elementos finitos para un conjunto de geometrías utilizadas en problemas de soldadura. Estos son problemas en los que se simula el aporte de material de soldadura sobre uno o dos sustratos metálicos base (ver Figura 1). El material se deposita formando uno o más cordones rectos conforme avanza el tiempo en la simulación. Esto hace que la malla deba modificarse de alguna manera para modelar la incorporación de material durante la simulación, y que en la misma puedan diferenciarse claramente cada uno de los cordones colocados.

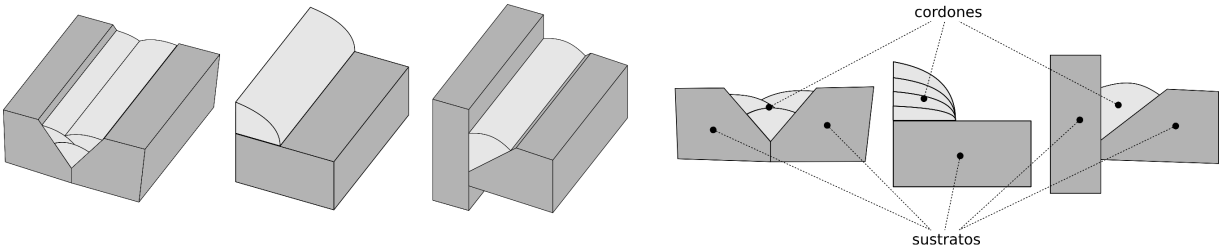

Figura 1: Posibles configuraciones geométricas

Todos los ejemplos de modelos propuestos pueden ser clasificados como modelos en dos-dimensiones-y-media (2.5D). Esta denominación suele ser utilizada para hacer referencia a geometrías tridimensionales que pueden ser descriptas mediante una geometría bidimensional y algún mecanismo de extrusión o proyección de la misma a lo largo de un eje o una curva. No existen limitantes formales para la forma en que se debe definir el modelo bidimensional que da origen al dominio total.

3. ESTRATEGIA DE RESOLUCIÓN

Para plantear una solución al problema presentado se aprovecha que el mismo puede analizarse como un problema en 2.5D: se construye inicialmente una malla de cuadriláteros bidimensional correspondiente a una vista frontal del dominio, y luego, mediante extrusión, se obtiene la malla final de hexaedros. De esta forma, la mayor parte del problema se analiza y resuelve en un espacio bidimensional reduciendo así en gran medida su complejidad. Sin embargo, una extrusión simple implica un tamaño de arista constante (proyectada sobre el eje de extrusión) y no resulta completamente adecuada. Por esto la extrusión se realiza colocando plantillas de hexaedros hasta rellenar el espacio (ver Figura 3). Estas plantillas estan diseñadas para ser intercambiables entre sí durante el avance del algoritmo (es decir, las fronteras entre ellas son topológicamente equivalentes) y permitir controlar el tamaño de los elementos para producir el desrefinamiento necesario en las zonas de los sustratos más distantes a los cordones. Sin embargo, para que se puedan utilizar correctamente estas plantillas en la extrusión que da origen a la malla 3D, es necesario imponer algunas restricciones importantes en la generación de las mallas 2D. Por esto, las mallas 2D de los sustratos se construyen combinando los perfiles de los bloques de extrusión de forma estructurada.

La incorporación de material de soldadura se simula activando capas de elementos en cada paso de tiempo. Para esto se deberán diferenciar los elementos correspondientes a los sustratos, y almacenar junto con cada elemento de un cordón de soldadura un indicador que permita determinar en qué instante de tiempo o en qué orden deben activarse.

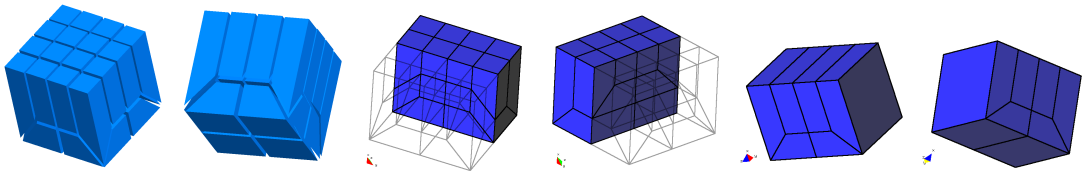

Figura 2: Detalle de las plantillas multi-bloque utilizadas

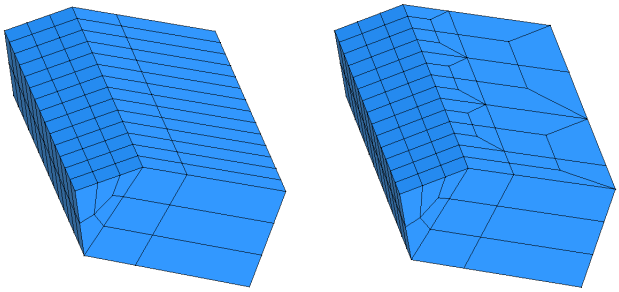

Figura 3: Extrusión simple vs. extrusión por bloques

Al analizar las posibles configuraciones geométricas a mallar, se observa que la variedad de configuraciones está acotada, lo cual permite plantear una taxonomía de casos y estudiar estrategias apropiadas para cada uno de ellos. Sin embargo, al evaluar esta alternativa se ha encontrado que en todos los casos el dominio total puede particionarse en un conjunto de subdominios de menor complejidad (ver Figura 4). Por subdominio de menor complejidad debe entenderse un subdominio mapeable a una figura geométrica simple y con una variación en el tamaño de elemento deseado (h) sobre el mismo nula o lineal y con una única dirección. De esta forma, al resolver individualmente los subdominios y luego integrarlos se resuelve el problema completo y al mismo tiempo se le otorga mayor generalidad a la solución desarrollada, ya que los algoritmos para los subdominios podrían reutilizarse posteriormente en otras configuraciones diferentes a las contempladas inicialmente.

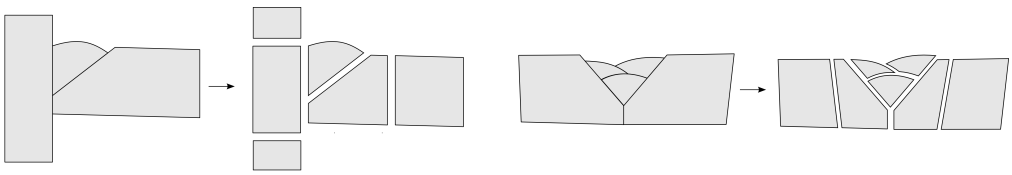

Figura 4: Subdivisión del dominio total en subdominios

Los algoritmos para la generación de mallas de cuadriláteros en dos dimensiones se desarrollaron individualmente para cada tipo de subdominio. En todos los casos se parte de una malla de borde con h variable. El tipo de subdominio se determina principalmente por la cantidad de lados del mismo. De esta forma, los tres tipos de subdominios básicos son el triángulo, el cuadrilátero y el pentágono. Sin embargo, dado que los lados no son necesariamente líneas rectas, según esta clasificación, se considera triángulo a cualquier forma de tres lados, aunque sus lados sean curvos. Es por esto que se dice que los dominios clasificados se asociarán topológicamente con triángulos, cuadriláteros, pentágonos, etc. aunque su representación no sea exactamente dicho polígono.

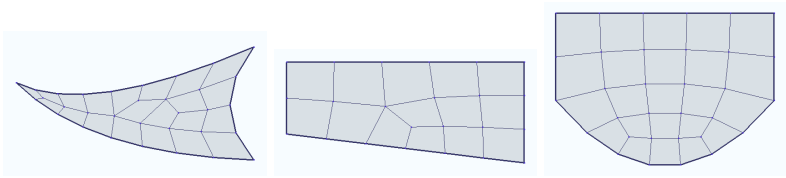

Figura 5: Ejemplos de mallas 2D para formas triangulares y cuadrangulares

Por otro lado, se diferencian también los dominios que presenten un h constante (cordones) de los dominios que presenten valores de h considerablemente menores para uno de sus lados y deban generar en consecuencia un proceso de desrefinamiento en algún sentido predeterminado (sustratos). Los cordones de material de soldadura incluirán sólo formas triangulares y cuadrangulares. Las primeras se utilizan generalmente para la primer fila de cordones (los cordones que se apoyan sobre el sustrato base), mientras que los segundos se utilizan generalmente para las filas restantes (los cordones que se apoyan sobre otros cordones). Los sustratos para el problema de soldadura pueden construirse combinando adecuadamente subdominios cuadrangulares y pentagonales.

El método para mallar segmentos con h variando linealmente y los algoritmos de mallado para cada tipo de subdominio no se exponen en el presente trabajo debido a su extensión (para mayor detalle, ver Novara (2009)). Sólo se dirá que dichos algorítmos dividen cada problema en un conjunto de configuraciones posibles y realizan para cada una un avance frontal desarrollado ad-hoc mallando de forma estructurada cuando es posible y completando las transiciones o huecos con plantillas multibloques predefinidas diseñadas para obtener lineas duales de la mayor longitud posible (Calvo, 2005). Todas las mallas de cuadriláteros se construyen de forma que no se requiere dedicar mayor esfuerzo al suavizado posterior de las mismas. Estas no presentan elementos invertidos ni degenerados. Sin embargo, aún así se aplica un número reducido de iteraciones de suavizado Laplaciano para mejorar la ubicación de algunos nodos, dado que es una estrategia local, de fácil implementación y que por su simpleza de cálculo y el hecho de que sólo se aplica en la malla bidimensional no representará una penalización notable sobre el tiempo de ejecución del proceso completo.

3.1. Ensamblado de Subdominios

Cuando se desarrollaron los mecanismos utilizados para la generación de mallas para formas simples con h constante (cordones), no se consideró ninguna restricción relativa a las mallas de frontera (es decir, las divisiones de sus lados). Sin embargo, cuando se requiere integrar estos algoritmos en la geometría real y completa, este tipo de consideraciones resulta fundamental para lograr compatibilizar las fronteras de cordones vecinos y obtener así una malla globalmente conexa y coherente. La estrategia seleccionada para resolver este problema consiste en determinar un orden adecuado, de forma que al iterar sobre la lista de cordones se puedan aplicar en cada uno las restricciones generadas por los anteriores. De esta forma, el primer cordón tendrá total libertad para escoger su frontera; el segundo, en caso de ser vecino del primero, deberá respetar los segmentos de frontera determinados por aquel; el tercero deberá respetar los de sus dos predecesores; y así hasta finalizar la lista. La determinación del orden adecuado es una de las principales dificultades.

El algoritmo iterativo que se utilizó parte de una aproximación inicial para dicho orden, intenta mallar con dicho orden, y en caso de no conseguirlo, altera la posición de uno de los cordones y vuelve a comenzar. La posición de cada cordón en la lista puede considerarse como

su prioridad a la hora de fijar las cantidades de divisiones para sus fronteras. El orden inicial está determinado por la cantidad de segmentos compartidos que tiene cada cordón en sus fronteras. Se colocan al principio de la lista los cordones que presenten un menor numero de segmentos compartidos. De esta forma, al generar las mallas de frontera de estos cordones, se estarán imponiendo la menor cantidad posible de restricciones para los cordones futuros. Para cada elemento de la lista, el algoritmo intenta reajustar la estrategia de mallado según las restricciones determinadas hasta el momento compensando las diferencias con la estrategia original modificando las cantidades de divisiones en las fronteras que aún permanecen libres (ver Figura 6).

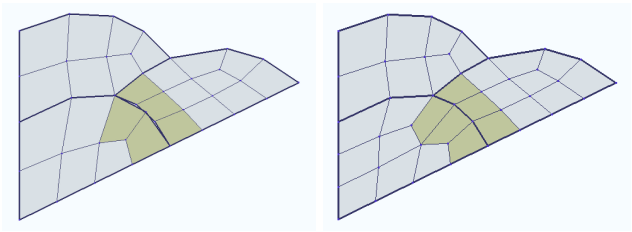

Figura 6: Grupo de tres cordones con fronteras incompatibles y posible solución

Cuando el algoritmo llega a un cordón en la lista cuya estrategia de mallado no es posible corregir para ajustarse a las fronteras impuestas, debido a las restricciones generadas en pasos anteriores, procede a reubicar ese cordón en una posición anterior a la posición actual dentro de la lista. Luego, el algoritmo elimina todas las restricciones y vuelve a recorrer la lista desde el principio. Es de esperar que este cambio se traduzca en un menor número de restricciones para la forma de cordón que detuvo el algoritmo en el ordenamiento anterior, ya que de esta forma tiene un menor número de predecesores.

Se debe notar que podría ocurrir que luego de algunos intentos la lista de cordones vuelva a contener un orden repetido (es decir, que ya se probó anteriormente). Sin embargo, se debe aclarar que cuando una forma intenta compatibilizar sus fronteras altera su estrategia de mallado y esta estrategia permanece alterada aún cuando se reinicia la lista y se eliminan todas las restricciones. Es por esto (el cambio de las condiciones iniciales para cada cordón individualmente) que el mismo orden no necesariamente implica la obtención de los mismos resultados. Si bien no es posible demostrar formalmente que esta estrategia genera un resultado válido en todos los casos, en la práctica se ha observado que efectivamente funciona para todas las pruebas realizadas y que en la gran mayoría de los casos no han sido necesarias más de dos o tres iteraciones (alteraciones al orden inicial).

Además de compatibilizar las fronteras entre cordones, también se debe garantizar la compatibilidad de las fronteras de los sustratos. Esta tarea resulta en algunos aspectos más sencilla debido a que los sustratos se definen generalmente como dos entidades geométricas, por lo que sólo existe una frontera común entre ellas y una frontera común al conjunto de cordones. Sin embargo, como se señaló anteriormente, para mejorar el control del tamaño de los elementos es conveniente subdividir los mismos en subdominios con sentidos de desrefinamientos aproximadamente constantes generando, en consecuencia, nuevas fronteras comunes a tener en cuenta para la composición de la malla total. Además, dado que al extruir estos subdominios se utilizará el método de extrusión por bloques, será necesario colocar algunas capas de elementos denominadas de adaptación en las fronteras.

La necesidad de colocar capas de adaptación surge del hecho de que el proceso de extrusión por bloques puede generar sobre las caras laterales de la geometría extruida mallas de cuadriláteros irregulares, donde aparecen bloques de desrefinamiento y nodos con valencias iguales a 3 ó 5. Resulta imposible colocar directamente sobre una de estas superficies el resultado de extruir otro subdominio, ya que el mismo requerirá una malla de cuadriláteros topológicamente regular sobre dicha superficie común para poder comenzar a colocar los bloques de desrefinamiento (ver Figura 7).

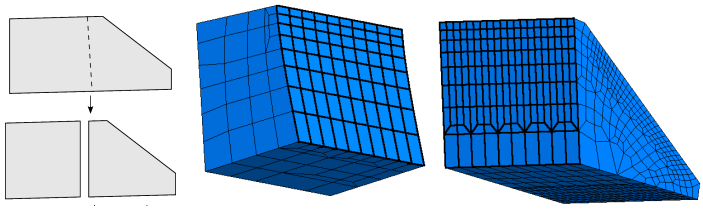

Figura 7: Fronteras incompatibles

Para corregir este problema, la capa de adaptación utiliza un conjunto de bloques especiales para la línea de elementos donde la malla se torna irregular en el primer dominio, de forma que dichas irregularidades se transfieran a una de las superficies laterales del segundo dominio (es decir, se buscará torcer las superficies duales que generan dichas irregularidades para desviarlas hacia la cara superior). Esto se logra mediante la colocación de una línea de bloques de adaptación. El resto de los elementos de la capa se obtienen por extrusión simple en dirección al mayor h, y colocando bloques 4a2 o 16a4 para rellenar hacia arriba (ver Figura 8).

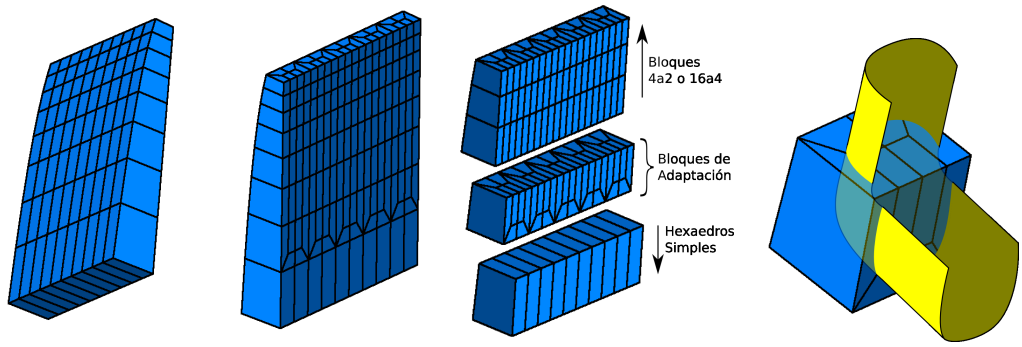

Figura 8: Composición de la capa de adaptación y su bloque base

El número de capas de adaptación requeridas está dado por la cantidad de líneas de la malla de frontera que incluyen bloques de desrefinamiento. Una desventaja del uso de estas capas es que las mallas bidimensionales de las formas que deban incluirlas no pueden ser generadas sin antes resolver completamente (incluyendo extrusión) la malla de la forma que condiciona dichas capas. Es decir, se debe extruir la primer forma, para determinar cuantas capas de adaptación se requerirán luego en la segunda, para generar en ésta una malla bidimensional que posibilite la colocación de dichas capas. Esto implica que la segunda forma no podrá colocar bloques de desrefinamiento en las primeras capas destinadas a la adaptación, sino que deberá generar una malla regular en la zona correspondiente.

La determinación del orden en que se resolverán las mallas de los sustratos está condicionada por dos factores: el sentido de desrefinamiento de cada uno de ellos y la necesidad de colocar capas de adaptación. El algoritmo de integración de sustratos, comienza analizando todos los pentágonos, ya que estos no contendrán capas de adaptación, y que eventualmente podrán ser subdivididos en un pentágono y un rectángulo para controlar más eficazmente el sentido de desrefinamiento.

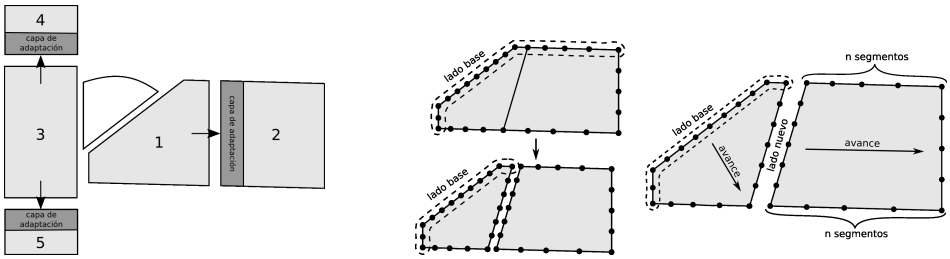

Figura 9: Subdivisión de sustratos

Una vez subdivididos todos los pentágonos, se procede a determinar el orden de mallado de los rectángulos restantes. Para ello, se tendrá en cuenta la imposición de restricciones sobre la composición de la capa de adaptación que cada uno realizará sobre los otros. Un algoritmo iterativo irá colocando en la lista los rectángulos que no reciben restricciones, o cuyas restricciones ya han sido determinadas. Cuando un rectángulo se ubica en su posición final en el orden de mallado, se determinan las restricciones que impondrá sobre los otros. De esta forma, en la siguiente iteración sobre la lista de rectángulos cuyo orden aún no se determinó se encontrará un conjunto de rectángulos cuyas restricciones acaban de ser fijadas que podrá moverse a la lista del orden final (ver Figura 9).

Una vez determinado el orden, para cada forma se procede a generar su malla bidimensional, suavizarla y extruirla. Como resultado del proceso de extrusión, además de obtenerse una parte de la malla final de hexaedros, se obtienen también las mallas de cuadriláteros (ahora en 3D) para las fronteras donde deban colocarse las capas de adaptación, para utilizar en los algoritmos de mallado de rectángulos.

4. IMPLEMENTACIÓN

La implementación final consiste en cuatro programas, dos de los cuales fueron desarrollados específicamente para este tipo de problema, mientras que los otros dos resultan de utilidad general. El sofware de mayor importancia es el generador de mallas propiamente dicho, el cual implementa todos los algortimos mencionados anteriormente, utilizando como interfaz un conjunto de archivos de texto con un formato particular que facilita la tarea. Para visualizar los resultados, realizar prototipos manualmente, y debugging gráfico durante el proceso de desarrollo se utilizaron las herramientas Quads (visualización y edición de mallas de cuadrilateros en 2 dimensiones) y MSuite (para la visualización de mallas 3D en general). Finalmente, el cuarto componente es una interfaz gráfica que permite al usuario diseñar las geometrías a mallar mediante herramientas tipo CAD, definir las propiedades particulares del problema de soldadura planteado, y encapsular de forma transparente al usuario la comunicación con el mallador.

De los cuatro componentes, el generador, el CAD, y el editor de cuadriláteros fueron desarrollados para este trabajo, mientras que MSuite es una herramienta de propósito más general desarrollada previamente a la cual sólo se le realizaron modificaciones menores para facilitar la

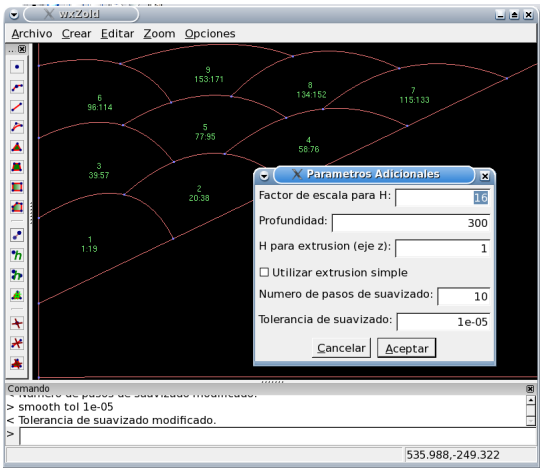

Figura 10: Captura de pantalla de la aplicación wxZold

visualización de propiedades específicas de este problema. Las tres herramientas desarrolladas en este trabajo se codificaron utilizando el lenguaje C++. El mallador sólo utiliza elementos estándar del mismo, mientras que el editor de cuadriláteros utiliza las bibliotecas OpenGL y GLUT, y la interfaz/CAD, OpenGL y wxWidgets. Por lo dicho, el resultado es altamente portable y ha sido verificado con éxito en plataformas Windows y GNU/Linux.

5. RESULTADOS

A continuación se presentan algunas mallas generadas con el software desarrollado junto con detalles del proceso de generación. Las pruebas presentadas se realizaron en una PC con un microprocesador Intel Core 2 Duo E7200 a 2.01GHz, Front Side Bus a 666MHz, 1.5GB de memoria RAM, Disco duro SATA-2 de 5400RPM y un sistema operativo basado en Slackware Linux 12.0.

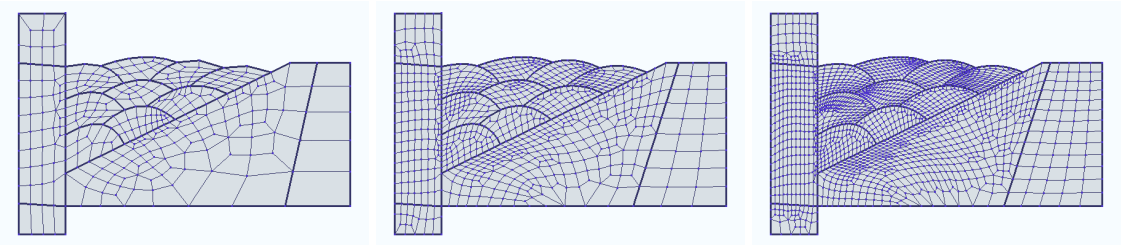

Figura 11: Malla 2D de una misma geometría variando la densidad de elementos

La figura 11 muestra las mallas bidimiensionales correspondientes a una misma geometría y la subdivsión en dominos, variando la densidad de elementos.

La figura 12 muestra una malla completa en 3D. Dicha malla se compone por dos sustratos y nueve cordones. La figura 13 muestra detalles de las zonas donde el mallador colocó bloques de desrefinamiento. La tabla 1 presenta información adicional acerca de esta malla (columna FH=16) y otras 5 similares variando el factor de escala global para los tamaños de elementos deseados. La columna FH indica dicho factor. Las columnas CN y CE indican las cantidades de nodos y elementos respectivamente. Las columnas TT, TM y TE muestran el tiempo en segundos que tomó cada generación: TT es el tiempo total (TT=TM+TE), TM es el tiempo real

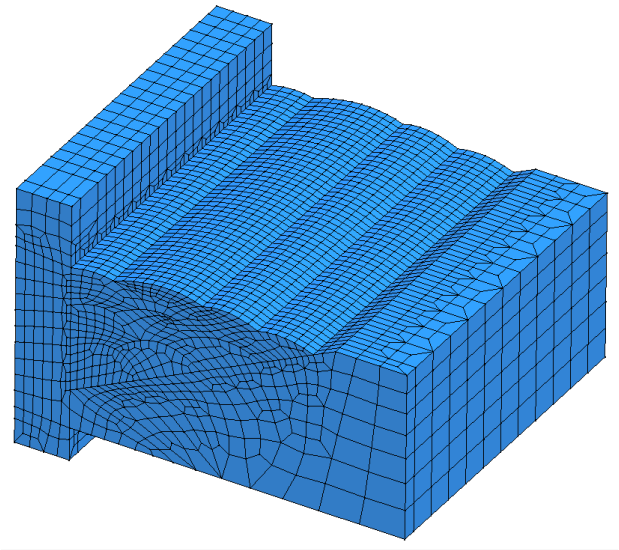

Figura 12: Malla generada por ZoldMesh

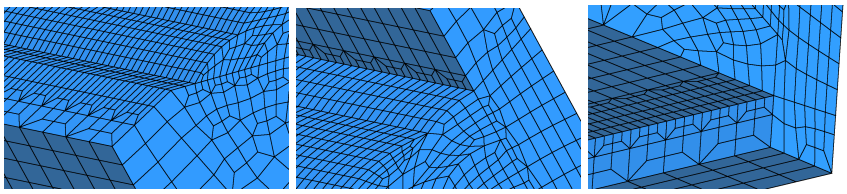

Figura 13: Detalles de la malla generada

para la generación de la malla en memoria, y TE es el tiempo que tomó la escritura en disco. La columna MV indica la cantidad de memoria total consumida por el proceso en kilobytes (incluyendo el ejecutable, las librerías, el stack y la memoria dinámica). RE muestra la cantidad de reordenamientos necesarios para encontrar un orden adecuado para el mallado de los cordones de soldadura.

FH	2	5	10	16	25	50
CN	2922198	685926	108995	35003	12928	3634
CE	2788820	641195	96564	29440	10428	2649
TT	52.58	13.86	2.14	0.67	0.22	0.06
TM	2.17	0.51	0.08	0.03	0.01	0.00
TE	50.41	13.35	2.04	0.64	0.21	0.06
MV	424840	100772	18008	7580	4676	3356
RE	0	0	0	0	0	0

Tabla 1: Detalles del proceso de generación

De la comparación mencionada se pueden realizar las siguientes observaciones:

- Para los casos más comunes, la generación de la malla en memoria toma menos de una décima de segundo

- El programa es capaz de generar en memoria una malla de más de dos millones de elementos en menos de 2.5 segundos
- La mayor parte del tiempo que consume la ejecución completa del programa se dedica a las operaciones de escritura en disco
- El tiempo que tomó la ejecución completa fue inferior al minuto en todos los casos
- El consumo de memoria para los casos habituales es inferior a 10MB
- El consumo en un caso extremo puede llegar a 0.5GB
- En ningún caso fue necesario alterar el orden inicial aproximado para lograr compatibilizar las fronteras de los cordones de soldadura

En las figuras 14 y 15 se muestra como varía el consumo de recursos según la cantidad de elementos de la malla. Se puede observar que en ambos casos, tanto para el consumo de memoria como para el tiempo ejecución, la dependencia es lineal.

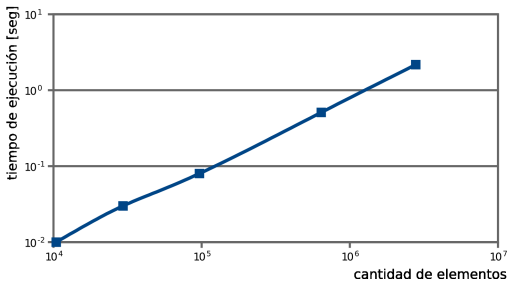

Figura 14: Relación entre la cantidad de elementos y el tiempo de mallado

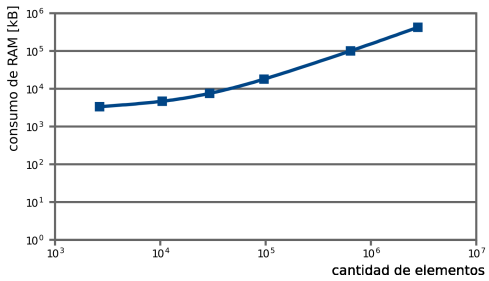

Figura 15: Relación entre la cantidad de elementos y el consumo de memoria

Las figuras 16 y 17 muestran casos en los que las mallas incluyen elementos de menor calidad. En el primer ejemplo, se han especificado tamaños de elementos relativamente grandes, que el algoritmo de compatibilización de fronteras ha debido modificar. Esta alteración, en combinación con la subdivisión interna de algunos cordones cuadrilaterales en un subcuadrilátero y un subtriángulo, genera, en uno de los nueve cordones, un conjunto de elementos pequeños en comparación al h especificado cuya transición produce ángulos de entre $24°$ y $155°$. La figura 16 muestra una posible optimización mediante un cambio topológico, eliminando el elemento donde se produce el entrecruzamiento de las líneas del dual fusionando dos de sus nodos opuestos.

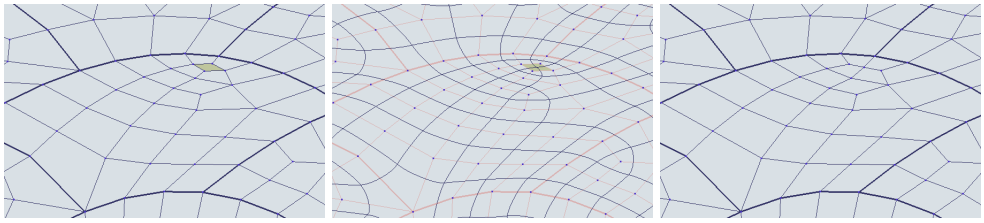

Figura 16: Elementos de baja calidad en la malla, su dual y posible solución

El segundo ejemplo corresponde a la zona triangular del pentágono, donde no se colocan bloques de desrefinamiento. En este caso el tamaño de elemento especificado es considerablemente más pequeño en la zona de los cordones en comparación al tamaño en los extremos más distantes de los sustratos. Dada la gran diferencia entre la cantidad de segmentos del lado del triángulo interior a la forma y las cantidades de los dos restantes, se debe introducir un alto número de cuadriláteros irregulares, resultando en una aplastamiento de los elementos colocados sobre el lado del triángulo común a un lado del pentágono original que presentaba mayor cantidad de segmentos. Aquí se observan ángulos de hasta $20°$.

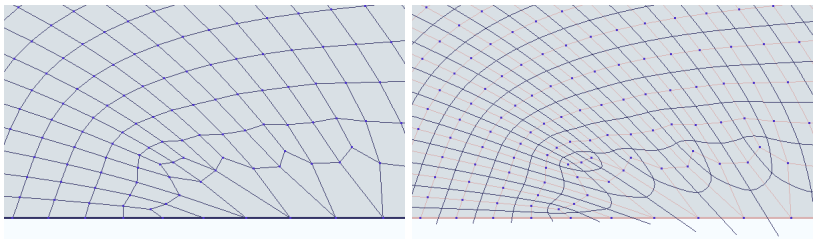

Figura 17: Elementos de baja calidad en la malla

6. CONCLUSIONES

Se han diseñado e implementado algoritmos de mallado para dar una solución completa a la generación de mallas de hexaedros para problemas de soldadura. La división del problema en subproblemas de menor complejidad facilitó enormemente el proceso de desarrollo de las soluciones individuales, pero obligó a dedicar un esfuerzo considerable al diseño de los algoritmos de ensamblado. Los métodos desarrollados para el mallado de formas simples podrían combinarse para resolver una amplia variedad de problemas cuyas geometrías pueden definirse en 2.5D. Sin embargo, el mallador implementado está diseñado para resolver específicamente el problema planteado y actualmente carece de mayor generalidad. A cambio de esta pérdida de generalidad, la solución propuesta es capaz de construir mallas de calidad aceptable con un costo computacional mínimo.

REFERENCIAS

Benzley S., Perry E., Merkely K., Clark B., y Sjaardama G. A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis. *Proc. 14th Ann. Int. Meshing Roundtable*, 1995.

Calvo N. *Generación de mallas tridimensionales por métodos duales*. Tesis de Doctorado, Doctorado en Ingeniería de la Facultad de Ingeniería y Ciencias Hídricas, 2005.

Carbonera C. y Shepherd J. A constructive approach to constrained hexahedral mesh generation. *15th Anniversary International Meshing Round Table*, páginas 435–452, 2006.

Kraft P. Automatic remeshing with hexahedral elements: Problems, solutions and applications. In *8th International Meshing Roundtable*, páginas 357–367. 1999.

Novara P. Generación de mallas para análisis de problemas de soldadura por el método de elementos finitos. Tesina de grado para la carrera Ingeniería en Informática de la Facultad de Ingeniería y Ciencias Hídricas de la Universidad Nacional del Litoral, 2009.

Owen S. A survey of unstructured mesh generation technology. *7th International Meshing roundtable*, 1998.

Owen S. An introduction to unstructured mesh generation. *10th Anniversary International Meshing Round Table, Sandia National Laboratorie*, 2001.

Pakal D. Hexahedron vs. tetrahedron elements for finite element analysis. *Curso de modelado geométrico, Carnegie Mellon University*, 1998.

Schneiders R. Octree-based hexaedral mesh generation. *International Journal of Computational Geometry & Applications*, páginas 383–398, 2000.

Schneiders R., Schindler R., y Weiler F. Octree-based generation of hexahedral element meshes. In *5th International Meshing Roundtable*. 1996.