Ruiz-Gironés, E. y Sarrate, J., Discretización de superficies mútiplemente conexas mediante submapping, *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería,* Vol. Vol 24, Issue 2, pp. 163-181, 2008

Discretización de superficies múltiplemente conexas mediante submapping

Eloi Ruiz-Gironés y Josep Sarrate

Laboratori de Càlcul Numèric (LaCàN)
Departament de Matemàtica Aplicada III
Universitat Politècnica de Catalunya
Jordi Girona 1-3, E–08034 Barcelona, España
Tel.: 34 93 401 69 11; Fax: 34 93 401 18 25
e-mail: eloi.ruiz@upc.edu; jose.sarrate@upc.edu

Resumen

Una de las técnicas más utilizadas para generar mallas estructuradas de cuadriláteros es el método de submapping. Este método descompone la geometría en piezas lógicamente equivalentes a un cuadrilátero y después malla cada una de ellas por separado manteniendo la compatibilidad de la malla mediante la resolución de un problema lineal entero. El algoritmo de submapping tiene dos limitaciones principales. La primera de ellas es que sólo se puede aplicar en geometrías tales que el ángulo entre dos aristas consecutivas es, aproximadamente, un múltiplo entero de $\pi/2$. La segunda limitación es que la geometría tiene que ser simplemente conexa. Con el objetivo de mitigar estas restricciones, en este artículo se presentan dos modificaciones originales que permiten reducir el efecto de dichas limitaciones. Finalmente, se presentan diversos ejemplos numéricos que ponen de manifiesto la robustez y la aplicabilidad de los algoritmos desarrollados.

Palabras clave: *Método de los elementos finitos, generación de mallas, submapping, cuadriláteros estructurados, programación lineal, interpolación transfinita, geometrías múltiplemente conexas.*

DISCRETIZATION OF MULTIPLY CONNECTED SURFACES USING SUBMAPPING

Summary

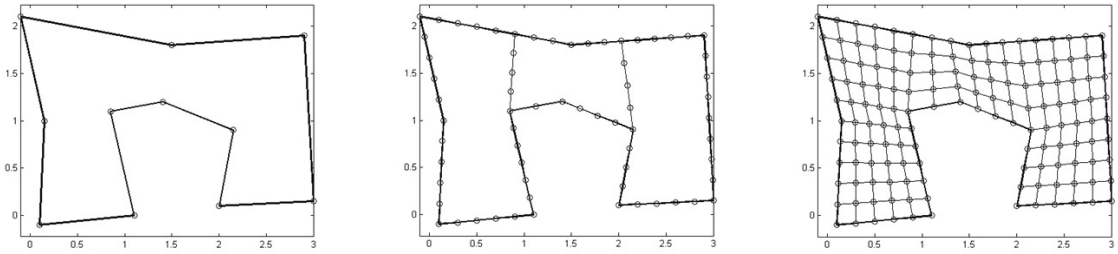
The submapping method is one of the most used techniques to generate structured quadrilateral meshes. This method splits the geometry into pieces logically equivalent to a quadrilateral. Then, it meshes each piece keeping the mesh compatibility by solving an integer linear problem. The submapping algorithm has two main limitations. First, the method can only be applied to geometries that the angle between edges is, approximately, an integer multiple of $\pi/2$. Second, the geometry must be simply connected. This article presents two original modifications to mitigate these limitations. Finally, it presents several numerical examples that show the applicability of the developed algorithms.

Keywords: *Finite elements method, mesh generation, submapping, structured quadrilaterals, linear programming, transfinite interpolation, multiply connected geometries.*

INTRODUCCIÓN

La generación de mallas se ha convertido en uno de los aspectos más relevantes de la aplicación del método de los elementos finitos a la resolución de problemas en ingeniería. Dicha generación es de gran importancia ya que una malla mal construida, en el sentido del tamaño y de la forma de los elementos, puede producir grandes errores en la solución numérica obtenida. Además, la construcción de una buena malla en un dominio complejo, que se adapte tanto a la geometría como al tamaño prescrito por el usuario, no es una tarea trivial y es costosa tanto en tiempo de cálculo como en tiempo de dedicación de personal cualificado. En la actualidad, la generación de la malla puede llegar a representar hasta el 80 % del tiempo total de una simulación de interés industrial.

Una de las técnicas más potentes para generar mallas estructuradas de cuadriláteros es el método de submapping[1,2]. La idea básica del método consiste en descomponer la geometría en piezas lógicamente equivalentes a un cuadrilátero y, a continuación, mallar cada pieza por separado preservando la compatibilidad entre las piezas mediante la resolución de un problema lineal entero. En nuestra implementación, se ha usado el método de la interpolación transfinita (TFI) [4] para mallar cada subdominio. La Figura 1 muestra la idea general del método de submapping. Es decir, dada una geometría inicial (Figura 1(a)), ésta se descompone en piezas (Figura 1(b)) y después se genera una malla en cada una de las piezas (Figura 1(c)).



(a) (b) (c)

Figura 1. Geometría mallada mediante el método de submapping. (a) Geometría inicial. (b) Geometría descompuesta. (c) Geometría mallada

Aunque el método de submapping es capaz de producir mallas de gran calidad, el método no es general y, por lo tanto, no se puede aplicar en cualquier geometría. El método tiene dos limitaciones que reducen su aplicabilidad. La primera limitación es que los ángulos que forman dos aristas consecutivas de la geometría tienen que ser, aproximadamente, múltiplos enteros de $\pi/2$. La segunda limitación es que la geometría tiene que ser simplemente conexa. Es decir, no puede tener *agujeros*. En este trabajo se presentan dos mejoras del método de submapping para reducir dichas limitaciones y mejorar la aplicabilidad del método [3]. Concretamente, se presenta un nuevo algoritmo para clasificar los nodos que forman la frontera del dominio. De esta forma, se puede reducir sustancialmente la restricción de que las aristas del contorno deben formar ángulos que sean aproximadamente múltiplos enteros de $\pi/2$. Por otra parte, se presenta un algoritmo automático que permita extender el método de submapping a dominios no simplemente conexos.

La estructura del artículo es la siguiente. En el segundo apartado se revisan los aspectos más importantes del método de submapping para superficies. En el tercer apartado se presenta el nuevo algoritmo para clasificar los vértices del contorno. En el cuarto apartado se presenta un algoritmo desarrollado para extender el método de submapping a dominios no simplemente conexos. En el quinto apartado se presentan diversos ejemplos que ponen de

manifiesto la aplicabilidad de los algoritmos desarrollados. Finalmente, en el sexto apartado se presentan las conclusiones de este trabajo.

SUBMAPPING

El método de submapping para superficies es una de las técnicas más potentes para obtener mallas estructuradas de cuadriláteros en geometrías bidimensionales. Aunque no todas las geometrías pueden ser malladas mediante este método, muchos casos de interés práctico pueden ser discretizados con el método de submapping. Tal y como se ha comentado en la introducción, la idea básica del método consiste en descomponer la geometría en piezas lógicamente equivalentes a un cuadrilátero y, a continuación, mallar cada pieza por separado manteniendo la compatibilidad de la malla entre las piezas. Dicha compatibilidad se impone a través de un problema de programación lineal entera.

Con el fin de generar una malla estructurada, se construirá una representación de la geometría en la que todas las aristas sean horizontales o verticales. Esta representación se define como el *espacio computacional* mientras que la geometría inicial se define como el *espacio físico*. El espacio computacional se usará para facilitar la tarea de descomponer la geometría. La Figura 2 muestra el espacio físico y el espacio computacional correspondiente a una geometría dada.

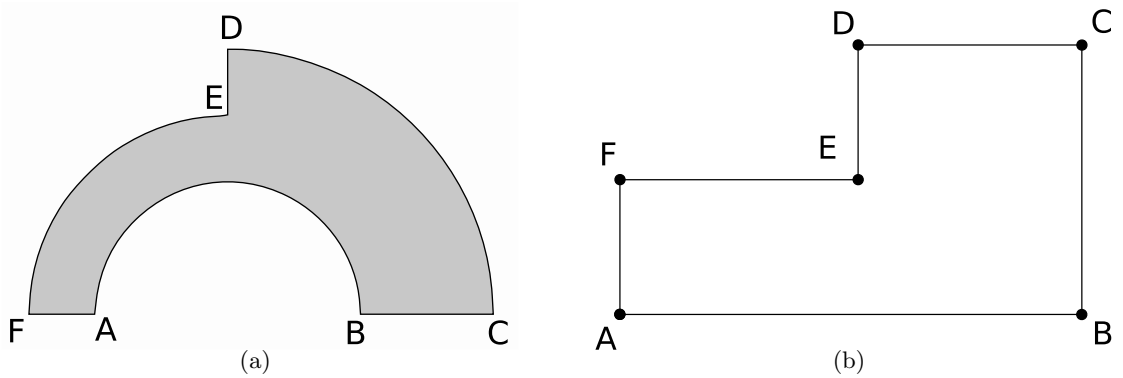


Figura 2. Espacio físico y computacional de una geometría. (a) Espacio físico. (b) Espacio computacional

El algoritmo de submapping se puede descomponer en seis pasos:

1. Clasificación de los vértices.
2. Clasificación de las aristas.
3. Discretización de la frontera
4. Construcción del espacio computacional.
5. Descomposición de la geometría en piezas lógicamente equivalentes a un cuadrilátero.
6. Discretización de cada pieza mediante el método TFI.

En los siguientes apartados se describe con más detalle cada uno de los pasos anteriores.

Clasificación de los vértices

Los vértices de la geometría inicial se clasifican según el ángulo que forman las dos aristas adyacentes. Este ángulo se define a través del ángulo que forman las tangentes de las aristas en el vértice común (Figura 3(a)). Puesto que los ángulos son aproximadamente múltiplos enteros de $\pi/2$, los vértices se clasifican como:

- **Lado**: el ángulo entre las arista es aproximadamente 0.
- **Final**: el ángulo entre las arista es aproximadamente $\pi/2$.
- **Reverso**: el ángulo entre las arista es aproximadamente π.
- **Esquina**: el ángulo entre las arista es aproximadamente $3\pi/2$.

La Figura 3(b) muestra una geometría en la que aparecen todos los tipos de vértices adecuadamente clasificados.

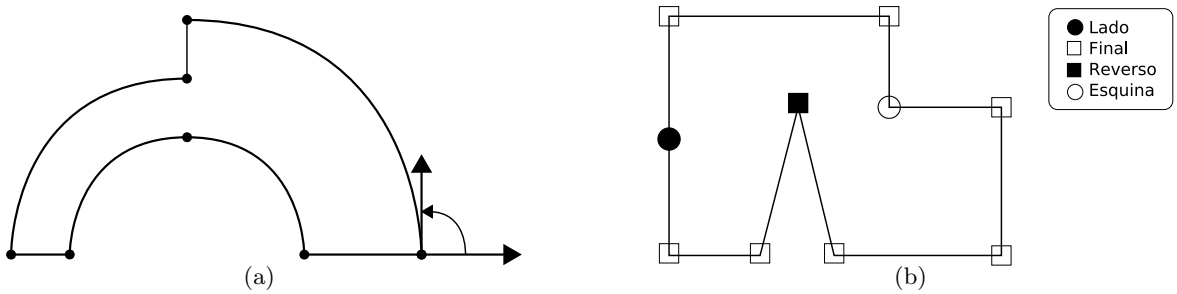


Figura 3. Clasificación de los vértices. (a) Ángulo definido entre dos aristas de la geometría presentada en la Figura 2. (b) Geometría en el espacio físico y su clasificación de los vértices

Una vez que los vértices están clasificados, se puede comprobar fácilmente si la superficie se puede mallar con el método de submapping. Concretamente, si la superficie es submallable, se tiene que cumplir la siguiente condición:

$$E - C - 2R = 4, \tag{1}$$

donde E, C y R son el número de vértices clasificados como *final*, *esquina* y *reverso*, respectivamente. Tradicionalmente, si no se cumple esta condición, el dominio no es mallable mediante submapping. En este sentido, se debe resaltar que en la tercera sección se desarrollará un nuevo algoritmo para conseguir una clasificación de los vértices que la cumpla. Obsérvese que esta condición es una condición necesaria, aunque no es suficiente. Por lo tanto, existen geometrías que cumplen esta condición y no pueden ser discretizadas mediante el método de submapping.

Es importante resaltar que la clasificación de los vértices es uno de los pasos más importantes del algoritmo ya que los pasos siguientes se basarán en esta clasificación. Si no se puede conseguir una clasificación correcta de los vértices, el algoritmo no finalizará correctamente.

Clasificación de las aristas

Las aristas de la geometría se clasifican según la dirección y sentido que tomarán en el espacio computacional. Por construcción, todas las aristas en el espacio computacional serán horizontales o verticales. Por consiguiente, cada arista se puede clasificar como:

- **+i**: la arista es horizontal y se mueve de izquierda a derecha.
- **-i**: la arista es horizontal y se mueve de derecha a izquierda.
- **+j**: la arista es vertical y se mueve de abajo a arriba.
- **-j**: la arista es vertical y se mueve de arriba a abajo.

La Figura 4 muestra la representación del espacio computacional de la geometría presentada en la Figura 2 y la clasificación de sus aristas.

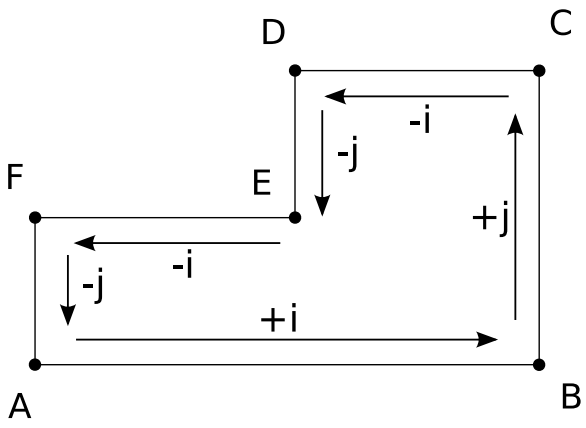


Figura 4. Clasificación de las aristas en el espacio computacional de la geometría presentada en la Figura 2

Nótese que el proceso de clasificación de las aristas se puede resolver sin tener explícitamente construido el espacio computacional (todavía no se ha definido ni la posición ni la longitud de cada arista). Para clasificar las aristas, se usará la clasificación de los vértices presentada anteriormente. La primera arista se clasifica como $+i$ de forma arbitraria. La clasificación de la siguiente arista viene inducida por la clasificación de su vértice común. De esta forma, si su vértice común es *final*, quiere decir que se gira $\pi/2$ en este vértice y, por lo tanto, la siguiente arista será clasificada como $+j$. En cambio, si la clasificación de su vértice común es *esquina*, entonces la clasificación de la siguiente arista será $-j$, ya que se ha girado $3\pi/2$ en el vértice.

Cuando el proceso de clasificación de las aristas concluye, se debe encontrar que la clasificación de la primera arista es $+i$. En caso contrario, los vértices no están correctamente clasificados o ha habido algún error en el proceso de clasificación de las aristas.

Finalmente, se definen los siguientes conjuntos de aristas:

$$\begin{aligned} I^+ &= \{\text{aristas clasificadas como } +i\}, \\ I^- &= \{\text{aristas clasificadas como } -i\}, \\ J^+ &= \{\text{aristas clasificadas como } +j\}, \\ J^- &= \{\text{aristas clasificadas como } -j\}. \end{aligned}$$

Discretización de la frontera

Para construir una malla estructurada de cuadriláteros, es necesario imponer las siguientes condiciones de compatibilidad en las aristas que definen la frontera de la geometría:

$$\begin{array}{rcl} \displaystyle\sum_{e\in I^+} n_e & = & \displaystyle\sum_{e\in I^-} n_e, \\ \displaystyle\sum_{e\in J^+} n_e & = & \displaystyle\sum_{e\in J^-} n_e, \end{array} \tag{2}$$

donde n_e es el número de divisiones (elementos) de la arista e. A fin de mantener el número de nodos en una cantidad razonable y, a la vez, cumplir la ecuación de compatibilidad (2), según[5] se debe solucionar el siguiente problema lineal:

$$\left\{ \begin{array}{l} \text{mín} \displaystyle\sum_e \omega_e n_e + M, \\ \text{restringido a:} \\ \displaystyle\sum_{e\in I^+} n_e = \sum_{e\in I^-} n_e, \\ \displaystyle\sum_{e\in J^+} n_e = \sum_{e\in J^-} n_e, \\ M \geq \dfrac{n_e}{N_e}, \qquad \text{para toda arista } e \\ n_e \geq N_e, \qquad \text{para toda arista } e \end{array} \right. \tag{3}$$

donde $\omega_e \geq 0$ son unos pesos y $N_e \geq 1$ es una cota inferior del número de elementos para la arista e. Es usual definir los pesos como $\omega_e = 1/l(e)$, donde $l(e)$ es la longitud de la arista e. De esta forma, se favorece que el algoritmo añada más elementos en las aristas de mayor longitud. La variable M tiene el siguiente significado:

$$M = \max_e \left\{ \frac{n_e}{N_e} \right\}.$$

Obsérvese que este problema tiene la formulación de un problema lineal entero. Por consiguiente, se puede solucionar mediante el método de Branch&Bound o el del punto interior[6]. En nuestra implementación se utiliza el primero de ellos. Además, en este trabajo se ha usado la librería GLPK[7] para resolver el problema lineal (3).

La resolución del problema lineal (3) proporciona las siguientes propiedades en la solución:

- La solución cumple la restricción de las ecuaciones de compatibilidad (2) y, por lo tanto, permitirá construir una malla de cuadriláteros estructurada en el interior de la geometría.
- Mediante el uso de las constantes N_e, el usuario prescribe el tamaño de elemento deseado en cada arista.
- El primer término en la función objetivo, $\displaystyle\sum_e \omega_e n_e$, tiene el efecto de minimizar el número de elementos en las aristas.
- El efecto de la variable M en la función objetivo es minimizar el máximo incremento del número de elementos en cada arista respecto el impuesto por el usuario.

Construcción del espacio computacional

Una vez solucionado el problema de compatibilidad (3), se construye el espacio computacional. Recuérdese que las aristas en el espacio computacional son horizontales o verticales. Además, también se impone que la longitud de cada arista sea el número de elementos hallado mediante la minimización del problema lineal (3). De esta manera, el tamaño de elemento en el espacio computacional será la unidad. Por lo tanto, la posición de cada nodo del espacio computacional tendrá coordenadas enteras. Finalmente, es preciso observar que las ecuaciones de compatibilidad (2) son necesarias para obtener un bucle de nodos cerrado que definen la frontera del espacio computacional. Por lo tanto, se puede comprobar la solución del problema de compatibilidad (3) en el momento de construir el espacio computacional.

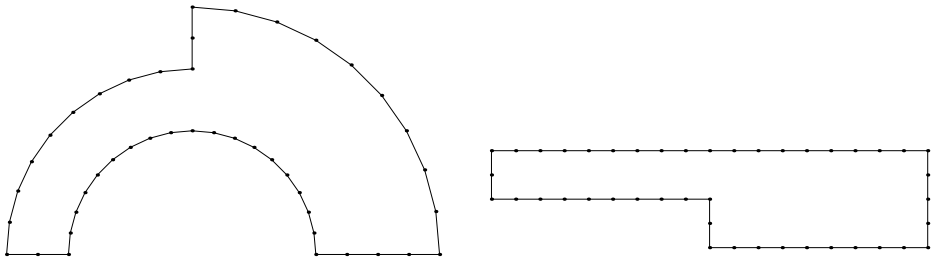


Figura 5. Espacio físico y computacional de la geometría presentada en la Figura 2. (a) Malla generada en la frontera del espacio físico. (b) Malla generada en la frontera del espacio computacional

La Figura 5 muestra la discretización del contorno (tanto en el dominio físico como en el computacional) de la geometría presentada en la Figura 2.

Descomposición de la geometría

Hasta este punto, se ha discretizado el contorno de la superficie y se ha creado el espacio computacional, donde se calcularán los cortes necesarios para mallar la geometría. Concretamente, se impone que los cortes de la geometría se producen en las zonas de concavidad. Por lo tanto, los puntos iniciales de los cortes estarán situados en nodos clasificados como *esquina* o *reverso*. Además, también se impone que los cortes sean horizontales o verticales en el espacio computacional. Por este motivo, los nodos clasificados como *esquina* tienen dos posibles líneas de corte, mientras que los nodos clasificados como *reverso* tienen tres. La Figura 6 muestra las posibles líneas de corte correspondientes a un nodo *esquina* y a un nodo *reverso* en el espacio computacional.

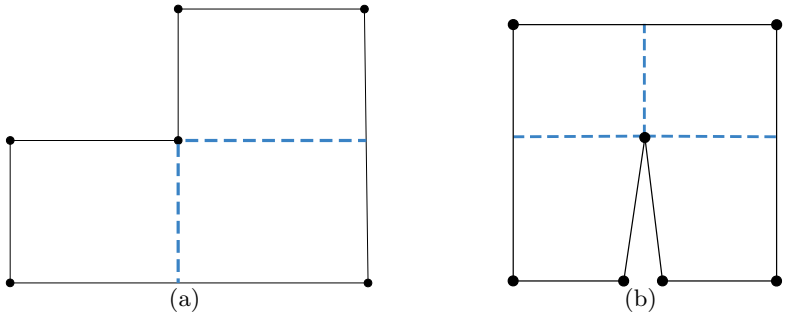


Figura 6. Posibles líneas de corte para nodos *esquina* y *reverso*. (a) Nodo esquina. (b) Nodo reverso

En la implementación realizada, para dividir la geometría se escoge la línea de corte más corta en el espacio computacional. Una vez encontrada la línea de corte, se deberá mallar dicha línea y se procederá a subdividir la geometría en dos piezas. Por lo tanto, resulta imprescindible saber cuantos elementos tendrá la línea de corte. En el espacio computacional, la longitud de una arista coincide con el número de elementos que contiene. En consecuencia, el número de elementos que tendrá la línea de corte será igual a su longitud. Seguidamente, el algoritmo se itera en cada pieza de forma recursiva hasta que no haya nodos *esquina* ni *reverso*.

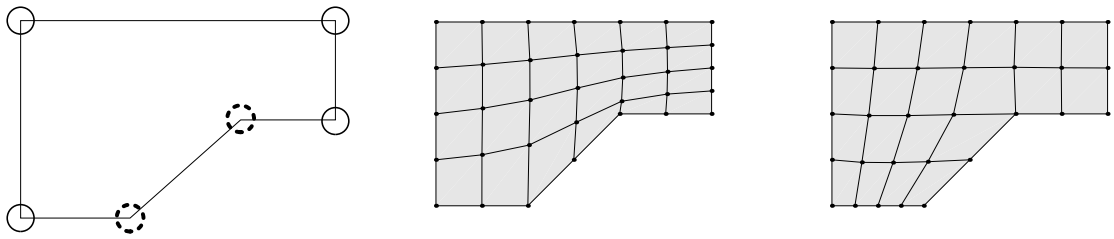
Discretización de los subdominios

Una vez que se ha obtenido una pieza de la geometría que no contiene nodos *esquina* ni *reverso*, ésta puede ser mallada. Para este fin, se utilizará el método de interpolación transfinita[4]. Como las piezas son lógicamente equivalentes a un cuadrilátero, existe una transformación entre el cuadrado unidad y la pieza que se desea mallar. Por lo tanto, se puede transferir una malla desde el cuadrado unidad hasta dicha pieza. El método TFI permite construir esta transformación.

ALGORITMO DE CLASIFICACIÓN DE LOS VÉRTICES

En las secciones anteriores, se ha resaltado que es necesario detectar una clasificación válida de los vértices ya que los pasos siguientes se basan en dicha clasificación. Recuérdese que una clasificación válida de los vértices debe cumplir la condicion (1).

Si en la geometría hay presentes ángulos dudosos (*fuzzy angles*), éstos se pueden clasificar de diversas formas. Entre todas esas clasificaciones, se debe de escoger una que cumpla la condición (1) y que, además, no sea demasiado diferente de la clasificación encontrada inicialmente. La Figura 7(a) muestra una geometría simple con dos ángulos dudosos. Para obtener una clasificación válida de los vértices, éstos se pueden clasificar como *lado* y *lado* o como *final* y *esquina*. Por consiguiente para esta geometría se pueden generar dos mallas diferentes. La primera malla, Figura 7(b), se ha generado clasificando los vértices dudosos como *lado* y *lado*. La segunda malla, Figura 7(c), se ha generado clasificando los vértices dudosos como *final* y *esquina*. Obsérvese que para cada clasificación se obtiene una malla distinta, aunque las dos son igualmente válidas.



(a) (b) (c)

Figura 7. Mallas generadas mediante el método de submapping con diferentes clasificaciones de los vértices. (a) Geometría con dos ángulos dudosos. (b) Clasificación de los vértices dudosos como *lado* y *lado*. (c) Clasificación de los vértices dudosos como *final* y *esquina*

El objetivo de esta sección es encontrar una nueva clasificación de los vértices que cumpla la ecuación (1) y que permita aplicar el método de submapping en un espectro más amplio de geometrías. Sean θ_i el ángulo que forman las aristas adyacentes al vértice i, definido

entre $-\pi \leq \theta_i < \pi$. Entonces, se cumple

$$\sum_{i=1}^{N_{vertices}} \theta_i = 2\pi.$$

Dividiendo la ecuación anterior entre $\pi/2$ se obtiene:

$$\sum_{i=1}^{N_{vertices}} \alpha_i = 4, \tag{4}$$

donde $\alpha_i = \dfrac{\theta_i}{\pi/2}$. Obsérvese que si la geometría estuviese definida por vértices cuyas aristas adyacentes formasen múltiplos de ángulos rectos, las variables α_i serían enteras. Sin embargo, para una geometría cualquiera, las variables α_i no tienen porque ser enteras. De hecho, verifican:

- **Lado**: el valor de α es aproximadamente 0.
- **Final**: el valor de α es aproximadamente 1.
- **Reverso**: el valor de α es aproximadamente -2.
- **Esquina**: el valor de α es aproximadamente -1.

Teniendo en cuenta estos valores de α_i, la ecuación (4) se puede reescribir como

$$\sum_{i=1}^{N_{vertices}} \alpha_i = \Big(0 \cdot S\Big) + \Big(1 \cdot E\Big) - \Big(2 \cdot R\Big) - \Big(1 \cdot C\Big),$$

donde S, E, R y C son el número de vértices clasificados como *lado*, *final*, *reverso* y *esquina*, respectivamente. Finalmente, simplificando se obtiene

$$E - C - 2R = 4.$$

Obsérvese que esta ecuación es la misma que la definida en (1).

Con el fin de obtener una clasificación de los nodos tal que los valores de α_i sean enteros y cumplan la restricción (4), se plantea minimizar el siguiente problema:

$$\begin{cases} \text{mín} \displaystyle\sum_{i=1}^{N_{vertices}} |\alpha_i - \overline{\alpha_i}|, \\ \text{restringido a:} \\ \displaystyle\sum_{i=1}^{N_{vertices}} \alpha_i = 4, \end{cases} \tag{5}$$

donde $\overline{\alpha_i}$ es la clasificación detectada inicialmente del vértice i. Nótese que este problema no sigue la formulación de un problema lineal entero debido a que la función objetivo no es una función lineal (contiene valores absolutos). A continuación, se transforma el problema (5) en un problema de programación lineal entera modificando la forma de la función objetivo.

Cada valor absoluto se descompone en la suma de dos variables, D_i y d_i, introduciendo dos nuevas ecuaciones. De esta manera, se obtiene:

$$\begin{aligned} |\alpha_i - \overline{\alpha_i}| &= D_i + d_i, \qquad i = 1, \ldots, N_{vertices}, \\ D_i &\geq \alpha_i - \overline{\alpha_i}, \qquad i = 1, \ldots, N_{vertices}, \\ d_i &\geq \overline{\alpha_i} - \alpha_i, \qquad i = 1, \ldots, N_{vertices}. \end{aligned}$$

Es fácil comprobar que si $\alpha_i - \overline{\alpha_i} \geq 0$, entonces, $D_i = \alpha_i - \overline{\alpha_i}$ y $d_i = 0$. Por otro lado, si $\alpha_i - \overline{\alpha_i} \leq 0$, entonces, $D_i = 0$ y $d_i = \overline{\alpha_i} - \alpha_i$. Se dice que la variable D_i es la parte positiva del valor absoluto mientras que la variable d_i es la parte negativa. Finalmente, para limitar la variación de la solución vértice a vértice, se imponen las ecuaciones

$$D_i + d_i \leq 1, \qquad i = 1, \ldots, N_{vertices}.$$

Estas últimas ecuaciones son equivalentes a

$$|\alpha_i - \overline{\alpha_i}| \leq 1, \qquad i = 1, \ldots, N_{vertices}.$$

De esta forma, además de minimizar la variación global de la solución, también se limitan los cambios de clasificación vértice a vértice. Insertando las expresiones anteriores en el problema (5), se obtiene:

$$\begin{cases} \text{mín} \displaystyle\sum_{i=1}^{N_{vertices}} \rho_i D_i + \omega_i d_i, \\ \text{restringido a:} \\ \displaystyle\sum_{i=1}^{N_{vertices}} \alpha_i = 4, \\ D_i \geq \alpha_i - \overline{\alpha_i}, & i = 1, \ldots, N_{vertices} \\ d_i \geq \overline{\alpha_i} - \alpha_i, & i = 1, \ldots, N_{vertices} \\ D_i + d_i \leq 1, & i = 1, \ldots, N_{vertices} \\ D_i, d_i \geq 0, & i = 1, \ldots, N_{vertices} \end{cases} \tag{6}$$

donde se ha cambiado la función objetivo mediante una suma ponderada con pesos ρ_i y ω_i para la parte positiva del valor absoluto, D_i, y la parte negativa del valor absoluto, d_i, respectivamente. Obsérvese que el problema (6) es un problema de programación lineal entera y que puede ser resuelto por el mismo método que el problema de compatibilidad (3).

Como se ha comentado anteriormente, este método proporciona soluciones que cumplen la ecuación de compatibilidad (1). Aunque es cierto que si una superficie se puede mallar mediante el método de submapping entonces se cumple dicha condición, no es cierto el recíproco. Por lo tanto, este método puede proporcionar soluciones no válidas. Básicamente, estas soluciones estan asociadas a vértices cuyos lados adyacentes forman ángulos que difieren sustancialmente de múltiplos enteros de $\pi/2$ (ángulos dudosos). Finalmente, aunque este método aumenta la automatización del proceso, a veces es mejor dejar que el usuario escoja la clasificación de los vértices.

Por lo tanto, la estrategia que se plantea en esta trabajo consta de dos pasos. Primero, se clasifican los vértices de acuerdo con la clasificación presentada inicialmente. Si esta clasificación no verifica la ecuación (1), entonces los vértices se clasifican resolviendo el problema lineal (6). A continuación, el método de submapping procede tal y como se ha descrito en esta sección, teniendo en cuenta la nueva clasificación de los vértices.

DOMINIOS NO SIMPLEMENTE CONEXOS

El algoritmo detallado hasta este punto es válido para superficies simplemente conexas. Las superficies múltiplemente conexas estan definidas por un bucle de aristas correspondiente a la frontera exterior y tantos bucles de aristas como contornos interiores. Por consiguiente, una vez colocado en el espacio computacional el bucle de aristas correspondiente a la frontera exterior, aparece el problema de colocar correctamente los bucles de aristas correspondientes a la frontera interior. Una posible solución para mallar superficies múltiplemente conexas mediante el método de submapping consiste en convertir la geometría inicial en una geometría simplemente conexa. Para este fin, se construyen aristas virtuales que conectan las fronteras interiores con la frontera exterior de la geometría. Las aristas virtuales, o segmentos de retorno, consisten en dos aristas superpuestas en las que los nodos generados coinciden. La Figura 8(a) presenta una geometría múltiplemente conexa definida por dos bucles de aristas. La Figura 8(b) muestra, en línea discontinua, dos aristas, una que conecta la frontera exterior con la interior y otra que conecta la frontera interior con la exterior. Finalmente, la Figura 8(c) muestra la geometría convertida a simplemente conexa.

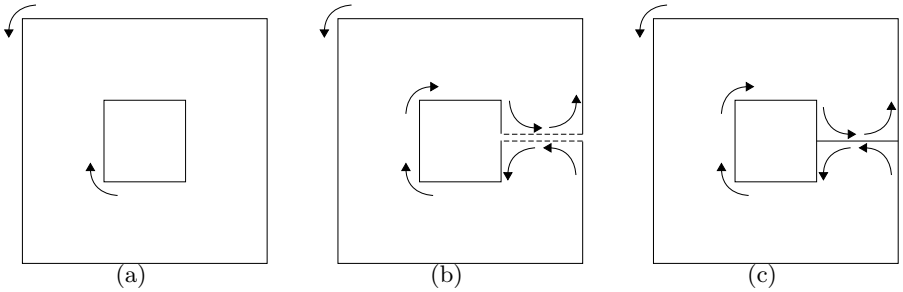


Figura 8. Superficie múltiplemente conexa convertida en simplemente conexa mediante una arista virtual. (a) Geometría múltiplemente conexa. (b) Arista virtual de corte. (c) Geometría simplemente conexa

En la referencia[1], se propone una alternativa para mallar este tipo de geometrías. Concretamente, la determinación de aristas virtuales se realiza en cuatro fases:

(I) Dada una geometría inicial múltiplemente conexa, se genera una malla de submapping sin tener en cuenta los contornos interiores. De esta forma, se tiene una malla estructurada de cuadriláteros que cubre los agujeros.

(II) Se ordenan los contornos interiores de más cercano a más lejano del contorno exterior, midiendo la distancia en el espacio físico. Los contornos interiores se procesarán por dicho orden.

(III) Para cada contorno interior, se escoge el vértice más cercano a la frontera exterior. Dicho vértice se define como el vértice inicial. A continuación, el vértice inicial se asocia al nodo de la malla que tiene más cercano. Este nodo se define como el nodo inicial.

(IV) Desde el nodo inicial, en el espacio computacional se busca la línea de corte más corta tal y como se ha detallado en el apartado anterior. Se define el nodo final como el nodo situado al final de la línea de corte. Entonces, se crea la arista virtual que une el vértice inicial y el nodo final.

(V) Una vez que todos los contornos interiores están conectados con el contorno exterior, el algoritmo de submapping procede de forma usual.

La Figura 9 muestra una geometría múltiplemente conexa convertida a simplemente conexa mediante dicho método. La desventaja de utilizar este método radica en que hay superficies múltiplemente conexas tales que no se puede generar una malla estructurada utilizando el método de submapping usando únicamente el contorno exterior y, en cambio, sí que se puede discretizar la geometría original mediante submapping. Obsérvese que es prerequisito poder discretizar mediante el método de submapping la geometría descrita por el contorno exterior. En el apartado de ejemplos numéricos se presenta una geometría múltiplemente conexa, ver Figura 16(a), en la que el contorno exterior no se puede mallar de forma automática mediante el método de submapping. Por consiguiente, el método anteriormente presentado no se podría aplicar.

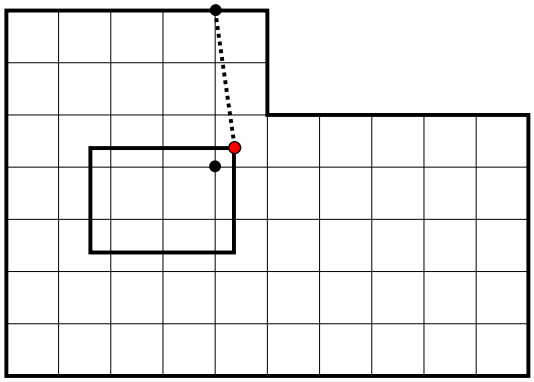


Figura 9. Geometría múltiplemente conexa convertida en simplemente conexa

El algoritmo propuesto en este trabajo se diferencia en que las aristas de corte no se buscan en el dominio computacional de una malla de submapping. De esta forma, no aparece el problema descrito anteriormente. En nuestro método, se impone que las aristas virtuales preserven la condición de los ángulos. Es decir, los nuevos ángulos creados insertando aristas virtuales tienen que ser aproximadamente múltiplos enteros de $\pi/2$.

La estrategia que se plantea consta de dos pasos:

(I) Cálculo de las aristas virtuales necesarias para convertir la geometría inicial en simplemente conexa.

(II) Una vez la geometría es simplemente conexa, el algoritmo de submapping puede proceder normalmente.

Con el objetivo de encontrar las aristas virtuales, primero se genera una discretización de la frontera del dominio. A partir de dicha discretización se genera una triangulación restringida de Delaunay del interior de la superfície[8]. Es importante resaltar que los vértices de dicha triangulación simpre pertenecen al contorno del dominio. Las aristas de la triangulación de Delaunay serán las candidatas a nuevas aristas geométricas. En nuestra implementación, la triangulación ha sido generada mediante la librería Triangle[9] y la discretización de la frontera del dominio tiene el mismo tamaño de elemento que la discretización que se quiere generar mediante el método de submapping. La Figura 10(a) muestra la triangulación restringida de Delaunay de una geometría múltiplemente conexa.

La conversión de una geometría múltiplemente conexa en simplemente conexa se realiza en cuatro fases:

(I) Se selecciona el bucle exterior de aristas.

(II) De entre todas las aristas de la triangulación de Delaunay que conectan el bucle exterior de aristas con algún bucle interior de aristas, se selecciona aquella que proporcione los mejores ángulos. Es decir, que los ángulos entre la arista de la malla y de la arista geométrica se *parezcan* más a un múltiplo entero de $\pi/2$. A igualdad de ángulos, se escoge la arista más corta.

(III) Si los extremos de la arista virtual escogida cortan alguna arista geométrica (una arista perteneciente al contorno), las aristas geométricas se dividen en dos.

(IV) Se actualiza el bucle exterior de aristas añadiendo la arista virtual y las aristas correspondientes al contorno interior conectado. A continuación, se repiten los pasos (II)-(IV) hasta que no haya más contornos interiores que conectar.

La Figura 10 presenta una geometría múltiplemente conexa convertida a simplemente conexa mediante el algoritmo detallado anteriormente. En cada una de las subfiguras, se resalta en línea gruesa el estado del bucle exterior de aristas. Obsérvese que una vez se ha conectado un contorno interior con el bucle de aristas que define el dominio exterior, el nuevo dominio exterior pasa a ser la unión de: 1. el dominio exterior original; 2. el bucle de aristas del dominio interior; 3. la arista virtual que los une. Finalmente, se muestra la malla generada en dicha superficie.

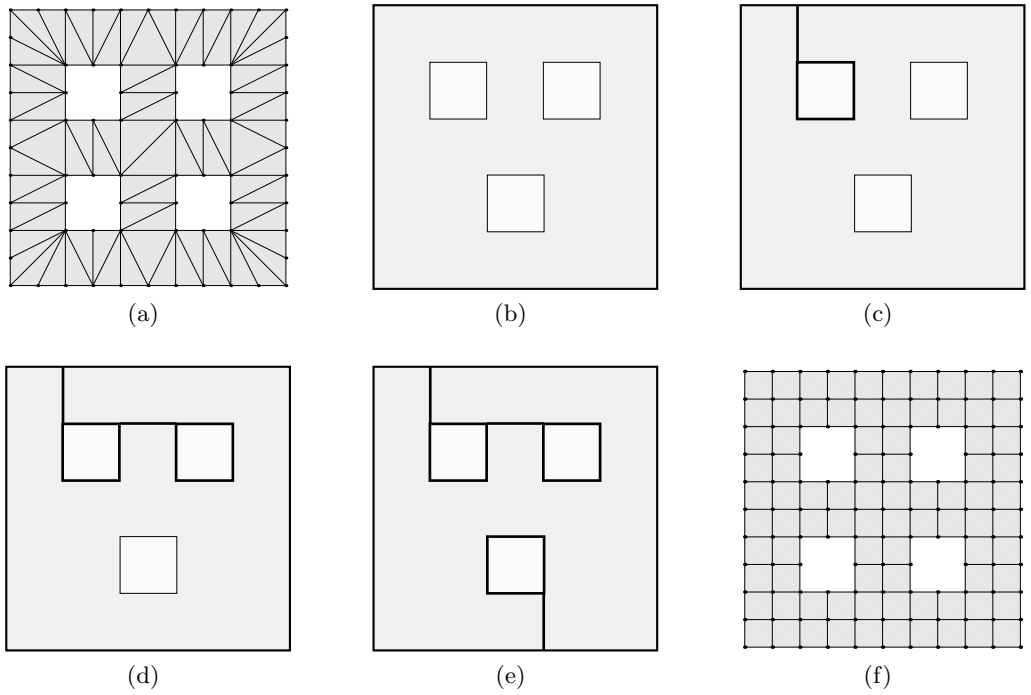


Figura 10. Geometría múltiplemente conexa convertida a simplemente conexa. (a) Triangulación restringida de Delaunay. (b) Primer paso: contorno exterior. (c) Segundo paso: unión del contorno exterior con el primer contorno interior. (d) Tercer paso: unión del contorno exterior con el segundo contorno interior. (e) Cuarto paso: unión del contorno exterior con el tercer contorno interior. (f) Malla final

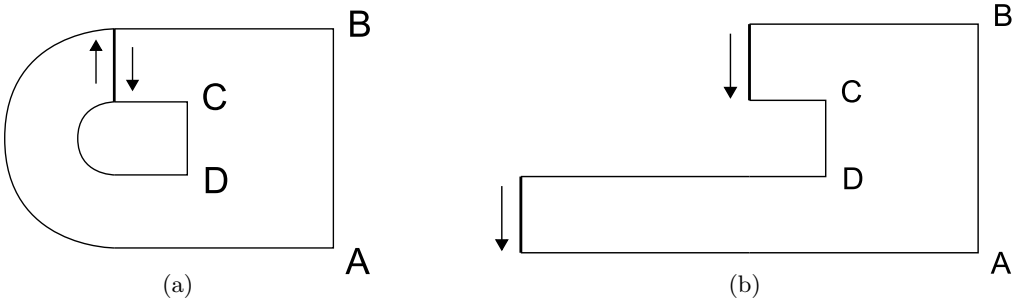


Figura 11. Espacio físico y computacional de una geometría convertida a simplemente conexa. (a) Espacio físico. (b) Espacio computacional

Una vez convertida la geometría en simplemente conexa, se consideran las aristas virtuales como aristas geométricas y el algoritmo de submapping puede proceder normalmente. Es importante hacer tres observaciones. Primero, las aristas virtuales aparecen dos veces en el bucle de aristas y, por este motivo tendrán dos clasificaciones. Aunque en el espacio físico una arista virtual aparece en dos sentidos opuestos, esta afirmación no es válida para el espacio computacional. La Figura 11 muestra un ejemplo de una geometría en la que no es válida la afirmación anterior. En el espacio físico, la arista virtual (resaltada en negrita) es vertical y aparece en el bucle de aristas en sentidos opuestos. En cambio, en el espacio computacional la arista virtual aparece dos veces en el mismo sentido.

La segunda observación consiste en resaltar que las aristas utilizadas para convertir la geometría en simplemente conexa son virtuales. Por lo tanto, se puede usar un suavizador de mallas para eliminar las pequeñas imperfecciones introducidas y adaptar mejor la malla a la geometría.

Finalmente, la tercera observación se refiere a la ecuación de compatibilidad de los vértices (1). Cuando se aplica a una superficie múltiplemente conexa, esta se transforma en

$$E - C - 2R = 4(1 - H), \tag{7}$$

donde H es el número de agujeros que tiene la geometría. Por consiguiente, para clasificar los vértices, el problema lineal a solucionar es:

$$\begin{cases} \text{mín} \displaystyle\sum_{i=1}^{N_{vertices}} \rho_i D_i + \omega_i d_i, & \\ \text{restringido a:} & \\ \displaystyle\sum_{i=1}^{N_{vertices}} \alpha_i = 4(1 - H), & \\ D_i \geq \alpha_i - \overline{\alpha_i}, & i = 1, \ldots, N_{vertices} \\ d_i \geq \overline{\alpha_i} - \alpha_i, & i = 1, \ldots, N_{vertices} \\ D_i + d_i \leq 1, & i = 1, \ldots, N_{vertices} \\ D_i, d_i \geq 0, & i = 1, \ldots, N_{vertices} \end{cases} \tag{8}$$

Al igual que sucedía con la condición (1), la condición (7) es una condición necesaria pero no suficiente para poder generar una malla en una geometría mediante el método de submapping. Por consiguiente, existen geometrías que cumplen la ecuación (7) y no son discretizables por este método. La Figura 12 muestra una superficie en la que la clasificación de los vértices cumple la ecuación (7) aunque ésta no puede ser discretizada mediante el método de submapping.

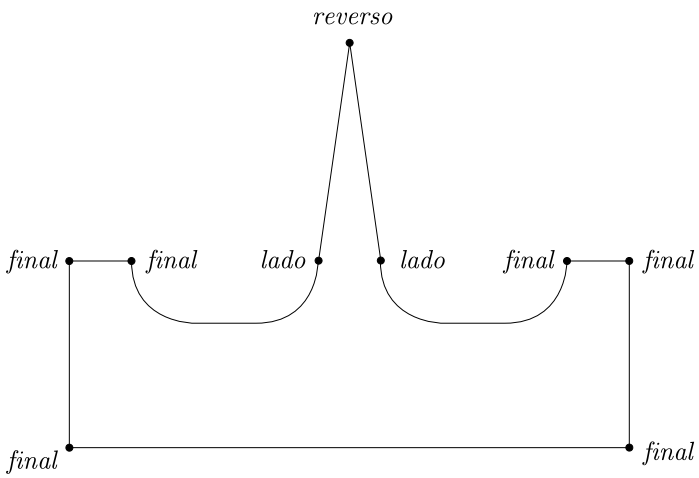


Figura 12. Geometría que cumple la ecuación (7) aunque no puede ser discretizada mediante submapping

La estrategia que se plantea consta de tres fases:

(I) Clasificación de los vértices.

(II) Si la clasificación de los vértices no cumple la ecuación (7), se clasifican los vértices teniendo en cuenta el nuevo problema lineal (8).

(III) Si la geometría es múltiplemente conexa, esta se transforma en simplemente conexa mediante el algoritmo descrito.

(IV) A partir de este punto, el método de submapping procede normalmente.

EJEMPLOS NUMÉRICOS

En esta sección se presentan ocho ejemplos de mallas generadas mediante el método de submapping. El primer ejemplo presenta una geometría que no contiene ningún vértice clasificado como *esquina* ni *reverso*. Por lo tanto, la geometría no se descompone y la malla se genera directamente aplicando el método TFI. La Figura 13 presenta la malla generada en la mitad de una tuerca. En este ejemplo la clasificación obtenida según la sección es correcta y no ha sido necesario resolver el problema (6) ni convertir la geometría a simplemente conexa.

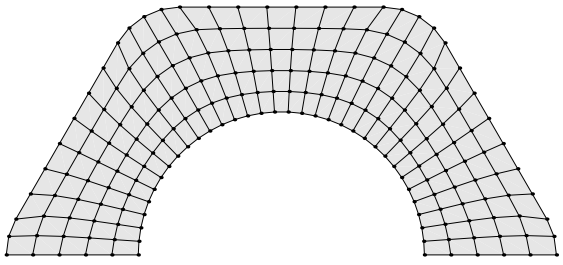
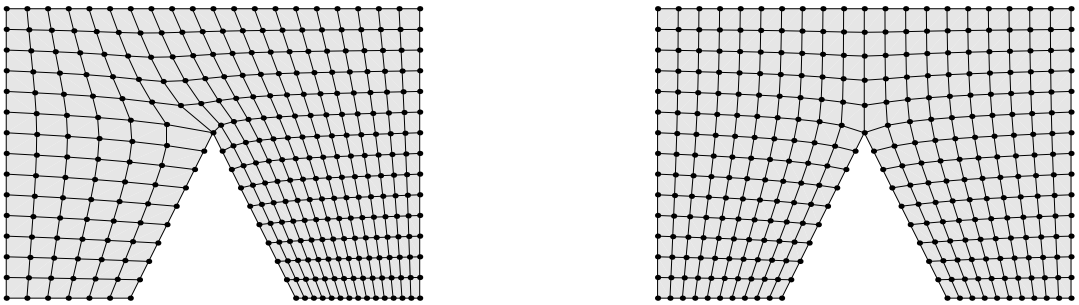


Figura 13. Malla creada en la mitad de una tuerca mediante el método de submapping

El segundo ejemplo presenta una geometría sencilla que contiene un vértice clasificado como *reverso*. El objetivo es ilustrar la importancia del término M en la definición de la función objetivo del problema lineal (3). Con este fin, se ha generado dos discretizaciones de esta geometría utilizando dos funciones objetivo diferentes en el problema lineal (3). Concretamente, éstas son:

- $f_1 = \sum_e \omega_e n_e = \sum_e \frac{n_e}{l(e)}$, donde $l(e)$ es la longitud de la arista e. La malla generada con esta función objetivo se muestra en la Figura 14(a).
- $f_2 = \sum_e \frac{n_e}{l(e)} + M = \sum_e \frac{n_e}{l(e)} + \max_e \left\{ \frac{n_e}{N_e} \right\}$, donde N_e es el número mínimo de elementos de la arista e. La malla generada con esta función objetivo se muestra en la Figura 14(b).



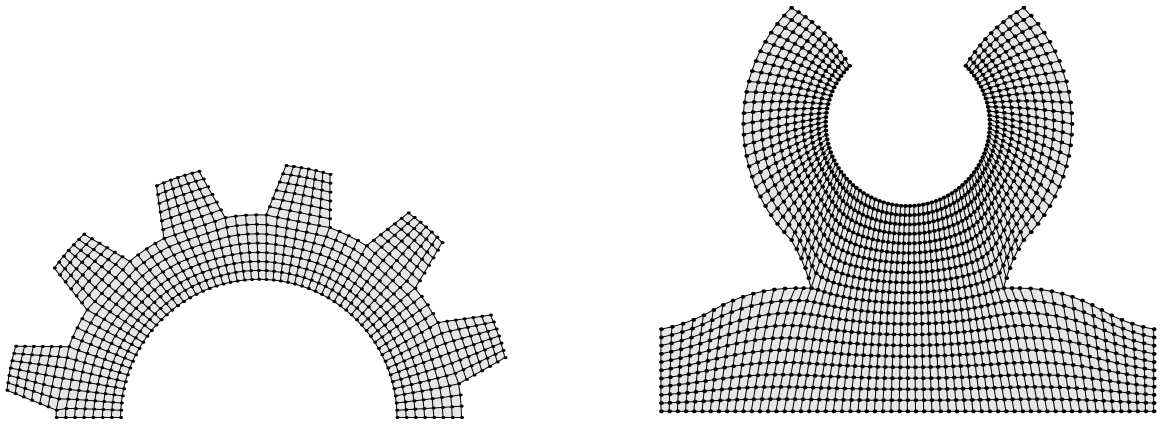
(a) (b)

Figura 14. Diferentes mallas generadas cambiando la función objetivo del problema (3). (a) Malla generada minimizando el número de elementos globalmente. (b) Malla generada minimizando el número de elementos en cada arista

Tal y como se puede comprobar, la malla resultante tiene mejor calidad usando la segunda función objetivo. El efecto de dicha función es el de repartir mejor los elementos en la frontera. En este ejemplo tampoco ha sido necesario recurrir al problema (6) para calcular una clasificación válida de los vértices ni tampoco ha sido necesario convertir la geometría a simplemente conexa.

El tercer ejemplo presenta una pieza mecánica. Este ejemplo muestra la automatización del proceso de descomposición de una geometría mediante el algoritmo de submapping. En este ejemplo la clasificación obtenida inicialmente ha sido correcta. Por lo tanto, no ha sido necesario recurrir al problema (6) para encontrar una clasificación de los vértices válida. La Figura 15(a) presenta la geometría y la malla generada mediante el método de submapping.

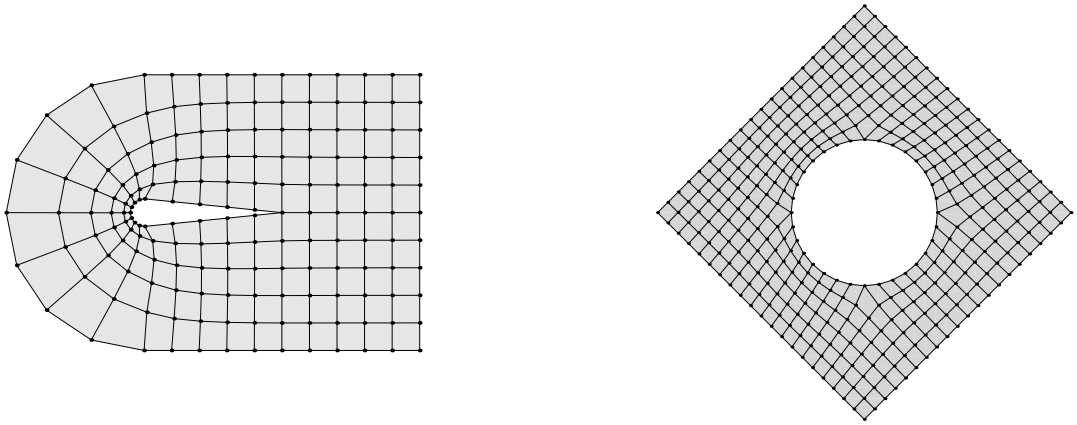
En el cuarto ejemplo se considera la discretización de un soporte mecánico. Este modelo no cumple la condición de compatibilidad de los vértices (1). La resolución del problema (6) conduce a una clasificación de los vértices válida para generar una malla mediante el método de submapping. La Figura 15(b) muestra la geometría del soporte mecánico y la malla generada mediante el método de submapping. Como en los casos anteriores, no ha sido necesaria la conversión de esta geometría en simplemente conexa.



(a) (b)

Figura 15. Mallas generadas en las geometrías del tercer y cuarto ejemplo mediante el método de submapping. (a) Malla generada en una pieza mecánica mediante el método de submapping. (b) Malla generada en un soporte mecánico mediante el método de submapping

El quinto ejemplo consiste en geometría múltiplemente conexa. Por consiguiente, se ha utilizado el método presentado en el tercer apartado para transformar dicha geometría en simplemente conexa. La Figura 16(a) presenta la malla de dicha geometría. Obsérvese que en este caso, se ha recurrido a la ecuación (7) para validar la clasificación de los vértices. Además, en este ejemplo no se podría aplicar el método desarrollado en [1], ya que la geometría definida tan solo por el contorno exterior no admite una malla generada mediante el método de submapping.

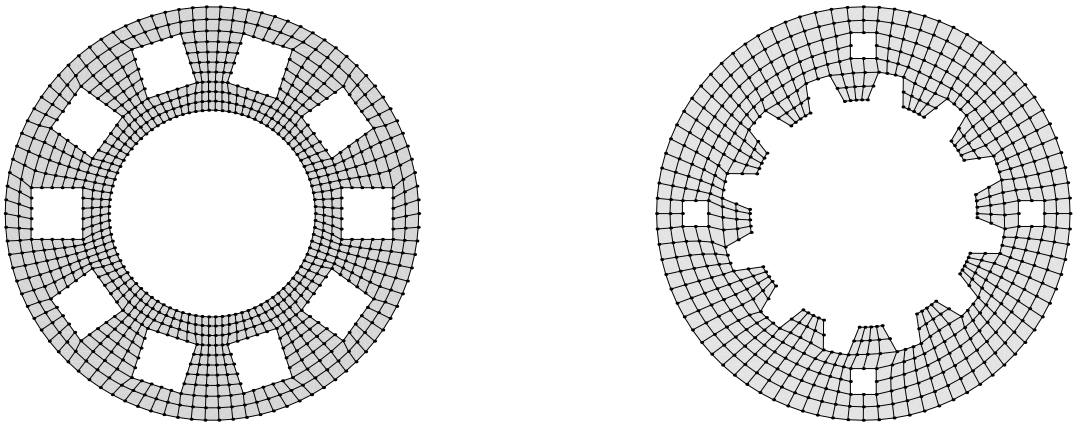


(a) (b)

Figura 16. Mallas generadas en las geometrías del quinto y sexto ejemplo mediante el método de submapping.(a) Malla generada en la geometría del quinto ejemplo mediante el método de submapping. (b) Malla generada en la geometría del sexto ejemplo mediante el método de submapping

En el sexto ejemplo se considera una geometría múltiplemente conexa en la que la clasificación de los vértices no verifica la ecuación (7). Por lo tanto, será necesario recurrir al problema (8) para encontrar una clasificación válida de los vértices. Además, una vez clasificados los vértices, se ha convertido la geometría a simplemente conexa. La Figura 16(b) muestra la malla generada en dicha geometría.

Los dos últimos ejemplos ilustran la aplicación del método desarrollado a la discretización de modelos delimitados por un contorno exterior y diversos contornos interiores. Las Figuras 17(a) y 17(b) presentan las mallas correspondientes a los ejemplos séptimo y octavo. En ambos casos, la clasificación de los nodos ha verificado la ecuación (7) y no ha sido necesario la resolución del problema (8).



(a) (b)

Figura 17. Discretización de dos dominios delimitados por varios contornos interiores. (a) Malla correspondiente al ejemplo siete. (b) Malla correspondiente al ejemplo ocho

CONCLUSIONES

En este trabajo se ha presentado una implementación de la técnica de submapping para generar mallas estructuradas de cuadriláteros. Tal y como se ha descrito anteriormente, el método procede en dos fases

(I) Descomposición de la geometría en piezas fácilmente discretizables.

(II) Discretización de cada pieza manteniendo la compatibilidad de la malla entre las diferentes partes.

Aunque el método de submapping proporciona mallas de buena calidad, no es una técnica general, en el sentido de que no se puede aplicar a la discretización de cualquier geometría. El método tiene dos limitaciones principales. La primera es que los ángulos entre los elementos que definen la geometría tienen que ser, aproximadamente, múltiplos enteros de $\pi/2$. La segunda es que la geometría tiene que ser simplemente conexa. En este trabajo se han detallado dos algoritmos para mitigar estas restricciones.

El primer algoritmo está focalizado en la obtención de una nueva clasificación de los vértices de forma que se amplie el rango de aplicabilidad del método. El nuevo algoritmo se basa en la resolución de otro problema lineal entero, en el que se impone que la clasificación de los vértices cumpla la condición (7). Aunque esta condición es necesaria para generar una malla mediante el método de submapping, no se trata de una condición suficiente. Por este motivo, existen geometrías que cumplen esta ecuación y, sin embargo, no se pueden mallar mediante submapping.

El segundo algoritmo se centra en la conversión de superficies múltiplemente conexas a geometrías simplemente conexas. La solución que se plantea consiste en conectar la frontera exterior con las fronteras interiores mediante aristas virtuales. De este modo, la geometría se convierte en simplemente conexa y el método de submapping puede proceder normalmente. Debe resaltarse que la elección de las líneas de corte se realiza a partir de una triangulación restringida de Delaunay.

Finalmente, es importante comentar que, tanto el método de submapping para superficies como el algoritmo de detección de ángulos mejorado y el algoritmo para discretizar superficies múltiplemente conexas han sido implementados dentro del entorno ez4u[10].

AGRADECIMIENTOS

Este trabajo ha sido financiado parcialmente por el Ministerio de Educación y Ciencia de España mediante los proyectos DPI2007-62395 y BIA2007-66965.

REFERENCIAS

1 D. White, "Automatic Quadrilateral and hexaedral meshing of pseudo-cartesian geometries using virtual subdivision", Department of Civil and Environmental Engineering, Brigham Young University, June (1996).

2 M. Whiteley, D. White, S. Benzley and T. Blacker, "Two and three-quarter dimensional meshing facilitators", *Engineering with computers*, Vol. **12**, pp. 144-154, (1996).

3 J.F. Thompson, B. Soni, N. Weatherill, "*Handbook of Grid Generation*", CRC Press, (1999).

4 E. Ruiz-Gironés, "Generación automática de mallas estructuradas mediante programación lineal e interpolación transfinita", tesina del Máster en Métodos Numéricos para Cálculo y Diseño en Ingeniería, Centre Internacional de Mètodes Numèrics en Enginyeria (CIMNE), (2008).

5 S.A. Mitchell, "High fidelity interval assignment", *International Journal of Computational Geometry and Applications*, Vol. **10**, N° 4, pp. 399-415, (2000).

6 A. Schrijver, "*Theory of Linear and Integer Programming*", John Wiley and Sons, (1998).

7 GNU linear programming kit (GLPK), *http://www.gnu.org/software/glpk/*.

8 P. Chew, "Constrained Delaunay triangulations", *Algorithmica*, Vol. **4**, N° 1, pp. 97–108, (1989).

9 J.R. Shewchuk, "A two-dimensional quality mesh generator and Delaunay triangulator", *http://www.cs.cmu.edu/ quake/triangle.html*.

10 X. Roca, J. Sarrate , "An interactive mesh generation environment for geometry-based simulations", *5th Workshop on Numerical Methods in Applied Science and Engineering (NMASE 06)*, (2006).