UNIVERSIDAD DE BUENOS AIRES
Facultad de Ciencias Exactas y Naturales
Departamento de Matemática

Tesis de Licenciatura

Una heurística para la generación de mallas de elementos finitos anisotrópicos 2D.

Francisco Mastroberti Bersetche

Director: Gabriel Acosta Rodríguez

Fecha de Presentación: 27/12/2013

Índice general

Introducción

Diversos problemas provenientes, en general, de las ciencias aplicadas o la ingeniería, son modelados a través de ecuaciones diferenciales en derivadas parciales. Una de las técnicas más utilizadas para la resolución numérica de estas ecuaciones, en instancias específicas de dichos problemas, es el método de elementos finitos. Esta técnica requiere, sin embargo, de una discretización del dominio Ω (el cual suele considerarse, por simplicidad, como un dominio poligonal) en el cual está planteada la ecuación. La discretización consiste en subdividir Ω en un numero finito de elementos, los cuales serán usualmente triángulos o cuadriláteros si se está en $\mathbb{R}^2$, o tetraedros o hexaedros en el caso de estar $\mathbb{R}^3$. Llamando típicamente τ a la malla (conjunto formado por los elementos), se tiene $\Omega = \cup_{T \in \tau} T$.

Ciertas propiedades asociadas a la malla resultan útiles para mejorar ya sea la calidad de la solución numérica obtenida, como también para reducir el costo computacional manteniendo a su vez un nivel fijo de error.

De estas propiedades, la regularidad de los elementos (es decir que, por ejemplo, en el caso de un triángulo, sus lados tengan aproximadamente el mismo tamaño) es de las más significativas. La teoría clásica de la estimación del error a priori y posteriori utiliza fuertemente esta propiedad.

Muchas veces las características del problema involucran grandes cambios de la solución en algunas direcciones, mientras que en otras se mantiene mayormente constante, como puede ser en los problemas de capas límite. Es deseable en estos casos, que los elementos sean "pequeños" en las direcciones en donde la solución cambia con rapidez, y "grandes" en las direcciones en donde hay poco cambio, para hacer un mejor uso de los recursos computacionales sin que esto incurra en un aumento del error.

Una de las técnicas para poder construir estas mallas en las que se utilizan elementos no regulares está centrada en considerar un tensor métrico $\mathcal{M}$, definido en Ω, adecuado a la propiedad que se busca mejorar, y pedir que los elementos sean regulares bajo esta nueva métrica.

Por lo general, se busca reducir la cantidad de elementos manteniendo un nivel fijo del error, utilizando para este fin tensores métricos basados, en ocaciones, en el hessiano de la solución. Si bien, a priori, el hessiano no es conocido, esto se logra resolviendo el problema con una malla inicial, construida bajo algún criterio, sobre la cual se obtiene una solución aproximada y, a partir de esta solución, se calcula una aproximación del hessiano. Este proceso se itera con la esperanza de que haya convergencia.

Sea cual fuere el motivo de lograr una malla regular en una cierta métrica $\mathcal{M}$, construirla no es un problema trivial. Existen muchos algoritmos a tal fin, con un gran número de enfoques distintos. Por ejemplo, en [1] y [2] se exponen métodos iterativos basados en el suavizado, refinamiento, des-refinamiento y retriangulación de la malla, utilizando, para la retriangulación, una version anisotrópica del criterio de Delaunay, mientras que en [3] se describe un método que utiliza diagramas de Voronoi anisotrópicos como base para la construcción de una triangulación de Delaunay anisotrópica, utilizando a su vez un criterio diferente al usado en [1] y [2]. Por otro

lado, en [4] se describe un método basado en el posicionamiento de los nodos pensados como centros de elipses, las cuales se acomodan en el dominio via fuerzas de atracción y repulsión, para luego realizar la triangulación con una versión anisotrópica del algoritmo de Delaunay.

El objetivo de este trabajo es, por un lado, exponer un algoritmo heurístico para la generación de mallas anisotrópicas siguiendo el enfoque de [1] y [2]. Por otro lado, se buscará implementar una segunda fase de refinamiento (ya sea local o global) utilizando una versión bidimensional del algoritmo de bisección propuesto en [5]. Por último se darán resultados teóricos sobre el funcionamiento del algoritmo de bisección bajo métricas anisotrópicas.

El Capítulo 1 está destinado a fijar notación e ideas generales. En el Capítulo 2 se detallan las cuatro operaciones básicas que utiliza la heurística, mientras que en el Capítulo 3 se explica de qué manera se combinan estas operaciones para dar forma al algoritmo general. En el Capítulo 4 se prueban resultados sobre el buen funcionamiento de la fase de refinamiento. Por último, el Capítulo 5 está dedicado a ejemplos numéricos del funcionamiento de la heurística.

Capítulo 1

Consideraciones generales

De aquí en adelante se considerará a $\Omega \subset \mathbb{R}^2$ como un conjunto abierto, acotado y poligonal, salvo se indique lo contrario.

Sea τ un conjunto de triángulos, diremos que τ es una triangulación de Ω si se cumple que $\overline{\Omega} = \cup_{T \in \tau} T$.

Adicionalmente, por una cuestión de simplicidad, vamos a considerar que los triángulos no son degenerados, es decir que, si $T \in \tau$, T es la cápsula convexa de tres puntos afínmente independientes.

El objetivo del algoritmo es, como ya se mencionó, partiendo de una métrica $\mathcal{M}$ y un conjunto Ω, ambos proveídos por el usuario, construir una triangulación de Ω de manera tal que las longitudes de los lados de un mismo triángulo sean lo más parecidas posible a 1, donde tales longitudes son medidas bajo la métrica $\mathcal{M}$.

No toda triangulación de Ω será deseable a la hora de aplicar el método de elementos finitos (en lo siguiente MEF). Hasta el momento, con lo que se definió, una triangulación de Ω podría contener elementos superpuestos. A continuación se fijará algo de notación y el concepto de malla admisible, el cual es una condición necesaria a la hora de aplicar MEF conforme.

1.1. Mallas admisibles

Sea τ una triangulación de Ω, y sea $T \in \tau$, se notará como $\mathcal{N}_T$ al conjunto de vértices de T, y $\mathcal{E}_T$ al conjunto de lados de T. De manera natural se definen los conjuntos:

$$\mathcal{N}_\tau := \bigcup_{T \in \tau} \mathcal{N}_T \qquad \text{y} \qquad \mathcal{E}_\tau := \bigcup_{T \in \tau} \mathcal{E}_T$$

Adicionalmente, si $E \in \mathcal{E}_\tau$ y $v \in \mathcal{N}_\tau$, se definen los conjuntos $\mathcal{N}_E := \{v \in \mathcal{N}_\tau$ tal que v es un extremo de $E\}$ y $\mathcal{N}_v := \{u \in \mathcal{N}_\tau$ para los cuales $\exists E \in \mathcal{E}_\tau$ tal que u y $v \in \mathcal{N}_E\}$.

Por otro lado, tomando v y E como antes, se definen los conjuntos $\omega_v := \{T \in \tau$, tales que $v \in \mathcal{N}_T\}$ y $\omega_E := \{T \in \tau$, tales que $E \in \mathcal{E}_T\}$.

Por último, y terminando con la introducción de notación, si se tiene u y $v \in \mathbb{R}^2$, se define E_{uv} como el segmento cuyos extremos son u y v.

Como se dijo anteriormente, no en cualquier triangulación será posible aplicar métodos conformes sobre el dominio Ω. Una serie de propiedades debe cumplirse sobre la malla para considerarla "válida". A continuación introducimos la idea de malla o triangulación admisible.

Definición 1.1.1 *Sea τ una triangulación de Ω, se dirá que τ es admisible si, $\forall\, T \in \tau$, $\forall E \in \mathcal{E}_T$, se cumple que o bien $E \subset \partial\Omega$, o existe $T' \in \tau$, $T' \neq T$, tal que $E \in \mathcal{E}_{T'}$.*

Teniendo, entonces, una triangulación admisible de Ω es posible aplicar MEF. Con lo cual es una propiedad fundamental que deberán tener las mallas generadas por el algoritmo.

1.2. Tensores métricos y regularidad

En la sección anterior se definió la propiedad que debe tener una malla para poder aplicar MEF conforme. Aparte de esto, el algoritmo tiene que lograr la propiedad de regularidad bajo cierta métrica, que será en general diferente a la métrica euclídea común.

Sea Γ una curva C^1 a trozos, y $\gamma : [0,1] \to \mathbb{R}^d$, una parametrización de la misma. En la métrica euclídea, la longitud de curva $|\Gamma|$ está definida como:

$$|\Gamma| := \int_0^1 \|\gamma'\| = \int_0^1 \sqrt{\gamma'(t) \cdot \gamma'(t)}dt \tag{2.1}$$

Se considera, entonces, el tensor métrico $\mathcal{M}$ definido como un campo $\mathcal{M} : \mathbb{R}^d \to \mathbb{R}^{d\times d}$, donde $\mathcal{M}(x)$ es simétrica y definida positiva $\forall x \in \mathbb{R}^d$ (en el contexto de este trabajo, se tendrá $d = 2$ ó 3).

Esta métrica nos da una nueva noción de distancia en $\mathbb{R}^d$ la cual, dependiendo de las características de $\mathcal{M}$, podría traer una pérdida de la isotropía del espacio. Teniendo como consecuencia, por ejemplo, que la longitud de un segmento medido en $\mathcal{M}$ pueda variar vía la aplicación de una rotación. Esta característica, que se conoce como anisotropía, es la que hace que elementos regulares en $\mathcal{M}$ sean "más largos" en una dirección que en otra. Típicamente, si fuese $\mathcal{M} \cong cte$, los elementos regulares tenderán a ser más largos en la dirección del autovector de autovalor más pequeño, y más cortos en la dirección perpendicular a éste.

La longitud de curva medida bajo $\mathcal{M}$, notada $|\Gamma|_{\mathcal{M}}$, se calcula como:

$$|\Gamma|_{\mathcal{M}} := \int_0^1 \|\gamma'\|_{\mathcal{M}} = \int_0^1 \sqrt{\gamma'(t) \cdot \mathcal{M}(\gamma(t))\gamma'(t)}dt \tag{2.2}$$

Con esta noción de medida, podemos dar a continuación la definición de regularidad que se manejará en este trabajo:

Definición 1.2.1 *Sea T un triángulo, diremos que T es regular en $\mathcal{M}$ si, $\forall$ E y $E' \in \mathcal{E}_T$ se cumple $|E|_{\mathcal{M}} = |E'|_{\mathcal{M}}$.*

En general, al construir una triangulación sobre un Ω cualquiera, no siempre será posible lograr una malla con todos sus elementos regulares, debido a que la definición anterior es extremadamente restrictiva. Sin embargo será deseable que los elementos sean lo más cercano posible a un elemento regular. A tal efecto, medidas de cuán cerca está un elemento de ser regular serán tratadas tanto en el Capítulo 3 como en el 4. Con lo cual se podrá comparar la regularidad entre dos elementos en función de las mencionadas medidas.

Es importante observar que, al ser la noción de regularidad de la malla independiente de la uniformidad (es decir que todos los elementos sean aproximadamente del mismo tamaño), es posible dar un tratamiento por separado a estos dos aspectos al momento de construirla. Es

decir, si bien el objetivo sería lograr que $\forall E \in \mathcal{E}_\tau$, $|E|_{\mathcal{M}} \approx 1$, a veces la condición de uniformidad puede relajarse en favor de la regularidad, o viceversa. Estos detalles son tratados también en el Capítulo 3.

Otro concepto que puede generalizarse bajo una nueva métrica $\mathcal{M}$ es el de ángulo entre dos vectores. A tal fin se considerará a $\mathcal{M}$ como un tensor métrico constante. O, dicho de otra forma, una matriz s.d.p. . Dados u y v dos vectores de $\mathbb{R}^d$ el ángulo entre ellos $\angle(u, v)$, en el caso de la métrica usual, se calcula de la siguiente forma:

$$\angle(u, v) := \arccos\left(\frac{\langle u, v\rangle}{\langle u, u\rangle^{\frac{1}{2}} \cdot \langle v, v\rangle^{\frac{1}{2}}}\right) \tag{2.3}$$

Donde $\langle u, v\rangle := u^t \cdot v$. Si se define $\langle u, v\rangle_{\mathcal{M}} := u^t \cdot \mathcal{M} \cdot v$, la noción de ángulo se puede generalizar de la siguiente forma:

$$\angle_{\mathcal{M}}(u, v) := \arccos\left(\frac{\langle u, v\rangle_{\mathcal{M}}}{\langle u, u\rangle_{\mathcal{M}}^{\frac{1}{2}} \cdot \langle v, v\rangle_{\mathcal{M}}^{\frac{1}{2}}}\right) \tag{2.4}$$

Capítulo 2

Operaciones básicas

La heurística que se propone en este trabajo consta de cuatro funciones básicas que operan sobre una malla admisible τ. Este capítulo está destinado a la descripción de cada una de ellas, mientras que en el Capítulo 3 puede verse de qué manera y en qué orden se utilizan.

La primera función que se describe es la función `Refinar`, la cual está destinada, via bisección de los elementos, a ser la herramienta que permite el refinamiento de lados demasiado grandes. Por otro lado, su contraparte, la función `ColapsarLado`, se utiliza para hacer desaparecer lados que sean demasiado pequeños. Por último, las funciones de suavizado y retriangulación se utilizan para regularizar la malla cuando sea necesario.

2.1. Función `Refinar`

El algoritmo de bisección que se utiliza es la versión bidimensional del algoritmo propuesto por D. Arnold, A. Mukherjee y L. Pouly en [5], el cual está pensado para ser aplicado sobre mallas admisibles de tetraedros.

En dicho artículo se prueban resultados tanto de la parada del algoritmo como también sobre la calidad de los elementos producidos por el mismo, resultados que, a su vez, son fácilmente generalizables para la versión aquí utilizada. Detalles sobre los mencionados resultados serán tratados más de cerca en el Capítulo 4.

Sea τ una triangulación admisible de Ω. Dado $T \in \tau$, se le asociará una estructura de datos simple con la finalidad de exponer de manera clara el funcionamiento del algoritmo de bisección.

Se define, entonces, partiendo de $T \in \tau$, el concepto de triángulo marcado (análogo 2D al de tetraedro marcado definido en [5]) de la siguiente forma:

Definición 2.1.1 *Sea τ una triangulación y sea $T \in \tau$, se llamará triángulo marcado asociado a T a la estructura de datos definida como la 3-upla $(\mathcal{N}_T, r, g)$, donde $r \in \mathcal{E}_T$, y $g \in \mathbb{N}_0$.*

Es importante aclarar que muchas veces se hará un abuso de notación y se llamará T al triángulo marcado asociado $(\mathcal{N}_T, r, g)$. Ya que toda la información que da $(\mathcal{N}_T, r, g)$ no es más que el triángulo T con un lado destacado r, y un número g. A dicho lado destacado se lo llamará en lo siguiente "lado de refinamiento", y diremos que g es la generación.

Teniendo en cuenta lo anterior, diremos que una malla admisible τ es una malla marcada, si todos sus elementos tienen una estructura de triángulo marcado asociado. O dicho de forma más simple, que a todos los elementos se les eligió un lado de refinamiento y una generación.

A continuación se define la función `BisectarTri`.

ALGORITMO $\{T_1, T_2\} =$ `BisectarTri`(T)

entrada: Un triángulo marcado T.

salida: Dos triángulos marcados T_1 y T_2.

1. Indexar $\mathcal{N}_T = \{v_1, v_2, v_3\}$ de forma tal que $r = \overline{v_1 v_2}$ y definir $v_r := \frac{v_1+v_2}{2}$.

2. Definir el triángulo marcado $T_i = (\mathcal{N}_{T_i}, r_i, g_i)$, $(i = 1, 2)$, de la siguiente forma: $\mathcal{N}_{T_i} := \{v_i, v_3, v_r\}$, $r_i := \overline{v_i v_3}$ y $g_i = g + 1$.

En este punto debemos observar que si se aplica `BisectarTri` sobre, por ejemplo, un triángulo T que cumpla $T \cap \partial\Omega = \emptyset$, reemplazando T por T_1 y T_2 en τ, se tendrá como resultado una malla no admisible. Ya que puede comprobarse fácilmente que para T_i, con $i = 1, 2$, si $E \in \mathcal{E}_{T_i}$, $E \notin \partial\Omega$ y $E \notin \mathcal{E}_{T'}$, cualquiera sea $T' \in \tau$, $T' \neq T_i$.

A continuación se introduce la idea de nodo colgado.

Definición 2.1.2 *Sea τ una triangulación y sea $v \in \mathcal{N}_\tau$. Diremos que v es un nodo colgado si existe $T \in \tau$ tal que se cumple $v \in T$ y $v \notin \mathcal{N}_T$. Adicionalmente diremos que un elemento T posee un nodo colgado si existe $v \in \mathcal{N}_\tau$ tal que $v \in T$ y $v \notin \mathcal{N}_T$.*

El desafío del algoritmo de refinamiento será entonces, bisectar los elementos que indique el usuario y lograr admisibilidad en la malla bisectando, a tal efecto, la menor cantidad de elementos adicionales, a su vez manteniendo una buena calidad de los nuevos elementos producidos.

Definida entonces la función `BisectarTri`, se detalla a continuación la función principal de esta sección.

ALGORITMO $\tau' =$ `Refinar`$(\tau, \mathcal{S})$

entrada: Una malla admisible marcada τ y $\mathcal{S} \subset \tau$.

salida: Una malla admisible marcada τ'.

1.$\overline{\tau} =$ `BisectarTris`$(\tau, \mathcal{S})$

2.$\tau' =$ `RefinarAdmisible`$(\overline{\tau})$

El primer paso del algoritmo resulta trivial, ya que simplemente hay que bisectar cada triángulo contenido en $\mathcal{S}$. Con lo cual la función `BisectarTris` puede definirse de la siguiente forma.

$$\texttt{BisectarTris}(\tau, \mathcal{S}) = (\tau \setminus \mathcal{S}) \bigcup_{T \in \mathcal{S}} \texttt{BisectarTri}(T)$$

El segundo paso del algoritmo está destinado a asegurar la admisibilidad de la salida, bisectando aquellos elementos que tengan nodos colgados, aplicando de forma recursiva. Más precisamente se define la función `RefinarAdmisible` de la siguiente manera:

ALGORITMO $\tau' =$ `RefinarAdmisible`(τ)

entrada: Una malla marcada τ que sea salida de `BisectarTris`.

salida: Una malla admisible marcada τ'.

1. Fijar $\mathcal{S} = \{T \in \tau$ tal que T posee un nodo colgado $\}$

2. Si $\mathcal{S} \neq \emptyset$ hacer

   $\overline{\tau} =$ `BisectarTris`$(\tau, \mathcal{S})$

   $\tau' =$ `RefinarAdmisible`$(\overline{\tau})$

3. Si no

$\tau' = \tau$

Eventualmente, debido a la recursión en `RefinarAdmisible`, podría darse el caso que el algoritmo no termine. Esto sin embargo, tal como se demuestra en [5], no sucederá. Detalles sobre este resultado se tratan más de cerca en el Capítulo 4.

2.2. Función `ColapsarLado`

Ya se ha detallado la herramienta que utiliza la heurística para tratar elementos cuyos lados sean demasiado grandes. Dado que la función `Refinar` muchas veces se verá obligada a bisectar elementos que en principio no se querían bisectar, teniendo como consecuencia la aparición de lados más pequeños de lo deseado, es necesario que la heurística posea la capacidad de "desrefinar" la malla.

Este objetivo se logra mediante la función `ColapsarLado`, a la cual se le indica el lado sobre el cual se quiere aplicar la operación, haciéndolo colapsar sobre su punto medio o sobre alguno de los extremos, siempre que sea posible.

Con el propósito de no deformar el borde de la triangulación original cuando la función se esté usando de forma iterativa dentro de la fase I, se utilizará el conjunto $\mathcal{N} \subseteq \mathcal{N}_\tau \cap \partial\Omega$, el cual contendrá los puntos originales del borde de la triangulación ingresada por el usuario. El propósito puede que quede más claro en la descripción de la fase I en el Capítulo 3. Por el momento basta ver a $\mathcal{N}$ como un conjunto de nodos destacados cuya posición no puede ser alterada.

A continuación se describe la función `ColapsarLado`:

ALGORITMO $\tau' =$ `ColapsarLado`$(\tau, E, \mathcal{N})$
entrada: Una malla admisible τ, un lado $E \in \mathcal{E}_\tau$, y el conjunto $\mathcal{N} \subseteq \mathcal{N}_\tau \cap \partial\Omega$.
salida: Una malla admisible τ'.
1. Fijar $\mathcal{N}_E = \{v \in \mathcal{N}_\tau \text{ tal que } v \in E\} = \{v_1, v_2\}$; $\omega_E = \{T \in \tau \text{ tal que } E \in \mathcal{E}_T\}$.
2. Definir $\overline{\tau} = \tau$.
3. Si $E \subset \partial\Omega$
   3.1. Si $v_1 \notin \mathcal{N}$ y $v_2 \notin \mathcal{N}$
      Hacer $\overline{\tau} = \overline{\tau} \setminus \omega_E$.
      Hacer para $i = 1, 2$, si $v_i \in \mathcal{N}_{\overline{\tau}}$, $v_i = \frac{v_1 + v_2}{2}$.
   3.2. Si $v_1 \in \mathcal{N}$ y $v_2 \notin \mathcal{N}$ o $v_1 \notin \mathcal{N}$ y $v_2 \in \mathcal{N}$
      Hacer $\overline{\tau} = \overline{\tau} \setminus \omega_E$.
      Hacer para $i = 1, 2$, si $v_i \in \mathcal{N}_{\overline{\tau}}$, $v_i = v^*$, donde $v^* \in \{v_1, v_2\} \cap \mathcal{N}$.
4. Si $E \not\subset \partial\Omega$
   4.1. Si $v_1 \notin \partial\Omega$ y $v_2 \notin \partial\Omega$
      Hacer $\overline{\tau} = \overline{\tau} \setminus \omega_E$.
      Hacer para $i = 1, 2$, si $v_i \in \mathcal{N}_{\overline{\tau}}$, $v_i = \frac{v_1 + v_2}{2}$.
   4.2. Si $v_1 \in \partial\Omega$ y $v_2 \notin \partial\Omega$ o $v_1 \notin \partial\Omega$ y $v_2 \in \partial\Omega$
      Hacer $\overline{\tau} = \overline{\tau} \setminus \omega_E$.
      Hacer para $i = 1, 2$, si $v_i \in \mathcal{N}_{\overline{\tau}}$, $v_i = v^*$, donde $v^* \in \{v_1, v_2\} \cap \partial\Omega$.
5. Si $\overline{\tau}$ es admisible y $\cup_{T \in \tau} = \Omega$, hacer
   $\tau' = \overline{\tau}$.
6. Si no

$\tau' = \tau$.

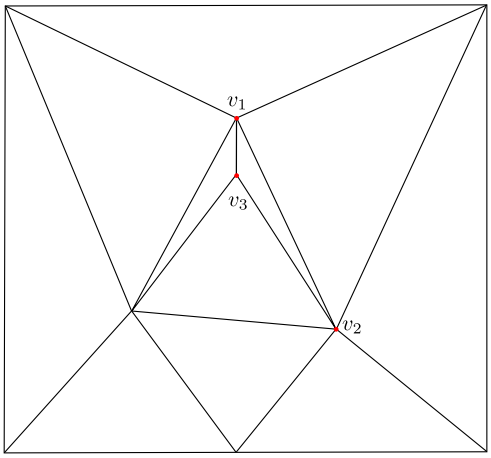


Figura 1: *En este ejemplo puede verse que si se colapsa el lado $\overline{v_1 v_2}$ sobre su punto medio la malla pierde admisibilidad, mientras que si se hace lo mismo con $\overline{v_1 v_3}$ no se presenta ningún problema.*

Dicho de forma más simple, el usuario indica a la función el lado E que quiere hacer desaparecer, si el lado es interior y sus extremos no tocan el borde, ésta reemplaza los vértices de los extremos de E por el punto medio del mismo y elimina a aquellos elementos que tenían a E como uno de sus lados. En caso de que E sea interior y sólo uno de sus extremos toque el borde, la función hace lo mismo que antes, con la salvedad que esta vez reemplaza los vértices de los extremos por el vértice que está en el borde. Si ambos vértices están en el borde y E es interior, la función no hace nada, ya que de otra manera deformaría la geometría de borde.

En el caso en que E está contenido en el borde, la función se fija si los extremos están en $\mathcal{N}$. De forma similar al caso anterior, si ninguno de los nodos está en $\mathcal{N}$ entonces colapsa el lado sobre el punto medio. Si sólo uno de ellos está en $\mathcal{N}$, colapsa sobre ese nodo. Si ambos están en $\mathcal{N}$, la función no hace nada.

En todos los casos comentados más arriba, antes de devolver la malla modificada, si es que la función la modificó, ésta se fija si la malla es admisible, de ser así da esa malla como salida, en caso contrario da como salida la misma malla que tomó como entrada.

Como se dijo antes, esta operación, combinada con la función `Refinar`, permite controlar el tamaño de los elementos, bisectando los lados que sean demasiado grandes o colapsando aquellos que sean demasiado pequeños. Sin embargo, a la hora de lograr elementos de buena calidad en la malla, estas dos operaciones no son suficientes.

En las siguientes dos secciónes se detallan las operaciones utilizadas para lograr un control más fino de los tamaños de los lados.

2.3. Suavizado

Los algoritmos de suavizado consisten en, bajo algún criterio adecuado, reposicionar un nodo de la malla en función de sus vecinos con el objetivo de mejorar la calidad de los elementos que contienen a dicho nodo. Por lo general se aplica de forma iterativa, ya sea reposicionando un

nodo a la vez o todos al mismo tiempo, consiguiendo, en la mayoría de los casos, mejorar la calidad de la malla en cada iteración.

En las mallas triangulares, se puede clasificar estos métodos, a modo ilustrativo, en tres grandes enfoques.

Por un lado, están los métodos basados en promedios, que, como su nombre lo indica, consisten en mover un nodo hacia algún promedio ponderado de sus vecinos. Más precisamente, la expresión general de estos métodos podría escribirse de la siguiente manera, siendo $v \in \mathcal{N}_\tau$ el nodo que se busca reposicionar, se calcula su nueva posición de la siguiente manera:

$$v = \sum_{u \in \mathcal{N}_v} w_u \cdot u$$

Donde la igualdad denota asignación, y los pesos w_u verifican $\sum_{u \in \mathcal{N}_v} w_u = 1$. Dentro de estos métodos se puede incluir el suavizado laplaciano, el más simple de ellos. Este suavizado consiste en reposicionar un punto en el promedio de sus vecinos. Es decir que opera sobre el nodo v de la siguiente forma:

$$v = \frac{1}{\#\mathcal{N}_v} \cdot \sum_{u \in \mathcal{N}_v} u$$

Hay una gran variedad estos métodos así como modificaciones basadas en ellos, a su vez aplicados a una gran variedad de objetivos. Por ejemplo en [6] se propone un método para deformar mallas manteniendo la calidad de los elementos. Dicho método consiste en calcular los pesos como la distancia relativa de cada nodo a sus vecinos vía la resolución de un problema de optimización no lineal, se continua aplicando la transformación a los nodos del borde para después reposicionar los interiores utilizando los pesos calculados previamente. Otro ejemplo puede verse en [7], donde se utiliza una version modificada del suavizado laplaciano con el objetivo de suavizar superficies ruidosas deformándolas lo menos posible en el proceso. El suavizado propuesto calcula la nueva posición del nodo como una combinación convexa de su posición original y el promedio de sus vecinos.

Por otro lado están los métodos basados en optimizar medidas de la calidad de los elementos. Es decir, partiendo de cierta función que mide la calidad de un elemento, midiendo con ella la calidad de los elementos que tienen a v como uno de sus vertices, se considera el problema de optimización que consiste en maximizar la calidad en función de la posición de v. A tal fin pueden utilizarse cualquiera de las muchas técnicas de optimización lineal o no lineal que sean más adecuadas en cada caso. Un ejemplo de esta idea se expone en [8], donde se describe un método basado en la minimización de una medida de distorsión de los elementos sobre los cuales incide el vértice que se busca reposicionar. El reposicionamiento a su vez se lleva a cabo utilizando un método de descenso valiéndose del gradiente de la función que mide la calidad.

Por último están los métodos basados en principios físicos. Por lo general se simulan fuerzas de atracción y/o repulsión entre los nodos. Dichas fuerzas suelen calcularse en función de las aristas que inciden sobre el nodo que se quiere reposicionar. Un ejemplo de esto puede encontrarse en [9], donde la malla es pensada como un sistemas de resortes y el reposicionamiento consiste en buscar la configuración de menor energía potencial. Otra de estas ideas se utiliza en [10], donde se busca reposicionar el nodo de forma tal que los ángulos de los elementos que inciden en él se mantengan lejos de 0 y π, intentando que dos ángulos adyacentes sean lo más parecidos posible. A tal efecto se utiliza un enfoque basado en la simulación de un sistema de resortes de

torsión, el cual genera una fuerza repulsiva entre dos segmentos que compartan un vértice cuyo ángulo esté cercano a cero, y atractiva en el caso que el ángulo esté serca de π. Luego, al igual que en el anterior caso, se busca la configuración de equilibrio.

Si bien tal vez no se abarquen todos los métodos basados en principios físicos , la siguiente expresión describe de forma genérica gran parte de ellos. Siendo el nodo $v \in \mathcal{N}_\tau$ el que se busca reposicionar, se tiene:

$$v = v + C \sum_{u \in \mathcal{N}_v} f(|E_{vu}|_{\mathcal{M}}) \cdot \frac{(v-u)}{|E_{vu}|_{\mathcal{M}}} \tag{3.1}$$

Donde al igual que antes, la igualdad denota asignación. Aquí E_{vu} es el lado que tiene como extremos a v y u, y C es una constante de escala que suele ajustarse empíricamente. La función f es la que simula el efecto de una fuerza. Cuando f es negativa la fuerza es atractiva mientras que en el caso contrario es repulsiva.

Como se dijo antes, esta clasificación es de carácter ilustrativo, hay métodos que combinan varios de estos enfoques a la ves, o incluso, métodos que puede reformularse de forma tal que pueden ser incluidos en dos clasificaciones distintas. Tal es el caso del suavizado laplaciano, que, buscando una función f adecuada, se lo puede ver como un método basado en principios físicos, donde las fuerzas que se simulan son siempre atractivas.

Muchos métodos de suavizado fueron probados para la heurística aquí presentada, la mayoría de ellos con resultados aceptables. Sin embargo, con el objetivo de favorecer, de alguna manera, el alineamiento de los lados con los autovalores de la métrica, se decidió implementar una modificación en la formulación genérica de los métodos basados en principios físicos.

La idea de favorecer dicha alineación surge de ver los patrones de las mallas regulares en métricas con anisotropía pronunciada, en donde por lo general, los elementos se acomodan de forma tal que su lado más corto se alínea con el autovector de autovalor más grande de $\mathcal{M}$, y los otros dos con otro autovector.

La modificación consiste en construir de forma apropiada pesos w_u asociados a los nodos $u \in \mathcal{N}_v$, utilizándolos de la siguiente manera:

$$v = v + C \sum_{u \in \mathcal{N}_v} f(|E_{vu}|_{\mathcal{M}}) \cdot \frac{(v-u)}{|E_{vu}|_{\mathcal{M}}} \cdot w_u \tag{3.2}$$

Para aclarar la forma en que se construyen los pesos w_u, primero introducimos la siguiente función auxiliar:

ALGORITMO $\theta = \texttt{Ang}(v, u)$
entrada: Dos nodos v y $u \in \mathcal{N}_\tau$.
salida: Un número real θ.
1. Fijar e indexar $\mathcal{S} := \{T \in \tau$ tal que $v \in \mathcal{N}_T$ y $u \in \mathcal{N}_T\}$.
2. Sea $T_i \in \mathcal{S}$, $1 \leq i \leq \#\mathcal{S}$, y sea $\mathcal{N}_{T_i} := \{v, u, v_i\}$, definir $\theta_i := \angle((u-v), (v_i - v))$.
3. Hacer $\theta = \sum_{i=1}^{\#\mathcal{S}} \theta_i$.
4. Si $E_{vu} \subset \partial\Omega$, hacer
   4.1. $\theta = \theta + \pi$.

A partir de lo anterior, se pueden definir los pesos de la siguiente forma:

$$w_u := \frac{\texttt{Ang}(v,u)}{4\pi}$$

En la práctica se observó que si bien el resultado final en la utilización del suavizado expuesto en (3.2) no difiere demasiado al de (3.1) (siempre que f simule sólo fuerzas repulsivas), utilizando el primero se logra una convergencia algo más rápida y estable en los casos donde se utilizan métricas con anisotropía pronunciada. Sin embargo ambos pueden utilizarse dando resultados muy similares. Para los casos donde f simula tanto fuerzas repulsivas como atractivas esta modificación genera mallas de mejor calidad.

Antes de exponer las funciones principales de esta sección, con el objetivo de simplificar el pseudo-código se introducirá una función auxiliar más. El objetivo de esta función será reconocer los casos en donde el suavizado empuja puntos hacia posiciones no deseadas, haciendo que se pierda la admisibilidad de la malla.

ALGORITMO $x = \texttt{NoDegenera}(v, v^*, \mathcal{R})$
entrada: Un nodo $v \in \mathcal{N}_\tau$, un punto $v^* \in \mathbb{R}^2$ y un conjunto $\mathcal{R} = \emptyset$.
salida: Un valor booleano x.
1. Fijar e indexar el conjunto $\omega_v \subseteq \tau$.
2. Fijar las variables booleanas x_i con $1 \leq i \leq \#w_v$.
3. Para $1 \leq i \leq \#w_v$, hacer:
3.1. Fijar $\mathcal{N}_{T_i} = \{v, u_i, w_i\}$.
3.2. Si v y v^* están en el mismo semi-espacio delimitado por la recta que para por u_i y v_i, hacer $x_i = \textit{verdadero}$, de lo contrario hacer $x_i = \textit{falso}$.
4. Para $1 \leq i \leq \#w_v$, siendo $\mathcal{N}_{T_i} = \{v, u_i, w_i\}$ hacer:
4.1. Si $x_i = \textit{falso}$ y $E_{u_i w_i} \subset \partial\Omega$, hacer $\mathcal{R} = \mathcal{R} \cup T_i$.
5. Hacer $x = x_1 \wedge ... \wedge x_{\#\omega_v}$.

Para dar una idea más clara, en la función anterior el nodo v juega el papel del nodo que se busca reposicionar, y v^* es la posición hacia la cual se lo busca llevar. Si v^* se sale fuera del conjunto $\cup_{T \in \omega_v} T$, entonces la función arroja como resultado el valor *falso* y la operación de suavizado (como se verá más adelante) no se llevará a cabo.

Adicionalmente, la función pide como entrada un conjunto vacío notado $\mathcal{R}$. A lo largo de la ejecución se guarda en dicho conjunto los elementos que contienen al nodo v y tocan el borde, por los cuales a su vez el nodo v intenta "salir".

La razón de identificar estos elementos se debe a que empíricamente se observó que elementos irregulares tienden a acumularse en los bordes (por lo general en los casos donde se utilizan métricas con una fuerte anisotropía) dado que el suavizado intenta forzar a algún nodo perteneciente a éstos fuera de la triangulación y, como se verá más adelante, la operación se cancela. Estos elementos recibirán un tratamiento especial cuyos detalles se exponen en el capítulo siguiente, en la descripción de la función `LimpiarBorde`.

A continuación se definen las funciones `SuavizarInterior` y `SuavizarBorde`, las cuales son las funciones principales de esta sección. También se define la función `Suavizar` con el propósito de usar de forma más cómoda las dos primeras. Dada una elección de f y C se define:

ALGORITMO $v^* = \texttt{SuavizarInterior}(v)$
entrada: Un nodo $v \in \mathcal{N}_\tau$, tal que $v \notin \partial\Omega$.

salida: Un punto $v^* \in \mathbb{R}^2$.

1. $\forall u \in \mathcal{N}_v$ definir $w_u = \frac{\texttt{Ang}(v,u)}{4\pi}$.

2.Hacer $v^* = v + C \sum_{u \in \mathcal{N}_v} f(|E_{vu}|_{\mathcal{M}}) \cdot \frac{(v-u)}{|E_{vu}|_{\mathcal{M}}} \cdot w_u$.

ALGORITMO $v^* = \texttt{SuavizarBorde}(v)$

entrada: Un nodo $v \in \mathcal{N}_\tau$, tal que $v \in \partial\Omega$.

salida: Un punto $v^* \in \mathbb{R}^2$.

1. Fijar $\{u_1, u_2\} := \{u \in \mathcal{N}_\tau$ tales que $E_{vu} \in \mathcal{E}_\tau$ y $E_{vu} \subset \partial\Omega\}$.

2. Si u_1, u_2 y v son colineales, notando u_p a la proyección ortogonal de u sobre la recta que pasa por u_1 y u_2, hacer:

2.1. Para $u \in \mathcal{N}_v$ hacer:

Si u_p está en el segmento $\overline{u_1 u_2}$ hacer $w_u := \texttt{Ang}(v,u)$, de lo contrario, hacer $w_u := 0$.

2.2 Hacer $w_{tot} = \sum_{u \in \mathcal{N}_v} w_u$ y redefinir $w_u = \frac{w_u}{w_{tot}}$.

2.3. Hacer $\overline{v} = v + C \sum_{u \in \mathcal{N}_v} f(|E_{vu}|_{\mathcal{M}}) \cdot \frac{(v-u_p)}{|\overline{vu_p}|_{\mathcal{M}}} \cdot w_u$.

2.4. Si $\overline{v}$ está en el segmento $\overline{u_1 u_2}$, hacer $v^* = \overline{v}$.

3. Si no, hacer

3.1. $v^* = v$.

ALGORITMO $\tau' = \texttt{Suavizar}(\tau, iter_s, \mathcal{R})$

entrada: Una triangulación admisible τ de Ω y un número natural $iter_s$.

salida: Una triangulación admisible τ' de Ω y un conjunto $\mathcal{R} = \emptyset$.

1. Hacer $\tau' = \tau$.

2. Hacer $iter_s$ veces

2.1. $\mathcal{R} = \emptyset$.

2.2. Para $v \in \mathcal{N}_{\tau'}$ hacer:

2.2.1. Si $v \notin \partial\Omega$

$v^* = \texttt{SuavizarInterior}(v)$.

Si $\texttt{NoDegenera}(v, v^*, \mathcal{R})$, hacer $v = v^*$.

2.2.2. Si no

$v = \texttt{SuavizarBorde}(v)$.

La primera función trabaja sobre puntos interiores de la triangulación, operando de forma ya descrita.

La segunda función opera sobre ciertos puntos que están en el borde, a los cuales se los puede mover a lo largo de una línea recta sin deformar la geometría de borde original. Al implementar una primera versión de este suavizado (en donde en la instrucción de la línea 2.1 a ningún peso w_u se le asigna valor nulo), se observó que cuando los segmentos del borde se alineaban, en algún sentido, con los autovectores de la métrica, se obtenían mallas de muy buena calidad (por ejemplo, ver figura 18 en el Capítulo 5), sin embargo cuando los bordes no se encontraban alineados se vieron casos en donde el algoritmo ni siquiera convergía, o lo hacía muy lento (por ejemplo en el caso de la figura 15 en el Capítulo 5). Esto se debió a que en esos casos, los nodos interiores que aplican "fuerza" sobre el nodo de borde que se buscaba reposicionar parecían generar puntos de equilibrio que eran o bien inestables o bien se encontraban en posiciones en las cuales, de ir hacia allí el nodo que se reposiciona, la malla sería no admisible.

Para solucionar esto, es decir que el algoritmo aproveche los bordes "buenos" y a la vez funcione de forma decente cuando el borde no es bueno, para el cómputo de la nueva posición

sólo se tomaron en cuenta aquellos nodos vecinos de v cuya proyección a la recta definida en la línea 2 de la función no dista demasiado de v.

La tercera función es, como se dijo más arriba, para utilizar las dos primeras de forma más cómoda, sobre todo al momento de presentar el pseudo-código de la fase I, en el Capítulo 3.

2.4. Retriangulación

Utilizando sólo las tres operaciones antes descritas, la fase I de la heurística es capaz de lograr mallas de calidad aceptable para niveles de refinamiento de medianos a altos. Dado que resulta de interés poder genera mallas de calidad y a la vez "gruesas" para aplicar la fase de refinamiento, fue necesario introducir una operación más. La utilización de esta operación produjo también un aumento en la velocidad de convergencia observada.

La operación que se eligió (basada en técnicas comunes utilizadas en este tipo de algoritmos), está destinada a, de alguna manera, hacer que se cumpla una versión anisotrópica se la condición de Delaunay.

Una triangulación se dice que es de Delaunay si se cumple que la circunferencia circunscrita de cada triángulo de la malla no contiene ningún vértice de otro triángulo. La condición es generalizable a más dimenciones pero en 2D resulta equivalente a pedir que dados dos triángulos que compartan un lado en común, la suma de sus ángulos opuestos sea menor a π. Para asegurar el cumplimiento de esta condición es posible, dados dos triángulos con un lado en común que no la cumplan, aplicar un intercambio de lados (ver función `Flip` más abajo). Esta operación se puede aplicar de forma iterativa sobre todos los pares de triángulos que no cumplan la condición hasta obtener una triangulación de Delaunay.

No hay una única manera de extender la triangulación de Delaunay a una métrica anisotrópica. A pesar de existir criterios más sofisticados que el utilizado en este trabajo, dado que el algoritmo puede generar mallas de calidad aceptable sin necesidad de utilizar esta técnica, se optó por utilizar una retriangulación similar a la utilizada en [1].

Aquí se la utiliza de forma totalmente heurística y la menor cantidad de veces posible, más enfocado a arreglar pequeños errores que a ser algo central. En la literatura pueden encontrarse muchas formas de hacer ésto, por lo general variando en la forma en la que se calcula la métrica que se utiliza para medir ángulos y decidir si aplicar o no el intercambio de lados (ver función `Flip` más adelante). Por ejemplo en [1] se computan los ángulos con el promedio de las métricas evaluadas sobre los nodos de los triángulos involucrados, mientras que en [4] prefieren utilizar la métrica evaluada en el baricentro del cuadrilátero formado por los triángulos involucrados.

La forma de calcular la métrica sobre la cual se miden los ángulos que se eligió en este trabajo, por simplicidad, es la misma que se utiliza en [1], aunque la aplicación en el algoritmo global es diferente.

Para definir de forma precisa la función principal de esta sección, y dejar bien en claro el criterio utilizado, primero se define la función auxiliar `Flip`.

ALGORITMO $\{T_1', T_2'\} =$ `Flip`(T_1, T_2)
entrada: Dos triángulos T_1 y $T_2 \in \tau$, tales que $\mathcal{E}_{T_1} \cap \mathcal{E}_{T_2} \neq \emptyset$.
salida: Dos triángulos T_1' y T_2'.
1. Definir E como el único lado que verifica $E \in \mathcal{E}_{T_1} \cap \mathcal{E}_{T_2}$.
2. Fijar $\mathcal{N}_E = \{u_1, u_2\}$ y $\mathcal{N}_{T_1} \cup \mathcal{N}_{T_2} \setminus \mathcal{N}_E = \{u_3, u_4\}$.

3. Para $i = 1, 2$ definir T_i como el triángulo formado por los puntos u_i, u_3 y u_4.

Adicionalmente se define la función auxiliar `CalcularAngTest` de la siguiente forma:

ALGORITMO $\alpha =$ `CalcularAngTest`(T_1, T_2)
entrada: Dos triángulos T_1 y $T_2 \in \tau$, tales que $\mathcal{E}_{T_1} \cap \mathcal{E}_{T_2} \neq \emptyset$.
salida: Un numero real α.
1. Definir E como el único lado que verifica $E \in \mathcal{E}_{T_1} \cap \mathcal{E}_{T_2}$.
2. Fijar $\mathcal{N}_{T_1} \cup \mathcal{N}_{T_2} \setminus \mathcal{N}_E = \{v_1, v_2\}$ y $\mathcal{N}_E = \{v_3, v_4\}$.
3. Definir $V_1 = (v_3 - v_1)$, $W_1 = (v_4 - v_1)$, $V_2 = (v_3 - v_2)$ y $W_2 = (v_4 - v_2)$.
4. Calcular $\mathcal{M}_p = \sum_i \frac{\mathcal{M}(v_i)}{4}$.
5. Para i= 1,2 , definir $\alpha_i = \angle_{\mathcal{M}_p}(V_i, W_i)$.
6. Hacer $\alpha = \alpha_1 + \alpha_2$.

Definido esto, se describe a continuación la función principal de esta sección.

ALGORITMO $\tau' =$ `Retriangular`(τ)
entrada: Una malla admisible τ .
salida: Una malla admisible τ'.
1. Hacer $\tau' = \tau$
2. Definir e indexar $\mathcal{E} = \mathcal{E}_{\tau'}$.
3. Siendo $\mathcal{E} = \cup_{i=1}^{\#\mathcal{E}_{\tau'}} E_i$, hacer , para $i = 1, ..., \#\mathcal{E}\tau'$
   3.1. Si $E_i \not\subset \partial\Omega$, siendo $\omega_{E_i} = \{T_1, T_2\}$, y `CalcularAngTest`$(T_1, T_2) > \pi$
   Fijar $\mathcal{N}_{T_1} \cup \mathcal{N}_{T_2} \setminus \mathcal{N}_E = \{v_1, v_2\}$ y $\mathcal{N}_E = \{v_3, v_4\}$.
   Hacer $\tau' = (\tau' \setminus \{T_1, T_2\}) \cup$ `Flip`(T_1, T_2) .
   Actualizar $\mathcal{E}$ haciendo $E_i = \overline{v_1 v_2}$.

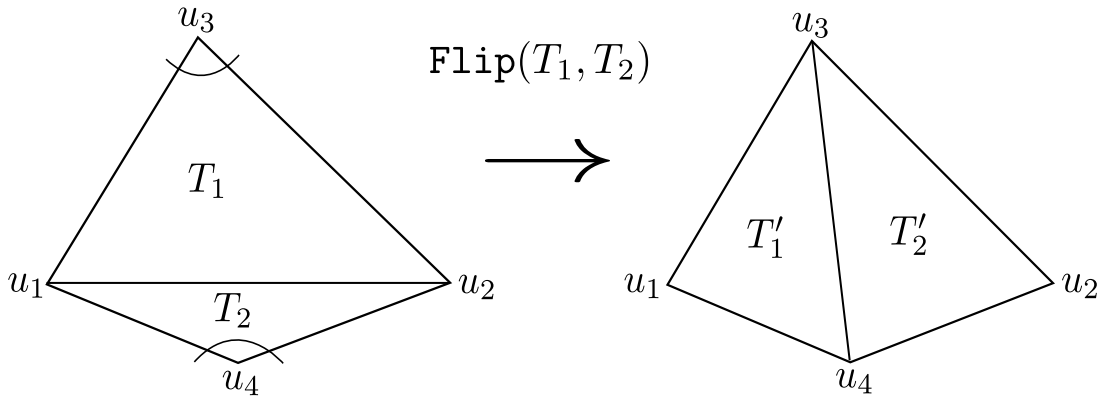
Visto de forma más simple, la función `Retriangular` recorre los lados de la triangulación τ, en caso de que sean lados interiores, se fija si la suma los ángulos opuestos al lado en cuestión (ángulos medidos en $\mathcal{M}$) da mayor a π, en caso afirmativo aplica la función `Flip`, de lo contrario no se hace nada.

Adicionalmente se define la función `Retriangular`$(\tau, iter_r)$, la cual simplemente aplica $iter_r$ veces la función definida anteriormente.

Si bien el pseudo-código está para exponer la idea del algoritmo, hay detalles importantes de la implementación que no están expuestos. Este es el caso del test que se realiza para saber si aplicar `Flip` o no. Dado que esta operación se realiza una cantidad considerable de veces, es necesario optimizarla lo mejor posible. A tal efecto, en [11] se exponen una serie de criterios para intercambiar lados, todos ellos equivalentes, analizando la cantidad de operaciones necesaria en cada uno de ellos y dando la versión anisotrópica para el más eficiente. En base a esto se tiene, siendo T_1 y T_2 tales que $\mathcal{N}_{T_1} = \{u_1, u_2, u_3\}$ y $\mathcal{N}_{T_2} = \{u_1, u_2, u_4\}$:

$$\texttt{CalcularAngTest}(T_1, T_2) > \pi \iff$$

$$[(u_2 - u_3) \times (u_1 - u_3)] \cdot (u_1 - u_4)^t \mathcal{M}_p (u_2 - u_4) + (u_2 - u_3)^t \mathcal{M}_p (u_1 - u_3) \cdot [(u_1 - u_4) \times (u_2 - u_4)] < 0$$


u_3
T_1
u_1
u_2
T_2
u_4
$\texttt{Flip}(T_1, T_2)$
u_3
T_1'
T_2'
u_1
u_2
u_4

Capítulo 3

Descripción de la heurística

Habiendo ya definido las funciones principales, se expondrá en este capítulo la forma de utilizarlas para lograr generar una malla lo más regular posible en $\mathcal{M}$.

3.1. Funciones auxiliares

Dado que en general no será posible lograr que todo los elementos de la malla sean regulares en el sentido estricto de la Definición (1.2.1), es necesario disponer de una medida de regularidad a efecto de maximizarla.

La manera de medir regularidad que se eligió para este algoritmo tiene como objetivo dar un tratamiento diferenciado a dos "tipos" de deformidad. Dichos tipos de deformidad quedan en claro con la definición de las siguientes funciones:

ALGORITMO $x =$ `EsDeformeDef`(T)
entrada: Un triángulo T.
salida: Un valor booleano x.
1. Fijar $\mathcal{E}_T = \{E_1, E_2, E_3\}$.
2. Definir $e_t = \sum_i |E_i|_{\mathcal{M}}$, y para i=1,2,3 $e_i = \frac{|E_i|_{\mathcal{M}}}{e_t}$.
3. Hacer $x = (e_1 < p_{min}) \vee (e_2 < p_{min}) \vee (e_3 < p_{min})$.

ALGORITMO $x =$ `EsDeformeExc`(T)
entrada: Un triángulo T.
salida: Un valor booleano x.
1. Fijar $\mathcal{E}_T = \{E_1, E_2, E_3\}$.
2. Definir $e_t = \sum_i |E_i|_{\mathcal{M}}$, y para i=1,2,3 $e_i = \frac{|E_i|_{\mathcal{M}}}{e_t}$.
3. Hacer $x = [(e_1 > p_{max}) \vee (e_2 > p_{max}) \vee (e_3 > p_{max})] \wedge \neg$`EsDeformeDef`$(T)$.

De esta forma, la función `EsDeformeDef` mide un tipo de deformidad que en lo siguiente se referirá como "deformidad por defecto". La misma tiene lugar cuando alguno de los lados es relativamente más chico que el resto. Para ver si esto sucede, se mide la proporción de perímetro total de T que posee el lado en cuestión. Si ésta es más pequeña que un mínimo predeterminado p_{min}, se considerá al elemento T deforme por defecto, y su lado más pequeño será candidato a ser colapsado.

Por otro lado, y de forma complementaria, la función `EsDeformeExc` mide un tipo de deformidad que se la llamará "deformidad por exceso". Ocurriendo esto cuando alguno de los lados es proporcionalmente más grande que los demás y ninguno de ellos es proporcionalmente más pequeño de lo permitido. Para medir lo anterior, al igual que en `EsDeformeDef`, se mide la proporción de perímetro que posee el lado en cuestión, y si este valor es mayor a un máximo predeterminado p_{max}, se considera al elemento deforme por exceso, y su lado más grande será candidato a ser bisectado.

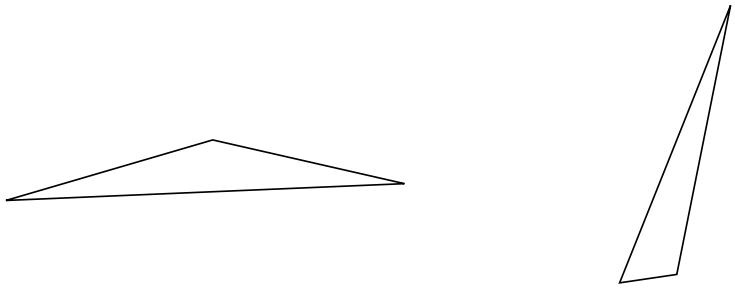


Figura 1: *A la izquierda puede verse un triángulo que es típicamente deforme por exceso y a la derecha uno que es deforme por defecto (siempre en la métrica euclídea). En el primer caso su lado más grande será candidato a ser bisectado. En el segundo, su lado más pequeño será candidato a ser colapsado.*

En el capítulo anterior, en la descripción del algoritmo de suavizado, se habló sobre ciertos elementos de mala calidad que tienden a acumularse en los bordes, en los casos en los que se trabaja sobre métricas con una fuerte anisotropía. A pesar de ser elementos deformes (que bien pueden ser detectados por las funciones anteriormente expuestas), dado que son consecuencia del intento del suavizado por enviar puntos fuera de la triangulación, se observó en la práctica que se obtienen buenos resultados haciendo colapsar ese nodo hacia el borde. Más precisamente se tiene la siguiente rutina:

ALGORITMO $\overline{\tau} =$ `LimpiarBorde`$(\tau, \mathcal{R}, \mathcal{M})$

entrada: Una triangulación admisible τ, un conjunto de tetraedros $\mathcal{R}$ tal que $\forall T \in \mathcal{R}$, T tiene exactamente uno de sus lados en el borde de Ω, y un conjunto $\mathcal{N} \subseteq \mathcal{N}_\tau \cap \partial\Omega$.

salida: Una triangulación admisible $\overline{\tau}$.

1. Para $T \in \mathcal{R}$, si $T \in \tau$, hacer:
   1.1. Fijar e indexar $\mathcal{E}_T = \{E_1, E_2, E_3\}$ de forma tal que $E_1 \subset \partial\Omega$
   1.2. Fijar E_i, $i \in \{2, 3\}$, como el lado que cumpla $|E_i|_\mathcal{M} \leq |E_j|_\mathcal{M}$, $j \in \{2, 3\}$, $j \neq i$.
   1.3. Si $|E_i|_\mathcal{M} < 1$, hacer $\overline{\tau} =$ `ColapsarLado`$(\tau, E_i, \mathcal{N})$.
   1.4. Si $\overline{\tau} = \tau$ y $|E_j|_\mathcal{M} < 1$, hacer $\overline{\tau} =$ `ColapsarLado`$(\tau, E_j, \mathcal{N})$.

Es decir, la función trata de hacer colapsar contra el borde el nodo que el algoritmo de suavizado intenta empujar hacia afuera de la triangulación, primero trata de hacer colapsar el lado más chico que tiene al nodo en cuestión como extremo, si no lo consigue, trata de hacer colapsar el otro lado.

3.2. Fase I

En esta sección se describirá de forma precisa la función `FaseI`. Este procedimiento tomará como entrada una malla admisible τ y un tensor métrico $\mathcal{M}$. La malla τ será modificada de forma iterativa haciendo uso de las operaciones presentadas en el capítulo anterior, respetando el borde de la geometría inicial, con el objetivo de lograr una triangulación en donde las medidas de los lados de todos los elementos bajo $\mathcal{M}$ sean lo más cercanas posible a 1, o al menos que los elementos sean bastante regulares y sus lados estén más o menos cerca de medir 1.

Con lo anterior vemos que esta heurística no es un generador de mallas en el sentido estricto de la palabra ya que debe partir de una malla admisible inicial. A tal efecto deberíamos contemplar una "fase 0", la cual consiste en, dada una geometría de borde, generar una malla admisible. Si bien este problema no es trivial, ha sido ampliamente tratado, con lo cual se supuso que no es excesivo pedirle al usuario que ingrese una malla admisible, siendo la admisibilidad la única propiedad que se pide.

Antes de presentar el pseudo-código, es necesario aclarar cuestiones que conciernen a la relación entre la métrica $\mathcal{M}$ y al nivel de refinamiento.

Como se dijo al principio, la heurística requiere que el usuario provea la métrica $\mathcal{M}$, pero también necesitará ser proveída de un factor de escala R. El factor se utilizará de la siguiente forma: Si $\mathcal{M}$ es la métrica ingresada por el usuario, se redefine como sigue $\mathcal{M} = \mathcal{M} \cdot \frac{1}{R^2}$, donde aquí el símbolo de igualdad se utiliza para denotar asignación.

Esto significa que si antes de introducir R un lado $E \in \mathcal{E}_\tau$ cualquiera medía $|E|_{\mathcal{M}}$, ahora su medida será $|E|_{\mathcal{M}} \cdot \frac{1}{R}$ (Ver ecuación (2.2)).

Pedir el factor R puede parecer redundante, ya que el usuario pudo haber dado como entrada $\mathcal{M} \cdot \frac{1}{R^2}$ en vez de $\mathcal{M}$, sin embargo el objetivo de esto es hacer que pueda manejarse de forma más cómoda el grado de refinamiento.

A continuación se describe en pseudo-código la función `FaseI`.

ALGORITMO $\tau' = \texttt{FaseI}(\tau, \mathcal{M}, R, iter, iter_s, iter_r)$

entrada: Una triangulación admisible τ de Ω, un tensor métrico $\mathcal{M}$ definido en Ω, un número real R, y tres enteros positivos $iter, iter_s$, e $iter_r$.

salida: Una triangulación admisible τ' de Ω.

1. Fijar $\mathcal{N} = \mathcal{N}_\tau \cap \partial\Omega$.
2. Redefinir $\mathcal{M} = \mathcal{M} \cdot \frac{1}{R^2}$ y hacer $\tau' = \tau$, y fijar $\mathcal{R} = \emptyset$.

3. Hacer $iter$ veces

3.1. Para cada $T \in \tau'$, medir sus lados con la métrica $\mathcal{M}$ y marcar como lado de refinamiento al de mayor medida, y fijar su generación en cero.

3.2. Fijar $\mathcal{S} = \{T \in \tau'$ que cumplen, siendo r_T el lado de refinamiento, $(|r_T|_{\mathcal{M}} > C_{sup}) \vee [(\forall E \in \mathcal{E}_T \quad C_{inf} < |E|_{\mathcal{M}} < C_{sup}) \wedge \texttt{EsDefromeExc}(T)]\}$.

3.3. $\tau' = \texttt{Refinar}(\tau', \mathcal{S})$.

3.4. Hacer $\mathcal{R} = \emptyset$.

3.5. $\tau' = \texttt{Suavizar}(\tau', iter_s, \mathcal{R})$.

3.6. $\tau' = \texttt{Retriangular}(\tau', iter_r)$.

3.7. $\tau' = \texttt{LimpiarBorde}(\tau', \mathcal{R}, \mathcal{N})$.

3.8. Fijar $\mathcal{Q} = \{T \in \tau' \text{ que cumplen, siendo } e_T \text{ su lado más chico, } (|e_T|_{\mathcal{M}} < C_{inf}) \vee [(\forall E \in \mathcal{E}_T \quad C_{inf} < |E|_{\mathcal{M}} < C_{sup}) \wedge \texttt{EsDefromeDef}(T)]\}$.

3.9. Para $T \in \mathcal{Q}$, siendo e_T su lado más chico, hacer:

3.9.1. Si $T \in \tau'$, hacer $\tau' = \texttt{ColapsarLado}(\tau', e_T, \mathcal{N})$.

3.10. Hacer $\mathcal{R} = \emptyset$.

3.11. $\tau' = \texttt{Suavizar}(\tau', iter_s, \mathcal{R})$.

3.12. $\tau' = \texttt{Retriangular}(\tau', iter_r)$.

3.13. $\tau' = \texttt{LimpiarBorde}(\tau', \mathcal{R}, \mathcal{N})$.

Para verlo de forma más clara, puede decirse que el funcionamiento consiste en identificar a todos los elementos cuyo lado más grande supera la cota máxima permitida C_{sup} y también a aquellos que sus lados se encuentran dentro del rango aceptable pero que son deformes por exceso. Se bisecta a todos estos elementos (los cuales fueron previamente marcados eligiendo al lado más grande como lado de refinamiento) y a continuación se aplica una regularización a la malla. Hecho esto, se procede a identificar a todos los elementos tales que la medida de su lado más chico esté por debajo de C_{inf}, o bien elementos que estén dentro del rango aceptable pero sean deformes por defecto. Se colapsa el lado más chico de cada uno de ellos, y luego se procede a regularizar la malla. El proceso se itera una cantidad *iter* de veces.

Queda pendiente aún fijar el valor de los muchos parámetros aparecidos en todas las funciones presentadas hasta el momento. Este no es un asunto menor, ya que el funcionamiento de la heurística puede ser bastante sensible ante el cambio de alguno de ellos. A continuación se presentan los valores de cada uno, ajustados empíricamente hasta lograr un funcionamiento aceptable. En la tabla figura el nombre del parámetro, su valor y la función en la que aparece.

Parámetro	Valor	Función que contiene al parámetro
C	0,2	`SuavizarInterior` y `SuavizarBorde`
p_{min}	0,25	`EsDeformeDef`
p_{max}	0,43	`EsDeformeExc`
C_{sup}	1,334	`FaseI`
C_{inf}	0,666	`FaseI`

Para finalizar, falta hacer una elección de la función f que se utiliza tanto en `SuavizarInterior` como en `SuavizarBorde`, la cual simula el efecto de una fuerza sobre el nodo que se busca reposicionar. Se optó por utilizar la función propuesta en [1], modificándola para que solamente simule el efecto de fuerzas repulsivas. La función utilizada en el artículo es $g : \mathbb{R} \to \mathbb{R}$, $g(x) := (1 - x^4)e^{-x^4}$. En este trabajo se utilizó $f := g^+$, es decir:

$$f(x) = \begin{cases} g(x) & x \in \{y \in \mathbb{R} \text{ tal que } g(x) > 0\} \\ 0 & \text{en otro caso} \end{cases}$$

Se observó en la práctica que utilizando solamente fuerzas repulsivas se obtenían mejores resultados. Esto puede deberse a la tendencia del algoritmo a generar una longitud promedio levemente menor a 1 (en la métrica $\mathcal{M}$), con lo cual de utilizarse la función g la mayoría de las fuerzas involucradas serían repulsivas. Sin embargo, en ese caso, las pocas fuerzas atractivas que se utilizan parecen no contribuir a la convergencia, al no dispersar de forma adecuada los nodos a lo largo del dominio.

A pesar de esto, los resultados utilizando tanto f como g son similares para un número de iteraciones suficientemente grande.

3.3. Fase II

Una vez obtenida una malla anisotrópica con la función `FaseI`, puede realizarse un proceso de refinamiento de la malla con el objetivo de reducir el error de aproximación. Las técnicas de refinamiento estándar (bisección por el lado más largo, cualquier algoritmo de bisección en general, cuadrisección, etc.) no aseguran, a priori, la generación de elementos de calidad en una malla anisotrópica. Sin embargo, como se verá más adelante, bajo ciertas hipótesis los elementos producidos por la función `Refinar` conservan una calidad similar a la de los elementos de la malla original (en cualquiera sea la métrica en la cual los elementos son regulares).

La idea de la fase de refinamiento consiste en, una vez conseguida una malla τ, más o menos regular sobre una métrica $\mathcal{M}$ con cierto nivel de refinamiento R, bisectar los elementos ya sea local o globalmente para conseguir un nivel más bajo del error, tratando de estropear lo menos posible la regularidad de la malla original en el proceso. A tal fin se utiliza el algoritmo de bisección ya descrito en el capítulo anterior.

Una de las posibles formas de utilizarlo es refinando de manera local elementos en donde se detecte que el error sea alto, haciendo uso de algún tipo estimador (a priori o a posteriori).

Otra posible forma es, una vez obtenida una malla más o menos regular sobre $\mathcal{M}$, refinar todos los elementos al mismo tiempo. A tal fin se expone más abajo la función `RefinarUniforme`.

La idea consiste en refinar de la forma más "pareja" posible los elementos, haciendo que el nivel de refinamiento de la malla se mantenga más o menos igual en todo el dominio.

El hecho de que los elementos obtenidos con el algoritmo de bisección mantengan una calidad (en la métrica $\mathcal{M}$) similar a la del elemento padre del cual se obtienen, como ya se dijo, se demuestra en el capítulo siguiente.

Se expone a continuación el pseudo código de la función `RefinarUniforme`:

ALGORITMO $\tau' =$ `RefinarUniforme`$(\tau, G, \mathcal{M})$

entrada: Una triangulación admisible τ de Ω, un número entero positivo G, y un tensor métrico $\mathcal{M}$ definido en Ω.

salida: Una triangulación admisible τ'.

1. Hacer $\tau' = \tau$.

2. Para cada $T \in \tau'$, medir sus lados con la métrica $\mathcal{M}$ y marcar como lado de refinamiento al de mayor medida, y fijar su generación en cero.

3. Definir $i = 0$. Mientras $i < G$ hacer:

   3.1. Fijar $\mathcal{S} = \{T \in \tau' \text{cuya generación es igual a } i\}$

   3.2. $\tau' =$ `Refinar`$(\tau', \mathcal{S})$

   3.3. $i = i + 1$

Visto de manera más simple, el algoritmo recibe además de la malla y la métrica, un número G que puede pensarse como "la mínima generación permitida". La malla que `RefinarUniforme` da como salida tendrá entonces elementos de generación como mínimo G y, por la forma en que realiza la elección de los elementos que se bisectan, en general la generación será más o menos pareja, logrando de esta manera una malla cuasi-uniforme.

Empíricamente puede verse que la uniformidad se mantiene mejor siempre que se utiliza G par. Esto se debe a que para valores pares del parámetro G, al terminar la rutina todos los elementos tendrán el mismo número de generación. La razón de esto es que, por ejemplo para $G = 2$, el algoritmo "cuadrisecta" los elementos, sin necesidad de que haya propagación de las bisecciones para asegurar la admisibilidad de la malla. Con lo cual, si se aplica `RefinarUniforme` sobre una malla más o menos regular, para luego aplicar nuevamente `FaseI`, se observan mejores resultados para G par.

Los detalles relacionados con variantes y modos de uso de los algoritmos expuestos se discuten al final del Capítulo 5.

Adicionalmente se define una variante de `RefinarUniforme`, en la cual antes de realizar el paso de bisección se vuelve a marcar como lado de refinamiento al lado más grande de cada elemento en la métrica $\mathcal{M}$:

ALGORITMO $\tau' =$ `RefinarUniforme*`$(\tau, G, \mathcal{M})$

entrada: Una triangulación admisible τ de Ω, un número entero positivo G, y un tensor métrico $\mathcal{M}$ definido en Ω.

salida: Una triangulación admisible τ'.

1. Hacer $\tau' = \tau$.
2. Para cada $T \in \tau'$ fijar su generación en cero.
3. Definir $i = 0$. Mientras $i < G$ hacer:

3.1. Para cada $T \in \tau'$, medir sus lados con la métrica $\mathcal{M}$ y marcar como lado de refinamiento al de mayor longitud.

3.2. Fijar $\mathcal{S} = \{T \in \tau' \text{cuya generación es igual a } i\}$

3.3. $\tau' =$ `Refinar`$(\tau', \mathcal{S})$

3.4. $i = i + 1$

Esta variante presenta un mejor comportamiento en cuanto al error de aproximación cuando se parte de mallas muy gruesas, sin embargo tiene la desventaja de tener que volver a medir y marcar en cada paso.

Capítulo 4

Refinamiento bajo una métrica anisotrópica

Habiendo detallado el funcionamiento de la heurística, el objetivo a continuación es probar algunos resultados sobre su funcionamiento. Más precisamente sobre el funcionamiento de la fase de refinamiento.

Dado que la fase I utiliza técnicas estándar para lograr su objetivo, puede no resultar extraño que en la mayoría de los casos funcione de forma decente. Sin embargo, en la segunda fase la métrica se utiliza para marcar los lados más largos de los triángulos solamente al principio, para continuar, luego, refinando una cantidad arbitraria de veces.

En vista de lo anterior surgen bastantes preguntas acerca de su funcionamiento ya que, a priori, no tendría por qué generar una malla regular en $\mathcal{M}$. Incluso, se verá más adelante que, bajo hipótesis razonables, ni siquiera hace falta marcar al principio los lados más grandes en $\mathcal{M}$, las marcas iniciales se pueden hacer al azar y aun así se mantendrá un buen funcionamiento.

En el artículo [5] se demuestra, de forma indirecta, que la calidad de los elementos que se obtienen bisectando la cantidad de veces que sea a un elemento padre, mientras se siga la regla de marcado, tendrán una calidad más o menos parecida a la de éste. A tal efecto ellos demuestran que la cantidad de tetraedros que es posible obtener aplicando el algoritmo sobre un elemento padre es finita salvo traslaciones y dilataciones. Esto siempre bajo la métrica euclídea.

Cabe preguntarse entonces, bajo qué condiciones los elementos generados a partir del algoritmo de bisección mantendrán la regularidad en la métrica $\mathcal{M}$, y también cómo dependerá ésta de la regularidad del elemento padre, así como de la misma métrica. Este capítulo está orientado a responder estas preguntas.

Todo el desarrollo de este capítulo se hará para una malla tridimensional de tetraedros, utilizando a tal efecto la versión tridimensional del ya mencionado algoritmo de bisección. Esto se hace, por un lado, para estar a tono con los resultados del artículo [5] que se utilizarán, y por otro lado, para darle una mayor generalidad a los resultados aquí alcanzados, ya que la generalización de los resultados de tres dimensiones a dos resulta trivial, no así su recíproco.

4.1. Medida de deformidad

El objetivo será, dada una medida de cuán deforme es un tetraedro sobre una métrica $\mathcal{M}$, lograr acotar la deformidad de un elemento producido por el algoritmo de bisección con algún múltiplo de la deformidad del elemento padre.

A tal efecto, se considera como elemento de referencia $\hat{T}$ a un tetraedro cuyos lados midan 1 (en la métrica euclídea), y esté centrado en el origen. Por ejemplo, para hacer una elección precisa, se elige aquel que cumple lo anterior y posee una de sus caras paralela al plano xy, y uno de sus lados paralelo a la recta x.

Sea $\mathcal{N}_{\hat{T}} = \{\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{x}_4\}$, se notará como $cc(\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{x}_4)$ a la cápsula convexa formada por esos puntos. Se tiene entonces $cc(\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{x}_4) = \hat{T}$.

Definición 4.1.1 *Sea $T = cc(x_1, x_2, x_3, x_4)$, con $x_i \in \mathbb{R}^3$, $i = 1, ..., 4$ afínmente independientes. Se define D_T como la transformación afín $D_T : \mathbb{R}^3 \to \mathbb{R}^3$ que cumple:*

$$D_T(x_i) = \hat{x}_i \quad \forall i = 1, ..., 4 \tag{1.1}$$

En lo siguiente se hará un abuso de notación y se llamará también D_T a la matriz de la transformación D_T. Lo mismo se hará también en general con cualquier transformación afín. Siempre que el contexto en el que se use de lugar a una confusión, se aclarará a cuál de los dos conceptos se está refiriendo.

Adicionalmente se notará, siendo $T = cc(x_1, x_2, x_3, x_4)$, y siendo D_T la transformación afín antes definida, $D_T(T) := cc(D_T(x_1), D_T(x_2), D_T(x_3), D_T(x_4))$.

A continuación se define la función Def, la cual permitirá medir cuánto hay que deformar a un elemento para llevarlo al elemento de referencia (o viceversa). Dado que las operaciones de traslación, rotación, simetría, y escala uniforme no deforman en el sentido que es de interés observar, surge de forma intuitiva utilizar el número de condición para medir la deformidad. Más precisamente se define:

Definición 4.1.2 *Sea τ el conjunto de todos los tetraedros no degenerados. Se define la función $Def : \tau \to \mathbb{R}$, como $Def(T) := Cond_2(D_T)$. Donde $Cond_2(D_T) = ||D_T||_2||D_T^{-1}||_2$.*

Cabe aclarar que en la anterior definición se utiliza la norma matricial $|| \cdot ||_2$, la misma es aquella inducida por la norma vectorial $|| \cdot ||_2$.

Buena definición:

Dado que Def está definida sobre los tetraedros no degenerados, D_T resulta inversible, con lo cual D_T^{-1} está bien definida. Resta ver que para dos posibles escrituras de un mismo tetraedro la función Def no cambia de valor. Esto es, sea $T_1 = cc(x_1, x_2, x_3, x_4)$, sea P una permutación del conjunto de índices $\{1, ..., 4\}$, y sea $T_2 = cc(x_{P(1)}, x_{P(2)}, x_{P(3)}, x_{P(4)})$, se busca ver que $Def(T_1) = Def(T_2)$.

A tal efecto se define la transformación afín O de tal manera que $O(\hat{x}_i) = \hat{x}_{P(i)}$, para $i = 1, ..., 4$. Por la Definición (4.1.1), se tiene que $D_{T_1}(x_i) = \hat{x}_i$ y $D_{T_2}(x_{P(i)}) = \hat{x}_i$. Además, aplicando O a ambos lados de la igualdad anterior, se tiene

$$O \circ D_{T_2}(x_{P(i)}) = O(x_i) = x_{P(i)}$$

De donde se deduce

$$O \circ D_{T_2} = D_{T_1}$$

A partir de esto se tiene el siguiente desarrollo:

$$Def(T_1) = Cond_2(D_{T_1}) = Cond_2(OD_{T_2}) \leq Cond_2(O)Cond_2(D_{T_2}) = Cond_2(O)Def(T_2) \quad (1.2)$$

Donde en el anteúltimo paso se utiliza la propiedad submultiplicativa de la norma matricial $||.||_2$. (ver Definición (4.1.2)).

Dado que el elemento de referencia tiene todos sus lados iguales, y está centrado en el origen, O puede ser vista como una composición de simetrías a lo largo de determinados planos que pasan por el origen. Con lo cual se deduce que $Cond_2(O) = 1$. De esto, junto con la ecuación (1.2), se sigue:

$$Def(T_1) \leq Def(T_2) \quad (1.3)$$

Dado que $O \circ D_{T_2} = D_{T_1}$, se tiene $D_{T_2} = O^{-1} \circ D_{T_1}$. Tomando esto en cuenta, y haciendo un desarrollo análogo al anterior se obtiene la desigualdad inversa. Luego, $Def(T_1) = Def(T_2)$ y, por lo tanto, Def está bien definida. ■

La función Def brinda una manera de medir la deformidad de un elemento. Dicha deformidad puede ser vista como una medida de distancia entre un elemento cualquiera y el elemento de referencia $\hat{T}$.

Mientras más grande sea $Def(T)$, más hay que deformar al elemento de referencia $\hat{T}$ para transformarlo en T (o viceversa), siendo 1 el valor mínimo que Def puede tomar en el caso que T tenga todas sus aristas de la misma medida. Esto es porque, en tal caso, T no es más que una traslación, escala uniforme y/o rotación de $\hat{T}$.

Hasta aquí se tiene una noción de deformidad sobre la métrica euclídea. A continuación se extiende este concepto de deformidad, pero bajo una métrica $\mathcal{M}$ constante.

Definición 4.1.3 *Sea τ el conjunto de todos los tetraedros no degenerados y sea $\mathbb{S} := \{\mathcal{M} \in \mathbb{R}^{3\times 3}$ tal que $\mathcal{M}$ es simétrica y definida positiva$\}$. Se define la función $Def : \tau \times \mathbb{S} \to \mathbb{R}$, como $Def(T, \mathcal{M}) := Def(\sqrt{\mathcal{M}}(T))$.*

La raíz en la definición anterior denota la raíz matricial, operación que está bien definida al ser $\mathcal{M}$ s.d.p. .

Para dar una idea de por qué se eligió esta definición de deformidad en la métrica $\mathcal{M}$, se puede suponer $T = cc(x_1, x_2, x_3, x_4)$ elemento regular sobre $\mathcal{M}$ (o sea, con todos sus lados de igual longitud medidos en $\mathcal{M}$). Si la Definición (4.1.3) nos diera una idea razonable de la deformidad en $\mathcal{M}$, debería suceder que $Def(T, \mathcal{M}) = 1$. Para ver que esto es así, es necesario que la longitud de todas las aristas de $\sqrt{\mathcal{M}}(T)$ sean iguales. Como $\sqrt{\mathcal{M}}(T) = cc(\sqrt{\mathcal{M}}(x_1), \sqrt{\mathcal{M}}(x_2), \sqrt{\mathcal{M}}(x_3), \sqrt{\mathcal{M}}(x_4))$, se tiene, para una arista E de $\sqrt{\mathcal{M}}(T)$, con extremos $\sqrt{\mathcal{M}}(x_i)$ y $\sqrt{\mathcal{M}}(x_j)$:

$$|E|^2 = (\sqrt{\mathcal{M}}x_i - \sqrt{\mathcal{M}}x_j)^t \cdot (\sqrt{\mathcal{M}}x_i - \sqrt{\mathcal{M}}x_j) = (x_i - x_j)^t \sqrt{\mathcal{M}}^t \cdot \sqrt{\mathcal{M}}(x_i - x_j)$$

Dado que $\sqrt{\mathcal{M}} \cdot \sqrt{\mathcal{M}} = \mathcal{M}$, trasponiendo a ambos lados de la desigualdad se tiene que

$$\sqrt{\mathcal{M}}^t \cdot \sqrt{\mathcal{M}}^t = \mathcal{M}^t = \mathcal{M}$$

Al ser $\mathcal{M}$ simétrica y definida positiva, la ecuación $X^2 = \mathcal{M}$ tiene una única solución simétrica y definida positiva, luego $\sqrt{\mathcal{M}}^t = \sqrt{\mathcal{M}}$. Con lo cual, a partir del desarrollo anterior, combinado con la ecuación (2.2) del Capítulo 1 se obtiene:

$$|E|^2 = (x_i - x_j)^t \mathcal{M}(x_i - x_j) = |\overline{x_i x_j}|^2_{\mathcal{M}} = m$$

Como se supuso T regular en $\mathcal{M}$, se tiene, para todo par de índices $i, j \in \{1, ..., 4\}$, $i \neq j$, $|\overline{x_i x_j}|^2_{\mathcal{M}} = m$. Con lo cual, cualquier arista E de $\sqrt{\mathcal{M}}(T)$ verifica $|E|^2 = m$. Luego, al medir todas lo mismo, $\sqrt{\mathcal{M}}(T)$ es una traslación, rotación, y/o escala uniforme de $\hat{T}$. Por lo tanto se tiene:

$$Def(T, \mathcal{M}) = Def(\sqrt{\mathcal{M}}(T)) = 1$$

Para finalizar esta sección, se mostrará la relación entre la medida de deformidad aquí presentada y la medida de deformidad utilizada en la teoría clásica del método de elementos finitos.

La medida clásica de deformidad de los elementos consiste en calcular el cociente entre el diámetro del elemento en cuestión y el supremo de los diámetros de todas las bolas contenidas en éste (diámetro interior).

Es decir, siendo A un conjunto acotado de $\mathbb{R}^d$, para una cierta norma $||\cdot||$ definida en $\mathbb{R}^d$, se define $h_A := \sup_{x,y \in A} ||x - y||$ y $\rho_A := \sup\{h_B$, con B una bola contenida en $A\}$. En base a esto, en el enfoque clásico, se dirá que una familia de triangulaciones τ_h (donde el índice h está relacionado con el tamaño de la malla) es regular si existe $\sigma > 0$ de forma tal que para todo h se cumpla:

$$\frac{h_T}{\rho_T} \leq \sigma \quad \forall T \in \tau_h$$

Con lo cual, $\frac{h_T}{\rho_T}$ puede ser vista como la medida de deformidad clásica.

A continuación se verá que la medida de deformidad utilizada en este trabajo es en esencia equivalente a la clásica, incluso, en algún sentido, en el caso anisotrópico. Más precisamente se tiene la siguiente proposición.

Proposición 4.1.1 *Sean T y $\hat{T}$ dos conjuntos de $\mathbb{R}^d$, cerrados, acotados, con interior no vacío y afínmente equivalentes, es decir, que existe $D_T : \mathbb{R}^d \to \mathbb{R}^d$ transformación afín tal que $D_T(T) = \hat{T}$, y sea $||\cdot||$ una norma definida en $\mathbb{R}^d$. Luego, se cumplen las siguientes desigualdades:*

$$\frac{h_T}{\rho_T} \cdot \frac{\rho_{\hat{T}}}{h_{\hat{T}}} \leq ||D_T^{-1}|| \cdot ||D_T|| \leq \frac{h_T}{\rho_T} \cdot \frac{h_{\hat{T}}}{\rho_{\hat{T}}} \tag{1.4}$$

Demostración:

Se buscará ver primero que se cumple:

$$||D_T^{-1}|| \cdot ||D_T|| \leq \frac{h_T}{\rho_T} \cdot \frac{h_{\hat{T}}}{\rho_{\hat{T}}} \tag{1.5}$$

En efecto, se tiene lo siguiente:

$$||D_T^{-1}|| = \sup_{x\neq 0} \frac{||D_T^{-1}\cdot x||}{||x||} = \sup_{||z||=1} ||D_T^{-1}\cdot z|| = \frac{1}{\rho_{\hat{T}}}\cdot \sup_{||z||=1} ||D_T^{-1}\cdot \rho_{\hat{T}} z|| = \frac{1}{\rho_{\hat{T}}}\cdot \sup_{||w||=\rho_{\hat{T}}} ||D_T^{-1}\cdot w|| \quad (1.6)$$

Sea un w cualquiera en $\mathbb{R}^n$ tal que $||w|| = \rho_{\hat{T}}$. Por definición de $\rho_{\hat{T}}$, existe una bola B de ese diámetro tal que $B \subseteq \hat{T}$. Luego, existen $\hat{x}$ y $\hat{y} \in B$ tal que $w = \hat{x} - \hat{y}$. Con esto se tiene que:

$$||D_T^{-1}\cdot w|| = ||D_T^{-1}\cdot(\hat{x}-\hat{y})|| = ||D_T^{-1}\cdot\hat{x} - D_T^{-1}\cdot\hat{y}|| = ||D_T^{-1}(\hat{x}) - D_T^{-1}(\hat{y})|| \quad (1.7)$$

Dado que $D_T^{-1}(\hat{x})$ y $D_T^{-1}(\hat{y}) \in T$, se sigue que $||D_T^{-1}(\hat{x}) - D_T^{-1}(\hat{y})|| \leq h_T$. Esto, combinado con la ecuación (1.6) implica:

$$||D_T^{-1}|| \leq \frac{h_T}{\rho_{\hat{T}}} \quad (1.8)$$

A su vez, queda probada también la desigualdad:

$$||D_T|| \leq \frac{h_{\hat{T}}}{\rho_T} \quad (1.9)$$

Y finalmente de (1.8) y (1.9) se deduce la desigualdad (1.5).

Cabe observar que al ser T y $\hat{T}$ conjuntos con interior no vacío, ambos contienen al menos una bola de radio $\varepsilon > 0$, para un ε lo suficientemente chico. De esto y sumado al hecho de que son conjuntos acotados, se sigue que $0 < h_T, h_{\hat{T}}, \rho_T, \rho_{\hat{T}} < \infty$. Adicionalmente se tiene que en la definición de ρ_T y ρ_T el supremo es en realidad un máximo, al ser T y $\hat{T}$ cerrados. Con lo cual $\hat{x}$ e $\hat{y}$ están bien definidos.

Queda por ver la desigualdad:

$$\frac{h_T}{\rho_T}\cdot\frac{\rho_{\hat{T}}}{h_{\hat{T}}} \leq ||D_T^{-1}||\cdot||D_T|| \quad (1.10)$$

Para hacer esto, se tiene que, por la definición de diámetro y al ser T un conjunto cerrado, existen $x, y \in T$ tales que $h_T = ||x-y||$. Al ser T y $\hat{T}$ afínmente equivalentes existen $\hat{x}$ e $\hat{y} \in \hat{T}$ tales que $D_T^{-1}(\hat{x}) = x$ y $D_T^{-1}(\hat{y}) = y$. Con lo cual se tiene:

$$h_T = ||x-y|| = ||D_T^{-1}(\hat{x}) - D_T^{-1}(\hat{y})|| = ||D_T^{-1}\cdot(\hat{x}-\hat{y})|| = ||\hat{x}-\hat{y}||\cdot||D_T^{-1}\cdot\frac{(\hat{x}-\hat{y})}{||\hat{x}-\hat{y}||}|| \quad (1.11)$$

En la tercera igualdad se utiliza el hecho de que, al ser D_T^{-1} transformación afín, $D_T^{-1}(\hat{x}) - D_T^{-1}(\hat{y}) = D_T^{-1}\cdot(\hat{x}-\hat{y})$, donde en le segundo término D_T^{-1} denota la matriz de la transformación.

Dado que $\hat{x}$ e $\hat{y} \in \hat{T}$, se tiene $||\hat{x}-\hat{y}|| \leq h_{\hat{T}}$. Esto junto con (1.11) y la definición de $||D_T^{-1}||$ implica:

$$h_T = ||\hat{x}-\hat{y}||\cdot||D_T^{-1}\cdot\frac{(\hat{x}-\hat{y})}{||\hat{x}-\hat{y}||}|| \leq h_{\hat{T}}\cdot\sup_{||z||=1}||D_T^{-1}\cdot z|| = h_{\hat{T}}\cdot||D_T^{-1}|| \quad (1.12)$$

Por otro lado, se buscará ver que

$$\frac{\rho_{\hat{T}}}{||D_T||} \leq \rho_T \quad (1.13)$$

A tal fin, se observa que, por la definición de $\rho_{\hat{T}}$ y al ser $\hat{T}$ un conjunto cerrado, existe $B(\hat{x}, \frac{\rho_{\hat{T}}}{2}) \subseteq \hat{T}$ bola centrada en un $\hat{x} \in \hat{T}$ de radio $\frac{\rho_{\hat{T}}}{2}$. A continuación se verá que la bola $B(x, \frac{\rho_{\hat{T}}}{2||D_T||}) \subseteq T$, con $D_T(\hat{x}) = x$.

En efecto, dado $y \in B(x, \frac{\rho_{\hat{T}}}{2||D_T||})$, se afirma que $D_T(y) \in B(\hat{x}, \frac{\rho_{\hat{T}}}{2}) \subseteq \hat{T}$. De ser cierto, esto traería como consecuencia que $y \in T$, dado que T y $\hat{T}$ son afínmente equivalentes, y por lo tanto, $B(x, \frac{\rho_{\hat{T}}}{2||D_T||}) \subseteq T$. Para ver que $D_T(y) \in B(\hat{x}, \frac{\rho_{\hat{T}}}{2})$, se tiene lo siguiente:

$$||\hat{x} - D_T(y)|| = ||D_T(x) - D_T(y)|| = ||D_T \cdot (x - y)|| \leq ||D_T|| \cdot ||x - y|| \tag{1.14}$$

Como se supuso $y \in B(x, \frac{\rho_{\hat{T}}}{2||D_T||})$, se tiene $||x - y|| \leq \frac{\rho_{\hat{T}}}{2||D_T||}$ con lo cual, a partir de (1.14) se tiene:

$$||\hat{x} - D_T(y)|| \leq ||D_T|| \cdot \frac{\rho_{\hat{T}}}{2||D_T||} = \frac{\rho_{\hat{T}}}{2} \tag{1.15}$$

Luego, $D_T(y) \in B(\hat{x}, \frac{\rho_{\hat{T}}}{2}) \subseteq \hat{T}$, lo que implica $y \in T$ y, por lo tanto, $B(x, \frac{\rho_{\hat{T}}}{2||D_T||}) \subseteq T$.

Del hecho de que T contenga una bola de diámetro $\frac{\rho_{\hat{T}}}{||D_T||}$, sumado a la definición de ρ_T se obtiene (1.13).

Finalmente, combinando (1.13) y (1.12) se obtiene:

$$\frac{h_T}{\rho_T} \leq \frac{h_{\hat{T}}}{\rho_{\hat{T}}} \cdot ||D_T^{-1}|| \cdot ||D_T|| \tag{1.16}$$

De donde se sigue (1.10) . ■

Ahora volviendo a T y $\hat{T}$ como un elemento cualquiera y el elemento de referencia respectivamente, $\frac{h_{\hat{T}}}{\rho_{\hat{T}}}$ puede ser pensado como una constante $C_{\hat{T}}$ que sólo depende del elemento de $\hat{T}$. Luego, a partir de la Proposición (4.1.1) y la Definición (4.1.2), se deduce de forma inmediata la siguiente relación:

$$\frac{h_T}{\rho_T} \cdot C_{\hat{T}}^{-1} \leq Def(T) \leq \frac{h_T}{\rho_T} \cdot C_{\hat{T}} \tag{1.17}$$

Con lo cual puede decirse que la medida de deformidad aquí utilizada es en esencia equivalente a la medida clásica, al menos en el caso isotrópico.

Cabe entonces hacerse la siguiente pregunta: ¿Es la medida de deformidad de la Definición (4.1.3) equivalente, en algún sentido, a la medida clásica? La respuesta es sí. Para ver esto, primero hay que hacer una observación acerca del cómputo de $Def(T, \mathcal{M})$. Se tiene, por definición, lo siguiente:

$$Def(T, \mathcal{M}) = Cond_2(D_{\sqrt{\mathcal{M}}(T)}) = ||D_{\sqrt{\mathcal{M}}(T)}||_2 \cdot ||D^{-1}_{\sqrt{\mathcal{M}}(T)}||_2 \tag{1.18}$$

Donde $T = cc(x_1, x_2, x_3, x_4)$ y $D_{\sqrt{\mathcal{M}}(T)}$ es aquella transformación afín que verifica

$$D_{\sqrt{\mathcal{M}}(T)}(\sqrt{\mathcal{M}}(x_i)) = \hat{x}_i \quad \forall i \in \{1, ..., 4\}$$

Dado que $(D_{\sqrt{\mathcal{M}}(T)} \circ \sqrt{\mathcal{M}})(x_i) = \hat{x}_i$, se sigue que $D_{\sqrt{\mathcal{M}}(T)} \circ \sqrt{\mathcal{M}} = D_T$, y por lo tanto $D_{\sqrt{\mathcal{M}}(T)} = D_T \circ \sqrt{\mathcal{M}}^{-1}$. Luego

$$Def(T,\mathcal{M}) = ||\sqrt{\mathcal{M}} \cdot D_T^{-1}||_2 \cdot ||D_T \cdot \sqrt{\mathcal{M}}^{-1}||_2 \tag{1.19}$$

Por otro lado se tiene que:

$$||x||_{\mathcal{M}}^2 = x^t \mathcal{M} x = x^t \sqrt{\mathcal{M}} \cdot \sqrt{\mathcal{M}} x = (\sqrt{\mathcal{M}} x)^t \cdot \sqrt{\mathcal{M}} x = ||\sqrt{\mathcal{M}}(x)||_2^2$$

Además, se define D_T^* como la transformación afín que cumple $D_T^*(x_i) = \sqrt{\mathcal{M}}^{-1}(\hat{x}_i)$. Y, en función de lo anterior, se define $Def^*(T) = Cond_{\mathcal{M}}(D_T^*) = ||D_T^*||_{\mathcal{M}} \cdot ||D_T^{*-1}||_{\mathcal{M}}$. La buena definición de esta función es análoga a la ya probada para $Def(T)$.

No está de más observar que esta nueva construcción es totalmente análoga a la hecha para $Def(T)$, con la salvedad que se hace todo bajo la norma $||\cdot||_{\mathcal{M}}$. Con lo cuál $Def^*(T)$ puede pensarse como una medida de cuán deforme es T en relación al elemento de referencia $\sqrt{\mathcal{M}}^{-1}(\hat{T})$, bajo la métrica $\mathcal{M}$.

Notar, además, que todas las aristas de $\sqrt{\mathcal{M}}^{-1}(\hat{T})$ miden 1 en la métrica $\mathcal{M}$. Con el objetivo de simplificar la notación, se define $\hat{T}^* := \sqrt{\mathcal{M}}^{-1}(\hat{T})$.

Ahora bien, como $D_T^*(x_i) = \sqrt{\mathcal{M}}^{-1}(\hat{x}_i)$, se tiene $(\sqrt{\mathcal{M}} \circ D_T^*)(x_i) = \hat{x}_i$, con lo cual $\sqrt{\mathcal{M}} \circ D_T^* = D_T$. En base a esto se deduce lo siguiente:

$$||D_T^*||_{\mathcal{M}} = \sup_{z \neq 0} \frac{||D_T^* \cdot z||_{\mathcal{M}}}{||z||_{\mathcal{M}}} = \sup_{z \neq 0} \frac{||\sqrt{\mathcal{M}} \cdot D_T^* \cdot z||_2}{||\sqrt{\mathcal{M}} \cdot z||_2} = \sup_{z \neq 0} \frac{||D_T \cdot z||_2}{||\sqrt{\mathcal{M}} \cdot z||_2} \tag{1.20}$$

Por otro lado se tiene

$$||D_T \cdot \sqrt{\mathcal{M}}^{-1}||_2 = \sup_{u \neq 0} \frac{||D_T \cdot \sqrt{\mathcal{M}}^{-1} \cdot u||_2}{||u||_2} = \sup_{z \neq 0} \frac{||D_T \cdot z||_2}{||\sqrt{\mathcal{M}} \cdot z||_2} \tag{1.21}$$

Donde en el último paso se hace el cambio de variable $u = \sqrt{\mathcal{M}} \cdot z$. De esto se deduce que $||D_T \cdot \sqrt{\mathcal{M}}^{-1}||_2 = ||D_T^*||_{\mathcal{M}}$. Con un argumento análogo puede verse que $||\sqrt{\mathcal{M}} \cdot D_T^{-1}||_2 = ||D_T^{*-1}||_{\mathcal{M}}$. Luego,

$$Def(T,\mathcal{M}) = Def^*(T) \tag{1.22}$$

Esto, combinado con la Proposición (4.1.1), implica

$$\frac{h_T}{\rho_T} \cdot C_{\hat{T}^*}^{-1} \leq Def(T,\mathcal{M}) \leq \frac{h_T}{\rho_T} \cdot C_{\hat{T}^*} \tag{1.23}$$

Donde $C_{\hat{T}^*} = \frac{h_{\hat{T}^*}}{\rho_{\hat{T}^*}}$, y los diámetros son medidos en la métrica $\mathcal{M}$ al momento de calcular $h_{\hat{T}^*}$, $\rho_{\hat{T}^*}$, h_T y ρ_T.

Con lo anterior, entonces, puede decirse que $Def(T,\mathcal{M})$ es equivalente a la medida clásica de deformidad, cuando para el cómputo de ésta se utiliza la norma $||\cdot||_{\mathcal{M}}$.

Dado que algunas veces importa generar mallas cuyos elementos no tengan ángulos máximos demasiado cercanos a π (como por ejemplo para asegurar un buen comportamiento del error cuando se utiliza la interpolada de Lagrange), cabe preguntarse si la regularidad en una

métrica arbitraria asegura que los ángulos máximos (ángulos medidos en la métrica euclídea) se mantengan lejos de π.

Estrictamente hablando la respuesta es no, ya que es posible construir una métrica y un triángulo regular en ella de forma tal que su ángulo máximo esté arbitrariamente cerca de π (donde la cercanía a π depende de la anisotropía de la métrica). Sin embargo, se observa empíricamente que la tendencia de los elementos a acomodarse de forma "alineada" a los autovectores de la métrica tiene como consecuencia que los ángulos máximos no sean demasiado grandes.

Para ver que cómo construir una métrica y un triángulo regular de forma tal que su ángulo máximo esté arbitrariamente cerca de π, dado $P \in \mathbb{R}_{>0}$ se define $\mathcal{M}_P$ de la siguinete forma:

$$\mathcal{M}_P := \begin{pmatrix} P & 0 \\ 0 & 1 \end{pmatrix}$$

Ahora, se consideran los nodos $\hat{x}_1 = (0, 1/2)$, $\hat{x}_2 = (0, -1/2)$ y $\hat{x}_3 = (\sqrt{3}/2, 0)$. Es fácil ver que el triángulo $\hat{T} = cc(x_1, x_2, x_3)$ es equilátero y sus lados miden 1 (en la métrica habitual). Si para $i \in \{1, 2, 3\}$ se define $x_i := \sqrt{\mathcal{M}}_P^{-1} \hat{x}_i$ y $T := \sqrt{\mathcal{M}}_P^{-1}(\hat{T})$ se tiene que todos los lados de T miden 1 en la métrica $\mathcal{M}_P$. Para ver esto se tiene el siguiente desarrollo, siendo i, $j \in \{1, 2, 3\}$, $i \neq j$:

$$|\overline{x_i x_j}|^2_{\mathcal{M}_P} = (x_i - x_j)^t \mathcal{M}_P (x_i - x_j) = [\sqrt{\mathcal{M}}_P^{-1}(\hat{x}_i - \hat{x}_j)]^t \mathcal{M}_P \cdot \sqrt{\mathcal{M}}_P^{-1}(\hat{x}_i - \hat{x}_j) =$$

$$= (\hat{x}_i - \hat{x}_j)^t \sqrt{\mathcal{M}}_P^{-1} \cdot \sqrt{\mathcal{M}}_P \cdot \sqrt{\mathcal{M}}_P \cdot \sqrt{\mathcal{M}}_P^{-1}(\hat{x}_i - \hat{x}_j) = (\hat{x}_i - \hat{x}_j)^t (\hat{x}_i - \hat{x}_j) =$$

$$= |\overline{\hat{x}_i \hat{x}_j}|^2 = 1$$

Por otro lado, a partir de la Figura 1 puede verse que el ángulo máximo de T tiende a π cuando P tiende a infinito.

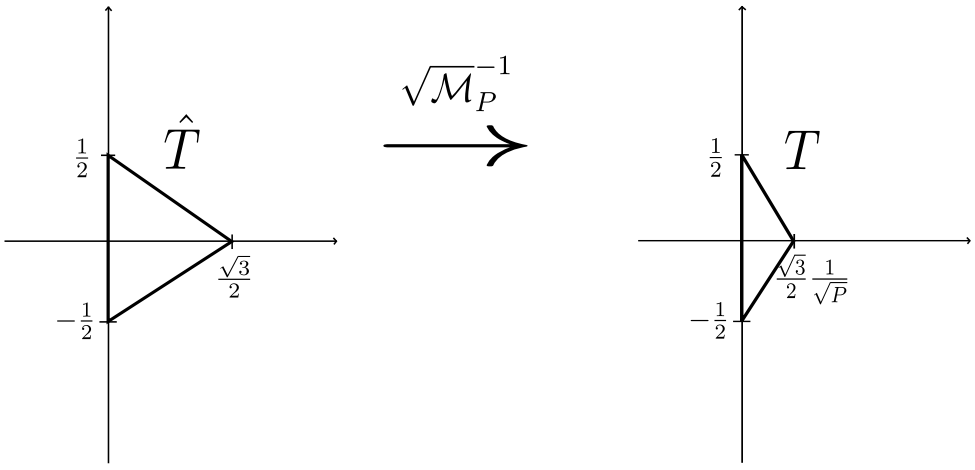


Figura 1: *Efecto de la transformación* $\sqrt{\mathcal{M}}_P^{-1}$ *sobre el triángulo* $\hat{T}$.

4.2. Bisección de tetraedros

En esta sección se expondrá la versión tridimensional del algoritmo de bisección utilizado en la heurística ya presentada, así como algunos resultados útiles sobre el mismo. Como se dijo anteriormente, tanto el algoritmo como los resultados se encuentran en [5].

Antes de presentar el algoritmo, se fijará algo de notación.

Por un lado, se considera $\Omega \subset \mathbb{R}^3$ un conjunto cerrado, conexo, poligonal, con interior no vacío.

Por otro lado, dado que se trabajará sobre una malla τ admisible de tetraedros, si $T \in \tau$ se notará como $\mathcal{F}_T$ al conjunto de caras del tetraedro T. Y además, de forma análoga a la notación ya utilizada anteriormente, se tendrá

$$\mathcal{F}_\tau = \bigcup_{T \in \tau} \mathcal{F}_T$$

Sea $F \in \mathcal{F}_\tau$ se definen $\mathcal{N}_F$ y $\mathcal{E}_F$, como el conjunto de vertices de F y el conjunto de aristas de F respectivamente.

La idea de admisibilidad para las mallas de tetraedros se extiende de manera natural a lo establecido en (1.1.1) de la siguiente manera:

Definición 4.2.1 *Sea τ un conjunto de tetraedros cerrados tal que $\cup_{T\in\tau} T = \overline{\Omega}$, se dirá que τ es admisible si, $\forall\ T \in \tau$, $\forall F \in \mathcal{F}_T$, se cumple que o bien $F \in \partial\Omega$, o existe $T' \in \tau$, $T' \neq T$, tal que $F \in \mathcal{F}_{T'}$.*

A cada tetraedro de la malla se le asociará una estructura de datos particular que en lo siguiente se llamará tetraedro marcado. Dicha estructura, análoga a la ya presentada estructura de triángulo marcado, es la unidad básica sobre la cual opera el algoritmo de bisección, y se detalla a continuación:

Definición 4.2.2 *Sea τ una malla de tetraedros y sea $T \in \tau$, se llamará tetraedro marcado asociado a T a la estructura de datos definida como la 4-upla $(\mathcal{N}_T, r_T, (m_F)_{F\in\mathcal{F}_T}, f_T)$, donde $r_T \in \mathcal{E}_T$ es la arista de refinamiento, $m_F \in \mathcal{E}_F$ con $F \in \mathcal{F}_T$ es la arista marcada de la cara F, con $m_F = r_T$ si $r_T \subset F$, y $f_T \in \{0, 1\}$.*

Cada arista marcada que no coincida con la arista de refinamiento, es o bien adyacente u opuesta a r_T. En base a esto puede clasificarse a los tetraedros marcados en los siguientes tipos:

- Tipo P, planar: Las aristas marcadas son coplanares. El tipo P será a su vez clasificado en los sub-tipos P_f y P_u, dependiendo de si $f_T = 1$ o $f_T = 0$, respectivamente.
- Tipo A, adyacente: Las aristas marcadas intersecan a la arista de refinamiento, pero no son coplanares.
- Tipo O, opuesto: Las aristas marcadas que no son de refinamiento no intersecan a la arista de refinamiento. En este caso, sólo un par de aristas opuestas están marcadas en el tetraedro. Una como la arista de refinamiento, y la otra como la arista marcada de las caras que no contienen a r_T.

- Tipo M, mixto. Sólo una de las aristas marcadas que no son de refinamiento interseca la arista de refinamiento.

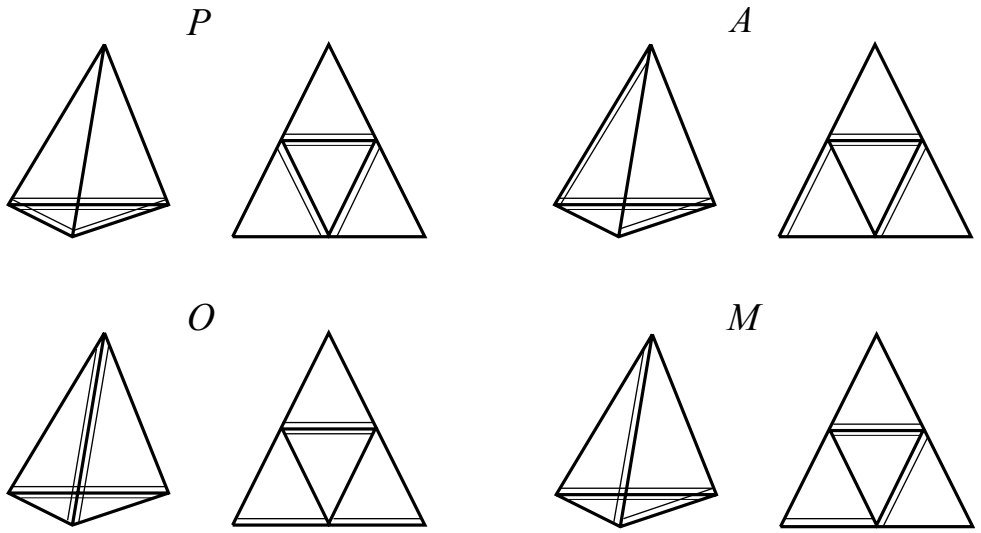


Figura 2: *Aquí se exponen los cuatro tipos de tetraedros, la arista de refinamiento está denotada por una doble línea.*

Cuando un tetraedro T es bisectado dando como resultado los hijos T_1 y T_2, una cara $F \in \mathcal{F}_{T_i}$ se la llamará "heredada" si $F \in \mathcal{F}_T$, "cortada" si existe $F' \in \mathcal{F}_T$ tal que $F \subsetneqq F'$, y "nueva" cuando no es ninguna de las anteriores. Cada uno de los hijos tendrá exactamente una cara heredada, dos cortadas, y una nueva, la cual, a su vez, la tendrán en común.

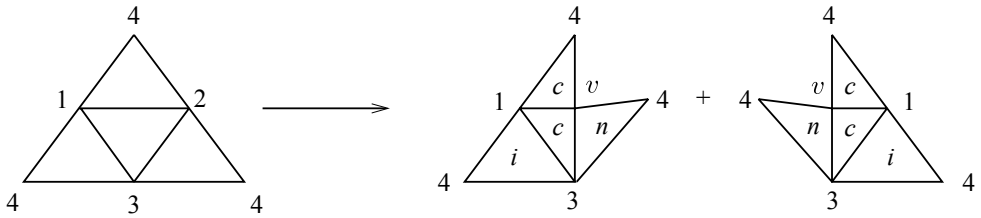


Figura 3: *Bisección de un tetraedro dando como resultado un nuevo vértice v, caras cortadas denotadas con c, caras heredadas denotadas con i, y una cara nueva denotada con n.*

A continuación se expone la función `BisectTet`.

ALGORITMO $\{T_1, T_2\} =$ `BisectTet`(T)
entrada: Un tetraedro marcado T.
salida: Dos tetraedros marcados T_1 y T_2.

1. Bisectar T uniendo el punto medio de r_T con cada uno de los vértices que no son extremos de r_T .

Marcar a los hijos de la siguiente manera:

2. La cara heredada, hereda la arista marcada del padre, y ésta se convierte en la arista de refinamiento del hijo.

3. En las caras cortadas, se marca la arista opuesta al nuevo vértice con respecto a la cara.

4. La cara nueva se marca de la misma manera para ambos hijos. Si el padre es de tipo P_f, se marca la arista que conecta el nuevo vértice con la nueva arista de refinamiento. De no ser así, se marca la arista opuesta al nuevo vértice.

5. $f_{T_i} = 1$ si y solo si T es de tipo P_u.

El efecto de la función anterior sobre los distintos tipos de tetraedros puede verse en la siguiente figura:

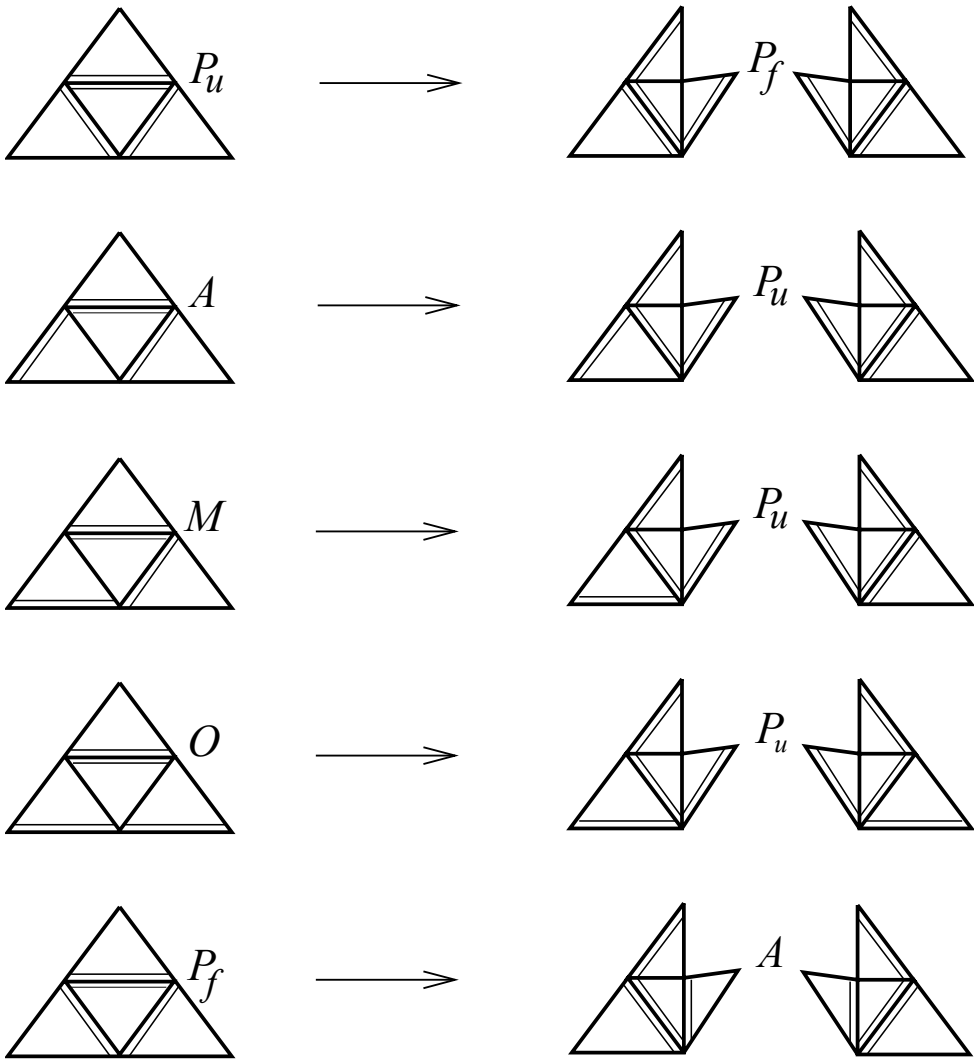


Figura 4: *A la izquierda el tetraedro de entrada y su tipo, a la derecha los hijos obtenidos aplicando* `BisectTet`.

`BisectTet` es la función básica utilizada en el algoritmo de refinamiento. Para poder describir el algoritmo general, primero es necesario aclarar una par de aspectos relevantes.

Por empezar, el concepto de nodo colgado que se utilizará es completamente análogo al ya definido en (2.1.2). Por otro lado, el algoritmo de refinamiento partirá de una malla admisible τ de tetraedros marcados, en donde, además, $f_T = 0, \forall T \in \tau$.

A continuación, y de forma muy parecida a lo ya descrito en el Capítulo 2, se detallan las

funciones principales del algoritmo de refinamiento.

ALGORITMO $\tau' = \texttt{Refine}(\tau, \mathcal{S})$
entrada: Una malla admisible marcada τ y $\mathcal{S} \subset \tau$.
salida: Una malla admisible marcada τ'.
1.$\overline{\tau} = \texttt{BisectTets}(\tau, \mathcal{S})$
2.$\tau' = \texttt{RefineToConformity}(\overline{\tau})$

El primer paso del algoritmo resulta trivial, ya que simplemente hay que bisectar cada triángulo contenido en $\mathcal{S}$. Con lo cual la función `BisectarTris` puede definirse de la siguiente forma.

$$\texttt{BisectTets}(\tau, \mathcal{S}) = (\tau \setminus \mathcal{S}) \bigcup_{T \in \mathcal{S}} \texttt{BisectTet}(T)$$

En el segundo paso del algoritmo, para obtener la admisibilidad de la malla, se continúa refinando tanto como sea necesario. A tal efecto se utiliza la función `RefineToConformity`, la cual se describe a continuación.

ALGORITMO $\tau' = \texttt{RefineToConformity}(\tau)$
entrada: Una malla marcada τ que sea salida de `BisectTets`.
salida: Una malla admisible marcada τ'.
1. Fijar $\mathcal{S} = \{T \in \tau$ tal que T posee un nodo colgado $\}$
2. Si $\mathcal{S} \neq \emptyset$ hacer
$\overline{\tau} = \texttt{BisectTets}(\tau, \mathcal{S})$
$\tau' = \texttt{RefineToConformity}(\overline{\tau})$
3. Si no
$\tau' = \tau$

Dado que no es evidente que la recursión en `RefineToConformity` termine, es necesario probar que el algoritmo para. A tal efecto, en [5], se demuestra el siguiente teorema.

Teorema 4.2.1 *Sea τ_0 una malla admisible, marcada, de manera tal que $f_T = 0 \quad \forall T \in \tau_0$. Para $k = 0, 1, ...$ se elige $\mathcal{S}_k \subseteq \tau_k$ de forma arbitraria, y se define $\tau_{k+1} = \texttt{Refine}(\tau_k, \mathcal{S}_k)$. Luego, para cada k, la aplicación de* `RefineToConformity` *dentro de la función* `Refine` *termina, produciendo una malla admisible de tetraedros marcados, siendo cada tetraedro en τ_k es de generación a lo sumo $3k$. Más aún, si en τ_k la máxima generación de un elemento es menor que $3m$, para algún entero m, entonces la máxima generación de un elemento en τ_{k+1} es menor o igual a $3m$.*

Otro resultado que será de importancia a la hora de probar el buen funcionamiento del algoritmo bajo métricas anisotrópicas, expuesto y demostrado también en [5], tiene que ver con la cantidad de posibles clases distintas de elementos que pueden aparecer al aplicar repetidas veces la función `Refine` sobre un elemento fijo.

A tal efecto, se definirá una relación de equivalencia entre los tetraedros.

Definición 4.2.3 *Sean T y T' dos tetraedros. Se dirá que son equivalentes si existe $F : \mathbb{R}^3 \to \mathbb{R}^3$ transformación afín, la cual se puede descomponer como $F = D \circ L$, con D una dilatación y L una traslación, tal que $F(T) = T'$.*

Uno de los puntos fuertes de este algoritmo de bisección es que, no importa cuántas veces se aplique la función `BisectTet` a un elemento y a sus hijos, todos los elementos obtenidos pertenecerán a una cantidad finita de clases de equivalencia. Para ver esto, en [5] demuestran el siguiente resultado.

Teorema 4.2.2 *Aplicando una cantidad arbitraria de veces la función* `BisectTet` *sobre un tetraedro marcado y los respectivos tetraedros obtenidos, solo es posible obtener representantes de, a lo sumo, 72 clases distintas de equivalencia.*

Es importante aclarar que en [5] este resultado no aparece enunciado de esta forma. Para demostrar lo anterior primero se establece un relación entre la forma de bisectar descrita en `BisectTet` y un método de bisección propuesto en [12], el cual está pensado para aplicarse en n-símplices y es referido en [5] como `BisectSimplex`. Dicha relación consiste en observar que, cuando $n = 3$, `BisectSimplex` y `BisectTet` son en esencia equivalentes, siempre y cuando se restrinja su utilización a tetraedros marcados de tipo P y A.

Se continúa, entonces, demostrando que la cantidad de clases de equivalencia obtenidas aplicando repetidas veces `BisectSimplex` sobre un n-simplex es a lo sumo $nn!2^{n-2}$. Con lo cual, todavía bajo la restricción de aplicar `BisectTet` a tetraedros de tipo P y A, la máxima cantidad de clases de equivalencia representadas por los tetraedros obtenidos es a lo sumo de 36. Y, finalmente, partiendo de la observación que al bisectar tetraedros de tipo O y M sus hijos son de tipo P, la cota se eleva a al doble, deduciéndose de forma inmediata el teorema enunciado anteriormente.

4.3. Bisección como transformación afín

En este punto ya se puede dar una idea un poco más precisa del objetivo de este capítulo. Si T_p es un tetraedro marcado y T se obtuvo de haber aplicado repetidas veces la función `BisectTet` sobre T_p, el objetivo será ver que existe una constante $C > 0$, que no depende ni de T_p ni de T ni de $\mathcal{M}$, de forma tal que

$$Def(T, \mathcal{M}) \leq C \cdot Def(T_p, \mathcal{M})$$

A tal efecto, en esta sección, se intentará establecer una relación entre una transformación afín aplicada sobre un tetraedro y el efecto de `BisectTet` aplicada sobre una estructura de tetraedro marcado asociada.

Dado que en esta sección será importante separar los conceptos de tetraedro como conjunto de $\mathbb{R}^3$ y el de tetraedro marcado, el primer concepto se notará de la misma manera que se viene haciendo hasta el momento, mientras que el otro, el de estructura de tetraedro marcado, se notará de manera explícita.

Para lograr establecer una relación útil entre los mencionados conceptos, será necesario interpretar la operación de bisección como una transformación afín. Más precisamente se tiene la siguiente definición:

Definición 4.3.1 *Sea $B : \mathbb{R}^3 \to \mathbb{R}^3$ una transformación afín, y sea $T = cc(x_1, x_2, x_3, x_4)$ un tetraedro. Se dirá que B es una bisección de T, en el nodo x_j con $j \in \{1, ..., 4\}$, sobre la arista $\overline{x_j x_l}$ con $l \in \{1, .., 4\}$, $l \neq j$, si se cumple que*

$$B(x_i) = x_i \quad \forall i \in \{1, ..., 4\}, i \neq j$$

y además

$$B(x_j) = \frac{x_j + x_l}{2}$$

A continuación se expone un resultado que permitirá relaciónar operaciones de bisección bajo transformaciones afines.

Proposición 4.3.1 *Sea $T = cc(x_1, x_2, x_3, x_4)$, sea $D : \mathbb{R}^3 \to \mathbb{R}^3$ transformación afín, de la forma $D(x) = D \cdot x + k$, con $k \in \mathbb{R}^3$, tal que $D(T)$ es un tetraedro no degenerado. Si B es una bisección de T, en el nodo $x_j \in \mathcal{N}_T$ sobre la arista $E = \overline{x_j x_l} \in \mathcal{E}_T$ con $j \neq l$, entonces la transformación $\hat{B}$, definida como $\hat{B} = D \circ B \circ D^{-1}$ es una bisección de $D(T)$, en el punto $D(x_j)$, sobre el lado $D(E) = \overline{D(x_j)D(x_l)} \in \mathcal{E}_{D(T)}$.*

Demostración:

En función de la Definición (4.3.1), se cumple lo siguiente:

$$B(x_i) = x_i \quad \forall i \in \{1, ..., 4\}, i \neq j \tag{3.24}$$

Y además

$$B(x_j) = \frac{x_j + x_l}{2} \tag{3.25}$$

Para ver que $\hat{B}$ es una bisección de $D(T)$, en el punto $D(x_j)$, sobre $D(E)$, debe cumplirse que:

$$\hat{B}(D(x_i)) = D(x_i) \quad \forall i \in \{1, ..., 4\}, i \neq j \tag{3.26}$$

Y

$$\hat{B}(D(x_j)) = \frac{D(x_j) + D(x_l)}{2} \tag{3.27}$$

Para ver (3.26), se tiene el siguiente desarrollo, $\forall i \in \{1, ..., 4\}, i \neq j$:

$$\hat{B}(D(x_i)) = D \circ B \circ D^{-1}(D(x_i)) = D \circ B(x_i) = D(x_i) \tag{3.28}$$

Donde para la primera igualdad se utiliza la definición de $\hat{B}$ y para la tercera lo establecido en (3.24).

Para ver que se cumple (3.27), se procede de la siguiente manera:

$$\hat{B}(D(x_j)) = D \circ B \circ D^{-1}(D(x_j)) = D \circ B(x_j) = D(\frac{x_j + x_l}{2}) \tag{3.29}$$

Donde en la primera igualdad se utilizó la definición de $\hat{B}$ y en la tercera se usó (3.25). Ahora, como se supone $D(x) = D \cdot x + k$, se tiene que

$$D(\frac{x_j + x_l}{2}) = \frac{D \cdot x_j}{2} + \frac{D \cdot x_l}{2} + k = \frac{D \cdot x_j}{2} + \frac{k}{2} + \frac{D \cdot x_l}{2} + \frac{k}{2} = \frac{D(x_j) + D(x_l)}{2} \tag{3.30}$$

Luego, de (3.30) y (3.29) se sigue (3.27). ■

Dado que interesa ver qué es lo que sucede con un elemento que se obtuvo aplicando varias veces `BisectTet` a un elemento padre, será necesario establecer una relación clara y precisa entre la aplicación de `BisectTet` y la aplicación de una bisección vista como transformación afín. A

tal fin, se define a continuación una modificación auxiliar de la función `BisectTet`:

ALGORITMO $T' = \texttt{BisectTet*}(T, x)$
entrada: Un tetraedro marcado T y un punto $x \in r_T \cap \mathcal{N}_T$.
salida: Un tetraedro marcado T'.
1. Fijar $\{T_1, T_2\} = \texttt{BisectTet}(T)$
2. Definir $T' \in \{T_1, T_2\}$ como aquel que cumple $x \notin \mathcal{N}_{T'}$.

Queda claro que un elemento obtenido aplicando repetidas veces `BisectTet*` sobre un elemento padre, se puede obtener aplicando repetidas veces `BisectTet` sobre el mismo elemento padre, y viceversa.

El objetivo de esta modificación es, por un lado, facilitar el enunciado y demostración de los sucesivos resultados de esta sección, y por otro, dar una mayor presición y rigurosidad sobre lo que se quiere expresar.

Es fácil ver que, tanto por la definición de `BisectTet` como por la de `BisectTet*`, si $T' = \texttt{BisectTet*}(T, x)$ con $x \in r_T \cap \mathcal{N}_T$, entonces si B es una bisección de T, en el nodo x, sobre la arista r_T, se tiene $B(T) \doteq T'$, donde el símbolo "$\doteq$" se utilizará entre una estructura de tetraedro marcado y un tetraedro, y querrá decir, por ejemplo en este caso, que la estructura T' es una estructura de tetraedro marcado asociada al tetraedro $B(T)$.

Resulta de importancia observar que si T es un tetraedro y $(\{x_1, x_2, x_3, x_4\}, r_T, (m_F)_{F\in\mathcal{F}_T}, f_T)$ es una estructura de tetraedro marcado asociada a T, la información que utiliza `BisectTet*` de las últimas tres componentes de la estructura no depende de la posición de los nodos.

Más precisamente, dada la estructura anterior, puede asociársele de manera equivalente la estructura $(\{x_1, x_2, x_3, x_4\}, r_T, (m_F)_{F\in\mathcal{F}_T}, f_T)^*$, donde en este contexto, la única diferencia será que las aristas se entenderán como un conjunto de dos índices, indicando sus extremos, y las caras serán un conjunto de tres índices indicando sus vértices. Así, por ejemplo, si $r_T = \overline{x_i x_j} \in \mathcal{E}_\tau$ en la estructura que se definió primero, se tendrá $r_T = \{i, j\}$ en la nueva estructura.

Para facilitar la escritura, en el contexto de la nueva interpretación de estructura de tetraedro marcado, se llamará $I = \{r_T, (m_F)_{F\in\mathcal{F}_T}, f_T\}$ a la configuración de marcas y, haciendo uso de esto, se notará

$$(\{x_1, x_2, x_3, x_4\}, r_T, (m_F)_{F\in\mathcal{F}_T}, f_T)^* := (\{x_1, x_2, x_3, x_4\}, I)^*$$

Queda claro que las dos estructuras son equivalentes, dado que una contiene toda la información para reconstruir la otra, y viceversa.

Además, por construcción de las funciones `BisectTet` y `BisectTet*`, se tiene que para fijar las marcas de un elemento hijo al aplicar `BisectTet` a un tetraedro marcado, la información del valor de los nodos no se utiliza. Con lo cual, la asignación de marcas es independiente de las posiciones de los vértices del tetraedro.

Con lo observado hasta el momento se tiene la siguiente proposición:

Proposición 4.3.2 *Sea T un tetraedro no degenerado y sea $D : \mathbb{R}^3 \to \mathbb{R}^3$ una transformación afín, tal que $D(T)$ es no degenerado. Si se considera la estructura $(\{x_1, ..., x_4\}, I)^*$ de tetraedro marcado asociado a T definida más arriba y se llama*

$$(\{\hat{x}_1, ..., \hat{x}_4\}, \hat{I})^* := \texttt{BisectTet}*((\{x_1, .., x_4\}, I)^*, x_i)$$

para algún $i \in \{1, ..., 4\}$, se afirma que:

$$\texttt{BisectTet}*((\{D(x_1), ..., D(x_4)\}, I)^*, D(x_i)) = (\{D(\hat{x}_1), ..., D(\hat{x}_4)\}, \hat{I})^* \tag{3.31}$$

Demostración:

Se deduce de la observación anterior sobre la independencia de I de la posición de los nodos y del hecho que `BisectTet*` aplica una bisección sobre el tetraedro $cc(x_1, ...x_4)$, cuyo comportamiento bajo una transformación afín es descrito en (4.3.1). ■

Como el objetivo es ver el comportamiento de la deformidad de un elemento a lo largo de sucesivas aplicaciones de `BisectTet*`, La siguiente proposición y las siguientes definiciones resultarán de utilidad.

Proposición 4.3.3 *Sea T un tetraedro no degenerado, y sea $(\{x_1, .., x_4\}, I)^*$ su estructura asociada de tetraedro marcado. Sea $D : \mathbb{R}^3 \to \mathbb{R}^3$ una transformación afín. Se define*

$$(\{x_1^0, .., x_4^0\}, I_0)^* := (\{x_1, .., x_4\}, I)^*$$

Y, recursivamente, para $j \geq 1$, se define:

$$(\{x_1^j, .., x_4^j\}, I_j)^* := \texttt{BisectTet}*((\{x_1^{j-1}, .., x_4^{j-1}\}, I_j)^*, x^{j-1}) \tag{3.32}$$

Con x^{j-1} contenido en la arista de refinamiento de $(\{x_1^{j-1}, .., x_4^{j-1}\}, I_{j-1})^*$ para $j \geq 1$. Se afirma que, $\forall j \geq 1$:

$$(\{D(x_1^j), .., D(x_4^j)\}, I_j)^* =$$

$$\texttt{BisectTet}*(D(x^{j-1})) \circ ... \circ \texttt{BisectTet}*((\{D(x_1^0), .., D(x_4^0)\}, I_0)^*, D(x^0)) \tag{3.33}$$

Donde

$$\texttt{BisectTet}*(x') \circ \texttt{BisectTet}*((\{x_1, ..., x_4\}, I)^*, x) :=$$

$$\texttt{BisectTet}*(\texttt{BisectTet}*((\{x_1, ..., x_4\}, I)^*, x)), x')$$

Demostración:

A partir de (3.32) y de la Proposición (4.3.2) se deduce que:

$$(\{D(x_1^j), .., D(x_4^j)\}, I_j)^* = \texttt{BisectTet}*((\{D(x_1^{j-1}), .., D(x_4^{j-1})\}, I_{j-1})^*, D(x^{j-1})) \tag{3.34}$$

De donde se sigue (3.33), aplicando un argumento recursivo.
■

Definición 4.3.2 *Sea T un tetraedro no degenerado. Sea $B : \mathbb{R}^3 \to \mathbb{R}^3$ una transformación afín. Se dirá que B es una sucesión de bisecciones de T si verifica que $B = B_{n-1} \circ ... \circ B_0$. Donde, llamando $T_0 := T$ y $T_j := B_{j-1} \circ ... \circ B_0(T_0)$ para $j > 0$, se tiene que B_j (con $0 \leq j \leq n-1$) es una bisección de T_j en el nodo $x^j \in \mathcal{N}_{T_j}$, sobre el lado $E_j \in \mathcal{E}_{T_j}$ con $x^j \in \mathcal{N}_{E_j}$, para alguna elección adecuada de los x^j y E_j.*

Definición 4.3.3 *Sea T un tetraedro no degenerado y sea B una sucesión de bisecciones $B = B_{n-1} \circ ... \circ B_0$ de T. Se dirá que B es una sucesión de bisecciones válida de T, si existe una estructura de tetraedro marcado $(\{x_1, ..., x_4\}, I)^*$ asociada y una elección adecuada de $x^j, 0 \leq j \leq n-1$ de forma tal que $B(T) \doteq \texttt{BisectTet}*(x^{n-1}) \circ ... \circ \texttt{BisectTet}*((\{x_1, ..., x_4\}, I)^*, x^0)$.*

Fijados los conceptos anteriores, se intentará ver a continuación de qué manera se comporta una sucesión válida de bisecciones a la cual se le aplica una transformación afín arbitraria.

Más precisamente, se tiene la siguiente proposición:

Proposición 4.3.4 *Sea B una sucesión de bisecciones válida sobre T, con T tetraedro no degenerado. Sea $D : \mathbb{R}^3 \to \mathbb{R}^3$ transformación afín, tal que $D(T)$ es un tetraedro no degenerado. Si se define $\hat{B} := D \circ B \circ D^{-1}$, entonces $\hat{B}$ es una sucesión de bisecciones válida sobre $D(T)$.*

Demostración:

Hay que ver primero que $\hat{B}$ es una sucesión de bisecciones. A tal efecto se observa que, en función de la Definición (4.3.2), $B = B_{n-1} \circ ... \circ B_0$, donde si se nota $T_0 := T$ y $T_j := B_{j-1} \circ ... \circ B_0(T)$ para $j > 0$, se tiene que B_j es una bisección de T_j, en el nodo $x^j \in \mathcal{N}_{T_j}$, sobre la arista $E_j \in \mathcal{E}_{T_j}$, con $x_j \in E_j$, para cierta elección de los x^j y E_j. Por la definición de $\hat{B}$ se tiene que:

$$\hat{B} = D \circ B \circ D^{-1} = D \circ B_{n-1} \circ ... \circ B_0 \circ D^{-1} = D \circ B_{n-1} \circ D^{-1} \circ ... \circ D \circ B_0 \circ D^{-1} \quad (3.35)$$

Si se define, para $0 \leq j \leq n-1$, $\hat{B}_j := D \circ B_j \circ D^{-1}$, de la Proposición (4.3.1) se deduce que $\hat{B}_j$ es una bisección de $D(T_j)$, en el nodo $D(x^j)$, sobre el lado $D(E_j)$. Además, de (3.35) se sigue que:

$$\hat{B} = \hat{B}_{n-1} \circ ... \circ \hat{B}_0 \quad (3.36)$$

Por otro lado, teniendo en cuenta la definición de T_j, y observando que $B_j = D^{-1} \circ \hat{B}_j \circ D$, se tienen las siguientes igualdades:

$$D(T_j) = D \circ B_{j-1} \circ ... \circ B_0(T) = \hat{B}_{j-1} \circ ... \circ \hat{B}_0 \circ D(T) \quad (3.37)$$

Luego, de (3.36) y (3.37) se sigue que $\hat{B}$ es una sucesión de bisecciones de $D(T)$.

Queda por ver que $\hat{B}$ es válida. Es decir, que existe una estructura de tetraedro marcado $(\{D(x_1), ..., D(x_4)\}, I)^*$ asociada a $D(T)$ y una elección adecuada de nodos x^j, $0 \leq j \leq n-1$, de forma tal que

$$\hat{B}(D(T)) \doteq \texttt{BisectTet}*(x^{n-1}) \circ ... \circ \texttt{BisectTet}*((\{D(x_1), ..., D(x_4)\}, I)^*, x^0)$$

En efecto, dado que B es una bisección válida, existe una estructura de tetraedro marcado $(\{x_1, ..., x_4\}, I)^*$ asociada a T y una elección de nodos x^j, $0 \leq j \leq n-1$ de forma tal que

$$B(T) \doteq \texttt{BisectTet}*(x^{n-1}) \circ ... \circ \texttt{BisectTet}*((\{x_1, ..., x_4\}, I)^*, x^0)$$

Considerando, entonces, esa configuración de marcas I, y esa elección de nodos x^j, se le asocia a $D(T)$ la siguiente estructura: $(\{D(x_1), ..., D(x_4)\}, I)^*$. Es decir, las mismas marcas de

la estructura que sirve para T, pero con los nodos transformados por D. Luego, se afirma lo siguiente:

$$D(T_j) \doteq \texttt{BisectTet}*(D(x^{j-1})) \circ ... \circ \texttt{BisectTet}*((\{D(x_1), ..., D(x_4)\}, I)^*, D(x^0)) \tag{3.38}$$

Para ver que esto es cierto, se define $(\{x_1^0, ..., x_4^0\}, I_0)^* := (\{x_1, ..., x_4\}, I)^*$, y, recursivamente, se define, para $j \geq 1$:

$$(\{x_1^j, ..., x_4^j\}, I_j)^* := \texttt{BisectTet}*((\{x_1^{j-1}, ..., x_4^{j-1}\}, I_{j-1})^*, x^{j-1}) \tag{3.39}$$

Luego, combinando (3.39) con la Proposición (4.3.3) se sigue que, $\forall j \geq 1$:

$$(\{D(x_1^j), .., D(x_4^j)\}, I_j)^* =$$

$$\texttt{BisectTet}*(D(x^{j-1})) \circ ... \circ \texttt{BisectTet}*((\{D(x_1^0), .., D(x_4^0)\}, I_0)^*, D(x^0)) \tag{3.40}$$

Observando que:

$$(\{D(x_1^0), ..., D(x_4^0)\}, I_0)^* = (\{D(x_1), ..., D(x_4)\}, I)^*$$

se deduce (3.38). A continuación, haciendo $j = n$ en (3.38) se tiene que:

$$D(T_n) \doteq \texttt{BisectTet}*(D(x^{n-1})) \circ ... \circ \texttt{BisectTet}*((\{D(x_1), ..., D(x_4)\}, I)^*, D(x^0)) \tag{3.41}$$

Ahora bien, como $T_n = B(T)$ y $B = D^{-1} \circ \hat{B} \circ D$, se tiene que:

$$D(T_n) = D \circ B(T) = D \circ D^{-1} \circ \hat{B} \circ D(T) = \hat{B}(D(T)). \tag{3.42}$$

Finalmente, de (3.41) y (3.42) se sigue que $\hat{B}$ es una sucesión de bisecciones válida de $D(T)$. ■

4.4. Control de la deformidad para una métrica constante

Hasta aquí, con lo desarrollado en la sección anterior, se dispone herramientas y propiedades útiles para entender el efecto de la aplicación de una sucesión de bisecciones sobre un tetraedro. La importancia esto es que, si una estructura de tetraedro marcado asociada a T es obtenida habiendo aplicado repetidas veces la función `BisectTet*` sobre una estructura de tetraedro marcado asociada a un elemento padre T_p, entonces existe una sucesión válida de bisecciones B sobre T_p tal que $B(T_p) = T$.

Esta relación entre la aplicación del algoritmo `BisectTet*` y la transformación afín B será lo que permitirá controlar la deformidad de T en términos de deformidad del elemento padre T_p de forma explícita.

El objetivo será probar que si a un elemento se le aplica una sucesión válida de bisecciones, entonces la calidad del elemento obtenido es comparable, en algún sentido, a la del elemento original.

Si se logra esto, entonces se habrá demostrado que cualquier malla obtenida de la aplicación de la función `Refine`, tendrá una calidad comparable a la de la malla original. Esto es porque

para cada elemento nuevo producido por `Refine` existirá una sucesión válida de bisecciones sobre un determinado elemento padre que permita obtenerlo.

Dadas las proposiciones y definiciones de las secciones anteriores, se puede enunciar en este momento, de forma precisa, el resultado principal de esta sección, orientado al control de la deformidad. Todo esto, por el momento, para el caso de una métrica constante.

Se tiene lo siguiente:

Proposición 4.4.1 *Sea T_p un tetraedro no degenerado, sea $T = B(T_p)$, donde B es una sucesión de bisecciones válida sobre T_p y sea $\mathcal{M}$ una matriz simétrica, definida positiva. Luego, existe una constante $C > 0$ que depende sólo del elemento de referencia $\hat{T}$ y del algoritmo* `BisectTet`*, de forma tal que:*

$$Def(T, \mathcal{M}) \leq C \cdot Def(T_p, \mathcal{M}) \tag{4.43}$$

Demostración:

Se tiene que, por la definición de la función Def, y usando el hecho de que $T = B(T_p)$:

$$Def(T, \mathcal{M}) = Def(B(T_p), \mathcal{M}) = Cond_2(D_{\sqrt{\mathcal{M}} \circ B(T_p)}) \tag{4.44}$$

Por otro lado, si $T_p = cc(x_1, ..., x_4)$, se tiene:

$$\sqrt{\mathcal{M}} \circ B(T_p) = cc(\sqrt{\mathcal{M}} \circ B(x_1), ..., \sqrt{\mathcal{M}} \circ B(x_4))$$

Entonces, por definición, se tendrá que $D_{\sqrt{\mathcal{M}} \circ B(T_p)}$ es la transformación afín que verifica:

$$D_{\sqrt{\mathcal{M}} \circ B(T_p)} \circ \sqrt{\mathcal{M}} \circ B(x_i) = \hat{x}_i, \quad \forall i \in \{1, ..., 4\} \tag{4.45}$$

Con lo cual, por definición de D_{T_p} se tendrá que

$$D_{\sqrt{\mathcal{M}} \circ B(T_p)} \circ \sqrt{\mathcal{M}} \circ B = D_{T_p}$$

Lo que implica:

$$D_{\sqrt{\mathcal{M}} \circ B(T_p)} = D_{T_p} \circ B^{-1} \circ \sqrt{\mathcal{M}}^{-1} \tag{4.46}$$

Ahora bien, como B es una sucesión de bisecciones válida sobre T_p, por lo visto en la Proposición (4.3.4), junto con el hecho de que $D_{T_p}(T_p) = \hat{T}$ se deduce que $\hat{B} = D_{T_p} \circ B \circ D_{T_p}^{-1}$, es una sucesión válida de bisecciones sobre $\hat{T}$.

Luego, de la observación anterior, de (4.44) y (4.46) se sigue que:

$$Def(T, \mathcal{M}) = Cond_2(D_{T_p} \cdot B^{-1} \cdot \sqrt{\mathcal{M}}^{-1}) = Cond_2(D_{T_p} \cdot (D_{T_p}^{-1} \cdot \hat{B} \cdot D_{T_p})^{-1} \cdot \sqrt{\mathcal{M}}^{-1}) =$$

$$Cond_2(\hat{B}^{-1} \cdot D_{T_p} \cdot \sqrt{\mathcal{M}}^{-1}) \leq Cond_2(\hat{B}^{-1}) \cdot Cond_2(D_{T_p} \cdot \sqrt{\mathcal{M}}^{-1}) \tag{4.47}$$

Donde en la desigualdad del final se utiliza la propiedad submultiplicativa de la norma matricial.

En base a lo anterior, observando que $Cond_2(\hat{B}^{-1}) = Cond_2(\hat{B})$ y $Def(T_p, \mathcal{M}) = Cond_2(D_{T_p} \cdot \sqrt{\mathcal{M}}^{-1})$, se sigue:

$$Def(T, \mathcal{M}) \leq Cond_2(\hat{B}) \cdot Def(T_p, \mathcal{M}) \tag{4.48}$$

Como $\hat{B}$ es una sucesión válida de bisecciones sobre $\hat{T}$, entonces el tetraedro $\hat{B}(\hat{T})$ puede ser obtenido aplicando la función `BisectTet*` sobre $\hat{T}$. Con lo cual, en vista de lo establecido en el Teorema (4.2.2), $\hat{B}(\hat{T})$ pertenecerá a una de las finitas clases de equivalencia de tetraedros que pueden obtenerse aplicando `BisectTet` sobre $\hat{T}$. Dichas clases son las establecidas en la Definición (4.2.3).

Sea $N \leq 72$ la cantidad de clases de equivalencia que pueden alcanzarse sobre $\hat{T}$, y sea T_i un representante de la i-ésima clase, con $i \in \{1, ..., N\}$. Por lo observado anteriormente, existe $i \in \{1, ..., N\}$ de forma tal que $\hat{B}(\hat{T})$ pertenece a i-ésima clase de equivalencia.

Luego, de la definición de las clases, se sigue que existe $F : \mathbb{R}^3 \to \mathbb{R}^3$ transformación afín que puede descomponerse como $F = D \circ L$, con L una traslación y D una dilatación, de forma tal que $F(T_i) = \hat{B}(\hat{T})$.

Por otro lado, para cada $i \in \{1, ..., N\}$ se define A_i como la transformación afín que verifica $A_i(\hat{T}) = T_i$. Dado que los elementos no son degenerados, la transformación queda bien definida para todo i. Por lo anterior se tiene que vale $F \circ A_i(\hat{T}) = \hat{B}(\hat{T})$, y al ser $\hat{T}$ no degenerado se sigue que

$$F \circ A_i = \hat{B}$$

De donde se deduce que:

$$Cond(\hat{B}) = Cond_2(F \cdot A_i) \leq Cond_2(F) \cdot Cond_2(A_i) \tag{4.49}$$

Ahora bien, cualquiera sea la clase de equivalencia a la cual pertenece $\hat{B}(\hat{T})$, la transformación F definida en cada caso siempre cumplirá $Cond_2(F) = 1$, ya que siempre podrá ser pensada como una dilatación seguida de una traslación. Se define, entonces, la constante C de la siguiente manera:

$$C := \max_{i \in \{1,...,N\}} Cond_2(A_i) \tag{4.50}$$

La cual resulta estar bien definida al ser todos los tetraedros involucrados no degenerados. Luego, combinando (4.48), (4.49), (4.50) y el hecho de que $Cond_2(F) = 1$, se tiene:

$$Def(T, \mathcal{M}) \leq Cond_2(F) \cdot Cond_2(A_i) \cdot Def(T_p, \mathcal{M}) \leq C \cdot Def(T_p, \mathcal{M}) \tag{4.51}$$

Con lo cual queda demostrado el resultado. ■

4.5. Generalización al caso de una métrica no constante

Se ha probado, hasta el momento, que el algoritmo de bisección funciona adecuadamente bajo una métrica que puede ser anisotrópica, pero constante.

Puede observarse que para deducir el buen funcionamiento bajo $\mathcal{M}$ constante, no era necesario hacer el desarrollo de las secciones anteriores. Ya que, en vista de la finitud de clases de equivalencia, enunciado en el Teorema (4.2.2), combinada con el hecho de que al ser $\mathcal{M}$ constante, la regularidad de un elemento no cambia con una traslación ni con una dilatación, es posible deducir que el nivel de deformidad alcanzado por cualquier tetraedro obtenido de aplicar

`BisectTet` estará acotado por arriba y por debajo. Dado que todos los tetraedros pertenecientes a una misma clase de equivalencia tendrán un mismo nivel de regularidad.

Sin embargo, ese enfoque, a priori, no brinda demasiada información sobre la dependencia de la calidad de un elemento en función del elemento padre del cual se obtuvo.

Así y todo, de poderse establecer dicha dependencia, la principal ventaja del enfoque que se da en este trabajo es que, por un lado, brinda una manera bastante intuitiva de generalizar la medida de deformidad al caso no constante, y por otro, se puede apreciar de forma detallada cómo se estropea la cota que controla la deformidad a medida que a la métrica $\mathcal{M}$ se le da un mayor margen para variar (ver Proposición (4.5.2)).

A continuación se define la generalización de la función Def a métricas no constantes.

Definición 4.5.1 *Sea T un tetraedro, y sea $\mathcal{M} : T \to \mathbb{R}^{3\times 3}$ un tensor métrico. Se define $Def_{\mathcal{M}}(T)$ de la siguiente manera:*

$$Def_{\mathcal{M}}(T) := \sup_{x\in T} Def(T, \mathcal{M}(x))$$

Es decir que se considerará como deformidad al peor caso posible dentro de todos los valores que toma $\mathcal{M}$ en el elemento en cuestión.

Una pregunta válida a partir de la definición anterior, es por qué generalizar de esa manera la deformidad. ¿Qué significa que $Def_{\mathcal{M}}(T)$ sea cercana a 1?

Para fijar ideas, dado que $Def_{\mathcal{M}}$ es fácilmente generalizable a triángulos, podemos suponer T como un triángulo equilátero, de forma tal que uno de sus lados sea el segmento $[0, 1] \times \{0\}$, y esté contenido en el semi-espacio $y \geq 0$.

Supongamos ahora que se tiene el tensor métrico

$$\mathcal{M}(x, y) := (\alpha|y| + 1) \cdot I$$

Donde I denota la matriz identidad de $\mathbb{R}^{2\times 2}$.

A partir de esto, suponiendo α suficientemente grande, resulta que la medida de los lados de T no son iguales. De hecho, la longitud de $[0, 1] \times \{0\}$ es siempre 1, mientras que la longitud de los demás lados crece arbitrariamente a medida que crece α. Sin embargo $Def_{\mathcal{M}}(T) = 1$. Ya que puntualmente, salvo por un factor de escala, $\mathcal{M}$ es en esencia la matriz identidad.

Por otro lado, siguiendo la misma idea, no resultaría complicado construir una métrica $\mathcal{M}'$ de forma tal que cerca de los lados del triángulo T se tenga $\mathcal{M}' = I$, pero que en el interior de T, lejos del borde, $\mathcal{M}'$ tenga un comportamiento muy anisotrópico. Con lo cual en este caso se tendría que los lados de T miden 1 en $\mathcal{M}'$, y sin embargo $1 << Def_{\mathcal{M}'}(T)$.

A partir de lo anterior puede pensarse, en un principio, que la medida de deformidad no funciona de forma adecuada, ya que da 1 en un caso en donde los lados del triángulo son distintos, y da un número mucho mayor a 1 en un caso en donde los lados del triángulo son iguales.

Entonces, ¿Qué es lo que está midiendo $Def_{\mathcal{M}}$, y por qué resulta importante? Lo que mide $Def_{\mathcal{M}}(T)$ es cuán deforme puede ser un elemento similar a T, contenido en T, y de tamaño lo suficientemente pequeño como para poder considerar $\mathcal{M}$ constante. Esto es importante porque da mucha información sobre la calidad de los elementos que se obtendrán refinando a T, ya que por lo visto en secciones anteriores, el algoritmo de refinamiento utilizado en este trabajo genera una cantidad finita de clases de triángulos cuya forma es, en algún sentido, parecida a la de T.

Lo anterior sumado a la hipótesis de que el tensor métrico sea localmente constante (continuo en algún sentido), nos dice que si $Def_{\mathcal{M}}(T) \approx 1$, entonces es posible refinar a T una cantidad arbitraria de veces con la garantía de que, una vez que los elementos obtenidos sean lo suficientemente pequeños, entonces sus lados serán más o menos iguales.

Con lo cual, en el primer caso, si se refina adecuadamente hasta lograr elementos de tamaño similar (tamaño medido en la métrica $\mathcal{M}$), que a su vez sean lo suficientemente pequeños, es posible obtener una malla uniforme en $\mathcal{M}$ con todos sus elementos de buena calidad. Por otro lado, en el segundo caso, al refinar el triángulo T no hay garantía de que todos los elementos sean de buena calidad, ya que algunos estarán cerca de donde la métrica es anisotrópica y tendrán una forma comparable a la de T. Con lo cual serán deformes sin importar cuán pequeños sean.

A continuación se verá que si T es obtenido a partir de una sucesión de bisecciones válida aplicada sobre un elemento padre T_p, entonces existe entre $Def_{\mathcal{M}}(T)$ y $Def_{\mathcal{M}}(T_p)$ una relación análoga a la que se describe en la Proposición (4.4.1) entre $Def(T, \mathcal{M})$ y $Def(T_p, \mathcal{M})$.

Proposición 4.5.1 *Sea T_p un tetraedro no degenerado, sea B una sucesión válida de bisecciones sobre T_p y sea $T := B(T_p)$. Luego, si se tiene $\mathcal{M} : T_p \to \mathbb{R}^{2\times 2}$ tensor métrico, siendo C la constante definida en (4.50), vale la siguiente expresión:*

$$Def_{\mathcal{M}}(T) \leq C \cdot Def_{\mathcal{M}}(T_p) \tag{5.52}$$

Demostración:

Por definición de $Def_{\mathcal{M}}(T)$ se tiene que:

$$Def_{\mathcal{M}}(T) = \sup_{x \in T} Def(T, \mathcal{M}(x)) \tag{5.53}$$

Pero, al ser $T = B(T_p)$ con B una sucesión de bisecciones válida sobre T_p, en función de la Proposición (4.4.1), existe $C > 0$, que sólo depende de $\hat{T}$ y `BisectTet`, tal que $\forall x \in T$ vale:

$$Def(T, \mathcal{M}(x)) \leq C \cdot Def(T_p, \mathcal{M}(x)) \tag{5.54}$$

De (5.53), (5.54), y del hecho de que $T \subset T_p$ (ya que T se obtiene bisectando cierta cantidad de veces T_p) se sigue:

$$Def_{\mathcal{M}}(T) \leq \sup_{x \in T} C \cdot Def(T_p, \mathcal{M}(x)) \leq C \cdot \sup_{x \in T_p} Def(T_p, \mathcal{M}(x)) = C \cdot Def_{\mathcal{M}}(T_p) \tag{5.55}$$

Con lo cual queda demostrado el resultado. ■

Si bien el resultado anterior establece una relación razonable entre la deformidad del elemento padre y el hijo, calcular o al menos aproximar el supremo en la definición de $Def_{\mathcal{M}}$ puede resultar computacionalmente costoso.

En vista de lo anterior, uno puede preguntarse cuánto varía $Def_{\mathcal{M}}(T_p)$ si $\mathcal{M}$ no varía demasiado sobre T_p. Ya que, en tal caso, si $Def_{\mathcal{M}}(T_p)$ no varía demasiado, entonces debería tenerse $Def_{\mathcal{M}}(T_p) \approx Def(T_p, \mathcal{M}(x))$, para algún $x \in T_p$ arbitrario y, a su vez, se debería poder controlar la deformidad de cualquier elemento obtenido a partir del refinamiento de T_p con $Def(T_p, \mathcal{M}(x))$.

Esta cuestión se responde con la siguiente proposición:

Proposición 4.5.2 *Sea T_p un tetraedro no degenerado, sea $T := B(T_p)$, con B sucesión válida de bisecciones sobre T_p, y sea $\mathcal{M} : T_p \to \mathbb{R}^{3\times 3}$ tensor métrico. Si se llama b al baricentro de T_p, y $\mathcal{M}_b := \mathcal{M}(b)$ y siendo C es la constante definida en (4.50), se tiene que dado $\varepsilon > 0$ existe $\delta > 0$ tal que si $||\mathcal{M}(x) - \mathcal{M}_b||_2 < \delta$, $\forall x \in T_p$ entonces:*

$$Def_{\mathcal{M}}(T) \leq C \cdot Def(T_p, \mathcal{M}_b) + \varepsilon \tag{5.56}$$

Demostración:

De la Proposición (4.5.1), y del hecho de que $Def_{\mathcal{M}}$ y Def son siempre positivas, se tiene que

$$Def_{\mathcal{M}}(T) \leq C \cdot Def_{\mathcal{M}}(T_p) \leq C \cdot |Def_{\mathcal{M}}(T_p) - Def(T_p, \mathcal{M}_b)| + C \cdot Def(T_p, \mathcal{M}_b) \tag{5.57}$$

Con lo cual bastará ver que dado $\varepsilon > 0$ existe $\delta > 0$ de tal que si $||\mathcal{M}(x) - \mathcal{M}_b||_2 < \delta$, $\forall x \in T_p$ entonces :

$$|Def_{\mathcal{M}}(T_p) - Def(T_p, \mathcal{M}_b)| \leq \frac{\varepsilon}{C} \tag{5.58}$$

En efecto, por la definición de $Def_{\mathcal{M}}$ se tiene:

$$|Def_{\mathcal{M}}(T_p) - Def(T_p, \mathcal{M}_b)| = |\sup_{x\in T_p} Def(T_p, \mathcal{M}(x)) - Def(T_p, \mathcal{M}_b)| \leq$$

$$\leq \sup_{x\in T_p} |Def(T_p, \mathcal{M}(x)) - Def(T_p, \mathcal{M}_b)| \tag{5.59}$$

Fijando $x \in T_p$, puede hacerse el siguiente desarrollo:

$$|Def(T_p, \mathcal{M}(x)) - Def(T_p, \mathcal{M}_b)| = |Cond_2(\sqrt{\mathcal{M}}(x) \cdot D_{T_p}^{-1}) - Cond_2(\sqrt{\mathcal{M}}_b \cdot D_{T_p}^{-1})| \tag{5.60}$$

Ahora bien, sea $\mathbb{I}$ el conjunto de las matrices inversibles de $\mathbb{R}^{3\times 3}$. Dado que la función $Cond_2 : (\mathbb{I}, ||\cdot||_2) \to \mathbb{R}_{\geq 1}$ es continua, se sigue que existe $\delta_1 > 0$ tal que si $||\sqrt{\mathcal{M}}(x) \cdot D_{T_p}^{-1} - \sqrt{\mathcal{M}}_b \cdot D_{T_p}^{-1}||_2 < \delta_1$ entonces:

$$|Cond_2(\sqrt{\mathcal{M}}(x) \cdot D_{T_p}^{-1}) - Cond_2(\sqrt{\mathcal{M}}_b \cdot D_{T_p}^{-1})| \leq \frac{\varepsilon}{C} \tag{5.61}$$

Dado que:

$$||\sqrt{\mathcal{M}}(x) \cdot D_{T_p}^{-1} - \sqrt{\mathcal{M}}_b \cdot D_{T_p}^{-1}||_2 \leq ||\sqrt{\mathcal{M}}(x) - \sqrt{\mathcal{M}}_b||_2 \cdot ||D_{T_p}^{-1}||_2 \tag{5.62}$$

será necesario utilizar la continuidad de la función raíz cuadrada matricial sobre el conjunto de las matrices simétricas y definidas positivas. Es decir, siendo $\mathbb{S}$ el conjunto de las matrices simétricas y definidas positivas de $\mathbb{R}^{3\times 3}$, se sabe que $\sqrt{\cdot} : (\mathbb{S}, ||\cdot||_2) \to (\mathbb{S}, ||\cdot||_2)$ es una función continua. Con lo cual existe $\delta > 0$ tal que si $||\mathcal{M}(x) - \mathcal{M}_b||_2 < \delta$ entonces:

$$||\sqrt{\mathcal{M}}(x) - \sqrt{\mathcal{M}}_b||_2 \leq \frac{\delta_1}{||D_{T_p}^{-1}||_2} \tag{5.63}$$

Luego, tomando este δ, y en función de las ecuaciones de (5.60) hasta (5.63) se deduce que:

$$|Def(T_p, \mathcal{M}(x)) - Def(T_p, \mathcal{M}_b)| \leq \frac{\varepsilon}{C} \quad \forall x \in T_p \tag{5.64}$$

Finalmente de la expresión anterior, combinada con (5.59) se sigue el resultado que se quería demostrar. ■

Cabe mencionar que la utilización del baricentro es totalmente arbitraria, puede ser reemplazado por cualquier punto de T_p sin que pierda validez la demostración que se acaba de hacer.

Otro aspecto importante para observar es que, en función de la demostración anterior, si se quisiera hacer un análisis de la sensibilidad de la cota de la deformidad en función de la variación de $\mathcal{M}$ sobre T_p, ésta dependerá de la sensibilidad del número de condición y de la raíz matricial vistas como funciones en los respectivos espacios en donde se encuentran definidas.

Capítulo 5

Ejemplos y resultados numéricos

Es este capítulo se presentarán ejemplos del funcionamiento de la heurísticas descrita en los capítulos 2 y 3. Se mostrará, de forma empírica, que el algoritmo cumple el objetivo de generar mallas regulares sobre una determinada métrica $\mathcal{M}$ y, a su vez, la fase II funciona según lo desarrollado en el capítulo anterior.

Los ejemplos de este capítulo consistirán, por lo general, en intentar aproximar una cierta función f definida en un dominio Ω, utilizando interpolación lineal a partir de la malla generada por el algoritmo. Es decir, sea $\Omega \subset \mathbb{R}^2$ abierto no vacío, conexo, acotado y poligonal, y sea $f : \Omega \to \mathbb{R}$ tal que $f \in C^2(\Omega)$, si τ es una triangulación de Ω, entonces se define la interpolada de Lagrange de f, $\Pi f \in C(\Omega)$, como la única función continua que cumple ser lineal sobre cada $T \in \tau$, y además:

$$\Pi f(x) = f(x) \quad \forall x \in \mathcal{N}_\tau$$

El error de interpolación se calculará tanto en la norma del espacio $L^2(\Omega)$ como también en la seminorma de $H^1(\Omega)$. Dichos errores se computan de la siguiente forma, respectivamente:

$$||e||_{L^2} = ||f - \Pi f||_{L^2(\Omega)} = \left(\int_\Omega (f - \Pi f)^2 \right)^{\frac{1}{2}} \tag{0.1}$$

$$|e|_{H^1} = |f - \Pi f|_{H^1(\Omega)} = \left(\sum_{T \in \tau} \int_T \left(\frac{\partial f}{\partial x} - \frac{\partial \Pi f}{\partial x} \right)^2 + \sum_{T \in \tau} \int_T \left(\frac{\partial f}{\partial y} - \frac{\partial \Pi f}{\partial y} \right)^2 \right)^{\frac{1}{2}} \tag{0.2}$$

Un aspecto importante a tener en cuenta será la elección de la métrica que se utilizará en dichos ejemplos. Dado que el objetivo de este trabajo está centrado en la construcción de una heurística que genera una malla teniendo a la métrica como un dato de entrada, los detalles sobre cómo construir métricas adecuadas para diferentes propósitos no serán tratados en profundidad.

Un ejemplo de lo mencionado anteriormente puede encontrarse en [13], en donde W. Huang expone una métrica en $\mathbb{R}^n$ basada en el hessiano, en la cual, según se demuestra, la regularidad de una malla de n-símplices redunda en una disminución del error de interpolación, cuando se interpola con polinomios de grado arbitrario.

A efecto de exponer de forma sencilla el funcionamiento de la heurística en las siguientes secciones, se optó por utilizar como métrica una pequeña modificación del hessiano de la función que se busca aproximar. Es decir que, teniendo f como antes, si $\mathcal{H}(x)$, con $x \in \Omega$, es el hessiano de f en el punto x, se tomará como métrica el tensor $\mathcal{M} : \Omega \to R^{2\times 2}$ definido como

$$\mathcal{M}(x) := |\mathcal{H}(x) + \varepsilon \cdot I| \quad \forall x \in \Omega \tag{0.3}$$

Donde, si $A \in \mathbb{R}^{2\times 2}$ es tal que A^2 es s.d.p. , se tiene $|A| := \sqrt{A^2}$, y, además, se nota I a la matriz identidad, y se toma $\varepsilon = 0{,}000001$.

Vale aclarar que en esencia se está utilizando la hessiano de f como métrica. Se le suma $\varepsilon \cdot I$ y se toma el módulo matricial para asegurar que lo que se obtenga sea una matriz s.d.p. , para cada $x \in \Omega$.

Esta métrica es prácticamente la misma utilizada en [2]. Por lo general, cuando se busca optimizar la relación error-costo computacional se suele utilizar métricas basadas en el hessiano de la función que se busca aproximar, bastante parecidas a la de la expresión (0.3).

Cabe mencionar que algunas veces, se puede pedir a la malla regularidad sobre $\mathcal{M}$ para lograr objetivos diferentes al de optimizar la relación error-costo computacional. Un ejemplo de esto es [15], en donde X. Li y W. Huang construyen una métrica apropiada sobre la cual cierta regularidad de la malla asegura el cumplimiento del principio del máximo discreto, para el caso de la ecuación $\nabla(\mathcal{D}\nabla u) = f$ sobre un dominio Ω, con $\mathcal{D} = \mathcal{D}(x)$ la matriz de difusión.

La heurística (tanto la fase I como la II) fue implementada en `C++`. Para los ejemplos de este capítulo se utilizó $iter = 20$, $iter_s = 3$, $iter_r = 1$, salvo se indique lo contrario, y fueron probados en una máquina con un procesador AMD Athlon(tm) II X2 245 de 2,90 GHz y 2,00 GB de RAM.

5.1. `FaseI` sobre una métrica anisotrópica

En esta sección se mostrará el funcionamiento de la función `FaseI` sobre un dominio $\Omega = [-1, 1]^2$, bajo una métrica anisotrópica no constante. La métrica en cuestión se define en base a la ecuación (0.3), siendo f para este ejemplo :

$$f(x, y) = \arctan(10 \cdot (y - 0{,}75 \cdot \sin \pi x)) \tag{1.4}$$

La malla inicial que se utilizó para este ejemplo puede verse en la figura 1. Partiendo de dicha malla, en las figuras de la 2 a la 4 puede verse la malla que `FaseI` entrega como salida.

En la descripción de cada imagen se encuentran los datos sobre el nivel de refinamiento R que fue utilizado, así como también datos sobre la regularidad de la malla, error y tiempo de CPU necesario para su generación. Los datos de regularidad y uniformidad están expresados en la media m de las medidas de todos los lados (medidos en la métrica $\mathcal{M}$) y su respectivo desvío estándar σ. Es decir, siendo τ la triangulación de salida podrán verse como datos en la descripción de la imagen:

$$m := \frac{1}{\#\mathcal{E}_\tau} \sum_{E\in\mathcal{E}_\tau} |E|_{\mathcal{M}} \qquad \sigma^2 := \frac{1}{\#\mathcal{E}_\tau} \sum_{E\in\mathcal{E}_\tau} (m - |E|_{\mathcal{M}})^2 \tag{1.5}$$

Si bien no es una medida de regularidad demasiado precisa, permite ver por separado la uniformidad y la regularidad. Es decir que, un desvío estándar bajo implicaría una malla bastante regular (aunque al revés no sea necesariamente cierto). Mientras que una media cercana a uno (combinada con un desvío estándar bajo) estaría diciendo que la malla se bastante uniforme.

En los ejemplos presentados se verá que la media es cercana a uno mientras que el desvío estándar es bajo.

Para los ejemplos de este capítulo se utilizó $iter = 20$, $iter_s = 3$, $iter_r = 1$, salvo se indique lo contrario.

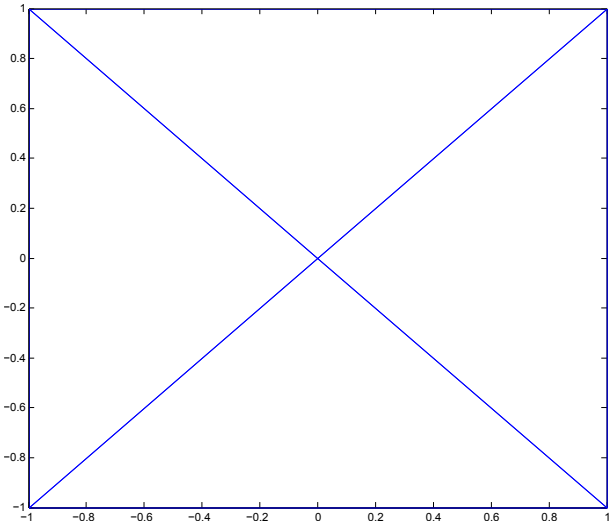


Figura 1: *Malla inicial que toma como entrada la función* `FaseI`

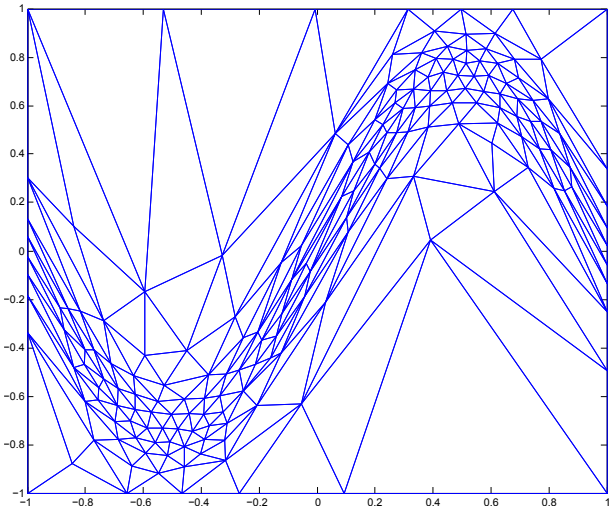


Figura 2: *Salida de* `FaseI` *con* $R = 0,5$, $m = 0{,}9808$, $\sigma = 0{,}1495$. $||e||_{L^2} = 0{,}0562$, $|e|_{H^1} = 1{,}8853$. *Cantidad de elementos: 338. Tiempo de CPU: 1,06 s.*

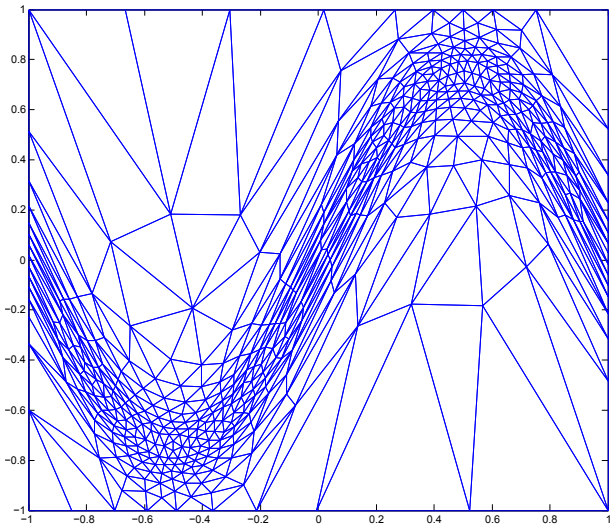


Figura 3: *Salida de* `FaseI` *con* $R = 0,3$, $m = 0{,}9384$, $\sigma = 0,1334$. $||e||_{L^2} = 0,0205$, $|e|_{H^1} = 1,0482$. *Cantidad de elementos: 938. Tiempo de CPU: 1,75 s.*

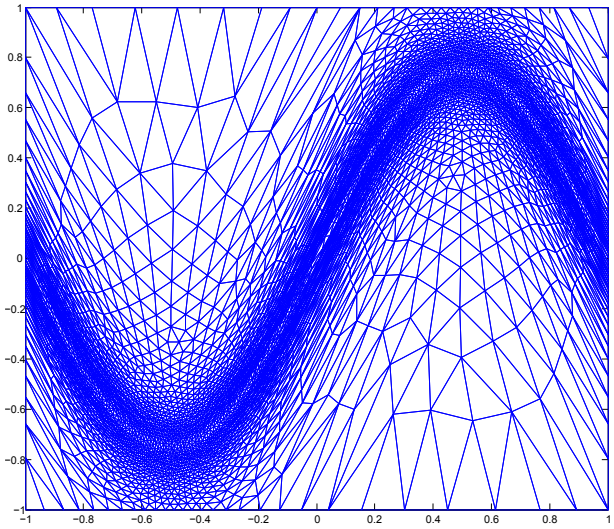


Figura 4: *Salida de* `FaseI` *con* $R = 0,1$, $m = 0,9317$, $\sigma = 0,09841$. $||e||_{L^2} = 0,0022$, $|e|_{H^1} = 0,3343$. *Cantidad de elementos: 8165. Tiempo de CPU: 8,08 s.*

5.2. `RefinarUniforme` aplicada al caso anterior

En esta sección se ejemplificará el funcionamiento de la fase II del algoritmo. A tal fin se utilizará como entrada la malla de la figura 2 y se probará la función `RefinarUniforme` para distintos niveles de refinamiento.

El buen funcionamiento de `RefinarUniforme` está garantizado por lo probado en el capítulo anterior. Para ver de forma empírica lo demostrado, se estimó la deformidad de la malla para cada nivel de refinamiento, observándose que la deformidad se mantiene más o menos constante conforme se avanza.

Para medir la deformidad de la malla, si se tiene τ la malla de salida de `RefinarUniforme`, se estimó el promedio de $Def(T, \mathcal{M}_b(T))$, siendo $T \in \tau$ y $b(T)$ su baricentro. Es decir, teniendo τ como antes, se define:

$$\overline{Def}_\tau := \frac{1}{\#\tau} \cdot \sum_{T \in \tau} Def(T, \mathcal{M}_{b(T)}) \tag{2.6}$$

A continuación puede verse la salida de la función `RefinarUniforme` para distintos valores del parámetro G (especificado al pie de cada figura). En la figura 8 puede verse la deformidad promedio en función de G. Se observa una tendencia a ser constante, lo cual es esperable en función de lo probado en el capítulo anterior.

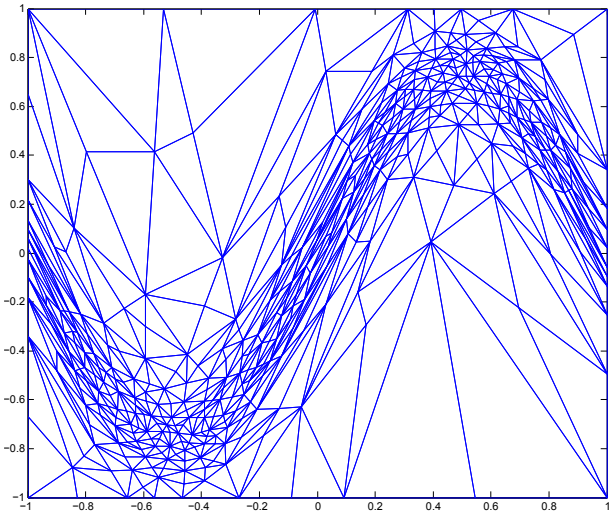


Figura 5: *Salida de* `RefinarUniforme` *para* $G = 1$. $||e||_{L^2} = 0,0293$, $|e|_{H^1} = 1,46$. *Cantidad de elementos: 815. Tiempo de CPU: 0,003 s.*

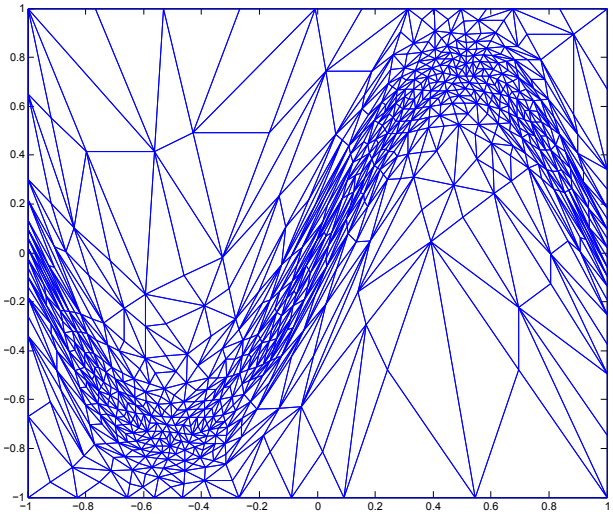


Figura 6: *Salida de* `RefinarUniforme` *para* $G = 2$. $||e||_{L^2} = 0,0188$, $|e|_{H^1} = 1,204$. *Cantidad de elementos: 1352. Tiempo de CPU: 0,007 s.*

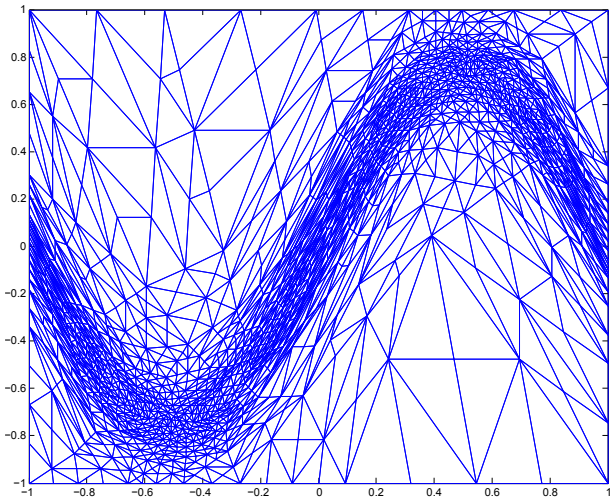


Figura 7: *Salida de* `RefinarUniforme` *para* $G = 3$. $||e||_{L^2} = 0,0094$, $|e|_{H^1} = 0,8165$. *Cantidad de elementos: 2982. Tiempo de CPU: 0,015 s.*

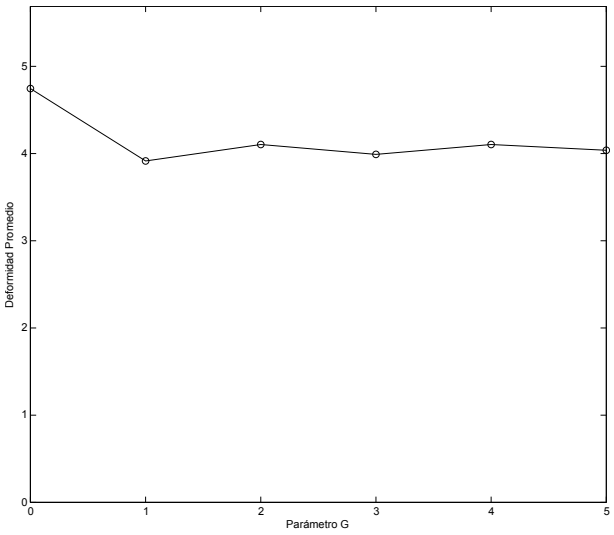


Figura 8: *Deformidad promedio, definida en 2.6, en función del parámetro G.*

5.3. `FaseI` en el caso isotrópico

Dado que gran cantidad de veces se utilizan mallas isotrópicas, y no es demasiado fácil apreciar a simple vista la calidad de los elementos generados sobre métrica extrañas, puede resultar de interés ver cómo funciona la heurística en el caso isotrópico.

Como la métrica es prácticamente el hessiano de la función, puede observarse que en las zonas del dominio en donde el hessiano es cercano a cero los elementos tienden a ser más grandes. Este efecto es justamente lo que se espera tener, dado que de esta forma se obtiene una buena representación de la función haciendo un uso inteligente de los recursos computacionales, que en este viene dado por la cantidad de elementos.

Este efecto, sin embargo, trae un problema cuando se quiere generar mallas como las anteriores pero sobre dominios más complicados. Esto se debe a que un borde complicado requiere un nivel alto de refinamiento para poder ser bien aproximado, lo cual puede no suceder en zonas del borde donde el hessiano de la función es cercano a cero.

Esta cuestión puede ser resuelta de varias maneras. Por ejemplo en el mallador anisotrópico BAMG [14], F. Hecht utiliza una longitud máxima permitida para los lados, donde dicha longitud está medida en la métrica euclídea. Esto hace que en las zonas en donde la métrica es cercana a cero los elementos no sean desproporcionadamente grandes, lo cual dota a la malla de una mejor capacidad de adaptación a bordes complicados, teniendo en estos casos una métrica localmente isotrópica adaptándose a un borde complicado.

Es por esto que también resulta de interés observar el comportamiento de `FaseI` en el caso isotrópico con geometrías complicadas.

Para ver su funcionamiento en el caso isotrópico basta con darle como entrada la métrica $\mathcal{M} \equiv I$, con I la matriz identidad de $\mathbb{R}^{2\times 2}$.

A continuación se exponen algunas mallas generadas con el algoritmo para diversos casos.

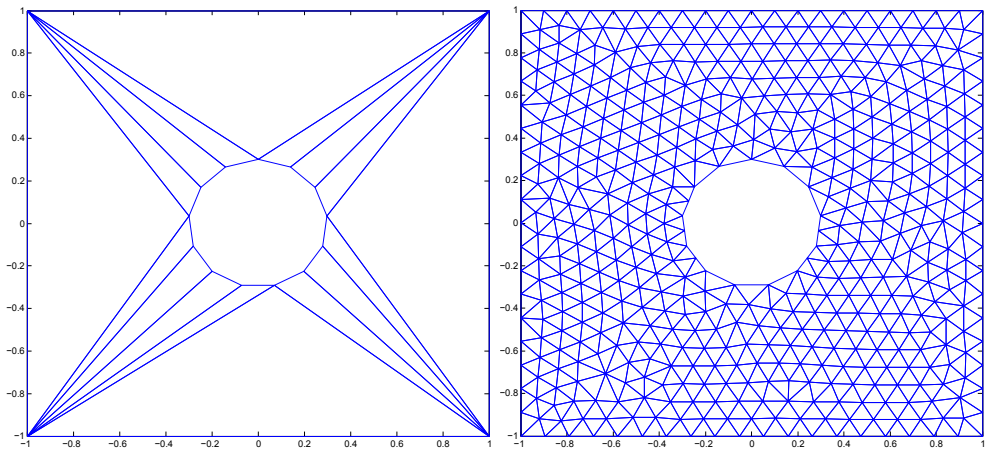


Figura 9: *A la izquierda la malla de entrada que recibe* `FaseI` *y a la derecha la salida, con* $\mathcal{M} = I$.

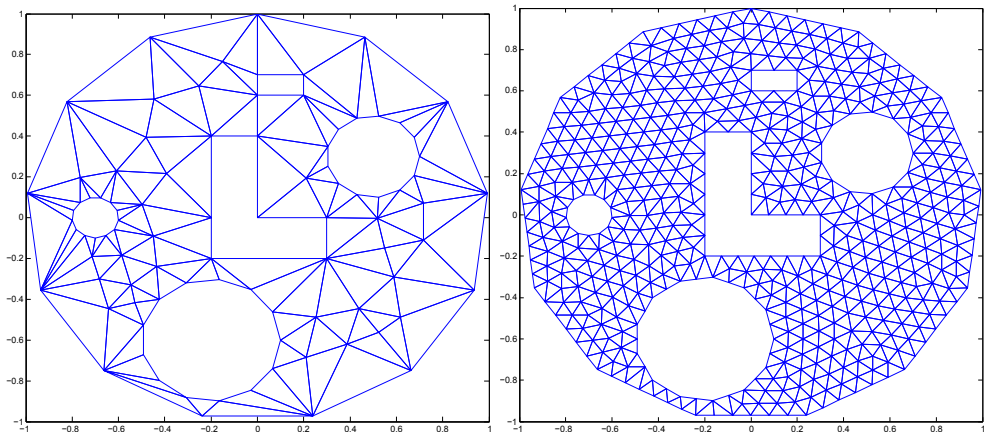


Figura 10: *A la izquierda la malla de entrada que recibe* `FaseI` *y a la derecha la salida, con* $\mathcal{M} = I$.

5.4. Error de interpolación

En esta sección se analizará la evolución del error de interpolación, definido en (0.1), en función del nivel de refinamiento.

Se tomaron en cuenta tres casos distintos y se estimó su error de interpolación tanto en la norma de $L^2(\Omega)$ como en la semi-norma de $H^1(\Omega)$ (ver (0.1) y (0.2)), los cuales se encontrarán expresados en función de la cantidad de elementos.

Como ya se dijo, se mostrarán tres ejemplos, para los cuales se usaron las siguientes funciones:

$$f_1(x,y) = \arctan(50 \cdot (y - 0{,}75 \cdot \sin \pi x)) + x^2 + y^2$$

$$f_2(x,y) = \frac{x^2}{2}$$

$$f_3(x,y) = \arctan(10 \cdot (x+1))$$

El primer caso es muy parecido al ya analizado en secciones anteriores, con la salvedad que presenta una anisotropía más fuerte. Se le sumó x^2+y^2 para evitar el problema de tener elementos grandes en donde el hessiano de $\arctan(50 \cdot (y - 0{,}75 \cdot \sin \pi x))$ es pequeño. El segundo caso es bastante simple, pero con anisotropía muy pronunciada. Por último, el tercer caso corresponde a la aproximación de una capa límite sobre un borde recto.

En función de lo establecido más arriba, siendo $\mathcal{H}_i$ el hessiano de la función f_i, se definen los tensores métricos de la siguiente forma:

$$\mathcal{M}_i(x) := |\mathcal{H}_i(x) + \varepsilon \cdot I| \quad \forall x \in \Omega_i \tag{4.7}$$

Donde Ω_i es el dominio que se utiliza en cada ejemplo.

Para f_1 y f_3 se utilizó la malla de la figura 1 como entrada, mientras que para el caso f_2 se utilizó la malla que puede verse a continuación.

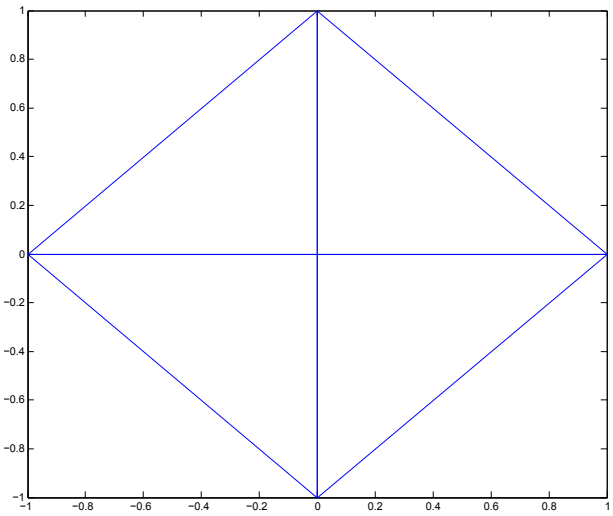


Figura 11: *Entrada para el segundo ejemplo.*

Para tener una idea de la eficacia de la fase II del algoritmo, en los gráficos del error se mostrará paralelamente el error de interpolación en función de la cantidad de elementos para

las mallas generadas por la función `RefinarUniforme` y `RefinarUniforme*`, partiendo siempre de la malla más gruesa generada por `FaseI` en cada caso.

Los gráficos del error se exponen en escala logarítmica. Dado que en esta escala se observa que el gráfico correspondiente a los errores de las mallas generadas por las tres funciones aproximan una recta, en la descripción de cada gráfico se exponen las pendientes aproximadas de éstas, calculadas mediante cuadrados mínimos (para el cómputo se omitieron los primeros datos con el objetivo de captar su comportamiento asintótico).

Adicionalmente se muestra el tiempo de CPU empleado para la generación de las mallas en función de la cantidad de elementos. A los tiempos correspondientes a las mallas generadas por `RefinarUniforme` y `RefinarUniforme*` se le suma el tiempo empleado para generar la malla de partida, para la cual se utilizó `FaseI`.

En todo los casos, se utilizó `FaseI` con $iter = 25$, $iter_s = 3$, e $iter_r = 1$. A continuación se exponen los resultados obtenidos.

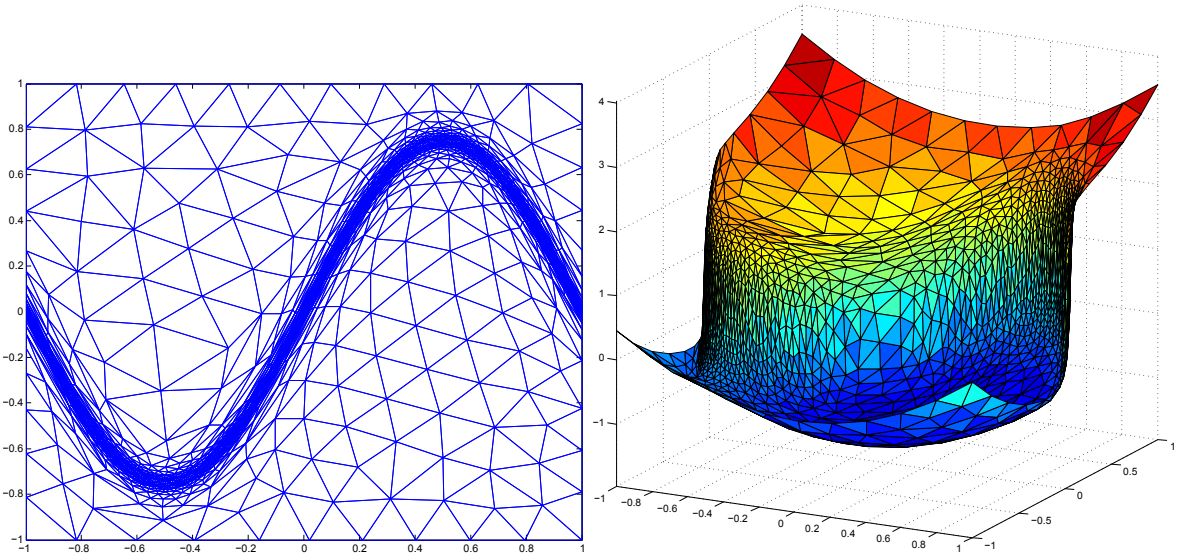


Figura 12: *A la izquierda la malla generada por* `FaseI` *para un nivel de refinamiento* $R = 0{,}3$ *bajo la métrica* $\mathcal{M}_1$*. A la derecha puede verse el gráfico de* Πf_1 *en función de la malla de la izquierda. Cantidad de elementos: 2375. Tiempo de CPU: 4,01 s.*

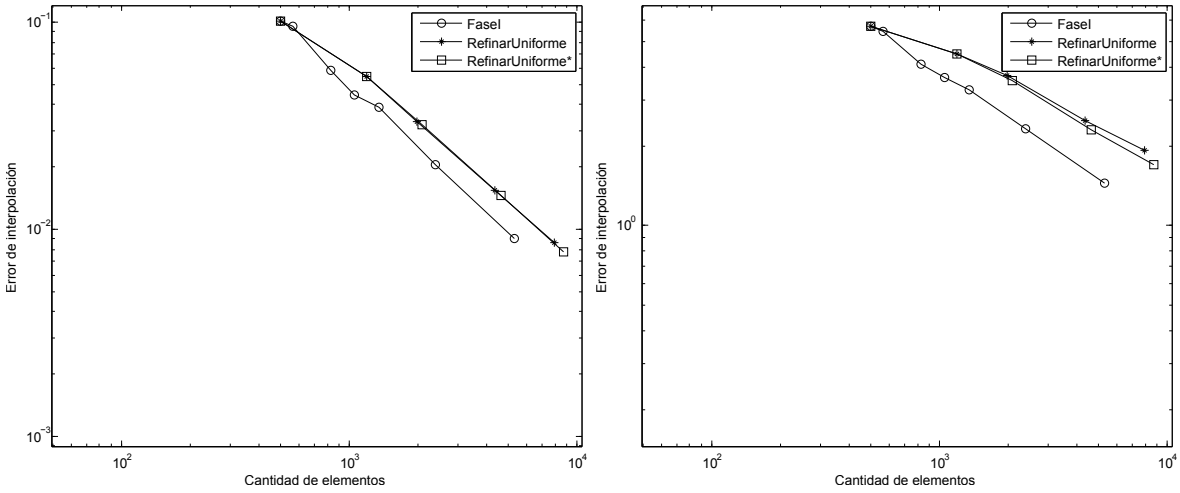


Figura 13: *Error de interpolación,* $||e||_{L^2}$ *a la izquierda y* $|e|_{H^1}$ *a la derecha, en función de la cantidad de elementos, usando* `FaseI`, `RefinarUniforme` *y* `RefinarUniforme*` *con la métrica* $\mathcal{M}_1$*. Las pendientes aproximadas son, para* `FaseI`*: -1,0308 en* $||\cdot||_{L^2}$ *y -0,5778 en* $|\cdot|_{H^1}$ *. Para* `RefinarUniforme`*: -0,9683 en* $||\cdot||_{L^2}$ *y -0,4514 en* $|\cdot|_{H^1}$*. Y para* `RefinarUniforme*`*: -0,9784 en* $||\cdot||_{L^2}$ *y -0,4938 en* $|\cdot|_{H^1}$*.*

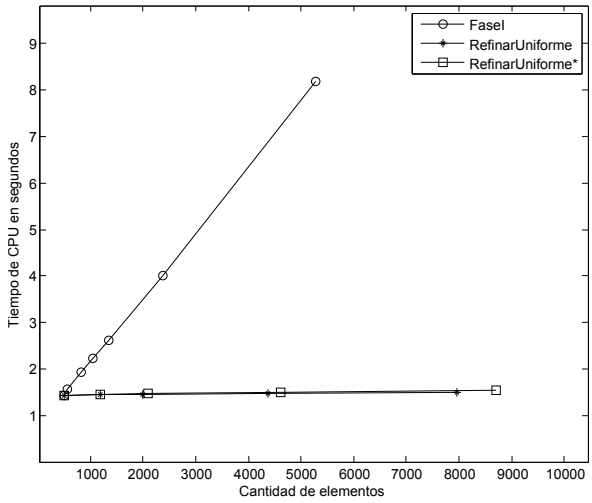


Figura 14: *Tiempo de CPU empleado en la generación de la malla en función de la cantidad de elementos, usando* $\mathcal{M}_1$*.*

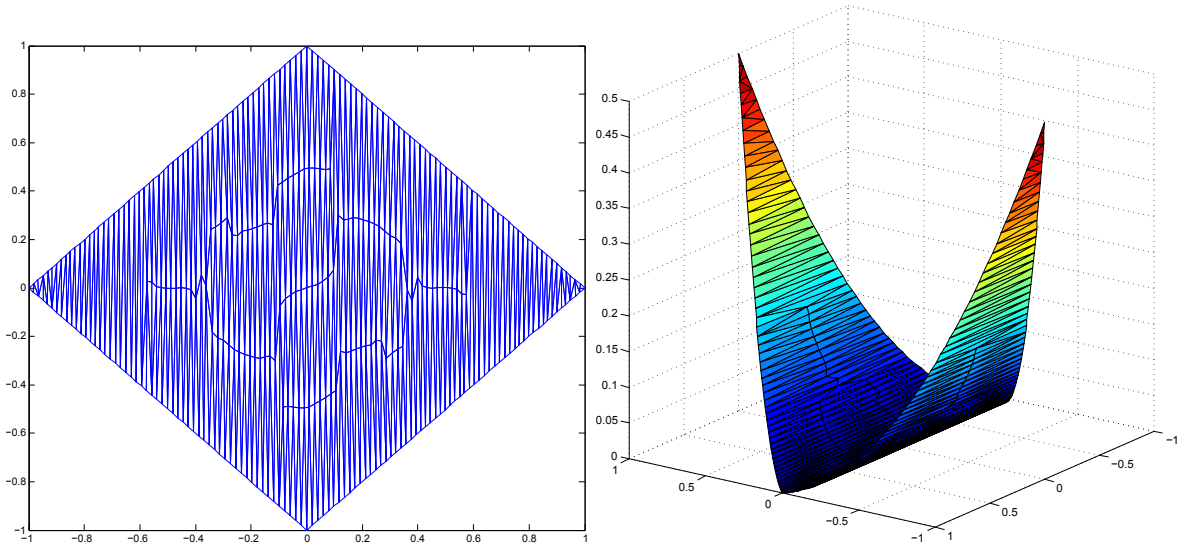


Figura 15: *A la izquierda la malla generada por* `FaseI` *para un nivel de refinamiento* $R = 0{,}02$*, bajo la métrica* $\mathcal{M}_2$*. A la derecha puede verse el gráfico de* Πf_2 *en función de la malla de la izquierda. Cantidad de elementos: 440. Tiempo de CPU: 1,00 s.*

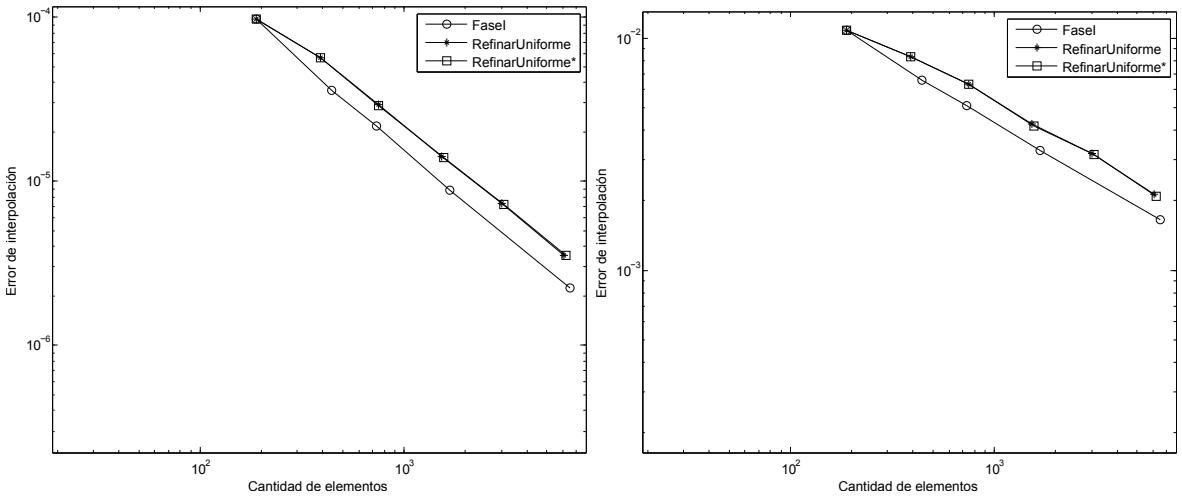


Figura 16: *Error de interpolación,* $||e||_{L^2}$ *a la izquierda y* $|e|_{H^1}$ *a la derecha, en función de la cantidad de elementos, usando* `FaseI`*,* `RefinarUniforme` *y* `RefinarUniforme*` *con la métrica* $\mathcal{M}_2$*. Las pendientes aproximadas son, para* `FaseI`*: -1.0624 en* $||\cdot||_{L^2}$ *y -0.5299 en* $|\cdot|_{H^1}$ *. Para* `RefinarUniforme`*: -1.0056 en* $||\cdot||_{L^2}$ *y -0.4978 en* $|\cdot|_{H^1}$*. Y para* `RefinarUniforme*`*: -1.0015 en* $||\cdot||_{L^2}$ *y -0.4991 en* $|\cdot|_{H^1}$*.*

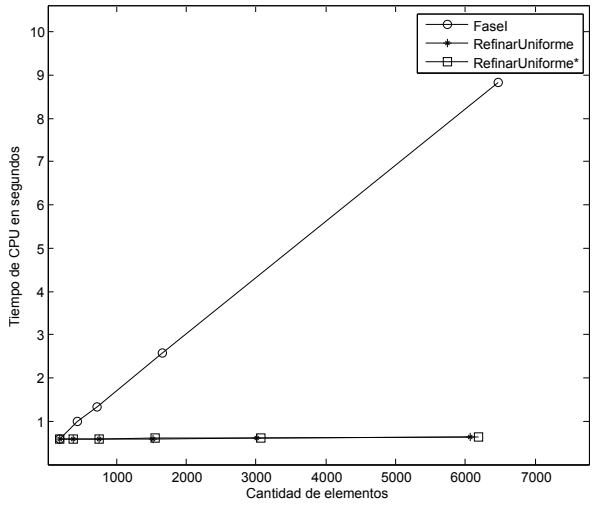


Figura 17: *Tiempo de CPU empleado en la generación de la malla en función de la cantidad de elementos, usando* $\mathcal{M}_2$.

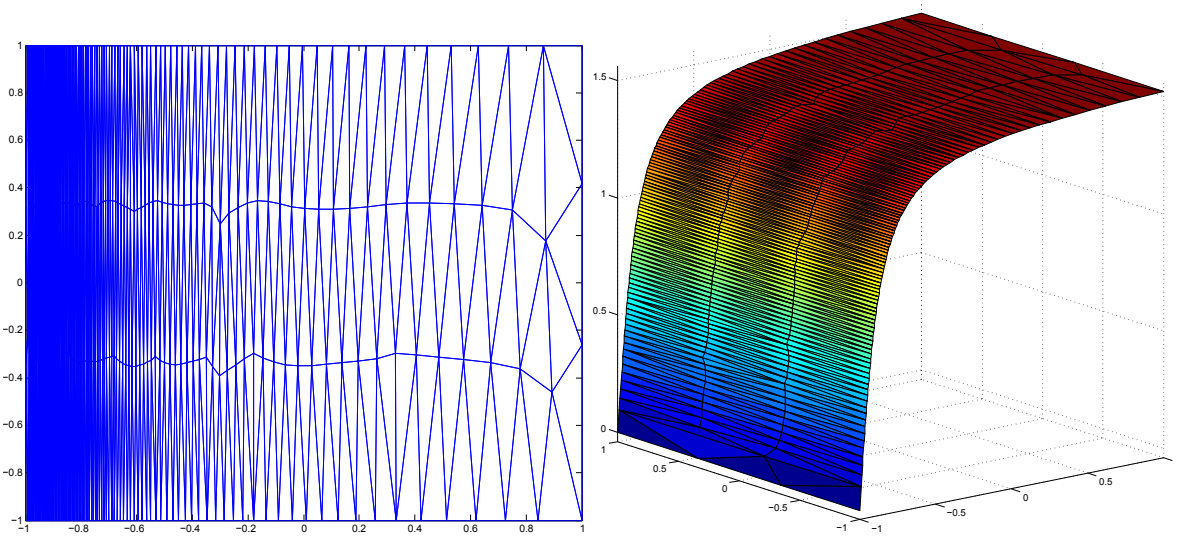


Figura 18: *A la izquierda la malla generada por* `FaseI` *para un nivel de refinamiento* $R = 0{,}025$*, bajo la métrica* $\mathcal{M}_3$*. A la derecha puede verse el gráfico de* Πf_3 *en función de la malla de la izquierda. Cantidad de elementos: 666. Tiempo de CPU: 1,818 s.*

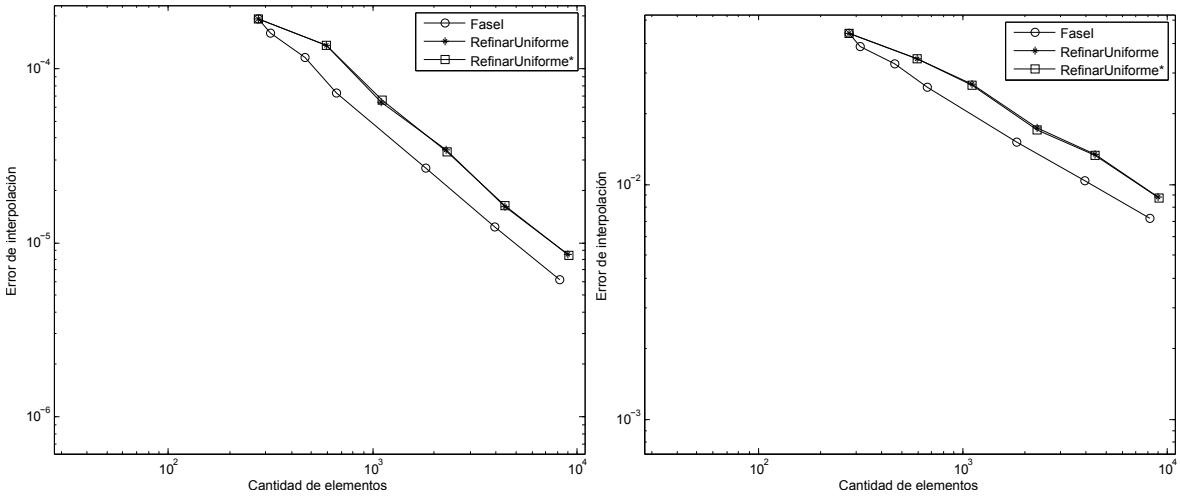


Figura 19: *Error de interpolación,* $||e||_{L^2}$ *a la izquierda y* $|e|_{H^1}$ *a la derecha, en función de la cantidad de elementos, usando* `FaseI`, `RefinarUniforme` *y* `RefinarUniforme*` *con la métrica* $\mathcal{M}_3$*. Las pendientes aproximadas son, para* `FaseI`*: -1.0156 en* $||\cdot||_{L^2}$ *y -0.5312 en* $|\cdot|_{H^1}$ *. Para* `RefinarUniforme`*: -1.0096 en* $||\cdot||_{L^2}$ *y -0.4973 en* $|\cdot|_{H^1}$*. Y para* `RefinarUniforme*`*: -1.0177 en* $||\cdot||_{L^2}$ *y -0.5014 en* $|\cdot|_{H^1}$*.*

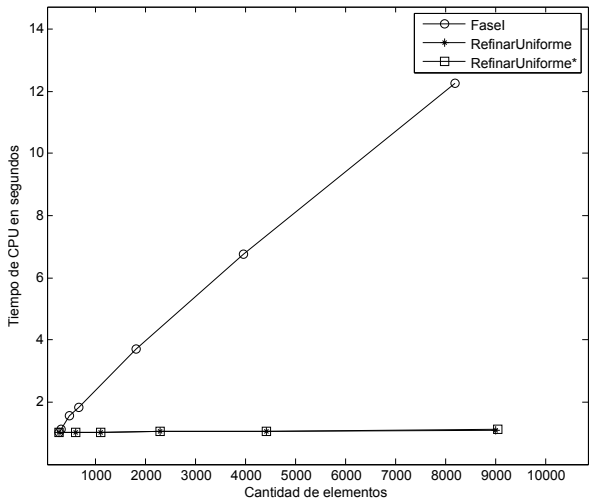


Figura 20: *Tiempo de CPU empleado en la generación de la malla en función de la cantidad de elementos, usando* $\mathcal{M}_3$*.*

De lo anterior se puede observar que las pendientes del error medido en $||\cdot||_{L^2}$ es aproximadamente -1 cuando se utiliza `FaseI`, y es levemente mayor cuando se utiliza `FaseI` para armar una malla inicial y se continúa refinando con `RefinarUniforme` y `RefinarUniforme*`. A su vez, el error medido en $|\cdot|_{H^1}$ presenta una pendiente aproximada de -1/2 cuando se utiliza sólo `FaseI`, y es levemente mayor cuando se utiliza `RefinarUniforme` y `RefinarUniforme*`. Además, puede verse que `RefinarUniforme*` tiene un mejor desempeño que `RefinarUniforme` en el caso en que el error de la malla inicial es más grande, mientras que en los otros casos ambas funciones tiene un desempeño similar.

En cuanto al tiempo de CPU empleado para la generación de las mallas, puede observarse un comportamiento lineal en la cantidad de elementos en todos los casos. El tiempo de generación utilizando una malla inicial hecha con `FaseI` y refinando luego con `RefinarUniforme` o `RefinarUniforme*` es notablemente menor al empleado utilizando solamente `FaseI`.

5.5. Comparación `FaseI` con BAMG

En esta sección se exponen algunos ejemplos más del funcionamiento `FaseI`, a la vez que se compara con mallas generadas por el mallador anisotrópico BAMG [14], utilizando la misma función a aproximar en cada caso, con una cantidad similar de elementos. Se observó que tanto la calidad (en términos de $\overline{Def}_\tau$) como los niveles de error $||e||_{L^2}$ y $|e|_{H^1}$ medidos para las mallas generadas con ambos algoritmos son bastante parecidos.

Los tiempos de CPU empleados por ambos algoritmos para la generación de las mallas fue bastante similar, siempre que se utilizaran una cantidad también similar de iteraciones.

Adicionalmente se observó que en los casos donde el hessiano presenta cambios muy bruscos, el algoritmo de Hecht genera mallas de mejor calidad para niveles bajos de refinamiento. Esto puede deberse, entre otras cosas, a la manera en que se construye la métrica que utiliza el BAMG, la cual difiere a la utilizada en este trabajo.

En los ejemplos se exponen los histogramas de la cantidad de elementos en función de $Def(T, \mathcal{M}_{b(T)})$ (deformidad en función de la métrica evaluada en el baricentro) y el ángulo máximo (medido en la métrica euclídea).

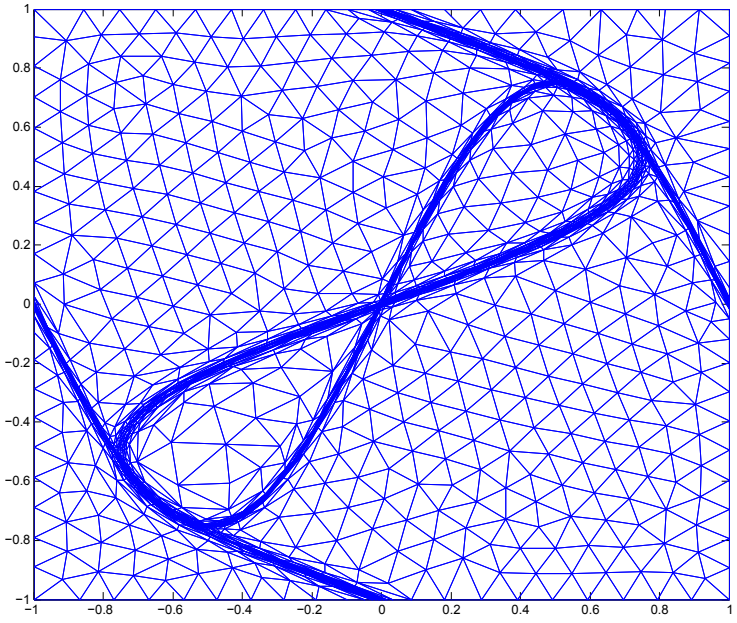


Figura 21: *Salida de* `FaseI` *aproximando la función* $f = \arctan(500 \cdot (y - 0{,}75 \cdot \sin \pi x)) + \arctan(100 \cdot (x - 0{,}75 \cdot \sin \pi y)) + 10x^2 + 10y^2$, *con* $R = 0{,}5$, $iter = 23$, $iter_s = 3$, $iter_r = 1$. $\overline{Def}_\tau = 17{,}36$, $||e||_{L^2} = 0{,}0573$, $|e|_{H^1} = 14{,}9$. *Cantidad de triángulos: 5701, tiempo de CPU: 11,43 s.*

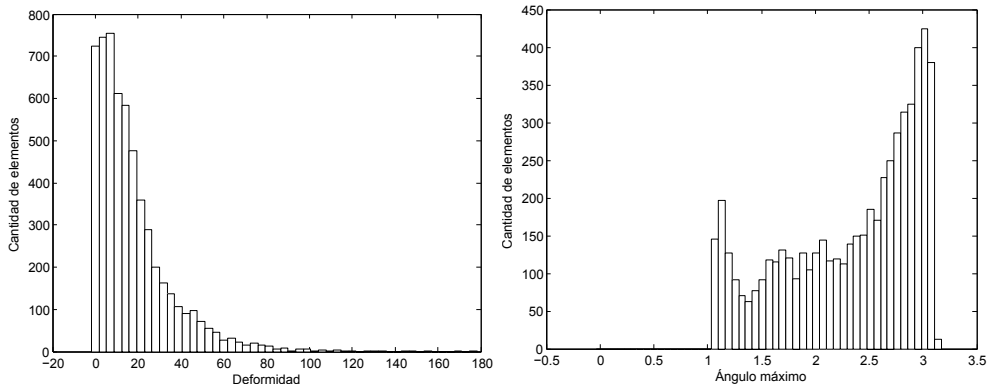


Figura 22: *A la derecha el histograma de la deformidad de los elementos, a la izquierda el histograma de los ángulos máximos (en la métrica euclídea), ambos para el caso de la figura anterior.*

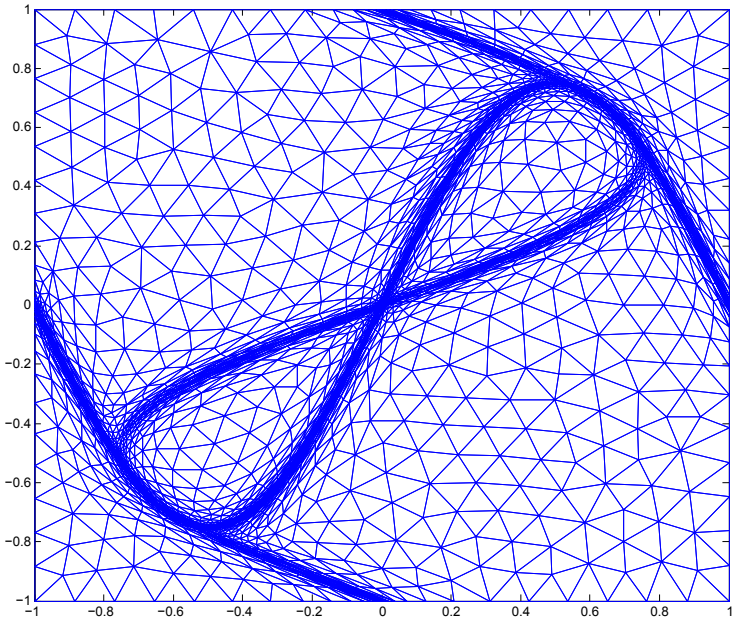


Figura 23: *Malla generada por el BAMG, aproximando la función* $f = \arctan(500 \cdot (y - 0{,}75 \cdot \sin \pi x)) + \arctan(100 \cdot (x - 0{,}75 \cdot \sin \pi y)) + 10x^2 + 10y^2$, *con* `err` $= 0{,}01$. $\overline{Def}_\tau = 14{,}76$, $||e||_{L^2} = 0{,}0583$, $|e|_{H^1} = 15{,}3$. *Cantidad de iteraciones: 25. Cantidad de triángulos: 6023. Tiempo de CPU: 14,69 s.*

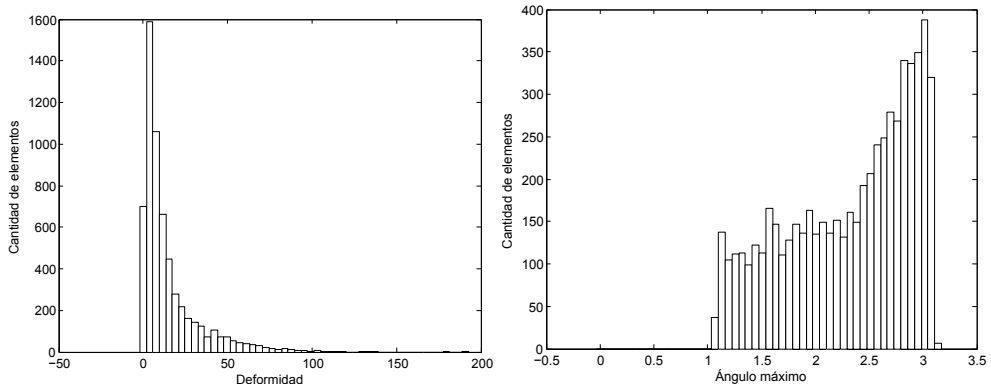


Figura 24: *A la derecha el histograma de la deformidad de los elementos, a la izquierda el histograma de los ángulos máximos (en la métrica euclídea), ambos para el caso de la figura anterior.*

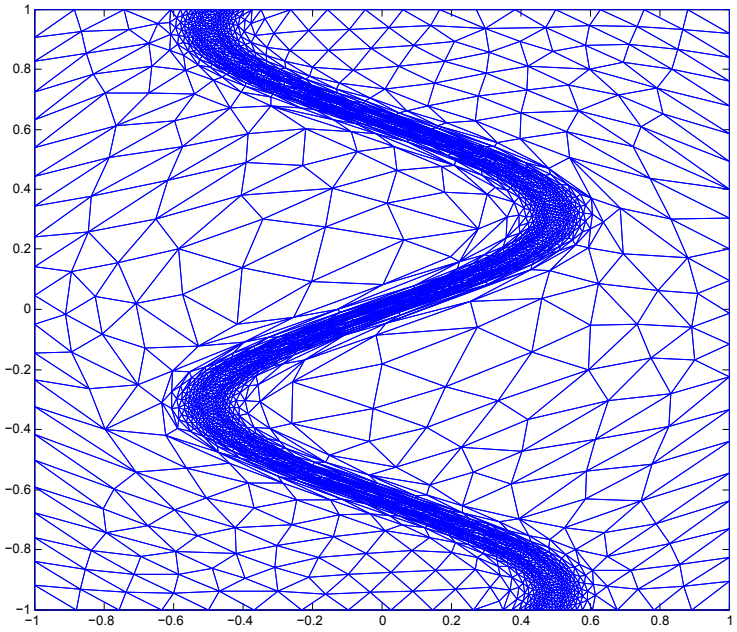


Figura 25: *Salida de* `FaseI` *aproximando la función* $f = \tanh(10(\sin 5y - 2x)) + xy^2 + y^3$, *con* $R = 0{,}18$, $iter = 17$, $iter_s = 3$, $iter_r = 1$. $\overline{Def}_\tau = 6{,}21$, $||e||_{L^2} = 0{,}0068$, $|e|_{H^1} = 0{,}8930$. *Cantidad de triángulos: 4547, tiempo de CPU: 4,53 s.*

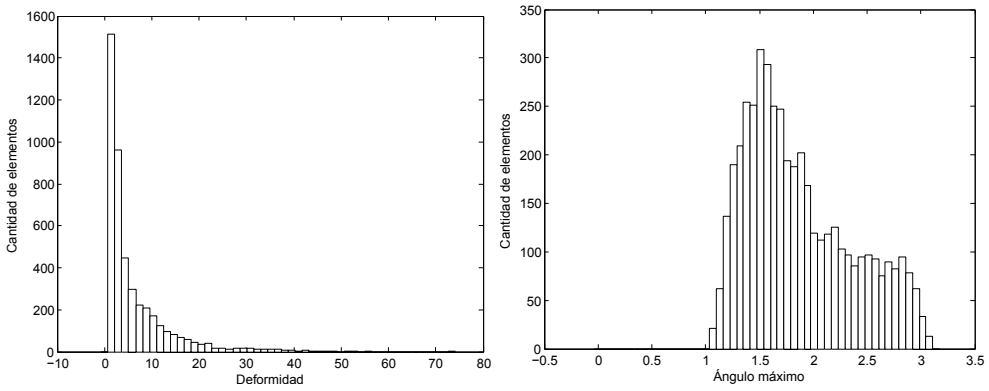


Figura 26: *A la derecha el histograma de la deformidad de los elementos, a la izquierda el histograma de los ángulos máximos (en la métrica euclídea), ambos para el caso de la figura anterior.*

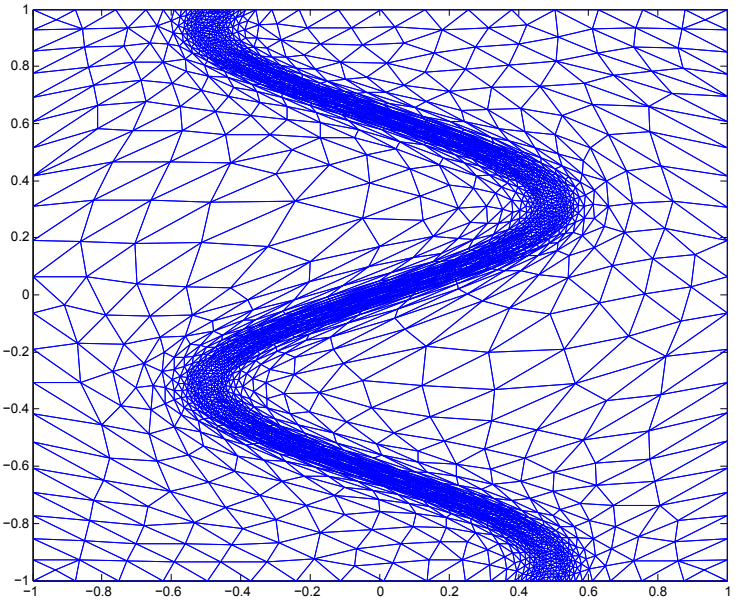


Figura 27: *Malla generada por el BAMG, aproximando la función* $f = \tanh(10(\sin 5y - 2x)) + xy^2 + y^3$, *con* `err` $= 0{,}005$. $\overline{Def}_\tau = 6{,}07$, $||e||_{L^2} = 0{,}0055$, $|e|_{H^1} = 0{,}948$. *Cantidad de iteraciones: 20. Cantidad de triángulos: 4539. Tiempo de CPU: 6,2 s.*

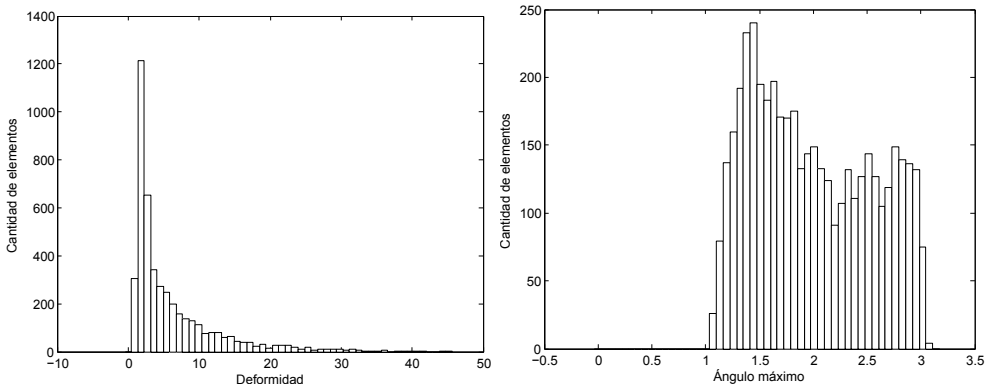


Figura 28: *A la derecha el histograma de la deformidad de los elementos, a la izquierda el histograma de los ángulos máximos (en la métrica euclídea), ambos para el caso de la figura anterior.*

5.6. Modos de uso

A partir de los resultados sobre el funcionamiento de la heurística expuestos hasta aquí, y dada la independencia de las dos fases, es posible considerar diferentes formas de combinarlas.

Dado que el costo computacional de utilizar la fase II (ya sea refinamiento local o global) es mucho menor al de la fase I, un modo de uso válido de la heurística por parte del usuario puede ser, dada una métrica, generar una malla más o menos regular con `FaseI`, de forma tal que la métrica sea más o menos constante sobre cada elemento. A continuación, aplicar `RefinarUniforme` o `RefinarUniforme*` con el objetivo de aumentar nivel de refinamiento. Es decir, orientar la utilización de la fase I para lograr regularidad, y destinar la fase II al nivel de refinamiento.

Las misma idea puede utilizarse con refinamiento local, utilizando la función `Refinar`, y a su vez valiéndose de algún tipo de estimador del error para seleccionar los elementos que deban ser refinados.

El BAMG posee la opción de realizar cuadrisecciones sobre los elementos de la malla terminada. Las cuadrisecciones se realizan de forma tal que al aplicarse sobre un triángulo equilátero genera cuatro sub-triángulos equiláteros. Esta técnica de refinamiento mantiene mejor la calidad que el algoritmo utilizado en `Refinar`. Para ver esto basta observar que al aplicar el refinamiento del BAMG al elemento de referencia sólo es posible generar elementos de dos clases de equivalencia (en el sentido de la Definición (4.2.3)), sobre los cuales a su vez la función Def resulta constante (ya que son copias del original, salvo rotación, traslación, y escala uniforme). Con lo cual haciendo un desarrollo análogo al que se hizo para el algoritmo `Refine`, en la Proposición (4.5.2) la desigualdad valdría con $C = 1$ (de hecho valdría la igualdad).

Si bien el método de cuadrisección tiene como ventaja generar elementos de mejor calidad, no puede ser aplicado de forma local por sí solo, y al aplicarlo de forma global no permite un control demasiado fino sobre la cantidad de elementos obtenidos, ya que la nueva malla contiene cuatro veces más elementos que la original, en contraposición a la aplicación de `RefinarUniforme` y `RefinarUniforme*` que generan una malla con más o menos el doble de elementos que la original.

5.7. Conclusiones

En este capítulo se vio, de forma empírica, que la heurística expuesta en los capítulos 2 y 3 cumple el objetivo de generar mallas regulares en una determinada métrica $\mathcal{M}$.

Adicionalmente, a raíz del desempeño de la función `RefinarUniforme`, se pudieron comprobar los resultados obtenidos en el Capítulo 4. Es decir que una vez que se obtiene una malla más o menos regular en la métrica $\mathcal{M}$, de forma tal que ésta sea más o menos constante sobre cada elemento, al aplicar el algoritmo de refinamiento la calidad de la malla se mantiene acotada por debajo, en función de la calidad de la malla original.

Por último, al ser mucho menor el tiempo de CPU empleado por la segunda fase del algoritmo, existen varias formas de combinar la heurística inicial y la fase de refinamiento de tal forma que puedan obtenerse mallas de buena calidad, con un nivel bajo de error, a un costo computacional bajo. Estos aspectos fueron tratados en la sección anterior.

Bibliografía

[1] F. J. Bossen, P. Heckbert, *A Pliant Method for Anisotropic Mesh Generation*, in Fifth International Meshing Roundtable, pp. 63-74. (1996)

[2] M. J. Castro-Díaz, F. Hecht, B. Mohammadi, O. Pironneau, *Anisotropic Unstructured Mesh Adaption for Flow Simulations*, Int. J. Numer. Meth. Fluids 25, pp. 475-491. (1997)

[3] F. Labelle, J. R. Shewchuk, *Anisotropic Voronoi Diagrams and Guaranteed-Quality Anisotropic Mesh Generation*, in Nineteenth Annual Symposium on Computational Geometry, New York, NY, USA, ACM, pp. 191-200. (2003)

[4] K. Shimada, A. Yamada, T. Itoh, *Anisotropic Triangular Meshing of Parametric Surfaces via Close Packing of Ellipsoidal Bubbles*, in Sixth International Meshing Roundtable, pp. 375-390. (1997)

[5] D. N. Arnold, A. Mukherjee, L. Pouly, *Locally Adapted Tetrahedral Meshes Using Bisection*, SIAM Journal on Scientic Computing 22, 2, pp. 431-448. (2001)

[6] S. M. Shontz, S. A. Vavasis, *A Mesh Warping Algorithm Based on Weighted Laplacian Smoothing*, Proceedings of the Tenth International Meshing Roundtable, Sandia National Laboratories, Santa Fe, NM , pp. 147-158. (2003)

[7] J. Vollmer, R. Mencl, H. Müller, *Improved Laplacian Smoothing of Noisy Surface Meshes*, In Proc. EuroGraphics '99, pp. 131-138. (1999)

[8] S. A. Canann, J. R. Tristano, M. L. Staten, *An Approach to Combined Laplacian and Optimization-Based Smoothing for Triangular, Quadrilateral, and Quad-Dominant Meshes*, Proceedings, 7th International Meshing Roundtable. (1998)

[9] P. Persson, G. Strang, *A Simple Mesh Generator in MATLAB*, SIAM Review, Vol. 46, No. 2. , pp. 329-345. (2004)

[10] T. Zhou, K. Shimada, *An Angle-Based Approach to Two-dimensional Mesh Smoothing*, Proceedings of the ninth International Meshing Roundtable, pp.373-384. (2000)

[11] F. J. Bossen, *Anisotropic mesh generation with particles*, Technical Report CMU-CS-96-134, CS Dept., Carnegie Mellon University. (1996)

[12] J. M. Maubach, *Local Bisection Refinement for N-simplicial Grids Generated by Reflection*, SIAM J. Sci. Comput., 16, pp. 210-227. (1995)

[13] W. Huang, *Metric Tensors for Anisotropic Mesh Generation*, J. Comput. Phys. , 204, pp. 633-665. (2005)

[14] F. Hecht, *Bidimensional Anisotropic Mesh Generator*, Technical Report, INRIA, Rocquencourt. (1997)

[15] X. Li, W. Huang, *An Anisotropic Mesh Adaptation Method for Finite Element Solution of Heterogeneous Anisotropic Diffusion Problem.* J. Comput. Phys. 229, 8072-8094. (2010)