UNIVERSIDAD DE BUENOS AIRES

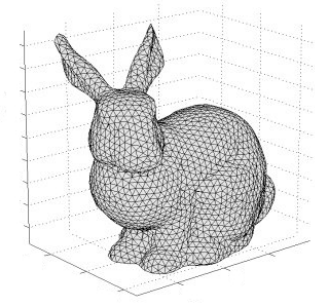
Facultad de Ciencias Exactas y Naturales

Departamento de Matemática

Tesis de Licenciatura

Generación de mallas de elementos finitos en 3D y aplicaciones a problemas elípticos.

Javier A. Kreiner



Director: Gabriel Acosta

Jueves 10 de Julio del 2008

Agradecimientos

En primer lugar quiero agradecerle a mi director de tesis, Gabriel Acosta, por aportar las ideas clave siempre que surgió algún problema, por dedicar un tiempo ilimitado a discutir conmigo de igual a igual, por tener la mejor disposición en todo momento y por su sentido del humor inacabable. Sin su invaluable guía hacer esta tesis hubiera sido imposible.

Quiero agradecer a mi familia por haber podido estudiar a lo largo de estos años sin mayores preocupaciones. A mi viejo por explicarme pacientemente varios temas relativos a la física. A mi vieja por soportar mi comportamiento huraño en épocas de estrés.

A Ricardo Durán por sus sugerencias, que me llegaron varias veces por intermedio de Gabriel.

A Julián Martínez por sus apuntes de alta calidad y explicaciones telefónicas que muchas veces me salvaron. A Manuel Maurette por las buenísimas salidas que siempre propone. A Nicolás Sirolli y a Ignacio Ojea por las amenas charlas que tuvimos en el bar. A Daniel Ziella por haber aportado varias ideas interesantes.

Índice

1. Resumen de la tesis

En esta tesis implementamos un algoritmo que permite construir la función distancia signada en dimensión 3 a objetos generados por aplicaciones comerciales de diseño con el objetivo de utilizarla en la generación de mallas de elementos finitos. La implementación se realizó sobre una placa de video de última generación, explotando la capacidad de cálculo paralelo de la misma. Como aplicación se resolvieron varios problemas elípticos en 3D usando elementos finitos lineales.

La tesis se organiza del siguiente modo: primero exhibimos brevemente el marco teórico estándar de las aproximaciones de Galerkin. Luego pasaremos a introducir las herramientas de software utilizadas y describimos el software desarrollado específicamente para esta tesis. Finalmente utilizamos las herramientas desarrolladas para resolver un problema de transferencia de calor en un disipador, calculamos un potencial electrostático dentro de cierto dipositivo y por último resolvemos un problema de elasticidad sobre un fémur humano.

2. Introducción al método de elementos finitos

El método de elementos finitos es un método numérico por el cual se pueden resolver complejos problemas científicos e ingenieriles. Sus orígenes se pueden identificar en los años 50, cuando se comenzó a aplicar en problemas de análisis estructural. Hoy en día es considerado uno de los mejores métodos para resolver eficientemente una amplia gama de problemas prácticos.

2.1. Aproximaciones de Galerkin

El método de elementos finitos es un caso particular de las llamadas aproximaciones de Galerkin. En esta sección describiremos su construcción y discutiremos algunas de sus propiedades teóricas.

Las ecuaciones diferenciales en derivadas parciales modelizan muchos problemas provenientes de diversas ramas de la física, la ingeniería, la biología, etc. Su estudio teórico raras veces brinda soluciones explícitas, centrándose en aspectos cualitativos de diversa especie, como ser propiedades de existencia, unicidad, regularidad, comportamiento asíntotico, etc. En este sentido, y con el desarrollo de las computadoras, se han generalizado diversos métodos numéricos que permiten hallar soluciones de manera aproximada. El método de Galerkin se basa en obtener una aproximación de la solución verdadera proyectando el espacio de soluciones sobre otro de dimensión finita, como veremos a continuación.

En ecuaciones elípticas lineales el procedimiento estándar consiste en llevar la ecuación diferencial a una forma variacional conveniente que en general adopta la forma:

$$\text{Hallar} \quad u \in H \quad \text{tal que} \quad a(u,v) = L(v) \quad \text{para todo} \quad v \in H \tag{1}$$

donde $L(.)$ y $a(.,.)$ representan formas lineales y bilineales respectivamente, continuas sobre un espacio de Hilbert H, que consideraremos dotado de cierto producto interno $< .,. >$, y de una norma $\| \cdot \|$ inducida por él.

La idea es aproximar la solución de (1) restringiendo nuestra búsqueda de la solución aproximada u_h a un nuevo espacio H_h de dimensión finita. En los ejemplos de esta tesis supondremos siempre que $H_h \subset H$, denominado caso conforme. El paramétro h hace referencia al nivel (resolución) de aproximación deseada y quedará claro su rol más adelante.

El problema aproximado toma la forma

$$\text{Hallar} \quad u_h \in H_h \quad \text{tal que} \quad a(u_h,v_h) = L(v_h) \quad \text{para todo} \quad v_h \in H_h \tag{2}$$

y para resolverlo lo usual es considerar una base $\{\phi_i\}_{i=1..N}$ de H_h, y escribiendo

$$u_h = \sum_{i=1}^{n} \lambda_i \phi_i,$$

se tiene que (2) es equivalente a hallar $\{\lambda_i\}_{i=1..N}$ tal que

$$\sum_{i=1..N} \lambda_i a(\phi_i, \phi_j) = L(\phi_j) \quad \text{para todo} \quad \phi_j, j = 1..N, \tag{3}$$

que no es otra cosa que un sistema lineal de ecuaciones en los λ_i, y que, bajo la condición de que la forma $a(.,.)$ sea coerciva y simétrica, tendrá siempre solución única, pues su matriz resultará simétrica y definida positiva.

Ahora interesa obtener cotas para el error $u - u_h$ con la intención de tener una noción precisa de la calidad de la aproximación. Para ello es muy útil la llamada ecuación del error que, si $H_h \subset H$, se escribe

$$a(u - u_h, v_h) = 0 \quad \text{para todo} \quad v_h \in H_h \tag{4}$$

y que se obtiene reemplazando v por $v_h \in H_h \subset H$ en (1) y restándole a esta ecuación (2). Notemos que (4) expresa un relación de ortogonalidad entre el error $u - u_h$ y el subespacio H_h. De hecho si $a(.,.)$ es simétrica continua y coerciva, entonces induce, como producto interno, una norma equivalente a la de H y entonces es de esperar que u_h posea las propiedades de ser la mejor aproximación de u sobre H_h. En efecto, a partir de la ecuación del error (4) es sencillo obtener cotas para $u - u_h$ en la norma $\|.\|$ dada por $< ., . >$. De allí, para $v_h \in H_h$ arbitrario, resulta

$$\|u - u_h\|^2 \gamma \leq a(u - u_h, u - u_h) = a(u - u_h, u - v_h) \leq C\|u - u_h\|\|u - v_h\|, \tag{5}$$

donde γ y C son las constantes de coercividad y continuidad, respectivamente, de la forma bilineal a. Una simple reorganización de esta desigualdad conduce al clásico Lema de Cea [4, 3],

$$\|u - u_h\| \leq \frac{C}{\gamma} inf_{v_h \in H_h} \|u - v_h\| \tag{6}$$

que indica, precisamente, que el error se comporta (salvo una constante multiplicativa) como la mejor aproximación.

De esta manera podemos reducir el problema de estudiar el error a hallar algún operador de

interpolación Π sobre el espacio H_h

$$\Pi : H \to H_h,$$

y estudiar su error $u - \Pi(u)$.

La construcción del operador Π dependerá de la naturaleza exacta de los espacios H y H_h. En los problemas prácticos H resulta ser un espacio de Sobolev y entonces es muy usual tomar H_h como un espacio de elementos finitos conveniente.

El procedimiento general consiste en realizar una partición $\mathcal{T}$ de Ω (siendo Ω el dominio sobre el que se plantea la ecuación diferencial) en subconjuntos simples denominados elementos, por ejemplo símplices - segmentos en una dimensión, triángulos en 2 dimensiones, tetraedros en 3 -, de manera tal que esta partición sea una triangulación admisible, esto es que si K_1, K_2, son dos elementos de $\mathcal{T}$ entonces

$$K_1 \cap K_2 = \emptyset \quad \text{o bien} \quad K_1 \cap K_2 = F$$

con F una cara, arista, o vértice en dimensión 3 (un lado o un vértice en dimensión 2) de K_1 y K_2 [3, 4]. Por ejemplo en el caso $H_h \subset H$, suele elegirse H_h como el espacio de funciones en H que restringidas a cada elemento $K \in \mathcal{T}$ resultan ser polinomiales de cierto grado y globalmente continuas (es decir, continuas entre elementos). El subíndice h está asociado al diámetro máximo de los elementos de $\mathcal{T}$. A medida que este diámetro tiende a cero, si los espacios H_h son elegidos adecuadamente, resultará que la solución obtenida convergerá a la solución del problema original en la norma de H.

Cuando H_h tiene la forma mencionada existe una teoría muy completa para acotar de manera sistemática $u - \Pi(u)$. En este enfoque la estimación de $u - \Pi(u)$ depende de ciertas características geométricas de los elementos utilizados. Para asegurar cotas de error uniformes dentro de una familia se requiere que estos cumplan la denominada *condición de regularidad*.

Definición 1. *Diremos que una triángulación o por abuso de lenguaje un elemento K (denotando un elemento arbitrario de la familia) es regular si existe $\sigma > 0$ independiente de K tal que*

$$\frac{h_K}{\rho_K} \leq \sigma \tag{7}$$

Donde h_K es el diámetro exterior y ρ_K es el diámetro interior del elemento. Bajo esta condición es posible, como veremos, acotar el error cometido al utilizar la aproximación u_h.

2.2. Operador de interpolación de Lagrange y cotas de error

Con el fin de obtener una cota para la interpolación de Lagrange introduciremos ahora la notación necesaria:

Denotaremos a los elementos triangulares (en 2 dimensiones) o tetraédricos (en 3 dimensiones) con $K \subset \mathbb{R}^n$ ($n = 2$ ó 3), con $\mathbf{p}_0$ denotaremos un vértice, con $\mathbf{v}_i$ ($\|\mathbf{v}_i\| = 1$) y l_i , $1 \leq i \leq n$ (con $n = 2$ ó 3) las direcciones y longitudes de los lados (*resp.: aristas*) concurrentes en $\mathbf{p}_0$ respectivamente. Podemos así escribir $K = c.c.\{\mathbf{p}_0, \mathbf{p}_0 + l_i\mathbf{v}_i\}_{1\leq i\leq n}$, con *c.c.* representando la cápsula convexa. Al elemento de referencia $\hat{K}$ lo definimos como $\hat{K} = c.c.\{\mathbf{0}, \mathbf{e}_i\}_{1\leq i\leq n}$ donde $\mathbf{e}_i$ representan los vectores canónicos. Con h_K y ρ_K denotaremos el diámetro exterior e interior de K respectivamente, es decir, el menor de los diámetros de las circunferencias circunscriptas al elemento, para el primer caso, y el mayor de las inscriptas para el segundo. Llamaremos $F : \hat{K} \to K$ a la transformación *afín* $F(\hat{x}) = B\hat{x} + \mathbf{p}_0$ donde la matriz B queda definida por $B\mathbf{e}_i = l_i\mathbf{v}_i$. A una función dada w, definida en K, le vamos a asociar otra función $\hat{w}$ definida en $\hat{K}$ de la siguiente manera: $\hat{w} = w \circ F$.

Con $\|.\|_{m,K}$, y $|.|_{m,K}$ denotaremos las normas y seminormas, respectivamente, en los espacios de Sobolev $H^m(K)$, esto es funciones de $L^2(K)$ con m derivadas distribucionales en $L^2(K)$. Es decir,

$$\|u\|_{m,K} = \sum_{0\leq\alpha\leq m} \|\frac{\partial^\alpha u}{\partial x^\alpha}\|_{L_2(K)}$$

y

$$|u|_{m,K} = \sum_{\alpha=m} \|\frac{\partial^\alpha u}{\partial x^\alpha}\|_{L_2(K)}.$$

Además, omitiremos el K cuando no resulte necesario explicitarlo.

Con $\mathcal{P}_l(K)$, denotaremos el conjunto de los polinomios de grado a lo sumo l sobre K. La interpolación de Lagrange sobre $\mathcal{P}_1$ se define a través de las funciones básicas $\hat{\phi}_i \in \mathcal{P}_1(\hat{K})$ $0 \leq i \leq n$ que verifican $\hat{\phi}_i(\hat{\mathbf{p}}_j) = \delta_i^j$ con $\hat{\mathbf{p}}_0 = \mathbf{0}$, $\hat{\mathbf{p}}_j = \mathbf{e}_j$. Si $\hat{u} \in H^2(\hat{K})$, se toma $\hat{\Pi}(\hat{u}) = \sum_{i=0}^n \hat{u}(\hat{\mathbf{p}}_i)\hat{\phi}_i$, que no es otra cosa que la función lineal que interpola a $\hat{u}$ en los vértices de $\hat{K}$. Conviene observar que en el marco en el que trabajamos la regularidad requerida $\hat{u} \in H^2(\hat{K})$ asegura en dimensión 2 y 3, por un teorema clásico de inmersión, que $\hat{u} \in \mathcal{C}^0(\hat{K})$, y entonces tienen sentido las evaluaciones $\hat{u}(\hat{\mathbf{p}}_i)$. Ahora para un K cualquiera y para una $u \in H^2(K)$, la interpolación lineal puede escribirse como $\Pi(u) = \hat{\Pi}(u \circ F) \circ F^{-1} = \hat{\Pi}(\hat{u}) \circ F^{-1}$. Por lo tanto tomando $\phi_i = \hat{\phi}_i \circ F^{-1}$ como funciones básicas sobre K, la interpolación se expresa $\Pi(u) = \sum_{i=0}^n u(\mathbf{p}_i)\phi_i$, donde los $\mathbf{p}_i$ representan ahora los correspondientes vértices de K. Note que estas funciones básicas ϕ_i son también lineales, debido

a que es afín el cambio de variables.

Dada una matriz M, vamos a denotar con $\|M\|$ su norma 2. Es decir que si $B \subset \mathbb{R}^n$ es la bola unitaria con la norma vectorial $\|\ \ \|_2$ (norma euclídea) y $B_r \subset \mathbb{R}^n$ es una bola de radio r, entonces $\|M\| = sup_{v\in B}\|M(v)\|_2 = \frac{1}{r} sup_{v\in B_r}\|M(v)\|_2$. Es sabido que toda otra norma es equivalente a ésta, y a los efectos de hacer acotaciones puede cambiarse una por otra salvando una constante multiplicativa.

El primer resultado que demostramos es el llamado Lema de Bramble-Hilbert:

Lema 1. *Existe una constante $C = C(K)$ tal que para toda $u \in H^2(K)$*

$$inf_{p\in\mathcal{P}(K)}\|u-p\|_{2,K} \leq C|u|_{2,K} \tag{8}$$

Demostración. Consideremos $\{f_i\}_{0\leq i\leq n}$ una base del espacio dual de $\mathcal{P}_1(K)$. Por estar en dimensión finita las f_i resultan continuas con la norma de $H^2(K)$ y podemos entonces (Hahn-Banach) suponerlas definidas sobre todo $H^2(K)$. Por ser $\{f_i\}_{0\leq i\leq n}$ una base del dual se tiene también que dichas f_i verifican la siguiente propiedad $(\mathcal{P})$:

$$(\mathcal{P}) \qquad Dado\ \ p \in \mathcal{P}_1(K),\ \ f_i(p) = 0\ \ \forall i\ \ 0 \leq i \leq n\ \ si\ \ y\ \ solo\ \ si\ \ p = 0.$$

Demostremos ahora:

$$\|u\|_{2,K} \leq C(|u|_{2,K} + \sum_{i=0}^{n} |f_i(u)|) \qquad \forall u \in H^2(K) \tag{9}$$

para ello supongamos que la desigualdad no se verifica para ninguna constante C, y elijamos entonces una sucesión $u_j \in H^2(K)$ tal que $||u_j||_{2,K} > j(|u_j|_{2,K} + \sum_{i=0}^{n} |f_i(u_j)|)$, es decir

$$\frac{1}{j}||u_j||_{2,K} > |u_j|_{2,K} + \sum_{i=0}^{n} |f_i(u_j)|.$$

Sea $\hat{u_j} = \frac{u_j}{||u_j||_{2,K}}$, entonces $||\hat{u_j}||_{2,K} = 1$ y

$$|\hat{u_j}|_{2,K} + \sum_{i=0}^{n} |f_i(\hat{u_j})| \to 0$$

Ahora bien, por estar $\hat{u_j}$ acotada en $H^2(K)$ existe una subsucesión, que seguimos indicando por

$\hat{u_j}$, convergente, en H^1, a cierta $\hat{u}$, $\hat{u} \in H^1(K)$, pero por otro lado se tiene que $|\hat{u_j}|_{2,K} \to 0$ de donde resulta que además $\hat{u_j}$ converge en $H^2(K)$(ya que la seminorma $|v|_{2,K}$ es lo que se debe agregar a la norma $||v||_{1,K}$ para obtener la norma de $H^2(K)$ de v, $||v||_{2,K}$). De estas observaciones se infiere que $\hat{u} \in H^2(K)$, y $|\hat{u}|_{2,K} = 0$. Notemos que esto último nos dice en particular que $\hat{u} \in \mathcal{P}_1(K)$, y teniendo en cuenta que $f_i(\hat{u}) = lim_{j\to\infty} f_i(\hat{u_j}) = 0 \ \forall i \ 1 \leq i \leq n$ y usando la propiedad $(\mathcal{P})$ debe ser $\hat{u} = 0$, en contra de la condición $|||\hat{u_j}||_{2,K} = 1$. Esto prueba (9).
Para finalizar la demostración notemos que por definición de espacio dual podemos elegir un $q_i \in \mathcal{P}_1(K)$ tal que $f_i(q_j) = \delta_{ij}$ y entonces si $f_i(u) = \lambda_i$ definiendo $q = -\sum_{i=0}^{n} \lambda_i q_i$ resulta que $q \in \mathcal{P}_1(K)$, $f_i(u+q) = 0$ para $1 \leq i \leq n$, y por lo tanto de (9) resulta

$$\inf_{p\in\mathcal{P}_1(K)} \|u-p\|_{2,K} \leq \|u+q\|_{2,K} \leq C|u|_{2,K} \tag{10}$$

la última desigualdad a causa de que $|q|_{2,K} = 0$ dado que $q \in \mathcal{P}_1(K)$ y sus derivadas segundas son por lo tanto nulas. Esto demuestra el lema. □

Lema 2. *Sea $K = F(\hat{K})$, para cada $u \in H^m(K)$ definimos como antes $\hat{u} = u \circ F$, entonces $\hat{u} \in H^m(\hat{K})$ y además existe una constante que depende sólo de la dimensión y de m tal que*

$$|\hat{u}|_{m,\hat{K}} \leq C\|B\|^m det(B)^{-1/2}|u|_{m,K}$$

donde, como ya se ha mencionado, B es la matriz que aparece en la definición de F, que es la transformación afín del dominio de referencia $\hat{K}$ al dominio de interés K y también se tiene análogamente

$$|u|_{m,K} \leq C\|B^{-1}\|^m det(B)^{1/2}|\hat{u}|_{m,\hat{K}}$$

Demostración. Dado que $|\hat{u}|_{m,\hat{K}} = \sum_{\alpha=m} \|\frac{\partial^\alpha \hat{u}}{\partial x^\alpha}\|_{L_p(K)}$, esta desigualdad se obtiene al combinar una desigualdad por el cambio de variables en la integral:

$$\int_{\hat{K}} \hat{u}^2(\hat{x})d\hat{x} = \int_{\hat{K}} (u(F(\hat{x})))^2 d\hat{x} = \int_K u^2(x)[Det(B)]^{-1}dx \tag{11}$$

y en consecuencia $\|\hat{u}\|_{2,\hat{K}} = [Det(B)]^{-1/2}\|u\|_{2,K}$ y por otra desigualdad debida a la regla de la cadena para la diferenciación:

$$|\frac{\partial \hat{u}}{\partial \hat{x_j}}| = |\sum_{i=1}^{n} b_{ji}\frac{\partial u}{\partial x_j}| \leq \|B_j\|_2 \|(\frac{\partial u}{\partial x_j})_{j=1..n}\|_2 \leq \max_{i,j}|b_{ij}|\|(\frac{\partial u}{\partial x_j})_{j=1..n}\|_2 \tag{12}$$

siendo B_j la fila j de la matriz B y b_{ij} sus coeficientes y por lo tanto, teniendo en cuenta que todas las normas son equivalentes en dimensión finita, podemos encontrar una constante $\hat{C}$ tal que $|\frac{\partial \hat{u}}{\partial \hat{x_j}}|^2 \leq \hat{C}\|B\|_2^2 \sum_{i=1}^{n} (\frac{\partial u}{\partial x_j})^2$.

Articulando ambas observaciones obtenemos:

$$|\hat{u}|_{1,2,\hat{K}} \leq C\|B\| det(B)^{-1/2} |u|_{1,2,K},$$

este es el resultado para $m = 1$. Para $m = 2$ se repite el argumento anterior y aparece un nuevo factor $\|B\|$ multiplicando, de donde resulta el lema. □

Seguimos acumulando elementos para llegar a la desigualdad que nos interesa. En el siguiente Lema se establecen cotas para $\|B\|$ y $\|B^{-1}\|$, dichas cotas llevan de manera natural a la condición de regularidad.

Proposición 1. *Sea K un triángulo o un tetraedro arbitrario, y $K = F(\hat{K})$, entonces*

$$\|B\| \leq \frac{h_K}{\rho_{\hat{K}}} \qquad \|B^{-1}\| \leq \frac{h_{\hat{K}}}{\rho_K}$$

Demostración. Basta considerar $\|B\| = \frac{1}{\rho_{\hat{K}}} sup_{\|\psi\|=\rho_{\hat{K}}} \|B(\psi)\|$. Por definición de $\rho_{\hat{K}}$ existen $\hat{x}, \hat{y} \in \hat{K}$ tales que $\hat{x} - \hat{y} = \psi$, en consecuencia $\|B(\psi)\| = \|F(\hat{x}) - F(\hat{y})\| \leq h_K$, de donde se obtiene la primer desigualdad propuesta. La otra sigue análogamente. □

Teorema 1. *Sea K un triángulo o un tetraedro, y $u \in H^2(K)$. Para $m = 0, 1$, se tienen la siguientes estimaciones*

$$\|u - \Pi(u)\|_{m,K} \leq C\|B\|^{2-m} \|B^{-1}\|^m \|u\|_{2,K} \tag{13}$$

donde C depende solo de $\hat{K}$, el dominio de referencia.

Demostración. Se demuestra primero para el dominio de referencia y después se cambia de variables. Observemos que el operador $\hat{\Pi} : H^m(\hat{K}) \to H^2(\hat{K})$ con $m = 0, 1$ resulta continuo.
En efecto, consideremos $\hat{u} \in H^2(\hat{K})$ entonces

$$\|\hat{\Pi}(\hat{u})\|_{m,\hat{K}} \leq \sum_{i=1}^{n} |\hat{u}(p_i)| \|\hat{\phi}_i\|_{m,\hat{K}} \leq C(\hat{K}) \|\hat{u}\|_{\infty,\hat{K}} \leq C(\hat{K}) \|\hat{u}\|_{2,\hat{K}} \tag{14}$$

donde hemos utilizado en la última desigualdad la inclusión de Sobolev $H^2(\hat{K}) \subset \mathcal{C}^0(\hat{K})$, que resulta válida para dimensión 2 y 3.

Ahora bien, para todo $\hat{p} \in \mathcal{P}_1(\hat{K})$ tenemos $\Pi(\hat{p}) = \hat{p}$ y de allí

$$\hat{u} - \Pi(\hat{u}) = (I - \Pi)(\hat{u} + \hat{p}) \qquad \forall \hat{u} \in H^2(K) \tag{15}$$

donde I representa la inclusión $I : H^2(\hat{K}) \to H^m(\hat{K})$. De (15), la continuidad de Π y de I y el Lema 1 obtenemos

$$|\hat{u} - \hat{\Pi}(\hat{u})|_{m,\hat{K}} \leq \|I - \hat{\Pi}\|_{\mathcal{L}(H^2(\hat{K}),H^m(\hat{K}))} inf_{\hat{p}\in\mathcal{P}_1(\hat{K})} \|\hat{u} + \hat{p}\|_{2,\hat{K}} \leq C(\hat{K})|\hat{u}|_{2,\hat{K}} \tag{16}$$

que es (13) con $K = \hat{K}$.
Ahora se considera

$$\hat{u} - \hat{\Pi}(\hat{u}) = (u - \Pi(u)) \circ F = (u - \Pi(u))\hat{}$$

así que cambiando variables se tiene del Lema 2

$$|u - \Pi(u)|_{m,K} \leq C\|B^{-1}\|^m |detB|^{1/2} |\hat{u} - \hat{\Pi}(\hat{u})|_{m,\hat{K}} \tag{17}$$

del mismo Lema 2

$$|\hat{u}|_{2,\hat{K}} \leq C\|B\|^2 |detB|^{-1/2} |u|_{2,K} \tag{18}$$

y de (16), (17), (18) se deduce (13). □

Como Corolario de este Teorema y de la Proposición 1 obtenemos la desigualdad clásica de interpolación:

$$\|u - \Pi(u)\|_{m,K} \leq C \frac{h_K^2}{\rho_K^m} \|u\|_{2,K} \tag{19}$$

Se pueden hacer dos observaciones: la primera es que si $m = 0$ la estimación permanecerá uniforme independientemente de la geometría de los elementos en cuestión, la segunda es que el control de la seminorma con $m = 1$ puede asegurarse *a priori* si requerimos una cota para $\frac{h_K}{\rho_K}$ que controle este cociente para todos los elementos de una triangulación dada del dominio. Esta es la llamada *condición de regularidad* que ya mencionamos, (1).

Finalmente podemos hilvanar los resultados de estas dos últimas secciones para obtener una acotación del error $u - u_h$ en terminos de la malla de elementos finitos. En efecto, gracias al lema de Cea (6)

$$\|u - u_h\| \leq \frac{C}{\gamma} \|u - \Pi u\|$$

vemos que para una forma bilineal continua y coerciva sobre el espacio $H^1(\Omega)$, y una triangulación regular de Ω obtenemos, utilizando aproximaciones en el espacio P_1, cotas para el error dependientes del parámetro h a través de (19) siempre que la solución u resulte suficientemente regular(p. ej. $u \in H^2$).

La implementación del método de elementos finitos requiere de la triangulación mencionada y en dimensión 2 y 3 ese problema presenta una seria dificultad. En dimesión 1 las triangulaciones son uniones de intervalos y en ese caso puede darse en general una expresion explícita del sistema (3).

Ejemplo unidimensional.

Consideremos el siguiente problema,

$$\begin{cases} -u''(x) = f(x) & x \in (0,1) \\ u(0) = u(1) = 0 \end{cases} \tag{20}$$

si multiplicamos la ecuación por cualquier función $v \in H_0^1(0,1)$ obtenemos:

$$-v(x).u''(x) = v(x).f(x) \tag{21}$$

Integrando:

$$\int_{[0,1]} -v(x).u''(x) = \int_{[0,1]} v(x).f(x). \tag{22}$$

Integrando por partes el primer término obtenemos la forma variacional de (20):

Hallar $u \in H_0^1$ tal que

$$\int_{[0,1]} v'(x).u'(x) = \int_{[0,1]} v(x).f(x) \quad \text{para toda} \quad v \in H_0^1. \tag{23}$$

Sea $0 = x_0 < x_1 < ... < x_{M+1} = 1$ una partición del intervalo $(0,1)$ en subintervalos $I_j = (x_{j-1}, x_j)$, $h_j = x_j - x_{j-1}$ y sea $h = max(h_j)$. Sea H_h el subespacio de funciones v continuas, lineales en cada subintervalo y tal que son nulas en $x = 0$ y en $x = 1$. Se tiene trivialmente que $H_h \subset H_0^1(0,1)$.

Construimos explícitamente un base nodal de H_h como sigue:

$$\phi_j(x) = \begin{cases} \frac{(x-x_{j-1})}{(x_j-x_{j-1})} & x \in [x_{j-1}, x_j] \\ \frac{(x-x_{j+1})}{(x_j-x_{j+1})} & x \in [x_j, x_{j+1}] \\ 0 & x \in ([0,1] \setminus [x_{j-1}, x_{j+1}]) \end{cases} \tag{24}$$

Tenemos entonces la aproximación de Galerkin siguiente:

Hallar $u_h \in H_h$ tal que

$$\int_{[0,1]} v_h'(x).u_h'(x) = \int_{[0,1]} v_h(x).f(x) \quad \text{para toda} \quad v_h \in H_h. \tag{25}$$

Escribiendo la solución $u_h = \sum_j \mu_j \phi_j(x)$ resulta

$$\sum_j \mu_j \int_{[0,1]} \phi_i'(x)\phi_j'(x)dx = \int_{[0,1]} f(x)\phi_i'(x)dx \qquad i = 1...n \tag{26}$$

Por otro lado $\int_{[0,1]} \phi_i'(x)\phi_j'(x)dx = 0$ si $|i-j| > 1$ ya que la multiplicación es nula en ese caso, y ademas:

$$\int_{[0,1]} \phi_j'(x)\phi_j'(x)dx = \int_{[x_{j-1},x_j]} \frac{1}{h_j^2}dx + \int_{[x_j,x_{j+1}]} \frac{1}{h_{j+1}^2}dx = \frac{1}{h_j} + \frac{1}{h_{j+1}} \tag{27}$$

similarmente

$$\int_{[0,1]} \phi_{j-1}'(x)\phi_j'(x)dx = -\int_{[x_{j-1},x_j]} \frac{1}{h_j^2}dx = -\frac{1}{h_j}. \tag{28}$$

De esta manera la matriz del sistema a resolver, denominada matriz A de rigidez, queda como sigue:

$$\begin{bmatrix} \frac{1}{h_1}+\frac{1}{h_2} & -\frac{1}{h_1} & . & . & . & . \\ -\frac{1}{h_1} & \frac{1}{h_2}+\frac{1}{h_3} & -\frac{1}{h_2} & . & . & . \\ . & -\frac{1}{h_2} & \frac{1}{h_3}+\frac{1}{h_4} & -\frac{1}{h_2} & . & . \end{bmatrix} \tag{29}$$

Y se debe resolver:

$$A\mu = b, \tag{30}$$

donde $b_i = \int_{[0,1]} f(x)\phi_i(x)dx$, que suele aproximarse numericamente utilizando el valor de la f en los nodos (p. ej. $b_i = \frac{h_i+h_{i+1}}{2} f(x_i)$).

Observemos finalmente que la forma bilineal

$$\langle u, v\rangle = \int_{[0,1]} u'v'dx$$

resulta trivialmente continua en $H_0^1(0,1)$, siendo además coerciva por la desigualdad de Poincaré. En consecuencia gracias a la teoría comentada previamente se tiene que

$$\|u - u_h\|_{1,(0,1)} \leq Ch\|f\|_{0,(0,1)},$$

de donde resulta la convergencia del método.

3. Generación de la malla

3.1. Introducción

En general los códigos generadores de mallas son extremadamente complicados, casi inaccesibles. La mayoría de las veces el usuario los utiliza como si fueran "cajas negras", es decir que no sabe los detalles de su funcionamiento y confía en que el algoritmo produce una salida correcta. Por eso, para tener una idea cabal del funcionamiento de todas las partes del código a utilizar, y en consecuencia poder modificarlo a nuestro antojo, decidimos hacer uso de un código denominado DistMesh, un generador de mallas escrito en Matlab, desarrollado por Per-Olof Persson y Gilbert Strang, de MIT [12] . La idea de este algoritmo es utilizar una analogía física para disponer los puntos y crear los tetraedros o triángulos de la forma más equilibrada posible. El resultado es que las mallas generadas son de muy alta calidad.

Para definir el objeto a mallar se utiliza lo que se denomina la distancia signada. Esta función toma en cada punto del espacio la distancia al borde del objeto, siendo negativa dentro y positiva fuera de él. Mostraremos cómo construir la función de distancia para objetos simples como esferas y cilindros. También mostraremos cómo obtener funciones de distancia aproximadas para la diferencia, unión o intersección de figuras simples. Por último describimos nuestra estrategia para objetos generales.

Junto con la función de distancia d contamos con una función h que indica en cada región del espacio cuánta resolución queremos que tenga la malla. Esto sirve para refinar la malla en lugares convenientes -donde la función solución del problema no es tan suave o donde la geometría es complicada.

Ahora describimos la técnica iterativa utilizada en este algoritmo. Los puntos de la malla se consideran como nodos de una estructura de "barras" o "resortes", se puede imaginar algo así como un puente colgante por ejemplo, las aristas - generadas por el algoritmo de Delaunay- de los tetraedros o triángulos son "barras" o "resortes" que ejercen, según su compresión o estiramiento, fuerza sobre los nodos. Cada nodo está sometido a una suma de fuerzas y se trata de reubicarlos de tal manera que se alcance un equilibrio, i.e. la suma de fuerzas en cada uno sea cero. Esto se hace iterativamente con el algoritmo de Euler. Si algún nodo, al ser reubicado, se desplaza fuera del objeto, entonces se usa la función de distancia para reproyectarlo hacia la superficie, en consecuencia es imperativo poder calcularla lo más rápido posible.

3.2. El algoritmo

Primero describiremos el algoritmo para 2 dimensiones.

Dado un conjunto P de puntos en el plano, se denomina triangulación de Delaunay a una subdivisión de su cápsula convexa en triángulos tal que

- Cada triángulo está formado por tres puntos de P.
- La intersección de dos triángulos distintos: es nula; consiste en un punto de P; o consiste en el segmento que une dos puntos de P.
- Ningún punto está dentro del circumcículo de un triángulo de la triangulación.

Si los puntos P están en posición general(no hay tres puntos sobre una misma línea o cuatro sobre un mismo círculo) entonces la triangulación de Delaunay es única. La triangulación de Delaunay tiene la propiedad de maximizar el mínimo ángulo de la triangulación[13].

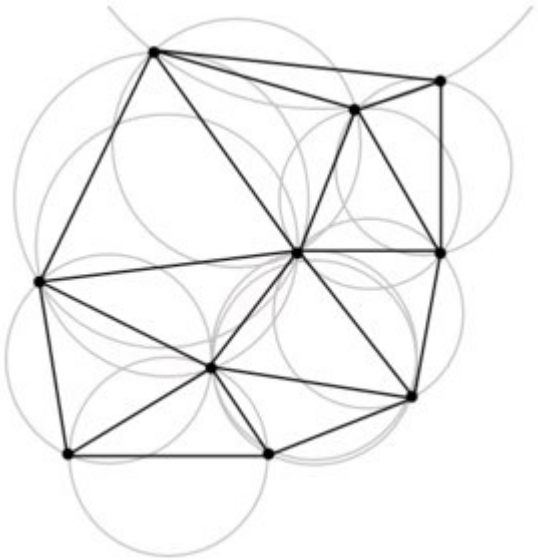


Figura 1: Triangulación de Delaunay de un conjunto de puntos.

Las aristas generadas por el algoritmo de Delaunay se consideran "resortes", cada uno tiene una fuerza de desplazamiento $f(l, l_0)$, que depende de su longitud actual l y su longitud en relajación l_0.

Las fuerzas externas sobre la estructura actúan sobre los bordes. Sobre cada nodo del borde actúa una fuerza normal a la superficie lo suficientemente fuerte como para matenerlo dentro del objeto.

La posición de los nodos se encuentra resolviendo las ecuaciones de equilibrio para las fuerzas. La esperanza es que -cuando $h(x) = 1$, es decir la resolución es uniforme- las barras tendrán todas casi la misma longitud, resultando en una estructura triangular regular (si todas tuvieran la misma longitud habría sólo triángulos equiláteros, algo ideal).

Para resolver las ecuaciones de equilibrio se escribe $p = [x, y]$, donde x e y son dos arreglos de $N \times 1$ con las coordenadas de cada nodo. $F(p)$ contiene las componentes verticales y horizontales de las fuerzas sobre cada nodo.

$$F(p) = [F_{int,x}(p), F_{int,y}(p)] + [F_{ext,x}(p), F_{ext,y}(p)]$$

Aquí F_{int} contiene las fuerzas internas de las barras y F_{ext} las fuerzas externas (normales al borde). $F(p)$ depende de la topología de las aristas, dada por la triangulación de Delaunay. $F(p)$ no es una función continua de los puntos, ya que al mover los puntos la topología cambia.

Se intenta resolver $F(p) = 0$ para algún conjunto de puntos. Este problema es relativamente difícil, en parte por la discontinuidad de F y en parte por las fuerzas actuando en el borde.

Una manera de resolverlo es introducir una dependencia temporal artificial. Para $p(0) = p_0$, se considera el sistema de ecuaciones diferenciales ordinarias:

$$\frac{dp}{dt} = F(p), \qquad t \geq 0 \tag{31}$$

Si encontramos una solución estacionaria tenemos que $F(p) = 0$. Aproximamos la solución de (31) utilizando el método de Euler forward. En el momento $t_n = n \times \Delta t$ tenemos:

$$p_{t_{n+1}} = p_{t_n} + F(p_{t_n}) \times \Delta t$$

Cuando se calcula la función de fuerza las posiciones de los puntos son conocidas y también la triangulación. Las fuerzas exteriores ingresan de la siguiente manera: todos los puntos que en una iteración se mueven fuera del dominio (el sector del espacio donde la distancia es negativa) son reproyectados hacia la superficie. De esta manera los puntos pueden moverse sobre el borde pero no fuera del objeto.

Hay muchas posibilidades para la fuerza $f(l, l_0)$ en cada barra. La función $k(l_0 - l)$ modela un

resorte tradicional lineal. La opción utilizada en el presente algoritmo es:

$$f(l, l0) = \begin{cases} k(l_0 - l) & l < l_0 \\ = 0 & l \geq l_0 \end{cases}$$

Otras funciones levemente no lineales dan buenos resultados. Pero esta función, que no tiene fuerzas repulsivas cuando $l = l_0$, funciona razonablemente bien y se puede calcular rápidamente. En el método propuesto los puntos no son obligados a quedarse sobre el borde, por eso es importante que la mayoría de las fuerzas sean repulsivas, para lograr que los puntos se dispersen a través de toda la geometría. Eso implica que f debería ser positiva cuando l está cerca de la longitud deseada, lo que se logra haciendo que l_0 sea levemente más grande que el valor deseado (un buen número en 2-D es l_0 un 20 % mayor que la longitud deseada).

Para mallas uniformes l_0 es constante. Si se quiere variabilidad en la resolución se utiliza h, la función de tamaño de los elementos. Esto es útil para manejar casos en que la geometría es irregular en algunos lugares o en que la función solución es menos suave en ciertas partes del dominio. La función h no da exactamente el tamaño de los elementos, sino la distribución relativa de puntos a través del dominio. Esto evita especificar una conexión entre h y la cantidad de elementos, haciendo la vida del usuario más simple. Por ejemplo, si $h(x, y) = 1 + x$ en el cuadrado unitario, entonces las longitudes de las aristas cerca de $x = 0$ serán aproximadamente la mitad de la longitud de las aristas cerca de $x = 1$. Esto es así sin importar el número de puntos o el tamaño de los elementos. Para encontrar el factor de escala, computamos la división entre el área generada por la longitud de la aristas en un momento dado y el área dada por la longitud "deseada" ($h(x, y)$ calculada en los puntos medios (x_i, y_y) de las aristas): Factor de Escala $= \left(\frac{\sum li^2}{\sum h(x_i, y_i)^2}\right)^{1/2}$.

La función h es especificada por el usuario. También puede ser calculada para tomar en cuenta automáticamente la irregularidad del dominio o la irregularidad de la solución para la implementación de un algoritmo adaptivo.

La posición de los nodos iniciales puede ser calculada de muchas maneras. Una de ellas es una distribución de puntos al azar. Para mallas con elementos regulares se disponen los puntos igualmente espaciados. Cuando se desea una distribución no uniforme $h(x, y)$ la convergencia es más veloz si se rechazan los puntos con probabilidad $\frac{1}{h(x,y)^2}$.

3.3. Explicación de Distmesh línea por línea

La primera línea especifica la convención de llamado a la función:

`function[p,t]=distmesh2d(fd,fh,h0,bbox,pfix,varargin)`

Esta función produce las siguientes salidas:

- La posición de los nodos p, un arreglo de N por 2 que contiene las coordenadas x e y de los N nodos.
- Los índices de los triangulos t. La fila asociada a cada triángulo tiene tres enteros, son los índices de los tres puntos en el arreglo de puntos p.

Las entradas son las siguientes:

- La geometría está dada por fd, que da la distancia signada al borde del dominio.
- La longitud relativa de las aristas viene dada por fh.
- El parámetro $h0$ especifica la distancia entre los puntos en la distribución inicial p_0.
- $bbox$ es una caja que contiene la región de interés, $bbox = [xmin, ymin; xmax, ymax]$.
- las posiciones de los nodos fijos están dadas por pfix.
- parámetros adicionales para fd y fh se dan en varargin.

Al comienzo del código se establecen 6 parámetros, estos son:

- *dptol*: si los movimientos totales de los nodos en una iteración son menores a este valor el algoritmos termina (ya convergió).
- *ttol*: controla cuanto pueden moverse los nodos antes de que ocurra una retriangulación de Delaunay.
- $Fscale$: controla la "presión interna".
- *deltat*: es el paso utilizado en el algoritmo de Euler.
- *geps*: es la tolerancia utilizada en los cálculos geométricos (i.e.: cualquier cosa a menos de *geps* de cero se considera nula).
- *deps*: la raíz cuadrada de la tolerancia de la máquina es utilizada para el dx en los cálculos de las derivadas espaciales (esto es óptimo para primeras diferencias unilaterales).

El código es:

```
dptol=.001; ttol=.1; Fscale=1.2; deltat=.2;
geps=.001*h0; deps=sqrt(eps)*h0;
```

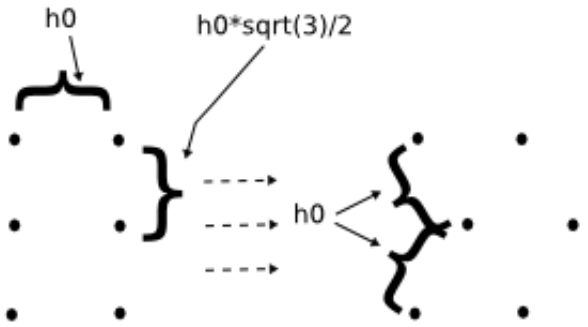
Los parámetros deps y geps se escalean con el tamaño de los elementos(h) para que no haya problemas de unidades. No sería coherente usar los mismos parámetros al mallar un planeta y al mallar una pelota de tenis. Es decir, un centímetro para un planeta es despreciable, en cambio para un objeto más pequeño es una distancia importante.

3.3.1. Distribución uniforme de puntos

El primer paso es crear una distribución uniforme de puntos dentro de la "bounding box" de la geometría, correspondiendo a triángulos equiláteros:

```
[x,y]=meshgrid(bbox(1,1):h0:bbox(2,1),bbox(1,2):h0*sqrt(3)/2:bbox(2,2));
x(2:2:end,:)=x(2:2:end,:)+h0/2; %Desplazar filas pares
p=[x(:),y(:)]; %Lista de las coordenadas de los nodos
```

La función *meshgrid* genera una grilla rectangular, dada por dos vectores x e y que dan las coordenadas de los nodos. Inicialmente las distancias son $\sqrt{3} * h_0/2$ en la dirección y. Al mover intercaladamente las filas de puntos $h_0/2$ hacia la derecha, todos los puntos estarán a una distancia h_0 de sus vecinos más cercanos. Las coordenadas se almacenan en un arreglo de 2 por N: p.



3.3.2. Eliminación de puntos exteriores al dominio y primera aproximación de la densidad de puntos deseada.

El próximo paso es descartar puntos fuera de la geometría:

```
p=p(feval(fd,p,varargin{:})<geps,:); %Conservar solo puntos con d<0
```

feval llama a la función fd y la evalúa en los puntos p, *varargin* son los parámetros adicionales que sirven para pasarle datos a una función de distancia construída a medida, por ejemplo. El resultado es un vector columna con las distancias signadas de cada uno de los nodos al borde de la geometría. Sólo los puntos interiores - con distancia negativa, usando tolerancia *geps* - son conservados. Luego calculamos $\frac{1}{h(x,y)^2}$ para cada nodo y rechazamos cada uno con esa probabilidad:

```
r0=1./feval(fh,p,varargin{:}).^2; % Probabilidad de quedarse con un punto
p=[pfix;p(rand(size(p,1),1)<r0./max(r0),:)]; % Metodo de rechazo de puntos
N=size(p,1); % Numero de puntos, N
```

Los puntos fijos se disponen en las primeras filas de los puntos.

3.3.3. Comienzo del bucle principal

Ahora el código entra en el bucle principal, allí la posición inicial de los N nodos es mejorada iterativamente. Se inicializa *pold* con la posición de los nodos y se comienza el bucle:

```
pold=inf; % para la primera iteración
while1
...
end
```

Antes de evaluar la función de fuerza una triangulación de Delaunay determina la topología. Esto se hace para p_0 y cada vez que se mueven los puntos. Para ahorrar en tiempo de cómputo, se utiliza una heurística aproximada que recalculará la triangulación cada vez que el máximo desplazamiento desde la última iteración es mayor que *ttol* (relativo al tamaño aproximado de los elementos l_0):

```
if max(sqrt(sum((p-pold).^2,2))/h0)>ttol %Algun movimientos grande?
  pold=p; % Salvar las posiciones actuales
  t = delaunayn(p); % Lista de triangulos
  pmid=(p(t(:,1),:)+p(t(:,2),:)+p(t(:,3),:))/3; % Computar los centroides
  t = t(feval(fd,pmid,varargin{:})<-geps,:); % Conservar triangulos interiores solamente
  ...
end
```

Las posiciones de los nodos luego de cada retriangulación son almacenadas en *pold*, y cada iteración compara las posiciones actuales de p con las anteriores *pold*. La función de MATLAB *delaunayn* genera una triangulación t de la cápsula convexa del conjunto de puntos y los triángulos fuera de la geometría son descartados. Para hacer esto calculamos el centroide de cada triángulo y descartamos aquellos que quedan afuera de la geometría. Esta técnica no es totalmente robusta, pero funciona bien en muchos casos y es muy fácil de implementar.

3.3.4. Creación de las barras

La lista de triángulos es un arreglo de tres columnas. Cada fila representa un triángulo definido por tres índices enteros. Al crear una lista de aristas cada triángulo contribuye tres pares de nodos. Como la mayoría de pares aparece dos veces (las aristas están en dos triángulos), los duplicados deben ser eliminados:

```
bars = [t(:,[1,2]);t(:,[1,3]);t(:,[2,3])]; % Barras interiores duplicadas
bars = unique(sort(bars,2),rows); % Barras unicas(pares de nodos)
```

3.3.5. Salida gráfica

Las próximas dos líneas dan feedback gráfico después de cada retriangulación:

```
trimesh(t,p(:,1),p(:,2),zeros(N,1))
view(2),axisequal,axisoff,drawnow
```

3.3.6. Creación de los vectores de fuerza y desplazamiento de los puntos

Cada barra es un vector de dos componentes, *barvec* contiene a todas las barras; sus longitudes están en L.

```
barvec = p(bars(:,1),:)-p(bars(:,2),:); % Lista de vectores barra
L = sqrt(sum(barvec.^2,2)); % L = longitud de las barras
```

Las longitudes deseadas vienen de evaluar $h(x, y)$ en el punto medio de las barras, multiplicamos por el factor de escala calculado y por $Fscale$, el factor de escala fijo para asegurar que la mayoría de las barras dan fuerzas repulsivas.

```
hbars = feval(fh,(p(bars(:,1),:)+p(bars(:,2),:))/2,varargin{:});
L0 = hbars*Fscale*sqrt(sum(L.^2)/sum(hbars.^2)); % L0 = Longitud deseada
F = max(L0-L,0); % Fuerza de cada barra(escalares)
```

La modificación de la posición de los nodos ocurre en el próximo bloque de código. La fuerza resultante *Ftot* es la suma de los vectores de fuerza en *Fvec* de todas las barras que desembocan en un nodo dado. Una fuerza de estiramiento lleva un signo positivo y su dirección viene dada por los dos componentes en *bars*. Se usa el comando sparse (a pesar de que *Ftot* es convertido inmediatamente en un array denso), por la útil propiedad de suma para índices repetidos.

```
Fvec = F./L*[1,1].*barvec; % Fuerzas de las barras(componentes x,y)
Ftot = full(sparse(bars(:,[1,1,2,2]),\
       ones(size(F))*[1,2,1,2],[Fvec,-Fvec],N,2));
Ftot(1:size(pfix,1),:)=0; % Fuerza = 0 en puntos fijos
p = p + deltat*Ftot; % Modificar la posicion de los nodos
```

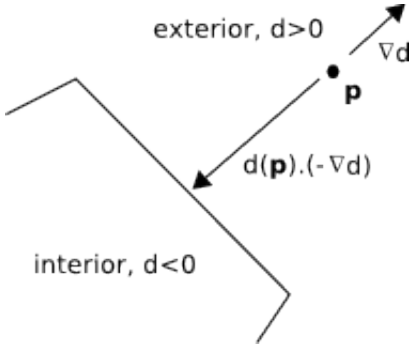
Ftot para los nodos fijos se pone nula para no moverlos.

3.3.7. Reproyección de puntos expulsados del dominio

Si un punto se mueve fuera de la geometría después de modificar p, es reproyectado hacia el punto más próximo del borde (usando la función de distancia). Esto corresponde a una fuerza de reacción normal al borde. Los puntos pueden moverse tangencialmente a lo largo del borde. El gradiente de $d(x, y)$ da la dirección (negativa) hacia el punto más cercano del borde y se obtiene derivando numéricamente:

```
d=feval(fd,p,varargin{:}); ix = d>0 ; % Identificacion
                          %de los puntos fuera del dominio(d>0)
dgradx = (feval(fd,[p(ix,1)+deps,p(ix,2)],varargin{:})-d(ix))
         / deps; % gradiente numerico
dgrady = (feval(fd,[p(ix,1),p(ix,2)+deps],varargin{:})-d(ix))
         / deps; % gradiente numerico
p(ix,:) = p(ix,:)-[d(ix).*dgradx,d(ix).*dgrady]; % Reproyeccion al borde
```

Esto funciona ya que, por ser una función de distancia, su gradiente (donde existe) cumple un caso particularmente simple de la ecuación Eikonal: $|\nabla d| = 1$.



3.3.8. Criterio de terminación del algoritmo

Finalmente, el criterio de terminación se basa en el máximo movimiento de un nodo en la iteración presente (excluyendo los puntos del borde):

```
if max(sqrt(sum(deltat*Ftot(d<-geps,:).^2,2))/h0) < dptol , break ; end
```

Este criterio es a veces demasiado estricto y se alcanza una malla de alta calidad mucho antes de que se cumpla. En estos casos el programa puede ser detenido manualmente, o se pueden usar otros tests. Un test simple es calcular la calidad de todos los elementos y terminar el programa si la menor de las calidades es suficientemente grande. El programa produce en 2 dimensiones mallas de muy alta calidad, con aristas de longitud muy similar entre sí.

3.4. Adaptación del algoritmo a 3 dimensiones

Con muy pocas modificaciones el programa funciona en 3 dimensiones. Lo único que se hace es modificarlo levemente para generar la grilla y modificar la salidad gráfica. Para la distribución inicial se utiliza una grilla regular de puntos. El Factor de Escala entre las barras sin estiramiento y el promedio real de sus longitudes ($Fscale$) es un parámetro importante y se utiliza un valor empírico que funciona bien: $1 + \frac{0{,}4}{2^{(dim-1)}}$.

En 3 dimensiones surge un nuevo problema, y es que un tetraedro puede tener la longitud de sus lados muy similar entre sí pero tener volumen casi nulo. Estos elementos pueden causar problemas cuando resolvemos un problema de elementos finitos. Todos los generadores de mallas obtenidas con Delaunay sufren de este problema. Sin embargo se han desarrollado técnicas para eliminar elementos malos de este tipo. Este problema está más allá de los alcances de esta tesis. Afortunadamente los resultados numéricos obtenidos cuando hay pocos elementos malos son buenos, si bien la teoría clásica no explica este fenómeno.

3.5. Construcción de otras funciones de distancia por intersección, unión y resta de varios objetos

Las funciones dunion, ddiff y dintersect combinan geometrías de manera aproximada. Supongamos que tenemos dos regiones A y B definidas por funciones de distancia d_A y d_B, entonces definimos:

$$d_{A\cup B}(x,y) = min(d_A(x,y), d_B(x,y))$$

$$d_{A\setminus B}(x,y) = max(d_A(x,y), -d_B(x,y))$$

$$d_{A\cap B}(x,y) = max(d_A(x,y), d_B(x,y))$$

Ahora mostraremos algunos ejemplos en tres dimensiones generados con funciones definidas en Matlab:

Ejemplo. Esfera unitaria en tres dimensiones

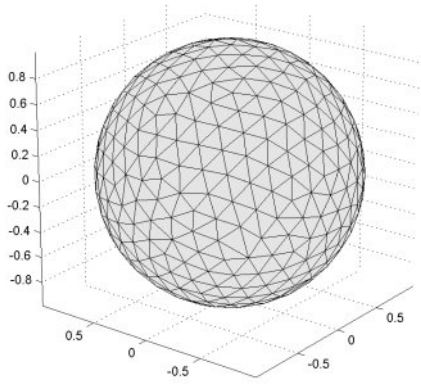


Figura 2: Exterior

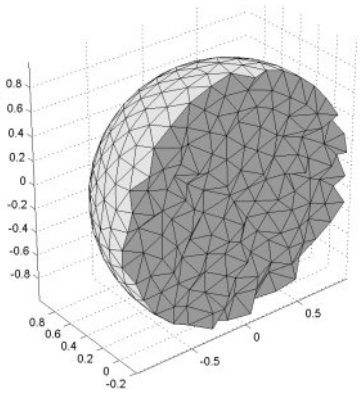


Figura 3: Interior

El código para generar esta malla es muy simple:

```
fd=inline('sqrt(sum(p.^2,2))-1','p');
[p,t]=distmeshnd(fd,@huniform,0.15,[-1,-1,-1;1,1,1],[]);
```

Ejemplo. Cilindro con un hoyo esférico

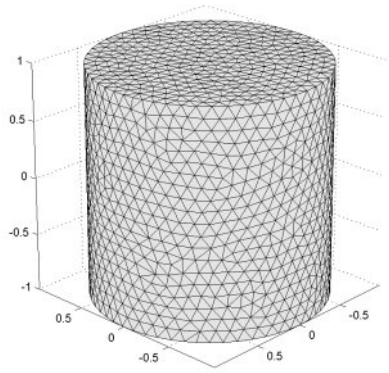


Figura 4: Exterior

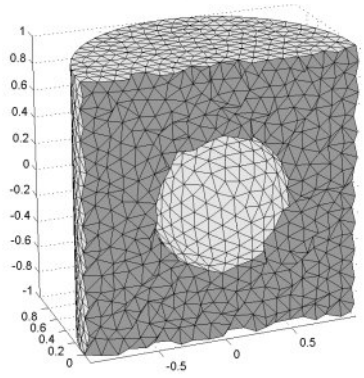
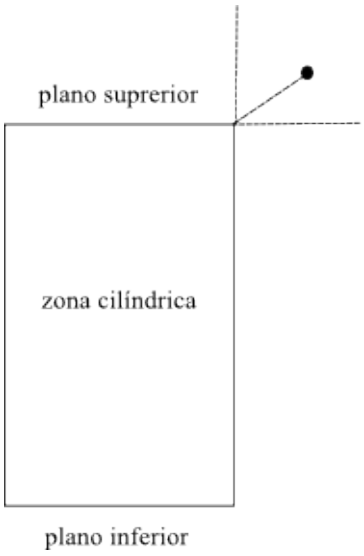


Figura 5: Interior

Para generar esta malla se realiza la diferencia entre un cilindro y una esfera. El cilindro se construye como intersección de dos regiones delimitadas por planos paralelos y una región cilíndrica infinita. En este caso hay que tener cuidado en los bordes que comparten los planos y la región cilíndrica ya que la función de intersección antes mencionada es sólo una aproximación. En efecto, la distancia que se realiza al punto de la figura que se ilustra más abajo no es la mínima entre las tres(al plano inferior, al plano superior o al cilindro) sino que es la raíz cuadrada de la suma de los cuadrados de las distancias al plano superior y al cilindro.



3.6. Medida de la calidad de los elementos

En la sección introductoria se mencionó que la calidad de la aproximación depende de que el cociente $\frac{h_k}{\rho_K}$ esté acotado. Por lo tanto, si resulta que el recíproco $\frac{\rho_k}{h_K}$ es grande para un elemento dado, podemos decir que ese elemento tiene buena "calidad". Con eso en mente definimos la siguiente medida de "calidad" de un elemento:

$$q_K = 3\frac{\rho_k}{h_K}$$

El factor 3 provoca que esta medida valga 1 para un tetraedro regular, que es el elemento más deseable. Para cualquier otro elemento será menor que 1. El cálculo de q_K se realiza fácilmente teniendo en cuenta las siguientes fórmulas para el inradio r y el circumradio R de un tetraedro definido por cuatro puntos p_1, p_2, p_3, p_4 (ver [17]). Llamemos $a = p_2 - p_1$,$b = p_3 - p_1$, $c = p_4 - p_1$:

$$r = \frac{|a \cdot b \times c|}{|b \times c| + |c \times a| + |a \times b| + |(b \times c) + (c \times a) + (a \times b)|}$$

$$R = \frac{|a^2(b \times c) + b^2(c \times a) + c^2(a \times b)|}{2|a \cdot b \times c|},$$

para un tetraedro, entonces, la medida de calidad será $q = 3\frac{r}{R}$.

Utilizando estas fórmulas generamos histogramas de las calidades de los elementos de la esfera y el cilindro cuyo mallado se exhibió más arriba:

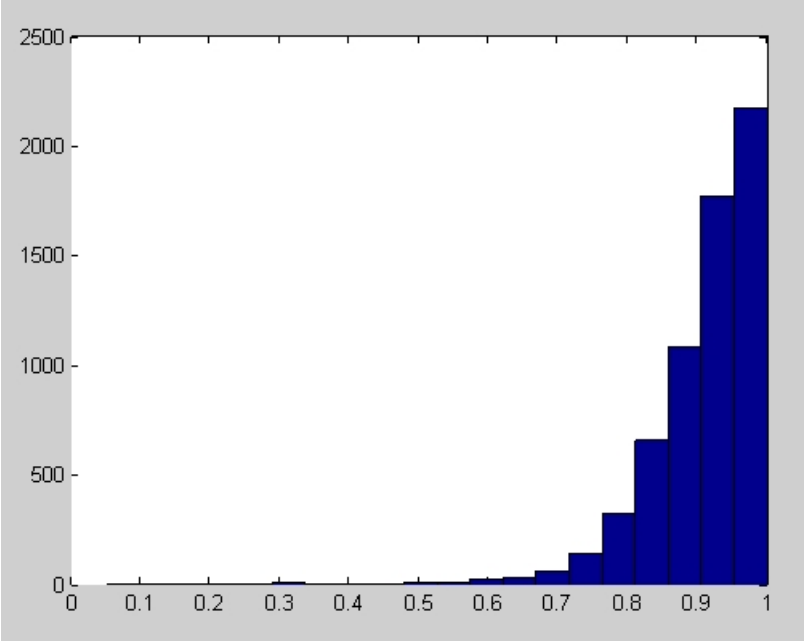


Figura 6: Histograma para la esfera

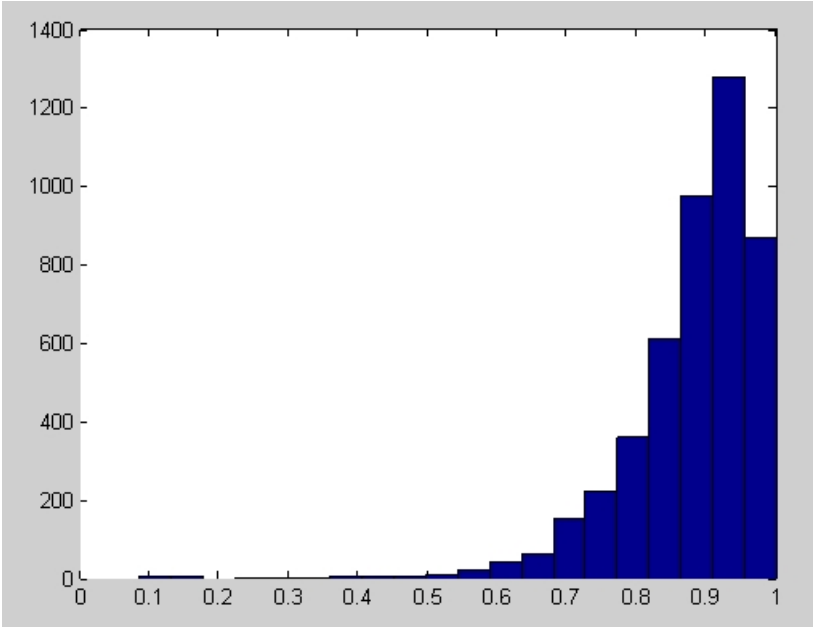


Figura 7: Histograma para el cilindro

En estas figuras se puede apreciar que la calidad de los elementos es muy buena.

4. Construcción de la función de distancia para dominios tridimensionales definidos por mallas de triángulos

Hemos descripto el funcionamiento del mallador que vamos a utilizar y exhibido algunos ejemplos. La primera observación a realizar es que tomará un gran esfuerzo definir objetos complicados como intersección, unión y diferencia de objetos más simples utilizando las funciones antes descriptas. Por tal razón decidimos tomar un camino diferente para definir dominios más complejos.

Para capturar la geometría de objetos arbitrarios decidimos representar su contorno por un conjunto de triángulos. Esta representación es ampliamente utilizada en paquetes de CAD [7] o , en su defecto, la mayoría de estos paquetes pueden exportar en un formato que se adecúe a este requisito. Esto nos permite utilizar las avanzadas herramientas de software existentes para el diseño de objetos a la hora de construir nuestro dominio. La implicación es que dominios curvos serán aproximados en una primera etapa por triángulos definiendo el contorno y luego mallaremos esa aproximación. En algunos problemas esto será una restricción, pero en muchos casos esta técnica da buenos resultados.

El programa que diseñamos para realizar esta tarea acepta archivos en formato 3ds, que es un formato originalmente introducido por Autodesk en su programa 3d Studio y uno de los más ampliamente difundidos. Elegimos este formato porque sin importar el formato del objeto que queramos mallar encontraremos un programa de conversión que lo llevara al formato 3ds. El objeto a mallar debe cumplir ciertos requisitos. Tiene que ser un objeto cuyos triángulos no se intersequen excepto en sus aristas o en sus vértices y que tenga un interior y un exterior bien definidos.

Cuando importamos el objeto a nuestro programa obtenemos un arreglo de n_p puntos, cada uno con tres coordenadas en formato de punto flotante y un arreglo de n_t triángulos, cada uno definido por tres enteros que indexan el arreglo de puntos anterior. La convención utilizada para distinguir el interior del exterior del objeto es la siguiente. Dado un triángulo T, definido por sus tres índices al arreglo de puntos: $T = (p_1, p_2, p_3)$, la secuencia de puntos p_1, p_2, p_3 va en sentido antihorario vista desde afuera del objeto. Esto implica que el producto vectorial $(p_2 - p_1) \times (p_3 - p_1)$ apunta hacia afuera del objeto.

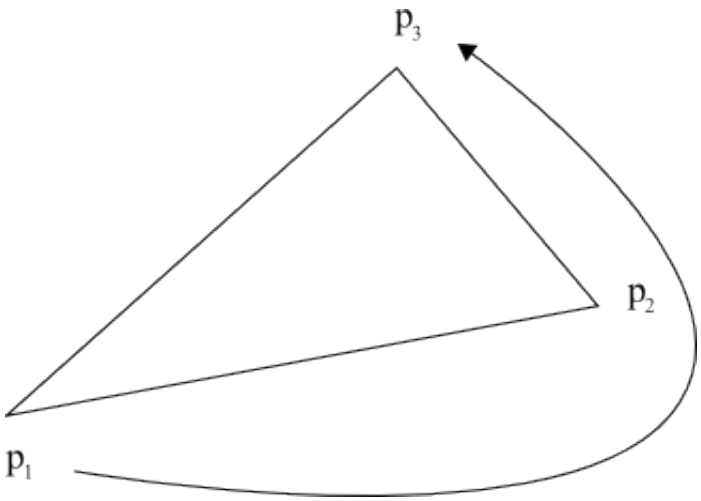


Figura 8: Triángulo visto desde afuera del objeto

Una vez definido el contorno por una malla de triángulos elegimos un prisma rectangular que contenga al objeto y dividimos el espacio encerrado por éste en una grilla rectangular con R, S y T divisiones en las dimensiónes x, y y z respectivamente. Calculamos la distancia signada desde cada una de las aristas de la grilla al objeto y almacenamos este arreglo de $R \times S \times T$ números. Utilizaremos este arreglo para obtener la distancia signada desde un punto arbitrario al objeto.

Dado un punto cualquiera, interpolamos trilinealmente los valores que aparecen en las aristas de la celda que contiene al punto en cuestión para obtener su distancia al contorno del objeto. De esta manera podemos rápidamente proveerle al mallador descripto en la sección anterior la distancia signada a una gran cantidad de puntos.

Los ejemplos siguientes muestran que esta técnica no sólo reproduce con facilidad dominios simples como los que habíamos exhibido previamente, sino que además permite mallar dominios mucho más complejos.

Ejemplo. Esfera unitaria en tres dimensiones

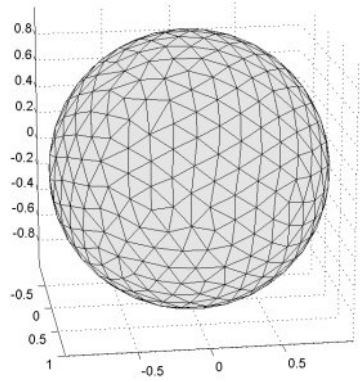


Figura 9: Exterior

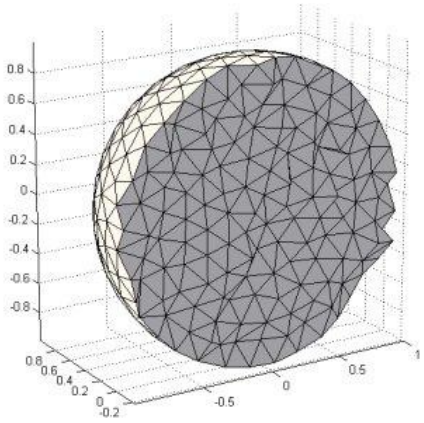


Figura 10: Interior

Ejemplo. Cilindro con un hoyo esférico

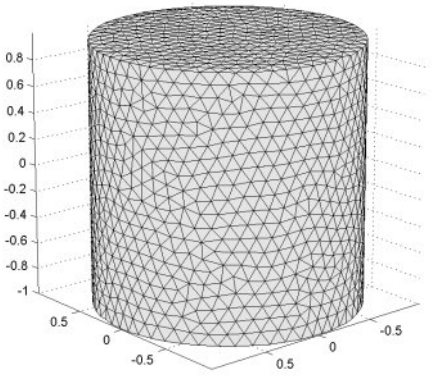


Figura 11: Exterior

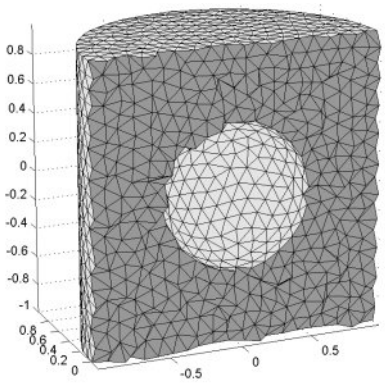


Figura 12: Interior

Ejemplo. Toro

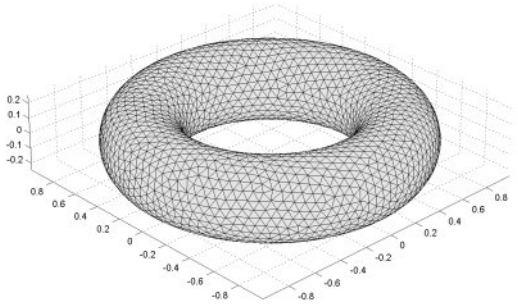


Figura 13: Exterior

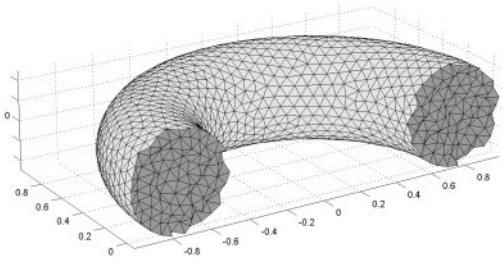


Figura 14: Interior

Ejemplo. Fémur

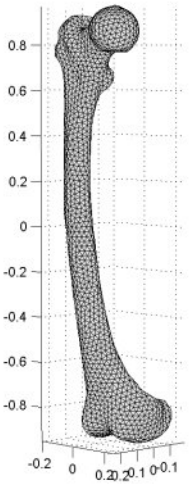


Figura 15: Exterior

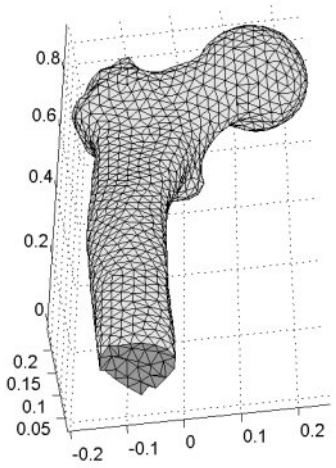


Figura 16: Interior

Ejemplo. Conejo

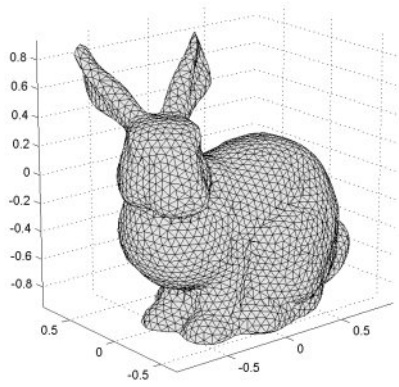


Figura 17: Exterior

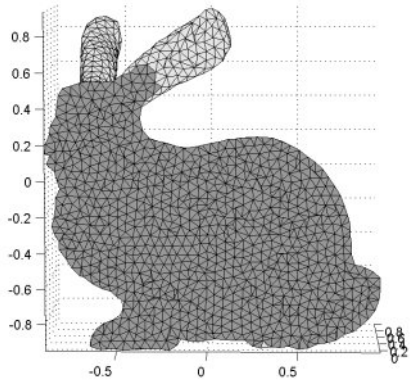


Figura 18: Interior

Ejemplo. Brida de vidrio

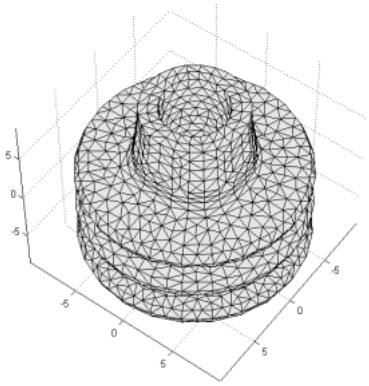


Figura 19: Exterior

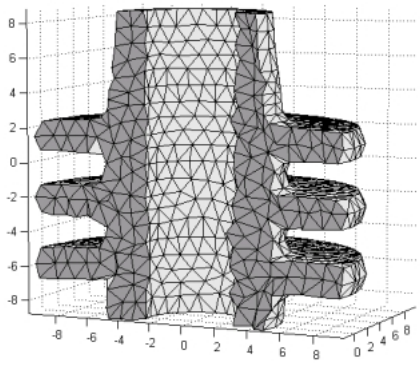


Figura 20: Interior

La interpolación trilineal es un caso particular de la interpolación en el espacio Q_1 sobre elementos cúbicos [3, 4]. En este sentido, se define como la interpolación sobre el espacio P_1 en un elemento de referencia y se obtiene en general vía cambio de variables.

El elemento de referencia será el cubo $\hat{C}$ con uno de sus vértices ubicado en el origen de coordenadas y su vértice diagonalmente opuesto ubicado en el punto $(1, 1, 1)$.

Supongamos que conocemos los valores nodales de cierta función $u(x, y, z)$ sobre las aristas del cubo $u_{000} = u(0, 0, 0)$, $u_{001} = u(0, 0, 1)$, $u_{010} = u(0, 1, 0)$, $u_{100} = u(1, 0, 0)$, $u_{011} = u(0, 1, 1)$, $u_{101} = u(1, 0, 1)$, $u_{110} = u(1, 1, 0)$, $u_{111} = u(1, 1, 1)$. Definimos las funciones de base $\hat{\phi_{ijk}}$, $0 \leq i, j, k \leq 1$

como sigue: $\hat{\phi_{000}}(x,y,z) = (1-x)(1-y)(1-z)$, $\hat{\phi_{100}}(x,y,z) = x(1-y)(1-z)$, y en general $\hat{\phi_{ijk}}(x,y,z) = (1-x)^{1-i}(1-y)^{1-j}(1-z)^{1-k}x^i y^j z^k$. Estas funciones tienen la cualidad de valer 1 en el nodo (la esquina del cubo) respectivo y ser nulas en los demás. Adicionalmente son lineales en cada coordenada tomada separadamente.

Con estas definiciones la interpoladora trilineal de u se expresa en el dominio de referencia como:

$$p(x,y,z) = \sum_{0 \le i,j,k \le 1} u_{ijk} \hat{\phi_{ijk}}.$$

Para un prisma cualquiera procedemos por cambio de variables. Considerando que estamos interesados en nuestro caso sólo en prismas con aristas paralelas a los lados llamamos

$$\hat{C} = [0,1] \times [0,1] \times [0,1] \quad \text{y} \quad C = [a_x, b_x] \times [a_y, b_y] \times [a_z, b_z]$$

al cubo de referencia y un prisma de genérico respectivamente. Observemos que la transformación afín $x = [a_x, a_y, a_z]^T + B\hat{x}$, mapea biyectivamente $\hat{C}$ en C, tomando la matriz B como:

$$B = \begin{bmatrix} b_x - a_x & 0 & 0 \\ 0 & b_y - a_y & 0 \\ 0 & 0 & b_z - a_z \end{bmatrix}$$

en consecuencia la transformación inversa, $\hat{x} = F^{-1}(x) = B^{-1}(x - [a_x, a_y, a_z]^T)$, con B^{-1} dado por

$$B^{-1} = \begin{bmatrix} \frac{1}{b_x - a_x} & 0 & 0 \\ 0 & \frac{1}{b_y - a_y} & 0 \\ 0 & 0 & \frac{1}{b_z - a_z} \end{bmatrix}$$

permite definir las funciones nodales en C. En efecto, $\phi_{ijk}(x,y,z) = \hat{\phi_{ijk}}(F^{-1}(x,y,z))$, resulta ser trilineal (ya que ϕ_{ijk} lo es y F^{-1} es afín con matriz diagonal) valiendo 1 en un nodo y 0 en los demás. En consecuencia, para una función u definida en C, escribimos su interpolada trilineal de la forma

$$p(x,y,z) = \sum_{0 \le i,j,k \le 1} u_{ijk} \phi_{ijk}. \tag{32}$$

Para interpolar trilinealmente una función cuyos valores conocemos en una grilla, simplemente realizamos la interpolación anterior en cada una de las celdas definidas por la grilla.

Ahora debemos entonces resolver el problema de construir la función de distancia signada al

objeto sobre una grilla volumétrica. La construcción de mapas de distancia ha sido un campo de estudio intenso en la última década [9], dadas sus múltiples aplicaciones. Es por eso que han surgido una serie de algoritmos. Nuestro primer acercamiento al problema será simple y luego paso a paso trataremos de refinarlo.

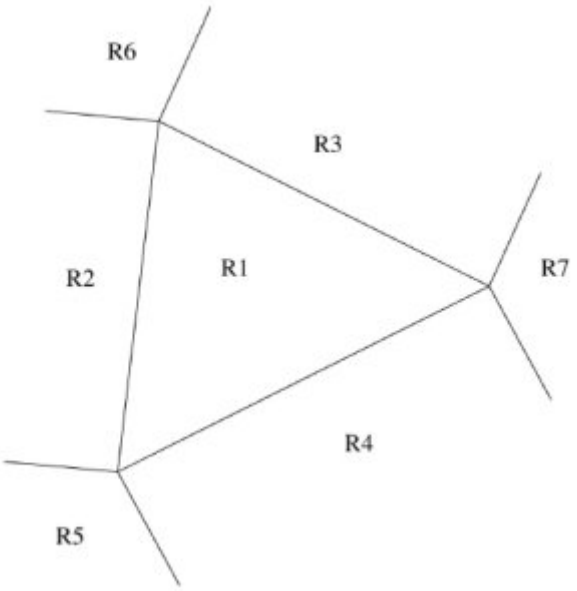
4.1. Algoritmo de fuerza bruta

El primer algoritmo utilizado para calcular la grilla de distancias es lo que se llama "brute-force"; por cada punto de la grilla calcula su distancia a todos los triángulos, de esta manera se identifica el triángulo más cercano y se almacena su índice, junto con la distancia realizada.

---

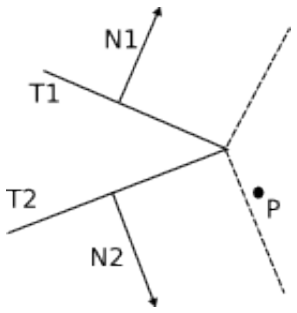
para todo p punto de la grilla
  dmin=inf
  para todo t triangulo del objeto
    calcular d, la distancia de p a t
    si d < dmin **entonces**
      dmin=d
      minindex=indice de t
    fin si
  fin para
fin para

Para calcular la distancia a un triángulo hay que distinguir entre los casos posibles:



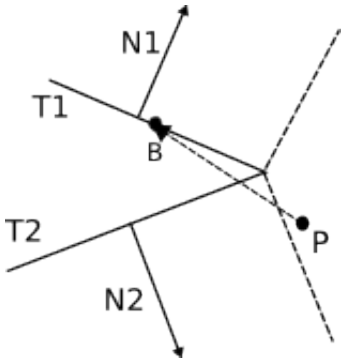
La figura ilustra un triángulo visto perpendicularmente desde arriba. Dado un punto al cual queremos calcular la distancia desde el triángulo, la fórmula a utilizar dependerá de la región del espacio donde éste se encuentra. Se divide el plano del triángulo como se muestra en la figura, las regiones espaciales de interés son aquellas definidas al desplazar la división del plano exhibida perpendicularmente a éste. Si el punto al cual queremos sacar la distancia se encuentra en las regiones R_5, R_6 o R_7 hay que obtener la distancia al vértice correspondiente; si se encuentra en las regiones R_2, R_3 o R_4 hay que obtener la distancia a la recta correspondiente; si se encuentra en la región R_1 hay que obtener la distancia al plano del triángulo.

Para decidir si un punto está adentro o afuera del objeto un procedimiento coherente sería el siguiente: se toma el triángulo más cercano y se decide si este triángulo está "mirando" al punto - es decir, su normal apunta en su dirección- , si es así el punto está afuera del objeto y la distancia es positiva. Si no es así el punto se encuentra en el interior del objeto y su distancia es negativa. El problema con este procedimiento es que si la distancia se realizó a una arista o a un vértice en vez de al plano del triángulo, esto puede no funcionar:



En este caso la mímina distancia se realiza a T_1 y a T_2 simultáneamente, si el índice que almacenamos es el de T_1 entonces diremos que P está adentro del objeto cuando en realidad está afuera.

Para solucionar este incoveniente consideramos el vector que une P con el baricentro B de T_1, si este vector interseca a algún otro triángulo T' vecino a T_1 antes que a T_1 mismo(en este caso T_2), la decisión de si P está adentro o afuera del objeto se tomará respecto a la normal de T'.



4.2. Optimización del cálculo de la grilla de distancias

Si N es el número de triángulos, entonces resulta que el algoritmo de la sección anterior es $O(N \times R \times S \times T)$, este número crece muy rápidamente a medida que aumentamos la resolución deseada. La virtud que tiene este algoritmo es que es extremadamente paralelizable. Es por eso que decidimos, luego de haberlo implementado en una CPU tradicional, implementarlo en una placa de video.

En los últimos años ha surgido en el campo de la informática un nuevo concepto denominado GPGPU (General Purpose computing on Graphics Processing Units) que intenta utilizar el alto poder paralelo (muchas operacionas idénticas en diferentes datos, un paradigma denominado SIMD, Single Instruction Multiple Data) existente en algunas placas de video actuales para realizar cálculos generales. Esto es posible gracias a que durante la última década su capacidad de procesamiento

aumentó enormemente (al intervenir cada vez más en los cálculos geométricos e impulsada por la economía de escala ligada a la industria del entretenimiento) y al agregado de etapas programables (y mayor precisión de los tipos de datos) en el "pipeline" geométrico, [11]. En particular decidimos utilizar CUDA (Compute Unified Device Architecture), una versión de C con algunas extensiones diseñada por la compañía NVIDIA para sus tarjetas aceleradoras y liberada al público a comienzos del 2007. La placa utilizada para correr el código compilado fue una placa de video NVIDIA Geforce 8800 GTS 512. Esta placa cuenta con 512 MB de memoria DDR3 onboard, un bus de datos de 256-bit y 128 procesadores "stream" trabajando en paralelo con una velocidad de reloj interno de 1650 Mhz. Dado que cada procesador es capaz de realizar una multiplicación-adición y otra multiplicación por ciclo de reloj esto da una capacidad de procesamiento teórica de: 633,6 GigaFLOPS= $(2+1) \times 1650MHz \times 128$ (FLOPS=FLoating point Operations Per Second). Los procesadores más rápidos de Intel (quad-core) llegan hoy en día a los 30-40 GigaFLOPS. Obviamente la flexibilidad de una CPU tradicional es mucho mayor que la que ofrece una GPU, pero si el algoritmo utilizado es altamente paralelizable las mejoras pueden ser fenomenales.

Los pasos que uno realiza habitualmente para implementar un programa en la GPU son:

1. La CPU transfiere los datos a la GPU
2. La GPU procesa los datos de manera independiente
3. Los resultados son retransferidos a la CPU desde la GPU

Como se dijo antes, la GPU consiste de varios procesadores trabajando en paralelo. Las porciones paralelas de una aplicación se ejecutan en la GPU como "kernels" (un mismo programa puede contar con varios kernels, cada uno cumpliendo una función distinta) , en cada instante están en ejecución muchas instancias de un mismo kernel. Cada una de esas instancias se denomina "thread" (en español hilo). La eficiencia de la GPU se basa en que ejecuta miles de threads simultáneamente.

Cada kernel es ejecutado como una grilla de bloques de threads. Un bloque de threads es un lote de threads que se ejecutará en paralelo.

La grilla de bloques puede tener una o dos dimensiones y cada bloque de la grilla de bloques tendrá sus correspondientes índices. De igual manera cada bloque puede tener 1, 2 o 3 dimensiones y cada thread de un bloque tendrá sus correspondientes índices. La dimensión máxima de la grilla dependerá del dispositivo, en nuestro caso las dimensiones máximas son 65535×65535. Las dimensiones máximas de los bloques son $512 \times 512 \times 64$ y deben ser tales que la cantidad total (la multiplicación de las 3 dimensiones) no sea mayor a 512, [10].

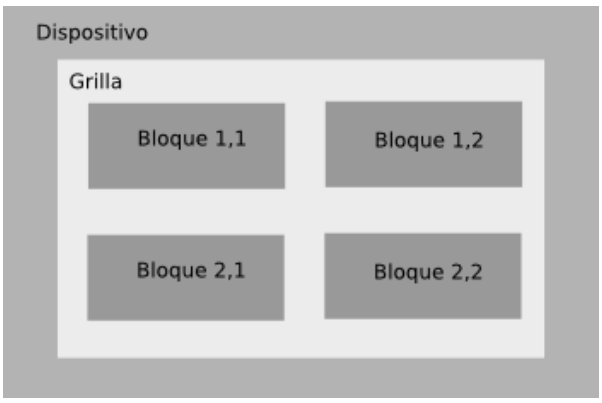


Figura 21: Grilla de bloques

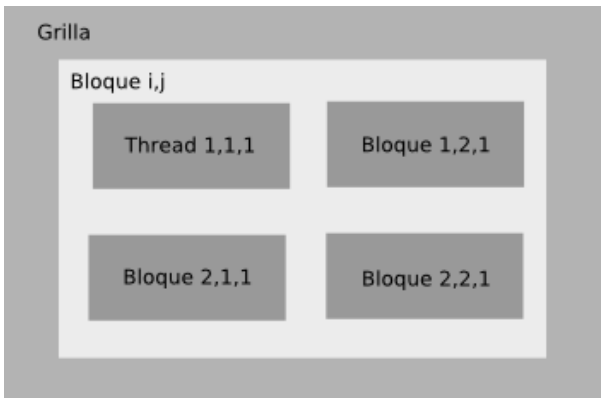


Figura 22: Bloque de threads

Lo que hicimos para paralelizar nuestro algoritmo es lo siguiente:

1. Inicializamos las estructuras de datos (triángulos y puntos) en la CPU y las transferimos a la GPU

2. Procesamos en paralelo todos los puntos de la grilla espacial, siendo el tamaño de la grilla de bloques $R \times S$ y el tamaño de cada bloque de threads $T \times 1 \times 1$, cada kernel consiste en un bucle a través de todos los triángulos que definen al objeto.

3. Copiamos la grilla de distancias desde la GPU a la CPU.

El pseudocódigo de cada kernel es:

---

dmin=inf
para todo t triangulo del objeto
  calcular d, la distancia de p a t
  si d < dmin **entonces**
    dmin=d
    minindex=indice de t
  fin si
fin para

Cada bloque de threads procesa todos los puntos con $x = cte$ e $y = cte$, es decir el índice de cada thread en un bloque indica la profundidad en z. Los dos índices (i, j) de un bloque dentro de la grilla de bloques indican las coordenadas x e y respectivamente de las que se ocupa ese bloque. En la figura se aprecia el funcionamiento del programa.

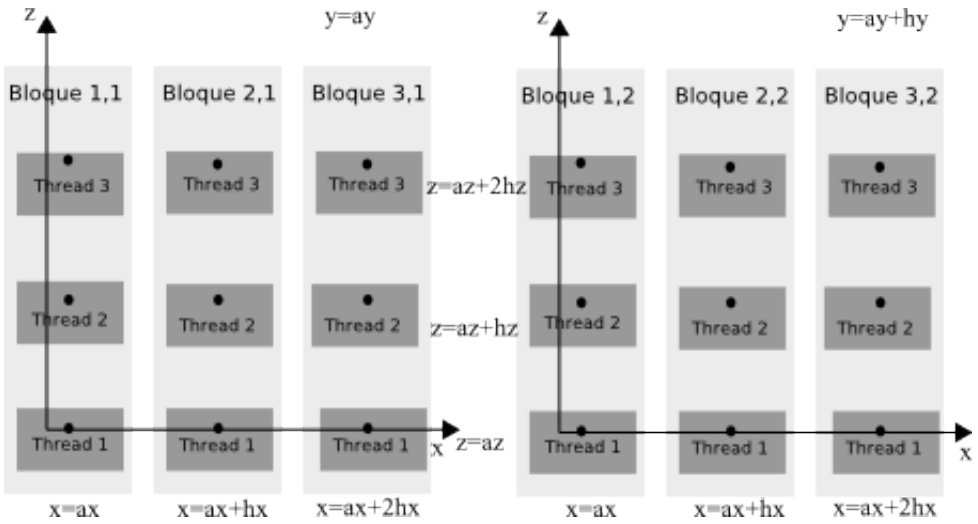
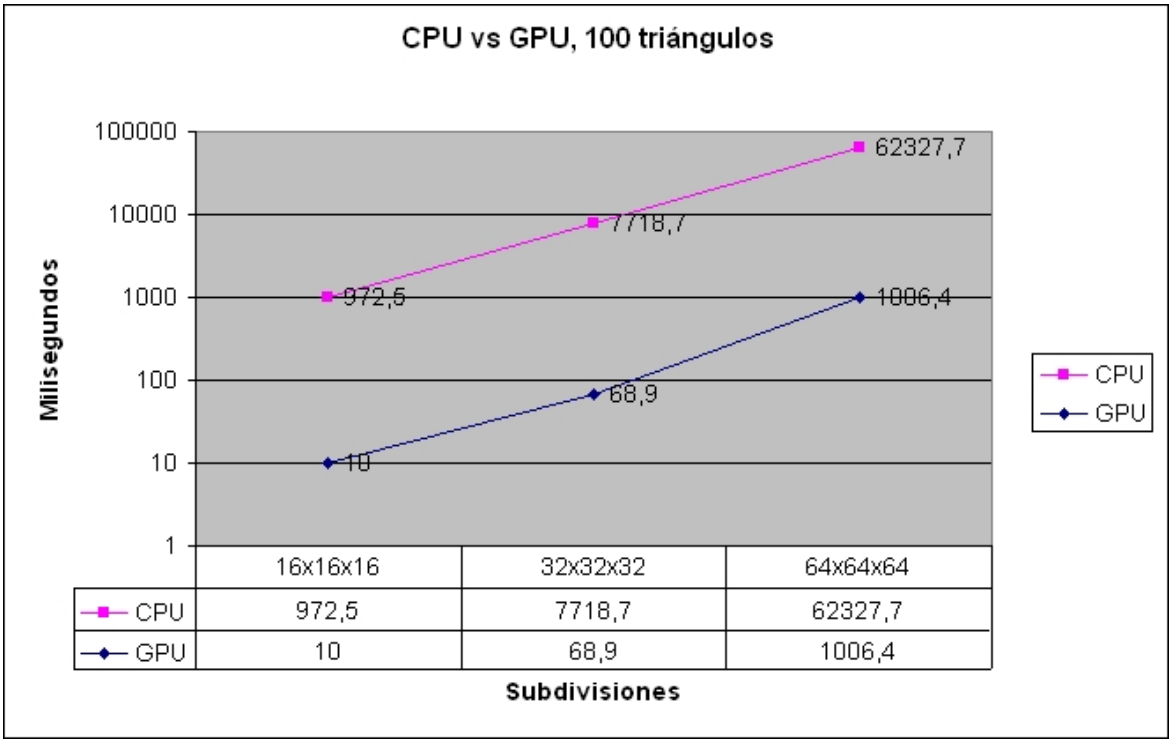


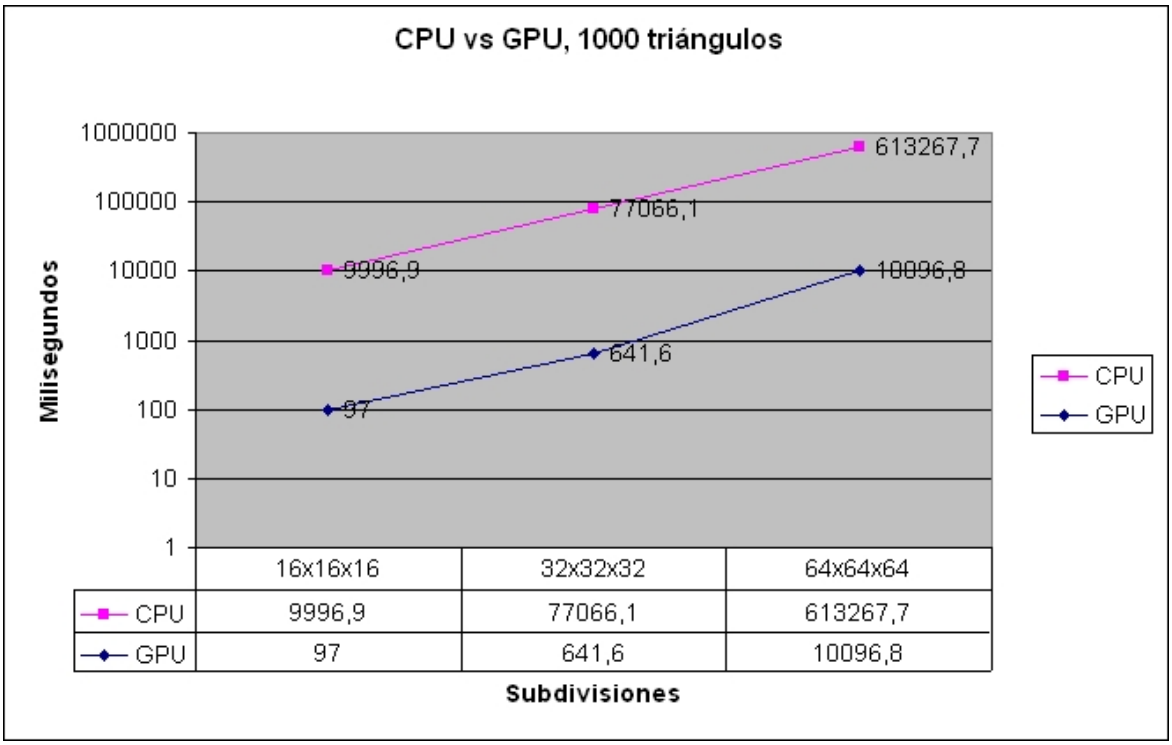
Figura 23: Visión esquemática del algoritmo

En este gráfico a_x, a_y, a_z representan la coordenadas iniciales de la grilla en x, y y z respectivamente. h_x, h_y, h_z representan el espaciado de la grilla en las coordenadas x, y y z respectivamente.

4.3. Speed-up de la versión paralela del algoritmo

En los gráficos que siguen se ve claramente que la optimización lograda con la nueva implementación es de alrededor 100 veces, una mejora muy significativa.


CPU vs GPU, 100 triángulos
Milisegundos
100000
10000
1000
100
10
1
62327,7
7718,7
972,5
1006,4
68,9
10
CPU
GPU
16x16x16
32x32x32
64x64x64
CPU 972,5 7718,7 62327,7
GPU 10 68,9 1006,4
Subdivisiones


CPU vs GPU, 1000 triángulos
Milisegundos
1000000
100000
10000
1000
100
10
1
613267,7
77066,1
9996,9
10096,8
641,6
97
CPU
GPU
16x16x16
32x32x32
64x64x64
CPU 9996,9 77066,1 613267,7
GPU 97 641,6 10096,8
Subdivisiones

4.4. Segundo paso en la optimización

Como se mencionó antes, durante la última década han surgido numerosos algoritmos para obtener el mapa discreto de distancias a un objeto. Estos algoritmos tratan de explotar la coherencia espacial de la función de distancia (es Lipschitz con constante 1). Algunos de ellos intentan descartar la cantidad de triángulos a los cuales hay que calcular la distancia desde un punto de la grilla dado (por ejemplo almacenando la superficie en una estructura de árbol que permite podar ramas rápidamente), otros obtienen la distancia cerca de la superficie y luego propagan el resultado al resto del volumen (por ejemplo a través de un esquema de diferencias finitas upwind), [9].

Para esta tesis decidimos hacer uso de una simple observación que acelerará el funcionamiento del algoritmo considerablemente para nuestros propósitos.

Lo primero que observamos es que el mallador necesita la función de distancia en dos instancias:

1. Para descartar los puntos fuera del objeto incialmente.

2. Para reproyectar puntos que han sido desplazados fuera del objeto en una iteración del algoritmo de Euler.

Sólo en el segundo caso es necesaria la función de distancia exacta, y además se necesita la distancia solamente cerca del borde. En el primer caso conocer el signo es suficiente. Por eso aproximaremos la función de distancia cerca de la superficie del objeto.

Consideremos un triángulo individual y encerrémoslo en una caja con lados paralelos a los ejes de la grilla. Si ahora expandimos esa caja en todas las direcciones una distancia d entonces está garantizado que la superficie de nivel d de la función distancia no signada de este triángulo estará contenida en esta nueva caja.

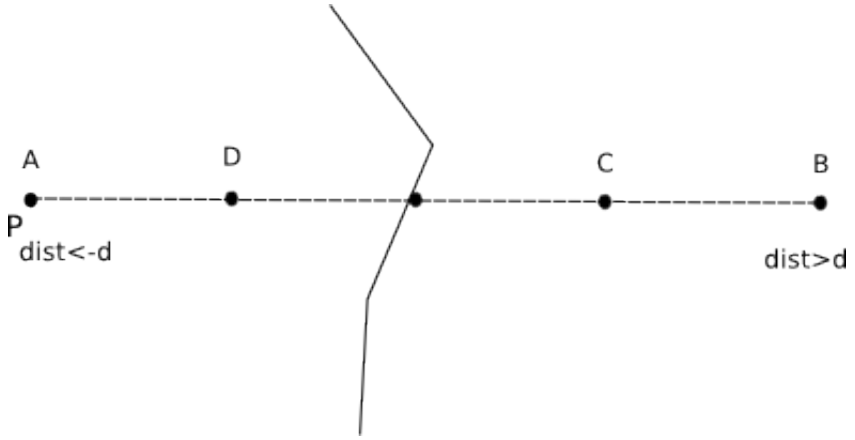
El algoritmo entonces tomará cada triángulo y calculará una caja conveniente, luego tomará cada uno de los puntos de la grilla contenidos en la caja y obtendrá su distancia al triángulo, si resulta que esta distancia es la menor realizada hasta el momento entonces la almacenará como la distancia mínima para este punto. Este procedimiento garantiza que los puntos de la grilla a menor o igual distancia que d de la superficie tengan calculada la distancia correcta. El pseudocódigo del algoritmo es el siguiente:

---

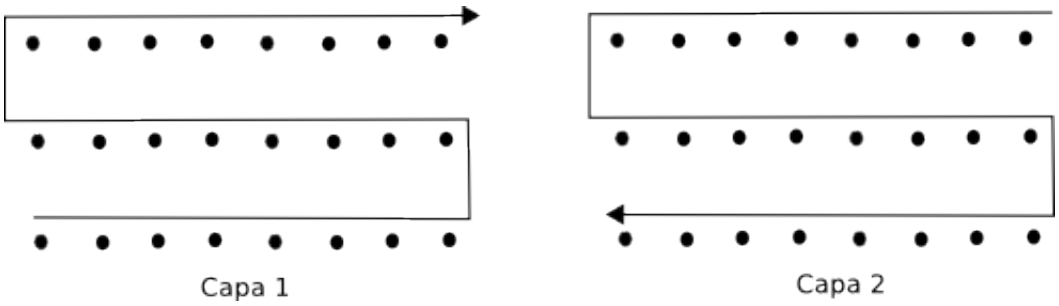
para todo t triangulo del objeto
  dmin=inf
  para todo p punto de dentro de la caja correspondiente

```
    calcular d, la distancia de p a t
    si  d < dmin entonces
      dmin=d
      minindice=indice de t
    fin si
  fin para
fin para
```

Finalmente debemos decidir si cada punto está afuera o adentro del objeto. Lo primero que hacemos es tomar los puntos que están a distancia menor o igual que d de la superficie del objeto. Sabemos que la distancia que se le ha asignado a estos puntos es correcta y que tenemos el índice del triángulo más cercano a cada uno de ellos, por lo tanto con ellos podemos utilizar el procedimiento mencionado en el algoritmo anterior. Todos los puntos que no tienen una distancia menor o igual que d (tanto aquellos para los cuales se ha calculado la distancia a algún triángulo como los que no) son identificados asignándoles una distancia muy grande (por ejemplo mayor a el diámetro de la grilla entera). De esta manera hemos separado a la grilla en dos partes, una en la que sabemos exactamente la distancia signada y otra en la que no. No obstante, para el primer paso del algoritmo de mallado (eliminación de los puntos fuera del dominio) es necesario saber el signo de la distancia en todos los puntos. Describiremos ahora cómo realizamos ese paso.
La observación es que si d es mayor que el diámetro de las celdas de la grilla entonces los puntos interiores del objeto que aún no tienen asignado su signo estarán "separados" en rectas paralelas a los ejes de la grilla de los puntos del objeto exteriores que aún no tienen asignado su signo. En efecto, si un punto de la grilla exterior al objeto con $x = k_1$ e $y = k_2$ no tiene asignada una distancia quiere decir que tiene una distancia signada mayor a d y si un punto de la grilla interior al objeto con $x = k_1$ e $y = k_2$ no tiene asignada una distancia quiere decir que tiene una distancia signada menor a $-d$ es decir la distancia entre ellos es mayor a $2d$ y por lo tanto hay al menos dos puntos de la grilla con $x = k_1$ e $y = k_2$ entre ellos que tienen la distancia correcta asignada. Aún más, uno de ellos tiene que tener distancia positiva y otro distancia negativa. En el dibujo que sigue se ilustra la situación:



Para demostrar esto llamemos B al punto con distancia mayor a d y consideremos el punto de la grilla justo a la izquierda de B, llamémoslo C. Como la distancia es una función contínua, es contínua en el segmento $\bar{BC}$, si en C la distancia fuera negativa entonces habría un punto Z entre B y C tal que la distancia allí es nula. Pero entonces Z es un punto de la superficie y como Z está a una distancia menor que d de B entonces B estaría a una distancia menor que d de la superficie, lo cual es absurdo, ya que tiene una distancia signada mayor a d. De la misma manera el punto D justo a la derecha de A tiene una distancia signada negativa. Esto demuestra lo afirmado antes. El algoritmo utilizado es entonces el siguiente. Se toma la capa de puntos de la grilla con menor z y se la recorre "zigzagueando", como ilustra la figura:

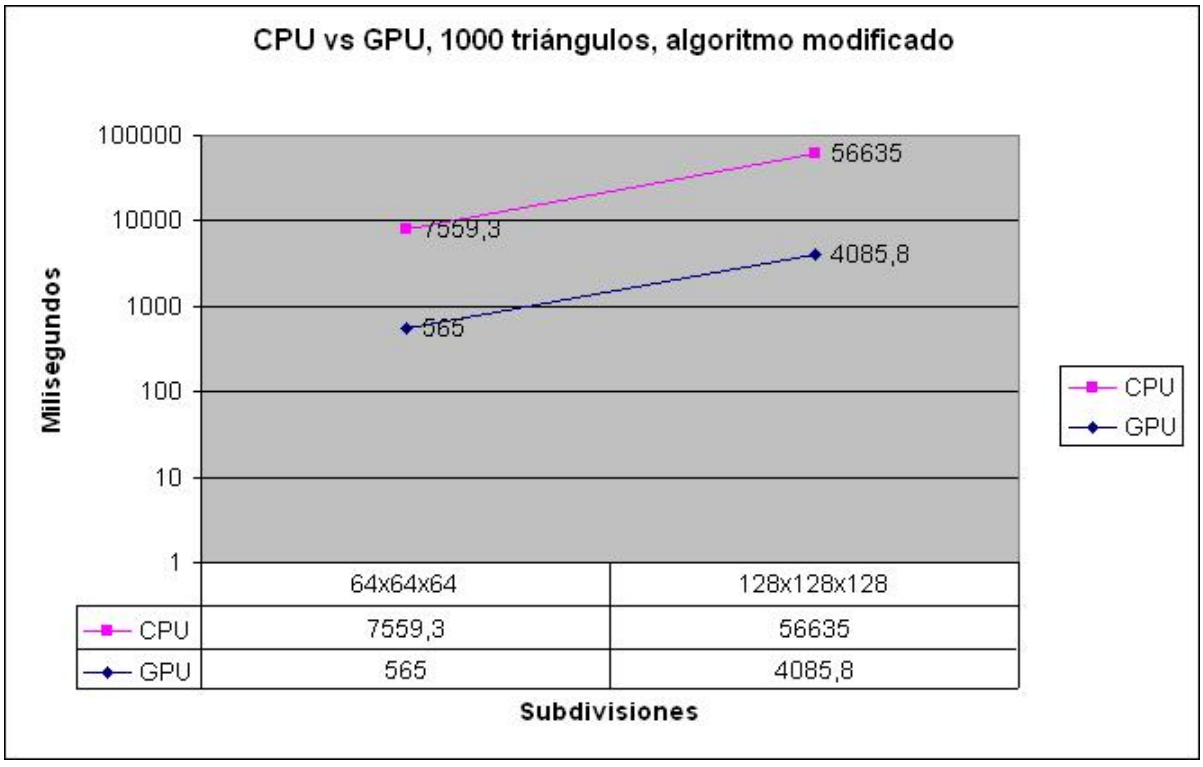


Una vez completada esa capa se comienza la capa justo adyacente, pero desde el lado "superior" y desde la derecha o la izquierda dependiendo de donde se culminó la capa precedente(dependerá de la paridad de los nodos). De esta manera se recorre toda la grilla. Al comienzo del recorrido, como la grilla contiene al objeto, sabemos que se comenzará con un punto que debe tener distancia positiva(y ése es el signo que le asignamos), cada vez que analizamos un nuevo punto, si su distancia no está asignada(es decir no sabemos su distancia exacta), se le asigna el signo de la

región del espacio correspondiente(interior o exterior al objeto). Si en cambio su distancia es conocida exactamente, no se modifica su signo y nos permite decidir en qué región del espacio nos encontramos. Al final de este paso todos los puntos tendrán el signo correcto.

4.5. Evaluación de la velocidad del nuevo algoritmo.

Con el objeto de evaluar cuanto tarda el nuevo algoritmo en producir un mapa de distancias parcial para ser utilizado con el mallador, fue implementado en dos versiones, una en la GPU y otra en la CPU. Decidimos calcular el mapa de distancias a un toro definido por 1000 triángulos, en una grilla de $64 \times 64 \times 64$ y luego otra de $128 \times 128 \times 128$, calculando la distancia exacta hasta una distancia d necesaria para que el mallador funcione (que en general es algo que se debe determinar empíricamente y depende del modelo a mallar). En el siguiente gráfico se exhiben los tiempos de corrida:



4.6. Cota del error cometido por la interpolación trilineal

Antes de pasar a la resolución de problemas prácticos demostraremos una cota máxima del error que se comete al realizar una aproximación trilineal de la función de distancia. Podríamos utilizar la acotación clásica para elementos Q_1 ([4]) porque la función de distancia es Lipschitz (está por lo tanto en $W^{1,\infty}$), sin embargo haremos la cuenta para obtener la constante de la acotación. Supongamos que el espaciado de la grilla en las tres direcciones es igual a h. Sea $d(x,y,z)$ la función de distancia real y sea $\Pi(x,y,z)$ la interpoladora trilineal en una celda C dada tal que $C = [a_x, a_x + h] \times [a_y, a_y + h] \times [a_z, a_z + h]$. Definamos $d_{ijk} = d(F((i,j,k)))$, con $i,j,k = 0,1$, y donde F es la transformación que lleva desde el cubo de referencia al cubo que nos ocupa; y sea $\Pi(x,y,z)$ la interpoladora trilineal en C. Es decir $\Pi(x,y,z) = \sum_{i,j,k} d_{ijk}\phi_{ijk}$, donde las ϕ_{ijk} son las funciones nodales y valen 1 en el nodo correspondiente y son nulas en los demás. Notemos que $\sum_{i,j,k} \phi_{ijk}(x,y,z) = 1$ en todo el volumen del cubo.

Sea $\tilde{x} \in C$ arbitrario. Tratemos de estimar el error cometido por la interpolación:

$$|\Pi(\tilde{x}) - d(\tilde{x})| = |\sum_{i,j,k} d_{ijk}\phi_{ijk}(\tilde{x}) - d(\tilde{x})| = |\sum_{i,j,k} d_{ijk}\phi_{ijk}(\tilde{x}) - \sum_{i,j,k} \phi_{ijk}(\tilde{x})d(\tilde{x})| =$$

$$|\sum_{i,j,k} [d_{ijk} - d(\tilde{x})]\phi_{ijk}(\tilde{x})| \tag{33}$$

Escribamos ahora la forma exacta de las funciones nodales ϕ_{ijk}. Según los visto al comienzo de esta sección

$$\phi_{000}(x,y,z) = \hat{\phi_{000}}(F^{-1}(x,y,z)) = \hat{\phi_{000}}(\frac{x-a_x}{b_x-a_x}, \frac{y-a_y}{b_y-a_y}, \frac{z-a_z}{g_z-a_z}) =$$

$$= (1 - \frac{x-a_x}{b_x-a_x})(1 - \frac{y-a_y}{b_y-a_y})(1 - \frac{z-a_z}{b_z-a_z}) = \frac{b_x-x}{b_x-a_x}\frac{b_y-y}{b_y-a_y}\frac{b_z-z}{b_z-a_z} = \frac{1}{h^3}(b_x-x)(b_y-y)(b_z-z),$$

de manera similar podemos escribir cada una de las funciones nodales ϕ_{ijk}. Por lo tanto, de (33), tomando en cuenta que la función distancia es Lipschitz con constante 1, i.e. $|d_{ijk} - d(\tilde{x})| \leq ||F(i,j,k) - \tilde{x}||$, tenemos :

$$|\sum_{i,j,k} [d_{ijk} - d(\tilde{x})]\phi_{ijk}(\tilde{x})| \leq \sum_{i,j,k} ||F(i,j,k) - \tilde{x}|||\phi_{ijk}(\tilde{x})|, \tag{34}$$

y en el lado derecho de la desigualdad hay 8 términos, uno por cada función nodal. Podríamos aquí decir que: (34) $\leq \sqrt{3}h$ ya que la máxima distancia entre dos puntos del cubo es $\sqrt{3}h$. Sin embargo intentaremos refinar un poco más la cota. Llamemos, para simplificar la notación, $\underline{x} = x - a_x$, $\underline{y} = y - a_y$, $\underline{z} = z - a_z$ y $\bar{x} = b_x - x = h - \underline{x}$, $\bar{y} = b_y - y$ y $\bar{z} = b_z - z$ (obsérvese que $0 \leq \underline{x}, \underline{y}, \underline{z}, \bar{x}, \bar{y}, \bar{z} \leq h$). Tomando en cuenta la simetría del lado derecho de (34) podemos suponer que $0 \leq \underline{z} \leq \underline{y} \leq \underline{x} \leq \frac{h}{2}$ y en consecuencia: $\frac{h}{2} \leq \bar{x} \leq \bar{y} \leq \bar{z} \leq h$. Para acotar al témino que involucra a ϕ_{000}, por ejemplo, realizamos lo siguiente:

$$||F(0,0,0) - \tilde{x}|||\phi_{000}(\tilde{x})| = \frac{1}{h^3}\sqrt{\underline{x}^2 + \underline{y}^2 + \underline{z}^2}\bar{x}\bar{y}\bar{z} \leq \frac{\sqrt{3}}{h^3}\underline{x}\bar{x}\bar{y}\bar{z} \tag{35}$$

Donde para eliminar la raíz cuadrada hemos utilizado las relaciones de orden entre los términos. El mismo procedimiento utilizado en (35) lo podemos utilizar para cada uno de los términos de (34), obteniendo:

$$\begin{array}{l}
\frac{1}{h^3}\sqrt{\bar{x}^2 + \underline{y}^2 + \underline{z}^2}\underline{x}\bar{y}\bar{z} \leq \frac{\sqrt{3}}{h^3}\bar{x}\underline{x}\bar{y}\bar{z} \\
\frac{1}{h^3}\sqrt{\underline{x}^2 + \bar{y}^2 + \underline{z}^2}\bar{x}\underline{y}\bar{z} \leq \frac{\sqrt{3}}{h^3}\bar{y}\bar{x}\underline{y}\bar{z} \\
\frac{1}{h^3}\sqrt{\underline{x}^2 + \underline{y}^2 + \bar{z}^2}\bar{x}\bar{y}\underline{z} \leq \frac{\sqrt{3}}{h^3}\bar{z}\bar{x}\bar{y}\underline{z} \\
\frac{1}{h^3}\sqrt{\bar{x}^2 + \bar{y}^2 + \underline{z}^2}\underline{x}\underline{y}\bar{z} \leq \frac{\sqrt{3}}{h^3}\bar{y}\underline{x}\underline{y}\bar{z} \\
\frac{1}{h^3}\sqrt{\bar{x}^2 + \underline{y}^2 + \bar{z}^2}\underline{x}\bar{y}\underline{z} \leq \frac{\sqrt{3}}{h^3}\bar{z}\underline{x}\bar{y}\underline{z} \\
\frac{1}{h^3}\sqrt{\underline{x}^2 + \bar{y}^2 + \bar{z}^2}\bar{x}\underline{y}\underline{z} \leq \frac{\sqrt{3}}{h^3}\bar{z}\bar{x}\underline{y}\underline{z} \\
\frac{1}{h^3}\sqrt{\bar{x}^2 + \bar{y}^2 + \bar{z}^2}\underline{x}\underline{y}\underline{z} \leq \frac{\sqrt{3}}{h^3}\bar{z}\underline{x}\underline{y}\underline{z}
\end{array} \tag{36}$$

Sumando todos los términos de la derecha en (36) obtenemos:

$$\begin{array}{rl}
\sum_{i,j,k} ||F(i,j,k) - \tilde{x}|||\phi_{ijk}(\tilde{x})| \leq \frac{\sqrt{3}}{h^3}(\underline{x}\bar{x}\bar{y}\bar{z} + \bar{x}\underline{x}\bar{y}\bar{z} + \bar{x}\bar{y}\underline{y}\bar{z} + \bar{x}\bar{y}\bar{z}\underline{z} \\
+\underline{x}\bar{y}\underline{y}\bar{z} + \underline{x}\bar{y}\bar{z}\underline{z} + \bar{x}\underline{y}\bar{z}\underline{z} + \underline{x}\underline{y}\bar{z}\underline{z})
\end{array} \tag{37}$$

ahora consideramos la parte entre paréntesis de (37) y sacamos factor común, el resultado es:

$$\underline{z}\bar{z}(\underline{x}\underline{y} + \underline{x}\bar{y} + \bar{x}\underline{y} + \bar{x}\bar{y}) + \underline{y}\bar{y}(\underline{x}\bar{z} + \bar{x}\bar{z}) + 2\underline{x}\bar{x}\bar{y}\bar{z} \tag{38}$$

el primer término de (38) es igual a $h^2\underline{z}\bar{z}$, donde se utilizó que $\underline{x} + \bar{x} = \underline{y} + \bar{y} = \underline{z} + \bar{z} = h$; el segundo término es igual a $h\underline{y}\bar{y}\bar{z}$. En otras palabras podemos ver que:

$$(38) = h^2\underline{z}\bar{z} + h\underline{y}\bar{y}\bar{z} + 2\underline{x}\bar{x}\bar{y}\bar{z}$$

y reagrupando esta última expresión:

$$(38) = \bar{z}[h^2\underline{z} + \bar{y}(h\underline{y} + 2\underline{x}\bar{x})] \tag{39}$$

la multiplicación $\underline{x}\bar{x}$ tiene como máximo valor $\frac{h^2}{4}$ y se alcanza ese valor en $\frac{h}{2}$ con lo cual poniendo $\underline{x} = \frac{h}{2}$, y teniendo en cuenta que esto no restringe los valores de $\underline{y}$ o $\underline{z}$, obtenemos que:

$$(38) \leq \bar{z}[h^2\underline{z} + \bar{y}(h\underline{y} + \frac{h^2}{2})]$$

Consideremos ahora la expresión $\bar{y}(h\underline{y} + \frac{h^2}{2}) = -h\underline{y}^2 + \frac{h^2}{2}\underline{y} + \frac{h^3}{2}$, esta expresión se maximiza en $\underline{y} = \frac{h}{2}$, y toma el valor $\frac{h^3}{2}$. Como asignar este valor a $\underline{y}$ no limita el movimiento de $\underline{z}$ podemos afirmar que:

$$(38) \leq \bar{z}(h^2\underline{z} + \frac{h^3}{2}) = -h^2\underline{z}^2 + \frac{h^3}{2}\underline{z} + \frac{h^4}{2}$$

Finalmente, ésta es una función cuadrática en $\underline{z}$, y su máximo valor se toma en $\underline{z} = \frac{h}{4}$, con lo cual podemos afirmar que:

$$(38) \leq \frac{9}{16}h^4 \tag{40}$$

Utilizando este resultado en (35) acotamos su valor:

$$(35) \leq \frac{9}{16}\sqrt{3}h$$

Cabe notar que el error de la aproximación puede ser mucho menor en general. En particular en las zonas del espacio donde la distancia mínima se realiza al plano de algún triángulo la función de distancia es afín y allí la aproximación es exacta.

5. Resolución por elementos finitos

Una vez armada la malla para el dominio de interés es momento de resolver el problema por el método elementos finitos. Para ello utilizamos un código, también en MATLAB, desarrollado por Jochen Alberty, Carsten Carstensen y Stefan A. Funken [1]. Al igual que el código de Per-Olof Persson, este código tiene la virtud de ser conciso y modificable, a diferencia de la mayoría de los programas comerciales. Está diseñado para la resolución de problemas elípticos con condiciones de borde mixtas (parte Dirichlet y parte Neumann).

5.1. Definición del problema a resolver

Sea $\Omega \subset R^3$ un dominio Lipschitz con borde Γ, consideramos el siguiente problema

$$\begin{array}{lll} -\Delta u = f & \text{en} & \Omega \\ u = u_D & \text{en} & \Gamma_D \\ \frac{\partial u}{\partial n} = g & \text{en} & \Gamma_N \end{array} \tag{41}$$

donde $f \in L^2(\Omega)$, $u_D \in H^1(\Omega)$ y $g \in L^2(\Gamma_N)$.

Suponemos que $\Gamma_D \subset \Gamma$ tiene longitud positiva (el código no considera condiciones de Neumann puras pues en ese caso la matriz de rigidez es singular y debe fijarse algun valor nodal para resolver el sistema), donde se asigna una condición de Dirichlet, y asumimos una condición de Neumann en el resto del borde: $\Gamma_N := \Gamma - \Gamma_D$. La condición no homogenea de Dirichlet en (41) se incorpora escribiendo $w = u - u_D$ por lo que w es cero sobre Γ_D. Si llamamos entonces $H^1_D(\Omega) := \{w \in H^1(\Omega) | w = 0 \text{ en } \Gamma_D \}$, buscamos $w \in H^1_D(\Omega)$ tal que:

$$\int_\Omega \nabla w \cdot \nabla v dx = \int_\Omega f v dx + \int_{\Gamma_N} g v ds - \int_\Omega \nabla u_D \cdot \nabla v dx \quad \forall v \in H^1_D(\Omega) \tag{42}$$

por el lema de Lax-Milgram, siempre existe una solución débil de (42), y la solución original u se recupera a través de $u = w + u_D$.

5.2. Discretización de Galerkin

Para su implementación, el problema es discretizado utilizando el método de Galerkin (expuesto en la primera sección de la tesis), donde $H^1_D(\Omega)$ es reemplazado por el espacio de dimensión finita $H_D = H_h \cap H^1_D$, siendo H_h el espacio de funciones lineales a trozos y continuas sobre una

triangulación adecuada. Sea $U_D \in H_h$ una función que aproxima u_D en Γ_D (por ejemplo U_D es el interpolador de u_D). Entonces el problema discretizado es el siguiente:

Encontrar $u_h \in H_D$ tal que

$$\int_\Omega \nabla u_h \cdot \nabla v_h dx = \int_\Omega f v_h dx + \int_{\Gamma_N} g v_h ds - \int_\Omega \nabla U_D \cdot \nabla v_h dx \ (\forall v_h \in H_D).$$

Sea $(\eta_1, ..., \eta_N)$ una base del espacio de H_h, y $I = \{i_1, \cdots, i_M\} \subset \{1, 2, \cdots, N\}$ un conjunto de indices tales que $\{\eta_{i_1}, \cdots, \eta_{i_M}\}$ resulte una base de H_D (notar que necesitamos determinar M coeficientes para definir una función de H_D, estos valores, como veremos, corresponden a los nodos interiores del dominio y los nodos Neumann, y como la condición no puede ser de Neumann pura tenemos que $M < N - 2$). Entonces, la ecuación anterior es equivalente a:

$$\int_\Omega \nabla u_h \cdot \nabla \eta_j dx = \int_\Omega f \eta_j dx + \int_{\Gamma_N} g \eta_j ds - \int_\Omega \nabla U_D \cdot \nabla \eta_j dx \ (\forall j \in I). \tag{43}$$

Además, como $u_h = \sum_{k \in I} x_k \eta_k$ y $U_D = \sum_{k=1}^N U_k \eta_k$, la ecuación anterior da como resultado un sistema lineal de ecuaciones:

$$Ax = B$$

La matriz de rigidez $A \in \mathbb{R}^{M \times M}$ y el lado derecho están definidos por:

$$A_{jk} = \int_\Omega \nabla \eta_j \cdot \nabla \eta_k dx \qquad (j, k \in I) \tag{44}$$

y

$$b_j = \int_\Omega f \eta_j dx + \int_{\Gamma_N} g \eta_j ds - \sum_{k=1}^N U_k \int_\Omega \nabla \eta_j \cdot \nabla \eta_k dx \qquad (j \in I) \tag{45}$$

La matriz de rigidez es esparsa, simétrica y definida positiva, de tal manera que este sistema tiene exactamente una solución $x \in \mathbb{R}^N$, que determina la solución de Galerkin $U_h = U_D + u_h$.

5.3. Representación de los datos de la triangulación de Ω

El dominio Ω se aproxima por un dominio poliedral P, suponemos que P puede ser cubierto por una triangulación regular T, i.e. $P = \bigcup_{T \in T} T$ [4].

Ahora describiremos las estructuras de datos utilizados. Los puntos de la triangulación están

almacenados en un arreglo de $N \times 3$. Los tetraedros de la triangulación están almacenados en un arreglo de $N \times 4$ enteros que indexan el arreglo de puntos. Finalmente hay dos arreglos de $N \times 3$, uno de ellos indica cuáles son las triángulos que están en el borde Dirichlet Γ_D y el otro cuáles están en el borde Neumann Γ_N. Estas estructuras las obtenemos, para un dominio particular, con el algoritmo de mallado presentado en la segunda sección de la tesis.

Podemos ahora, entonces, definir explícitamente las funciones nodales que serán base de H_D: $\eta_j(x_k, y_k, z_k) = \delta_{jk}$, o sea las fuciones lineales en cada elemento tetraedral y continuas en P que valen 1 en un nodo y 0 en los demas. Las integrales (44) y (45) se pueden calcular como suma sobre todos los elementos, i.e.,

$$A_{jk} = \sum_{T \in T} \int_T \nabla \eta_j \cdot \nabla \eta_k dx \qquad (j, k \in I)$$

y

$$b_j = \sum_{T \in T} \int_\Omega f \eta_j dx + \sum_{E \subset \Gamma_N} \int_E g \eta_j ds - \sum_{k=1}^{N} U_k \int_\Omega \nabla \eta_j \cdot \nabla \eta_k dx \qquad (j \in I)$$

5.4. Construcción de la matriz de rigidez

La matriz de rigidez local está determinada por las coordenadas de los vertices del elemento correspondiente. Para un elemento tetraedral T sean $(x_1, y_1, z_1), (x_2, y_2, z_2)$, (x_3, y_3, z_3) y (x_4, y_4, z_4) los vértices y η_1 , η_2, η_3 y η_4 las funciones nodales correspondientes en H_h, esto es:

$$\eta_j(x_k, y_k, z_k) = \delta_{jk}, \qquad j, k = 1, 2, 3, 4$$

Hay una forma compacta de escribir este valor:

$$\eta_j(x, y, z) = \begin{vmatrix} 1 & x & y & z \\ 1 & x_{j+1} & y_{j+1} & z_{j+1} \\ 1 & x_{j+2} & y_{j+2} & z_{j+2} \\ 1 & x_{j+3} & y_{j+3} & z_{j+3} \end{vmatrix} \Big/ \begin{vmatrix} 1 & x_j & y_j & z_j \\ 1 & x_{j+1} & y_{j+1} & z_{j+1} \\ 1 & x_{j+2} & y_{j+2} & z_{j+2} \\ 1 & x_{j+2} & y_{j+2} & z_{j+3} \end{vmatrix}$$

En efecto, esta es una función lineal en cada elemento que comparte el nodo j, que vale 1 en ese nodo y 0 en los otros. De aquí obtenemos:

$$\nabla\eta_j(x,y,z) = \frac{1}{6|T|}\begin{pmatrix} -\begin{vmatrix} 1 & y_{j+1} & z_{j+1} \\ 1 & y_{j+2} & z_{j+2} \\ 1 & y_{j+3} & z_{j+3} \end{vmatrix} \\ \begin{vmatrix} 1 & x_{j+1} & z_{j+1} \\ 1 & x_{j+2} & z_{j+2} \\ 1 & x_{j+3} & z_{j+3} \end{vmatrix} \\ -\begin{vmatrix} 1 & x_{j+1} & y_{j+1} \\ 1 & x_{j+2} & y_{j+2} \\ 1 & x_{j+3} & y_{j+3} \end{vmatrix} \end{pmatrix}$$

Los índices deben entenderse módulo 4. $|T|$ es el área de T:

$$6|T| = \begin{vmatrix} x_{j+1}-x_j & y_{j+1}-y_j & z_{j+1}-z_j \\ x_{j+2}-x_j & y_{j+2}-y_j & z_{j+2}-z_j \\ x_{j+3}-x_j & y_{j+3}-y_j & z_{j+3}-z_j \end{vmatrix}$$

Finalmente, la entrada correspondiente en la matriz de rigidez es:

$$M_{jk} = \int_T \nabla\eta_j(\nabla\eta_k)^T dx =$$

$$= \tfrac{|T|}{(6|T|)^2}\begin{pmatrix} -\begin{vmatrix} 1 & y_{j+1} & z_{j+1} \\ 1 & y_{j+2} & z_{j+2} \\ 1 & y_{j+3} & z_{j+3} \end{vmatrix} \\ \begin{vmatrix} 1 & x_{j+1} & z_{j+1} \\ 1 & x_{j+2} & z_{j+2} \\ 1 & x_{j+3} & z_{j+3} \end{vmatrix} \\ -\begin{vmatrix} 1 & x_{j+1} & y_{j+1} \\ 1 & x_{j+2} & y_{j+2} \\ 1 & x_{j+3} & y_{j+3} \end{vmatrix} \end{pmatrix}^T \cdot \begin{pmatrix} -\begin{vmatrix} 1 & y_{k+1} & z_{k+1} \\ 1 & y_{k+2} & z_{k+2} \\ 1 & y_{k+3} & z_{k+3} \end{vmatrix} \\ \begin{vmatrix} 1 & x_{k+1} & z_{k+1} \\ 1 & x_{k+2} & z_{k+2} \\ 1 & x_{k+3} & z_{k+3} \end{vmatrix} \\ -\begin{vmatrix} 1 & x_{k+1} & y_{k+1} \\ 1 & x_{k+2} & y_{k+2} \\ 1 & x_{k+3} & y_{k+3} \end{vmatrix} \end{pmatrix}$$

De nuevo los índices son módulo 4. Esto mismo se puede escribir simultáneamente para todos los índices como sigue:

$$M = \frac{\begin{vmatrix} x_{j+1}-x_j & y_{j+1}-y_j & z_{j+1}-z_j \\ x_{j+2}-x_j & y_{j+2}-y_j & z_{j+2}-z_j \\ x_{j+3}-x_j & y_{j+3}-y_j & z_{j+3}-z_j \end{vmatrix}}{6} \cdot GG^T$$

con

$$G := \begin{pmatrix} 1 & 1 & 1 & 1 \\ x_j & x_{j+1} & x_{j+2} & x_{j+3} \\ y_j & y_{j+1} & y_{j+2} & y_{j+3} \\ z_j & z_{j+1} & z_{j+2} & z_{j+3} \end{pmatrix}^{-1} \cdot \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Para ver esto hagamos explícita la forma de G:

$$G = \frac{1}{\begin{vmatrix} 1 & 1 & 1 & 1 \\ x_j & x_{j+1} & x_{j+2} & x_{j+3} \\ y_j & y_{j+1} & y_{j+2} & y_{j+3} \\ z_j & z_{j+1} & z_{j+2} & z_{j+3} \end{vmatrix}} \cdot \begin{pmatrix} -\begin{vmatrix} 1 & 1 & 1 \\ y_{j+1} & y_{j+2} & y_{j+3} \\ z_{j+1} & z_{j+2} & z_{j+3} \end{vmatrix} & \begin{vmatrix} 1 & 1 & 1 \\ x_{j+1} & x_{j+2} & x_{j+3} \\ z_{j+1} & z_{j+2} & z_{j+3} \end{vmatrix} & -\begin{vmatrix} 1 & 1 & 1 \\ x_{j+1} & x_{j+2} & x_{j+3} \\ y_{j+1} & y_{j+2} & y_{j+3} \end{vmatrix} \\ \begin{vmatrix} 1 & 1 & 1 \\ y_j & y_{j+2} & y_{j+3} \\ z_j & z_{j+2} & z_{j+3} \end{vmatrix} & -\begin{vmatrix} 1 & 1 & 1 \\ x_j & x_{j+2} & x_{j+3} \\ z_j & z_{j+2} & z_{j+3} \end{vmatrix} & \begin{vmatrix} 1 & 1 & 1 \\ x_j & x_{j+2} & x_{j+3} \\ y_j & y_{j+2} & y_{j+3} \end{vmatrix} \\ -\begin{vmatrix} 1 & 1 & 1 \\ y_j & y_{j+1} & y_{j+3} \\ z_j & z_{j+1} & z_{j+3} \end{vmatrix} & \begin{vmatrix} 1 & 1 & 1 \\ x_j & x_{j+1} & x_{j+3} \\ z_j & z_{j+1} & z_{j+3} \end{vmatrix} & -\begin{vmatrix} 1 & 1 & 1 \\ x_j & x_{j+1} & x_{j+3} \\ y_j & y_{j+1} & y_{j+3} \end{vmatrix} \\ \begin{vmatrix} 1 & 1 & 1 \\ y_j & y_{j+1} & y_{j+2} \\ z_j & z_{j+1} & z_{j+2} \end{vmatrix} & -\begin{vmatrix} 1 & 1 & 1 \\ x_j & x_{j+1} & x_{j+2} \\ z_j & z_{j+1} & z_{j+2} \end{vmatrix} & \begin{vmatrix} 1 & 1 & 1 \\ x_j & x_{j+1} & x_{j+2} \\ y_j & y_{j+1} & y_{j+2} \end{vmatrix} \end{pmatrix}$$

La siguiente rutina en MATLAB genera la contribución a M de cada elemento:

```
function M = STIMA3(vertices)
d = size(vertices,2);
D_eta = [ones(1,d+1);vertices'] \ [zeros(1,d);eye(d)];
```

```
M = det([ones(1,d+1);vertices']) * D_eta * D_eta' / prod(1:d);
```

5.5. Construcción del lado derecho de la ecuación

Para construir el lado derecho se utilizan las fuerzas de volumen (f). Utilizando el valor de f en el centro de gravedad $(x_S, y_S, z_S) = \frac{1}{4}\sum_{i=j}^{j+3}(x_i, y_i, z_i)$ de T, la integral $\int_T f\eta_j dx$ es aproximada por:

$$\int_T f\eta_j dx \approx \frac{1}{24} f(x_S, y_S) \begin{vmatrix} x_{j+1} - x_j & x_{j+2} - x_j & x_{j+3} - x_j \\ y_{j+1} - y_j & y_{j+2} - y_j & y_{j+3} - y_j \\ z_{j+1} - z_j & z_{j+2} - z_j & z_{j+3} - z_j \end{vmatrix}$$

El código resultante es:

```
b(Elements3(j,:)) = b(Elements3(j,:)) + ...
  det([1,1,1,1];Coordinates(Elements3(j,:),:)'])* ...
  f(sum(Coordinates(Elements3(j,:),:))/4) / 24;
```

Para que esto funcione correctamente el orden de los nodos de un tetraedro debe ser tal que:

$$6|T| = \begin{vmatrix} 1 & 1 & 1 & 1 \\ x_k & x_l & x_m & x_n \\ y_k & y_l & y_m & y_n \\ z_k & z_l & z_m & z_n \end{vmatrix}$$

sea positivo. En nuestro caso el mallador se encarga de que esto ocurra.

Los valores de f están dados por la función $f.m$, que depende del problema. Esta función es llamada con coordenadas de puntos en Ω como argumento y devuelve las fuerzas de volumen en esos puntos.

De igual manera, las condiciones de Neumann contribuyen al lado derecho. La integral $\int_E g\eta_j ds$ es aproximada utilizando el valor de g en el centro del triángulo (x_M, y_M, z_M) de E, que tiene superficie $|E|$, por

$$\int_E g\eta_j ds \approx \frac{|E|}{6} g(x_M, y_M, z_M).$$

```
b(Neumann(j,:)) = b(Neumann(j,:)) + ...
norm(cross(Coordinates(Neumann(j,3),:)-Coordinates(Neumann(j,1),:), ...
 Coordinates(Neumann(j,2),:)-Coordinates(Neumann(j,1),:)))...
 * g(sum(Coordinates(Neumann(j,:),:))/3)/6;
```

Los valores de g están dados en $g.m$ y de nuevo dependen del problema. Esta función se llama con las coordenadas de puntos en Γ_N y devuelve los correspondientes valores.

5.6. Incorporando las condiciones de Dirichlet

Numerando adecuadamente los nodos del sistema de ecuaciones lineales que resulta de la construcción descrita en la sección previa, sin incorporar condiciones de Dirichlet, el problema se puede escribir de la siguiente forma:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{pmatrix} \cdot \begin{pmatrix} \vec{U} \\ \vec{U_D} \end{pmatrix} = \begin{pmatrix} \vec{b} \\ \vec{b_D} \end{pmatrix}$$

Con $\vec{U} \in R_M, \vec{U_D} \in R_{M-N}$. Donde $\vec{U}$ son los valores en los nodos libres que hay que determinar, $\vec{U_D}$ son los valores en los nodos sobre el borde Dirichlet y por lo tanto son conocidos a priori. Entonces, el primer bloque de ecuaciones puede ser escrito como:

$$A_{11} \cdot \vec{U} = \vec{b} - A_{12} \cdot \vec{U_D}.$$

El código en MATLAB es:

```
u = sparse(size(Coordinates,1),1);
u(unique(Dirichlet)) = u_D(Coordinates(unique(Dirichlet),:));
b = b - A*u;
```

Los valores de u_D en los nodos sobre Γ_D están dados por la función $u_D.m$ que depende del problema. La función es llamada con coordenadas de puntos en Γ_D y devuelve los valores correspondientes.

5.6.1. Cálculo de la solución

El sistema de ecuaciones es resuelto en MATLAB a través del operador \ que cumple el papel de la inversa a izquierda.

```
u(FreeNodes) = A(FreeNodes,FreeNodes)\b(FreeNodes);
```

Donde FreeNodes es el conjunto de índices de los nodos libres. MATLAB aprovecha que la matriz es simétrica, positiva definida y esparsa para resolver el sistema eficientemente.

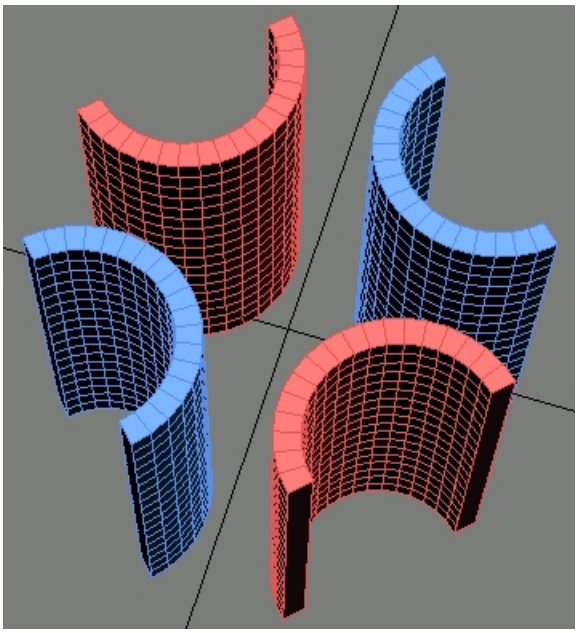
6. Aplicaciones

Habiendo descripto todas las herramientas pasamos ahora a demostrar su utilidad. Exhibiremos tres ejemplos, el primero de electrostática, el segundo de calor estacionario y el tercero de elasticidad lineal. En cada caso habrá que adaptar el código y el método de visualización a las necesidades del problema específico.Para simplificar los ejemplos trabajaremos sin tomar en cuenta las unidades, hecho que no afecta la forma de las soluciones, siendo el objetivo demostrar la utilidad práctica del código.

6.1. Un Problema de Electrostática

El primer ejemplo que abordaremos se trata de un modelo simple de un cuadrupolo, dispositivo que es utilizado, por ejemplo, como parte de un acelerador de partículas y cuya función, en este contexto, es enfocar un haz de partículas cargadas. El haz se puede utilizar a su vez, por ejemplo, para impactar sobre un blanco de litio, obteniéndose así una fuente de iones que tiene diversas utilidades. Una de ellas es la llamada terapia por captura neutrónica en boro, que es una metodología en desarrollo a nivel internacional para combatir ciertos tipos de cáncer.

Consideremos los objetos que se observan en la siguiente figura:

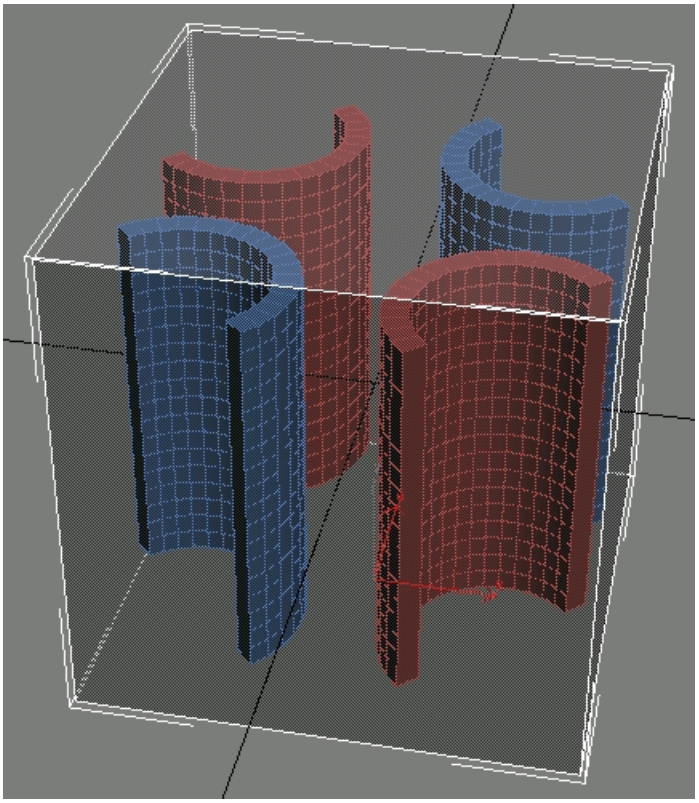


Se trata de cuatro semitubos con 0,45 de radio exterior, 0,35 de radio interior y 1,5 de altura

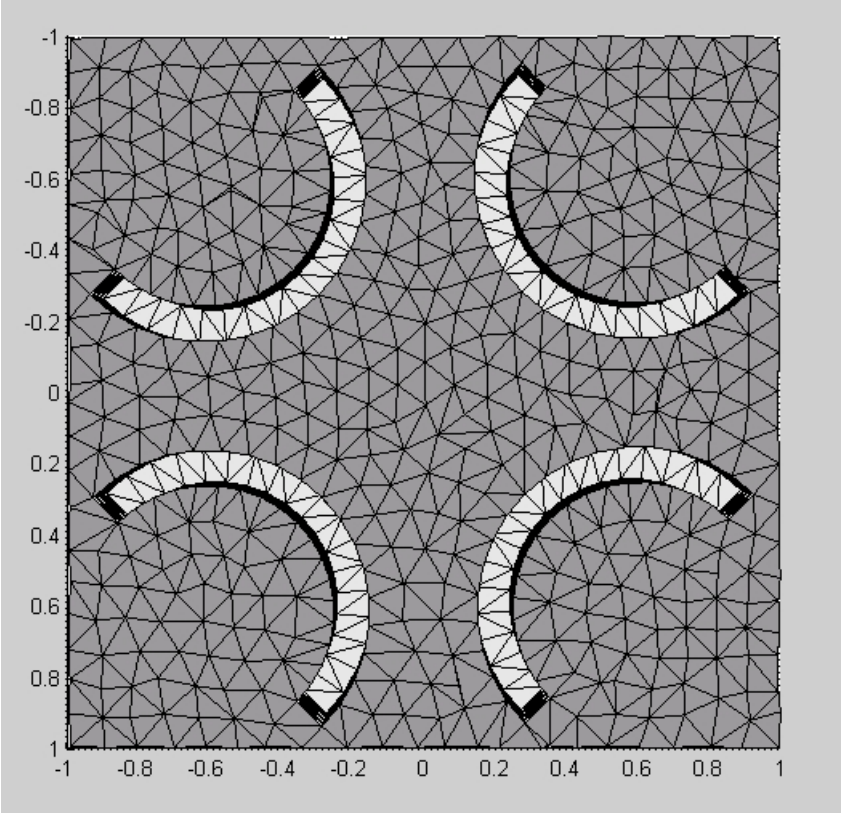
dispuestos simétricamente alrededor del eje z que configuran el modelo simplificado de cuadrupolo.

Nuestro objetivo es calcular el potencial eléctrico generado por el cuadrupolo. Obviaremos las estructuras de soporte, que no tienen efecto sobre el campo generado, ya que carecen de carga. Para realizar los cálculos les asignaremos a los dos semicilindros rojos un potencial negativo constante de -10 y a los dos semicilindros azules un potencial constante de 10.

Con el propósito de mallar una zona del espacio restringida dispusimos los semitubos dentro de una caja con dimensiones $2 \times 2 \times 2$, sobre la cual supusimos un potencial nulo, la disposición de los objetos se observa en la figura:



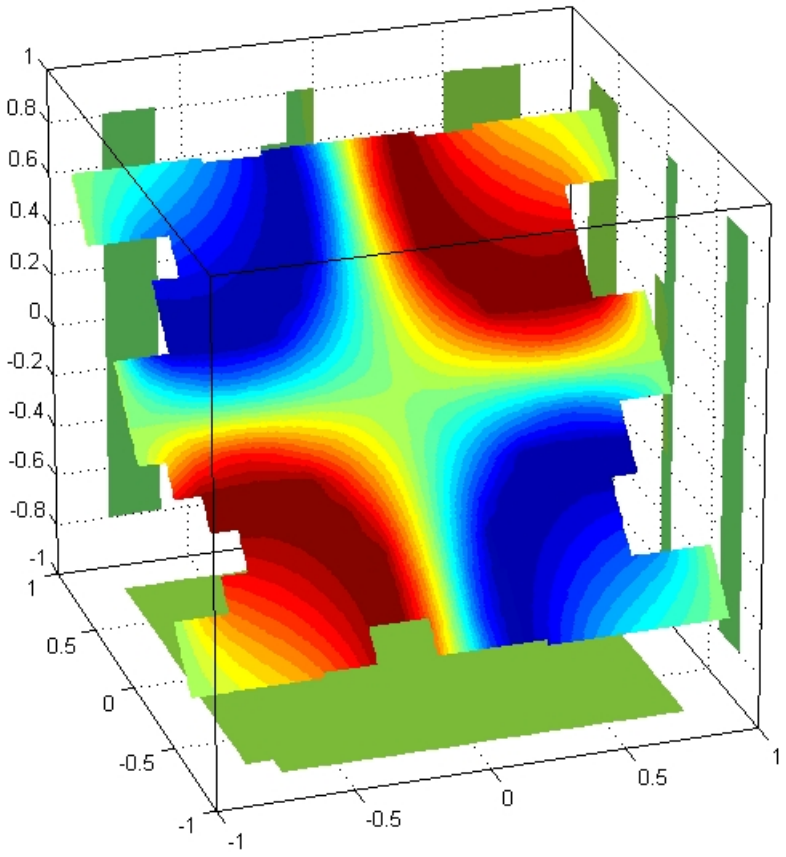
Luego calculamos el mapa de distancias hasta una distancia conveniente y utilizamos el mallador descripto en la sección 3 para obtener la triángulación del dominio que se aprecia en la figura que sigue:



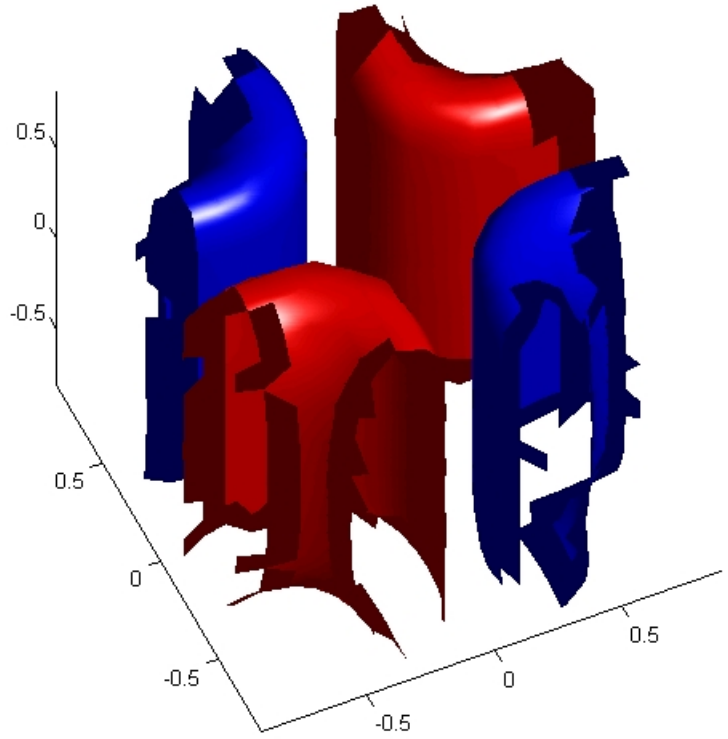
Encontrar el potencial eléctrico generado por esta configuración en la zona mallada es una simple cuestión de resolver la ecuación de Laplace con las condiciones de contorno antes mencionadas. En efecto, la ecuación de Maxwell para la divergencia del campo eléctrico es: $\nabla.E = \frac{\rho}{\epsilon_0}$, donde ρ es la densidad volumétrica de carga y ϵ_0 es la permitividad dieléctrica del vacío. Además para un caso electrostático el campo eléctrico se obtiene como el gradiente de una función potencial: $E = -\nabla V$, entonces $\Delta E = -\frac{\rho}{\epsilon_0}$. Por lo tanto en una zona del espacio libre de cargas el potencial cumple la ecuación de Laplace: $\Delta V = 0$ Para utilizar el código de resolución descripto en la sección anterior debemos seleccionar los triángulos correspondientes a las condiciones de Dirichlet y orientarlos positivamente vistos desde el exterior del dominio. Para ello se escribió un pequeño programa que determina las caras exteriores teniendo en cuenta que son las únicas que pertenecen a un solo tetraedro (es decir no las comparten dos tetraedros distintos) y que las orienta usando la observación de que el interior de cada tetraedro pertenece al dominio. Una vez identificados los triángulos del contorno se seleccionan los puntos en donde se cumplirá cada condición en función de sus coordenadas. Por ejemplo, uno de los semicilindros azules está en el primer cuadrante xy pero sus puntos no tienen coordenadas x o y mayores a $0,97$, de la misma manera se identifican los

puntos correspondientes a cada semicilindro y a la caja y se les asignan las condiciones de contorno mencionadas más arriba.

Luego de correr la simulación obtenemos el valor del potencial en los nodos definidos por la malla. Para obtener una interpolación sobre una grilla tridimensional utilizamos el comando de Matlab: *griddatan*(). Primero realizamos un corte oblicuo y lo graficamos:



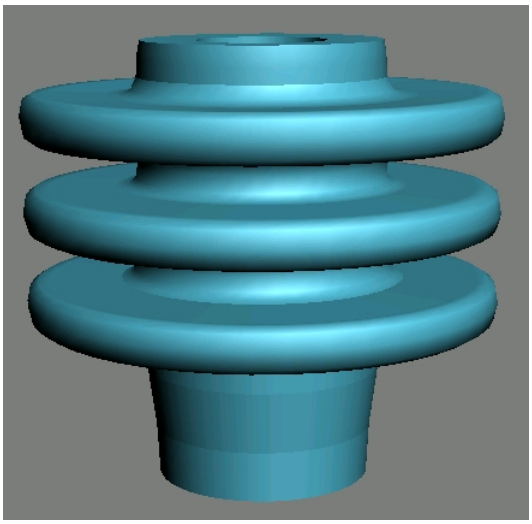
También graficamos algunas superficies equipotenciales:

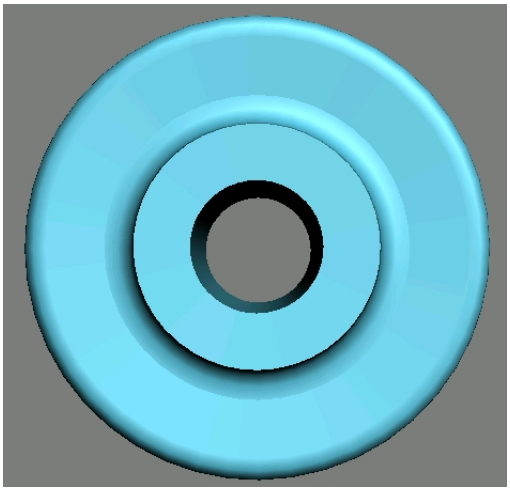


6.2. Un Problema de Calor Estacionario

En el segundo ejemplo consideraremos un problema que surge en procesos de la industria química. Muchas veces se necesita refrigerar fluídos utilizando bridas de vidrio como la que se ilustra en las figura siguientes:

Brida de vidrio





el fluído atraviesa el orificio cilíndrico de la brida y el enfriamiento se realiza por la transferencia de calor del fluído al medio circundante (aire) a través de la brida.

En general este tipo de procesos está dominado por la convección desde y hacia las superficies. Nuestro fin es obtener las temperaturas en el volumen de la brida, que servirán, por ejemplo, para obtener el poder de disipación de calor del sistema, un parámetro importante para evaluar su performance. Una manera conveniente de estudiar este problema es utilizar el coeficiente de transferencia de calor, h. Este coeficiente describe la influencia del flujo del fluído y los flujos convectivos. Utilizándolo evitamos modelar el campo de flujo del fluído, simplificando la simulación a realizar [6].

En estas condiciones debemos mallar la brida y resolver la ecuación de calor estacionario dentro de su volumen: $\nabla \cdot (-k\nabla T) = 0$. En los bordes exteriores del dominio, donde hace contacto con el fluído y con el aire circundante se deben cumplir las condiciones de borde: $k.\frac{\partial T}{\partial n} = q_0 + h(T_{inf} - T)$ donde h es el coeficiente de transferencia de calor y T_{inf} la temperatura del medio circundante. El coeficiente h en el lado del fluído se puede suponer constante. Supondremos, por su parte, que en la parte expuesta al ambiente hay una corriente de aire que fluye perpendicularmente al eje z y por eso la expresión de h en el exterior es más complicada y depende del ángulo de rotación alrededor del eje z. Como se puede observar en este caso la derivada normal depende del valor de la temperatura misma, por eso debemos modificar levemente el esquema presentado en la sección anterior y escribir en lugar de (43) la siguiente ecuación:

$$\int_\Omega \nabla u_h \cdot \nabla \eta_j dx + \int_{\Gamma_N} h u_h \eta_j ds =$$

$$= \int_\Omega f \eta_j dx + \int_{\Gamma_N} (q_0 + h T_{inf}) \eta_j ds - \int_\Omega \nabla U_D \cdot \nabla \eta_j dx \ (\forall j \in I)$$

Esto modifica la matriz A de la siguiente manera:

$$A_{jk} = \sum_{T \in T} [\int_T \nabla \eta_j \cdot \nabla \eta_k dx + \int_{\Gamma_N} h \eta_j \eta_k ds] \qquad (j, k \in I)$$

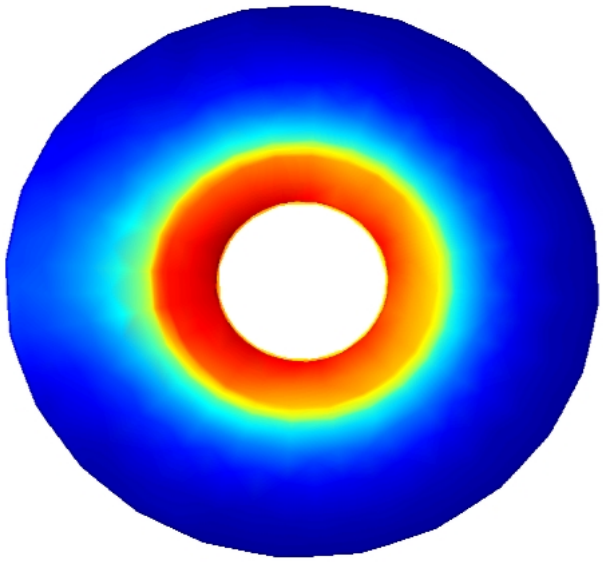
Para calcular la segunda integral para cada triángulo del borde observamos que el valor de η_j en el baricentro de un triángulo es $\frac{1}{3}$ y entonces aproximamos la integral por $h(b_E)\frac{1}{9}|E|$, donde $|E|$ es el área del triángulo y b_E es su baricentro. Tenemos que agregar dos partes al código de construcción de la matriz:

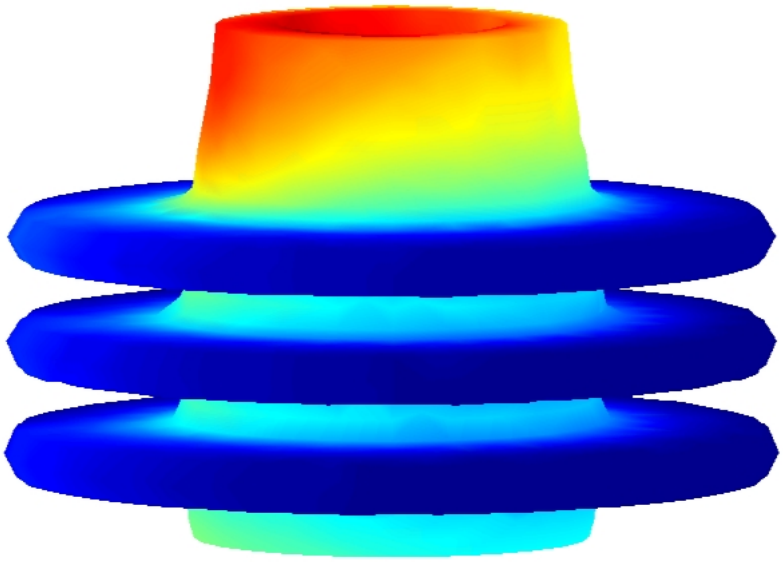
```
%INTERIOR

for j=1 : size(trisint,1)
  A(trisint(j,:),trisint(j,:)) = A(trisint(j,:),trisint(j,:)) ...
      +h*ones(3,3)*norm(cross(coordinates(trisint(j,3),:) ...
      -coordinates(trisint(j,1),:), ...
      coordinates(trisint(j,2),:)-coordinates(trisint(j,1),:)))/18;
end

%EXTERIOR

for j=1 : size(trisext,1)
  A(trisext(j,:),trisext(j,:)) = A(trisext(j,:),trisext(j,:)) ...
      +h*angulo(coordinates(trisext(j,1),1:2))*...
      ones(3,3)*norm(cross(coordinates(trisext(j,3),:) ...
      -coordinates(trisext(j,1),:), ...
```

```
        coordinates(trisext(j,2),:)-coordinates(trisext(j,1),:)))/18;
end
```

Donde *trisint* identifican a los triángulos de las caras en la parte interior de la brida (en contacto con el flído) y *trisext* aquellos en la parte cilíndrica interior (en contacto con el ambiente). La función angulo aquí expuesta varía según el ángulo alrededor del eje z siendo su valor máximo constante cuando el ángulo esta en $[0, \frac{\Pi}{2}]$ o en $[\frac{3}{2}\Pi, \Pi]$, es decir en el sector de la brida frente al flujo de aire, y decreciendo paulatinamente a 0 en la parte posterior de la brida.

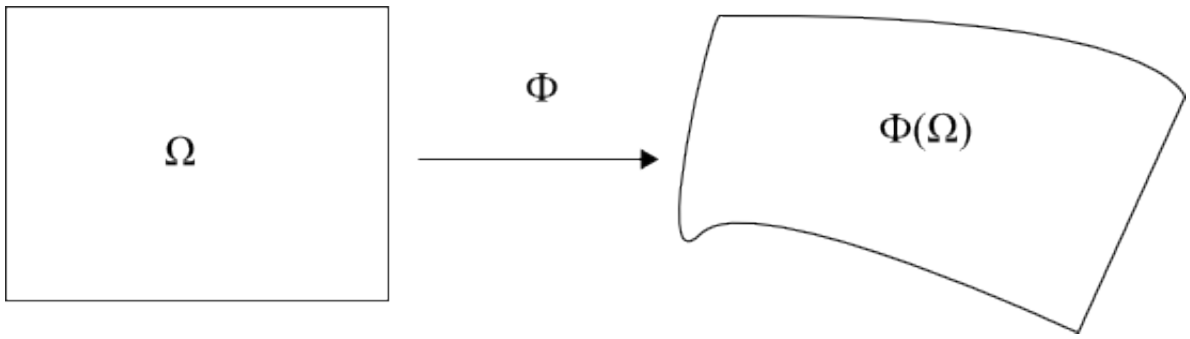
Finalmente graficamos la solución obtenida:





6.3. Un Problema de Elasticidad Lineal

Consideremos un cuerpo representado como $\overline{\Omega}$ con $\Omega \subset \mathbb{R}^3$ un dominio que para nuestros propósitos consideramos Lipschitz con frontera Γ. Si el cuerpo es elástico se verá deformado después de la acción de ciertas fuerzas que obren sobre él . Si representamos esta deformación como la aplicación de una función Φ



sobre el dominio original, entonces $\Phi(\Omega)$ representa el cuerpo después de la deformación. Más precisamente, una función $\Phi : \Omega \to \mathbb{R}^3$ biyectiva se denomina una deformación si es suficientemente regular y verifica

$$det(\nabla\Phi) > 0 \tag{46}$$

donde $\nabla\Phi$ representa la matriz diferencial y se denomina el gradiente de deformaciones (la condición (46) evita la interpenetrablidad del sólido).

En la teoría de elasticidad se distinguen dos tipos de fuerzas externas actuantes sobre el cuerpo, las fuerzas de volumen f (como la gravedad) que actúan sobre todo Ω y las de superficie g que actuan sobre Γ. En general, para muchos casos existen condiciones que confinan ciertas partes de la superficie a estar inmóviles (si el cuerpo se apoya por ejemplo en una superficie, etc.) o sometidas a una deformación predefinida. Si bien la función Φ describe la configuración final de cuerpo, suele utilizarse en la modelización del proceso de deformación otra variable denominada el desplazamiento $u : \Omega \to \mathbb{R}^3$ definida a través de la función identidad Id como

$$\Phi = Id + u$$

a partir de la cual se obtiene Φ inmediatamente. Las ecuaciones de elasticidad lineal (para pequeñas deformaciones) pueden escribirse a través del siguiente sistema (ver [5])

$$\left\{ \begin{array}{rcccc} div\sigma & = & -f & \text{en} & \Omega \\ u & = & 0 & \text{en} & \Gamma_D \\ \sigma\eta & = & g & \text{en} & \Gamma_N \end{array} \right. \tag{47}$$

donde σ es el tensor de tensiones que se obtiene a través de la relación constitutiva (generalizacion de la Ley de Hooke)

$$\sigma = C\varepsilon(u) \tag{48}$$

que relaciona linealmente las deformaciones (el tensor de deformaciones)

$$\varepsilon(u) = \frac{\nabla u + \nabla u^T}{2} \tag{49}$$

con las tensiones (tensor de tensiones). Notar que en este sentido la divergencia div que figura en el ecuación (47) debe interpretarse actuando fila a fila en la matriz σ. Por su parte en la identidad (48), la C depende de las propiedades elásticas del material, de hecho

$$\sigma = \frac{E}{1+\nu}\left(\varepsilon + \frac{\nu}{1-2\nu} tr(\varepsilon) Id\right)$$

con Id la matriz identidad (las constantes E y ν se denominan módulo de elasticidad y radio de

Poisson). En (47) utilizamos la notación estándar Γ_D y Γ_N para las condiciones de Dirichlet y de Neumann respectivamente. Por otro lado condiciones de Dirichlet no homogéneas pueden cambiarse por homogéneas del modo usual, o sea restando una función adecuada que coincida con u en Γ_D. La ecuacion (47) admite una formulación débil del tipo (1) del siguiente modo. Sea $H^1(\Omega)^3$, el espacio de funciones definidas sobre Ω en $\mathbb{R}^3$ con todas sus componentes en $H^1(\Omega)$, la norma de este espacio se define del siguiente modo, si $u = (u_1, u_2, u_3)$ entonces

$$\|u\|_{H^1(\Omega)} = \sum_{i=1}^{3} \|u_i\|_{H^1(\Omega)}.$$

Si llamamos

$$H_{\Gamma_D} = \{v \in H^1(\Omega)^3 : \quad v = 0 \ \text{ en } \ \Gamma_D\}$$

entonces la forma débil se escribe [5]:

Hallar $u \in H_{\Gamma_D}$ tal que

$$\int_\Omega \varepsilon(v) : C\varepsilon(u)dx = \int_\Omega f.vdx + \int_{\Gamma_N} g.vds \qquad \text{para toda} \qquad v \in H_{\Gamma_D} \tag{50}$$

la notación $A : B$ indica para matrices cuadradas del mismo tamaño $tr(AB^T)$ (siendo tr la traza) por lo que la integral es de una función escalar, lo mismo que las integrales del miembro derecho en las que $f.v$ y $g.v$ representan el producto interno usual de vectores de $\mathbb{R}^3$. Para matrices $M \in \mathbb{R}^{3\times 3}$ la norma de Frobenius se define como

$$\|M\|_F = \sqrt{\sum_{1\le i,j\le 3} M_{i,j}^2} = \sqrt{M : M}$$

es fácil ver que

$$|A : B| = |tr(AB^T)| \le \|A\|_F \|B\|_F$$

y desde aquí se tiene que la forma bilineal

$$\langle u, v\rangle = \int_\Omega \varepsilon(v) : C\varepsilon(u)dx$$

es contínua debido a que

$$\|\varepsilon(u)\|_F \le \|\nabla u\|_F$$

gracias a (49), y por ende

$$|\int_\Omega \varepsilon(v) : C\varepsilon(u)dx| \leq C\int_\Omega \|\varepsilon(v)\|_F\|\varepsilon(u)\|_F dx \leq C\sqrt{\int_\Omega \|\nabla v\|_F^2 dx}\sqrt{\int_\Omega \|\nabla u\|_F^2 dx}$$

que conduce a

$$|\int_\Omega \varepsilon(v) : C\varepsilon(u)dx| \leq C\|u\|_{H^1(\Omega)}\|v\|_{H^1(\Omega)}.$$

Por su parte, la coercividad de la forma, o sea

$$\|u\|^2_{H^1(\Omega)} \leq C|\int_\Omega \varepsilon(u) : C\varepsilon(u)dx| \tag{51}$$

no es inmmediata, pero es válida en el espacio H_{Γ_D} y se obtiene de la denominada desigualdad de Korn

$$\|u\|^2_{H^1(\Omega)} \leq \int_\Omega \|\varepsilon(u)\|_F^2 dx$$

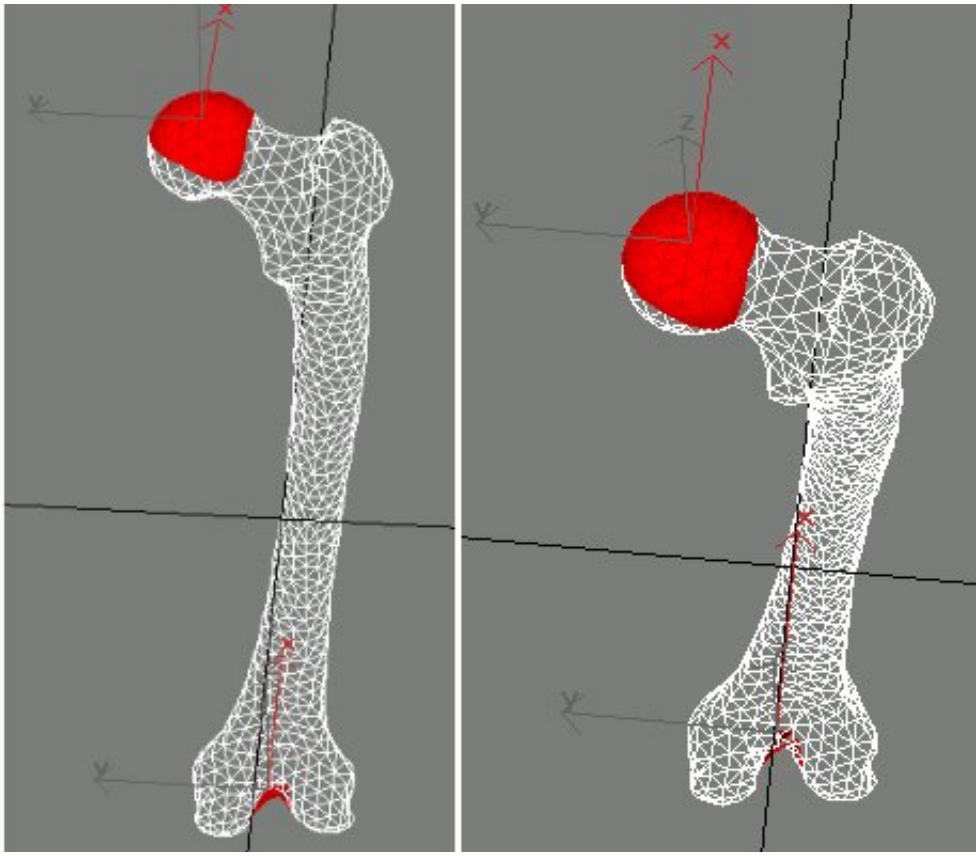
[5]. Por otro lado la forma lineal

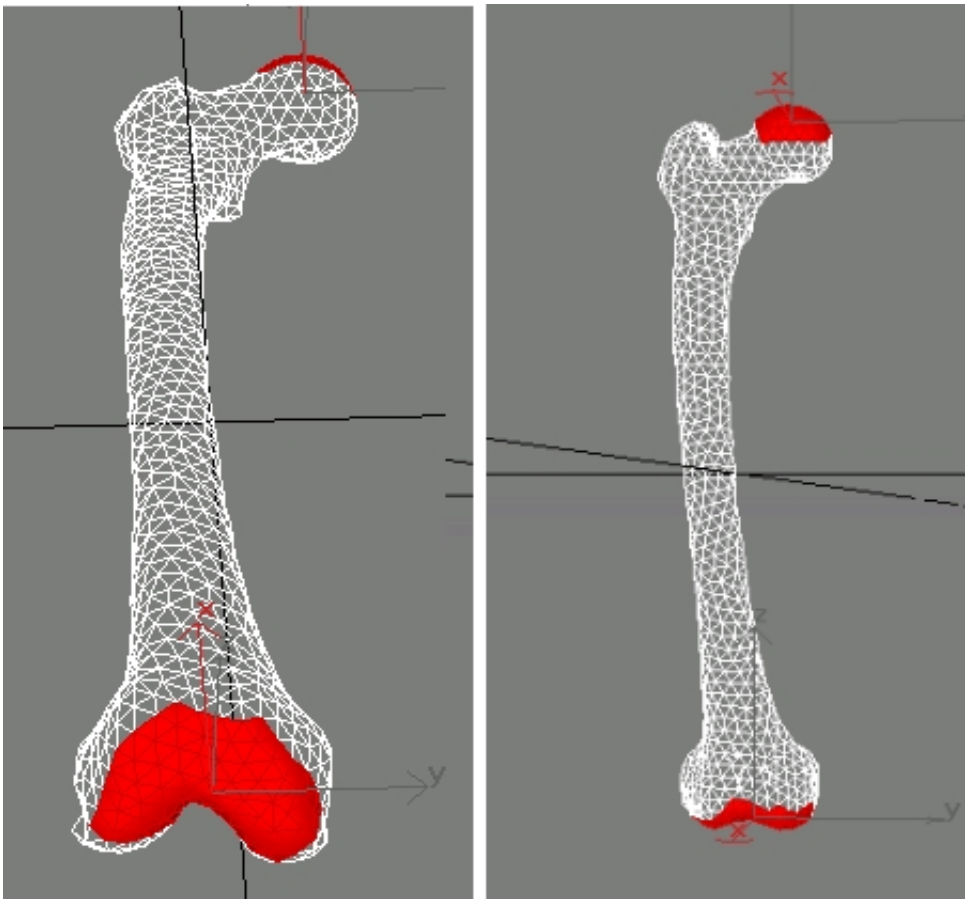
$$L(v) = \int_\Omega f.vdx + \int_{\Gamma_N} g.vds$$

resulta continua bajo la hipótesis de $f, g \in L^2$ usando la desigualdad de Cauchy-Schwartz en ambos sumandos y el teorema de trazas en el segundo.

De este modo vemos que las ecuaciones lineales de elasticidad caen en el contexto general presentado antes obteniendose existencia y unicidad de la solución de (50) en H_{Γ_D}. Asi es que podemos utilizar aproximaciones de Galerkin construidas sobre espacios de elementos finitos, obteniéndose estimaciones del error como en el caso del problema de Poisson siempre que se tenga suficiente regularidad para la solución. En particular tomando V_h el espacio de funciones lineales a trozos sobre una triangulación adecuada que se anulen en Γ_D, se tiene que $V_h^3 \subset H_{\Gamma_D}$.

Para resolver el presente ejemplo utilizamos el código descripto en [2], que es una modificación del que fue presentado en la quinta sección de esta tesis adaptada a la resolución de problemas de elasticidad lineal. Se define $\Gamma_D = \Gamma_{D_1} \cup \Gamma_{D_2}$ como se ve en las figuras.





En particular para la parte Dirichlet superior Γ_{D_1} (situada en la cabeza del fémur) consideramos las condiciones

$$u(\Gamma_{D_1}) = (0, 0, -0 \cdot 01)$$

que equivale a un desplazamiento del 0,5 % de la longitud total del hueso. En la zona baja del femur Γ_{D_2} tomamos condiciones homogeneas (desplazamientos nulos)

$$u(\Gamma_{D_2}) = (0, 0, 0)$$

por su parte las condiciones de Neuman son homogéneas en $\Gamma_N = \Gamma \setminus \Gamma_D$ (esa zona es libre de tensiones). Para elegir los triángulos correspondientes se creó un pequeño programita que permite seleccionar los triángulos deseados visualmente, a la manera de un programa de edición de mallas como el 3d Studio. Para la resolución numérica tomamos valores $\nu = 3$ y $E = 100$. En la representación gráfica de la solución la escala de colores indica la densidad de energía elástica de corte *dec.*

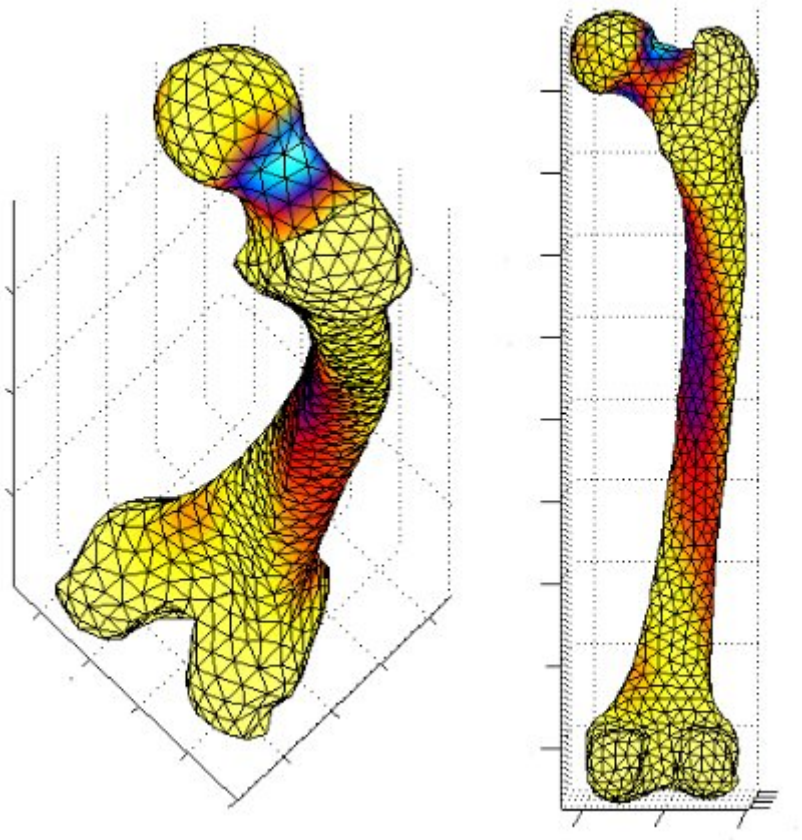
Esta se obtiene a través del desviador de tensiones:

$$dev(\sigma) = \sigma - \frac{tr(\sigma)}{3} Id_{3\times 3}$$

del siguiente modo,

$$dec = \frac{1}{4\mu} \|dev(\sigma)\|_F.$$

Donde μ es una constante elástica. Como se observa en la figura la energía de corte es considerablemente alta en el denominado cuello anatómico, que es el punto de fractura más común en personas con baja densidad ósea.



Referencias

[1] J. Alberty, C. Carstensen, S. A. Funken, Remarks Around 50 Lines of Matlab: Short Finite Element Method Implementation, Numer. Algorithms, 20, 1999.

[2] J. Alberty, C. Carstensen, S. A. Funken, R. Klose, Matlab Implementation of the Finite Element Method in Elasticity, Preprint, 2002.

[3] S. Brenner, R. Scott, The Mathematical Theory of Finite Element Methods, 2nd. Ed., Springer, 2002.

[4] P. G. Ciarlet, The Finite Element Method for Elliptic Problems, North-Holland, 1978.

[5] P. G. Ciarlet, Mathematical Elasticity, Vol I: Three-Dimensional Elasticity, Amsterdam: North-Holland, 1988.

[6] COMSOL AB, Cooling Flange Solved with COMSOL Multiphysics 3.2,

`http://www.mathe.tu-freiberg.de/~ernst/Lehre/PWR-EC/cooling_flange.pdf`

[7] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, Computer Graphics: Principles and Practice, Second Edition in C, Addison-Wesley Professional, 1995.

[8] C. Johnson, Numerical Solutions of Partial Differential Equations by the Finite Element Method, Cambridge University Press, 1995.

[9] M. W. Jones, J. A. Bærentzen, M. Sramek, 3D Distance Fields: A Survey of Techniques and Applications, Visualization and Computer Graphics, IEEE Transactions on Volume 12, Issue 4, July-Aug. 2006.

[10] NVIDIA Corporation, NVIDIA CUDA Compute Unified Device Architecture, Programming Guide 1.1,

`http://developer.download.nvidia.com/compute/cuda/1_1/NVIDIA_CUDA_Programming_Guide_1.1.pdf`

[11] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell, A Survey of General-Purpose Computation on Graphics Hardware, Computer Graphics Forum, 26(1):80–113, March 2007.

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

[13] J. R. Shewchuk, Lecture Notes on Delaunay Mesh Generation, Department of EECS, UC Berkeley.

[14] J. M. Steele, The Cauchy-Schwarz Master Class, Cambridge University Press, 2006.

[15] O.C. Zienkiewicz, R.L. Taylor , The Finite Element Method, Volume 1: The Basis, Fifth Edition, Butterworth-Heinemann,2000.

[16] Interpolation and Multidimensional Arrays in MATLAB,

`http://esra.univ-paris1.fr/matlab5/techdoc/using_ml/ch_5_p14.html`

[17] Tetrahedron Wikidoc,

`http://www.wikidoc.org/index.php/Tetrahedron`