Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO

Grado en Matemáticas

Interpolación bidimensional sobre triangulaciones

Autora: Claudia Enrique García
Tutor: Luis Mª Abia Llera
Año 2023-2024

Índice general

Prefacio

La presente Memoria de Trabajo Fin de Grado se centra en el estudio de la interpolación polinómica sobre triangulaciones de un dominio, con enfásis en la construcción de interpolantes polinómicos a trozos con regularidad C^1 y C^2.

Interpolantes de clase C^1 son de interés en el método de elementos finitos para problemas de valores en la frontera de ecuaciones diferenciales elípticas de cuarto orden: ecuación biarmónica y ecuación para la deformación de placas. En el contexto general de la teoría de la aproximación estos interpolantes se representan de forma muy conveniente por técnicas derivadas de las utilizadas en la representación de Bézier de los splines en una variable. La memoria parte de primeros principios describiendo la extensión de las técnicas de Bézier a dos variables. Se estudia en el capítulo 1 la B-forma de un polinomio de dos variables, el algoritmo de Casteljau de evaluación del polinomio y sus derivadas, el análisis de las condiciones de regularidad en términos de los coeficientes de la B-forma, y los algoritmos de subdivisión y elevación de grado.

En el capítulo 2 se introducen definiciones generales para referirse a una triangulación, y sus refinamientos de Powell-Sabin, Clough-Tocher y Wang que serán necesarios para la definición de los macroelementos respectivos en el capítulo 4.

Todos los interpolantes polinómicos a trozos que se construyen en el capítulo 4 son ejemplos de espacios de splines generalizados, cuya definición, propiedades de regularidad y problemas de interpolación asociados son objeto del capítulo 3.

En el capítulo 4, se construye cada uno de los interpolantes en los macroelementos ya mencionados y el interpolante de Argyris en un triángulo (de grado 5). Se formulan además teoremas para el error de interpolación de

cada uno de ellos.

Por último, el capítulo 5 ilustra la efectividad de estos interpolantes en algunos problemas test. Aunque el cálculo de los coeficientes de la B-forma de estos interpolantes, y su evaluación efectiva con el algoritmo de Casteljau se han programado en MATLAB partiendo de primeros principios, hemos aliviado la carga de elaboración de los programas haciendo uso de software Splinepack para la generación de triangulaciones y la evaluación de la función de Franke (problema test).

Capítulo 1

Técnicas de Bezier en triángulos

1.1. Coordenadas baricéntricas

Las coordenadas baricéntricas, son un elemento muy importante que utilizaremos a la hora de trabajar con polinomios sobre triángulos. Sea T un triángulo en $\mathbb{R}^2$, que tiene como vértices los puntos

$$v_i = (x_i, y_i), \quad i = 1, 2, 3.$$

Supongamos que v_1, v_2, v_3 son no colineales. Vamos a referirnos a dicho triángulo como $T := \langle v_1, v_2, v_3 \rangle$.

El área de este triángulo, será

$$A_T = \frac{1}{2} det(M),$$

siendo M la matriz:

$$\begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix}$$

Está claro que el área será positiva siempre que coloquemos los vértices en sentido antihorario.

Veamos cómo podemos referirnos a cualquier punto del plano $v := (x, y)$ respecto del triángulo T.

Lema 1.1.1. *Cada punto $v := (x, y) \in \mathbb{R}^2$ tiene una representación única de la forma*

$$v = b_1 v_1 + b_2 v_2 + b_3 v_3$$

con

$$1 = b_1 + b_2 + b_3$$

Estos números b_1, b_2, b_3 son las coordenadas baricéntricas del punto v relativas al triángulo T.

Demostración. Si representamos el sistema en forma matricial, obtenemos,

$$\begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} \tag{1.1}$$

donde M es la matriz que acabamos de escibir en (1.1). Ya sabemos que el área de T es $A_T = \frac{1}{2} det(M)$. Este área es positiva por como hemos definido T, por lo que será un sistema compatible determinado y tendremos una única solución para b_1, b_2, b_3. □

Utilizando la regla de Cramer para resolver (1.1), obtenemos fórmulas generales para estas coordenadas baricéntricas b_1, b_2 y b_3:

$$b_1 = \frac{(x_2y_3 - y_2x_3) - x(y_3 - y_2) + y(x_3 - x_2)}{det(M)}$$

$$b_2 = \frac{(x_1y_3 - y_1x_3) - x(y_3 - y_1) + y(x_3 - x_1)}{det(M)}$$

$$b_3 = \frac{(x_1y_2 - y_1x_2) - x(y_2 - y_1) + y(x_2 - x_1)}{det(M)}$$

donde

$$det(M) = (x_2 - x_1)(y_3 - y_2) - (x_3 - x_2)(y_2 - y_1).$$

Podemos observar que cada $b_i, i = 1, 2, 3$ es un polinomio lineal $b_i(x, y)$.

Estas coordenadas también nos dan información sobre dónde se encuentra un punto respecto a un triángulo. Dado un triángulo T, sean $\{\Omega_i\}_{i=0}^{6}$ los interiores de las 6 regiones que obtenemos si extendemos los lados de T. Para todo $0 \leq i \leq 6$, el signo de cada una de las coordenadas baricéntricas b_1, b_2, b_3 es el mismo para todos los puntos $v \in \Omega_i$.

Teorema 1.1.2. *[2] Un punto v está en el interior de T si y solo si todas sus coordenadas baricéntricas con respecto de T son positivas.*

En la siguiente imagen podemos observar cómo va a ser el signo de las coordenadas baricéntricas según donde se encuentre el punto respecto al triángulo central $T := \langle v_1, v_2, v_3 \rangle$.

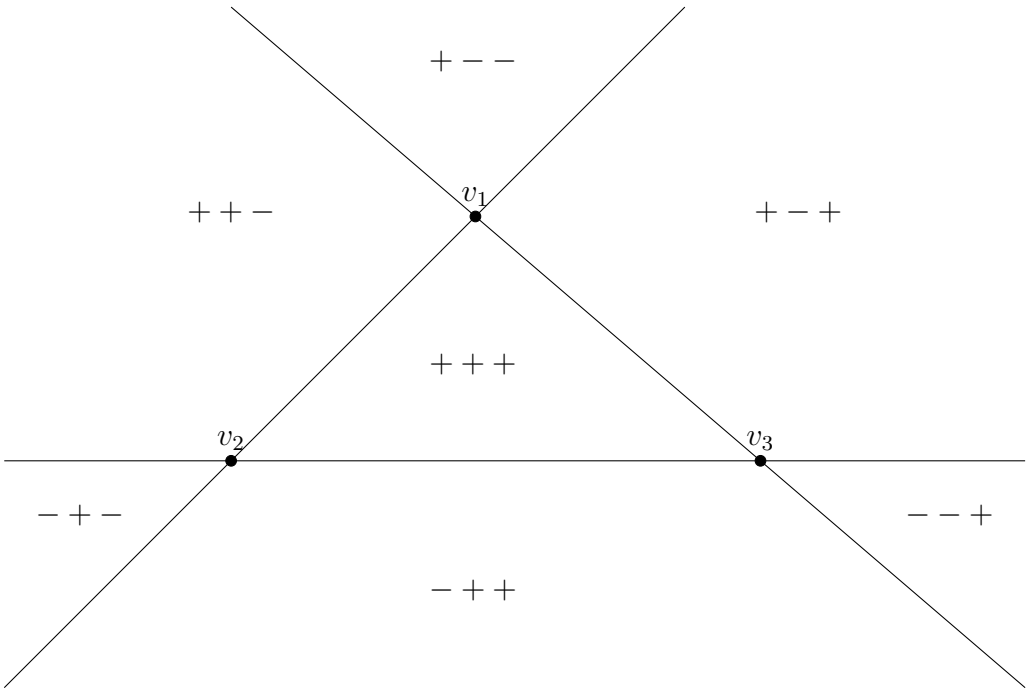


Figura 1.1: Signo de las coordenadas baricéntricas.

Otras definiciones básicas que utilizaremos más adelante son las que introduciremos a continuación.

Definición 1.1.3. Sea $T := \langle v_1, v_2, v_3 \rangle$ un triángulo, el baricentro de T es el punto donde se cortan las medianas de un triángulo, es decir, $v_T := (v_1 + v_2 + v_3)/3$.

Definición 1.1.4. Sea $T := \langle v_1, v_2, v_3 \rangle$ un triángulo, el incentro de T es el punto donde se cortan las tres bisectrices de sus ángulos internos.

Es interesante notar que dado un punto $v \in T$, si construimos $T_1 := \langle v, v_2, v_3 \rangle$, $T_2 := \langle v_1, v, v_3 \rangle$ y $T_3 := \langle v_1, v_2, v \rangle$, las coordenadas baricéntricas de v relativas a T, están dadas por

$$b_i = \frac{A_{T_i}}{A_T}, i = 1, 2, 3,$$

por lo que es obvio que estas coordenadas serán positivas en estos casos.

Para representar el espacio de polinomios de grado d, utilizamos

$$P_d := \{p(x,y) = \sum_{0 \le i+j \le d} c_{ij} x^i y^j\}.$$

A continuación, analizaremos cómo podemos expresar un polinomio en términos de las coordenadas baricéntricas asociadas a un triángulo.

1.2. Polinomios de Bernstein

Continuando con la misma notación que en la sección anterior, T es un triángulo fijado, y para cada $v = (x,y) \in \mathbb{R}^2$, b_1, b_2, b_3 son sus coordenadas baricéntricas.

Definición 1.2.1. Dado un entero $d > 0$, las **bases de Bernstein de grado d relativas a** T están definidas como

$$B^d_{ijk} := \frac{d!}{i!j!k!} b_1^i b_2^j b_3^k, \quad i+j+k = d, \tag{1.2}$$

donde i, j, k son enteros no negativos.

Como ya vimos que b_1, b_2, b_3, son polinomios lineales en x e y, se deduce que $B^d_{ijk}(x,y)$ también será un polinomio en x e y de grado d, que siguiendo la misma notación, también puede ser escrito como $B^d_{ijk}(v)$.

Siempre que tengamos algún subíndice i, j, k negativo, asumiremos por convenio que $B^d_{ijk} = 0$. Estos polinomios tienen propiedades interesantes, como pueden ser las siguientes:

- $\sum_{i+j+k=d} B^d_{ijk} \equiv 1 \quad$ para todo $(x,y) \in \mathbb{R}^2$
- $0 \le B^d_{ijk}(v) \le 1 \quad$ para todo $(x,y) \in T$

La primera se deduce del desarrollo del trinomio de la suma de las coordenadas baricéntricas que ya sabemos que suman 1,

$$1 = (b_1 + b_2 + b_3)^d = \sum_{i+j+k=d} B^d_{ijk}.$$

La segunda es una consecuencia de la anterior propiedad y de que las coordenadas baricéntricas sean positivas siempre que el punto pertenezca al triángulo.

Teorema 1.2.2. *Los polinomios* $\{B^d_{ijk}\}_{i+j+k=d}$ *son linealmente independientes y forman una base del espacio vectorial de dimensión* $\binom{d+2}{2}$*,* P_d*, de polinomios de grado* d*.*

Demostración. El número de polinomios de Bernstein en $\{B^d_{ijk}\}_{i+j+k=d}$ es igual a $\binom{d+2}{2}$, la dimensión de P_d, por lo que solo necesitamos demostrar que todos los polinomios de la forma $x^k y^l$ con $0 \leq k+l \leq d$ se pueden representar con elementos de B^d. Para comenzar esta demostración, partiremos de la definición de coordenadas baricentricas para cualquier punto del plano. El sumatorio de todos los polinomios de Bernstein es 1, y para dimensión 1 se tiene que,

$$\begin{bmatrix} x \\ y \end{bmatrix} = b_1 \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + b_2 \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} + b_3 \begin{bmatrix} x_3 \\ y_3 \end{bmatrix}$$

y usando la primera de las propiedades descritas para estos polinomios para el caso de $d-1$, se tiene que,

$$\begin{aligned} x &= b_1 x_1 + b_2 x_2 + b_3 x_3 \\ &= (b_1 x_1 + b_2 x_2 + b_3 x_3)\Big(\sum_{i+j+k=d-1} B^{d-1}_{ijk}(x,y)\Big) \\ &= \sum_{i+j+k=d} \frac{1}{d}(i x_1 + j x_2 + k x_3) B^d_{ijk}(x,y). \end{aligned}$$

Analogamente, tenemos que para y,

$$y = \sum_{i+j+k=d} \frac{1}{d}(i y_1 + j y_2 + k y_3) B^d_{ijk}(x,y)$$

Por lo que podemos afirmar que x e y se pueden representar en función de polinomios de Bernstein.

Ahora vamos a proceder por inducción sobre el grado. Asumimos que se cumple para $d-1$, entonces,

$$x^{k-1} y^l = \sum_{i+j+k=d-1} c_{ijk} B^{d-1}_{ijk}(x,y)$$

para unos coeficientes c_{ijk}. Pero entonces,

$$x^k y^l = (b_1 x_1 + b_2 x_2 + b_3 x_3) \sum_{i+j+k=d-1} c_{ijk} B^{d-1}_{ijk}(x,y)$$

Multiplicando los términos y reordenando, obtenemos

$$x^k y^l = \sum_{i+j+k=d} d_{ijk} B^d_{ijk}(x, y)$$

siendo d_{ijk} constantes, lo que finaliza la demostración. □

Acabaremos esta sección viendo que los polinomios de Bernstein B^d_{ijk} tienen un único máximo.

Teorema 1.2.3. *Para todo $i + j + k = d$, sea $v \in T = \langle v_1, v_2, v_3 \rangle$, B^d_{ijk} alcanza un único máximo en el punto $\xi_{ijk} := (iv_1 + jv_2 + kv_3)/d$, de modo que*

$$B^d_{ijk}(v) < B^d_{ijk}(\xi_{ijk}) \quad v \neq \xi_{ijk} \quad v \in T.$$

Demostración. Vamos a ver primero el caso en el que $i, j, k > 0$. En este caso, ya hemos visto que $\xi_{ijk} \in T$. Para ver los máximos de B^d_{ijk} debemos ver las derivadas en dos direcciones independientes, que calculamos utilizando el Lema 1.5.2., que probaremos más adelante,

$$D_{v_1 - v_2} B^d_{ijk}(v) = B^d_{ijk}(v)(\frac{i}{b_1} - \frac{j}{b_2})$$

$$D_{v_1 - v_3} B^d_{ijk}(v) = B^d_{ijk}(v)(\frac{i}{b_1} - \frac{k}{b_3}).$$

Como sabemos que B^d_{ijk} es siempre positivo, las expresiones anteriores se anularán si y solo si $ib_2 - jb_1 = 0$ y $i(1 - b_1 - b_2) - kb_1 = 0$. La única solución de este sistema es $(b_1, b_2, b_3) = (i/d, j/d, k/d)$ como queríamos probar.

Todos los demás posibles casos sobre el signo de i, j, k se demuestran de forma análoga. □

Al conjunto de todos los puntos máximos de los polinomios de Bernstein B^d_{ijk} asociados a un triángulo T los vamos a definir de la siguiente manera:

$$\boxed{D_{d,T} := \{\xi_{ijk} := (iv_1 + jv_2 + kv_3)/d\}_{i+j+k=d}}$$

Por ejemplo, en un triángulo equilátero, para $d = 2$, serán los siguientes puntos:

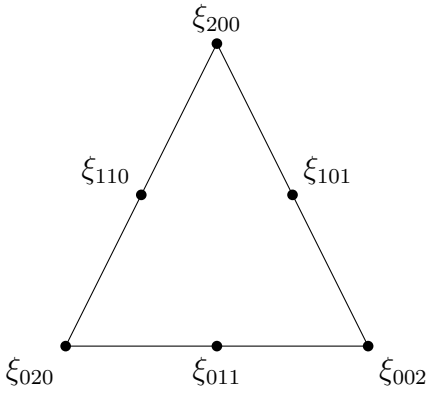


Figura 1.2: Puntos máximos de B^2_{ijk}

En el caso de $d = 3$, estos puntos serán los siguientes:

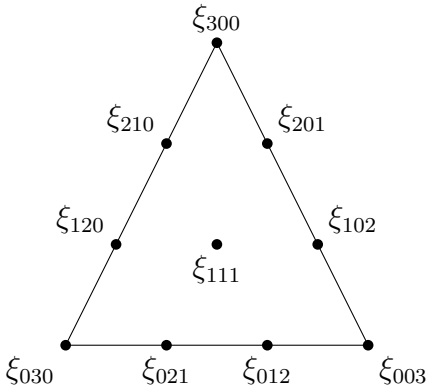


Figura 1.3: Puntos máximos de B^3_{ijk}

Y podríamos continuar sucesivamente para grados más altos.

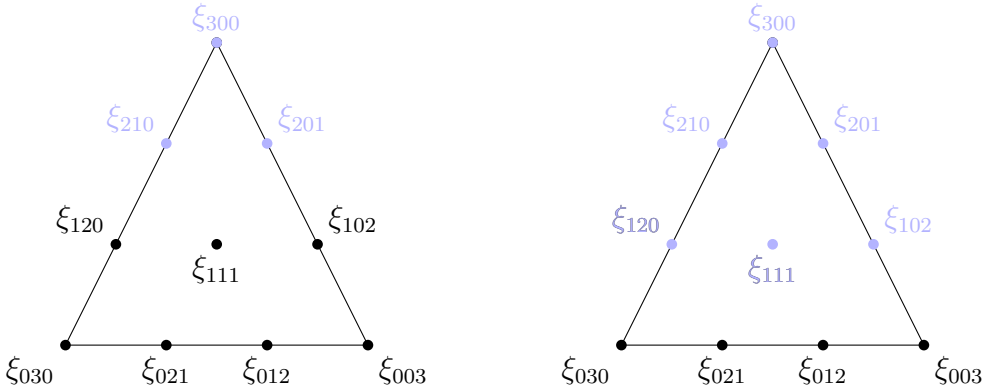
Dos conceptos que utilizaremos más adelante son los siguientes. Dado $0 \leq m \leq d$, nos referiremos al conjunto de puntos dominantes

$$R^T_m(v_1) := \{\xi_{d-m,j,m-j}\}$$

como el anillo de radio m alrededor del vértice v_1. Al conjunto

$$D^T_m(v_1) := \cup^m_{n=0} R^T_n(v_1)$$

lo llamaremos disco de radio m alrededor de v_1. De igual manera serán definidos para el resto de vértices.



(a) Puntos del disco $D_1^T(v_1)$ (b) Puntos del disco $D_2^T(v_1)$

Figura 1.4

1.3. B-forma

Como ya hemos visto en la sección anterior, los polinomios de Bernstein son una base del espacio de polinomios P_d. Por definición de base, para todos los polinomios p de grado d, existirán unos coeficientes $\{c_{ijk}\}_{i+j+k=d}$ que llamaremos **B-coeficientes** de p tales que

$$p = \sum_{i+j+k=d} c_{ijk} B_{ijk}^d \tag{1.3}$$

A esta forma de representar el polinomio la llamamos representación Bernstein Bezier, o más comúnmente, **B-forma** de p

También podemos escribir el polinomio p de forma única utilizando los puntos $D_{d,T} := \{\xi_{ijk} := (iv_1 + jv_2 + kv_3)/d\}_{i+j+k=d}$ de la forma

$$p = \sum_{\xi \in D_{d,T}} c_\xi B_\xi^d.$$

1.3.1. Almacenamiento de la B-forma

Podemos observar que para almacenar un polinomio expresado en su B-forma, es suficiente almacenar en un vector sus B-coeficientes y en otro los vértices del triángulo que estamos tomando de referencia.

Como convenio, se establece el orden de almacenaje de los B-coeficientes en un vector en orden lexicográfico. Es decir, si c_{ijk} va antes que c_{lmn} se debe a que $i > l$, ó si $i = l$, debe ser $j > m$, ó si $i = l$ y $j = m$, debe ser $k > n$.

Por ejemplo, en el caso $d = 2$, el vector estará ordenado de la siguiente manera:

$$c_{200}, c_{110}, c_{101}, c_{020}, c_{011}, c_{002}$$

En el caso $d = 3$, el vector estará ordenado de la siguiente manera:

$$c_{300}, c_{210}, c_{201}, c_{120}, c_{111}, c_{102}, c_{030}, c_{021}, c_{012}, c_{003}$$

1.4. Algoritmo de Casteljau

Supongamos que tenemos un polinomio p en su B-forma, pero no conocemos el valor de $p(v)$, siendo $v := (x, y)$ un punto con coordenadas baricéntricas (b_1, b_2, b_3). Vamos a explicar un algoritmo que nos permite evaluar $p(v)$.

Para entender la siguiente demostración, debemos tener en cuenta que

$$B^d_{ijk} := \frac{d!}{i!j!k!} b_1^i b_2^j b_3^k = b_1 B^{d-1}_{i-1,jk} + b_2 B^{d-1}_{ij-1,k} + b_3 B^{d-1}_{ijk-1}.$$

Los B-coeficientes de la B-forma de P los vamos a denotar de la siguiente manera:

$$c^{(0)}_{ijk} := c_{ijk}, \quad i + j + k = d.$$

Teorema 1.4.1. *Para todo $l = 1, ..., d$, sea*

$$c^{(l)}_{ijk} := b_1 c^{(l-1)}_{i+1,j,k} + b_2 c^{(l-1)}_{i,j+1,k} + b_3 c^{(l-1)}_{i,j,k+1},$$

entonces para todo $i + j + k = d - l$ se tiene que

$$p(v) = \sum_{i+j+k=d-l} c^{(l)}_{ijk} B^{d-l}_{ijk}(v). \tag{1.4}$$

Demostración. Vamos a demostrarlo por inducción sobre l. Obviamente para $l = 0$, (1.4) se cumple. Lo suponemos cierto para $l - 1$, es decir, cuando el grado es $d - l + 1$, y veamos que pasa para $d - l$. Por el resultado que hemos obtenido anteriormente,

$$
\begin{aligned}
p(v) &= \sum_{i+j+k=d-l+1} c_{ijk}^{(l-1)} B_{ijk}^{d-l+1}(v) \\
&= \sum_{i+j+k=d-l+1} c_{ijk}^{(l-1)} (b_1 B_{i-1,j,k}^{d-l}(v) + b_2 B_{i,j-1,k}^{d-l}(v) + b_3 B_{i,j,k-1}^{d-l}(v)) \\
&= \sum_{i+j+k=d-l+1,i1} b_1 c_{ijk}^{(l-1)} B_{i-1,j,k}^{d-l}(v) + \sum_{i+j+k=d-l+1,j1} b_2 c_{ijk}^{(l-1)} B_{i,j-1,k}^{d-l}(v) \\
&+ \sum_{i+j+k=d-l+1,k1} b_3 c_{ijk}^{(l-1)} B_{i,j,k-1}^{d-l}(v) \\
&= \sum_{i+j+k=d-l} b_1 c_{i+1,j,k}^{(l-1)} B_{i,j,k}^{d-l}(v) \\
&+ \sum_{i+j+k=d-l} b_2 c_{i,j+1,k}^{(l-1)} B_{i,j,k}^{d-l}(v) + \sum_{i+j+k=d-l} b_3 c_{i,j,k+1}^{(l-1)} B_{i,j,k}^{d-l}(v) \\
&= \sum_{i+j+k=d-l} c_{ijk}^{(l)} B_{ijk}^{d-l}(v)
\end{aligned}
$$

□

De este teorema, deducimos a donde queríamos llegar, que es que cuando $l = d$, como $B_{000}^0 \equiv 1$, de la expresión (1.4) se obtiene que

$$\boxed{p(v) = c_{000}^{(d)}}$$

1.4.1. Definición del algoritmo

Vamos a definir el algoritmo de Casteljau para, a partir de nuestros b-coeficientes, llegar a $c_{000}^{(d)} = p(v)$.

$$
\begin{gathered}
\text{Para } l = 1, ..., d \\
\text{Para } i + j + k = d - l \\
c_{ijk}^{(l)} := b_1 c_{i+1,j,k}^{(l-1)} + b_2 c_{i,j+1,k}^{(l-1)} + b_3 c_{i,j,k+1}^{(l-1)}
\end{gathered}
$$

Teorema 1.4.2. *Los coeficientes del algoritmo de Casteljau se pueden expresar como*

$$c_{ijk}^{(l)} = \sum_{\nu+\mu+\kappa=l} c_{i+\nu,j+\mu,k+\kappa} B_{\nu\mu\kappa}^{l}(v), \quad i + j + k = d - l.$$

Demostración. Si escribimos

$$c_{ijk}^{(1)} = (b_1E_1 + b_2E_2 + b_3E_3)c_{ijk},$$

siendo $E_1c_{ijk} = c_{i+1,j,k}$, $E_2c_{ijk} = c_{i,j+1,k}$ y $E_3c_{ijk} = c_{i,j,k+1}$, podemos escribir la fórmula del algoritmo como

$$c_{ijk}^{(l)} = (b_1E_1 + b_2E_2 + b_3E_3)c_{ijk}^{(l-1)}.$$

Repetimos esto $l-1$ veces y obtenemos que

$$\begin{aligned} c_{ijk}^{(l)} &= (b_1E_1 + b_2E_2 + b_3E_3)^l c_{ijk} \\ &= \sum_{\nu+\mu+\kappa=l} B_{\nu\mu\kappa}^l(v) E_1^\nu E_2^\mu E_3^\kappa c_{ijk} \\ &= \sum_{\nu+\mu+\kappa=l} c_{i+\nu,j+\mu,k+\kappa} B_{\nu\mu\kappa}^l(v). \end{aligned}$$

que demuestra lo que queríamos probar. □

1.5. Derivadas

En esta sección veremos cómo podemos escribir los coeficientes de la derivada direccional de un polinomio p escrito en su b-forma relativa a un triángulo T, utilizando también el algoritmo de Casteljau visto anteriormente.

Definición 1.5.1. Dada una función diferenciable f, su derivada direccional respecto un punto v de $\mathbb{R}^2$ con respecto de un vector director u en $\mathbb{R}^2$ se define como

$$D_u f(v) := \frac{d}{dt} f(v + tu)|_{t=0}.$$

Como sabemos que todo punto v de $\mathbb{R}^2$ se puede representar de forma única con sus coordenadas baricéntricas (b_1, b_2, b_3) que suman 1, es lógico pensar que todo vector $u = v - v_1$ de $\mathbb{R}^2$ se podrá definir únicamente con unas coordenadas direccionales baricéntricas de u con respecto de T, (a_1, a_2, a_3), que suman 0 y que cumplan que $a_i = b_i - c_i$ para $i = 1, 2, 3$ si (b_1, b_2, b_3) son las coordenadas baricéntricas de v y (c_1, c_2, c_3) son las coordenadas baricéntricas de v_1.

Primero veremos un lema que vamos a utilizar más adelante:

Lema 1.5.2. *Sea u un vector con coordenadas direccionales baricéntricas (a_1, a_2, a_3). Para cualquier $i + j + k = d$,*

$$D_u B_{ijk}^d(v) = d[a_1 B_{i-1,j,k}^{d-1}(v) + a_2 B_{i,j-1,k}^{d-1}(v) + a_3 B_{i,j,k-1}^{d-1}(v)] \tag{1.5}$$

Demostración. Si (b_1, b_2, b_3) son las coordenadas baricéntricas de un punto v, las del punto $v + tu$ serán $(b_1 + ta_1, b_2 + ta_2, b_3 + ta_3)$ y entonces

$$B^d_{ijk}(v + tu) = \frac{d!}{i!j!k!}[(b_1 + ta_1)^i(b_2 + ta_2)^j(b_3 + ta_3)^k].$$

Si diferenciamos con respecto de t y evaluamos en $t = 0$, llegamos por definición a

$$D_u B^d_{ijk}(v) = \frac{d!}{i!j!k!}[ib_1^{i-1}a_1b_2^jb_3^k + b_1^ijb_2^{j-1}a_2b_3^k + b_1^ib_2^jkb_3^{k-1}a_3],$$

y si agrupamos los términos correspondientes, obtenemos la expresión a la que queríamos llegar. □

Veremos cómo sería una derivada direccional de un polinomio p escrito en su B-forma relativa a un triángulo T.

Teorema 1.5.3. *Sea p un polinomio escrito en su B-forma relativa a un triángulo T, y sea u un vector director definido por sus coordenadas direccionales baricéntricas $a := (a_1, a_2, a_3)$, $a_1 + a_2 + a_3 = 0$, si llamamos $c^{(1)}_{ijk}(a)$ a los coeficientes tras un paso del algoritmo de Casteljau, se tiene que*

$$D_u p(v) = d \sum_{i+j+k=d-1} c^{(1)}_{ijk}(a) B^{d-1}_{ijk}(v).$$

Demostración. Aplicando la derivada direccional a $p(v)$, obtenemos

$$D_u p(v) = \sum_{i+j+k=d} c_{ijk} D_u B^d_{ijk}(v).$$

Si utilizamos (1.5) como vimos en el lema anterior,

$$D_u p(v) = \sum_{i+j+k=d} c_{ijk} d[a_1 B^{d-1}_{i-1,j,k}(v) + a_2 B^{d-1}_{i,j-1,k}(v) + a_3 B^{d-1}_{i,j,k-1}(v)],$$

reagrupando correctamente los coeficientes de $B^{d-1}_{ijk}(v)$ para $i+j+k = d-1$, obtenemos los $c^{(1)}_{ijk}$ y llegamos a la expresión buscada. □

Teorema 1.5.4. *Sea $1 \leq m \leq d$, y suponemos que nos dan un conjunto $u_1, ..., u_m$ de m direcciones descritas por sus coordenadas direccionales baricéntricas como,*

$$a^{(i)} := (a^{(i)}_1, a^{(i)}_2, a^{(i)}_3) \quad \text{con } a^{(i)}_1 + a^{(i)}_2 + a^{(i)}_3 = 0,$$

para $i = 1, ..., m$. *Entonces, siendo* $c_{ijk}^{(m)}(a^{(1)}, ..., a^{(m)})$ *los valores obtenidos tras realizar los* m *primeros pasos del algoritmo de Casteljau usando* $a^{(1)}, ..., a^{(m)}$ *sucesivamente en ese orden, tenemos,*

$$D_{u_m} \cdots D_{u_1} p(v) = \frac{d!}{(d-m)!} \sum_{i+j+k=d-m} c_{ijk}^{(m)}(a^{(1)}, ..., a^{(m)}) B_{ijk}^{d-m}(v). \quad (1.6)$$

Demostración. Si aplicamos el teorema 1.5.3 m veces, obtenemos la fórmula buscada. □

Si con $T := \langle v_1, v_2, v_3 \rangle$ tomamos un vector u no paralelo al lado $r := \langle v_2, v_3 \rangle$, vamos a ver la relación que hay entre los B-coeficientes de un polinomio p definido en T y las derivadas evaluadas en los puntos v del lado e.

Ya sabemos que si $a := (a_1, a_2, a_3)$ son las coordenadas direccionales de u, se tiene que,

$$D_u^m p(v) = \frac{d!}{(d-m)!} \sum_{j+k=d-m} c_{0jk}^{(m)}(a) B_{0jk}^{d-m}(v).$$

siendo $c_{0jk}^{(m)}(a)$ los coeficientes obtenidos de c_{ijk} tras aplicar m veces el algoritmo de Casteljau con el vector a.

Ahora veamos el reciproco. Supongamos que conocemos todos los coeficientes $\{c_{ijk}\}_{0 \le i \le m-1}$ de p. Estos se encuentran asociados al lado e y a las primeras $m-1$ filas paralelas a e. Veamos cómo podemos calcular los coeficientes $\{c_{mjk}\}_{j+k=d-m}$ a partir de los valores de las derivadas $D_u^m p(v)$ en los $d-m+1$ puntos a lo largo del lado e.

Lema 1.5.5. *Supongamos que conocemos los coeficientes* $\{c_{ijk}\}_{0 \le i \le m-1}$ *del polinomio* p. *También conocemos* $r := (D_u^m p(\eta_0), ..., D_u^m p(\eta_{d-m}))^T$ *para unos puntos distintos* $\eta_0, ..., \eta_{d-m}$ *en el interior de* $e := \{v_2, v_3\}$. *Entonces, los coeficientes* $c := (c_{m,d-m,0}, ..., c_{m,0,d-m})^T$ *se pueden calcular de forma única a partir de* r *y* $\{c_{ijk}\}_{0 \le i \le m-1}$.

Demostración. Si evaluamos la expresión,

$$D_u^m p(v) = \frac{d!}{(d-m)!} \sum_{j+k=d-m} c_{0jk}^{(m)}(a) B_{0jk}^{d-m}(v).$$

en los puntos $\eta_0, ..., \eta_{d-m}$, nos lleva al sistema de ecuaciones,

$$Mc = r,$$

siendo $c := (c^{(m)}_{0,d-m,0}, ..., c^{(m)}_{0,0,d-m})^T$ y siendo M la matriz de tamaño $(d-m+1) \times (d-m+1)$,

$$M := \frac{d!}{(d-m)!}[B^{d-m}_{0,d-m-j,j}(\eta_i)]^{d-m}_{i,j=0}.$$

Esta matriz contiene los valores de las bases de Bernstein y es no singular para todo $\eta_0, ..., \eta_{d-m}$ distintos. □

Lema 1.5.6. *Sea $m \leq \rho < d/2$. Supongamos que conocemos los coeficientes*

$$C := \bigcup_{i=0}^{m-1} \{c_{ijk}\}_{j+k=d-i} \cup \{c_\xi\}_{\xi \in D_\rho(v_2)} \cup \{c_\xi\}_{\xi \in D_\rho(v_3)}$$

del polinomio p. También conocemos $r := (D^m_u p(\eta_{\rho-m+1}), ..., D^m_u p(\eta_{d-\rho+1}))^T$ para unos puntos distintos $\eta_{\rho-m+1}, ..., \eta_{d-\rho+1}$ en el interior de $e := \{v_2, v_3\}$. Entonces, $c := (c_{m,d-\rho-1,\rho-m+1}, ..., c_{m,\rho-m+1,d-\rho-1})^T$ se pueden calcular de forma única a partir de r y C.

Demostración. Si evaluamos la expresión,

$$D^m_u p(v) = \frac{d!}{(d-m)!} \sum_{j+k=d-m} c^{(m)}_{0jk}(a) B^{d-m}_{0jk}(v).$$

en los puntos $\eta_{\rho-m+1}, ..., \eta_{d-\rho+1}$, nos lleva al sistema de ecuaciones,

$$Mc = r, \tag{1.7}$$

siendo $c^{(m)} := (c^{(m)}_{m,d-\rho-1,\rho-m+1}, ..., c^{(m)}_{m,\rho-m+1,d-\rho-1})^T$ y siendo M la matriz,

$$M := \frac{d!}{(d-m)!}[B^{d-m}_{0,d-m-j,j}(\eta_i)]^{d-\rho-1}_{i,j=\rho-m+1}.$$

Esta matriz, igual que en el teorema anterior, contiene los valores de las bases de Bernstein y es no singulas para todo $\eta_{\rho-m+1}, ..., \eta_{d-\rho+1}$ distintos. □

1.5.1. Definición del algoritmo

Vamos a definir un algoritmo para cuando tengamos dos vectores u y $\hat{u}$ con coordenadas direccionales (a_1, a_2, a_3) y $(\hat{a}_1, \hat{a}_2, \hat{a}_3)$ respectivamente. Como dijimos antes, (b_1, b_2, b_3) son las coordenadas baricéntricas del punto v. El algoritmo para computar $D^m_u D^{\hat{m}}_{\hat{u}} p(v)$ para un polinomio $p \in P_d$ y $0 \leq m + \hat{m} \leq d$.

- Para $l = 1, ..., m$
  Para todo $i + j + k = d - l$
  $c_{ijk}^{(l)} := a_1 c_{i+1,j,k}^{(l-1)} + a_2 c_{i,j+1,k}^{(l-1)} + a_3 c_{i,j,k+1}^{(l-1)}$
- Para $l = m + 1, ..., m + \hat{m}$
  Para todo $i + j + k = d - l$
  $c_{ijk}^{(l)} := \hat{a}_1 c_{i+1,j,k}^{(l-1)} + \hat{a}_2 c_{i,j+1,k}^{(l-1)} + \hat{a}_3 c_{i,j,k+1}^{(l-1)}$
- Para $l = m + \hat{m} + 1, ..., d$
  Para todo $i + j + k = d - l$
  $c_{ijk}^{(l)} := b_1 c_{i+1,j,k}^{(l-1)} + b_2 c_{i,j+1,k}^{(l-1)} + b_3 c_{i,j,k+1}^{(l-1)}$
- $D_u^m D_{\ddot{u}}^{\hat{m}} p(v) = d(d-1)(d-m-\hat{m}+1) c_{000}^{(d)}$

Aplicando el Teorema 1.5.3., se ve que los coeficientes con superíndice l, en el primer paso, son los coeficientes de $D_u^l p$, en el segundo paso, son los coeficientes de $D_u^m D_{\ddot{u}}^{l-m} p$ y el paso 3 simplemente es el algoritmo de Casteljau con los coeficientes de $D_u^m D_{\ddot{u}}^{\hat{m}} p$.

Por tanto, al igual que en el algoritmo descrito anteriormente para evaluar $p(v)$, nos basta con conocer $c_{000}^{(d)}$ para tener el valor de la doble derivada direccional $D_u^m D_{\hat{u}}^{\hat{m}} p(v)$ de un polinomio $p \in P_d$.

1.6. Condiciones de regularidad entre triángulos

A continuación, vamos a dar condiciones para la derivabilidad al unir dos polinomios en triángulos adyacentes $T = \langle v_1, v_2, v_3 \rangle$ y $\hat{T} = \langle v_4, v_3, v_2 \rangle$, que comparten el lado $e := \langle v_2, v_3 \rangle$.

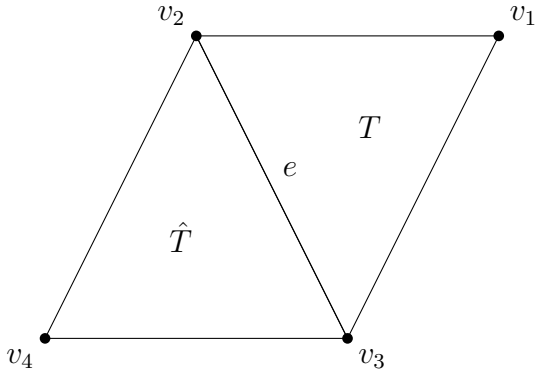


Figura 1.5

Si $\{B^d_{ijk}\}$ y $\{\hat{B}^d_{ijk}\}$ son las bases de Bernstein asociadas a T y a $\hat{T}$ respectivamente, tenemos el siguiente resultado:

Teorema 1.6.1. *Sean*

$$p(v) := \sum_{i+j+k=d} c_{ijk} B^d_{ijk}(v)$$

y

$$\hat{p}(v) := \sum_{i+j+k=d} \hat{c}_{ijk} \hat{B}^d_{ijk}(v)$$

con la notación anterior. Si u es un vector director no paralelo a e, se tiene que

$$D^n_u p(v) = D^n_u \hat{p}(v) \quad \textit{para } v \in e \textit{ y } n = 0, ..., r \tag{1.8}$$

si y solo si

$$\hat{c}_{njk} = \sum_{\nu+\mu+\kappa=n} c_{\nu,k+\mu,j+\kappa} B^n_{\nu\mu\kappa}(v_4) \quad j+k = d-n \quad n = 0, ..., r. \tag{1.9}$$

Demostración. Si restringimos p y $\hat{p}$ al lado e, cumplen las condiciones de continuidad a lo largo de e si y solo si

$$\hat{c}_{0jk} = c_{0kj}, \quad j+k = d,$$

que justamente es el caso en el que $r = 0$. Para probarlo en los casos $r > 0$, debemos observar que la prueba es equivalente para cuando diferenciamos respecto al vector $u = v_4 - v_2$, ya que las derivadas de p y $\hat{p}$ correspondientes a la dirección $v_3 - v_2$ coinciden en todos los puntos de e, y las derivadas en todas las demás direcciones serán combinaciones lineales de D_u y $D_{v_3-v_2}$.

Sean $b = (b_1, b_2, b_3)$ las coordenadas baricéntricas de v_4 respecto de T. Las coordenadas direccionales de u con respecto a T serán $a = (b_1, b_2 - 1, b_3)$ y con respecto a $\hat{T}$ serán $\hat{a} = (1, 0, -1)$. Para cada $0 \leq n \leq r$,

$$D^n_u p|_e = \frac{d!}{(d-n)!} \sum_{j+k=d-n} c^{(n)}_{0jk}(a) B^{d-n}_{0jk}$$

$$D^n_u \hat{p}|_e = \frac{d!}{(d-n)!} \sum_{j+k=d-n} \hat{c}^{(n)}_{0jk}(\hat{a}) \hat{B}^{d-n}_{0jk}$$

Donde $c^{(n)}_{0jk}(a)$ y $\hat{c}^{(n)}_{0jk}(\hat{a})$ son los coeficientes obtenidos tras aplicar n veces el algoritmo de Casteljau a $\{c_{ijk}\}$ y a $\{\hat{c}_{ijk}\}$ usando a y $\hat{a}$ respectivamente.

Como sabemos que para todos los puntos $v \in e$, se cumple que $\hat{B}^{d-n}_{0jk}(v) = B^{d-n}_{0kj}(v)$, para que se cumpla (1.8), debe cumplirse que

$$\hat{c}^{(n)}_{0jk}(\hat{a}) = c^{(n)}_{0kj}(a), \quad j+k = d-n, n = 0, ..., r \tag{1.10}$$

Siguiendo el algoritmo de Casteljau, tenemos

$$\hat{c}^{(n)}_{0jk}(\hat{a}) = \sum_{m=0}^{n} (-1)^{n-m} \binom{n}{m} \hat{c}_{m,j,d-j-m} \quad j+k = d-n.$$

Pero también por el algoritmo de Casteljau, si escribimos $c^{(1)}_{ijk} = (b_1E_1 + b_2E_2 + b_3E_3)c_{ijk}$, siendo $E_1c_{ijk} = c_{i+1,j,k}$, $E_2c_{ijk} = c_{i,j+1,k}$ y $E_3c_{ijk} = c_{i,j,k+1}$, podemos escribir la fórmula del algoritmo como

$$c^{(l)}_{ijk} = (b_1E_1 + b_2E_2 + b_3E_3)c^{(l-1)}_{ijk}$$

Con esta notación vemos que

$$\begin{aligned} c^{(n)}_{0kj}(a) &= (b_1E_1 + (b_2-1)E_2 + b_3E_3)^n c_{0kj} \\ &= (b_1E_1 + b_2E_2 + b_3E_3 - E_2)^n c_{0kj} \\ &= \sum_{m=0}^{n} (-1)^{n-m} \binom{n}{m} (b_1E_1 + b_2E_2 + b_3E_3)^m c_{0,k+n-m,j} \\ &= \sum_{m=0}^{n} (-1)^{n-m} \binom{n}{m} c^{(m)}_{0,d-j-m,j}(b), \quad j+k = d-n. \end{aligned}$$

Finalmente, de estas dos definiciones de $\hat{c}^{(n)}_{0jk}(\hat{a})$ y $c^{(n)}_{0kj}(a)$ tenemos que se cumple (1.10) si y solo si,

$$\hat{c}_{n,j,d-j-n} = c^{(n)}_{0,d-j-n,j}(b), \quad j = 0, ..., n, \quad n = 0, ..., r,$$

lo cual teniendo en cuenta el (teorema 1.4.2) es equivalente a (1.9), lo que demuestra el teorema. □

1.7. Derivadas en los vértices

En esta sección vamos a ver la relación que existe entre las derivadas de un polinomio p en un vértice, y sus correspondientes B-coeficientes asociados a los puntos dominantes en un disco de dicho vértice.

Un conjunto M de pares enteros no negativos es un conjunto inferior si para cada $(m, n) \in M$, todos los pares de la forma (i, j) con $0 \leq i \leq m$ y $0 \leq j \leq n$ también pertenecen a M.

Lema 1.7.1. *[2] Sea M un conjunto inferior, y*

$$f(m,n) = \sum_{i=0}^{m}\sum_{j=0}^{n}\binom{m}{i}\binom{n}{j}(-1)^{i+j}g(i,j), \quad \textit{para todo } (m,n) \in M.$$

Entonces,

$$g(i,j) = \sum_{m=0}^{i}\sum_{n=0}^{j}\binom{i}{m}\binom{j}{n}(-1)^{m+n}f(m,n), \quad \textit{para todo } (i,j) \in M.$$

Se demuestra con doble inducción.

Teorema 1.7.2. *Sea $T := \langle v_1, v_2, v_3\rangle$, fijamos $0 \leq \rho \leq d$. El conjunto de derivadas $\{D^m_{v_2-v_1}D^n_{v_3-v_1}p(v_1)\}_{m+n\leq\rho}$ pueden ser calculadas a partir del conjunto de B-coeficientes de p correspondientes a los puntos dominantes del disco $D^T_\rho(v_1)$ y viceversa.*

Demostración. Evaluando (1.6) en v_1, obtenemos que,

$$\boxed{\mathrm{D}^m_{v_2-v_1}D^n_{v_3-v_1}p(v_1) = \tfrac{(-1)^{m+n}d!}{(d-m-n)!}\sum_{i=0}^{m}\sum_{j=0}^{n}\binom{m}{i}\binom{n}{j}(-1)^{i+j}c_{d-i-j,i,j},} \quad (1.11)$$

para cualquier $0 \leq m+n \leq d$. Para ver la relación contraria, debemos aplicar el lema 1.7.1 a la expresión (1.11), para obtener:

$$\boxed{\mathrm{c}_{d-i-j,i,j} = \sum_{m=0}^{i}\sum_{n=0}^{j}\binom{i}{m}\binom{j}{n}\frac{(d-m-n)!}{d!}D^m_{v_2-v_1}D^n_{v_3-v_1}p(v_1).} \quad (1.12)$$

□

Gracias a este teorema, somos capaces de expresar las derivadas de un polinomio p en un vértice a partir de sus B-coeficientes asociados a los puntos dominantes en un disco de dicho vértice y viceversa.

1.8. Superficies de control

Definición 1.8.1. Dada una función f definida en un triángulo T, la superficie de $\mathbb{R}^3$

$$G_f := \{(x, y, f(x, y)) : (x, y) \in T\}$$

se define como el grafo de la función.

Si definimos el grafo anterior para un polinomio p en su B-forma $p := \sum_{i+j+k=d} c_{ijk} B^d_{ijk}$, obtenemos su B-parcela G_p.

Hay otra superficie que podemos definir asocida a p. Dado un triángulo $T = \langle v_1, v_2, v_3 \rangle$ y el conjunto de puntos ya definido $D_{d,T} := \{\xi_{ijk} = (iv_1 + jv_2 + kv_3)/d\}_{i+j+k=d}$, vamos a dividir el triángulo T uniendo todos los puntos vecinos ξ_{ijk} pertenecientes a $D_{d,T}$. Obtenemos de esta manera $\binom{d+1}{2} + \binom{d}{2}$ triángulos congruentes dentro de T de forma que

$$T_{ijk} := \langle \xi_{i+1,j,k}, \xi_{i,j+1,k}, \xi_{i,j,k+1} \rangle, \quad i+j+k = d-1,$$

$$\hat{T}_{ijk} := \langle \xi_{i,j+1,k+1}, \xi_{i+1,j,k+1}, \xi_{i+1,j+1,k} \rangle, \quad i+j+k = d-2.$$

Dado un polinomio p, si tenemos en cuenta

$$s_p(v) = \begin{cases} s_{ijk}(v) & para \quad v \in T_{ijk}, i+j+k = d-1 \\ \\ \hat{s}_{ijk}(v) & para \quad v \in \hat{T}_{ijk}, i+j+k = d-2 \end{cases}$$

donde s_{ijk} es el polinomio que interpola los valores en los vértices de T_{ijk}, y $\hat{s}_{ijk}$ es el polinomio que interpolar los valores en los vértices de $\hat{T}_{ijk}$. Bajo estas condiciones definimos $C_p := G_{s_p}$ el grafo asociado a s_p, al cual llamamos **superficie de control asociada a** p.

Es importante darse cuenta, que s_p es una función continua lineal a trozos, es decir un spline lineal C^0.

La unión de los lados de C_p nos da lugar a N_p, la **red de control asociada a** p.

1.9. Subdivisiones

Sea p un polinomio escrito en su B-forma en un triángulo $T = \langle v_1, v_2, v_3 \rangle$. Cualquier punto interior de T, lo divide en tres subtriángulos.

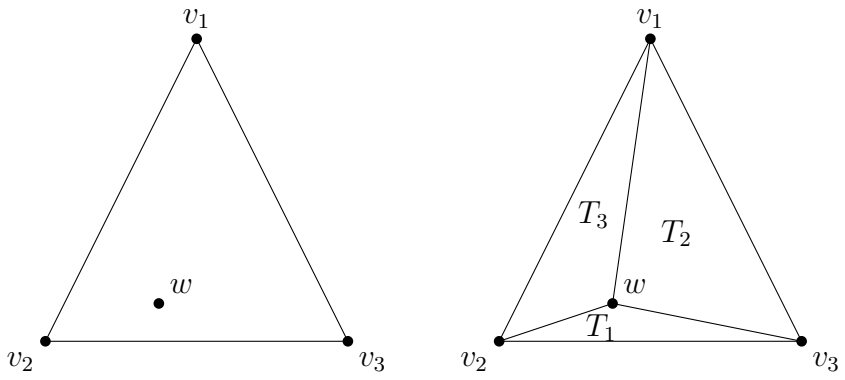


Figura 1.6: Triángulo dividido en 3.

Se tiene que

$$T_1 := \langle w, v_2, v_3 \rangle, \quad T_2 := \langle w, v_3, v_1 \rangle, \quad T_3 := \langle w, v_1, v_2 \rangle.$$

Vamos a ver cómo podemos escribir el polinomio p en función de cada uno de estos subtriángulos.

Teorema 1.9.1. *Dado un triángulo T y un punto w en su interior con coordenadas baricéntricas $a := (a_1, a_2, a_3)$. Para cada $T_l := \langle w, v_{l+1}, v_{l+2} \rangle$ con $l = 1, 2, 3$, los polinomios de Bernstein asociados serán $B_{ijk}^{T_l,d}$. Entonces para cualquier*

$$p = \sum_{i+j+k=d} c_{ijk} B_{ijk}^d,$$

se tiene que

$$p(v) = \begin{cases} \displaystyle\sum_{i+j+k=d} c_{0jk}^{(i)} B_{ijk}^{T_1,d}(v), & v \in T_1 \\ \displaystyle\sum_{i+j+k=d} c_{i0k}^{(j)} B_{ijk}^{T_2,d}(v), & v \in T_2 \\ \displaystyle\sum_{i+j+k=d} c_{ij0}^{(k)} B_{ijk}^{T_3,d}(v), & v \in T_3 \end{cases}$$

donde $c_{ijk}^{(\nu)}(a)$ son los coeficientes tras aplicar ν veces el algoritmo de Casteljau con $a := (a_1, a_2, a_3)$ y empezando en los coeficientes $c_{ijk}^{(0)} = c_{ijk}$.

Demostración. Vamos a realizar la prueba para uno de los triángulos, T_1, por ejemplo, y se realizará de manera análoga para los demás. Dado un $v \in T_1$,

tenemos que $\hat{b}_1(v), \hat{b}_2(v), \hat{b}_3(v)$ son las coordenadas baricéntricas de v relativas al triángulo T_1, por lo que

$$v = \hat{b}_1 w + \hat{b}_2 v_2 + \hat{b}_3 v_3.$$

Si sustituimos $w = a_1 v_1 + a_2 v_2 + a_3 v_3$, tendremos que,

$$v = \hat{b}_1 a_1 v_1 + (\hat{b}_1 a_2 + \hat{b}_2) v_2 + (\hat{b}_1 a_3 + \hat{b}_3) v_3.$$

Esto implica que, para todo $\nu + \beta + \gamma = d$,

$$\begin{aligned} B^{T,d}_{\nu\beta\gamma}(v) &= \frac{d!(\hat{b}_1 a_1)^\nu (\hat{b}_1 a_2 + \hat{b}_2)^\beta (\hat{b}_1 a_3 + \hat{b}_3)^\gamma}{\nu!\beta!\gamma!} \\ &= \sum_{\mu=0}^{\beta} \sum_{\kappa=0}^{\gamma} B^{T_1,d}_{\nu+\mu+\kappa,\beta-\mu,\gamma-\kappa}(v) B^{T,\nu+\mu+\kappa}_{\nu\mu\kappa}(w). \end{aligned}$$

Si sustituimos en

$$p(v) = \sum_{\nu+\beta+\gamma=d} c_{\nu\beta\gamma} B^{T,d}_{\nu\beta\gamma}(v),$$

tendremos que

$$p(v) = \sum_{\nu+\beta+\gamma=d} c_{\nu\beta\gamma} \sum_{\mu=0}^{\beta} \sum_{\kappa=0}^{\gamma} B^{T_1,d}_{\nu+\mu+\kappa,\beta-\mu,\gamma-\kappa}(v) B^{T,\nu+\mu+\kappa}_{\nu\mu\kappa}(w).$$

Si elegimos el caso en el que $\beta = j + \mu, \gamma = k + \kappa$ y $\nu + \mu + \kappa = i$, vemos que el coeficiente correspondiente a $B^{T_1,d}_{ijk}$ es

$$\sum_{\nu+\mu+\kappa=i} c_{\nu,j+\mu,k+\kappa} B^{T,i}_{\nu\mu\kappa}(w).$$

Por el Teorema 1.4.2, esto es igual a $c^{(i)}_{0jk}$, lo que demuestra la igualdad que queríamos demostrar para $v \in T_1$. □

Para el caso $d = 3$, podemos visualizar estos coeficientes de p de la siguiente manera:

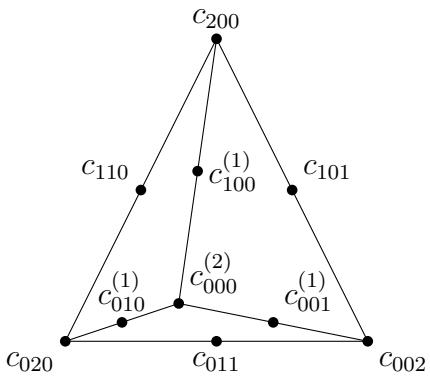


Figura 1.7: Coeficientes de subdivisión

Lo visto en el teorema anterior, es también válido si el punto v pertenece a un lado de T. En este caso, solo se dividirá en dos triángulos y será más sencillo operar.

1.10. Elevación de grado

En esta sección, partiremos de que cualquier polinomio de grado d se puede expresar como un polinomio de grado $\hat{d}$ siendo $\hat{d} > d$, y veremos cómo encontrar los B-coeficientes de p expresado en un grado mayor.

Veremos cómo podemos aumentar el grado en una unidad.

Teorema 1.10.1. *Sea p un polinomio definido en un triángulo T escrito en su B-forma. Entonces si,*

$$p = \sum_{i+j+k=d+1} c^{[d+1]}_{ijk} B^{[d+1]}_{ijk},$$

siendo $c^{[d]}_{ijk} := c_{ijk}$ sus coeficientes y B^{d+1}_{ijk} los polinomios de Bernstein de grado $d+1$ asociados a T se tiene que para $i+j+k = d+1$,

$$c^{[d+1]}_{ijk} := \frac{ic^{[d]}_{i-1,j,k} + jc^{[d]}_{i,j-1,k} + kc^{[d]}_{i,j,k-1}}{d+1}. \tag{1.13}$$

Demostración. Si en

$$p = \sum_{i+j+k=d} c_{ijk} B^d_{ijk},$$

multiplicamos ambos lados por $1 = b_1 + b_2 + b_3$, obtenemos,

$$p = \sum_{i+j+k=d} c^{[d]}_{ijk} \frac{d!}{i!j!k!} b_1^i b_2^j b_3^k (b_1 + b_2 + b_3).$$

Si multiplicamos y reordenamos los términos, obtenemos el término de la igualdad (1.13). □

En la igualdad (1.13) se debe tener en cuenta que los coeficientes con subíndice negativo, son igual a cero por convenio.

Si queremos elevar el grado en más de una unidad, repetiremos lo visto tantas veces como queramos.

1.11. Bases duales para los polinomios de Bernstein

Llamaremos base dual de $B^d = \{B^d_{ijk}\}_{i+j+k=d}$ al conjunto de funcionales lineales $\{\lambda_{ijk}\}_{i+j+k=d}$ definidos en el espacio de polinomios de dimensión d, P_d con la propiedad de que

$$\lambda_{\nu\mu\kappa} B_{ijk} = \delta_{ijk,\nu\mu\kappa}, \quad \text{para } i+j+k=d \text{ y } \nu+\mu+\kappa=d,$$

siendo $\delta_{ijk,\nu\mu\kappa} = 1$ si $(i,j,k) = (\nu,\mu,\kappa)$ y $\delta_{ijk,\nu\mu\kappa} = 0$ si $(i,j,k) \neq (\nu,\mu,\kappa)$. Vamos a definir explicitamente estos funcionales. Siendo $m := \binom{d+2}{2}$ la dimensión de B^d, vamos a definir los polinomios de Bernstein $\{B^d_{ijk}\}_{i+j+k=d}$ como $\{g_1, ..., g_m\}$, y los puntos dominantes asociados $\{\xi_{ijk}\}_{i+j+k=d}$ como $\{t_1, ..., t_m\}$. Sea M la matriz

$$M := [g_j(t_i)]_{i,j=1}^n,$$

dados $i+j+k=d$, vamos a definir r^{ijk} como el vector de dimensión m con ceros en todas las posiciones excepto un uno en la posición

$$l := \binom{d-i+1}{2} + d - i - j + 1,$$

tal que $t_l = \xi_{ijk}$.

Si llamamos $a^{ijk} := (a_1^{ijk}, ..., a_m^{ijk})^T$ a la solución del sistema

$$Ma^{ijk} = r^{ijk},$$

vamos a definir,

$$\boxed{\lambda_{ijk} p := \sum_{\nu=1}^{m} a_\nu^{ijk} p(t_\nu)}$$

Teorema 1.11.1. *Los funcionales lineales $\{\lambda_{ijk}\}_{i+j+k=d}$ forman una base dual de $B^d = \{B^d_{ijk}\}_{i+j+k=d}$.*

Demostración. Del sistema

$$Ma^{ijk} = r^{ijk},$$

deducimos que para todo $\nu + \mu + \kappa = d$ se tiene que,

$$\lambda_{\nu\mu\kappa} B^d_{ijk} = \begin{cases} 0, & (\nu, \mu, \kappa) \neq (i, j, k) \\ \\ 1, & (\nu, \mu, \kappa) = (i, j, k) \end{cases}$$

de donde inmediatamente se deduce la dualidad.

□

Capítulo 2

Triangulaciones

En este capítulo vamos a ver qué es una triangulación y varias propiedades de estas. También veremos algunos tipos de triangulaciones con las que trabajaremos más adelante.

2.1. Definiciones

Definición 2.1.1. Llamamos a un conjunto $\Delta := \{T_1, ..., T_{n_t}\}$ una triangulación de $\Omega = \cup_{i=1}^{n_t} T_i$ si toda intersección entre dos triángulos de Δ es un vértice común o un lado común.

Es una definición un poco genérica, ya que también incluiría un conjunto de triángulos completamente separados, o una triangulación en un dominio Ω con agujeros en su interior.

Definición 2.1.2. Llamamos vértices de la triangulación $\Delta := \{T_1, ..., T_{n_t}\}$ a todos los vértices de los triángulos de Δ.

Un vértice es vértice frontera, si es un punto frontera de Ω. En caso contrario, es un vértice interior. A todos los vértices frontera los llamaremos ν_F y a los interiores ν_I.

Definición 2.1.3. Llamamos lados de la triangulación $\Delta := \{T_1, ..., T_{n_t}\}$ a todos los lados de los triángulos de Δ.

Un lado es lado frontera, si es un lado frontera de Ω. En caso contrario, es un lado interior. A todos los lados frontera los llamaremos ε_F y a los interiores ε_I.

En la siguiente figura, vemos un ejemplo de una triangulación sobre un octógono irregular cualquiera. Podemos observar que tiene 11 vértices, de los

cuales, 8 son vértices frontera y 3 son vértices interiores. También tiene 8 lados frontera, y 14 lados interiores.

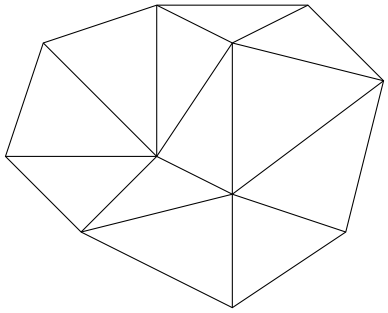


Figura 2.1: Triangulación de un octógono irregular.

2.2. Puntos de dominio en una triangulación

Al conjunto de todos los puntos máximos de los polinomios de Bernstein asociados a un triángulo T, los definíamos como

$$D_{d,T} := \{\xi_{ijk} := (iv_1 + jv_2 + kv_3)/d\}_{i+j+k=d}.$$

Ahora suponemos que $\Delta = \{T_i\}_{i=1}^{n_t}$ es una triangulación de un dominio Δ. El conjunto de puntos dominantes ahora será:

$$D_{s,\Delta} := \cup_{T\in\Delta} D_{d,T}. \tag{2.1}$$

Los puntos de este conjunto que se encuentren en un lado compartido por dos triángulos, solo se incluirán una vez.

Por ejemplo, para la triangulación dibujada en la Figura 2.1, tenemos que los puntos de dominio para $d = 2$, serán los siguientes.

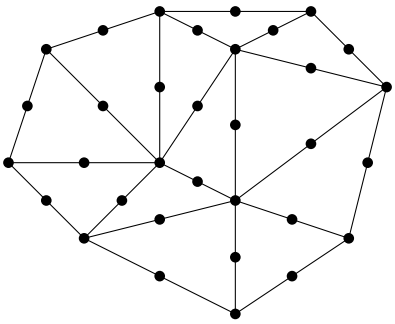


Figura 2.2: Puntos dominantes de una triangulación.

2.3. Estrellas

Definición 2.3.1. Sea v un vértice de una triangulación Δ. Llamamos estrella de v, $estrella(v)$ al conjunto de triángulos al rededor de v.

Por ejemplo, para la triangulación dibujada en la Figura 2.1, tenemos que la estrella de v, siendo v el vértice marcado en la imagen, es la siguiente,

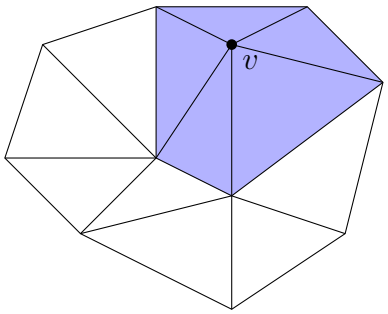


Figura 2.3: Estrella del vértice v.

Usualmente, denotamos $estrella^1(v) := estrella(v)$, y para $i > 1$, $estrella^i(v)$ es el conjunto de triángulos de Δ que tienen intersección no vacía con $estrella^{i-1}(v)$.

Podemos observar que en el ejemplo anterior, $estrella^2(v) = \Delta$, es toda la triangulación, ya que tiene intersección no vacía con todos los triángulos restantes.

A partir de esto podemos también definir estrellas de triángulos.

Definición 2.3.2. Sea $estrella^0(T) := T$. Para todo $j \geq 1$, definiremos la estrella de T de orden j como $estrella^j(T) := \cup\{estrella(v) : v \in estrella^{j-1}(T)\}$

Un ejemplo para ver la $estrella(T)$ siendo T uno de los triángulos de la triangulación de la Figura 2.1, es el siguiente.

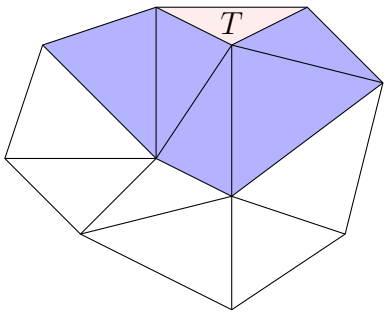


Figura 2.4: Estrella del triángulo T.

2.4. Almacenamiento

Los elementos necesarios para determinar una triangulación son los siguientes:

- Número de vértices n_v.
- Número de triángulos n_T.
- Vector x y vector y de tamaño n_v tal que (x_i, y_i) sean las coordenadas cartesianas del i-ésimo vértice.
- Matriz de tamaño $n_T \times 3$ que contiene en cada fila los índices de los vértices del triángulo correspondiente.

También se pueden utilizar otras técnicas para almacenarlas, como por ejemplo, en vez de usar una matriz con los vértices de los triángulos, podemos usar una matriz con los índices de los lados de la triangulación, o una lista con los vértices adyacentes a uno dado, o una lista con los lados saliendo de cada vértice que encontramos si recorremos cada triángulo, o alguna otra manera similar. La más usada e intuitiva es la que primero hemos descrito.

Hay otros parámetros adicionales que nos dan información sobre una triangulación como pueden ser el número de vértices frontera, el área de cada triángulo, etc.

2.5. Refinamientos

En esta sección veremos el concepto de refinamiento de una triangulación sobre el mismo dominio Ω. Veremos alguna propiedad y los diferentes tipos de refinamientos que nos serán útiles más adelante.

Supongamos que tenemos dos triangulaciones Δ y Δ_R del mismo dominio Ω. Diremos que Δ_R es un refinamiento de Δ si sucede lo siguiente:

- Todo vértice de Δ es también vértice de Δ_R.
- Todo triángulo $t \in \Delta_R$ es un subtriángulo de algún triángulo T en Δ.

Hay muchos tipos de refinamientos que podemos usar. Uno de los más comunes es el refinamiento uniforme, que consiste en dividir cada triángulo T de una triangulación Δ en cuatro triángulos conectando los puntos medios de los tres lados de T.

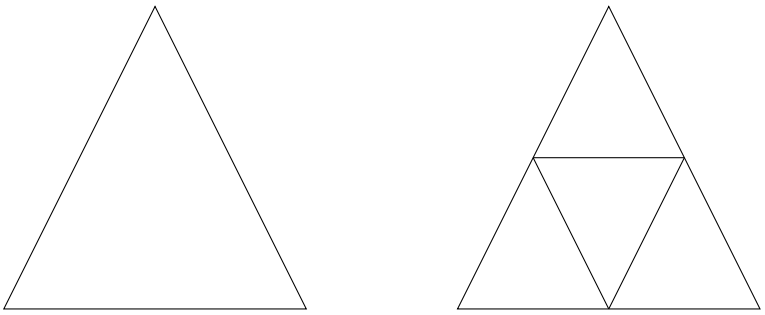


Figura 2.5: Refinamiento uniforme

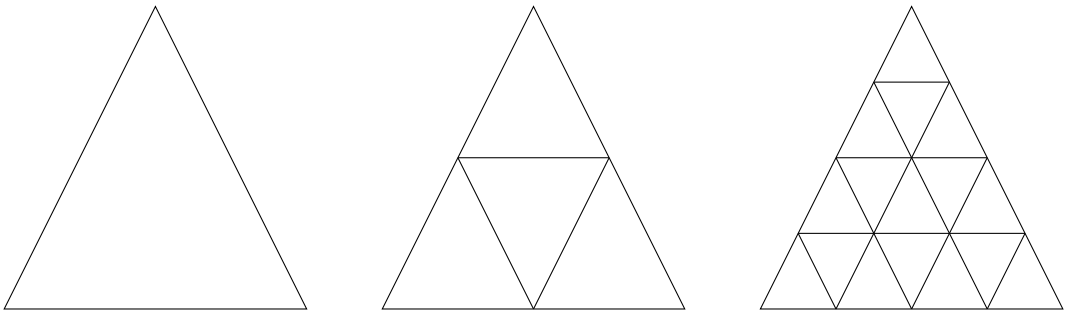


Figura 2.6: Refinamiento uniforme sobre refinamiento uniforme

A continuación, vamos a ver algunos tipos de refinamientos que nos interesarán más adelante a la hora de construir macro elementos.

2.5.1. Clough-Tocher

Para cada triánlo T de la triangulación Δ:

- Encontramos su baricentro $u_T := (v_1 + v_2 + v_3)/3$.
- Unimos u_T a cada vértice de T.

Como resultado, tendremos cada triángulo T de la triangulación Δ dividido en 3 triángulos.

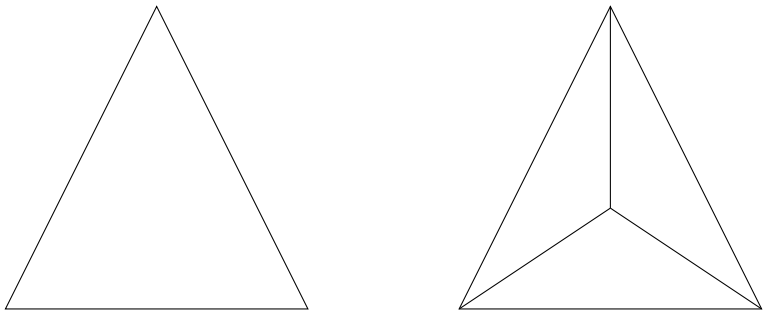


Figura 2.7: Refinamiento de Clough-Tocher

Esto aplicado a todos los triángulos T de la triangulación Δ, da lugar al refinamiento de Clough-Tocher Δ_{CT}.

2.5.2. Powell-Sabin

Para cada triánlo T de la triangulación Δ:

- Encontramos su incentro u_T.
- Unimos u_T a cada vértice de T.
- Debemos conectar u_T y $u_{\hat{T}}$ siempre que T y $\hat{T}$ tengan un lado común, siendo $\hat{T}$ otro triángulo de la triangulación Δ.
- Para los lados frontera de la triangulación, debemos conectar el punto medio de estos lados al incentro del triángulo al que pertenezcan.

Como resultado, tendremos cada triángulo T de la triangulación Δ dividido en 6 triángulos.

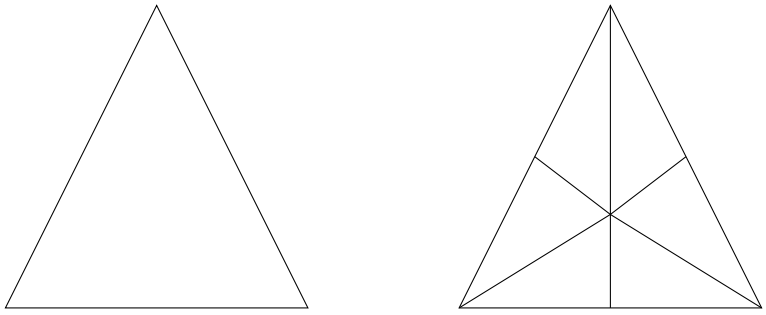


Figura 2.8: Refinamiento de Powell-Sabin

Esto aplicado a todos los triángulos T de la triangulación Δ, da lugar al refinamiento de Powell-Sabin Δ_{PS}.

2.5.3. Powell-Sabin 12

Para cada triánlo T de la triangulación Δ:

- Realizamos el refinamiento de Clough-Tocher antes visto, basado en dividir cada triángulo teniendo en cuenta su baricentro u_T.
- Unimos el baricentro u_T del triángulo a los puntos medios de sus lados.
- Unimos los puntos medios de los lados entre sí.

Como resultado, tendremos cada triángulo T de la triangulación Δ dividido en 12 triángulos.

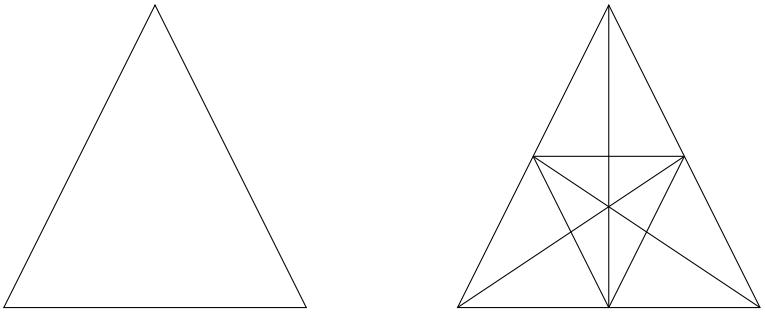


Figura 2.9: Refinamiento de Powell-Sabin-12

Esto aplicado a todos los triángulos T de la triangulación Δ, da lugar al refinamiento de Powell-Sabin-12 Δ_{PS12}.

2.5.4. Wang

Para cada triánlo $T = \langle v_1, v_2, v_3 \rangle$ de la triangulación Δ:

- Consideramos los siguientes puntos:

$$w_1^T := \frac{4v_1 + 2v_2 + v_3}{7},$$
$$w_2^T := \frac{v_1 + 4v_2 + 2v_3}{7},$$
$$w_3^T := \frac{2v_1 + v_2 + 4v_3}{7}$$

- Tomamos $T^* := \langle w_1, w_2, w_3 \rangle$ como triángulo central dentro del triángulo T.

- Unimos cada vértice w_i de T^* a todos los vértices v_i de T para $i = 1, 2, 3$.

Como resultado, tendremos cada triángulo T de la triangulación Δ dividido en 7 triángulos.

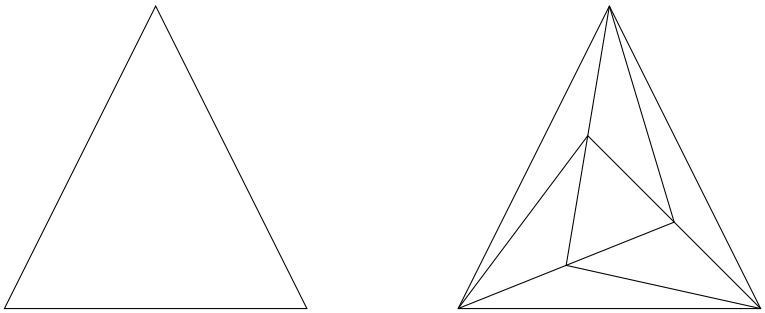


Figura 2.10: Refinamiento de Wang

Esto aplicado a todos los triángulos T de la triangulación Δ, da lugar al refinamiento de Wang Δ_W.

Capítulo 3

Métodos Bernstein-Bézier para espacios de Splines

En este capítulo, vamos a ver cómo podemos representar splines en su B-forma, que ventajas tiene esto, y varias propiedades de estos.

3.1. El espacio $S_d^0(\Delta)$

Como vimos en el capítulo anterior, el conjunto de puntos de dominio de una triangulación se representan como en (2.1). Vamos a usarlos para parametrizar el espacio de splines C^0 de grado d. Este espacio vectorial lo definiremos como:

$$S_d^0(\Delta) := \{s \in C^0(\Omega) : s|_{T_i} \in P_d, i = 1, ..., N\} \tag{3.1}$$

Es decir, estos elementos restringidos a cada triángulo de la triangulación son polinomios de grado d y son continuos en todo el dominio Ω.

Como S_d^0 está formado por elementos de P_d, ya vimos que para cada triángulo $T \in \Delta$ y dado $s \in S_d^0(\Delta)$, existe un único conjunto de coeficientes $\{c_\xi\}_{\xi \in D_{d,T}}$ tales que

$$s|_T = \sum_{\xi \in D_{d,T}} c_\xi B_\xi^{T,d}$$

Por la definición de S_d^0, s es continuo, por lo que cuando ξ se encuentre en un lado común de dos triángulos, T y $\hat{T}$ por ejemplo, los coeficientes c_ξ de $s|_T$ y $s|_{\hat{T}}$ serán iguales. Por tanto, cada spline $s \in S_d^0\Delta$ tiene unos coeficientes únicos $\{c_\xi\}_{\xi \in D_{d,\Delta}}$, llamados **B-coeficientes de s** y cada conjunto

de coeficientes $\{c_\xi\}_{\xi \in D_{d,\Delta}}$, tienen un único spline asociado definido por

$$s = \sum_{\xi \in D_{d,\Delta}} c_\xi B_\xi^{T,d}$$

Esto da lugar a dos resultados:

Teorema 3.1.1. *[2] Todo spline $s \in S_d^0(\Delta)$ está determinado de forma única por su conjunto de B-coeficientes $\{c_\xi\}_{\xi \in D_{d,\Delta}}$.*

Teorema 3.1.2. *[2] Si llamamos n_v, n_e y n_t al número de vértices, lados y triángulos de una triangulación Δ, se tiene que*

$$dim S_d^0(\Delta) = \#D_{d,\Delta} = n_v + (d-1)n_e + \binom{d-1}{2} n_t.$$

3.2. Almacenamiento, representación y evaluación

Como hemos visto en la sección anterior, para almacenar toda la información de un spline $s \in S_d^0(\Delta)$, es suficiente con almacenar un vector c con sus B-coeficientes, suponiendo que tenemos también la información de la triangulación Δ. Hay que acordar un convenio para guardar los coeficientes en un determinado orden.

Suponemos que los triángulos y lados están orientados, cada triángulo $T := \langle v_1, v_2, v_3 \rangle$ con los vértices y lados en sentido antihorario. Para almacenar los B-coeficientes, listamos primero los puntos de los vértices, luego los de los lados orientados y finalmente los del interior de cada triángulo en el orden lexicográfico que describimos para almacenar la B-forma de un polinomio sobre un triángulo. Por tanto, como vimos en la sección anterior, el vector c tendrá un total de $n_v + (d-1)n_e + \binom{d-1}{2} n_t$ elementos.

Veamos cómo podemos evaluar un spline s en un punto del plano dado v. Primero debemos identificar en qué triángulo está el punto a evaluar, y a continuación seguiremos un proceso parecido al que seguíamos para usar el algoritmo de Casteljau. El algoritmo para evaluar en $v = (x, y)$ el spline $s \in S_d^0(\Delta)$ conociendo el vector c, es el siguiente.

1. Debemos encontrar el triángulo T que contiene a v. Una técnica es calcular las coordenadas baricéntricas del punto relativas a un triángulo y comprobar sus signos para identificar donde se encuentra respecto al triángulo T.

2. Una vez conocemos el triángulo T que lo contiene, obtenemos del vector c los B-coeficientes de $s|_T$.

3. Usamos el algoritmo de Casteljau como vimos en la sección 1.4, es decir, realizamos la siguiente recurrencia:

$$\begin{gathered}\text{Para } l = 1, ..., d \\ \text{Para } i + j + k = d - l \\ c_{ijk}^{(l)} := b_1 c_{i+1,j,k}^{(l-1)} + b_2 c_{i,j+1,k}^{(l-1)} + b_3 c_{i,j,k+1}^{(l-1)}\end{gathered}$$

Tras esto, llegaremos al valor de $c_{000}^{(d)} = s(v)$.

Para evaluar las derivadas de s, seguiremos el mismo algoritmo, pero evaluaremos las derivadas de $s|_T$ como ya habíamos visto, en vez de evaluar simplemente $s|_T$.

3.3. Derivabilidad en $S_d^{r,p}(\Delta)$

Podemos pedir que el espacio de splines no solo sea continuo.

Definición 3.3.1. Llamaremos espacio de splines de grado d y derivabilidad r al espacio

$$S_d^r(\Delta) := C^r(\Omega) \cap S_d^0(\Delta)$$

para r y d enteros y $0 \leq r < d$.

Muchas veces, vamos a necesitar trabajar con splines que tienen más derivabilidad en ciertos vértices y no en los demás puntos.

Definición 3.3.2. Un spline $s \in S_d^0(\Delta)$ diremos que es C^ρ en un vértice v si para todos los triángulos T que compartan vértice v, se cumple que $s|_T$ tienen derivadas comunes hasta orden ρ en ese vértice y escribiremos $s \in C^\rho(v)$.

Esto nos permite definir un nuevo espacio de supersplines cuando esto sucede con todos los vértices $V := \{v_1, ..., v_{n_v}\}$ de una triangulación, para $0 \leq r \leq \rho \leq d$.

$$\boxed{S_d^{r,\rho}(\Delta) := \{s \in S_d^r(\Delta) : s \in C^\rho(v) \text{ para todo } v \in V\}.}$$

Podemos generalizar aún más este super espacio si asignamos una derivabilidad diferente a cada vértice de la triangulación. Para ello, definiremos $\rho := \{\rho_v\}_{v \in V}$ con $0 \leq r \leq \rho_v \leq d$, y el espacio se definirá como,

$$\boxed{S_d^{r,\rho}(\Delta) := \{s \in S_d^r(\Delta) : s \in C^{\rho_v}(v) \text{ para todo } v \in V\}.}$$

3.3.1. Derivabilidad en los lados

También podemos definir condiciones para que un spline en dos triángulos adyacentes tenga regularidad C^r, como vimos en el Teorema 1.6.1. La notación utilizada es la siguiente. Estos dos triángulos son $T := \langle v_1, v_2, v_3\rangle$ y $\hat{T} := \langle v_2, v_3, v_4\rangle$, que comparten el lado $e := \langle v_2, v_3\rangle$. Sean $\{B^d_{ijk}\}$ y $\{\hat{B}^d_{ijk}\}$ los polinomios de Bernstein de s asociados a T y a $\hat{T}$, y $\{c_{ijk}\}_{i+j+k=d}$, $\{\hat{c}_{ijk}\}_{i+j+k=d}$ sus coeficientes correspondientes, tenemos que

$$s|_T(v) := \sum_{i+j+k=d} c_{ijk} B^d_{ijk}(v)$$

$$s|_{\hat{T}}(v) := \sum_{i+j+k=d} \hat{c}_{ijk} \hat{B}^d_{ijk}(v).$$

Para cualquier spline $s \in S^0_d(\Delta)$, vamos a definir el funcional diferencial de orden n,

$$\tau^n_{j,e} s := c_{n,d-j,j-n} - \sum_{\nu+\mu+\kappa=n} \hat{c}_{\nu,j-n+\mu,d-j+\kappa} \hat{B}^n_{\nu\mu\kappa}(v_1). \tag{3.2}$$

Dado un conjunto Γ de funcionales lineales $\tau^n_{j,e} s$, asociados con los lados de la triangulación Δ, definimos el espacio de splines diferenciables como,

$$S^\Gamma_d(\Delta) := \{s \in S^0_d(\Delta) : \tau^n_{j,e} s = 0 \text{ para todo } \tau^n_{j,e} \in \Gamma\}$$

3.3.2. Derivabilidad en los vértices

Siguiendo la notación del apartado anterior, como el Teorema 1.6.1 nos afirmaba que si u es un vector director no paralelo a e, se tiene que

$$D^n_u p(v) = D^n_u \hat{p}(v) \quad \text{para } v \in e \text{ y } n = 0, ..., r$$

si y solo si

$$\hat{c}_{njk} = \sum_{\nu+\mu+\kappa=n} c_{\nu,k+\mu,j+\kappa} B^n_{\nu\mu\kappa}(v_4) \quad j + k = d - n, \quad n = 0, ..., r.$$

Por cómo hemos definido $\tau^n_{j,e}$, este teorema es equivalente a que un spline en $S^\Gamma_d(\Delta)$ tendrá regularidad C^r a lo largo del lado e siempre que Γ incluya todos los funcionales lineales $\{\tau^n_{j,e}\}^d_{j=n}$, para $n = 1, ..., r$. Veamos cómo afecta esto a los lados.

Lema 3.3.3. *Sea $s \in S_d^0(\Delta)$. Sea v un vértice de Δ y sean $e_1, ..., e_m$ los lados interiores de Δ unidos a v. Si para todo $i = 1, ..., m$, se tiene que*

$$\tau_{j,e_i}^n s = 0, \text{ con } n \leq j \leq \rho \text{ y } 1 \leq n \leq \rho, \tag{3.3}$$

entonces

$$s \in C^\rho(v).$$

Demostración. Dado $s \in S_d^0(\Delta)$, podemos mirar sus B-coeficientes asociados al dominio de puntos en el disco $D_\rho(v)$ como los coeficientes de un spline g en $S_\rho^0(\Delta)$. Entonces $s \in C^\rho(v)$ si y solo si g reduce a un único polinomio. Pero g reduce a un polinomio si y solo si se cumple que $\tau_{j,e_i}^n s = 0,$ con $n \leq j \leq \rho$ y $1 \leq n \leq \rho$. □

Teorema 3.3.4. *Su pongamos que para algún vértice v, $s \in S_d^0(\Delta) \cap C^\rho(v)$ y sean $T_1, ..., T_m$ los triángulos de Δ que comparten el vértice v. Podemos fijar los coeficientes de s correspondientes a los puntos dominantes de $D_\rho(v) \cap T_1$ como valores arbitrarios, y los coeficientes correspondientes a los demás puntos de $D_\rho(v)$ serán determinados por las condiciones de regularidad.*

Demostración. Como vimos en la demostración del lema anterior, podemos asociar los coeficientes de s correspondientes a los puntos dominantes del disco $D_\rho(v)$ como coeficientes de un polinomio g de grado d. Podemos fijar los correspondientes coeficientes correspondientes a puntos dominantes de $D_\rho(v) \cap T_1$ como valores arbitrarios y esto determinará las derivadas $\{D_x^\alpha D_y^\beta s(v)\}_{0 \leq \alpha+\beta \leq \rho}$, que a su vez determina los coeficientes de g restringido a cada uno de los demás triángulos. $T_2, ..., T_m$. □

3.4. Conjuntos minimales de determinación

Sabemos que todo spline s perteneciente a $S_d^0(\Delta)$ está determinado de forma única por sus B-coeficientes $\{c_\xi\}_{\xi \in D_{d,\Delta}}$, siendo $D_{d,\Delta}$ el conjunto de puntos dominantes de $S_d^0(\Delta)$.

Supongamos que $S := S_d^\Gamma(\Delta)$ es un subespacio vectorial de $S_d^0(\Delta)$. Para que un spline $s \in S_d^0(\Delta)$ pertenezca a S, tiene que cumplir unas condiciones de regularidad, no se le pueden asignar valores aleatorios a sus coeficientes.

Definición 3.4.1. Supongamos M es un subconjunto de $D_{d,\Delta}$. Entonces M es un conjunto de determinación para S si dado un spline $s \in S$, si todos sus B-coeficientes c_ξ son 0 para todo $\xi \in M$, entonces $s \equiv 0$.

Definición 3.4.2. Llamamos conjunto minimal de determinación para S, a un conjunto de determinación M tal que no haya ninguno más pequeño que él.

Teorema 3.4.3. *Supongamos que S es un subespacio vectorial de $S_d^0(\Delta)$ de dimensión m y M es un conjunto de determinación para S. Entonces,*

$$m \leq \#M.$$

Además, si M es un conjunto de determinación para S con $\#M = m$, entonces M es un conjunto de determinación mínimal.

Demostración. Vamos a definir en $S_d^0(\Delta)$ para cada $\xi \in D_{d,\Delta}$, un funcional lineal γ_ξ, tal que $\gamma_\xi s$ es el B-coeficiente de s asociado al punto dominante ξ.

Vamos a razonar por reducción al absurdo. Sea $B_1, ..., B_m$ una base de S, supondremos que M es un conjunto de determinación para S con $\#M < m$. Deberá entonces existir más de una solución del sistema homogéneo

$$\sum_{j=1}^{m} a_j \gamma_\xi B_j = 0, \quad \xi \in M$$

pero esto es imposible ya que $B_1, ..., B_m$ son linealmente independientes, por lo tanto, debe ser $m \leq \#M$.

De aquí deducimos que si el cardinal de M es m, este debeser un conjunto minimal de determinación. □

Dado un conjunto minimal de determinación M de S, suponemos que asignamos valores a los coeficientes $\{c_\xi\}_{\xi \in M}$. Entonces para todo $\alpha \in D_{d,\Delta} \backslash M$, los demás c_α, podemos deducirlos de los coeficientes conocidos c_ξ utilizando las condiciones de regularidad. Diremos que c_α depende de c_ξ, si cuando cambiamos el valor c_ξ, también cambia el valor c_α, y entonces definiremos para estos puntos:

$$P_\alpha := \{\xi \in M : c_\alpha \text{ depende de } c_\xi\}.$$

Siendo M un minimo conjunto de determinación para S, definiremos el siguiente concepto.

Definición 3.4.4. Diremos que M es local si existe un entero ℓ no dependiente de Δ tal que

$$P_\alpha \subseteq estrella^\ell(T_\alpha), \quad \text{para } \alpha \in D_{d,\Delta} \backslash M$$

siendo T_α un triángulo que contiene a α.

3.5. Bases locales

Para los métodos de interpolación que vamos a utilizar, no nos hará falta conocer bases del espacio, pero hay muchos otros para los cuales es necesario el uso de bases. Para un espacio de splines arbitrario S, si tenemos un minimo conjunto de determinación para él, es fácil construir una base.

Sea M un minimo conjunto de determinación para un espacio de splines $S \subseteq S_d^0(\Delta)$ sobre una triangulación Δ. Para cada $\xi \in M$, vamos a considerar el funcional γ_ξ que definimos en la sección anterior y que cumple que para todo $s \in S$, $\gamma_\xi s$ es el B-coeficiente de s correspondiente al punto dominante ξ. Con esta notación, se puede afirmar el siguiente teorema.

Teorema 3.5.1. *Sea M un conjunto minimal de determinación para un espacio de splines $S \subseteq S_d^0(\Delta)$. Entonces, para cada $\xi \in M$, hay un único spline $\psi_\xi \in S$ tal que*

$$\gamma_\alpha \psi_\xi = \delta_{\alpha,\xi}, \quad \text{para } \alpha \in M. \tag{3.4}$$

Además, $\Psi := \{\psi_\xi\}_{\xi \in M}$ es una base del espacio S.

Demostración. El cardinal de M es igual a la dimensión de S, por ser M conjunto minimal de determinación, por lo que se debe cumplir la primera parte del teorema.

Para ver que $\Psi := \{\psi_\xi\}_{\xi \in M}$ es una base, supondremos que

$$s := \sum_{\xi \in M} c_\xi \psi_\xi \equiv 0.$$

Entonces, para todo $\alpha \in M$ tendremos que $c_\alpha = \gamma_\alpha s = 0$, lo que implica que los splines pertenecientes a Ψ son linealmente independientes.

Como ya sabemos que $\#\Psi = \#M$, que a su vez es igual a la dimensión de S, se tiene que Ψ es una base. □

Capítulo 4

Macroelementos

Para ver lo que es un macroelemento, es necesario definir previamente lo que es un conjunto nodal minimal de determinación.

4.1. Conjunto nodal minimal de determinación

Hasta ahora, hemos parametrizado los espacios de splines $S \subseteq S_d^0(\Delta)$ usando la forma de Bernstein-Bézier, es decir, mediante los B-coeficientes asociados a subconjuntos de $D_{d,\Delta}$.

Vamos a parametrizar espacios de splines en terminos de los parámetros nodales, también llamados grados de libertad o condiciones de interpolación. Para ello, vamos a considerar funcionales lineales de la forma:

$$\lambda := \varepsilon_t \sum_{\alpha+\beta=m} a_{\alpha,\beta} D_x^\alpha D_y^\beta, \tag{4.1}$$

donde ε_t denota el operador evaluación en t. A t se le llama **punto soporte del funcional** λ .

A partir de estos funcionales, vamos a dar las siguientes definiciones:

Definición 4.1.1. Supongamos que $\mathcal{N} = \{\lambda_i\}_{i=1}^n$ es un conjunto de funcionales lineales de la forma (4.1). Diremos que $\mathcal{N}$ es un **conjunto nodal de determinación** para S si

$$\lambda s = 0 \quad \text{para todo } \lambda \in \mathcal{N}$$

implica,

$$s \equiv 0.$$

Además, diremos que $\mathcal{N}$ es un **conjunto nodal minimal de determinación** de S si no existe un conjunto nodal de determinación para S con menos elementos.

Teorema 4.1.2. *Un conjunto nodal de determinación $\mathcal{N}$ de S es minimal si y solo si $\#\mathcal{N} = dimS$.*

Demostración. Igual que demostramos el teorema 3.4.3, pero en este caso, asignaremos valores arbitrarios a $\{\lambda_s\}_{\lambda\in N}$, y todos los coeficientes de s serán determinados de forma única. □

Vamos a ver un ejemplo de un conjunto nodal de determinación para $S_2^0(\Delta)$.

Ejemplo 4.1.3. Dada una triangulación Δ, sea V el conjunto de todos los vértices de Δ y E el conjunto de todos los lados de Δ. Para cada lado $e \in E$, definimos m_e el punto medio de e. Entonces $\mathcal{N} := \{\varepsilon_v\}_{v\in V} \cup \{\varepsilon_{m_e}\}_{e\in E}$, es un conjunto nodal minimal de determinación para $S_2^0(\Delta)$.

Ahora supongamos que tenemos $s \in S_2^0(\Delta)$ y nos dan los valores de $\{s(v)\}_{v\in V}$ y $\{s(m_e)\}_{e\in E}$. Para cada $v \in V$, el valor $s(v)$ determina únicamente el B-coeficiente asociado al punto dominante en v. Además, para cada lado $e := \langle v_1, v_2\rangle$, el B-coeficiente asociado al punto dominante $\xi = m_e$ es fácil ver que es $c_\xi = 4s(m_e) - s(v_1) - s(v_2)$.

□

Los funcionales de un conjunto nodal minimal de determinación $\mathcal{N} = \{\lambda_i\}_{i=1}^n$ definen un problema de interpolación en S unisolvente: existe un único $s \in S$ tal que dados valores arbitrarios z_i, $i = 1, ..., n$, cumple que $\lambda_i s = z_i$, $i = 1, ..., n$.

Si $\mathcal{N} = \{\lambda_i\}_{i=1}^n$ es un conjunto nodal minimal de determinación para un espacio de splines S, para cada $1 \leq i \leq n$, existe un único spline $\phi_i \in S$ tal que

$$\lambda_j \phi_i = \delta_{ij}, \quad j = 1, ..., n.$$

Se determina así una base de S que premite expresar explícitamente el interpolante en S en términos de los datos de interpolación:

$$s(x) = \sum_{i=1}^{n} z_i \phi_i(x).$$

A la base de splines $\Phi = \{\phi_i\}_{i=1}^n$ la llamaremos N-base de S.

Por ejemplo, esta base permite dar una solución formal a ciertos problemas de interpolación de Hermite. Supongamos que f es tantas veces diferenciable como para que los valores de $\lambda_i f$ se puedan definir. Definamos

$$Hf := \sum_{i=1}^{n} (\lambda_i f)\phi_i.$$

Entonces, se tiene que $s := Hf$ satisface,

$$\lambda_j s = \lambda_j f, \quad j = 1, ..., n. \tag{4.2}$$

Entonces, el operador interpolante de Hermite H cumple que,

$$Hs = s, \quad \text{para todo } s \in S.$$

Definición 4.1.4. Supongamos que $\mathcal{N}$ es un conjunto nodal minimal de determinación de un espacio de splines $S \subseteq S_d^0(\Delta)$ y supongamos que el orden de la derivada más alta involucrada en los funcionales lineales en $\mathcal{N}$ es $\bar{m}$. Diremos que $\mathcal{N}$ es **local** si cumple que existe un entero ℓ que no dependa de Δ tal que para todo $s \in S$, $T \in \Delta$ y $\xi \in D_T$, el coeficiente c_ξ de s se puede obtener de los datos nodales en los puntos de $\Omega_T := \text{estrella}^\ell(T)$. Diremos que $\mathcal{N}$ es **estable** si cumple que existe una constante K que solo depende de ℓ y del menor ángulo en Δ tal que para todo $s \in S$, $T \in \Delta$ y $\xi \in D_{d,T}$,

$$|c_\xi| \leq K \sum_{j=0}^{\bar{m}} |T|^j |s|_{j,\Omega_T},$$

donde $|T|$ denota el diámetro de T y $|s|_{j,\Omega_T} = max_{\alpha+\beta=j} max_{x\in\Omega_T} |D_x^\alpha D_y^\beta s(x)|$.

A partir de estas definiciones, daremos una cota de error para el problema de interpolación de Hermite en un espacio de splines S con un conjunto nodal minimal de determinación local y estable, que más tarde comprobaremos en la práctica.

Teorema 4.1.5. *[2] Supongamos que $\mathcal{N}$ es un conjunto nodal minimal de determinación de un espacio de splines S, y sea l y $\bar{m}$ las constantes definidas en la Definición 4.1.4. Sea H el operador de Hermite definido también en la Definición 4.1.4. Dado un triángulo T en Δ, sea $\Omega_T := \text{estrella}^\ell(T)$. Entonces, para todo $f \in C^{m+1}(\Omega_T)$ con $\bar{m} \leq m \leq d$,*

$$\|D_x^\alpha D_y^\beta (f - Hf)\|_T \leq K|T|^{m+1-\alpha-\beta} |f|_{m+1,\Omega_T}, \tag{4.3}$$

para todo $0 \leq \alpha + \beta \leq m$. Si Ω_T es convexo, K solo depende de d, l y del menor ángulo de los triángulos de Ω_T. Si no es convexo, K también depende de la constante de Lipschitz de la frontera de Ω_T.

4.2. Definición de macroelementos

En este capítulo vamos a ver como podemos resolver el problema de interpolación de Hermite (4.2) ($\lambda_j s = \lambda_j f$), sin construir ninguna base.

Vamos a estar interesados en trabajar con espacios de splines definidos en triangulaciones Δ_R obtenidas aplicando un refinamiento a todos los triángulos T de una triangulación dada Δ. En concreto, estaremos interesados en los casos en los que Δ_R es obtenida de Δ tras aplicar refinamientos tipo Powell-Sabin, Clough-Tocher o Wang.

Sea $\mathcal{N}$ un conjunto nodal minimal de determinación para un espacio de splines $S \subseteq S_d^0(\Delta_R)$. Para cada triángulo $T \in \Delta$, tenemos,

$$\mathcal{N}_T := \{\lambda \in \mathcal{N} : \text{ el soporte } t \text{ de } \lambda \text{ está contenido en } T\}$$

Definición 4.2.1. Diremos que S es un espacio de macroelementos si existe un conjunto nodal minimal de determinación $\mathcal{N}$ para S tal que para todo triángulo $T \in \Delta$, $s|_T$ está determinado de forma única por los valores $\{\lambda_S\}_{\lambda \in \mathcal{N}_T}$.

Si S es un espacio de macroelementos con conjunto nodal de determinación $\mathcal{N}$, el spline interpolante de Hermite s que hemos definido anteriormente puede ser computado triángulo a triángulo, y además $s|_T$ se puede computar con los datos de interpolación en los puntos en T.

En este capítulo vamos a ver ejemplos de creación de estos splines.

4.3. Argyris

En esta sección vamos a discutir un nuevo interpolante para el cual no nos va a ser necesario considerar ningún refinamiento sobre la triangulación. Sea Δ una triangulación en un dominio Ω y sea V su conjunto de vértices, vamos a ver el espacio de supersplines:

$$S_5^{1,2}(\Delta) := \{s \in S_5^1(\Delta) : s \in C^2(v) \text{ para todo } v \in V\}$$

Para cada lado e de Δ, sea n_e un vector unitario perpendicular al lado e, y m_e será el punto medio del lado e. Si consideramos E el conjunto de lados de Δ, tendremos el siguiente teorema.

Teorema 4.3.1. *Sea Δ una triangulación con vértices $\{(x_i, y_i)\}_{i=1}^n$. Supongamos que tenemos un conjunto de números reales $\{z_i^{\nu\mu}\}_{0 \le \nu+\mu \le 2}$ para*

$i = 1, ..., n$, y $\{z_e\}_{e \in E}$. Entonces existe un único spline $s \in S_5^{1,2}(\Delta)$ que satisface:

$$D_x^\nu D_y^\mu s(x_i, y_i) = z_i^{\nu\mu}, \quad 0 \leq \nu + \mu \leq 2, i = 1, ..., n, \tag{4.4}$$

y

$$D_{n_e} s(m_e) = z_e, \quad \textit{para todo } e \in E. \tag{4.5}$$

Es posible computar los coeficientes que van a definir este spline interpolante en cada triángulo. Supongamos que $T := \langle v_1, v_2, v_3 \rangle$ es un triángulo en Δ, y $\{c_{ijk}^T\}_{i+j+k=5}$ serán los correspondientes B-coeficientes de $s|_T$.

$$\begin{array}{ccccccccccc}
 & & & & c_{500}^T & & & & & & \\
 & & & & c_{410}^T & & c_{401}^T & & & & \\
 & & & c_{320}^T & & c_{311}^T & & c_{302}^T & & & \\
 & & c_{230}^T & & c_{211}^T & & c_{212}^T & & c_{203}^T & & \\
 & c_{140}^T & & c_{131}^T & & c_{122}^T & & c_{113}^T & & c_{104}^T & \\
c_{050}^T & & c_{041}^T & & c_{032}^T & & c_{023}^T & & c_{014}^T & & c_{005}^T
\end{array}$$

Figura 4.1: B-coeficientes de un polinomio quíntico

Vamos a comenzar calculando los coeficientes correspondientes a los puntos dominantes localizados en los discos de radio 2 asociados a cada vértice. Para calcular los coeficientes asociados a los puntos dominantes del disco $D_2^T(v_i)$, siendo $i = 1, 2, 3$, vimos en el teorema 1.7.2 que los podemos obtener a partir de los valores de la derivada $D_x^\nu D_y^\mu s(x_i, y_i)$, es decir, $z_i^{\nu\mu}$, que también los conocemos. Para calcular dichos coeficientes, solo tendremos que sustituir nuestros valores en la fórmula (1.12). Los coeficientes de los puntos dominantes asociados a los vértices, serán el valor del spline en dicho vértice, es decir, $c_{500}^T = z_1, c_{050}^T = z_2$ y $c_{005}^T = z_3$. De ahí obtendremos las siguientes expresiones para el disco de v_1:

$$\begin{aligned}
c_{500}^T &= z_1, \\
c_{410}^T &= [h_2 z_1^{10} + \tilde{h}_2 z_1^{01}]/5 + z_1, \\
c_{401}^T &= [h_3 z_1^{10} + \tilde{h}_3 z_1^{01}]/5 + z_1, \\
c_{320}^T &= [h_2^2 z_1^{20} + 2h_2\tilde{h}_2 z_1^{11} + \tilde{h}_2^2 z_1^{02}]/20 + 2c_{410}^T - z_1, \\
c_{311}^T &= [h_2 h_3 z_1^{20} + (h_2\tilde{h}_3 + h_3\tilde{h}_2) z_1^{11} + \tilde{h}_2\tilde{h}_3 z_1^{02}]/20 + c_{401}^T + c_{410}^T - z_1, \\
c_{302}^T &= [h_3^2 z_1^{20} + 2h_3\tilde{h}_3 z_1^{11} + \tilde{h}_3^2 z_1^{02}]/20 + 2c_{401}^T - z_1,
\end{aligned}$$

donde hemos considerado $h_i := x_i - x_1$ y $\tilde{h}_i := y_i - y_1$ para $i = 2, 3$.

Si nos fijamos en la Figura 4.1, hemos definido los B-coeficientes de $s|_T$ que están en el disco de radio 2 alrededor de v_1. Estas mismas fórmulas para v_2 y v_3 definirán los coeficientes de los discos alrededor de ellos.

Nos faltarían los siguientes coeficientes:

$$c^T_{211} \qquad c^T_{212}$$
$$c^T_{122}$$

Figura 4.2

Para definir los que nos faltan procederemos de la siguiente manera. Consideramos el lado $e := \langle v_2, v_3 \rangle$. Utilizando el dato que tenemos de la derivada en el punto medio de e, z_e, y considerando (a_1, a_2, a_3) las coordenadas direccionales con respecto a T del vector unitario perpendicular a e con dirección hacia el interior del triángulo. Conociendo los coeficientes calculados anteriormente, estamos en condiciones de aplicar el Lema 1.5.6, y resolviendo el sistema (1.7) tendremos determinados de forma única los coeficientes c_{122}, c_{221} y c_{212}.

$$\begin{aligned} c^T_{122} =& \frac{16}{30a_1} z_e - \frac{1}{6}[c^T_{140} + 4c^T_{131} + 4c^T_{113} + c^T_{104}] \\ & - \frac{a_2}{6a_1}[c^T_{050} + 4c^T_{041} + 6c^T_{032} + 4c^T_{023} + c^T_{014}] \\ & - \frac{a_3}{6a_1}[c^T_{041} + 4c^T_{032} + 6c^T_{023} + 4c^T_{014} + c^T_{005}]. \end{aligned}$$

Procedemos de igual manera para los dos coeficientes que quedan. Para calcular c^T_{221}, vamos a considerar el lado $e := \langle v_1, v_2 \rangle$, y ahora (a_1, a_2, a_3) son las coordenadas direccionales con respecto a T del nuevo vector unitario perpendicular a e con dirección hacia el interior del triángulo, tenemos,

$$\begin{aligned} c^T_{221} =& \frac{16}{30a_1} z_e - \frac{1}{6}[c^T_{401} + 4c^T_{311} + 4c^T_{131} + c^T_{041}] \\ & - \frac{a_2}{6a_1}[c^T_{500} + 4c^T_{410} + 6c^T_{320} + 4c^T_{230} + c^T_{140}] \\ & - \frac{a_3}{6a_1}[c^T_{410} + 4c^T_{320} + 6c^T_{230} + 4c^T_{140} + c^T_{050}], \end{aligned}$$

y con la misma notación pero ahora $e := \langle v_3, v_1 \rangle$,

$$\begin{aligned} c^T_{212} =& \frac{16}{30a_1} z_e - \frac{1}{6}[c^T_{014} + 4c^T_{113} + 4c^T_{311} + c^T_{410}] \\ & - \frac{a_2}{6a_1}[c^T_{005} + 4c^T_{104} + 6c^T_{203} + 4c^T_{302} + c^T_{401}] \\ & - \frac{a_3}{6a_1}[c^T_{104} + 4c^T_{203} + 6c^T_{302} + 4c^T_{401} + c^T_{500}]. \end{aligned}$$

Ya tendríamos calculados todos los coeficientes de forma única para garantizar que se cumplan las condiciones de regularidad.

Por tanto, para toda función $f \in C^2(\Omega)$, hay un único spline $s \in S^{1,2}_5(\Delta)$ que resuelve el problema de interpolación de Hermite

$$\begin{aligned} D^\alpha_x D^\beta_y s(v) &= D^\alpha_x D^\beta_y f(v), \quad \text{para todo } v \in V, 0 \leq \alpha + \beta \leq 2 \\ D_{m_e} s(n_e) &= D_{m_e} f(n_e), \quad \text{para todo } e \in E. \end{aligned}$$

Esto define una aplicación lineal $\mathcal{I}^1_P$ entre $C^2(\Omega)$ y el espacio de supersplines $S^{1,2}_5(\Delta)$, el cuál reproduce polinomios de grado 5.

Estamos en condiciones de aplicar el Teorema 4.1.5, que más tarde comprobaremos que se cumple en la práctica.

Teorema 4.3.2. *[2] Para todo $f \in C^{m+1}(\Omega)$ con $1 \leq m \leq 5$,*

$$\|D^\alpha_x D^\beta_y (f - \mathcal{I}^1_P f)\|_\Omega \leq K|T|^{m+1-\alpha-\beta}|f|_{m+1,\Omega},$$

para todo $0 \leq \alpha + \beta \leq m$. Si Ω es convexo, la constante K depende solo del menor ángulo de Δ. Si Ω no es convexo, K también depende la la constante de Lipschitz de la frontera de Ω.

4.4. Clough-Tocher

Dada una triangulación Δ en un dominio Ω, Δ_{CT} será su refinamiento de Clough-Tocher, descrito tal y como vimos en la subsección 2.5.1. A los triángulos originales de Δ vamos a llamarlos macro-triángulos, y a los de Δ_{CT} que han surgido tras realizar el refinamiento como micro-triángulos.

Llamaremos V al conjunto de vértices de Δ y L será el conjunto de lados de Δ. Para cada lado $e := \langle u, v \rangle \in L$, sea $m_e := (u+v)/2$ el punto medio de e y sea D_{n_e} la derivada direccional asociada al vector correspondiente a rotar e 90 grados en sentido contrario a las agujas del reloj. La llamaremos derivada cruzada. Vamos a considerar el espacio de macroelementos: $S^1_3(\Delta_{CT})$.

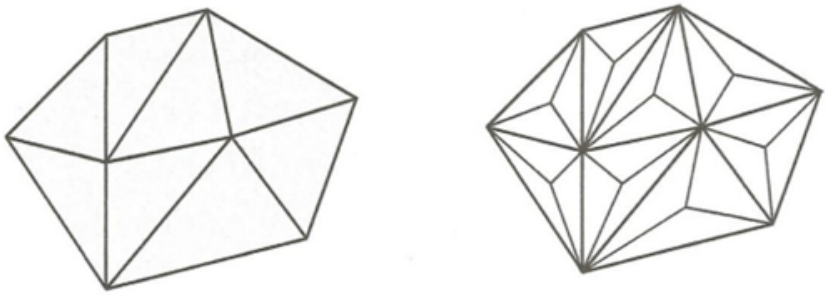


Figura 4.3: Triangulación y su refinamiento de Clough-Tocher

Teorema 4.4.1. *Dada una triangulación Δ con vértices $\{(x_i, y_i)\}_{i=1}^n$ y un conjunto de números reales $\{z_i, z_i^x, z_i^y\}_{i=1}^n$ y $\{z_e\}_{e\in L}$, existe un único spline $s \in S_3^1(\Delta_{CT})$ que cumple*

$$\begin{aligned} &s(x_i, y_i) = z_i, \\ &D_x s(x_i, y_i) = z_i^x, \qquad i = 1, ..., n. \\ &D_y s(x_i, y_i) = z_i^y, \\ &D_{n_e} s(m_e) = z_e, \end{aligned} \tag{4.6}$$

para todos los vértices y lados de Δ.

Vamos a ver cómo va a ser este único spline s. Supongamos que $T := \langle v_1, v_2, v_3\rangle$ es uno de los macro-triángulos de Δ que ha sido dividido en tres triángulos para formar Δ_{CT}. La restricción $s|_T$ va a ser un spline cúbico que consiste en la unión de tres polinomios cúbicos con regularidad C^1. Los B-coeficientes para cada triángulo serán los siguientes:

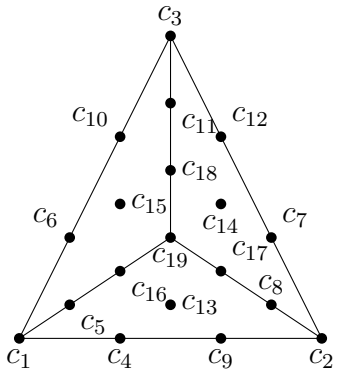


Figura 4.4: B-coeficientes en un triángulo de la triangulación.

El superspline está definido por esos 19 B-coeficientes en su restricción $s|_T$. Los B-coeficientes asociados a los puntos dominantes de los vértices de cada macrotriángulo, corresponderán con el valor z_i del spline en cada vértice, que es conocido. Para calcular los coeficientes asociados a los puntos dominantes del disco $D_1^T(v_i)$, siendo $i = 1, 2, 3$, vimos en el teorema 1.7.2 que los podemos obtener a partir de los valores de la derivada $D_u s(x_i, y_i)$, es decir, z_i^x y z_i^y, que también los conocemos. Para calcular dichos coeficientes, solo tendremos que sustituir nuestros valores en la fórmula (1.12) y de ahí obtendremos las siguientes expresiones:

$$\begin{aligned}
c_1 &= z_1, \quad c_2 = z_2, \quad c_3 = z_3, \\
c_4 &= [(x_2 - x_1)z_1^x + (y_2 - y_1)z_1^y]/3 + z_1 \\
c_5 &= [(x_c - x_1)z_1^x + (y_c - y_1)z_1^y]/3 + z_1 \\
c_6 &= [(x_3 - x_1)z_1^x + (y_3 - y_1)z_1^y]/3 + z_1 \\
c_7 &= [(x_3 - x_2)z_2^x + (y_3 - y_2)z_2^y]/3 + z_2 \\
c_8 &= [(x_c - x_2)z_2^x + (y_c - y_2)z_2^y]/3 + z_2 \\
c_9 &= [(x_1 - x_2)z_2^x + (y_1 - y_2)z_2^y]/3 + z_2 \\
c_{10} &= [(x_1 - x_3)z_3^x + (y_1 - y_3)z_3^y]/3 + z_3 \\
c_{11} &= [(x_c - x_3)z_3^x + (y_c - y_3)z_3^y]/3 + z_3 \\
c_{12} &= [(x_2 - x_3)z_3^x + (y_2 - y_3)z_3^y]/3 + z_3.
\end{aligned}$$

Consideramos el lado $e := \langle v_1, v_2 \rangle$ y suponemos que (a_1, a_2, a_3) son las coordenadas direccionales de n_e con respecto a T. Conociendo los coeficientes calculados anteriormente, estamos en condiciones de aplicar el Lema 1.5.6, y resolviendo el sistema (1.7) tendremos determinados de forma única los coeficientes c_{13}, c_{14} y c_{15}. Entonces tenemos que:

$$c_{13} = \frac{4}{6a_3} z_e - \frac{1}{2}(c_5 + c_8) - \frac{a_1}{2a_3}(c_1 + 2c_4 + c_9) - \frac{a_2}{2a_3}(c_4 + 2c_9 + c_2).$$

Ahora vamos a considerar el lado $e := \langle v_2, v_3 \rangle$ y suponemos que (a_4, a_5, a_6) son las coordenadas direccionales de n_e con respecto a T Entonces tenemos que:

$$c_{14} = \frac{4}{6a_6} z_e - \frac{1}{2}(c_8 + c_{11}) - \frac{a_4}{2a_6}(c_2 + 2c_7 + c_{12}) - \frac{a_5}{2a_6}(c_7 + 2c_{12} + c_3).$$

De igual manera con el lado $e := \langle v_3, v_1 \rangle$ y suponemos que (a_7, a_8, a_9) son las coordenadas direccionales de n_e con respecto a T Entonces tenemos que:

$$c_{15} = \frac{4}{6a_9} z_e - \frac{1}{2}(c_{11} + c_5) - \frac{a_7}{2a_9}(c_3 + 2c_{10} + c_6) - \frac{a_8}{2a_9}(c_{10} + 2c_6 + c_1).$$

Los coeficientes restantes de $s|_T$ serán calculados para cumplir las condiciones de regularidad pedidas según vimos en el Teorema 3.3.4 para que se cumpla (3.3):

$$\begin{aligned} c_{16} &= (c_{15} + c_5 + c_{13})/3, \\ c_{17} &= (c_{13} + c_8 + c_{14})/3, \\ c_{18} &= (c_{14} + c_{11} + c_{15})/3, \\ c_{19} &= (c_{16} + c_{17} + c_{18})/3. \end{aligned}$$

Ya tendríamos definidos los 19 B-coeficientes que determinan $s|_T$. Construido de esta manera, hemos visto que es la forma única que cumple las condiciones pedidas en el teorema de existencia y unicidad.

Por tanto, para toda función $f \in C^1(\Omega)$, hay un único spline $s \in S_3^1(\Delta_{CT})$ que resuelve el problema de interpolación de Hermite

$$\begin{aligned} D_x^\alpha D_y^\beta s(v) &= D_x^\alpha D_y^\beta f(v), \quad \text{para todo } v \in V, 0 \leq \alpha + \beta \leq 1 \\ D_{m_e} s(n_e) &= D_{m_e} f(n_e), \quad \text{para todo } e \in E. \end{aligned}$$

Esto define una aplicación lineal $\mathcal{I}_{CT}^1$ entre $C^1(\Omega)$ y el espacio de supersplines $S_3^1(\Delta_{CT})$, la cuál reproduce polinomios de grado 3.

Estamos en condiciones de aplicar el Teorema 4.1.5, que más tarde comprobaremos que se cumple en la práctica.

Teorema 4.4.2. *[2] Para todo $f \in C^{m+1}(\Omega)$ con $1 \leq m \leq 3$,*

$$\|D_x^\alpha D_y^\beta (f - \mathcal{I}_{CT}^1 f)\|_\Omega \leq K|T|^{m+1-\alpha-\beta}|f|_{m+1,\Omega},$$

para todo $0 \leq \alpha + \beta \leq m$. Si Ω es convexo, la constante K depende solo del menor ángulo de Δ. Si Ω no es convexo, K también depende la la constante de Lipschitz de la frontera de Ω.

4.5. Powell-Sabin

Dada una triangulación Δ en un dominio Ω, Δ_{PS} será su refinamiento de Powell-Sabin, siguiendo los pasos que vimos en la subsección 2.5.2.

Cada triángulo de Δ (macro-triángulo) es dividido en 6 subtriángulos (micro-triángulos). Utilizando la misma notación que en el apartado anterior, en este caso vamos a trabajar en el espacio de macroelementos $S_2^1(\Delta_{PS})$.

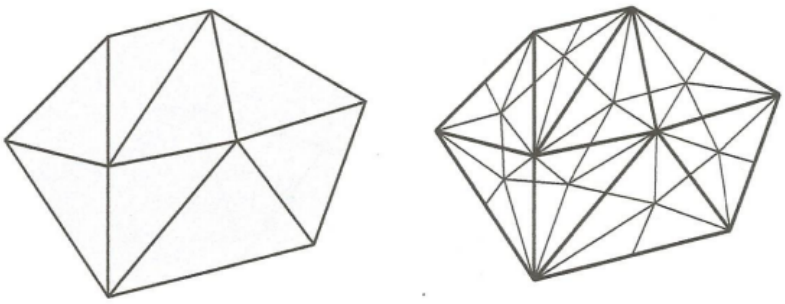


Figura 4.5: Triangulación y su refinamiento de Powell-Sabin

Teorema 4.5.1. *Dada una triangulación* Δ *con vértices* $\{(x_i, y_i)\}_{i=1}^n$ *y un conjunto de números reales* $\{z_i, z_i^x, z_i^y\}_{i=1}^n$*, existe un único spline* $s \in S_2^1(\Delta_{PS})$ *que cumple*

$$\begin{aligned} &s(x_i, y_i) = z_i, \\ &D_x s(x_i, y_i) = z_i^x, \qquad i = 1, ..., n. \\ &D_y s(x_i, y_i) = z_i^y, \end{aligned} \tag{4.7}$$

para todos los vértices de Δ*.*

Vamos a ver cómo podemos computar el spline descrito en el teorema anterior. Supongamos que $T := \langle v_1, v_2, v_3 \rangle$ es uno de los macro-triángulos de Δ que ha sido dividido en 6 subtriángulos para formar Δ_{PS}. Ahora consideramos la restricción $s|_T$ al macro-triángulo T. Es un spline cuadrático que consiste en 6 polinomios cuadráticos unidos entre ellos con regularidad C^1. Este subspline está determinado por 19 coeficientes donde c_1, c_2, c_3 están asociados con los vértices v_1, v_2, v_3, respectivamente. Vamos a dar fórmulas explicitas para estos coeficientes en términos de los datos que tenemos sobre los tres vértices.

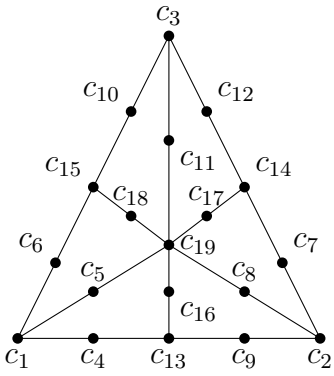


Figura 4.6: B-coeficientes en un triángulo de la triangulación.

Vamos a llamar $u_T = (x_c, y_c)$ al incentro de T, y suponemos que las coordenadas baricéntricas de u_T con respecto de T son (a_1, a_2, a_3). Sean los lados de T, $e_1 := \langle v_1, v_2\rangle$, $e_2 := \langle v_2, v_3\rangle$, y $e_3 := \langle v_3, v_1\rangle$. Para cada $i = 1, 2, 3$, vamos a considerar el vértice de Δ_{PS} en el lado e_i como $w_i := (\hat{x}_i, \hat{y}_i) = r_i v_i + s_i v_{i+1}$, considerando $v_4 = v_1$. Los B-coeficientes estarán determinados al igual que vimos en el caso de Clough-Tocher. Para calcular los coeficientes asociados a los puntos dominantes del disco $D_1^T(v_i)$, siendo $i = 1, 2, 3$, vimos en el Teorema 1.7.2 que los podemos obtener a partir de los valores de la derivada $D_u s(x_i, y_i)$, es decir, z_i^x y z_i^y, que también los conocemos. Para calcular dichos coeficientes, solo tendremos que sustituir nuestros valores en la fórmula (1.12). De igual manera que antes, los coeficientes de los puntos dominantes asociados a los vértices, será el valor del spline en dicho vértice, es decir, $c_i = z_i$ para $i = 1, 2, 3$. De ahí obtendremos las siguientes expresiones:

$$
\begin{aligned}
c_1 = z_1, \quad c_2 = z_2, \quad c_3 = z_3,\\
c_4 &= [(\hat{x}_1 - x_1)z_1^x + (\hat{y}_1 - y_1)z_1^y]/2 + z_1\\
c_5 &= [(x_c - x_1)z_1^x + (y_c - y_1)z_1^y]/2 + z_1\\
c_6 &= [(\hat{x}_3 - x_1)z_1^x + (\hat{y}_3 - y_1)z_1^y]/2 + z_1\\
c_7 &= [(\hat{x}_2 - x_2)z_2^x + (\hat{y}_2 - y_2)z_2^y]/2 + z_2\\
c_8 &= [(x_c - x_2)z_2^x + (y_c - y_2)z_2^y]/2 + z_2\\
c_9 &= [(\hat{x}_1 - x_2)z_2^x + (\hat{y}_1 - y_2)z_2^y]/2 + z_2\\
c_{10} &= [(\hat{x}_3 - x_3)z_3^x + (\hat{y}_3 - y_3)z_3^y]/2 + z_3\\
c_{11} &= [(x_c - x_3)z_3^x + (y_c - y_3)z_3^y]/2 + z_3\\
c_{12} &= [(\hat{x}_2 - x_3)z_3^x + (\hat{y}_2 - y_3)z_3^y]/2 + z_3
\end{aligned}
$$

Los restantes coeficientes de $s|_T$ pueden definirse de la siguiente manera. Serán calculados para cumplir las condiciones de regularidad pedidas según vimos en el Teorema 3.3.4 para que se cumpla (3.3)

$$
\begin{aligned}
c_{13} &= r_1 c_4 + s_1 c_9,\\
c_{14} &= r_2 c_7 + s_2 c_{12},\\
c_{15} &= r_3 c_{10} + s_3 c_6,\\
c_{16} &= r_1 c_5 + s_1 c_8,\\
c_{17} &= r_2 c_8 + s_2 c_{11},\\
c_{18} &= r_3 c_{11} + s_3 c_5,
\end{aligned}
$$

y por último,

$$c_{19} = a_1 c_5 + a_2 c_8 + a_3 c_{11}.$$

Ya tendríamos definidos los únicos 19 B-coeficientes que determinan $s|_T$ y cumplen las ccondiciones del teorema de existencia y unicidad.

Por tanto, para toda función $f \in C^1(\Omega)$, hay un único spline $s \in S_2^1(\Delta_{PS})$ que resuelve el problema de interpolación de Hermite

$$D_x^\alpha D_y^\beta s(v) = D_x^\alpha D_y^\beta f(v), \quad \text{para todo } v \in V, 0 \leq \alpha + \beta \leq 1.$$

Esto define una aplicación lineal $\mathcal{I}_{PS}^1$ entre $C^1(\Omega)$ y el espacio de supersplines $S_2^1(\Delta_{PS})$, la cuál reproduce polinomios de grado 2.

Estamos en condiciones de aplicar el Teorema 4.1.5, que más tarde comprobaremos que se cumple en la práctica.

Teorema 4.5.2. *[2] Para todo $f \in C^{m+1}(\Omega)$ con $1 \leq m \leq 2$,*

$$\|D_x^\alpha D_y^\beta (f - \mathcal{I}_{PS}^1 f)\|_\Omega \leq K|T|^{m+1-\alpha-\beta}|f|_{m+1,\Omega},$$

para todo $0 \leq \alpha + \beta \leq m$. Si Ω es convexo, la constante K depende solo del menor ángulo de Δ. Si Ω no es convexo, K también depende la la constante de Lipschitz de la frontera de Ω.

4.6. Powell-Sabin 12

En este caso, vamos a ver otro método para encontrar un spline cuadrático con regularidad C^1. Ahora, dada una triangulación Δ, Δ_{PS12} será su refinamiento de Powell-Sabin12, siguiendo los pasos que vimos en la subsección 2.5.3. Cada triángulo de Δ (macro-triángulo), es dividido en 12 subtriángulos (micro-triángulos). La división en 12 triángulos parece una complicación innecesaria, pero da muchas ventajas con respecto del refinamiento Powell-Sabin.

En este caso vamos a trabahar sobre el espacio de macroelementos $\tilde{S}_2^1(\Delta_{PS12})$. El espacio $S_2^1(\Delta_{PS12})$ tiene dimensión $3n + n_e$, siendo n y n_e el número de vértices y lados de Δ respectivamente. Vamos a proporcionar una derivada en los puntos medios de cada lado de Δ. Obtendremos el siguiente subespacio más pequeño.

$$\tilde{S}_2^1(\Delta_{PS12}) := \{s \in S_2^1(\Delta_{PS}) : D_{n_e} s|_e \in P_1^1 \text{ para cada lado } e \text{ de } \Delta\},$$

donde D_{n_e} es la derivada direccional asociada al vector unitario n_e correspondiente con rotar e 90 grados en sentido antihorario. La dimensión de este nuevo espacio es $3n$.

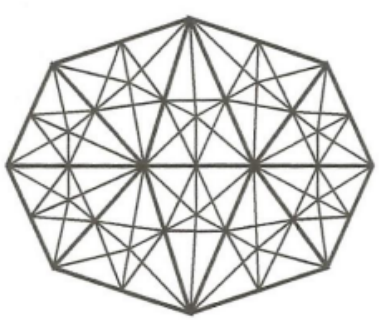


Figura 4.7: Refinamiento de Powell-Sabin12 de una triangulación

Teorema 4.6.1. *Dada una triangulación Δ con vértices $\{(x_i, y_i)\}_{i=1}^n$ y un conjunto de números reales $\{z_i, z_i^x, z_i^y\}_{i=1}^n$, existe un único spline $s \in \tilde{S}_2^1(\Delta_{PS12})$ que cumple*

$$\begin{aligned} &s(x_i, y_i) = z_i, \\ &D_x s(x_i, y_i) = z_i^x, \qquad i = 1, ..., n. \\ &D_y s(x_i, y_i) = z_i^y, \end{aligned} \tag{4.8}$$

para todos los vértices de Δ.

Para computar este interpolante, supongamos que $T := \langle v_1, v_2, v_3 \rangle$ es uno de los triángulos de Δ (macro-triángulo) que ha sido dividido en 12 triángulos (micro-triángulos) para formar Δ_{PS12}. Ahora consideraremos la restricción $s|_T$ al macro triángulo T. Será un spline cuadrático formado por 12 polinomios cuadráticos unidos con regularidad C^1. Este subspline está determinado por 28 coeficientes.

4.7. Wang

Por último, dada una triangulación Δ, vamos a considerar su refinamiendo de Wang, Δ_W, que vimos en la subsección 2.5.4. Cada triángulo de Δ (macro-triángulo), es dividido en 7 subtriángulos (micro-triángulos).

Vamos a hacer uso del espacio de macro-elementos C^2 siguiente,

$$S_2(\Delta_W) := \{s \in S_5^2(\Delta_W) : s \text{ es } C^3 \text{ a lo largo de los lados de } T \in \Delta\}$$

Es un espacio de splines de grado 5. Para cada lado $e := \langle u, v \rangle$ de Δ, n_e va a ser un vector unitario perpendicular al lado e. Si definimos m_e como el punto medio de cada lado e, y $m_e^1 := (2u + v)/3$ y $m_e^2 := (u + 2v)/3$, tendremos el siguiente teorema.

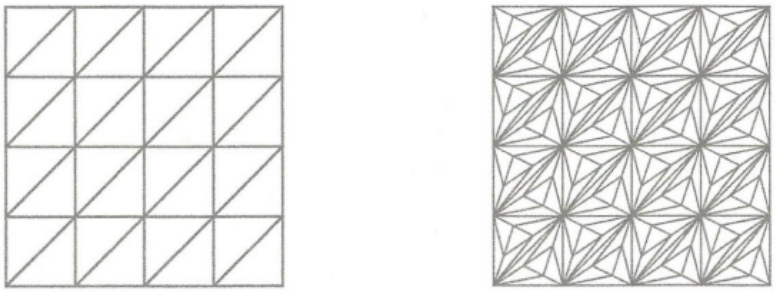


Figura 4.8: Triangulación y su refinamiento de Wang.

Teorema 4.7.1. *Sea Δ una triangulación con vértices $\{(x_i, y_i)\}_{i=1}^n$, y suponemos que tenemos un conjunto de números reales $\{z_i^{\nu\mu}\}_{0\leq\nu+\mu\leq 2}$ para $i = 1, ..., n$, y $\{z_e^1, z_e^2, z_e^3\}_{e\in E}$. Entonces existe un único spline $s \in S_2(\Delta_W)$ que satisface:*

$$D_x^\nu D_y^\mu s(x_i, y_i) = z_i^{\nu\mu}, \quad 0 \leq \nu + \mu \leq 2, i = 1, ..., n. \tag{4.9}$$

y

$$\begin{aligned} &D_{n_e} s(m_e) = z_e^1, \\ &D_{n_e}^2 s(m_e^1) = z_e^2, \quad \textit{para todo } e \in E. \\ &D_{n_e}^2 s(m_e^2) = z_e^3, \end{aligned} \tag{4.10}$$

Como con los otro espacios de macro-elementos vistos en las secciones anteriores, para cada triángulo T de Δ, es posible definir los B-coeficientes del spline interpolante s asociado con el dominio de puntos en T utilizando los datos dados de derivadas en vértices y en los puntos de los lados de T, $\{z_i^{\nu\mu}\}_{0\leq\nu+\mu\leq 2}$ para $i = 1, ..., n$, y $\{z_e^1, z_e^2, z_e^3\}_{e\in E}$. Para cada $s|_T$ necesitaremos un total de 96 B-coeficientes.

En la siguiente figura, veremos los puntos asociados a los B-coeficientes solo pertenecientes a los lados de un triángulo T de la triangulación.

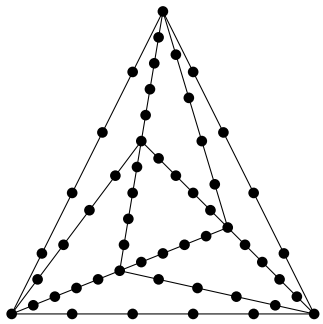


Figura 4.9: B-coeficientes en los lados de un triángulo de la triangulación.

Capítulo 5

Implementación en MATLAB

Vamos a hacer uso de los métodos que hemos descrito en la teoría. Crearemos funciones que nos devuelvan los B-coeficientes de los interpolantes, crearemos programas que evaluen en un punto utilizando estos interpolantes y por último, programas que evaluen el error del interpolante en una malla de puntos. Todo ello para los métodos de Powell-Sabin, Clough-Tocher y Argyris vistos en la teoría. Para las funciones que evaluan estos interpolantes, deberemos pasarle a la función la información donde queremos interpolar, es decir, la triangulación, el punto que queremos evaluar, y una vez que sepamos en qué triángulo de dicha triangulación estamos, los valores correspondientes de la función en los vértices del macro triángulo.

Estas funciones, nos proporcionaran los coeficientes c del polinomio interpolador en su B-forma, con los que, a través de un código en Matlab se aplicará el algoritmo de Casteljau para obtener el valor pedido. Veremos qué herramientas tiene Matlab para ejecutar este proceso y qué valores obtenemos, teniendo en cuenta que cada interpolante tiene una precisión.

Utilizaremos el paquete splinepack [4] de Matlab, del cuál podemos obtener ejemplos de triangulaciones o ejemplos de scripts que interpolan usando los B-coeficientes de la representación Bernstein-Bézier, entre otros recursos. El paquete splinepack de Matlab también tiene funciones propias definidas como es la función $franke2$ que utilizaremos para interpolarla con nuestra implementación de los métodos presentados en esta memoria, como por ejemplo, las funciones que calculan los B-coeficientes de los interpolantes.

Nuestro objetivo es usar en MATLAB los métodos vistos para interpolar en una triangulación.

5.1. Triangulaciones en MATLAB

Primero veremos cómo se define y se representa la triangulación. MATLAB tiene una estructura especifica que veremos a continuación llamada *triangulation*, la cual almacena vértices, lados y triángulos de la triangulación, y las relaciones entre ellos. Para almacenar en MATLAB una triangulación también se puede usar la función de splinepack[4] *readtri*, y para representarla *triplot*, propia de MATLAB.

```
[nv,x,y,nt,TRI]=readtri;
triplot(TRI,x,y);
```

Aquí nv es el número de vértices con coordenadas almacenadas en x e y, nt el número de triángulos y TRI la estructura que almacena las relaciones de la triangulación.

Se podrían añadir otros datos como el área de cada triángulo, los vértices hilados frontera, el número de lados, los índices de lados y vértices, etc. con la función *trilists* de splinepack[4].

```
[nb, ne, nt, v1, v2, v3, el, e2, e3, iel, ie2, tril, trir, bay, vadj,
eadj, adjstart, tadj, tstart, area, TRI] =
trilists (x, y, TRI)
```

En la función MATLAB anterior tendríamos los siguientes datos:
nb = número de vértices exteriores,
nv = número de vértices,
ne = número de lados,
nt = número de triángulos,
bdy = lista de vértices exteriores,
v1, v2, v3 = lista de los índices de los vértices de cada triángulo,
e1, e2, e3 = lista de los índices de los lados de cada triángulo,
area = lista con el área de cada triángulo,
ie1, ie2 = lista de los puntos finales de cada lado,
tril= lista de los índices de los triángulos a la izquierda de cada lado,
trir = lista de los índices de los triángulos a la derecha de cada lado,
vadj = lista de los vértices adyacentes a cada vértice,
eadj = lista de los lados adyacentes a cada vértice,
adjstart = lista de los pointers a las estrellas de los grupos en vadj, eadj,
tadj = lista de los triángulos adyacetes a cada vértice,
tstart = lista de los pointers a las estrellas de los grupos en tadj.

Si queremos guardar una triangulación de la forma más sencilla, podemos utilizar la función propia de MATLAB *triangulation(T,P)*, que tiene como segundo argumento una matriz P de tamaño $n \times 2$ con las coordenadas x e y de todos los vértices de la triangulación, siendo n el número de vértices en la triangulación, y como primer argumento una matriz T $m \times 3$ con la relación de los vértices de los triángulos, siendo m el número de triángulos de la triangulación. Por ejemplo, si el primer tiángulo está formado por los vértices del vector de coordenadas con índices 1, 2, 4 y el segundo por los vértices con índices 5, 3, 6, tendremos como primer argumento de la función el vector [1,2,4;5,6,3].

Si *TR* es una triangulación, tenemos que *TR.points* nos devolverá las coordenadas de los vértices y *TR.ConnectivityList* nos devolverá la lista de relaciones entre los vértices para formar los triángulos. Para representar la triangulación, utilizaremos la función *triplot*.

Ejemplo 5.1.1. Si tenemos una triángulación formada por el triángulo con vértices (0,0), (1,2) y (1/2,2), y por el triángulo (0,0), (1,2) y (2,1), que comparten el lado de (0,0) a (1,2), la definiremos en Matlab como:

```
TR=triangulation([1 2 3; 1 2 4],[0,0;1, 2;1/2,2;2,1]);
triplot(TR);
```

Si lo ejecutamos, obtendremos por pantalla la siguiente representación.

Una vez definida la triángulación, tenemos varias herramientas interesantes que podemos utilizar.

Algunas funciones que utilizaremos son las siguientes. Cuando queramos convertir las coordenadas cartesianas de un punto a coordenadas baricéntricas relativas a un triángulo de una triangulación, utilizaremos la función *cartesianToBarycentric*, en la cual, el primer argumento será la triangulación, el segundo el índice del triángulo respecto al cual queremos las coordenadas y el tercero serán las coordenadas cartesianas del punto. Por ejemplo si queremos expresar el punto (1,1/2) respecto al primer triángulo de la triangulación de la figura anterior escribiremos:

```
ccarte = [1,1/2];
cbari = cartesianToBarycentric(TR,1,[1,1/2]);
```

La variable cbari será [3/4, 7/4, -3/2], correspondiente a las coordenadas baricentricas del punto con respecto al triángulo deseado. Notemos, que al

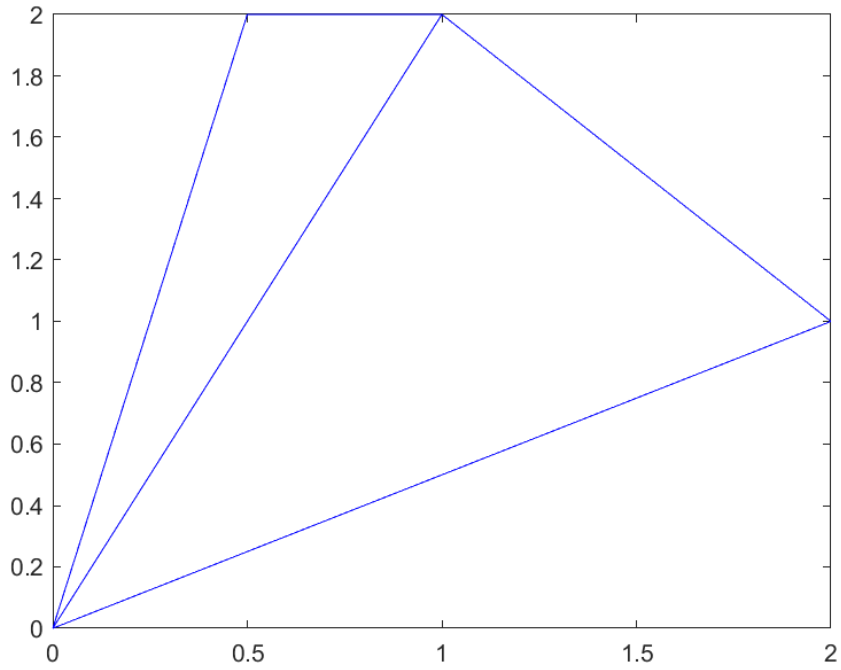


Figura 5.1: Triangulación básica representada por Matlab.

haber una coordenada baricéntrica negativa, se determina que el punto es exterior al triángulo.

La función *barycentricToCartesian* funciona igual, pero convierte las coordenadas baricentricas con respecto a un triángulo, en cartesianas.

Para encontrar en qué triángulo se encuentra un punto en una triangulación, definiremos el siguiente bucle for:

```
for j=1:length(TR(:,1))
    if cartesianToBarycentric(TR,j,C)>=0
        break
    end
end
```

Figura 5.2

Este bucle parará cuando estemos en el triángulo en el cuál todas las coordenadas baricéntricas son no negativas, y por tanto, será donde se encuentra el punto. La variable j corresponderá con el índice de dicho triángulo dentro

de la triangulación definida.

Cuando queramos encontrar el incentro de un triángulo dentro de una triangulación, utilizaremos la función *incent* y como argumentos la pasaremos la triangulación y el índice del triángulo de la triangulación del cuál queramos su incentro.

5.2. Funciones en MATLAB

A partir de estos conocimientos de triangulaciones, se han podido crear funciones interpolantes y scripts que las llaman para evaluar un punto en una triangulación cualquiera con unos datos de interpolación deseados. Las funciones serán para el caso de Powell-Sabin, Clough-Tocher y Argyris. En todos los casos, se llamará a la función metiendo como argumentos los datos de interpolación necesarios, por ejemplo, en el caso de Powell-Sabin, introduciremos cuatro variables: las coordenadas de los vértices, los valores de nuestra función a interpolar en los vértices, los valores de la derivada con respecto de x de la función evaluada en los vértices y los valores de la derivada con respecto de y de la función evaluada en los vértices. En el caso de Clough-Tocher: las coordenadas de los vértices, los valores de nuestra función a interpolar en los vértices, los valores de la derivada con respecto de x de la función evaluada en los vértices, los valores de la derivada con respecto de y de la función evaluada en los vértices y los valores de la derivada direccional con respecto de n_e de la función evaluada en los puntos medios de los lados. En el caso de Argyris: las coordenadas de los vértices, los valores de nuestra función a interpolar en los vértices, los valores de las derivadas hasta orden dos de la función evaluada en los vértices y los valores de la derivada direccional con respecto de n_e de la función evaluada en los puntos medios de los lados. Todas las funciones nos devolverán un vector c con los coeficientes del polinomio interpolador en la base de Bernstein-Bézier.

En los scripts de comprobación, se pedirá que sea introducido por pantalla la triangulación y el punto en los que vamos a interpolar. Una vez encontrado el triángulo concreto de la triangulación al que pertenece el punto, se proporcionaran los vértices correspondientes y se pedirá al usuario que introduzca los demás datos necesarios para llamar a las funciones. En el caso de Clough-Tocher y Argyris, también se verán por pantalla los puntos medios de los lados y el vector n_e que se utiliza para definir uno de los valores de interpolación. En Clough-Tocher y Argyris, nos ayudaremos de las funciones *ct* y *arg15* del paquete splinepack [4] para calcular alguno de los coeficientes.

Una vez tengamos el vector c y las coordenadas baricéntricas del punto a interpolar con respecto al triángulo al que pertenece dentro del refinamiento del triángulo principal, en los casos de Powell-Sabin y Clough-Tocher, y respecto al triángulo principal en el caso de Argyris, procederemos a aplicar el algoritmo de Casteljau para obetener un resultado.

En Powell-Sabin y en Clough-Tocher aplicaremos el algoritmo de Casteljau en 2 y 3 pasos respectivamente (dependiendo del grado y del triángulo del refinamiento al que pertenezca el punto) y se mostrará por pantalla el resultado.

```
%Primero l=1, i+j+k=1
switch j
    case 1
        %PRIMER TRIÁNGULO
        crnuevo100=bari(1)*c(1)+bari(2)*c(4)+bari(3)*c(5);
        crnuevo010=bari(1)*c(4)+bari(2)*c(13)+bari(3)*c(16);
        crnuevo001=bari(1)*c(5)+bari(2)*c(16)+bari(3)*c(19);
    case 2
        %SEGUNDO TRIÁNGULO
        crnuevo100=bari(1)*c(13)+bari(2)*c(9)+bari(3)*c(16);
        crnuevo010=bari(1)*c(9)+bari(2)*c(2)+bari(3)*c(8);
        crnuevo001=bari(1)*c(16)+bari(2)*c(8)+bari(3)*c(19);
    case 3
        %TERCER TRIÁNGULO
        crnuevo100=bari(1)*c(2)+bari(2)*c(7)+bari(3)*c(8);
        crnuevo010=bari(1)*c(7)+bari(2)*c(14)+bari(3)*c(17);
        crnuevo001=bari(1)*c(8)+bari(2)*c(17)+bari(3)*c(19);
    case 4
        %CUARTO TRIÁNGULO
        crnuevo100=bari(1)*c(14)+bari(2)*c(12)+bari(3)*c(17);
        crnuevo010=bari(1)*c(12)+bari(2)*c(3)+bari(3)*c(11);
        crnuevo001=bari(1)*c(17)+bari(2)*c(11)+bari(3)*c(19);
    case 5
        %QUINTO TRIÁNGULO
        crnuevo100=bari(1)*c(3)+bari(2)*c(10)+bari(3)*c(11);
        crnuevo010=bari(1)*c(10)+bari(2)*c(15)+bari(3)*c(18);
        crnuevo001=bari(1)*c(11)+bari(2)*c(18)+bari(3)*c(19);
    case 6
        %SEXTO TRIÁNGULO
        crnuevo100=bari(1)*c(15)+bari(2)*c(6)+bari(3)*c(18);
        crnuevo010=bari(1)*c(6)+bari(2)*c(1)+bari(3)*c(5);
        crnuevo001=bari(1)*c(18)+bari(2)*c(5)+bari(3)*c(19);
end
%Ahora l=2, i+j+k=0 (segundo paso del algoritmo)
res=bari(1)*crnuevo100+bari(2)*crnuevo010+bari(3)*crnuevo001
```

Figura 5.3: Algoritmo de Castelaju para Powell-Sabin.

```
%Primero l=2, i+j+k=2
switch j
    case 1
        %PRIMER TRIÁNGULO
        %Primer paso (i+j+k=2)
        cnuevo200=bari(1)*c(1)+bari(2)*c(4)+bari(3)*c(5);
        cnuevo110=bari(1)*c(4)+bari(2)*c(9)+bari(3)*c(13);
        cnuevo020=bari(1)*c(9)+bari(2)*c(2)+bari(3)*c(8);
        cnuevo011=bari(1)*c(13)+bari(2)*c(8)+bari(3)*c(17);
        cnuevo002=bari(1)*c(16)+bari(2)*c(17)+bari(3)*c(19);
        cnuevo101=bari(1)*c(5)+bari(2)*c(13)+bari(3)*c(16);
        %Segundo paso (i+j+k=1)
        cnuevo100=bari(1)*cnuevo200+bari(2)*cnuevo110+bari(3)*cnuevo101;
        cnuevo010=bari(1)*cnuevo110+bari(2)*cnuevo020+bari(3)*cnuevo011;
        cnuevo001=bari(1)*cnuevo101+bari(2)*cnuevo011+bari(3)*cnuevo002;
    case 2
        %SEGUNDO TRIÁNGULO
        %Primer paso (i+j+k=2)
        cnuevo200=bari(1)*c(2)+bari(2)*c(7)+bari(3)*c(8);
        cnuevo110=bari(1)*c(7)+bari(2)*c(12)+bari(3)*c(14);
        cnuevo020=bari(1)*c(12)+bari(2)*c(3)+bari(3)*c(5);
        cnuevo011=bari(1)*c(14)+bari(2)*c(11)+bari(3)*c(18);
        cnuevo002=bari(1)*c(17)+bari(2)*c(18)+bari(3)*c(19);
        cnuevo101=bari(1)*c(8)+bari(2)*c(14)+bari(3)*c(17);
        %Segundo paso (i+j+k=1)
        cnuevo100=bari(1)*cnuevo200+bari(2)*cnuevo110+bari(3)*cnuevo101;
        cnuevo010=bari(1)*cnuevo110+bari(2)*cnuevo020+bari(3)*cnuevo011;
        cnuevo001=bari(1)*cnuevo101+bari(2)*cnuevo011+bari(3)*cnuevo002;
    case 3
        %TERCER TRIÁNGULO
        %Primer paso (i+j+k=2)
        cnuevo200=bari(1)*c(3)+bari(2)*c(10)+bari(3)*c(11);
        cnuevo110=bari(1)*c(10)+bari(2)*c(6)+bari(3)*c(15);
        cnuevo020=bari(1)*c(6)+bari(2)*c(1)+bari(3)*c(5);
        cnuevo011=bari(1)*c(15)+bari(2)*c(5)+bari(3)*c(16);
        cnuevo002=bari(1)*c(18)+bari(2)*c(16)+bari(3)*c(19);
        cnuevo101=bari(1)*c(11)+bari(2)*c(15)+bari(3)*c(18);
        %Segundo paso (i+j+k=1)
        cnuevo100=bari(1)*cnuevo200+bari(2)*cnuevo110+bari(3)*cnuevo101;
        cnuevo010=bari(1)*cnuevo110+bari(2)*cnuevo020+bari(3)*cnuevo011;
        cnuevo001=bari(1)*cnuevo101+bari(2)*cnuevo011+bari(3)*cnuevo002;
end
%Ahora l=2, i+j+k=0 (segundo paso del algoritmo)
res=bari(1)*cnuevo100+bari(2)*cnuevo010+bari(3)*cnuevo001
```

Figura 5.4: Algoritmo de Castelaju para Clough-Tocher.

En Argyris usaremos tres bucles anidados, sin que haya refinamiento en el triángulo.

```
for l = 1:5 %argyris es quintico
    for i = 0:5-l
        for j = 0:5-l-i
            for k=0:5-l-i-j
                if (i+j+k)==(5-l)
                    cantiguo(i+1,j+1,k+1)=
                        bari(1)*cantiguo(i+2,j+1,k+1)+
                        bari(2)*cantiguo(i+1,j+2,k+1)+
                        bari(3)*cantiguo(i+1,j+1,k+2);
                end
            end
        end
    end
end
res=cantiguo(1,1,1)
```

Figura 5.5: Algoritmo de Castelaju para Argyris.

Finalmente, los 3 ficheros devuelven el resultado de la interpolación tras realizar el Algoritmo de Casteljau.

5.3. Errores en Powell-Sabin

Vamos a utilizar el programa creado para interpolar con el método de Powell-Sabin con tres ejemplos test. Como primer ejemplo probaremos con el polinomio $f(x, y) = x^2 + 4xy + 20$ de grado 2. Obtendremos un error 0 en todos los valores que probemos y en todas las triangulaciones posibles, ya que Powell-Sabin interpola de manera exacta polinomios de hasta grado 2. Como segundo ejemplo consideraremos la función de Franke, definida más abajo, y por último, la función $f(x, y) = x^3y^2$. Consideraremos siempre dos triangulaciones diferentes y sucesivos refinamientos de las mismas. Los resultados que mostramos se refieren únicamente a la función de Franke y al polinomio de grado 5 ya que en el caso cuadrático los errores han sido siempre 0.

- Función de Franke

  Esta función está definida como:

$$\begin{aligned} F(x,y) =& 0{,}75exp(-(9x-7)^2/4 - (9y-7)^2/4) \\ &+ 0{,}75exp(-(9x-10)^2/49 + (9y-10)/10) \\ &+ 0{,}5exp(-(9x-2)^2/4 - (9y-6)^2/4) \\ &- 0{,}2exp(-(9x-5)^2 - (9y-2)^2). \end{aligned}$$

Utilizaremos las funciones $franke2$ y $franke2d$ del paquete splinepack de MATLAB para obtener los valores de la función de Franke y sus derivadas parciales hasta segundo orden en x y en y en los puntos que queramos. En este caso, las usaremos para obtener el valor de la función y de sus derivadas evaluados en los vértices del triángulo al que pertenezca el punto que vamos a interpolar.

Comenzaremos con la función de franke en la triangulación definida por los datos 'tri36.dat', archivo encontrado en el paquete splinepack [4]. Esta triangulación consta de 36 vértices. Realizaremos dos refinamientos uniformes, ayudandonos de la función $refine$, también incluida en el paquete splinepack[4], para interpolar cada vez en una triangulación más fina. En el primer refinamiento, pasará a tener 125 vértices, y tras el segundo, pasará a tener 465 vértices.

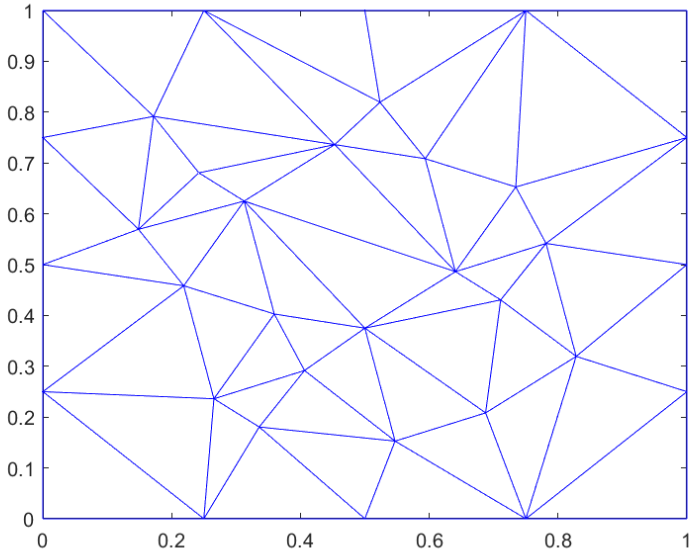


Figura 5.6: Triangulación con 36 vértices.

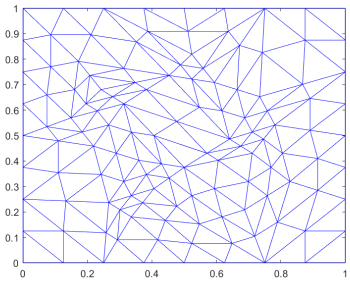


Figura 5.7: Primer refinamiento uniforme.

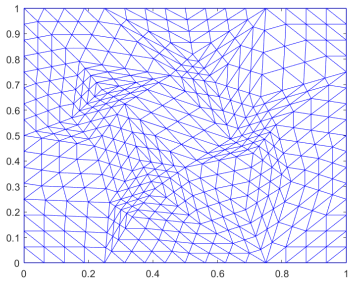


Figura 5.8: Segundo refinamiento uniforme.

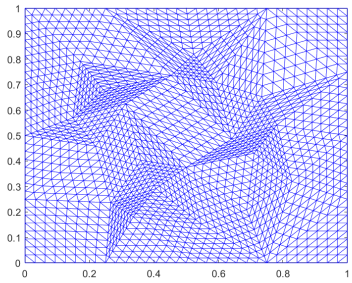


Figura 5.9: Tercer refinamiento uniforme.

Tras interpolar el punto (0.4,0.6) en los diferentes refinamientos, obtenemos la siguiente tabla de errores, teniendo en cuenta que n es el número de vértices de la triangulación y el resultado de evaluar la función de Franke en el punto (0.4,0.6) es 0.4682.

n	res	error máximo
36	0.4963	2.81(-2)
125	0.4674	7.67(-4)
465	0.4682	8.37(-6)
1793	0.4682	4.31(-6)

Ahora vamos a utilizar otra triangulación, para la que será necesario realizar 5 refinamientos uniformes para obtener un resultado casi exacto. Es una triangulación de tipo 1 con 9 vértices, encontrada también en los datos disponibles en el paquete splinepack de Matlab, en el fichero 'type1.9'.

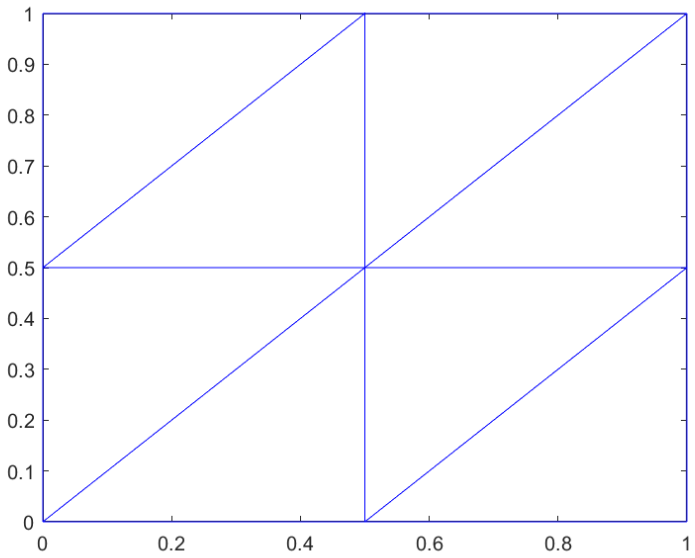
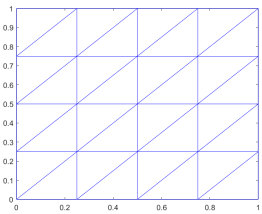
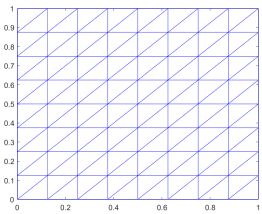


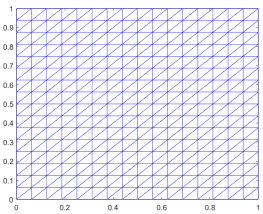
Figura 5.10: Triangulación con 9 vértices.



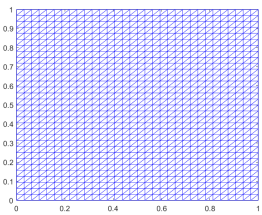
(a) Refinamiento 1



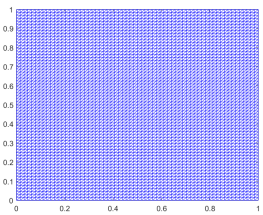
(b) Refinamiento 2



(c) Refinamiento 3



(d) Refinamiento 4



(e) Refinamiento 5

Figura 5.11: Primeros refinamientos uniformes.

Evaluando el mismo punto, obtenemos la siguiente tabla de errores:

n	res	error máximo
9	0.3871	8.10(-2)
25	0.4733	5.12(-3)
81	0.4672	9.52(-4)
289	0.4676	5.36(-4)
1089	0.4683	1.42(-5)

Podemos observar que esta vez tuvimos que refinar más la triangulación para llegar al resultado casi exacto, ya que esta tenía menos vértices que la anterior.

- Interpolaremos un polinomio de grado 5. Ahora vamor a relizar la misma prueba pero con el polinomio $f(x, y) = x^3y^2$. El valor real del punto (0.4,0.6) evaluado en dicho polinomio es 0.023. Con la primera triangulación, obtenemos la siguiente tabla de errores:

n	res	error máximo
36	0.0233	2.47(-4)
125	0.0231	9.50(-5)
465	0.0230	6.48(-7)
1793	0.0230	6.53(-7)

Con la segunda triangulación de menos vértices obtenemos:

n	res	error máximo
9	0.02338	3.43(-4)
25	0.02306	2.97(-5)
81	0.02299	4.15(-5)
289	0.02302	1.01(-5)
1089	0.02304	5.01(-7)

A continuación, realizaremos la evaluación en una malla equiespaciada de 10000 puntos sobre cada triangulación y sus refinamientos para ver cuál sería el error máximo en cada caso.

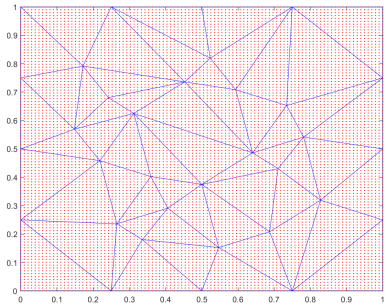


Figura 5.12: Triangulación de 36 vértices con 10000 puntos equiespaciados.

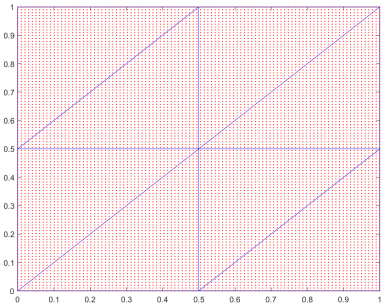


Figura 5.13: Triangulación de 9 vértices con 10000 puntos equiespaciados.

Ejecutaremos el fichero que realiza esta evaluación en la malla, introduciendo la triangulación deseada y sus refinamientos, con el número de puntos que queramos en la malla, el método de Powell-Sabin y el anterior polinomio y la función de Franke, obtenemos los siguientes errores.

función	n	error máximo
$f(x,y) = x^3y^2$	36	9.65(-1)
	125	1.86(-2)
	465	9.43(-5)
	1793	1.28(-5)
	9	3.06(-2)
	25	9.96(-3)
	81	5.02(-4)
	289	5.80(-5)
	1089	6.51(-6)
$Franke$	36	2.79(-1)
	125	1.54(-2)
	465	3.44(-2)
	1793	8.59(-3)
	9	4.96(-1)
	25	9.03(-2)
	81	1.42(-2)
	289	1.71(-3)
	1089	1.92(-4)

5.4. Errores en Clough-Tocher

Se procederá de la misma forma que en Powell-Sabin. Evaluando el polinomio $f(x, y) = y^3 + x^2y$, que nos da un error de 0 en todos los casos, ya que Clough-Tocher devuelve resultados exactos para polinomios hasta de grado 3. Ahora realiaremos la prueba con la función de Franke sobre las dos triangulaciones ya vistas de 36 y de 9 vértices y sus refinamientos, y más tarde con el polinomio $f(x, y) = x^4y^2$, también con ambas triangulaciones.

El fichero de comprobación de Clough-Tocher, obtiene los coeficienes necesarios para interpolar de la función *ct*, definida en el paquete splinepack, a la cual debemos propocionarla los datos de interpolación de la función correspondiente en los vértices del triángulo de la triangulación al que pertenece el punto a interpolar.

- Comenzamos interpolando la función de Franke nuevamente en el punto (0.4,0.6), donde tiene un valor exacto de 0.468167. En la triangulación de 36 vértices descrita en la Figura 5.6, obtenemos la siguiente tabla de errores:

n	res	error máximo
36	0.4945	2.64(-2)
125	0.4714	3.28(-3)
465	0.4681	6.43(-7)
1793	0.4681	1.16(-6)

  Realizando el mismo proceso sobre la triangulación de 9 vértices descrita en la Figura 5.10, también interpolando la función de Franke en el mismo punto, obtenemos la siguiente tabla de errores.

n	res	error máximo
9	0.3969	7.12(-2)
25	0.4496	1.85(-2)
81	0.4684	2.59(-4)
289	0.4690	8.57(-4)
1089	0.4681	3.79(-8)

- En este caso, interpolaremos de nuevo sobre las dos triangulaciones anteriores en el punto (0.4,0.6) pero esta vez interpolaremos un polinomio de grado 6 $f(x, y) = x^4y^2$. El valor exacto que buscamos en este caso es 0.009216.

Introduciendo los datos de interpolación correspondientes al nuevo polinomio, sobre la triangulación de 36 vertices de la Figura 5.6, obtenemos la siguiente tabla de errores.

n	res	error máximo
36	0.009330	1.14(-4)
125	0.009226	1.04(-5)
465	0.009215	9.93(-8)
1793	0.009216	9.05(-9)

De igual manera, sobre la triangulación de 9 vértices de la Figura 5.10, obtenemos la siguiente tabla de errores.

n	res	error máximo
9	0.008286	9.29(-4)
25	0.008943	2.72(-4)
81	0.009211	4.59(-6)
289	0.009116	9.95(-5)
1089	0.009216	1.68(-8)

De nuevo, realizaremos la evaluación en una malla equiespaciada de 10000 puntos sobre cada triangulación y sus refinamientos para ver cuál sería el error máximo en cada caso. Serán los puntos de las Figuras 5.11 y 5.12 y sus refinamientos correspondientes, siendo n el número de vértices de la triangulación.

función	n	error máximo
$f(x,y) = x^4y^2$	36	9.75(-1)
	125	2.46(-2)
	465	1.61(-2)
	1793	7.48(-3)
	9	8.43(-2)
	25	3.71(-2)
	81	2.69(-2)
	289	2.01(-2)
	1089	6.95(-3)
Franke	36	3.27(-1)
	125	1.97(-1)
	465	2.03(-2)
	1793	1.15(-2)
	9	4.99(-1)
	25	8.36(-2)
	81	3.47(-2)
	289	2.02(-2)
	1089	1.00(-2)

Podemos observar que este método es algo más exacto que el de Powell-Sabin y que nuevamente, cuantos más vértices tenga la triangulación sobre la que interpolamos, menor será el error.

5.5. Errores en Argyris

Vamos a realizar la comprobación de los errores con nuestra función de Matlab que interpola mediante el método de Argyris. Para ello usaremos las dos triangulaciones vistas en el apartado anerior con sus refinamientos. La primera es la que tenemos descrita en la Figura 5.6 con 36 vértices, y la segunda es la representada en la Figura 5.10 con únicamente 9 vértices. Interpolaremos primero el polinomio $f(x,y) = 5x^2y^3 + y^4 + x^3y + 81$. En todos los casos, sobre cualquier tipo de triangulación, con este polinomio obtenemos un resultado exacto, ya que Argyris interpola de manera exacta en polinomios hasta orden 5. Ahora, igual que en los casos anteriores, interpolaremos dos funciones diferentes. La primera la función de Franke, y la segunda un

polinomio de grado 9.

- En el primer caso interpolaremos la función de Franke en el punto (0.4,0.6), igual que en los ejemplos anteriores. Ejecutaremos la función creada para comprobar ejemplos concretos con Argyris, e introduciremos los datos de interpolación requeridos. El resultado exacto que buscamos es 0.46816774.

  Sobre la triangulación de 36 vértices definida anteriormente y representada en la Figura 5.6, obtenemos la siguiente tabla de errores.

n	res	error máximo
36	0.4711	3.01(-3)
125	0.4683	1.95(-4)
465	0.4681	2.44(-9)
1793	0.4681	3.27(-9)

  Sobre la otra triangulación de 9 vértices definida anteriormente y representada en la Figura 5.10, obtenemos la siguiente tabla de errores.

n	res	error máximo
9	0.4418	2.63(-2)
25	0.4641	4.03(-3)
81	0.4681	1.64(-6)
289	0.4681	1.32(-6)
1089	0.4681	1.04(-9)

- En este segundo caso realizaremos la prueba con un polinomio de grado 9 para ver cómo se comporta. El polinomio será el siguiente

$$f(x, y) = x^5 y^4.$$

  Ahora el resultado exacto es 0.001327. Comenzaremos con la triangulación de 9 vértices vista en la Figura 5.10. Introduciendo los datos correctos de interpolación, obtenemos un valor de 0.0015, y en los sucesivos refinamientos de la triangulación hasta llegar a un resultado más exacto, obtenemos lo siguiente:

n	res	error máximo
9	0.001501	1.74(-4)
25	0.001354	2.69(-5)
81	0.001327	4.66(-8)
289	0.001327	6.69(-9)
1089	0.001327	1.09(-11)

Si ahora con este mismo polinomio de grado 9 interpolamos sobre la triangulación de 36 vértices de la Figura 5.6, que hemos estado usando anteriormente, para el mismo punto, obtenemos los siguientes errores:

n	res	error máximo
36	0.001328	1.10(-6)
125	0.001327	3.25(-7)
465	0.001327	5.38(-11)
1793	0.001327	2.20(-12)

De nuevo, realizaremos la evaluación en una malla equiespaciada de 10000 puntos sobre cada triangulación y sus refinamientos para ver cuál sería el error máximo en cada caso. Serán los puntos de las Figuras 5.11 y 5.12 y sus refinamientos correspondientes, siendo n el número de vértices de la triangulación.

función	n	error máximo
$f(x,y) = x^5y^4$	36	9.54(-1)
	125	2.49(-7)
	465	4.42(-9)
	1793	6.22(-11)
	9	9.92(-6)
	25	2.03(-7)
	81	3.90(-9)
	289	4.89(-11)
	1089	4.21(-13)
$Franke$	36	4.17(-1)
	125	4.59(-4)
	465	1.03(-5)
	1793	1.59(-7)
	9	3.22(-1)
	25	4.05(-2)
	81	2.64(-3)
	289	6.62(-5)
	1089	1.36(-6)

Podemos concluir, como ya habíamos visto en los teoremas de errores, que el método de Argyrirs es mucho más preciso que el de Clough-Tocher y el de Powell-Sabin, y el método Clough-Tocher es más preciso que el de Powell-Sabin, y como hemos comprobado, cuantos más vértices tenga la triangulación sobre la que trabajamos, es decir, cuanto más refinada esté, menor será el error que obtendremos.

Apéndice

Funciones que calculan los coeficientes de los interpolantes

```
%POWELL-SABIN

function c=powells(v,z,zx,zy)

c(1:3)=z(1:3);
%w sera el vector con los coeficientes de los puntos
%medios de los vertices
w=(v(1:3,:)+v([2,3,1],:))/2;

%debemos calcular tambien el incentro
TR=triangulation([1:3],v(:,1),v(:,2));
vc = incenter(TR);

%y sus coordenadas baricentricas
denominador = (v(2,2)-v(3,2))*(v(1,1)-v(3,1))
            + (v(3,1)-v(2,1))*(v(1,2)-v(3,2));

a(1)= ((v(2,2)-v(3,2))*(vc(1)-v(3,1)) +
       (v(3,1)-v(2,1))*(vc(2)-v(3,2))) / denominador;
a(2)= ((v(3,2)-v(1,2))*(vc(1)-v(3,1)) +
       (v(1,1)-v(3,1))*(vc(2)-v(3,2))) / denominador;
a(3) = 1 - a(1) - a(2);

c(4:6)=(([w(1,1),vc(1),w(3,1)]-v(1,1) )*zx(1)+
        ([w(1,2),vc(2),w(3,2)]-v(1,2) )*zy(1)) /2+z(1);
```

```
c(7:9)=(([w(2,1),vc(1),w(1,1)]-v(2,1) )*zx(2)+
        ([w(2,2),vc(2),w(1,2)]-v(2,2) )*zy(2)) /2+z(2);
c(10:12)=(([w(3,1),vc(1),w(2,1)]-v(3,1) )*zx(3)+
        ([w(3,2),vc(2),w(2,2)]-v(3,2) )*zy(3)) /2+z(3);

c(13:18)=([c(4),c(7),c(10),c(5),c(8),c(11)]+
          [c(9),c(12),c(6),c(8),c(11),c(5)])/2;

c(19)=a(1)*c(5)+a(2)*c(8)+a(3)*c(11);

end
```

```
%CLOUGH-TOCHER

function c=clought(v,z,zx,zy,ze,e)

c(1:3)=z(1:3);

%debemos calcular tambien el baricentro
vb=(v(1,:)+v(2,:)+v(3,:))/3;

c(4:6)=(([v(2,1),vb(1),v(3,1)]-v(1,1))*zx(1)+
        ([v(2,2),vb(2),v(3,2)]-v(1,2))*zy(1))/3 +z(1);
c(7:9)=(([v(3,1),vb(1),v(1,1)]-v(2,1))*zx(2)+
        ([v(3,2),vb(2),v(1,2)]-v(2,2))*zy(2))/3 +z(2);
c(10:12)=(([v(1,1),vb(1),v(2,1)]-v(3,1))*zx(3)+
          ([v(1,2),vb(2),v(2,2)]-v(3,2))*zy(3))/3 +z(3);

c(13)=(4*ze(1))/(6*e(3,1))-(c(5)+c(8))/2-
      (e(1,1)(c(1)+2*c(4)+c(9)))/(2*e(3,1))-
      (e(2,1)*(c(4)+2*c(9)+c(2)))/(2*e(3,1));
c(14)=(4*ze(2))/(6*e(3,2))-(c(8)+c(11))/2-
      (e(1,2)*(c(2)+2*c(7)+c(12)))/(2*e(3,2))-
      (e(2,2)*(c(7)+2*c(12)+c(3)))/(2*e(3,2));
c(15)=(4*ze(3))/(6*e(3,3))-(c(11)+c(5))/2-
      (e(1,3)*(c(3)+2*c(10)+c(6)))/(2*e(3,3))-
      (e(2,3)*(c(10)+2*c(6)+c(1)))/(2*e(3,3));

c(16)=(c(15)+c(5)+c(13))/3;
c(17)=(c(13)+c(8)+c(14))/3;
```

```
c(18)=(c(14)+c(11)+c(15))/3;
c(19)=(c(16)+c(17)+c(18))/3;


end
```

```
%ARGYRIS

function c=Argyris(v,z,zxy,ze,e)

hx(1:3,1)=v(1:3,1)-v(1,1);
hy(1:3,1)=v(1:3,2)-v(1,2);

hx(1:3,2)=v(1:3,1)-v(2,1);
hy(1:3,2)=v(1:3,2)-v(2,2);

hx(1:3,3)=v(1:3,1)-v(3,1);
hy(1:3,3)=v(1:3,2)-v(3,2);

%Guardamos los coeficientes en el vector c en el orden
%lexicografico que hemos visto:
%c_500,c_410,c_401,c_320,c_311,c_302,c_230,c_221,c_212,
%c_203,c_140,c_131,c_122,c_113,c_104,c_050,c_041,c_032,
%c_023,c_014,c_005

c(1)=z(1);
c(2)=(hx(2,1)*zxy(1,3)+hy(2,1)*zxy(1,4))/5 +z(1);
c(3)=(hx(3,1)*zxy(1,3)+hy(3,1)*zxy(1,4))/5 +z(1);
c(4)=(hx(2,1)*hx(2,1)*zxy(1,1)+2*hx(2,1)*hy(2,1)*
      zxy(1,2)+hy(2,1)*hy(2,1)*zxy(1,5))/20+2*c(2)-z(1);
c(5)=(hx(2,1)*hx(3,1)*zxy(1,1)+(hx(2,1)*hy(3,1)+
      hx(3,1)*hy(2,1))*zxy(1,2)+hy(2,1)*hy(3,1)*
      zxy(1,5))/20+c(2)+c(3)-z(1);
c(6)=(hx(3,1)*hx(3,1)*zxy(1,1)+2*hx(3,1)*hy(3,1)*
      zxy(1,2)+hy(3,1)*hy(3,1)*zxy(1,5))/20+2*c(3)-z(1);


c(16)=z(2);
c(17)=(hx(3,2)*zxy(2,3)+hy(3,2)*zxy(2,4))/5 +z(2);
c(11)=(hx(1,2)*zxy(2,3)+hy(1,2)*zxy(2,4))/5 +z(2);
```

```
c(18)=(hx(3,2)*hx(3,2)*zxy(2,1)+2*hx(3,2)*hy(3,2)*
        zxy(2,2)+hy(3,2)*hy(3,2)*zxy(2,5))/20+
        2*c(17)-z(2);
c(12)=(hx(3,2)*hx(1,2)*zxy(2,1)+(hx(3,2)*hy(1,2)+
        hx(1,2)*hy(3,2))*zxy(2,2)+hy(3,2)*hy(1,2)*
        zxy(2,5))/20+c(17)+c(11)-z(2);
c(7)=(hx(1,2)*hx(1,2)*zxy(2,1)+2*hx(1,2)*hy(1,2)*
      zxy(2,2) + hy(1,2)*hy(1,2)*zxy(2,5))/20+2*c(11)
      -z(2);

c(21)=z(3);
c(15)=(hx(1,3)*zxy(3,3)+hy(1,3)*zxy(3,4))/5 +z(3);
c(20)=(hx(2,3)*zxy(3,3)+hy(2,3)*zxy(3,4))/5 +z(3);
c(10)=(hx(1,3)*hx(1,3)*zxy(3,1)+2*hx(1,3)*hy(1,3)*
        zxy(3,2)+hy(1,3)*hy(1,3)*zxy(3,5))/20+2*c(15)
        -z(3);
c(14)=(hx(1,3)*hx(2,3)*zxy(3,1)+(hx(1,3)*hy(2,3)+
        hx(2,3)*hy(1,3))*zxy(3,2)+ hy(2,3)*hy(1,3)*
        zxy(3,5))/20+c(15)+c(20)-z(3);
c(19)=(hx(2,3)*hx(2,3)*zxy(3,1)+2*hx(2,3)*hy(2,3)*
        zxy(3,2)+hy(2,3)*hy(2,3)*zxy(3,5))/20+
        2*c(20)-z(3);
c(13)=(16*ze(2))/(30*e(2,1))-(c(11)+4*c(12)+4*c(14)+
        c(20))/6-(e(2,2)*(c(16)+4*c(17)+6*c(18)+
        4*c(19)+c(20)))/(6*e(2,1))-(e(2,3)*(c(17)+
        4*c(18)+6*c(19)+4*c(20)+c(21)))/(6*e(2,1));
c(8)=(16*ze(1))/(30*e(1,1))-(c(3)+4*c(5)+4*c(12)+
      c(17))/6-(e(1,2)*(c(1)+4*c(2)+6*c(4)+4*c(7)+
      c(11)))/(6*e(1,1))-(e(1,3)*(c(2)+4*c(4)+
      6*c(7)+4*c(11)+c(16)))/(6*e(1,1));
c(9)=(16*ze(3))/(30*e(3,1))-(c(20)+4*c(14)+4*c(5)+
      c(2))/6-(e(3,2)*(c(21)+4*c(15)+6*c(10)+4*c(6)+
      c(3)))/(6*e(3,1))-(e(3,3)*(c(15)+4*c(10)+6*c(6)+
      4*c(3)+c(1)))/(6*e(3,1));
end
```

Programas de evaluación

```
trianinicial = input("Introduce una triangulacion:");
```

```
triplot(traininicial)
title('Triangulacion en la que quieres interpolar')

C = input("Introduce el punto en el que quieras evaluar:");

for j=1:length(traininicial(:,1))
    if cartesianToBarycentric(traininicial,j,C)>=0
        break
    end
end

v=traininicial.Points(traininicial(j,:),:);

disp("Los vertices del triangulo de la triangulacion
donde se encuentra el punto y realizaremos el
refinamiento son:");
disp(v);

%Descomenta el caso en el que estes

%PARA INTRODUCIR LOS VALORES POR EL TECLADO
%Valores de p en los vertices
%z = input("Introduce los valores de tu funcion a interpolar
%en los vertices:");

%Valores de px en los vertices
%zx = input("Introduce los valores de la derivada de x
%en los vertices:");

%Valores de py en los vertices
%zy = input("Introduce los valores de la derivada de y
%en los vertices:");


%PARA EL POLINOMIO x^3y^2
% z=[v(1,1)^3*v(1,2)^2,v(2,1)^3*v(2,2)^2,v(3,1)^3*v(3,2)^2];
% zx=[3*v(1,1)^2*v(1,2)^2,3*v(2,1)^2*v(2,2)^2,3*v(3,1)^2*v(3,2)^2];
% zy=[2*v(1,1)^3*v(1,2),2*v(2,1)^3*v(2,2),2*v(3,1)^3*v(3,2)];

%PARA LA FUNCION DE FRANKE
```

```
d1=franke2d(v(1,1),v(1,2));
d2=franke2d(v(2,1),v(2,2));
d3=franke2d(v(3,1),v(3,2));
z=[d1(1),d2(1),d3(1)];
zx=[d1(2),d2(2),d3(2)];
zy=[d1(3),d2(3),d3(3)];


%Coeficientes del polinomio de interpolacion
c=powells(v,z,zx,zy);

%Debemos encontrar en que triangulo del refinamiento se
%encuentra el punto a evaluar gracias a las coordenadas
%baricentricas con respecto a los triangulos que se
%forman con el incentro y utilizar el algoritmo de
%Casteljau con los coeficientes correspondientes.


%Debemos definir la triangulacion de Powell-Sabin
%para realizar el refinamiento.
triangulogrande=triangulation([1:3],v);
vc = incenter(triangulogrande);
%w sera el vector con los coeficientes de los puntos
%medios de los vertices
w=(v(1:3,:)+v([2,3,1],:))/2;

P=[v;w;vc];
T = [1 4 7; 4 2 7; 2 5 7; 5 3 7;3 6 7;6 1 7];
TR = triangulation(T,P);
triplot(TR)
title('Triangulo donde se encuentra tu punto a evaluar
con refinamiento PS')

%Dependiendo a que triangulo del refinamiento de
%Powell-Sabin pertenezca vamos a diferenciar los
%6 diferentes casos.
%Empezando por el primer triangulo que sera formado
%por v_1, el incentro y el punto medio del lado v_1v_2
%y los ordenaremos yendo hacia v_2. Para caso seran
%unas coordenadas baricentricas diferentes.
```

```
%Con el siguiente for encontraremos a que triangulo de
%la triangulacion pertenece el punto C y calcularemos
%las coordenadas baricentricas dependiendo de esto.
for j=1:6
    if cartesianToBarycentric(TR,j,C)>=0
        bari = cartesianToBarycentric(TR,j,C);
        break
    end
end

%Vamos a realizar el algoritmo de Casteljau con los coeficientes
%correspondientes dependiendo al triangulo al que
%pertenezca el punto. En este caso como es un
%polinomio cuadratico, la dimension es 2 y solo
%deberemos realizar dos pasos de este algoritmo.

%Primero l=1, i+j+k=1
switch j
    case 1
        %PRIMER TRIANGULO
        crnuevo100=bari(1)*c(1)+bari(2)*c(4)+bari(3)*c(5);
        crnuevo010=bari(1)*c(4)+bari(2)*c(13)+bari(3)*c(16);
        crnuevo001=bari(1)*c(5)+bari(2)*c(16)+bari(3)*c(19);
    case 2
        %SEGUNDO TRIANGULO
        crnuevo100=bari(1)*c(13)+bari(2)*c(9)+bari(3)*c(16);
        crnuevo010=bari(1)*c(9)+bari(2)*c(2)+bari(3)*c(8);
        crnuevo001=bari(1)*c(16)+bari(2)*c(8)+bari(3)*c(19);
    case 3
        %TERCER TRIANGULO
        crnuevo100=bari(1)*c(2)+bari(2)*c(7)+bari(3)*c(8);
        crnuevo010=bari(1)*c(7)+bari(2)*c(14)+bari(3)*c(17);
        crnuevo001=bari(1)*c(8)+bari(2)*c(17)+bari(3)*c(19);
    case 4
        %CUARTO TRIANGULO
        crnuevo100=bari(1)*c(14)+bari(2)*c(12)+bari(3)*c(17);
        crnuevo010=bari(1)*c(12)+bari(2)*c(3)+bari(3)*c(11);
        crnuevo001=bari(1)*c(17)+bari(2)*c(11)+bari(3)*c(19);
    case 5
        %QUINTO TRIANGULO
        crnuevo100=bari(1)*c(3)+bari(2)*c(10)+bari(3)*c(11);
```

```
        crnuevo010=bari(1)*c(10)+bari(2)*c(15)+bari(3)*c(18);
        crnuevo001=bari(1)*c(11)+bari(2)*c(18)+bari(3)*c(19);
    case 6
        %SEXTO TRIANGULO
        crnuevo100=bari(1)*c(15)+bari(2)*c(6)+bari(3)*c(18);
        crnuevo010=bari(1)*c(6)+bari(2)*c(1)+bari(3)*c(5);
        crnuevo001=bari(1)*c(18)+bari(2)*c(5)+bari(3)*c(19);
end
%Ahora l=2, i+j+k=0 (segundo paso del algoritmo)
res=bari(1)*crnuevo100+bari(2)*crnuevo010+bari(3)*crnuevo001
```

Programas que evalua los errores en una malla

```
%Definir el tamano de la red de puntos
n = 100; % Numero de puntos en cada direccion

%Generar las coordenadas x e y para la red de puntos
[xm, ym] = meshgrid(linspace(0, 1, n), linspace(0, 1, n));

%Inicializar el vector de resultados
resultados_vector = zeros(n*n, 1);
resultados_vector_real = zeros(n*n, 1);
error = zeros(n*n, 1);


%Definimos la triangulacion sobre la que interpolaremos
trianinicial=triangulation(relaciones, vertices);

for kk = 1:n
    for jj = 1:n
        C=[xm(kk,jj),ym(kk,jj)];

        %Aqui pones el fichero de comprobacion
        %del metodo que quieras
        %comprobaciongeneralPS;
        %combrobaciongeneralCT;
        comprobaciongeneralArgyris;
        resultados_vector((kk-1)*n + jj) = res;
```

```
        %Depende de la funcion que interpoles
        %defines el resultado:
        %resultados_vector_real((kk-1)*n + jj) =
        %xm(kk,jj)^5*ym(kk,jj)^4;
        resultados_vector_real((kk-1)*n + jj) =
        franke2(xm(kk,jj),ym(kk,jj));

        error((kk-1)*n + jj)=
        resultados_vector_real((kk-1)*n + jj)-
        resultados_vector((kk-1)*n + jj);
    end
end
max(error)

% Graficar los puntos pintados de rojo
triplot(trianinicial);
hold on;
scatter(xm(:), ym(:), 2, 'r', 'filled');
hold off;
```

Bibliografía

[1] L.L. Schumaker, 2015. *Spline Functions. Computational Methods*, SIAM, Philadelphia, USA.

[2] M.J. Lai and L.L. Schumaker, 2007. *Spline Functions on Triangulations*, Cambridge University Press, Cambridge, USA.

[3] Ph.G. Ciarlet, 2002. *The Finite Element Method for Elliptic Problems*, SIAM, Philadelphia, USA.

[4] L.L. Schumaker, 2015. *Splinepak package*, Nashboro Press, USA.

[5] K. Hölling and J. Hörner, 2013. *Approximation and Modeling with B-Splines*, SIAM, Philadelphia, USA.

[6] T. Tantau, 2007. *The TikZ and PGF Packages*, Institut für Theoretische Informatik, Universität zu Lübek, Germany.

[7] V. Dominguez and F.J. Sayas, 2006. *A simple Matlab implementation of the Argyris element*, Universidad Pública de Navarra, Pamplona, Spain.

[8] T. Wang, 1992. *A C^2-quintic spline interpolation scheme on triangulation*, Jilin University, Jilin, China.