Universidad de Valladolid

Facultad de Ciencias

Trabajo Fin de Grado

Grado en Matemáticas

Técnicas de Bézier e Interpolación mediante Splines

Autora: Alicia Nieto Ramos

Tutora: María Paz Calvo Cabrero

Índice general

Introducción

El diseño gráfico y el diseño asistido por ordenador surgieron a mediados de los años 60 y se han convertido a día de hoy, en dos herramientas fundamentales tanto en ciencia como en ingeniería. El objetivo de este Trabajo de Fin de Grado es el estudio de los splines de Bézier a trozos definidos en el plano, partiendo de los principales conceptos y propiedades relacionados con los polinomios de Bernstein y las curvas de Bézier, ilustrando con dos ejemplos el uso de estos en el ámbito del diseño.

En el Capítulo 1, se definen los polinomios de Bernstein tanto en $[0, 1]$ como en un intervalo $[a, b]$ real arbitrairio, y se enuncian y demuestran algunas de sus propiedades más significativas. También se enuncia y prueba el Teorema de aproximación de Weierstrass utilizando estos polinomios.

En el Capítulo 2, se emplea la base que forman los polinomios de Bernstein para escribir curvas en el plano, lo que se conoce como *representación de Bézier de una curva*. Los coeficientes que acompañan a los polinomios de Bernstein en dicha representación, reciben el nombre de *nodos de Bézier*. A continuación se estudia el algoritmo de De Casteljau, fundamental en la construcción recursiva de las curvas de Bézier y se finaliza el capítulo con resultados relativos a la derivación de dichas curvas.

En el Capítulo 3, se estudian las operaciones más usuales que se realizan cuando se trabaja con curvas de Bézier: la operación de subdivisión (que permite definir parte de la traza de una curva de Bézier definiendo nuevos nodos de Bézier a partir de los iniciales), la elevación de grado (profundizando también en la elevación reiterada del grado de una curva de Bézier) y finalmente, la reducción de grado.

En el Capítulo 4, se da la definición de spline de Bézier a trozos, y se especifican las condiciones que debe cumplir un spline para poseer regularidad $\mathcal{C}^r$, $r \in \mathbb{N}$. Se tratan con mayor profundidad los casos de spline cuadrático con regularidad $\mathcal{C}^1$ y de spline cúbico con regularidad $\mathcal{C}^2$.

En el Capítulo 5, se consideran dos imágenes (la letra S de la tipografía Times New Roman© y una firma) y se efectúa la interpolación mediante splines de Bézier de varios grados con y sin regularidad, para ilustrar la aplicación directa de estas curvas en el área del diseño gráfico. Se emplea MATLAB® como programa principal y GeoGebra como herramienta auxiliar para determinar los nodos de Bézier de manera interactiva. En el caso

de la firma, se corrobora el resultado obtenido mediante InkScape, programa orientado al diseño gráfico.

Finalmente, en el Apéndice A se han incluido las funciones de MATLAB® que implementan los distintos algoritmos estudiados a lo largo de este trabajo y se han utilizado para la elaboración de las gráficas que aparecen en la memoria. Se destaca la función `bspline_cubico.m`, que efectúa la interpolación mediante splines cúbicos de Bézier para L trozos con regularidad $\mathcal{C}^2$, crucial para las aproximaciones de los ejemplos presentes en el Capítulo 5.

Me gustaría agradecer a la Dra. María Paz Calvo Cabrero su implicación, dedicación y paciencia mostrados durante la elaboración de este trabajo.

Capítulo 1

Polinomios de Bernstein

En este capítulo vamos a definir y a estudiar las principales propiedades de los polinomios de Bernstein.

1.1. Definición y propiedades

A medida que fueron evolucionando las técnicas interpolatorias, surgieron distintas bases del espacio de polinomios de grado $\leq n$, al que denotaremos por $\mathbb{P}_n$.

Si bien la más conocida es la base de monomios $\{1, x^2, \ldots, x^n\}$, Lagrange introduce junto a su polinomio interpolador una nueva base $\{L_0(x), L_1(x), \ldots, L_n(x)\}$, donde

$$L_i(x) = \frac{\prod_{j \neq i}(x - x_j)}{\prod_{j \neq i}(x_i - x_j)}, \qquad i = 0, \ldots, n,$$

y Newton la base $\{w_0(x), w_1(x), \ldots, w_n(x)\}$ donde $w_i(x) \in \mathbb{P}_i$ y

$$\begin{aligned} w_0(x) &= 1, \\ w_i(x) &= \prod_{j=0}^{i-1}(x - x_j), \qquad i = 1, \ldots, n, \end{aligned}$$

que, a diferencia de la primera, dependen de $n + 1$ nodos concretos $x_0, \ldots, x_n$.

Sin embargo, es en los últimos treinta años del siglo pasado cuando el auge tanto del diseño como de la construcción asistidos por ordenador hace que la teoría de aproximación se centre en la forma geométrica de los polinomios y en buscar nuevas bases que permitan dibujar y manipular curvas de la forma más eficiente posible. Surgen así los polinomios de Bernstein, los cuales desempeñan un papel fundamental en la representación de las curvas de Bézier tal y como se mostrará a lo largo de este capítulo (ver Deuflhard y Hohmann (2003)).

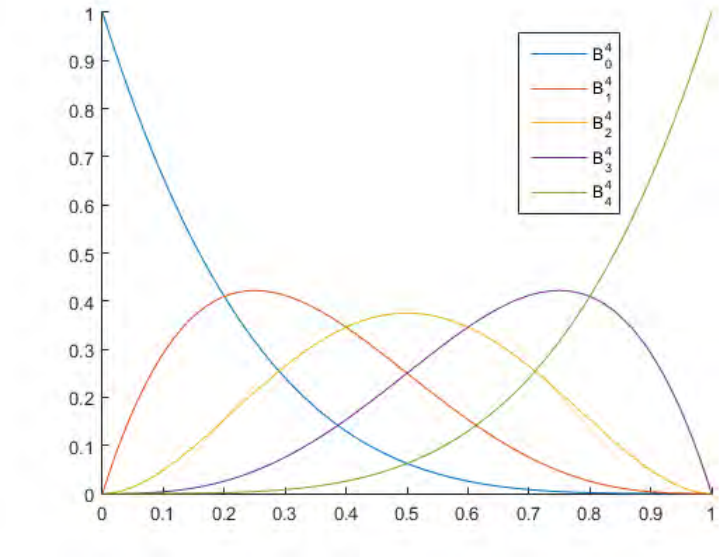


Figura 1.1: Polinomios de Bernstein para $n = 4$ definidos en el intervalo $[0, 1]$.

Definición 1.1. *Se define el* ***i-ésimo polinomio de Bernstein de grado*** **n** ***con respecto al intervalo*** $[\mathbf{0}, \mathbf{1}]$, $B_i^n(\lambda) \in \mathbb{P}_n$, *como*

$$B_i^n(\lambda) := \binom{n}{i}(1-\lambda)^{n-i}\lambda^i, \quad i = 0, \ldots, n. \tag{1.1.1}$$

En la Figura 1.1 se han dibujado los polinomios de Bernstein de grado 4.

Considerando la aplicación afín,

$$\begin{array}{rccl} \lambda: & [a,b] & \longrightarrow & [0,1] \\ & t & \longrightarrow & \lambda(t) = \dfrac{t-a}{b-a}, \end{array} \tag{1.1.2}$$

se define el **i-ésimo polinomio de Bernstein de grado n con respecto al intervalo** $[\mathbf{a}, \mathbf{b}]$, $a < b$, $a, b \in \mathbb{R}$, como

$$B_i^n(t; a, b) := B_i^n(\lambda(t)) = \binom{n}{i}\left(1 - \frac{t-a}{b-a}\right)^{n-i}\left(\frac{t-a}{b-a}\right)^i = \frac{1}{(b-a)^n}\binom{n}{i}(b-t)^{n-i}(t-a)^i,$$

para $i = 0, \ldots, n$.

Enunciaremos y probaremos los resultados correspondientes a los polinomios de Bernstein para $[0, 1]$ que, utilizando la aplicación (1.1.2), se generalizan a cualquier intervalo $[a, b]$, $a, b \in \mathbb{R}$, $a < b$.

Exponemos en el siguiente teorema las propiedades más significativas de estos polinomios.

Teorema 1.2. *Los polinomios de Bernstein $B_i^n(\lambda)$, $0 \leq i \leq n$, satisfacen:*

1. *$\lambda = 0$ es una raíz de $B_i^n(\lambda)$ con multiplicidad i.*
2. *$\lambda = 1$ es una raíz de $B_i^n(\lambda)$ con multiplicidad $n - i$.*
3. *Propiedad de simetría:*
$$B_i^n(\lambda) = B_{n-i}^n(1-\lambda), \quad i = 0, \ldots, n. \tag{1.1.3}$$
4. *$(1-\lambda)B_0^n(\lambda) = B_0^{n+1}(\lambda)$, y $\lambda B_n^n(\lambda) = B_{n+1}^{n+1}(\lambda)$.*
5. *Los polinomios de Bernstein $B_i^n(\lambda)$ son no negativos en $[0,1]$ y forman una partición de la unidad, esto es,*
$$B_i^n(\lambda) \geq 0, \quad \lambda \in [0,1],$$
$$\sum_{i=0}^{n} B_i^n(\lambda) = 1, \quad \lambda \in [0,1]. \tag{1.1.4}$$
6. *$B_i^n(\lambda)$ tiene exactamente un máximo en $[0,1]$ en el punto $\lambda = \dfrac{i}{n}$.*
7. *Los polinomios de Bernstein satisfacen la relación de recurrencia*
$$B_i^n(\lambda) = \lambda B_{i-1}^{n-1}(\lambda) + (1-\lambda)B_i^{n-1}(\lambda), \quad i = 1, \ldots, n, \quad \lambda \in [0,1]. \tag{1.1.5}$$
8. *Los polinomios de Bernstein forman una base $B = \{B_0^n(\lambda), \ldots, B_n^n(\lambda)\}$ de $\mathbb{P}_n$.*
9. *Las derivadas de los polinomios de Bernstein $B_i^n(\lambda)$ en el intervalo $[0,1]$ vienen dadas por*
$$\begin{aligned} \frac{d}{d\lambda}B_0^n(\lambda) &= -nB_0^{n-1}(\lambda), \\ \frac{d}{d\lambda}B_i^n(\lambda) &= n(B_{i-1}^{n-1}(\lambda) - B_i^{n-1}(\lambda)), \quad i = 1, \ldots, n-1, \\ \frac{d}{d\lambda}B_n^n(\lambda) &= nB_{n-1}^{n-1}(\lambda). \end{aligned} \tag{1.1.6}$$

Demostración.

1 y 2. Inmediatas por la Definición 1.1.

3. $B_{n-i}^n(1-\lambda) = \binom{n}{n-i}(1-(1-\lambda))^{n-(n-i)}(1-\lambda)^{n-i} = \binom{n}{i}(1-\lambda)^{n-i}\lambda^i = B_i^n(\lambda)$.

4. Utilizando la expresión (1.1.1), notamos que $B_0^n(\lambda) = (1-\lambda)^n$ y $B_n^n(\lambda) = \lambda^n$.

   Es claro entonces que
$$\begin{aligned} B_0^{n+1}(\lambda) &= (1-\lambda)^{n+1} = (1-\lambda)B_0^n(\lambda), \\ B_{n+1}^{n+1}(\lambda) &= \lambda^{n+1} = \lambda B_n^n(\lambda). \end{aligned}$$

5. Para $\lambda \in [0,1]$, $1-\lambda \geq 0$ y $\lambda \geq 0$. Por lo tanto,

$$B_i^n(\lambda) = \binom{n}{i}(1-\lambda)^{n-i}\lambda^i \geq 0, \quad i = 0, \ldots, n.$$

Por otro lado, gracias al teorema del binomio de Newton,

$$1 = ((1-\lambda)+\lambda)^n = \sum_{i=0}^{n}\binom{n}{i}(1-\lambda)^{n-i}\lambda^i = \sum_{i=0}^{n} B_i^n(\lambda), \quad \lambda \in [0,1].$$

y queda probada la segunda igualdad.

6. Analizamos por separado los casos $i = 0$, $i = n$, $0 < i < n$.

- Si $i = 0$,

$$B_0^n(\lambda) = (1-\lambda)^n \Rightarrow \frac{d}{d\lambda}B_0^n(\lambda) = -n(1-\lambda)^{n-1} \leq 0, \ \lambda \in [0,1].$$

Es decir, $B_0^n(\lambda)$ es estrictamente decreciente en $[0,1]$ y, por tanto, alcanza el máximo en $\lambda = 0$ (y el mínimo en $\lambda = 1$).

- Si $i = n$, $B_n^n(\lambda) = \lambda^n$ cuyo máximo en $[0,1]$ se alcanza en $\lambda = 1$ (y mínimo $\lambda = 0$), por lo que se verifica la propiedad.

- Si $0 < i < n$, como las funciones $B_i^n(\lambda)$ son $\mathcal{C}^\infty$ por tratarse de polinomios, derivamos respecto de λ e igualamos a cero para ver cuál es el candidato a extremo relativo,

$$\frac{d}{d\lambda}B_i^n(\lambda) = \binom{n}{i}(1-\lambda)^{n-1-i}\lambda^{i-1}(i-n\lambda) = 0 \Leftrightarrow \begin{cases} \lambda = 1 \Rightarrow B_i^n(1) = 0, \\ \lambda = 0 \Rightarrow B_i^n(0) = 0, \\ \lambda = \frac{i}{n} \Rightarrow B_i^n(\frac{i}{n}) > 0. \end{cases}$$

Por lo que en $\lambda = \dfrac{i}{n}$ se alcanza el máximo que buscábamos.

7. Gracias a las propiedades de los números combinatorios, se tiene

$$\begin{aligned} \lambda B_{i-1}^{n-1}(\lambda) + (1-\lambda)B_i^{n-1}(\lambda) &= \binom{n-1}{i-1}(1-\lambda)^{n-i}\lambda^i + \binom{n-1}{i}(1-\lambda)^{n-i}\lambda^i \\ &= \lambda^i(1-\lambda)^{n-i}\left[\binom{n-1}{i-1} + \binom{n-1}{i}\right] \\ &= \lambda^i(1-\lambda)^{n-i}\binom{n}{i} = B_i^n(\lambda). \end{aligned}$$

8. Debido a que en un espacio vectorial de dimensión $n+1$ un conjunto de $n+1$ vectores linealmente independientes es también sistema generador, bastará con probar que

los polinomios de Bernstein $B_i^n(\lambda)$ son linealmente independientes para demostrar el resultado.

Veamos pues que, para $b_0, \ldots, b_n \in \mathbb{R}$,

$$\sum_{i=0}^{n} b_i B_i^n(\lambda) = 0 \Leftrightarrow b_0 = \cdots = b_n = 0.$$

La implicación hacia la izquierda es trivial. Probemos la implicación hacia la derecha.

Para ello utilizamos el teorema del binomio de Newton, y reescribimos $B_i^n(\lambda)$ como

$$\begin{aligned} B_i^n(\lambda) &= \binom{n}{i}\lambda^i \sum_{j=0}^{n-i}\binom{n-i}{j}(-1)^j\lambda^j = \sum_{j=0}^{n-i}(-1)^j\binom{n-i}{j}\binom{n}{i}\lambda^{i+j} \\ &= \sum_{j=i}^{n}(-1)^{j-i}\binom{n}{i}\binom{n-i}{j-i}\lambda^j = \sum_{j=i}^{n}(-1)^{j-i}\binom{n}{j}\binom{j}{i}\lambda^j, \quad \lambda \in [0,1]. \end{aligned}$$

Si para todo $\lambda \in [0,1]$, se verifica

$$\begin{aligned} 0 &= b_0 B_0^n(\lambda) + \cdots + b_n B_n^n(\lambda) = b_0 \sum_{j=0}^{n}(-1)^j\binom{n}{j}\binom{j}{0}\lambda^j \\ &+ b_1 \sum_{j=1}^{n}(-1)^{j-1}\binom{n}{j}\binom{j}{1}\lambda^j + \cdots + b_n \sum_{j=n}^{n}(-1)^{j-n}\binom{n}{j}\binom{j}{n}\lambda^j \\ &= b_0 + \lambda\left(\sum_{i=0}^{1} b_i(-1)^{i+1}\binom{n}{1}\binom{1}{i}\right) + \cdots + \lambda^n\left(\sum_{i=0}^{n} b_i(-1)^{i+n}\binom{n}{n}\binom{n}{i}\right), \end{aligned}$$

necesariamente tiene que verificarse que

$$\begin{cases} b_0 = 0, \\ \sum_{i=0}^{1} b_i(-1)^{i+1}\binom{n}{1}\binom{1}{i} = 0, \\ \cdots \\ \sum_{i=0}^{n} b_i(-1)^{i+n}\binom{n}{n}\binom{n}{i} = 0. \end{cases}$$

Y llevando a cabo una sustitución progresiva comenzando por introducir el valor $b_0 = 0$ en el resto de ecuaciones, obtenemos que necesariamente $b_0 = \cdots = b_n = 0$ tal y como queríamos probar.

9. Inmediato derivando en la expresión (1.1.1),

$$\frac{d}{d\lambda}B_i^n(\lambda) = \binom{n}{i}\left[i(1-\lambda)^{n-i}\lambda^{i-1} - (n-i)(1-\lambda)^{n-1-i}\lambda^i\right],\ i = 0,\ldots,n,$$

y empleando la Propiedad 4.

□

Todos los resultados anteriores pueden generalizarse a polinomios de Bernstein $B_i^n(t;a,b)$ definidos en un intervalo $[a,b]$ cualquiera. En este caso, teniendo en cuenta (1.1.2), $B_i^n(t;a,b)$ alcanza su valor máximo en $t = a + \frac{i}{n}(b-a)$, como puede apreciarse en la Figura 1.2 para el caso $[a,b] = [5,9]$ y grado 6.

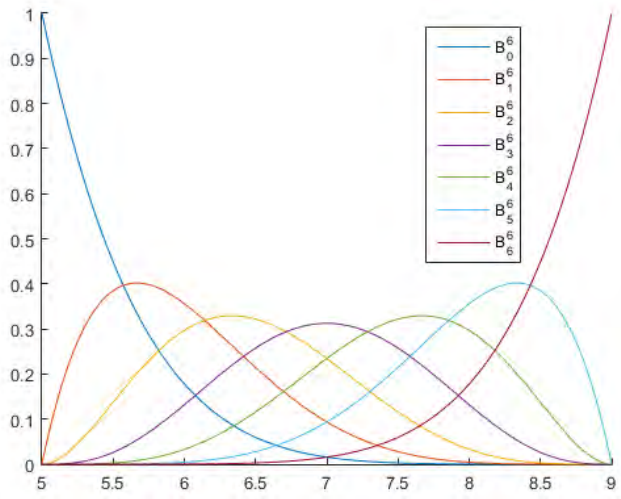


Figura 1.2: Polinomios de Bernstein para $n = 6$ definidos en el intervalo $[5,9]$.

Observación 1.3. *Gracias a la propiedad de simetría de los polinomios de Bernstein* (1.1.3)*, puede afirmarse que*

$$\begin{aligned}
B_i^n(t;a,b) &= B_i^n\left(\frac{t-a}{b-a}\right) = \binom{n}{i}\frac{1}{(a-b)^n}(t-b)^{n-i}(a-t)^i \\
&= \binom{n}{n-i}\frac{1}{(-1)^n(b-a)^n}(b-t)^{n-i}(-1)^{n-i}(t-a)^i(-1)^i \\
&= \binom{n}{n-i}\frac{1}{(a-b)^n}(t-b)^{n-i}(a-t)^i \\
&= \binom{n}{n-i}\left(\frac{t-b}{a-b}\right)^{n-i}\left(\frac{a-t}{a-b}\right)^i = B_{n-i}^n\left(\frac{t-b}{a-b}\right) = B_{n-i}^n(t;b,a),\ t\in[a,b].
\end{aligned}$$

Por tanto,

$$\sum_{i=0}^{n} b_i B_i^n(t;a,b) = \sum_{i=0}^{n} b_{n-i} B_i^n(t;b,a),\ t \in [a,b].$$

Finalmente vamos a mostrar el papel que juegan estos polinomios en una demostración constructiva del teorema de aproximación de una función continua en un intervalo acotado por polinomios. Antes de centrarnos en el teorema debido a Weierstrass en 1885 (ver Cheney (1982)), enunciamos y probamos un lema necesario en la posterior demostración del mismo.

Lema 1.4. *Se verifica que*

$$a) \quad \lambda = \sum_{i=0}^{n} \frac{i}{n} B_i^n(\lambda), \quad \lambda \in [0,1]. \tag{1.1.7}$$

$$b) \quad \sum_{i=0}^{n} \left(\frac{i}{n}\right)^2 B_i^n(\lambda) = \left(1 - \frac{1}{n}\right)\lambda^2 + \frac{\lambda}{n}, \quad \lambda \in [0,1]. \tag{1.1.8}$$

Demostración. a)

$$\begin{aligned} \sum_{i=0}^{n} \frac{i}{n} B_i^n(\lambda) &= \sum_{i=1}^{n} \frac{i}{n} \binom{n}{i} (1-\lambda)^{n-i} \lambda^i = \sum_{i=1}^{n} \binom{n-1}{i-1} (1-\lambda)^{n-1-(i-1)} \lambda^{i-1} \lambda \\ &= \lambda \sum_{j=0}^{n-1} \binom{n-1}{j} (1-\lambda)^{n-1-j} \lambda^j = \lambda \sum_{j=0}^{n-1} B_j^{n-1}(\lambda) = \lambda, \quad \lambda \in [0,1]. \end{aligned}$$

b) Utilizando de nuevo el binomio de Newton,

$$(p+q)^n = \sum_{k=0}^{n} \binom{n}{k} p^k q^{n-k}, \quad p, q \in \mathbb{R},$$

derivamos con respecto de p a ambos lados,

$$\frac{d}{dp}\left(\sum_{k=0}^{n} \binom{n}{k} p^k q^{n-k}\right) = \sum_{k=0}^{n} \binom{n}{k} k\, p^{k-1} q^{n-k} = n(p+q)^{n-1},$$

y obtenemos

$$(p+q)^{n-1} = \sum_{k=0}^{n} \binom{n}{k} \frac{k}{n}\, p^{k-1} q^{n-k}.$$

Derivamos esta nueva expresión,

$$\frac{d}{dp}\left(\sum_{k=0}^{n} \binom{n}{k} \frac{k}{n}\, p^{k-1} q^{n-k}\right) = \sum_{k=0}^{n} \binom{n}{k} \frac{k}{n} (k-1) p^{k-2} q^{n-k} = (n-1)(p+q)^{n-2},$$

y por tanto,

$$p^2\ (p+q)^{n-2} = \sum_{k=0}^{n} \binom{n}{k} \frac{k\,(k-1)}{n\,(n-1)}\, p^k q^{n-k}.$$

Tomando $p = \lambda$ y $q = 1-\lambda$,

$$\lambda^2 = \sum_{k=0}^{n} \binom{n}{k} \frac{k\,(k-1)}{n\,(n-1)}\, \lambda^k (1-\lambda)^{n-k}.$$

Finalmente, gracias a (1.1.7),

$$\begin{aligned}
\left(1-\frac{1}{n}\right)\lambda^2 + \frac{\lambda}{n} &= \sum_{k=0}^{n} \binom{n}{k} \frac{k(k-1)}{n(n-1)} \left(1-\frac{1}{n}\right) \lambda^k (1-\lambda)^{n-k} \\
&+ \frac{1}{n} \sum_{k=0}^{n} \binom{n}{k} \frac{k}{n}\, \lambda^k (1-\lambda)^{n-k} \\
&= \sum_{k=0}^{n} \binom{n}{k} \lambda^k (1-\lambda)^{n-k} \left(\frac{k\,(k-1)}{n^2} + \frac{k}{n^2}\right) \\
&= \sum_{k=0}^{n} \left(\frac{k}{n}\right)^2 \binom{n}{k} \lambda^k (1-\lambda)^{n-k}.
\end{aligned}$$

□

Teorema 1.5 (**Teorema de aproximación de Weierstrass**). *Sea $f : [a,b] \longrightarrow \mathbb{R}$ continua. Dado $\epsilon > 0$, existe un polinomio P tal que*

$$|f(t) - P(t)| \leq \epsilon,\ \forall t \in [a,b].$$

Demostración. Se puede suponer sin pérdida de generalidad, que $[a,b] = [0,1]$, pues cualquier intervalo $[a,b]$ puede transformarse en el $[0,1]$ mediante la transformación afín (1.1.2).

Consideramos la sucesión de polinomios $\{P_n\}_{n\in\mathbb{N}}$, donde para cada $n \in \mathbb{N}$,

$$P_n(\lambda) = \sum_{i=0}^{n} f\left(\frac{i}{n}\right) B_i^n(\lambda) = \sum_{i=0}^{n} \binom{n}{i} (1-\lambda)^{n-i} \lambda^i f\left(\frac{i}{n}\right), \quad \lambda \in [0,1].$$

Demostraremos que $\{P_n\}_{n\in\mathbb{N}}$ converge uniformemente a f en $[0,1]$. Gracias a la igualdad (1.1.4) del Teorema 1.2,

$$f(\lambda) - P_n(\lambda) = \sum_{i=0}^{n} \binom{n}{i} (1-\lambda)^{n-i} \lambda^i \left(f(\lambda) - f\left(\frac{i}{n}\right)\right), \quad \lambda \in [0,1].$$

Al ser f continua en un compacto, es uniformemente continua y acotada, y por tanto, dado $\epsilon > 0$, existe $\delta > 0$ tal que $|f(\lambda_1) - f(\lambda_2)| < \frac{\epsilon}{2}$, si $|\lambda_1 - \lambda_2| < \delta,\ \lambda_1, \lambda_2 \in [0,1]$, y además, existe $M > 0$ tal que $|f(\lambda)| < M$, $\forall \lambda \in [0,1]$.

Fijados $\lambda \in [0,1]$ y $n \in \mathbb{N}$, dividimos los índices i en dos subconjuntos A y B como sigue:

- $i \in A$ si $\left|\frac{i}{n} - \lambda\right| < \delta$. En este caso:

$$\left|\sum_{i\in A}\binom{n}{i}(1-\lambda)^{n-i}\lambda^i\left(f(\lambda) - f\left(\frac{i}{n}\right)\right)\right| \leq \frac{\epsilon}{2}\sum_{i\in A}\binom{n}{i}(1-\lambda)^{n-i}\lambda^i \leq \frac{\epsilon}{2}.$$

- $i \in B$ si $\left|\frac{i}{n} - \lambda\right| \geq \delta$, en cuyo caso

$$\begin{aligned}\left|\sum_{i\in B}\binom{n}{i}(1-\lambda)^{n-i}\lambda^i\left(f(\lambda) - f\left(\frac{i}{n}\right)\right)\right| &\leq 2M\sum_{i\in B}\binom{n}{i}(1-\lambda)^{n-i}\lambda^i \\ &\leq 2M\sum_{i\in B}\frac{\left(\frac{i}{n}-\lambda\right)^2}{\left(\frac{i}{n}-\lambda\right)^2}\binom{n}{i}(1-\lambda)^{n-i}\lambda^i.\end{aligned}$$

  Gracias a las ecuaciones (1.1.7) y (1.1.8) y a la igualdad (1.1.4) del Teorema 1.2,

$$\begin{aligned}2M\sum_{i\in B}\frac{\left(\frac{i}{n}-\lambda\right)^2}{\left(\frac{i}{n}-\lambda\right)^2}\binom{n}{i}(1-\lambda)^{n-i}\lambda^i &\leq 2M\sum_{i=0}^{n}\frac{\left(\frac{i}{n}-\lambda\right)^2}{\left(\frac{i}{n}-\lambda\right)^2}\binom{n}{i}(1-\lambda)^{n-i}\lambda^i \\ &\leq \frac{2M}{\delta^2}\frac{\lambda(1-\lambda)}{n} \leq \frac{M}{2\delta^2 n},\end{aligned}$$

  donde para la última desigualdad ha sido necesario utilizar que $\lambda(1-\lambda) \leq \frac{1}{4}$ para $\lambda \in [0,1]$.

Consideramos ahora n suficientemente grande como para que $\frac{M}{2\delta^2 n} \leq \frac{\epsilon}{2}$, digamos n_0. Finalmente $\forall n \geq n_0$,

$$\begin{aligned}|f(\lambda) - P_n(\lambda)| &= \left|\sum_{i=0}^{n}\binom{n}{i}(1-\lambda)^{n-i}\lambda^i\left(f(\lambda) - f\left(\frac{i}{n}\right)\right)\right| \\ &\leq \left|\sum_{i\in A}\binom{n}{i}(1-\lambda)^{n-i}\lambda^i\left(f(\lambda) - f\left(\frac{i}{n}\right)\right)\right| \\ &+ \left|\sum_{i\in B}\binom{n}{i}(1-\lambda)^{n-i}\lambda^i\left(f(\lambda) - f\left(\frac{i}{n}\right)\right)\right| \leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon\end{aligned}$$

quedando así el teorema demostrado. □

Capítulo 2

Representación de Bézier de un polinomio $\mathbf{P} \in \mathbb{P}_n^2$

Gracias a la Propiedad (8) del Teorema 1.2, sabemos que los polinomios de Bernstein forman una base tanto si están definidos en $[0,1]$ como si lo están en $[a,b]$ para $a < b,\ a,b \in \mathbb{R}$.

Definición 2.1. *Una **curva polinomial** (o **polinomio**) de **grado n** en $\mathbb{R}^2$ es una función* $\mathbf{P}$ *de la forma*

$$\begin{array}{rccl} \mathbf{P}: & \mathbb{R} & \longrightarrow & \mathbb{R}^2 \\ & t & \longrightarrow & \mathbf{P}(t) = \displaystyle\sum_{i=0}^{n} \mathbf{a}_i t^i, \quad \mathbf{a}_0, \ldots, \mathbf{a}_n \in \mathbb{R}^2,\ \mathbf{a}_n \neq 0. \end{array}$$

El espacio de polinomios de grado menor o igual que n en $\mathbb{R}^2$ se denota mediante $\mathbb{P}_n^2$.

Definición 2.2. *Dada una curva polinomial $\mathbf{P} \in \mathbb{P}_n^2$ se definen los **puntos de Bézier** de $\mathbf{P}$ o **puntos de control** de $\mathbf{P}$ como los coeficientes $\mathbf{b}_0, \ldots, \mathbf{b}_n \in \mathbb{R}^2$ de la representación de $\mathbf{P}$ en la base de polinomios de Bernstein en $[a,b]$,*

$$\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t; a, b), \quad t \in [a,b]. \tag{2.0.1}$$

A la representación de $\mathbf{P}$ en esta base se le conoce como **representación de Bézier** de $\mathbf{P}$, y al polígono que surge de unir mediante segmentos los puntos de Bézier que están consecutivos se le conoce como **polígono de Bézier** de $\mathbf{P}$.

Notación. *Aunque esta definición está expresada en términos del intervalo genérico $[a,b]$, en numerosas ocasiones se hablará de puntos de Bézier haciéndose referencia a la representación de $\mathbf{P}$ en la base $\{B_0^n(\lambda), \ldots, B_n^n(\lambda)\}$ donde cada $B_i^n(\lambda)$ está definido en $[0,1]$.*

Observación 2.3. *Dada una aplicación afín* $\phi : \mathbb{R}^2 \longrightarrow \mathbb{R}^2,\ \phi(\mathbf{u}) = A\mathbf{u}+\mathbf{v},\ \mathbf{v} \in \mathbb{R}^2,\ A \in \mathcal{M}_{2,2}$ *se verifica que las imágenes* $\phi(\mathbf{b}_i)$ *de los puntos de Bézier de un polinomio* $\mathbf{P} \in \mathbb{P}_n^2$ *definido en* $[0,1]$ *son los puntos de Bézier de* $\phi \circ \mathbf{P}$

$$\begin{aligned}\sum_{i=0}^{n} \phi\left(\mathbf{b}_i\right) B_i^n(\lambda) &= \sum_{i=0}^{n} \left(A\mathbf{b}_i + \mathbf{v}\right) B_i^n(\lambda) = \sum_{i=0}^{n} A\mathbf{b}_i B_i^n(\lambda) + \sum_{i=0}^{n} \mathbf{v} B_i^n(\lambda) \\ &= A\left(\sum_{i=0}^{n} \mathbf{b}_i B_i^n(\lambda)\right) + \mathbf{v} = \phi\left(\sum_{i=0}^{n} \mathbf{b}_i B_i^n(\lambda)\right) = \phi(\mathbf{P}(\lambda)),\quad \lambda \in [0,1].\end{aligned}$$

Notamos que para probar las igualdades anteriores, se ha utilizado la expresión (1.1.4).

Antes de continuar, ilustramos mediante la Figura 2.1, cómo influye en la estructura de la curva la ordenación de los nodos de Bézier en la definición del polígono. Para ello, se consideran los puntos

$$\begin{aligned}\mathbf{A} &= (-0{,}8, 2{,}55),\\ \mathbf{B} &= (-5{,}36, -1{,}77),\\ \mathbf{C} &= (0{,}72, -1{,}85),\\ \mathbf{D} &= (-1{,}56, -3{,}61),\end{aligned}$$

y para $\lambda \in [0,1]$ se consideran,

(a) $\displaystyle\sum_{i=0}^{3} \mathbf{b}_i B_i^n(\lambda) = \mathbf{A}B_0^3(\lambda) + \mathbf{B}B_1^3(\lambda) + \mathbf{C}B_2^3(\lambda) + \mathbf{D}B_3^3(\lambda),$

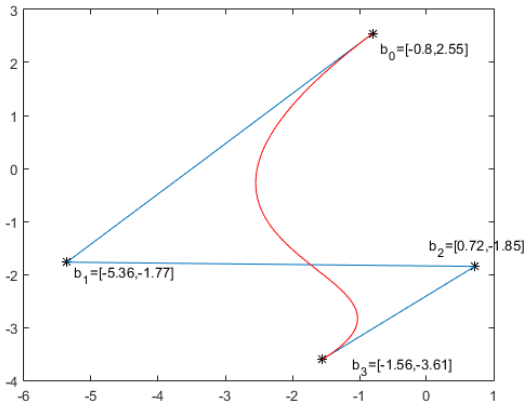
(b) $\displaystyle\sum_{i=0}^{3} \mathbf{b}_i B_i^n(\lambda) = \mathbf{D}B_0^3(\lambda) + \mathbf{A}B_1^3(\lambda) + \mathbf{B}B_2^3(\lambda) + \mathbf{C}B_3^3(\lambda),$

(c) $\displaystyle\sum_{i=0}^{3} \mathbf{b}_i B_i^n(\lambda) = \mathbf{A}B_0^3(\lambda) + \mathbf{D}B_1^3(\lambda) + \mathbf{B}B_2^3(\lambda) + \mathbf{C}B_3^3(\lambda),$

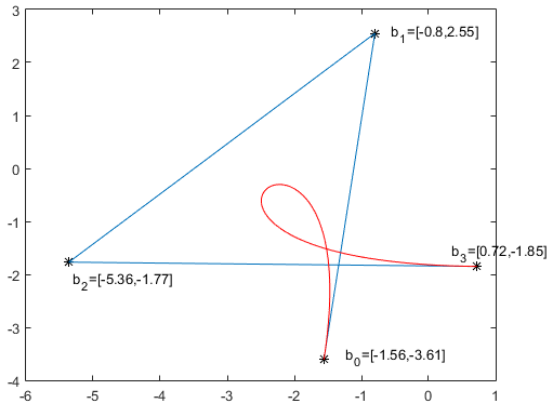
(d) $\displaystyle\sum_{i=0}^{3} \mathbf{b}_i B_i^n(\lambda) = \mathbf{A}B_0^3(\lambda) + \mathbf{B}B_1^3(\lambda) + \mathbf{D}B_2^3(\lambda) + \mathbf{C}B_3^3(\lambda),$

que muestran cuatro curvas distintas que pueden obtenerse al permutar dichos nodos. Pese a que existen veinticuatro reordenaciones posibles de los cuatro puntos del plano $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$, muchas de ellas se repiten debido a la Propiedad (1.1.3), como es por ejemplo el caso de $(\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3) = (\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ y $(\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3) = (\mathbf{D}, \mathbf{C}, \mathbf{B}, \mathbf{A})$.

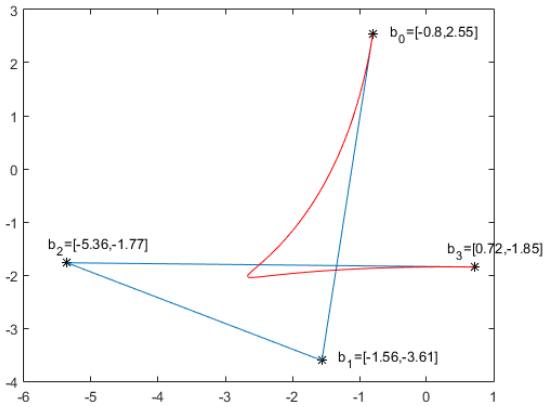
En todos los casos observamos que la curva pasa por el primer punto y por el último. Esto es consecuencia de las Propiedades 1 y 2 del Teorema 1.2. Aunque se justificará más adelante, se observa también en las cuatro figuras que el segmento que une los dos primeros nodos tiene la misma dirección que la tangente a la curva en el primer nodo, y que el segmento que une los dos últimos nodos tiene la misma dirección que la tangente a la curva en el último nodo.



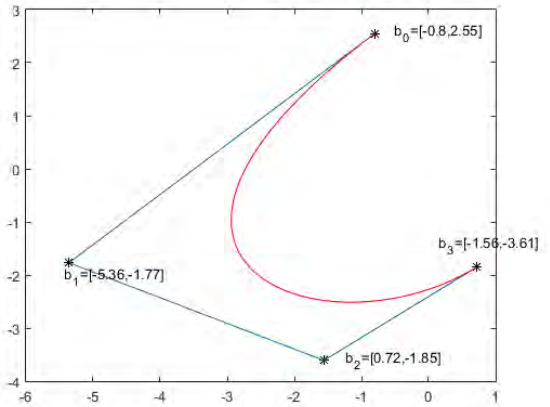
(a) $(\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3) = (\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$.



(b) $(\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3) = (\mathbf{D}, \mathbf{A}, \mathbf{B}, \mathbf{C})$.



(c) $(\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3) = (\mathbf{A}, \mathbf{D}, \mathbf{B}, \mathbf{C})$.



(d) $(\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3) = (\mathbf{A}, \mathbf{B}, \mathbf{D}, \mathbf{C})$.

Figura 2.1: Ejemplo de polinomios de Bézier cúbicos que utilizan los mismos nodos pero en distinto orden.

2.1. Algoritmo de De Casteljau

De manera análoga a la tabla de diferencias divididas utilizada en la construcción de la forma de Newton del polinomio interpolador de Lagrange, existe un algoritmo para la construcción de la representación de Bézier de un polinomio $\mathbf{P}$ de grado menor o igual que n en el plano definido por un polígono $\mathbf{b}_0, \ldots, \mathbf{b}_n$. Este construye el polinomio utilizando combinaciones continuas convexas de puntos de Bézier y la información que nos aportan las derivadas.

Introducimos a continuación la definición de polinomios parciales de $\mathbf{P}$.

Definición 2.4. *Sea* $\mathbf{P}(\lambda) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(\lambda),\ \lambda \in [0,1]$. *Se definen los* ***polinomios parciales*** $\mathbf{b}_i^k(\lambda) \in \mathbb{P}_k^2$ *de* $\mathbf{P}$ *como*

$$\mathbf{b}_i^k(\lambda) := \sum_{j=0}^{k} \mathbf{b}_{i+j} B_j^k(\lambda) = \sum_{j=i}^{i+k} \mathbf{b}_j B_{j-i}^k(\lambda), \quad i = 0, \ldots, n-k, \quad \lambda \in [0,1].$$

Para un polinomio $\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t; a, b),\ t \in [a,b]$, *los polinomios parciales* $\mathbf{b}_i^k(t) \in \mathbb{P}_k^2$ *se definen de manera similar, utilizando el cambio de variable* (1.1.2),

$$\mathbf{b}_i^k(t; a, b) := \mathbf{b}_i^k(\lambda(t)) = \sum_{j=0}^{k} \mathbf{b}_{i+j} B_j^k(t; a, b) = \sum_{j=i}^{i+k} \mathbf{b}_j B_{j-i}^k(t; a, b), \ \ i = 0, \ldots, n-k, \ \ t \in [a,b].$$

De este modo, el polinomio parcial $\mathbf{b}_i^k \in \mathbb{P}_k^2$ es justamente el polinomio de Bézier definido mediante los puntos de control $\mathbf{b}_i, \ldots, \mathbf{b}_{i+k}$.

En particular, $\mathbf{P}(t) = \mathbf{b}_0^n(t)$ y $\mathbf{b}_i^0(t) = \mathbf{b}_i,\ t \in [a,b]$, y para los extremos se verifica $\mathbf{b}_i^k(a) = \mathbf{b}_i$ y $\mathbf{b}_i^k(b) = \mathbf{b}_{i+k}$.

Expresamos a continuación el lema que va a ser fundamental en el desarrollo del algoritmo de De Casteljau:

Lema 2.5. *Los polinomios parciales* $\mathbf{b}_i^k(t; a, b)$ *de* $\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t; a, b)$ *satisfacen la relación de recurrencia*

$$\mathbf{b}_i^k(t; a, b) = (1 - \lambda(t))\mathbf{b}_i^{k-1}(t; a, b) + \lambda(t)\mathbf{b}_{i+1}^{k-1}(t; a, b), \qquad k = 0, \ldots, n; \quad i = 0, \ldots, n-k. \tag{2.1.1}$$

con $\lambda(t)$ *dado por* (1.1.2), $t \in [a,b]$.

Demostración. Basta insertar la relación de recurrencia (1.1.5) en la definición de los

polinomios parciales $\mathbf{b}_i^k(t)$ para obtener

$$\begin{aligned}
\mathbf{b}_i^k(t;a,b) &= \sum_{j=0}^{k} \mathbf{b}_{i+j} B_j^k(t;a,b) = \mathbf{b}_i B_0^k(t;a,b) + \mathbf{b}_{i+k} B_k^k(t;a,b) + \sum_{j=1}^{k-1} \mathbf{b}_{i+j} B_j^k(t;a,b) \\
&= \mathbf{b}_i(1-\lambda(t))B_0^{k-1}(t;a,b) + \mathbf{b}_{i+k}\lambda(t)B_{k-1}^{k-1}(t;a,b) \\
&+ \sum_{j=1}^{k-1} \mathbf{b}_{i+j}\Big((1-\lambda(t))B_j^{k-1}(t;a,b) + \lambda(t)B_{j-1}^{k-1}(t;a,b)\Big) \\
&= \sum_{j=0}^{k-1} \mathbf{b}_{i+j}(1-\lambda(t))B_j^{k-1}(t;a,b) + \sum_{j=1}^{k} \mathbf{b}_{i+j}\lambda(t)B_{j-1}^{k-1}(t;a,b) \\
&= (1-\lambda(t))\mathbf{b}_i^{k-1}(t;a,b) + \lambda(t)\mathbf{b}_{i+1}^{k-1}(t;a,b), \quad t \in [a,b].
\end{aligned}$$

□

Gracias a que $\mathbf{b}_i^0(t) = \mathbf{b}_i,\ i = 0,\ldots,n,\ t \in [a,b]$, podemos ir construyendo $\mathbf{P}(t) = \mathbf{b}_0^n(t)$ mediante combinaciones continuas convexas a partir de los puntos de Bézier, tal y como se expone en el Lema 2.5. Este proceso recursivo de construcción de $\mathbf{P}(t) = \mathbf{b}_0^n$ se conoce con el nombre de **algoritmo de De Casteljau**, y aparece esquematizado en la Figura 2.2.

$$\begin{array}{lllllllll}
b_n & = & b_n^0 & & & & & & \\
 & & & \searrow & & & & & \\
b_{n-1} & = & b_{n-1}^0 & \rightarrow & b_{n-1}^1 & & & & \\
 & \vdots & & & & \ddots & & & \\
b_1 & = & b_1^0 & \rightarrow & \cdots & \rightarrow & b_1^{n-1} & & \\
 & & & \searrow & & & & \searrow & \\
b_0 & = & b_0^0 & \rightarrow & \cdots & \rightarrow & b_0^{n-1} & \rightarrow & b_0^n
\end{array}$$

Figura 2.2: Algoritmo de De Casteljau esquematizado.

A simple vista puede parecer que el algoritmo de De Casteljau tan solo utiliza los distintos $\mathbf{b}_i$ para construir $\mathbf{P}$, sin rastro alguno de las derivadas de $\mathbf{P}$. Sin embargo, la siguiente sección muestra cómo las derivadas k-ésimas $\mathbf{P}^{(k)},\ k \in \mathbb{N}$, están ocultas tras los polinomios auxiliares $\mathbf{b}_i^k,\ i = 0,\ldots,n-k$.

2.2. Derivación de polinomios representados en la forma de Bézier

En la sección anterior se comentó que como consecuencia del Teorema 1.2 el polígono y la curva de Bézier comienzan y acaban ambos en los mismos puntos. Parece intuitivo

pensar que las tangentes en los nodos $\mathbf{b}_0, \mathbf{b}_n$ coinciden con las rectas más próximas correspondientes al polígono de Bézier. Para ver si esta suposición es cierta o es una mera ilusión óptica, procedemos a analizar las derivadas de las curvas representadas en la forma (2.0.1). Comenzamos analizando las derivadas de los polinomios representados en la forma de Bézier definidos en el intervalo $[0, 1]$.

Teorema 2.6. *Sea* $\mathbf{P}(\lambda) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(\lambda)$ *la representación de Bézier con respecto a* $[0, 1]$ *de un polinomio.*

La derivada k-ésima de $\mathbf{P}$ *satisface*

$$\mathbf{P}^{(k)}(\lambda) = \frac{n!}{(n-k)!} \sum_{i=0}^{n-k} \triangle^k \mathbf{b}_i B_i^{n-k}(\lambda), \quad \lambda \in [0, 1], \tag{2.2.1}$$

donde $\triangle$ *denota el operador de diferencias progresivas, definido por*

$$\begin{aligned} \triangle \mathbf{b}_i &= \mathbf{b}_{i+1} - \mathbf{b}_i, \\ \triangle^k \mathbf{b}_i &= \triangle^{k-1} \mathbf{b}_{i+1} - \triangle^{k-1} \mathbf{b}_i, \quad k > 1, \quad i = 0, \ldots, n-k. \end{aligned}$$

Demostración. Razonamos mediante inducción sobre k.

- Para $k = 1$, utilizando las ecuaciones (1.1.6) tenemos

$$\begin{aligned} \mathbf{P}'(\lambda) &= -n\mathbf{b}_0 B_0^{n-1}(\lambda) + \sum_{i=0}^{n-1} n\mathbf{b}_i \left(B_{i-1}^{n-1}(\lambda) - B_i^{n-1}(\lambda) \right) + n\mathbf{b}_n B_{n-1}^{n-1}(\lambda) \\ &= n \Bigg[\mathbf{b}_n B_{n-1}^{n-1}(\lambda) - \mathbf{b}_0 B_0^{n-1}(\lambda) \\ &+ \mathbf{b}_1 B_0^{n-1}(\lambda) + \sum_{i=1}^{n-2} \left(\mathbf{b}_{i+1} - \mathbf{b}_i \right) B_i^{n-1}(\lambda) - \mathbf{b}_{n-1} B_{n-1}^{n-1}(\lambda) \Bigg] \\ &= n \sum_{i=0}^{n-1} (\mathbf{b}_{i+1} - \mathbf{b}_i) B_i^{n-1}(\lambda) \\ &= n \sum_{i=0}^{n-1} \triangle \mathbf{b}_i B_i^{n-1}(\lambda). \end{aligned}$$

- Supongamos que $\mathbf{P}^{(k)}(\lambda) = \frac{n!}{(n-k)!} \sum_{i=0}^{n-k} \triangle^k \mathbf{b}_i B_i^{n-k}(\lambda)$, y veamos que

$$\mathbf{P}^{(k+1)}(\lambda) = \frac{n!}{(n-k-1)!} \sum_{i=0}^{n-k-1} \triangle^{k+1} \mathbf{b}_i B_i^{n-k-1}(\lambda).$$

De nuevo, utilizando las ecuaciones (1.1.6) pero esta vez derivando $\mathbf{P}^{(k)}$,

$$\begin{aligned}
\mathbf{P}^{(k+1)}(\lambda) &= \frac{n!}{(n-k)!}\sum_{i=0}^{n-k}\triangle^k\mathbf{b}_i\frac{d}{d\lambda}B_i^{n-k}(\lambda)\\
&= \frac{n!}{(n-k)!}\Bigg[-(n-k)\triangle^k\mathbf{b}_0B_0^{n-k-1}(\lambda)+(n-k)\triangle^k\mathbf{b}_{n-k}B_{n-k-1}^{n-k-1}(\lambda)\\
&+ (n-k)\sum_{i=0}^{n-k-1}\triangle^k\mathbf{b}_i(B_{i-1}^{n-k-1}(\lambda)-B_i^{n-k-1}(\lambda))\Bigg]\\
&= \frac{n!}{(n-k-1)!}\Bigg[-\triangle^k\mathbf{b}_0B_0^{n-k-1}(\lambda)+\triangle^k\mathbf{b}_{n-k}B_{n-k-1}^{n-k-1}(\lambda)\\
&+ \sum_{i=0}^{n-k-1}\triangle^k\mathbf{b}_i\Big(B_{i-1}^{n-k-1}(\lambda)-B_i^{n-k-1}(\lambda)\Big)\Bigg]\\
&= \frac{n!}{(n-k-1)!}\Bigg[\sum_{i=0}^{n-k-1}\Big(\triangle^k\mathbf{b}_{i+1}-\triangle^k\mathbf{b}_i\Big)B_i^{n-k-1}(\lambda)\Bigg]\\
&= \frac{n!}{(n-k-1)!}\sum_{i=0}^{n-k-1}\triangle^{k+1}\mathbf{b}_iB_i^{n-k-1}(\lambda).
\end{aligned}$$

□

Observación 2.7. *Para extender el Teorema 2.6 al caso de polinomios de Bézier representados en un intervalo cualquiera* $[a,b]$, $a,b\in\mathbb{R}$, $a<b$, *basta con derivar la función* λ *definida en* (1.1.2), $\lambda'(t)=\frac{1}{b-a}$, *y emplear la regla de la cadena para obtener finalmente,*

$$\mathbf{P}^{(k)}(t;a,b)=\frac{n!}{(n-k)!}\sum_{i=0}^{n-k}\triangle^k\mathbf{b}_iB_i^{n-k}(t;a,b)\left(\frac{1}{b-a}\right)^k,\quad t\in[a,b].$$

Corolario 2.8. *Sea* $\mathbf{P}$ *una curva de Bézier definida en* $[0,1]$. *Entonces para* $\lambda=0,1$,

$$\begin{aligned}
\mathbf{P}^{(k)}(0) &= \frac{n!}{(n-k)!}\triangle^k\mathbf{b}_0,\\
\mathbf{P}^{(k)}(1) &= \frac{n!}{(n-k)!}\triangle^k\mathbf{b}_{n-k}.
\end{aligned}$$

En particular,

$$\begin{array}{lllllll}
\mathbf{P}(0) &=& \mathbf{b}_0, & & \mathbf{P}(1) &=& \mathbf{b}_n,\\
\mathbf{P}'(0) &=& n(\mathbf{b}_1-\mathbf{b}_0), & & \mathbf{P}'(1) &=& n(\mathbf{b}_n-\mathbf{b}_{n-1}),\\
\mathbf{P}''(0) &=& n(n-1)(\mathbf{b}_2-2\mathbf{b}_1+\mathbf{b}_0), & & \mathbf{P}''(1) &=& n(n-1)(\mathbf{b}_n-2\mathbf{b}_{n-1}+\mathbf{b}_{n-2}).
\end{array}$$

Demostración. Basta con tener en cuenta que $B_i^{n-k}(0)=\delta_{0,i}$ y $B_i^{n-k}(1)=\delta_{n-k,i}$ y el resultado es inmediato gracias al Teorema 2.6. □

El corolario anterior confirma las sospechas que habíamos expuesto con anterioridad. En efecto, en los nodos extremos la tangente a la curva coincide con la pendiente de la recta que une los primeros dos puntos (respectivamente los dos últimos) del polígono de Bézier.

Es especialmente significativo el hecho de que en un nodo extremo la curva está determinada, hasta la derivada k-ésima, por los k puntos de Bézier más próximos, tal y como puede verse en el Corolario 2.8. Esta propiedad será crucial más adelante, en el Capítulo 4, cuando se trate de unir varios segmentos de curva en un único spline regular.

El teorema siguiente afirma que la curva polinómica $\mathbf{P}$ está siempre contenida en la envolvente convexa de sus puntos de Bézier.

Teorema 2.9. *La imagen $\mathbf{P}([a,b])$ de un polinomio $\mathbf{P} \in \mathbb{P}_n^2$ en la representación de Bézier $\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t;a,b)$ con respecto de $[a,b]$ está contenida en la envolvente convexa de los puntos de Bézier $\mathbf{b}_i$, es decir,*

$$\mathbf{P}(t) \in \mathbf{co}(\{\mathbf{b}_0, \ldots, \mathbf{b}_n\}), \quad t \in [a,b].$$

Demostración. Recordamos que la envolvente convexa $\mathbf{co}(A)$ de $A \subset \mathbb{R}^2$ es el conjunto de todas las combinaciones convexas de puntos de A, esto es,

$$\mathbf{co}(A) = \left\{ \sum_{i=1}^{m} \lambda_i x_i,\ m \in \mathbb{N}\ ,\ x_i \in A,\ \lambda_i \geq 0\ ,\ \sum_{i=1}^{m} \lambda_i = 1 \right\}.$$

En $[a,b]$, los polinomios de Bernstein forman una partición no negativa de la unidad, esto es $B_i^n(t;a,b) \geq 0$ para $t \in [a,b]$ y $\sum_{i=0}^{n} B_i^n(t;a,b) = 1$. Entonces

$$\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t;a,b)$$

es una combinación convexa de los puntos de Bézier $\mathbf{b}_0, \ldots, \mathbf{b}_n$. □

Teorema 2.10. *Sea $\mathbf{P}(\lambda) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(\lambda)$ un polinomio en la representación de Bézier en $[0,1]$. Entonces las derivadas $\mathbf{P}^{(k)}$ para $k = 0, \ldots, n$ pueden construirse a partir de los polinomios parciales $\mathbf{b}_i^k$ mediante la relación*

$$\mathbf{P}^{(k)}(\lambda) = \frac{n!}{(n-k)!} \triangle^k \mathbf{b}_0^{n-k}(\lambda), \tag{2.2.2}$$

donde $\triangle \mathbf{b}_i^k = \mathbf{b}_{i+1}^k - \mathbf{b}_i^k$.

Demostración. En virtud del Teorema 2.6 sabemos que $\mathbf{P}^{(k)}(\lambda) = \dfrac{n!}{(n-k)!}\displaystyle\sum_{i=0}^{n-k} \triangle^k \mathbf{b}_i B_i^{n-k}(\lambda)$.

Por la linealidad del operador $\triangle^k$,

$$\sum_{j=0}^{n-k} \triangle^k \mathbf{b}_j B_j^{n-k}(\lambda) = \triangle^k \sum_{j=0}^{n-k} \mathbf{b}_j B_j^{n-k}(\lambda) = \triangle^k \mathbf{b}_0^{n-k}(\lambda),$$

lo que concluye la demostración. □

Como consecuencia del Teorema 2.10, sabemos que la derivada k-ésima de $\mathbf{P}$ se obtiene a partir de la columna $(n-k)$-ésima del algoritmo de De Casteljau.

En particular, para $\lambda \in [0,1]$,

$$\begin{aligned} \mathbf{P}(\lambda) &= \mathbf{b}_0^n(\lambda), \\ \mathbf{P}'(\lambda) &= n(\mathbf{b}_1^{n-1}(\lambda) - \mathbf{b}_0^{n-1}(\lambda)), \\ \mathbf{P}''(\lambda) &= n(n-1)(\mathbf{b}_2^{n-2}(\lambda) - 2\mathbf{b}_1^{n-2}(\lambda) + \mathbf{b}_0^{n-2}(\lambda)). \end{aligned}$$

Observación 2.11. *La expresión* (2.2.2)*, puede extenderse al caso de polinomios de Bézier definidos en un intervalo cualquiera* $[a,b]$, $a,b \in \mathbb{R}, a < b$*, tal y como se hizo en la Observación 2.7,*

$$\mathbf{P}^{(k)}(t;a,b) = \frac{n!}{(n-k)!} \triangle^k \mathbf{b}_0^{n-k}(t;a,b) \left(\frac{1}{b-a}\right)^k, \quad t \in [a,b].$$

Capítulo 3

Operaciones con curvas de Bézier

En este capítulo vamos a estudiar las operaciones más habituales que se realizan con curvas de Bézier. Comenzamos con la **subdivisión**, que permite definir parte de la traza de una curva de Bézier mediante otra curva de Bézier, seguimos con la **elevación del grado**, que dada una curva de Bézier de grado n permite definir una curva de Bézier de grado $n+1$ cuyas trazas coincidan, y finalizando con la **reducción del grado**, que dada una curva de Bézier de grado n establece los n nodos que definen la curva de Bézier de grado $n-1$ que mejor aproxima a la inicial, todas ellas definiendo los nuevos nodos de Bézier a partir de los iniciales.

3.1. Subdivisión

Si bien es cierto que la manera más sencilla de trabajar con curvas de Bézier es definiéndolas en $[0,1]$, mencionamos en la Sección 1.1 que no tiene por qué ser necesariamente esta su definición (ver Farin (1990)). Supongamos que conocemos los puntos de control $\mathbf{b}_0, \ldots, \mathbf{b}_n$ que definen una curva

$$\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t), \quad t \in [0,1], \tag{3.1.1}$$

y que queremos hallar $\mathbf{c}_0, \ldots, \mathbf{c}_n$ que definan la parte de $\mathbf{P}$, que corresponde a la variación del parámetro únicamente en $[0,c],\ 0 < c < 1$.

Notación. *En los capítulos anteriores, se empleaba λ para denotar parámetros definidos en $[0,1]$ y t para denotar parámetros definidos en $[a,b],\ a < b,\ a,b \in \mathbb{R}$ cualquiera. Durante esta sección van a introducirse numerosos cambios de variable, por lo que se introduce una nueva notación, pasando a utilizar t o s en lugar de λ cuando el parámetro varía en $[0,1]$. En las expresiones (3.1.1) y (3.1.2), por ejemplo, se están empleando t y s en lugar de λ para denotar parámetros variando en $[0,1]$.*

Para simplificar la notación introducimos un nuevo parámetro $s = \frac{t}{c}$, de tal manera que cuando t varía entre $[0, c]$, entonces $s \in [0, 1]$ y definimos,

$$\mathbf{Q}(s) = \sum_{i=0}^{n} \mathbf{c}_i B_i^n(s), \quad s \in [0, 1]. \tag{3.1.2}$$

Nuestro objetivo a continuación será hallar los valores de los puntos de control desconocidos $\mathbf{c}_j,\ j = 0, \ldots, n$, para que $\mathbf{Q}(\frac{t}{c}) = \mathbf{P}(t),\ \text{con } t \in [0, c]$, lo que se conoce con el nombre de **algoritmo de subdivisión**.

Para ello, comenzamos considerando las derivadas de $\mathbf{P}$ y $\mathbf{Q}$ en el punto $t = s = 0$. Debido a que la traza de la curva $\mathbf{Q}$ está contenida en la traza de la curva $\mathbf{P}$, todas las derivadas de ambas en $s = t = 0$ han de coincidir, es decir,

$$\mathbf{P}^{(r)}(0) = \mathbf{Q}^{(r)}(0), \quad r = 0, \ldots, n. \tag{3.1.3}$$

Consideramos, para $0 \leq r \leq n$, los polinomios de grado menor o igual que r,

$$\mathbf{b}_0^r(t) = \sum_{i=0}^{r} \mathbf{b}_i B_i^r(t), \quad \mathbf{c}_0^r(s) = \sum_{i=0}^{r} \mathbf{c}_i B_i^r(s), \quad s, t \in [0, 1].$$

Debido a que los polígonos de $\mathbf{b}_0^r(t)$ y $\mathbf{c}_0^r(s)$ corresponden a subpolígonos de $\mathbf{P}$ y $\mathbf{Q}$ respectivamente, gracias a (3.1.3) sabemos que coinciden en todas las derivadas hasta orden r en $s = t = 0$, y por tanto son idénticos, es decir,

$$\mathbf{b}_0^r(t) = \mathbf{c}_0^r(s), \quad s = \frac{t}{c}, \quad t \in [0, c].$$

Tomando en particular $t = c$ y, correspondientemente $s = 1$,

$$\mathbf{b}_0^r(c) = \mathbf{c}_0^r(1),$$

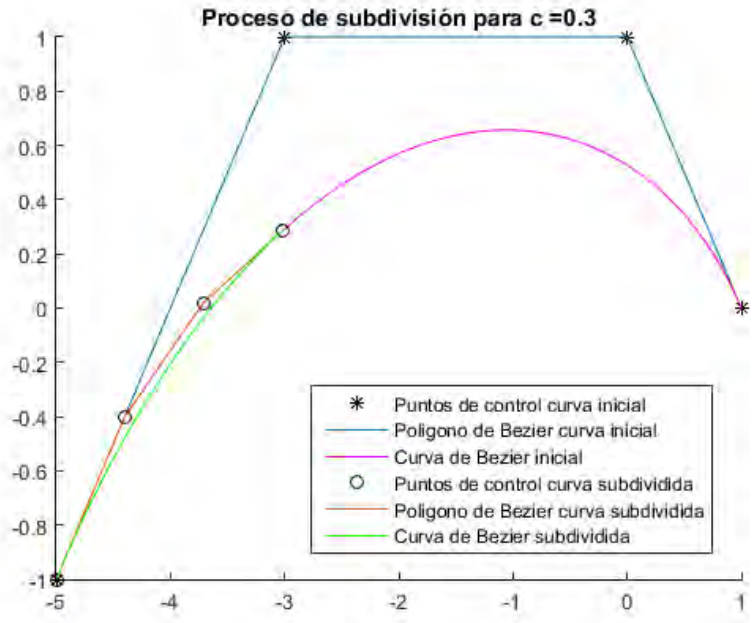
y debido a que $\mathbf{c}_0^r(1) = \sum_{i=0}^{r} \mathbf{c}_i B_i^r(1) = \mathbf{c}_r$, hemos encontrado la manera de calcular el parámetro $\mathbf{c}_r$ a partir de $\mathbf{b}_0, \ldots, \mathbf{b}_r$ tal y como queríamos. Surge así la denominada *fórmula de subdivisión para curvas de Bézier*

$$\mathbf{c}_r = \mathbf{b}_0^r(c), \quad r = 0, \ldots, n. \tag{3.1.4}$$

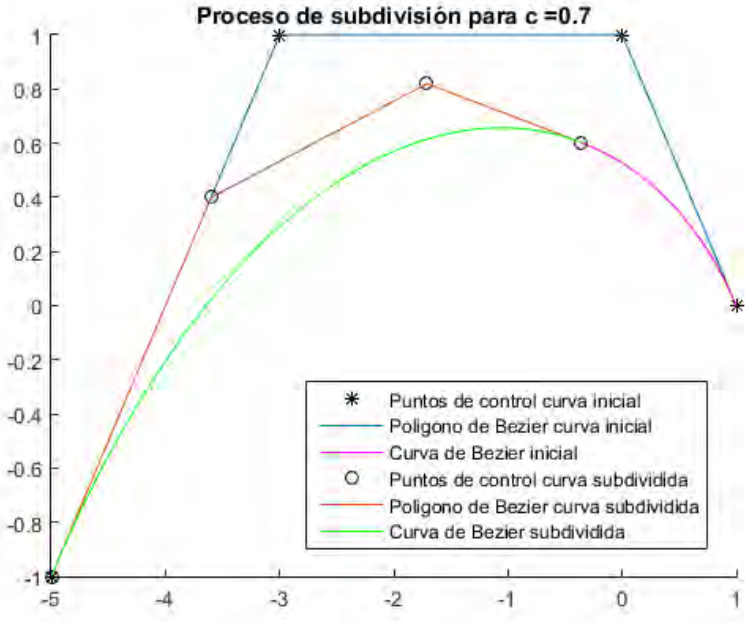
De este modo, podemos utilizar el algoritmo de De Casteljau para construir recursivamente los polinomios parciales $\mathbf{b}_0^j,\ j = 0, \ldots, n$, y evaluarlos en el punto c, para obtener los puntos de control de la curva de Bézier correspondiente al intervalo $[0, c]$.

En la Figura 3.1 se muestra el resultado de la subdivisión de una curva de Bézier, para dos valores distintos del parámetro c. Se ha incluido junto con el polígono de la curva original, el polígono correspondiente a la subdivisión.

Observación 3.1. *Gracias a la propiedad de simetría* (1.1.3) *los puntos de control correspondientes a* $[c, 1]$ *se pueden calcular en función de los polinomios parciales* $\mathbf{b}_{n-j}^j$.



(a) Subdivisión para $c = 0{,}3$



(b) Subdivisión para $c = 0{,}7$

Figura 3.1: Ejemplo que muestra el resultado de la subdivisión de una curva de Bézier para dos valores distintos de c. En azul y rosa el polígono y la curva de Bézier originales, en rojo y verde el polígono y la curva de Bézier definidos en (3.1.2), resultado del proceso de subdivisión.

3.2. Elevación del grado

Supongamos ahora que estamos empleando $\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t),\ t \in [0,1]$, curva de Bézier de grado n, para aproximar un dibujo o diseño. Imaginemos que, por más que modificamos el polígono, la curva no se ajusta lo suficiente a nuestro dibujo. Elevar el grado de la curva sin modificar mucho su forma permitiría un mayor ajuste al dibujo pero, sin desperdiciar el trabajo anterior ya realizado para determinar la curva de grado n, $\mathbf{P}$.

Por tanto, nuestro problema se reduce a buscar $n+2$ puntos de control

$$\mathbf{b}_0^{(1)}, \ldots, \mathbf{b}_{n+1}^{(1)},$$

que describan la curva $\mathbf{P}$, que a su vez tiene por puntos de control

$$\mathbf{b}_0, \ldots, \mathbf{b}_n.$$

Es decir, los nuevos vértices $\mathbf{b}_0^{(1)}, \ldots, \mathbf{b}_{n+1}^{(1)}$ deben ser tales que

$$\mathbf{P}(t) = \sum_{j=0}^{n} \mathbf{b}_j B_j^n(t) = \sum_{j=0}^{n+1} \mathbf{b}_j^{(1)} B_j^{n+1}(t), \quad t \in [0,1].$$

Operando, puede observarse que,

$$\begin{aligned}
\mathbf{P}(t) &= (1-t)\mathbf{P}(t) + t\mathbf{P}(t) \\
&= \sum_{i=0}^{n} \mathbf{b}_i \binom{n}{i} t^i (1-t)^{n+1-i} + \sum_{i=0}^{n} \mathbf{b}_i \binom{n}{i} t^{i+1} (1-t)^{n-i} \\
&= \sum_{i=0}^{n} \mathbf{b}_i \binom{n+1}{i} \frac{n+1-i}{n+1} t^i (1-t)^{n+1-i} + \sum_{i=0}^{n} \mathbf{b}_i \binom{n+1}{i+1} \frac{i+1}{n+1} t^{i+1} (1-t)^{n-i} \\
&= \sum_{i=0}^{n} \frac{n+1-i}{n+1} \mathbf{b}_i B_i^{n+1}(t) + \sum_{i=0}^{n} \frac{i+1}{n+1} \mathbf{b}_i B_{i+1}^{n+1}(t) \\
&= \sum_{i=0}^{n} \frac{n+1-i}{n+1} \mathbf{b}_i B_i^{n+1}(t) + \sum_{i=1}^{n+1} \frac{i}{n+1} \mathbf{b}_{i-1} B_i^{n+1}(t), \quad t \in [0,1]
\end{aligned}$$

Extendiendo el límite del primer sumatorio hasta $i = n+1$ y el del segundo sumatorio hasta $i = 0$ (pues los términos añadidos son nulos), puede concluirse que

$$\mathbf{P}(t) = \sum_{i=0}^{n+1} \frac{n+1-i}{n+1} \mathbf{b}_i B_i^{n+1}(t) + \sum_{i=0}^{n+1} \frac{i}{n+1} \mathbf{b}_{i-1} B_i^{n+1}(t),\ t \in [0,1].$$

Combinando ambas sumas y comparando los coeficientes, obtenemos la expresión deseada

$$\mathbf{b}_i^{(1)} = \frac{i}{n+1} \mathbf{b}_{i-1} + \left(1 - \frac{i}{n+1}\right) \mathbf{b}_i, \quad i = 0, \ldots, n+1. \tag{3.2.1}$$

Por lo tanto, los nuevos vértices del polígono de grado $n+1$ se obtienen a partir del polígono de grado n mediante interpolación lineal a trozos en los valores del parámetro $\frac{i}{n+1}$. Más adelante veremos cómo el polígono resultante está más próximo a la curva $\mathbf{P}$ que el original.

Observación 3.2. *El procedimiento anterior tiene importantes aplicaciones en el diseño de superficies, pues para numerosos algoritmos que producen superficies a partir de curvas es necesario que todas las curvas empleadas sean del mismo grado. Bastaría pues con elevar el grado de todas las curvas involucradas hasta el de aquella con grado más alto estableciendo nuevos vértices mediante* (3.2.1).

3.2.1. Elevación del grado de forma repetida

El proceso de elevación del grado asigna un polígono $\mathcal{E}P$ de grado $n+1$ al polígono original P de grado n. Si repetimos este proceso r veces, obtendremos los polígonos $P, \mathcal{E}P, \mathcal{E}^2P, \ldots, \mathcal{E}^rP$, de tal manera que el polígono $\mathcal{E}^rP$ tiene por vértices $\mathbf{b}_0^{(r)}, \ldots, \mathbf{b}_{n+r}^{(r)}$. El siguiente teorema da una fórmula explícita para hallar estos nodos.

Teorema 3.3. *Sea* $\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t),\ t \in [0,1]$, *una curva de Bézier de grado* n, *con* P *su polígono de Bézier asociado, definido mediante los puntos* $\mathbf{b}_0, \ldots, \mathbf{b}_n$. *Entonces el polígono* $\mathcal{E}^rP$ *obtenido tras repetir el proceso de elevación de grado* r *veces, tiene por vértices* $\mathbf{b}_0^{(r)}, \ldots, \mathbf{b}_{n+r}^{(r)}$, *donde*

$$\mathbf{b}_i^{(r)} = \sum_{j=0}^{n} \mathbf{b}_j \binom{n}{j} \frac{\binom{r}{i-j}}{\binom{n+r}{i}}, \quad 0 \le i \le n+r, \tag{3.2.2}$$

entendiendo siempre que el número combinatorio $\binom{a}{b} = 0$ *tanto si* $b < 0$, *como si* $b > a$.

Demostración. Mediante inducción sobre r.

- Para $r = 1$, $b_0^{(1)} = b_0$, $b_{n+1}^{(1)} = b_n$ y para $1 \le i \le n$, utilizando (3.2.1) se tiene

$$\begin{aligned} \mathbf{b}_i^{(1)} &= \frac{i}{n+1}\mathbf{b}_{i-1} + \left(1 - \frac{i}{n+1}\right)\mathbf{b}_i = \binom{n}{i-1}\frac{1}{\binom{n+1}{i}}\mathbf{b}_{i-1} + \binom{n}{i}\frac{\binom{1}{0}}{\binom{n+1}{i}}\mathbf{b}_i \\ &= \sum_{j=0}^{n} \mathbf{b}_j \binom{n}{j} \frac{\binom{1}{i-j}}{\binom{n+1}{i}}, \end{aligned}$$

  donde de los $n+1$ sumandos que aparecen en la última expresión son todos nulos excepto para $j = i, i-1$.

- Supongamos que $\mathbf{b}_i^{(r)} = \sum_{j=0}^{n} \mathbf{b}_j \binom{n}{j} \frac{\binom{r}{i-j}}{\binom{n+r}{i}}, \quad 0 \le i \le n+r$, y veamos que

$$\mathbf{b}_i^{(r+1)} = \sum_{j=0}^{n} \mathbf{b}_j \binom{n}{j} \frac{\binom{r+1}{i-j}}{\binom{n+r+1}{i}}, \quad 0 \leq i \leq n+r+1.$$

Sabemos que para $0 \leq i \leq n+r+1$,

$$\begin{aligned}
\mathbf{b}_i^{(r+1)} &= \frac{i}{n+r+1}\mathbf{b}_{i-1}^{(r)} + \left(1 - \frac{i}{n+r+1}\right)\mathbf{b}_i^{(r)} \\
&= \frac{i}{n+r+1}\sum_{j=0}^{n} \mathbf{b}_j \binom{n}{j} \frac{\binom{r}{i-j-1}}{\binom{n+r}{i-1}} + \left(1 - \frac{i}{n+r+1}\right)\sum_{j=0}^{n} \mathbf{b}_j \binom{n}{j} \frac{\binom{r}{i-j}}{\binom{n+r}{i}} \\
&= \frac{i}{n+r+1}\sum_{j=0}^{n} \mathbf{b}_j \binom{n}{j} \frac{r!(n+r-i+1)!(i-1)!}{(n+r)!(i-j)!(r-i+j+1)!}(r+1) \\
&= \sum_{j=0}^{n} \mathbf{b}_j \binom{n}{j} \frac{(r+1)!(n+r-i+1)!i!}{(n+r+1)!(i-j)!(r-i+j+1)!} \\
&= \sum_{j=0}^{n} \mathbf{b}_j \binom{n}{j} \frac{\binom{r+1}{i-j}}{\binom{n+r+1}{i}}.
\end{aligned}$$

□

Investiguemos ahora qué sucede si elevamos el grado repetidas veces.

Proposición 3.4. *Los polígonos $\mathcal{E}^r P$ convergen a la curva que ellos mismos definen cuando $r \to \infty$.*

Demostración. En la demostración de la Proposición 3.4 vamos a utilizar los tres resultados auxiliares que se enuncian a continuación y cuya demostración se omite.

Resultado 1. *Gracias a la Fórmula de Stirling, podemos afirmar que cuando $r \to \infty$ e $i \to \infty$ (que es justamente lo que ocurre en este caso),*

$$\begin{aligned}
r! &\sim \sqrt{2\pi r}\left(\frac{r}{e}\right)^r, \\
i! &\sim \sqrt{2\pi i}\left(\frac{i}{e}\right)^i, \\
(i-j)! &\sim \sqrt{2\pi(i-j)}\left(\frac{i-j}{e}\right)^{i-j}, \\
(n+r)! &\sim \sqrt{2\pi(n+r)}\left(\frac{n+r}{e}\right)^{n+r}.
\end{aligned}$$

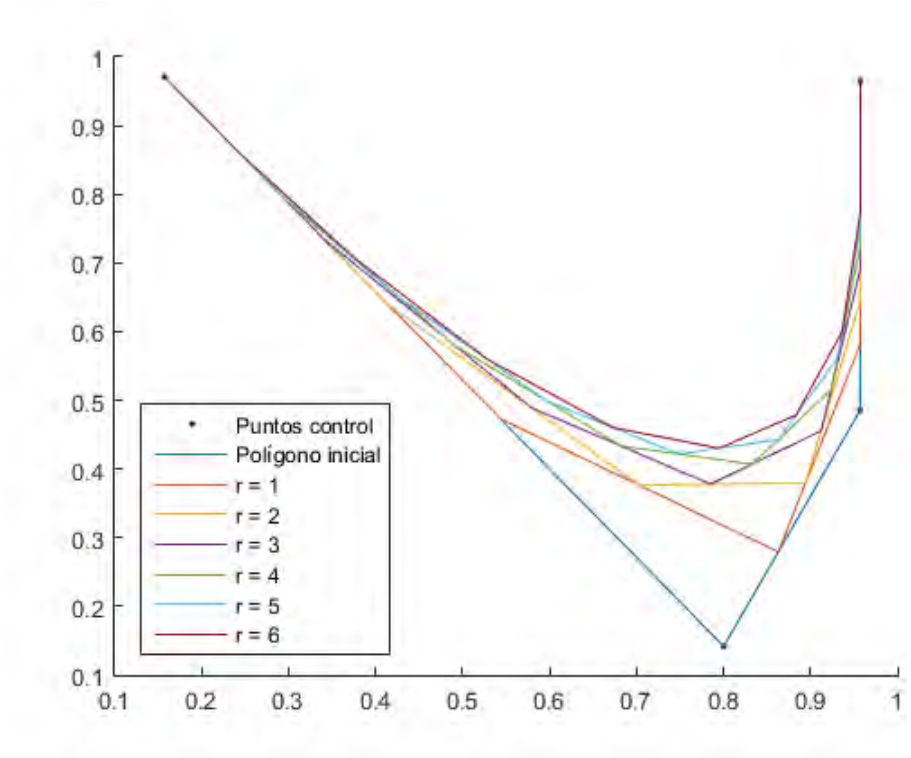


Figura 3.2: Elevación repetida del grado para una curva de Bézier cúbica hasta $r = 6$.

Resultado 2.

$$\lim_{r\to\infty}\left(\frac{r}{n+r}\right)^{r} = \lim_{r\to\infty}\left[\left(1+\frac{1}{\frac{-(n+r)}{n}}\right)^{\frac{-(n+r)}{n}}\right]^{\frac{-nr}{n+r}} = e^{\lim_{r\to\infty}\frac{-nr}{n+r}} = e^{-n}.$$

$$\lim_{i\to\infty}\left(\frac{i}{i-j}\right)^{i} = \lim_{i\to\infty}\left[\left(1+\frac{1}{\frac{i-j}{j}}\right)^{\frac{i-j}{j}}\right]^{\frac{ij}{i-j}} = e^{\lim_{i\to\infty}\frac{ij}{i-j}} = e^{j}.$$

Resultado 3.

$$\lim_{r,i\to\infty}\sqrt{\frac{ri}{(i-j)(n+r)}} = \lim_{r,i\to\infty}\sqrt{\frac{ri}{ri+ni-nj-jr}} = \lim_{r,i\to\infty}\sqrt{\frac{1}{1+\frac{n}{r}-\frac{nj}{ri}-\frac{j}{i}}} = 1.$$

Fijamos ahora un valor del parámetro $t \in [0,1]$, y para cada $r \in \mathbb{N}$ consideramos

$$i_r = \min_{0\le i\le n+r}\left|t-\frac{i}{n+r}\right|, \quad 0 \le i_r \le n+r,$$

de tal manera que cuando $r \to \infty$ (y $i_r \to \infty$), $\frac{i_r}{n+r} \to t$. Si logramos probar que,

$$\lim_{\frac{i_r}{r+n}\to t}\frac{\binom{r}{i_r-j}}{\binom{n+r}{i_r}} = t^j(1-t)^{n-j}, \tag{3.2.3}$$

habremos finalizado la demostración.

$$\begin{aligned}
\lim_{\frac{i_r}{r+n}\to t} \frac{\binom{r}{i_r-j}}{\binom{n+r}{i_r}} &= \lim_{\frac{i_r}{r+n}\to t} \frac{r!i_r!(r+n-i_r)!}{(i_r-j)!(r-i_r+j)!(n+r)!} \\
&\overset{1}{=} \lim_{\frac{i_r}{r+n}\to t} \frac{2\pi\sqrt{ri_r}r^r i_r^{i_r}(r+n-i_r)!e^{i_r-j+n+r}}{2\pi e^{r+i_r}\sqrt{(i_r-j)(n+r)}(i_r-j)^{i_r-j}(n+r)^{n+r}(r-i_r+j)!} \\
&= \lim_{\frac{i}{r+n}\to t} e^{n-j}\left(\frac{r}{n+r}\right)^r\left(\frac{i_r}{i_r-j}\right)^{i_r}\sqrt{\frac{ri_r}{(i_r-j)(n+r)}}\frac{(i_r-j)^j(r+n-i_r)!}{(n+r)^n(r-i_r+j)!} \\
&\overset{2}{=} \lim_{\frac{i_r}{r+n}\to t} e^{n-j}e^{-n+j}\sqrt{\frac{ri_r}{(i_r-j)(n+r)}}\frac{(i_r-j)^j(r+n-i_r)!}{(n+r)^n(r-i_r+j)!} \\
&\overset{3}{=} \lim_{\frac{i_r}{r+n}\to t} \frac{(i_r-j)^j}{(n+r)^n}(r-i_r+n)\cdots(r-i_r+j+1) \\
&= \lim_{\frac{i_r}{r+n}\to t} \frac{(i_r-j)^j}{(n+r)^n}(n+r)^{n-j}\left(1-\frac{i_r}{n+r}\right)\cdots\left(1-\frac{n+i_r-j-1}{n+r}\right) \\
&= \lim_{\frac{i_r}{r+n}\to t} \frac{(i_r-j)^j}{(n+r)^n}(1-t)^{n-j}(n+r)^{n-j} \\
&= \lim_{\frac{i_r}{r+n}\to t}\left(\frac{i_r-j}{n+r}\right)^j(1-t)^{n-j} = \lim_{\frac{i_r}{r+n}\to t}\left(\frac{i_r}{n+r}\right)^j(1-t)^{n-j} = t^j(1-t)^{n-j}.
\end{aligned}$$

Por tanto, hemos demostrado que, fijado $t \in [0,1]$,

$$\begin{aligned}
\lim_{\frac{i_r}{r+n}\to t} \mathbf{b}_{i_r}^{(r)} &= \lim_{\frac{i_r}{r+n}\to t}\left[\sum_{j=0}^{i_r-r}\mathbf{b}_j\binom{n}{j}\frac{\binom{r}{i_r-j}}{\binom{n+r}{i_r}}\right] = \sum_{j=0}^{i_r-r}\mathbf{b}_j\binom{n}{j}\left[\lim_{\frac{i_r}{r+n}\to t}\frac{\binom{r}{i_r-j}}{\binom{n+r}{i_r}}\right] \\
&= \sum_{j=0}^{i_r-r}\mathbf{b}_j\binom{n}{j}t^j(1-t)^{n-j} = \mathbf{P}(t).
\end{aligned}$$

□

Dicho de manera informal, hemos probado que para cada $t \in [0,1]$, el nodo i-ésimo del polígono inicial P de grado n más cercano a $\mathbf{P}(t)$, converge al valor $\mathbf{P}(t)$ al aumentar sucesivamente el grado del polinomio (es decir, cuando $r \to \infty$).

La Figura 3.2 ilustra este hecho para una curva de grado 3. Se han incluido junto con el polígono original formado por cuatro puntos de control, los polígonos que se van obteniendo al elevar el grado hasta 6 veces.

3.3. Reducción del grado

El proceso de elevación del grado resulta, en cierto modo, redundante, pues la curva está descrita mediante más información de la que realmente necesita. El proceso inverso puede resultar más interesante:

Dada una curva de Bézier de grado n definida mediante los $n+1$ puntos de control $\mathbf{b}_0, \ldots, \mathbf{b}_n$, ¿podemos encontrar una curva de Bézier de grado $n-1$ definida mediante n puntos de control $\hat{\mathbf{b}}_0, ..., \hat{\mathbf{b}}_{n-1}$ que la aproxime? Denominaremos a este aún hipotético algoritmo, **reducción del grado.**

Supongamos que los $\mathbf{b}_i$ se obtuvieron a partir de los $\hat{\mathbf{b}}_i$ empleando el proceso de elevación del grado descrito en la Sección 3.2. Entonces, según (3.2.1) debería ser,

$$\mathbf{b}_i = \frac{i}{n}\hat{\mathbf{b}}_{i-1} + \frac{n-i}{n}\hat{\mathbf{b}}_i, \quad i = 0, \ldots, n. \tag{3.3.1}$$

De esta ecuación derivan dos fórmulas recursivas que expresan la generación de los $\hat{\mathbf{b}}_i$ a partir de $\mathbf{b}_i$,

$$\overrightarrow{\mathbf{b}_i} = \frac{n\mathbf{b}_i - i\overrightarrow{\mathbf{b}_{i-1}}}{n-i}, \quad i = 0, \ldots, n-1, \tag{3.3.2}$$

$$\overleftarrow{\mathbf{b}_{i-1}} = \frac{n\mathbf{b}_i - (n-i)\overleftarrow{\mathbf{b}_i}}{i}, \quad i = n, n-1, \ldots, 1, \tag{3.3.3}$$

donde los $\overrightarrow{\mathbf{b}_i}$ se obtienen de despejar en (3.3.1) de izquierda a derecha, mientras que los $\overleftarrow{\mathbf{b}_i}$ se obtienen de despejar en (3.3.1) de derecha a izquierda. Además, de (3.3.2) y (3.3.3) se deduce que $\overrightarrow{\mathbf{b}_0} = \mathbf{b}_0$ y $\overleftarrow{\mathbf{b}_{n-1}} = \mathbf{b}_n$.

Observación 3.5. *En las ecuaciones* (3.3.2) *y* (3.3.3)*, aparecen dos términos indefinidos, $\overrightarrow{\mathbf{b}_{-1}}$ y $\overleftarrow{\mathbf{b}_n}$. Sin embargo van precedidos de un factor multiplicativo nulo, por lo que ambas expresiones son coherentes.*

En la mayoría de las ocasiones, la reducción del grado no puede llevarse a cabo de forma exacta, por lo que es vista como un método de aproximación. Si la curva dada tiene grado $n-1$ inicialmente, entonces $\overrightarrow{\mathbf{b}_i}$ y $\overleftarrow{\mathbf{b}_i}$, $i = 0, \ldots, n-1$, producirán ambas la curva original de grado $n-1$, pero al ser 3.3.2 y 3.3.3 fórmulas de extrapolación, son numéricamente inestables y por tanto lo más usual será obtener aproximaciones poco afinadas e inservibles.

En la Figura 3.3 se puede intuir el hecho de que todos los vectores $\overrightarrow{\mathbf{b}_i} - \overleftarrow{\mathbf{b}_i}$ son paralelos. Para verlo, debemos observar que los triángulos $\overleftarrow{\mathbf{b}_{i-1}}, \overrightarrow{\mathbf{b}_{i-1}}, \mathbf{b}_i$ y $\overleftarrow{\mathbf{b}_i}, \overrightarrow{\mathbf{b}_i}, \mathbf{b}_i$ son similares.

Demostración. Gracias a (3.3.2) y a (3.3.3),

$$\overleftarrow{\mathbf{b}_{i-1}} = \frac{(n-i)(\mathbf{b}_i - \overleftarrow{\mathbf{b}_i}) + i\mathbf{b}_i}{i}, \qquad \overrightarrow{\mathbf{b}_i} = \frac{(n-i)\mathbf{b}_i - i(\overrightarrow{\mathbf{b}_{i-1}} - \mathbf{b}_i)}{n-i}.$$

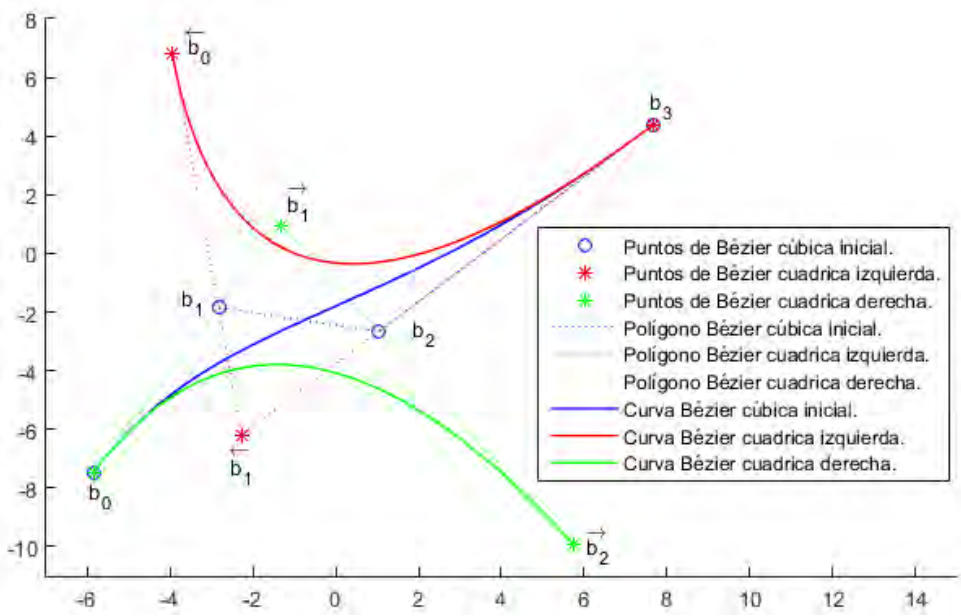


Figura 3.3: Ejemplo de reducción del grado. En azul la curva de Bézier cúbica a aproximar, en verde la curva de Bézier cuadrática que tiene por puntos de control los definidos mediante (3.3.2) y en rojo la curva de Bézier cuadrática que tiene por puntos de control los definidos mediante (3.3.3).

Despejando se obtiene,

$$\frac{\mathbf{b}_i - \overleftarrow{\mathbf{b}_i}}{\overleftarrow{\mathbf{b}_{i-1}} - \mathbf{b}_i} = \frac{i}{n-i}, \qquad \frac{\overrightarrow{\mathbf{b}_i} - \mathbf{b}_i}{\mathbf{b}_i - \overrightarrow{\mathbf{b}_{i-1}}} = \frac{i}{n-i},$$

por lo que,

$$\frac{\mathbf{b}_i - \overleftarrow{\mathbf{b}_i}}{\overleftarrow{\mathbf{b}_{i-1}} - \mathbf{b}_i} = \frac{\overrightarrow{\mathbf{b}_i} - \mathbf{b}_i}{\mathbf{b}_i - \overrightarrow{\mathbf{b}_{i-1}}},$$

tal y como se quería probar. □

A continuación, se da una fórmula que permite calcular los nodos $\overrightarrow{\mathbf{b}_i},\ i = 0, \ldots, n-1$ a partir de $\mathbf{b}_0, \ldots, \mathbf{b}_i,\ i = 0, \ldots, n-1$.

Corolario 3.6.

$$\overrightarrow{\mathbf{b}_i} = \frac{1}{\binom{n-1}{i}} \sum_{j=0}^{i} (-1)^{i+j} \binom{n}{j} \mathbf{b}_j, \quad i = 0, \ldots, n-1.$$

Demostración. Para $i = 0$, es obvio pues

$$\frac{1}{\binom{n-1}{0}} (-1)^0 \binom{n}{0} \mathbf{b}_0 = \mathbf{b}_0 = \overrightarrow{\mathbf{b}_0}.$$

Supongamos que

$$\overrightarrow{\mathbf{b}_i} = \frac{1}{\binom{n-1}{i}} \sum_{j=0}^{i} (-1)^{i+j} \binom{n}{j} \mathbf{b}_j.$$

y veamos que

$$\overrightarrow{\mathbf{b}_{i+1}} = \frac{1}{\binom{n-1}{i+1}} \sum_{j=0}^{i+1} (-1)^{i+1+j} \binom{n}{j} \mathbf{b}_j.$$

Para ello, tan solo hay que tener en cuenta que,

$$\begin{aligned}
\overrightarrow{\mathbf{b}_{i+1}} &= \frac{n\mathbf{b}_{i+1} - (i+1)\overrightarrow{\mathbf{b}_i}}{n-i-1} \\
&= \frac{1}{n-i-1}\left[n\mathbf{b}_{i+1} - \frac{i+1}{\binom{n-1}{i}} \sum_{j=0}^{i} (-1)^{i+j} \binom{n}{j} \mathbf{b}_j \right] \\
&= \frac{1}{\frac{(n-1)!}{(i+1)!(n-i-2)!}} \left[\frac{n!}{(i+1)!(n-i-1)!} \mathbf{b}_{i+1} + \sum_{j=0}^{i} (-1)^{i+j+1} \binom{n}{j} \mathbf{b}_j \right] \\
&= \frac{1}{\binom{n-1}{i+1}} \left[\binom{n}{i+1} \mathbf{b}_{i+1} + \sum_{j=0}^{i} (-1)^{i+j+1} \binom{n}{j} \mathbf{b}_j \right] \\
&= \frac{1}{\binom{n-1}{i+1}} \sum_{j=0}^{i+1} (-1)^{i+1+j} \binom{n}{j} \mathbf{b}_j.
\end{aligned}$$

□

En la Figura 3.3 puede intuirse que (3.3.2) es una buena aproximación al polinomio de grado $n = 3$ cerca de $\mathbf{b}_0$ y (3.3.3) de $\mathbf{b}_n = \mathbf{b}_3$. Es por esta razón por la que parece coherente combinar ambas expresiones,

$$\hat{\mathbf{b}}_i = (1-\lambda_i)\overrightarrow{\mathbf{b}_i} + \lambda_i \overleftarrow{\mathbf{b}_i}, \quad i = 0, \ldots, n-1.$$

buscando así $\lambda_i \in \mathbb{R},\ i = 0, \ldots, n-1$ tales que los $\hat{\mathbf{b}}_i$ generen el polinomio de grado $n-1$ que mejor aproxime al de grado n dado. La búsqueda de estos $\lambda_i \in \mathbb{R},\ i = 0, \ldots, n-1$ no es sencilla. En primer lugar, es necesario expresar el polinomio de grado n dado en la base de los Polinomios de Chebyschev $\{T_0(t), \ldots, T_n(t)\}$, definidos en $[-1, 1]$,

$$\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{t}_i T_i(t), \qquad t \in [-1, 1].$$

Posteriormente se trunca el término de mayor grado $\mathbf{t}_n T_n(t)$, obteniendo así el único polinomio $\mathbf{Q}$ de grado $n-1$ que dista del dado lo menos posible. Dicho de otro modo, encontramos así el polinomio $\mathbf{Q} \in \mathbb{P}_{n-1}$ tal que

$$\max_{-1 \le t \le 1} ||\mathbf{P}(t) - \mathbf{Q}(t)|| \le \max_{-1 \le t \le 1} ||\mathbf{P}(t) - \mathbf{R}(t)||, \quad \forall \mathbf{R} \in \mathbb{P}_{n-1},$$

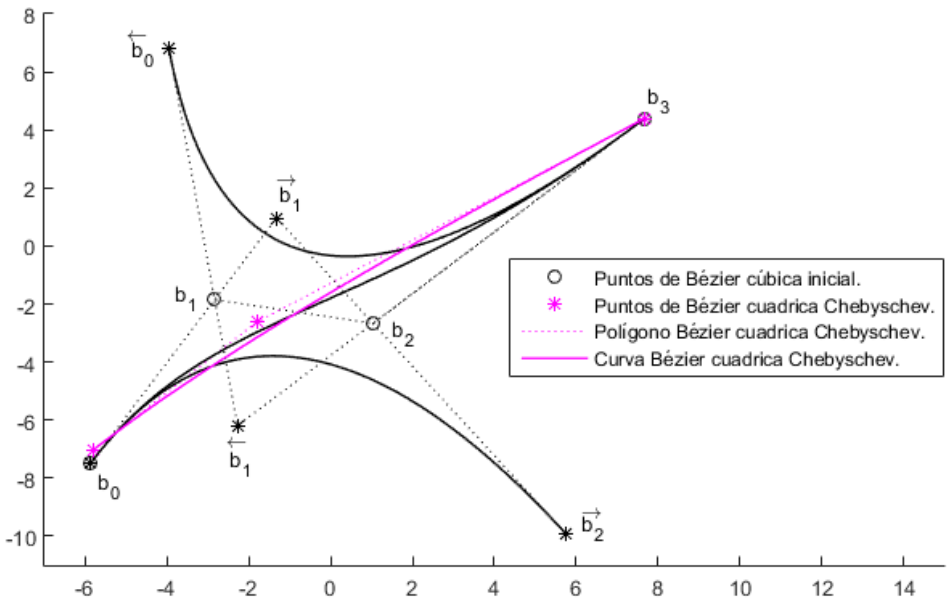


Figura 3.4: En negro, las curvas de la Figura 3.3. En rosa, aproximación mediante el polinomio de grado $n-1=2$ definido por los nodos $\hat{\mathbf{b}}_0, \hat{\mathbf{b}}_1$ y $\hat{\mathbf{b}}_2$ donde $\lambda_0 = \frac{1}{2^5}, \lambda_1 = \frac{1}{2}$ y $\lambda_2 = 1$.

y se reescribe el polinomio resultante, de nuevo, en la base de los Polinomios de Bernstein. A este proceso se le conoce con el nombre de *Economización de Chebyschev*. Finalmente, se obtiene que los λ_i óptimos vienen dados por,

$$\lambda_i = \frac{1}{2^{2n-1}} \sum_{j=0}^{i} \binom{2n}{2j}, \qquad i = 0, \ldots, n-1. \tag{3.3.4}$$

No profundizaremos más en este resultado, pues excede en extensión al alcance de este trabajo. Para más información, consultar Eck (1993). La Figura 3.4 muestra la aproximación del ejemplo de la Figura 3.3 en color negro junto la aproximación mediante los λ_i definidos en (3.3.4) en rosa.

Capítulo 4

Splines en la forma de Bézier

Si bien es cierto que las curvas de Bézier son de extrema utilidad en el ámbito del diseño, presentan numerosos problemas cuando la curva a aproximar tiene forma complicada. En este caso, su representación de Bézier tendrá grado alto, y en la práctica, polinomios de grado ≥ 10 resultan de una complejidad muy elevada (tal y como podrá apreciarse posteriormente en la Figura 5.5 del Capítulo 5). Un ejemplo de este suceso fue observado por Carl Runge al intentar aproximar la función

$$f(x) = \frac{1}{1+x^2}, \quad x \in [-5, 5],$$

mediante el polinomio P_n que interpola los nodos $x_j^{(n)} = -5 + 10\frac{j}{n},\ j = 0, \ldots, n$, pues este presenta una oscilación de gran amplitud cerca de los extremos del intervalo. Además el error de interpolación tiende a infinito cuando crece el grado del polinomio (ver Epperson (1987)),

$$\lim_{n\to\infty} \left(\max_{-5\leq x\leq 5} |f(x) - P_n(x)| \right) = \infty.$$

Este problema desaparece al interpolar a trozos utilizando una partición cada vez más fina. De manera análoga, se pueden construir curvas de Bézier a trozos para evitar el uso de curvas polinómicas de grado tan elevado.

4.1. Parámetros globales y locales

Definición 4.1. *Dados $u_0 < \cdots < u_L$, $u_i \in \mathbb{R}$, $0 \leq i \leq L$, llamamos **spline de Bézier** con **L trozos** (o simplemente **spline de Bézier a trozos**) a toda aplicación continua*

$$\mathbf{s} : [u_0, u_L] \longrightarrow \mathbb{R}^2,$$

*donde cada intervalo $[u_i, u_{i+1}]$, $i = 0, \ldots, L-1$, tiene por imagen un segmento de curva de Bézier. Llamaremos **polígono de Bézier a trozos de** $\mathbf{s}$ al conjunto formado por los polígonos de Bézier de todos los segmentos de curva.*

Para cada valor del parámetro $u \in [u_0, u_L]$ obtendremos un punto $\mathbf{s}(u)$ en la curva $\mathbf{s}$ y al mismo tiempo existirá un único subintervalo $[u_i, u_{i+1}]$ para un i concreto de tal manera que $u \in [u_i, u_{i+1}]$. Podemos introducir un *parámetro local* t_i para cada intervalo $[u_i, u_{i+1}]$,

$$t_i = \frac{u - u_i}{u_{i+1} - u_i} = \frac{u - u_i}{\triangle_i}. \tag{4.1.1}$$

De este modo t_i varía desde 0 hasta 1 mientras u varía desde u_i hasta u_{i+1}. Para referirnos al $i-$ésimo segmento de $\mathbf{s}$, emplearemos la notación $\mathbf{s}_i$ de tal manera que para $u \in [u_i, u_{i+1}],\ \mathbf{s}(u) = \mathbf{s}_i(t_i),\ t_i \in [0, 1]$.

Tras la introducción de parámetros locales $t_i \in [0, 1],\ 0 \leq i \leq L$, la definición de las derivadas se transformaría en

$$\frac{d\mathbf{s}(u)}{du} = \frac{d\mathbf{s}_i(t_i)}{dt_i}\frac{dt_i}{du} = \frac{1}{\triangle_i}\frac{d\mathbf{s}_i(t_i)}{dt_i}. \tag{4.1.2}$$

4.2. Condiciones de regularidad

Supongamos que conocemos dos curvas de Bézier, $\mathbf{s}_0$ y $\mathbf{s}_1$, definidas mediante los puntos de control $\mathbf{b}_0, \ldots, \mathbf{b}_n$ y $\mathbf{b}_n, \ldots, \mathbf{b}_{2n}$ respectivamente (las dos curvas pasan por el punto $\mathbf{b}_n$). Pensemos en cada una de ellas como una curva en si misma. Imaginemos que al mismo tiempo dichas curvas son segmentos de una única curva mayor $\mathbf{s}$ definida en un intervalo $[u_0, u_2]$ y con imagen en $\mathbb{R}^2$. Supongamos sin pérdida de generalidad que la curva $\mathbf{s}_0$ es el *segmento de la izquierda* definido en el intervalo $[u_0, u_1]$ mientras que la curva $\mathbf{s}_1$ es el *segmento de la derecha* definido en el intervalo $[u_1, u_2]$. Gracias a la Sección 3.1 sabemos que los dos polígonos $\mathbf{b}_0, \ldots, \mathbf{b}_n$ y $\mathbf{b}_n, \ldots, \mathbf{b}_{2n}$, han de ser el resultado de un proceso de subdivisión. Por tanto sus vértices han de estar relacionados mediante

$$\mathbf{b}_{n+i} = \mathbf{b}^i_{n-i}(t), \quad i = 0, \ldots, n,$$

donde $t = \dfrac{u_2 - u_0}{u_1 - u_0}$ es la coordenada local de u_2 con respecto al intervalo $[u_0, u_1]$.

Supongamos ahora que modificamos arbitrariamente $\mathbf{b}_{2n}$, produciendo así que las dos curvas no definan ya un mismo polinomio global $\mathbf{s}$. Sin embargo, siguen coincidiendo en todas las derivadas de orden $0, \ldots, n-1$ en el punto $u = u_1$, pues $\mathbf{b}_{2n}$ no ejerce influencia alguna en las derivadas de orden menor que n en dicho punto (ver Teoremas 2.6 y 2.10). De manera similar, podemos modificar los nodos $\mathbf{b}_{2n-1}, \ldots, \mathbf{b}_{2n-r}$ y seguir manteniendo la continuidad de todas las derivadas hasta la de orden $n - r - 1$.

Farin (1997) describe precisamente cómo hemos encontrado de este modo una condición necesaria y suficiente que han de cumplir las curvas de Bézier para tener regularidad $\mathcal{C}^r,\ 0 \leq r < n$.

Definición 4.2. *Sean dos curvas de Bézier* $\mathbf{s}_0$ *y* $\mathbf{s}_1$ *definidas en* $[u_0, u_1]$ *y* $[u_1, u_2]$ *respectivamente mediante los puntos de control* $\mathbf{b}_0, \ldots, \mathbf{b}_n$ *y* $\mathbf{b}_n, \ldots, \mathbf{b}_{2n}$. *Diremos que la curva*

de Bézier $\mathbf{s}$ *definida por*

$$\begin{array}{rccl} \mathbf{s}: & [u_0,u_2] & \longrightarrow & \mathbb{R}^2 \\ & u & \longrightarrow & \mathbf{s}(u)=\begin{cases} \mathbf{s}_0(t_0)=\mathbf{s}_0\left(\dfrac{u-u_0}{u_1-u_0}\right), & si \quad u\in[u_0,u_1], \\ \mathbf{s}_1(t_1)=\mathbf{s}_1\left(\dfrac{u-u_1}{u_2-u_1}\right), & si \quad u\in[u_1,u_2], \end{cases} \end{array} \tag{4.2.1}$$

es $\mathbf{r}$ ***veces continuamente diferenciable*** *en* u_1 *si y solo si*

$$\mathbf{b}_{n+i}=\mathbf{b}_{n-i}^{i}(t), \quad i=0,\ldots,r, \tag{4.2.2}$$

donde $t=\dfrac{u_2-u_0}{u_1-u_0}$ *es la coordenada local de* u_2 *con respecto al intervalo* $[u_0,u_1]$ *y*

$$\mathbf{b}_{n-i}^{i}(t):=\sum_{j=0}^{i}\mathbf{b}_{n-i+j}B_j^i(t)=\sum_{j=n-i}^{n}\mathbf{b}_jB_{j-n+i}^i(t), \qquad i=0,\ldots,n-i.$$

Se puede manejar otra condición de regularidad $\mathcal{C}^r$, obtenida igualando las derivadas de $\mathbf{s}_0$ y $\mathbf{s}_1$ y aplicando la regla de la cadena.

Gracias a (2.2.1) y a la Observación 2.7 se puede deducir que

$$\begin{aligned} \frac{d^k\mathbf{s}_0}{du^k}(t_0) &= \frac{n!}{(n-k)!}\sum_{i=0}^{n-k}\triangle^k\mathbf{b}_iB_i^{n-k}(t_0)\frac{d^kt_0}{du^k}, \qquad t_0=\frac{u-u_0}{u_1-u_0}, \\ \frac{d^k\mathbf{s}_1}{du^k}(t_1) &= \frac{n!}{(n-k)!}\sum_{i=0}^{n-k}\triangle^k\mathbf{b}_{n+i}B_i^{n-k}(t_1)\frac{d^kt_1}{du^k}, \quad t_1=\frac{u-u_1}{u_2-u_1}. \end{aligned}$$

Tras reemplazar $\dfrac{d^kt_0}{du^k}$ y $\dfrac{d^kt_1}{du^k}$ por $\left(\dfrac{1}{u_1-u_0}\right)^k$ y $\left(\dfrac{1}{u_2-u_1}\right)^k$ respectivamente y evaluando en $t_0=1$ y $t_1=0$,

$$\begin{aligned} \frac{d^k\mathbf{s}_0}{du^k}(1) &= \frac{n!}{(n-k)!}\sum_{i=0}^{n-k}\triangle^k\mathbf{b}_iB_i^{n-k}(1)\left(\frac{1}{u_1-u_0}\right)^k, \\ \frac{d^k\mathbf{s}_1}{du^k}(0) &= \frac{n!}{(n-k)!}\sum_{i=0}^{n-k}\triangle^k\mathbf{b}_{n+i}B_i^{n-k}(0)\left(\frac{1}{u_2-u_1}\right)^k. \end{aligned}$$

Igualando el lado derecho de ambas expresiones,

$$\triangle^k\mathbf{b}_{n-k}B_{n-k}^{n-k}(1)\left(\frac{1}{\triangle_0}\right)^k=\triangle^k\mathbf{b}_nB_0^{n-k}(0)\left(\frac{1}{\triangle_1}\right)^k,$$

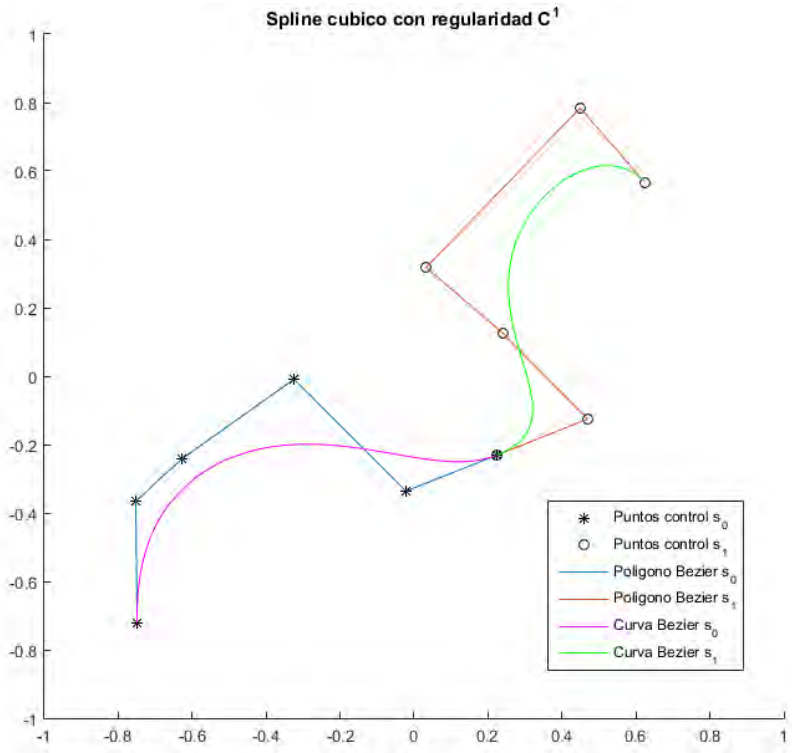


Figura 4.1: Ejemplo de spline de Bézier de grado 5 formado por dos trozos, que sí presenta regularidad $\mathcal{C}^1$ en el nodo que las une. En este caso, $u_0 = 0, u_1 = 1, u_2 = 2$.

Por lo que finalmente obtenemos la condición

$$\triangle^k \mathbf{b}_{n-k} \left(\frac{1}{\triangle_0}\right)^k = \triangle^k \mathbf{b}_n \left(\frac{1}{\triangle_1}\right)^k, \quad k = 0, \ldots, r, \text{ donde } \triangle_0 = u_1 - u_0, \ \triangle_1 = u_2 - u_1. \tag{4.2.3}$$

Condiciones para derivadas de orden más alto fueron tratadas por primera vez por E. Staerk en 1976. Trataremos los casos $\mathcal{C}^1$ y $\mathcal{C}^2$ en este capítulo.

4.2.1. Regularidad $\mathcal{C}^1$

Sea $\mathbf{s}$ un spline de Bézier con dos trozos similar al definido en (4.2.1), donde $\mathbf{s_0}$ está definido mediante los puntos $\mathbf{b}_0, \ldots, \mathbf{b}_n$ y $\mathbf{s_1}$ mediante $\mathbf{b}_n, \ldots, \mathbf{b}_{2n}$. Gracias a (4.2.2), para $i = 1$, sabemos que $\mathbf{s}$ presenta regularidad $\mathcal{C}^1$ en $u = u_1$ si y solo si

$$\mathbf{b}_{n+1} = \mathbf{b}_n + \frac{\triangle_1}{\triangle_0}(\mathbf{b}_n - \mathbf{b}_{n-1}),$$

es decir, si $\mathbf{b}_{n+1}$ se obtiene mediante interpolación lineal en los puntos $\mathbf{b}_n$ y $\mathbf{b}_{n-1}$.

Debido a (4.1.2) para $L = 2$,

$$\frac{d}{du}\mathbf{s}(u) = \frac{1}{\triangle_0}\frac{d}{dt_0}\mathbf{s}_0(t_0) = \frac{1}{\triangle_1}\frac{d}{dt_1}\mathbf{s}_1(t_1),$$

y evaluando en $u = u_1$ (equivalentemente, $t_0 = 1$ y $t_1 = 0$),

$$\begin{aligned}\frac{d}{dt_0}\mathbf{s}_0(1) &= n\triangle\mathbf{b}_{n-1}\\ \frac{d}{dt_1}\mathbf{s}_1(0) &= n\triangle\mathbf{b}_n,\end{aligned}$$

por lo que

$$\triangle_1\triangle\mathbf{b}_{n-1} = \triangle_0\triangle\mathbf{b}_n, \tag{4.2.4}$$

tal y como cabría esperar de (4.2.3) para $k = 1$.

En la Figura 4.1 se muestra un spline de Bézier de grado 5 formado por dos trozos que sí presenta regularidad $\mathcal{C}^1$.

La ecuación (4.2.4) muestra cómo para tener regularidad $\mathcal{C}^1$, los tres puntos $\mathbf{b}_{n-1}$,$\mathbf{b}_n$ y $\mathbf{b}_{n+1}$ tienen que ser colineales y han de encontrarse en la proporción

$$(u_1 - u_0) : (u_2 - u_1) = \triangle_0 : \triangle_1.$$

Sin embargo esta colinealidad en los nodos es necesaria para garantizar regularidad $\mathcal{C}^1$, pero no es suficiente. Supongamos ahora que $\triangle\mathbf{b}_{n-1} = \triangle\mathbf{b}_n = 0$, es decir, que los tres puntos $\mathbf{b}_{n-1}$, $\mathbf{b}_n$ y $\mathbf{b}_{n+1}$ coinciden. En este caso, la curva $\mathbf{s}$ tiene vector tangente nulo en el nodo $\mathbf{b}_n$ y se verifica (4.2.4), lo que conllevaría la afirmación de que la curva $\mathbf{s}$ es diferenciable sean cuales sean las longitudes de $\triangle_0$ y $\triangle_1$. Sin embargo, vectores tangentes nulos pueden darse en cúspides o esquinas, lo que contradice el concepto de diferenciabilidad. La Figura 4.2 muestra un ejemplo de este caso, para un spline de Bézier cuadrático formado por tres curvas.

4.2.2. Regularidad $\mathcal{C}^2$

Al igual que en los apartados anteriores, sea $\mathbf{s}$ un spline de Bézier formado por dos segmentos $\mathbf{s}_0$ y $\mathbf{s}_1$, definidos en $[u_0, u_1]$ y $[u_1, u_2]$ respectivamente. Asumamos que $\mathbf{s}$ es $\mathcal{C}^1$, verificándose por tanto (4.2.4).

Para conseguir regularidad $\mathcal{C}^2$, en virtud de (4.2.2) para el caso $r = 2$, debe corroborarse que los dos polinomios cuadráticos con puntos de control

$$\begin{aligned}&\mathbf{b}_{n-2},\ \mathbf{b}_{n-1},\ \mathbf{b}_n \quad \text{definido en} \quad [u_0, u_1] \text{ y}\\ &\mathbf{b}_n,\ \mathbf{b}_{n+1},\ \mathbf{b}_{n+2} \quad \text{definido en} \quad [u_1, u_2],\end{aligned}$$

describen el mismo polinomio global cuadrático. Equivalentemente, ha de existir un polígono

$$\mathbf{b}_{n-2},\ \mathbf{d},\ \mathbf{b}_{n+2} \ \text{ definido en } \ [u_0, u_2]$$

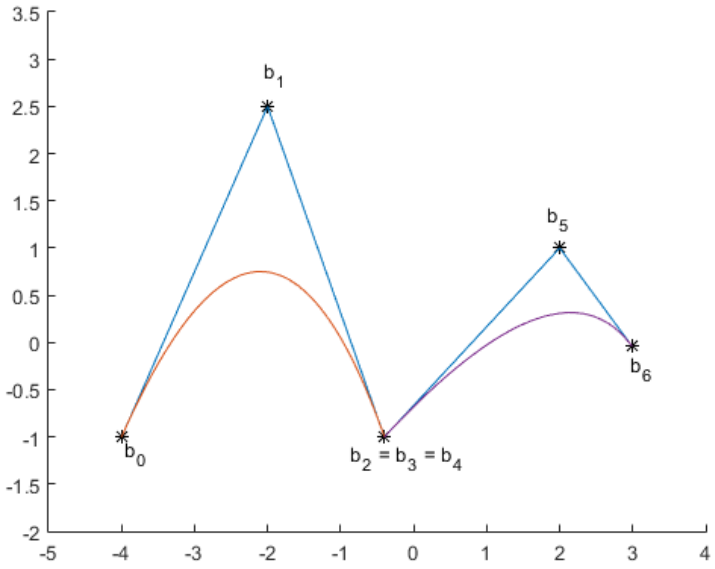


Figura 4.2: Ejemplo en el que $\mathbf{b}_{n-1} = \mathbf{b}_n = \mathbf{b}_{n+1}$ (en este caso $n = 3$), por lo que se cumple 4.2.4 pero no hay regularidad $\mathcal{C}^1$.

que describa dicho polinomio. Los dos subpolígonos descritos arriba se obtendrían por tanto de este último mediante el proceso de subdivisión para el valor del parámetro u_1.

La condición que ha de verificar un spline $\mathbf{s}$ de clase $\mathcal{C}^1$ para presentar regularidad $\mathcal{C}^2$ en u_1 es la existencia de un punto $\mathbf{d}$ tal que,

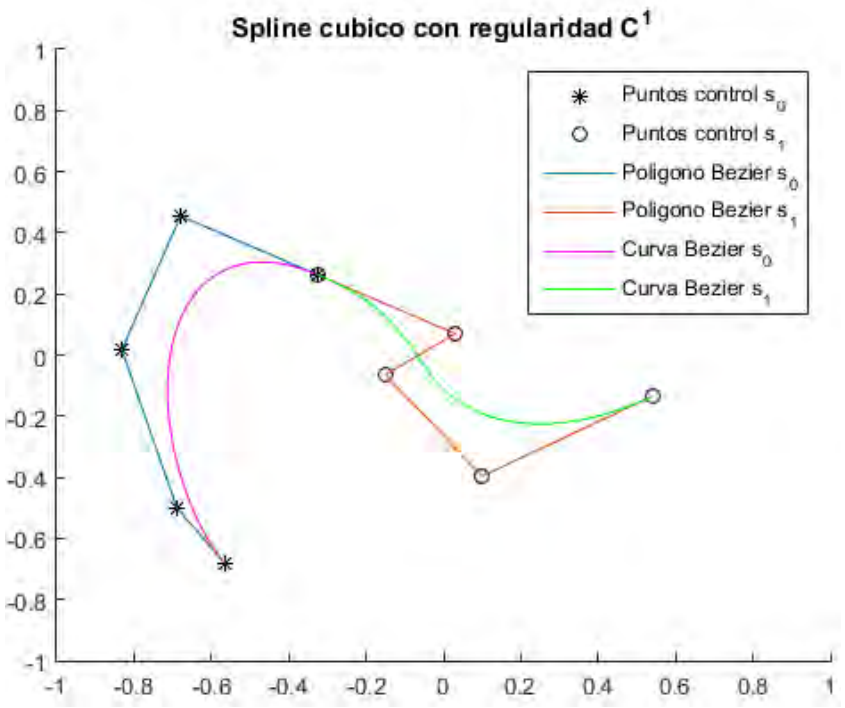
$$\begin{aligned} \mathbf{b}_{n-1} &= (1-t_1)\mathbf{b}_{n-2} + t_1\mathbf{d}, \\ \mathbf{b}_{n+1} &= (1-t_1)\mathbf{d} + t_1\mathbf{b}_{n+2}, \end{aligned} \tag{4.2.5}$$

donde $t_1 = \dfrac{\triangle_0}{u_2 - u_0}$ es el parámetro local de u_1 con respecto al intervalo $[u_0, u_2]$.

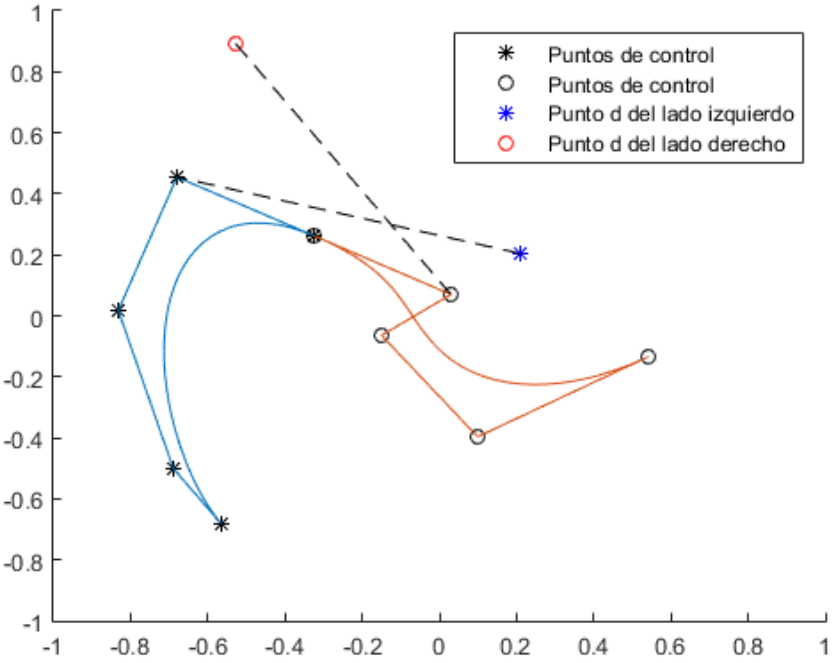
Esta condición permite comprobar fácilmente si una curva dada es o no $\mathcal{C}^2$ para un $u = u_i$, $i = 1, \ldots, L-1$, pues bastará con construir los puntos auxiliares $\mathbf{d}$ tanto del polígono de la izquierda como del de la derecha y analizar si coinciden o no.

Otra forma de comprobar si una curva dada presenta o no regularidad $\mathcal{C}^2$ en un nodo $\mathbf{b}_n$ consistiría en construir la segunda derivada por la izquierda y por la derecha e igualar ambas expresiones.

En las Figuras 4.3 y 4.4 se muestran splines de Bézier a trozos con regularidad $\mathcal{C}^1$ y $\mathcal{C}^2$, respectivamente. Además, en la Figura 4.3 (b) se comprueba gráficamente a partir de los polígonos de cada trozo que no se tiene regularidad $\mathcal{C}^2$, y en la Figura 4.4 se aprecia gráficamente que sí se satisface la condición (4.2.5) que garantiza regularidad $\mathcal{C}^2$.



(a) Dos curvas $\mathbf{s}_0$ y $\mathbf{s}_1$ junto a sus polígonos de Bézier asociados, que sí presentan regularidad $\mathcal{C}^1$.



(b) Sin embargo, no presentan regularidad $\mathcal{C}^2$, pues se obtienen dos valores distintos de $\mathbf{d}$ en las dos ecuaciones (4.2.5).

Figura 4.3: Ejemplo de spline de Bézier de grado 4 definido mediante dos trozos, sí presenta regularidad $\mathcal{C}^1$ pero no regularidad $\mathcal{C}^2$. En este caso $u_0 = 0, u_1 = 1$ y $u_2 = 2$.

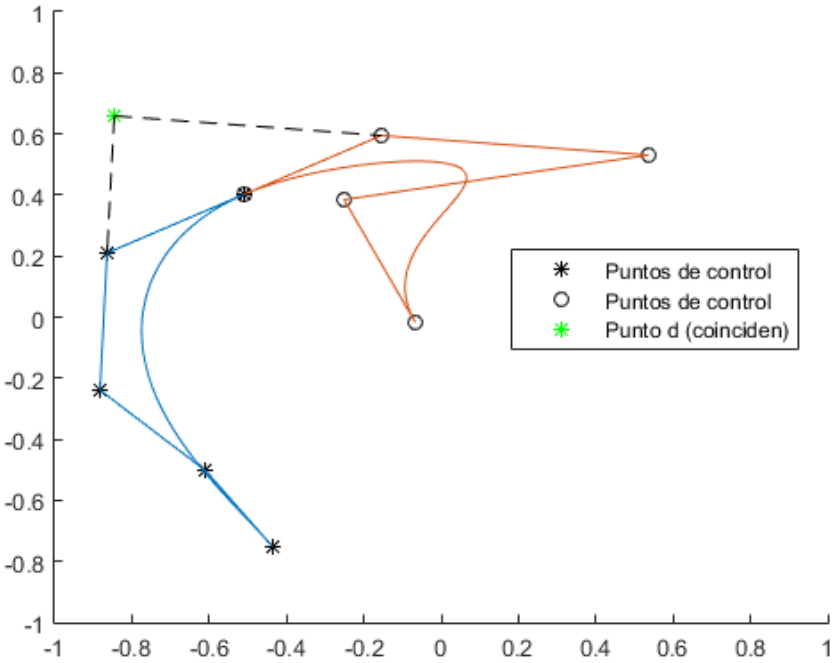


Figura 4.4: Ejemplo que muestra un spline de Bézier de grado cuatro definido mediante dos trozos que sí presenta regularidad $\mathcal{C}^2$. En este caso, $u_0 = 0, u_1 = 1$ y $u_2 = 2$.

4.3. Splines de Bézier cuadráticos a trozos con regularidad $\mathcal{C}^1$

En la Sección 4.2 se analizaba la regularidad de splines de Bézier $\mathbf{s}$ definidos mediante únicamente dos trozos $\mathbf{s}_0$, $\mathbf{s}_1$. En esta sección, trataremos el caso general de splines definidos mediante L trozos, para $L \in \mathbb{N}$ pero únicamente para el caso cuadrático.

Sean $u_0 < \cdots < u_L$, $u_i \in \mathbb{R}$, $i = 0, \ldots, L$, y consideremos los intervalos $[u_i, u_{i+1}]$, $i = 0, \ldots, L-1$. Sea $\mathbf{s} : [u_0, u_L] \longrightarrow \mathbb{R}^2$ un spline cuadrático $\mathcal{C}^1$ definido a trozos en dichos intervalos, de tal manera que para $u \in [u_i, u_{i+1}]$, con $i = 0, \ldots, L-1$ concreto,

$$\mathbf{s}(u) = \mathbf{s}_i(t_i) = \mathbf{s}_i\left(\frac{u - u_i}{u_{i+1} - u_i}\right) = \mathbf{s}_i\left(\frac{u - u_i}{\triangle_i}\right).$$

Por ser cuadrático a trozos,

$$\mathbf{s}_i(t_i) = \sum_{j=0}^{2} \mathbf{b}_{2i+j} B_j^2(t_i; u_i, u_{i+1}), \quad i = 0, \ldots, L-1.$$

Llamamos **nodos internos de Bézier** a los puntos $\mathbf{b}_{2i+1}$, $0 \leq i \leq L-1$ y **nodos de unión** a los $\mathbf{b}_{2i}$, $0 \leq i \leq L$.

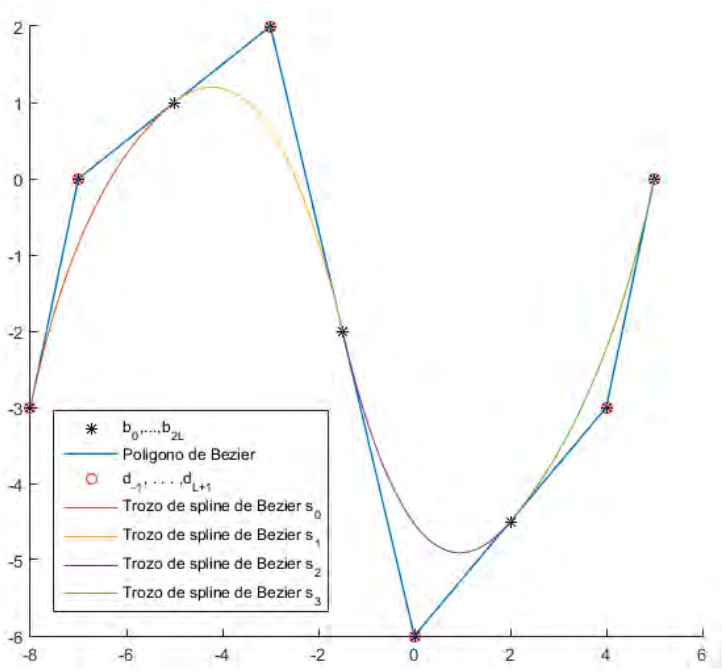


Figura 4.5: Ejemplo de spline de Bézier cuadrático definido mediante cuatro trozos con regularidad $\mathcal{C}^1$.

La curva cuadrática $\mathbf{s}$ está completamente determinada por la sucesión de nodos,

$$\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_3, \ldots, \mathbf{b}_{2i+1}, \ldots, \mathbf{b}_{2L-1}, \mathbf{b}_{2L}, \tag{4.3.1}$$

pues los nodos restantes tienen una única expresión posible, resultante de las condiciones de regularidad $\mathcal{C}^1$ (4.2.4),

$$\mathbf{b}_{2i} = \frac{\triangle_i}{\triangle_{i-1} + \triangle_i}\mathbf{b}_{2i-1} + \frac{\triangle_{i-1}}{\triangle_{i-1} + \triangle_i}\mathbf{b}_{2i+1}, \quad i = 1, \ldots, L-1.$$

De este modo podemos definir una curva de Bézier a trozos cuadrática con regularidad $\mathcal{C}^1$ con menos nodos de los que son necesarios para definir el polígono de Bézier a trozos completo. La mínima cantidad de información necesaria es:

- La sucesión $u_0 < \cdots < u_L,\ u_i \in \mathbb{R},\ i = 0, \ldots, L$.
- El polígono definido por los nodos (4.3.1).

Observación 4.3. *Usualmente se lleva a cabo un cambio de notación y se denota al polígono* (4.3.1) *mediante* $\mathbf{d}_{-1}, \mathbf{d}_0, \ldots, \mathbf{d}_{L-1}, \mathbf{d}_L$.

Los splines de Bézier cuadráticos a trozos con regularidad $\mathcal{C}^1$ presentan una propiedad importante de la que no disfruta una sola curva de Bézier: *el control local*. Se ha podido comprobar en los capítulos anteriores que cuando se trabaja con una curva de Bézier, un

cambio en uno de los vértices de control afecta a la curva entera. Es lo que se denomina un *cambio global*. Esto supone que si la curva de Bézier no se ajusta al contorno que quiere reproducir en una zona determinada, pero sí lo hace en otra, los cambios que se introduzcan en los puntos de control para mejorar la aproximación en la primera zona puedan destruir la buena aproximación ya conseguida en la segunda. Sin embargo, la modificación de un punto de control en un spline cuadrático de Bézier a trozos, va a afectar a lo sumo a tres de los segmentos de curva, lo que permite poder modificar algún punto de control para mejorar el ajuste en las zonas donde se observan mayores discrepancias sin afectar a las zonas en las que ya se ha conseguido un buen ajuste.

Como una consecuencia de esta propiedad de control local, pueden introducirse segmentos rectilíneos dentro de los splines cuadráticos de Bézier a trozos con regularidad $\mathcal{C}^1$, pues si tres puntos de control consecutivos están alineados, el segmento de curva que definen ha de ser lineal. Esto no es posible si en su lugar empleamos una única curva de Bézier de grado elevado, pues no podrá contener fragmentos de recta a no ser que lo sea ella en sí misma. Estos splines se utilizan con frecuencia en el diseño de fuentes, tal y como se verá posteriormente en la Sección 5.1.

4.4. Splines de Bézier cúbicos a trozos con regularidad $\mathcal{C}^2$

Tal y como se hizo en la Sección 4.3, consideramos $u_0 < \cdots < u_L$, $u_i \in \mathbb{R}$, $i = 0, \ldots, L$, $L \in \mathbb{N}$, y los intervalos $[u_i, u_{i+1}]$, $i = 0, \ldots, L-1$, solo que esta vez con el propósito de construir $\mathbf{s} : [u_0, u_L] \longrightarrow \mathbb{R}^2$ spline cúbico de Bézier a trozos con regularidad $\mathcal{C}^2$, de tal manera que para $u \in [u_i, u_{i+1}]$, con $i = 0, \ldots, L-1$ concreto,

$$\mathbf{s}(u) = \mathbf{s}_i(t_i) = \mathbf{s}_i\left(\frac{u - u_i}{u_{i+1} - u_i}\right) = \mathbf{s}_i\left(\frac{u - u_i}{\triangle_i}\right).$$

Por ser cúbico a trozos,

$$\mathbf{s}_i(t_i) = \sum_{j=0}^{3} \mathbf{b}_{3i+j} B_j^3(t_i; u_i, u_{i+1}), \quad i = 0, \ldots, L-1.$$

Veamos qué condiciones deben establecerse sobre los nodos para conseguir la regularidad deseada, y cómo se construyen y relacionan entre sí.

En primer lugar, consideremos dos segmentos de curva adyacentes, digamos $\mathbf{s}_{i-1}$ y $\mathbf{s}_i$ para un $i = 0, \ldots, L-1$ concreto. Para conseguir regularidad $\mathcal{C}^1$ en u_i, los nodos adyacentes han de estar en la proporción $\triangle_{i-1} : \triangle_i$, es decir,

$$\mathbf{b}_{3i} = \frac{\triangle_i}{\triangle_{i-1} + \triangle_i} \mathbf{b}_{3i-1} + \frac{\triangle_{i-1}}{\triangle_{i-1} + \triangle_i} \mathbf{b}_{3i+1}, \quad i = 1, \ldots, L-1. \tag{4.4.1}$$

Para obtener regularidad $\mathcal{C}^2$, ha de existir un punto auxiliar $\mathbf{d}_i$ de tal manera que los nodos $\mathbf{b}_{3i-2}, \mathbf{b}_{3i-1}, \mathbf{d}_i$ y $\mathbf{d}_i, \mathbf{b}_{3i+1}, \mathbf{b}_{3i+2}$ estén en la misma proporción $\triangle_{i-1} : \triangle_i$, tal y como se deduce de las condiciones (4.2.5).

Es por esto que, para $i = 1, \ldots, L-1$,

$$\mathbf{b}_{3i-1} = \left(\frac{\triangle_i}{\triangle_{i-1}+\triangle_i}\right)\mathbf{b}_{3i-2} + \left(\frac{\triangle_{i-1}}{\triangle_{i-1}+\triangle_i}\right)\mathbf{d}_i, \tag{4.4.2}$$

$$\mathbf{b}_{3i-2} = \left(\frac{\triangle_{i-1}}{\triangle_{i-2}+\triangle_{i-1}}\right)\mathbf{d}_{i-1} + \left(\frac{\triangle_{i-2}}{\triangle_{i-2}+\triangle_{i-1}}\right)\mathbf{b}_{3i-1}. \tag{4.4.3}$$

Reescribiendo (4.4.3) como

$$\mathbf{b}_{3i-1} = \left(\frac{\triangle_{i-1}+\triangle_{i-2}}{\triangle_{i-2}}\right)\mathbf{b}_{3i-2} + \left(\frac{\triangle_{i-1}}{\triangle_{i-2}}\right)\mathbf{d}_{i-1},$$

y restando (4.4.3) de (4.4.2), obtenemos que

$$\mathbf{b}_{3i-2}\left(\frac{\triangle_{i-1}+\triangle_{i-2}}{\triangle_{i-2}} - \frac{\triangle_i}{\triangle_{i-1}+\triangle_i}\right) = \frac{\triangle_{i-1}}{\triangle_i+\triangle_{i-1}}\mathbf{d}_i + \frac{\triangle_{i-1}}{\triangle_{i-2}}\mathbf{d}_{i-1},$$

o lo que es equivalente,

$$\mathbf{b}_{3i-2}\left(\frac{\triangle_{i-1}}{\triangle_{i-2}}\right)\left(\frac{\triangle_i+\triangle_{i-1}+\triangle_{i-2}}{\triangle_{i-1}+\triangle_i}\right) = \frac{\triangle_{i-1}}{\triangle_i+\triangle_{i-1}}\mathbf{d}_i + \frac{\triangle_{i-1}}{\triangle_{i-2}}\mathbf{d}_{i-1}.$$

Llamando,

$$\Gamma_i = \triangle_i + \triangle_{i-1} + \triangle_{i-2} \tag{4.4.4}$$

y despejando $\mathbf{b}_{3i-2}$ obtenemos que,

$$\mathbf{b}_{3i-2} = \frac{\triangle_{i-2}}{\Gamma_i}\mathbf{d}_i + \frac{\triangle_{i-1}+\triangle_i}{\Gamma_i}\mathbf{d}_{i-1}. \tag{4.4.5}$$

Finalmente, introducimos este valor en (4.4.2) para obtener así

$$\mathbf{b}_{3i-1} = \frac{\triangle_{i-2}+\triangle_{i-1}}{\Gamma_i}\mathbf{d}_i + \frac{\triangle_i}{\Gamma_i}\mathbf{d}_{i-1}. \tag{4.4.6}$$

Juntando las condiciones (4.4.1), (4.4.5) y (4.4.6), se consiguen las expresiones de todos

los nodos del polígono del spline de Bézier cúbico a trozos,

$$
\begin{aligned}
\mathbf{b}_0 &= \mathbf{d}_{-1}, \\
\mathbf{b}_1 &= \mathbf{d}_0, \\
\mathbf{b}_2 &= \frac{\triangle_1}{\triangle_0+\triangle_1}\mathbf{d}_0 + \frac{\triangle_0}{\triangle_1+\triangle_0}\mathbf{d}_1, \\
&\ldots \\
\mathbf{b}_{3i} &= \frac{\triangle_i}{\triangle_{i-1}+\triangle_i}\mathbf{b}_{3i-1} + \frac{\triangle_{i-1}}{\triangle_{i-1}+\triangle_i}\mathbf{b}_{3i+1}, \qquad i = 1,\ldots,L-1, \\
\mathbf{b}_{3i-1} &= \frac{\triangle_{i-2}+\triangle_{i-1}}{\Gamma_i}\mathbf{d}_i + \frac{\triangle_i}{\Gamma_i}\mathbf{d}_{i-1}, \qquad i = 2,\ldots,L-1, \\
\mathbf{b}_{3i-2} &= \frac{\triangle_{i-2}}{\Gamma_i}\mathbf{d}_i + \frac{\triangle_{i-1}+\triangle_i}{\Gamma_i}\mathbf{d}_{i-1}, \qquad i = 2,\ldots,L-1, \\
&\ldots \\
\mathbf{b}_{3L-2} &= \frac{\triangle_{L-1}}{\triangle_{L-1}+\triangle_{L-2}}\mathbf{d}_{L-1} + \frac{\triangle_{L-2}}{\triangle_{L-1}+\triangle_{L-2}}\mathbf{d}_L, \\
\mathbf{b}_{3L-1} &= \mathbf{d}_L, \\
\mathbf{b}_{3L} &= \mathbf{d}_{L+1},
\end{aligned}
\tag{4.4.7}
$$

empleando la misma notación que la mencionada en la Observación 4.3. De manera análoga al caso de splines de Bézier cuadráticos a trozos con regularidad $\mathcal{C}^1$, la partición del espacio parámetrico, $u_0 < \cdots < u_L$, junto al polígono formado por $\mathbf{d}_{-1},\ldots,\mathbf{d}_{L+1}$, determinan completamente los splines de Bézier cúbicos a trozos con regularidad $\mathcal{C}^2$.

La Figura 4.6 muestra las proporciones de los parámetros de un spline cúbico de Bézier a trozos con regularidad $\mathcal{C}^2$. En ella aparecen, además de los parámetros, los nodos $\mathbf{b}_0,\ldots,\mathbf{b}_{3L}$ en negro y el polígono que los une en naranja, los nodos $\mathbf{d}_{-1},\ldots,\mathbf{d}_{L+1}$ en rojo y el polígono que los une en azul. Los trozos de curva en diferentes colores.

La Figura 4.7 muestra un ejemplo de spline de Bézier cúbico con 3 trozos y regularidad $\mathcal{C}^2$. En negro los nodos $\mathbf{b}_0,\ldots,\mathbf{b}_9$ y el polígono que los une en azul. En rojo los nodos $\mathbf{d}_{-1},\ldots,\mathbf{d}_4$ y el polígono que los une en naranja. Los trozos de curva en diferentes colores.

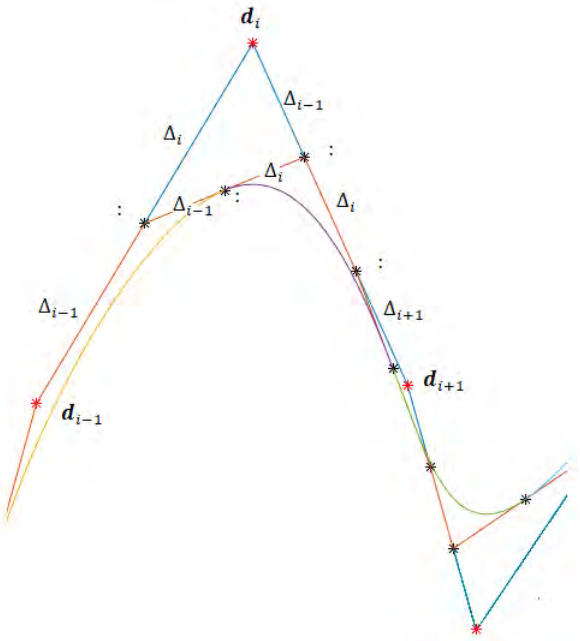


Figura 4.6: Ejemplo que muestra las proporciones de los parámetros en un spline cúbico a trozos con regularidad $\mathcal{C}^2$.

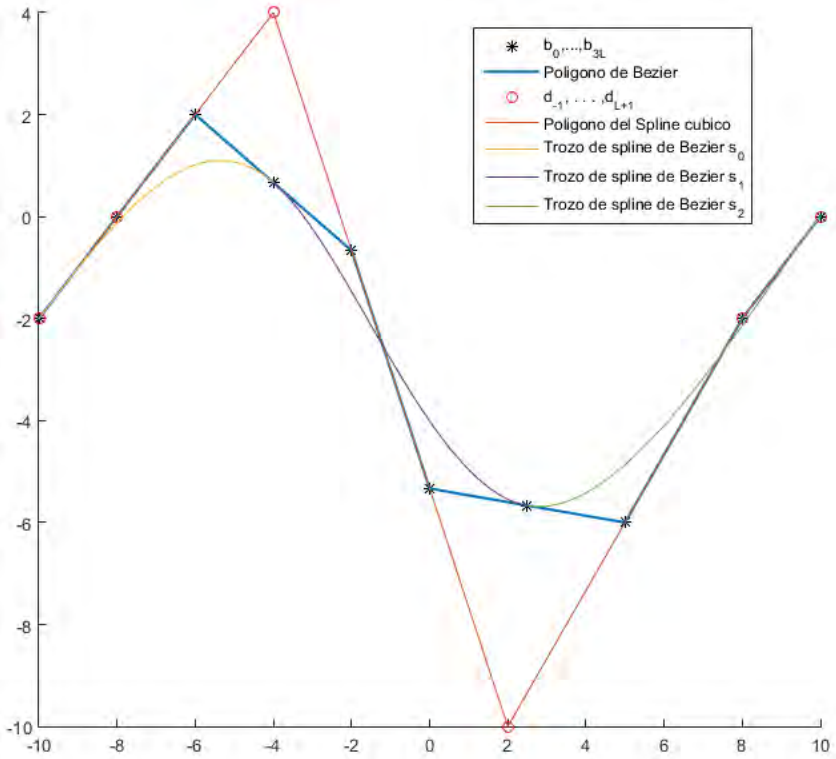


Figura 4.7: Ejemplo de spline de Bézier cúbico definido mediante tres trozos con regularidad $\mathcal{C}^2$.

Capítulo 5

Ejemplos prácticos: diseño mediante splines de Bézier a trozos

En los capítulos anteriores, se ha mencionado la utilidad de las curvas de Bézier y de los splines de Bézier a trozos en el diseño gráfico y asistido por ordenador. Si bien es cierto que las funciones `bez_cubatrozos.m` y `bspline_cubico.m` programadas en MATLAB® (ver Funciones 11, 12 del Apéndice A), cuyas primeras líneas son,

```
function [] = bez_cubatrozos(b,u);
function [] = bspline_cubico(d,u);
```

permiten aproximar diseños mediante splines de Bézier cúbicos a trozos (en el caso de `bspline_cubico.m`, con regularidad $\mathcal{C}^2$), es necesario conocer una aproximación previa de los nodos $\mathbf{b}_0, \ldots, \mathbf{b}_{3L}$, en el caso de la primera función y de $\mathbf{d}_{-1}, \mathbf{d}_0, \ldots, \mathbf{d}_L, \mathbf{d}_{L+1}$, en el caso de la segunda, para poder efectuar la interpolación de los mismos.

Existen diversas herramientas que permiten elegir interactivamente dichos nodos, como por ejemplo, las funciones del paquete MuPAD de MATLAB®. A su vez, el programa GeoGebra resulta ser especialmente rápido y sencillo, pues al arrastrar cualquier nodo por la pantalla, actualiza su valor automáticamente. De este modo, podremos considerar puntos en las figuras a interpolar, definir en ellos splines de Bézier a trozos, y desplazar manualmente los nodos hasta que las curvas se aproximen lo máximo posible al contorno del dibujo. El programa GeoGebra se ha utilizado, por tanto, como herramienta auxiliar para determinar los nodos de los polígonos, pero se ha utilizado MATLAB® para construir funciones que construyen y dibujan los splines de Bezier a trozos, a partir de un conjunto de nodos dado.

En este capítulo se muestran dos ejemplos en los que se han utilizado splines de Bézier a trozos con y sin regularidad $\mathcal{C}^2$ para aproximar dos curvas dadas:

1. El contorno de la letra **S** de la tipografía Times New Roman© (Figura 5.1 (a)).
2. Una firma (Figura 5.8).

Observación 5.1. *Durante el resto del capítulo, se considerará* $0 = u_0 < \cdots < u_L = L$. *Se ha considerado de esta forma para que* $\triangle_i = u_{i+1} - u_i = i + 1 - i = 1,\ i = 0, \ldots, L-1$. *Para cada trozo de curva se considerará el parámetro* t_i *definido en la ecuación* (4.1.1), *de tal manera que para* $u \in [i, i+1]$ *se tendrá que* $t_i \in [0, 1]$.

5.1. Splines de Bézier a trozos al diseñar la letra S

Antes de iniciar el proceso de interpolación, es necesario destacar que en la letra que queremos representar sólo existen unos pocos puntos angulosos en los extremos de la figura (ver Figura 5.1 (a)), por lo que buscaremos regularidad en la curva aproximante.

5.1.1. Aproximación mediante splines de Bézier cúbicos considerando 24 nodos, sin imponer condiciones de regularidad

Comenzamos construyendo un spline de Bézier cúbico formado por 6 trozos para recuperar el contorno curvo de la letra S. Para ello fijamos 6 polígonos de Bézier de manera que cada uno esté formado por dos puntos sobre el contorno de la letra, que denotaremos por A y que serán los extremos de cada curva, y por dos puntos alejados del contorno, que denotaremos por B. Tendremos entonces dos grupos de 10 nodos según la secuencia $ABBABBABBA$, para determinar dos splines de Bézier cúbicos de 3 trozos cada uno, y cuatro nodos más para los segmentos rectilíneos representados en color azul en la Figura 5.1 (b), lo que hace un total de 24 nodos. Definiendo un segmento de curva de Bézier para cada cuaterna de nodos mediante el comando de GeoGebra

```
Bi_{x}= {x(b_{i}),x(b_{i+1}),x(b_{i+2}),x(b_{i+3})}
Bi_{y}= {y(b_{i}),y(b_{i+1}),y(b_{i+2}),y(b_{i+3})}
elemento(Bi_{x},1)*(1-x)^3+
elemento(Bi_{x},2)*3*(1-x)^2*x+
elemento(Bi_{x},3)*3*(1-x)*x^2+
elemento(Bi_{x},4)*x^3

elemento(Bi_{y},1)*(1-x)^3+
elemento(Bi_{y},2)*3*(1-x)^2*x+
elemento(Bi_{y},3)*3*(1-x)*x^2+
elemento(Bi_{y},4)*x^3

curva(fi(u),gi(u),u,0,1)
```

obtenemos la Figura 5.1 (c).

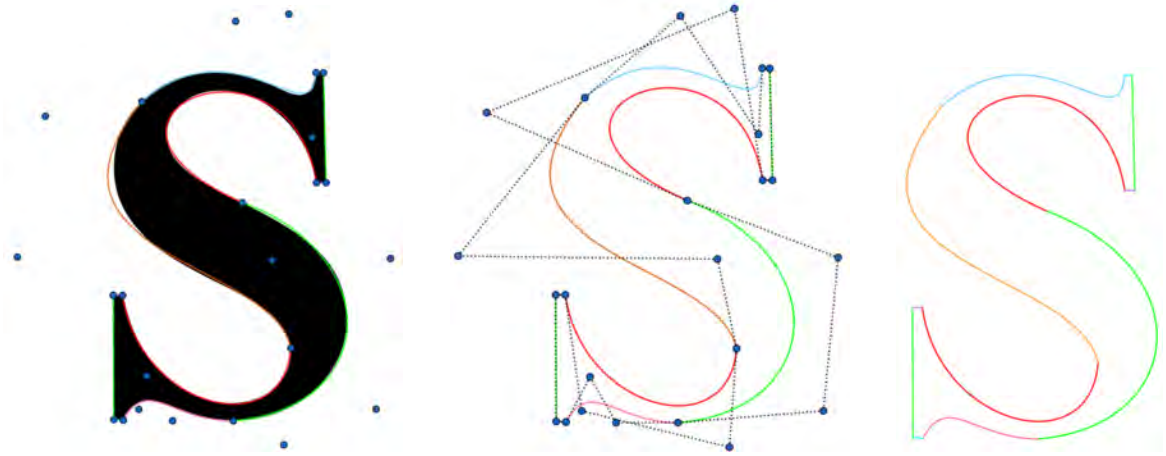
Los segmentos de curva obtenidos apenas se asemejan al contorno original. Sin embargo, utilizando las ya mencionadas facilidades que ofrece el programa GeoGebra, hemos desplazado las posiciones de los puntos de control de la forma B (los que no están sobre el contorno de la letra) para obtener una aproximación mejor, como muestra la Figura 5.2 (a).

(a) Imagen de partida.

(b) Selección de los nodos según la secuencia $ABBABBABBA$.

(c) Comienzo del proceso de interpolación mediante splines de Bézier cúbicos empleando 24 nodos.

Figura 5.1: Inicio del proceso de interpolación.



(a) Imagen inicial junto a los puntos de control (en azul) y al spline de Bézier cúbico a trozos que interpola a la letra S.

(b) Spline de Bézier cúbico a trozos junto al polígono de Bézier asociado.

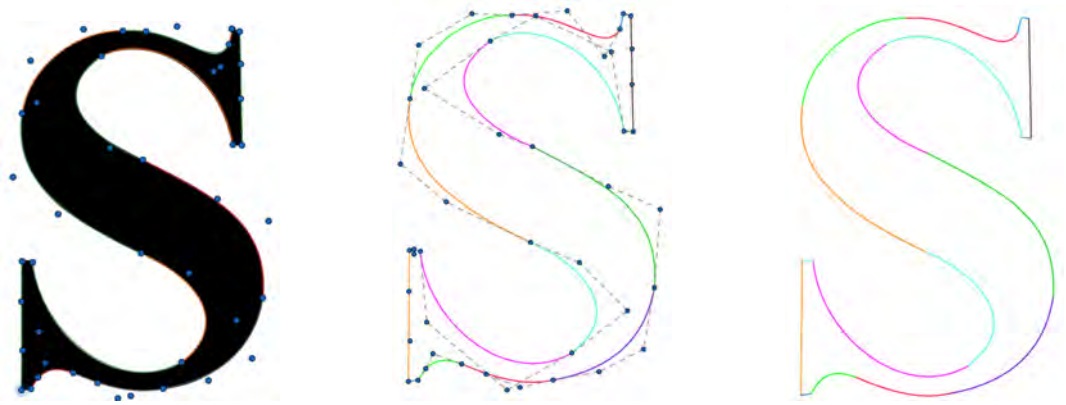
(c) Spline de Bézier cúbico a trozos que interpola a la letra S.

Figura 5.2: Interpolación mediante splines de Bézier cúbicos a trozos considerando 24 nodos, sin imponer condiciones de regularidad, tras modificar los nodos tipo B.

La aproximación conseguida es bastante similar a la letra original, pero ¿y si repetimos el proceso anterior aumentando esta vez el número de nodos considerado?

5.1.2. Aproximación mediante splines de Bézier cúbicos a trozos considerando 40 nodos, sin imponer condiciones de regularidad

Si consideramos 40 nodos en lugar de 24, obtenemos como resultado la Figura 5.3.



(a) Imagen inicial junto a los puntos de control (en azul) y al spline de Bézier cúbico a trozos que interpola a la letra S.

(b) Spline cúbico de Bézier a trozos junto al polígono de Bézier asociado.

(c) Spline cúbico de Bézier a trozos que interpola a la letra S.

Figura 5.3: Interpolación mediante splines de Bézier cúbicos a trozos considerando 40 nodos, sin imponer condiciones de regularidad.

La Figura 5.3(a) muestra una aproximación notablemente mejor que la anterior (es lógico, pues hemos efectuado un proceso de subdivisión) y aparentemente buena. Por esta razón podríamos pensar, ¿y si en vez de emplear splines de Bézier **cúbicos**, optamos por emplear splines de Bézier **cuadráticos** para aproximar la imagen?

5.1.3. Aproximación mediante splines de Bézier cuadráticos a trozos considerando 35 nodos, sin imponer condiciones de regularidad

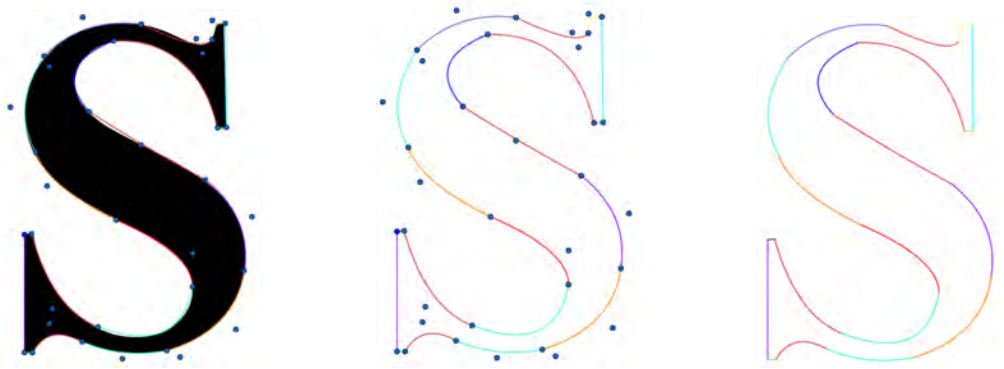
En este caso, consideramos 35 nodos según la secuencia $ABABA, \ldots, ABA$ (el polígono de Bézier de cada segmento de curva tiene ahora tres puntos de control: dos sobre la curva y uno exterior), y ejecutamos para cada trío de nodos, el comando

```
Bi_{x}= {x(b_{i}),x(b_{i+1}),x(b_{i+2})}
Bi_{y}= {y(b_{i}),y(b_{i+1}),y(b_{i+2})}
elemento(Bi_{x},1)*(1-x)^2+
elemento(Bi_{x},2)*2*(1-x)*x+
elemento(Bi_{x},4)*x^2

elemento(Bi_{y},1)*(1-x)^2+
elemento(Bi_{y},2)*2*(1-x)*x+
elemento(Bi_{y},4)*x^2

curva(fi(u),gi(u),u,0,1)
```

obteniendo así la Figura 5.4. De nuevo se han representado en color azul los puntos de control y se han utilizado distintos colores para los diferentes trozos de curva.



(a) Imagen inicial junto a los puntos de control (en azul) y al spline de Bézier cuadrático a trozos que interpola a la letra S.

(b) Spline de Bézier cuadrático a trozos junto al polígono de Bézier asociado.

(c) Spline de Bézier cuadrático a trozos que interpola a la letra S.

Figura 5.4: Interpolación mediante splines de Bézier cuadráticos a trozos considerando 35 nodos, sin imponer condiciones de regularidad.

En este caso, comienzan a apreciarse puntos angulosos nada deseables en el diseño de esta letra. No parece que la misma se haya llevado a cabo mediante un único trazo tal y como cabría esperar de la imagen original. Parece que bajar el grado no ha proporcionado un resultado favorable, ¿y subirlo?

5.1.4. Aproximación mediante splines de Bézier de grado 6 definidos a trozos considerando 18 nodos, sin imponer condiciones de regularidad

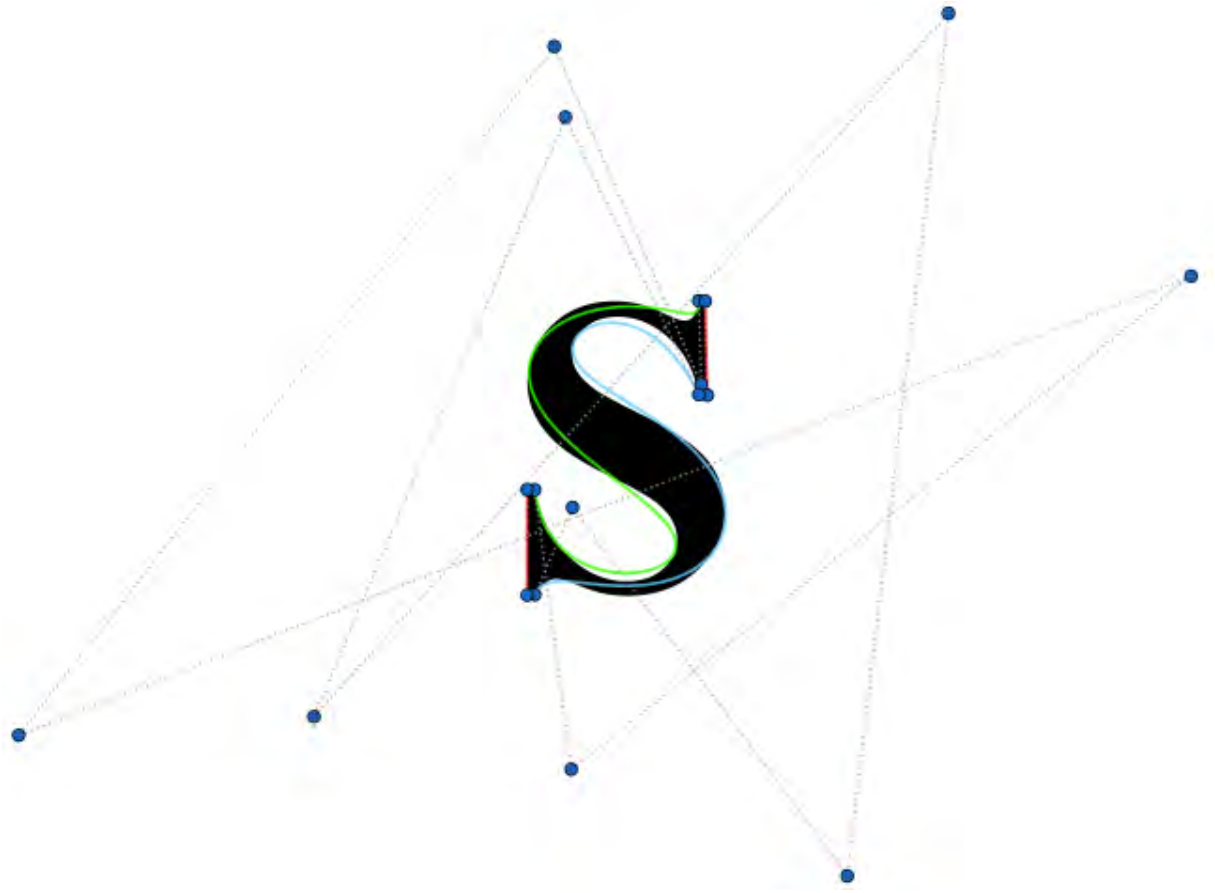
Considerando en este caso la sucesión de nodos de la forma $ABBBBBA, \ldots, ABBBBBA$, y generando un trozo de curva de Bézier de grado 6 para cada agrupación de siete nodos, se obtiene la Figura 5.5. En primer lugar, se aprecia que los puntos exteriores que determinan los dos polígonos de Bézier utilizados, estan mucho más alejados de la imagen que en los casos anteriores, y en segundo lugar vemos que la aproximación es notablemente peor. Se trata de un claro ejemplo de cómo el empleo de curvas de grado elevado no proporciona mejores resultados que curvas de orden más bajo en el proceso de interpolación (tal y como se mencionó al inicio del Capítulo 4).

Sin embargo, sí se observa la regularidad deseada al comienzo del capítulo, pues el trazo de la curva es suave y no presenta puntos angulosos. Lamentablemente, la similitud entre el contorno de la figura y el spline es menor que en cualquiera de los casos anteriores. Este hecho está directamente relacionado con el carácter global de las curvas de Bézier, pues al modificar levemente cualquiera de los nodos del tipo B en un segmento de curva, todo el spline se ve afectado, lo que supone una complicación añadida si la curva es de grado alto y la figura a describir es compleja como la que nos ocupa.

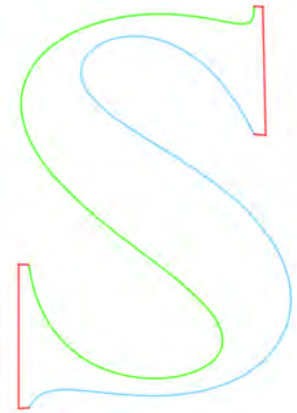
En las aproximaciones anteriores llevadas a cabo mediante splines de Bézier a trozos para varios grados distintos, menos en la que emplea dos curvas de grado seis, se ha llegado a conseguir una aproximación bastante fiel de la curva que describe el contorno de la letra S. Sin embargo, a excepción de la mostrada en la Figura 5.5 (la cual ha sido descartada por no ajustarse suficiente a la imagen inicial), en todas ellas se puede apreciar falta de regularidad en los nodos del tipo A. En las Figuras 5.2, 5.3 y 5.4 se observa que en los puntos donde se produce cambio de color las dos curvas que se juntan lo hacen formando pequeños ángulos. El spline de Bézier empleado en la siguiente sección cumple las condiciones de regularidad y aproximación deseadas desde el comienzo del epígrafe.

5.1.5. Aproximación mediante splines de Bézier cúbicos a trozos, imponiendo condiciones de regularidad $\mathcal{C}^2$

En este caso, comenzamos eligiendo una serie de nodos con el cursor sin secuencia alguna, y definiendo el resto a partir de los iniciales mediante la relaciones especificadas en (4.4.7) para que el spline de Bézier definido a trozos resultante, presente regularidad $\mathcal{C}^2$. Así, en la Figura 5.6 (a) están representados los nodos que se han escogido manualmente en color azul (es decir, $\mathbf{d}_{-1}, \mathbf{d}_0, \ldots, \mathbf{d}_L, \mathbf{d}_{L+1}$), los nodos que se han generado mediante las condiciones (4.4.7) en color gris (es decir, $\mathbf{b}_0, \ldots, \mathbf{b}_{3L}$), el polígono asociado al spline cúbico de Bézier a trozos con regularidad $\mathcal{C}^2$ en color rosa y los segmentos de curva en diferentes colores. El resultado final es muy satisfactorio tanto en regularidad como en aproximación. Además permite un control local y no global como ocurría en las subsecciones anteriores.



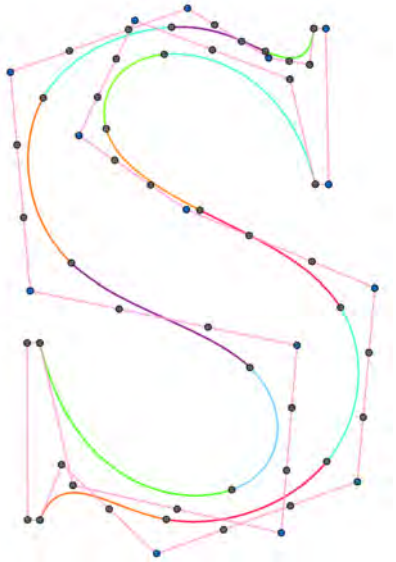
(a) Imagen inicial junto a los puntos de control (en azul) y al spline de Bézier de grado 6 a trozos que interpola a la letra S.



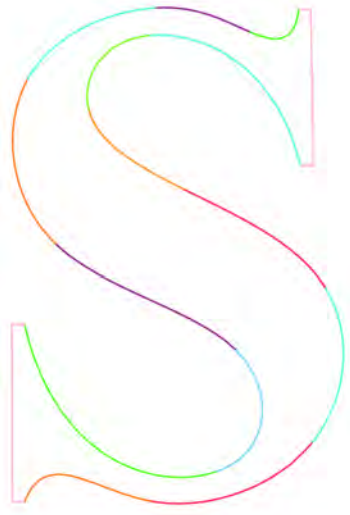
(b) Spline de Bézier de grado 6 definido a trozos que interpola a la letra S.

Figura 5.5: Interpolación mediante splines de Bézier de grado 6 definidos a trozos considerando 18 nodos, sin imponer condiciones de regularidad.

(a) Imagen inicial junto a los nodos $\mathbf{b}_0, \ldots, \mathbf{b}_{3L}$ en gris, a los nodos $\mathbf{d}_{-1}, \mathbf{d}_0, \ldots, \mathbf{d}_L, \mathbf{d}_{L+1}$ en azul, al polígono de Bézier en rosa y al spline de Bézier cúbico a trozos con regularidad $\mathcal{C}^2$ en diferentes colores.



(b) Nodos $\mathbf{b}_0, \ldots, \mathbf{b}_{3L}$ en gris, a los nodos $\mathbf{d}_{-1}, \mathbf{d}_0, \ldots, \mathbf{d}_L, \mathbf{d}_{L+1}$ en azul, al polígono de Bézier en rosa y al spline de Bézier cúbico a trozos con regularidad $\mathcal{C}^2$ en diferentes colores, sin mostrar la imagen inicial.



(c) Spline de Bézier cúbico a trozos con regularidad $\mathcal{C}^2$ que interpola a la letra S.

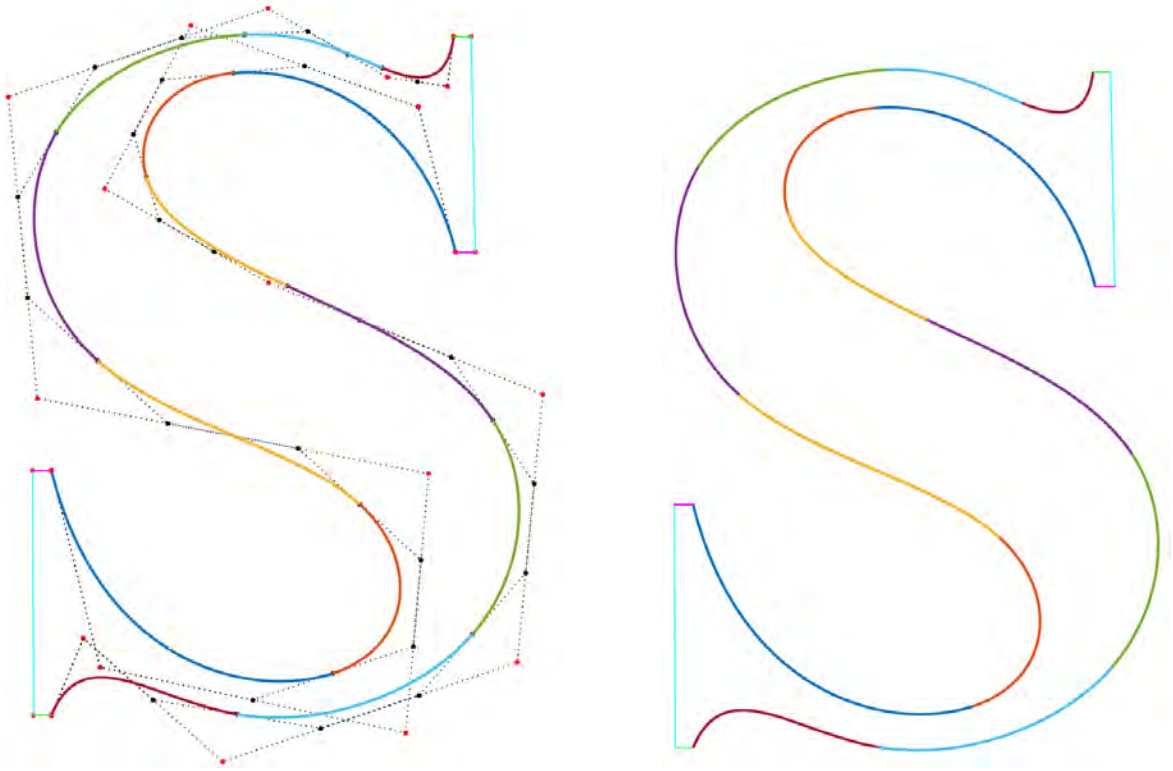
Figura 5.6: Interpolación mediante splines de Bézier cúbicos a trozos, imponiendo condiciones de regularidad $\mathcal{C}^2$.

5.1.6. Mediante funciones en MATLAB®

Una vez obtenida la aproximación descrita en la Subsección 5.1.5, hemos exportado el valor de los nodos $\mathbf{d}_{-1}, \mathbf{d}_0, \ldots, \mathbf{d}_L, \mathbf{d}_{L+1}$, que en la Figura 5.6 aparecen en color azul. Hemos programado en MATLAB® la función `bspline_cubico.m` cuya primera línea es

```
function [] = bspline_cubico(d,u);
```

(ver el código en la Función 12 del Apéndice A) y la hemos ejecutado con dichos valores, para obtener la Figura 5.7, la cual muestra una muy buena aproximación a la imagen inicial (es la misma aproximación que aparece en la Figura 5.6).



(a) Nodos $\mathbf{b}_0, \ldots, \mathbf{b}_{3L}$ en negro, los nodos $\mathbf{d}_{-1}, \mathbf{d}_0, \ldots, \mathbf{d}_L, \mathbf{d}_{L+1}$ en rojo, el polígono de Bézier en gris punteado y el spline de Bézier cúbico a trozos con regularidad $\mathcal{C}^2$ en diferentes colores.

(b) Spline de Bézier cúbico a trozos con regularidad $\mathcal{C}^2$ que interpola a la letra S.

Figura 5.7: Interpolación mediante splines de Bézier cúbicos a trozos, imponiendo condiciones de regularidad $\mathcal{C}^2$ exportando los nodos $\mathbf{d}_{-1}, \mathbf{d}_0, \ldots, \mathbf{d}_L, \mathbf{d}_{L+1}$ de la Figura 5.6 e introduciéndolos en la función `bspline_cubico.m` programada en MATLAB® (ver Anexo (A, función 12).

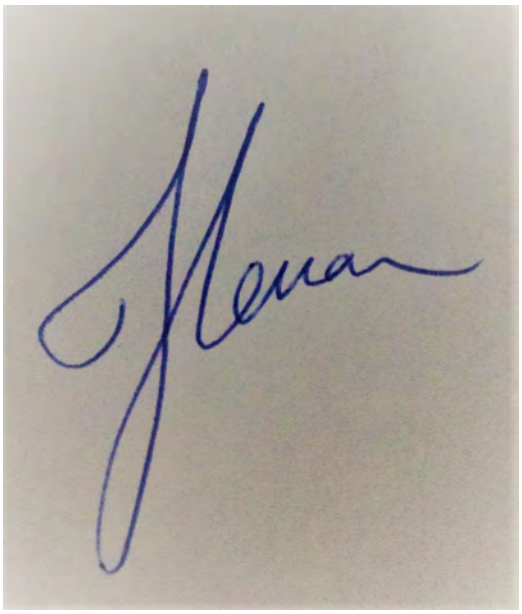


Figura 5.8: Ejemplo de firma a interpolar

5.2. Splines de Bézier a trozos al diseñar una firma

Las curvas de Bézier y los splines resultan de gran utilidad a la hora de trasladar un dibujo esbozado en un papel, como pueda ser un anagrama, un logotipo o una firma, a formato digital. Tratemos de interpolar mediante estas la firma que se muestra en la Figura 5.8. Al tratarse de una firma efectuada mediante un único trazo, parece intuitivo buscar una cierta regularidad en el spline interpolador.

5.2.1. Aproximación mediante splines de Bézier cúbicos a trozos considerando 39 nodos, sin imponer condiciones de regularidad

De forma análoga a la narrada en la Sección 5.1.2, la ya mencionada utilidad del programa GeoGebra, permite fácilmente la recolocación de los nodos del tipo B para que los trozos de curva de Bézier se ajusten en mayor medida a la figura, tal y como muestra la Figura 5.10. Tras este ajuste, se obtiene la interpolación mediante splines de Bézier cúbicos a trozos, que se muestra en la Figura 5.11.

La Figura 5.11, pese a que sí se asemeja a la Figura 5.8, presenta falta de regularidad en las confluencias de algunos segmentos de curva. Con el propósito de conseguir regularidad $\mathcal{C}^2$ en la curva resultante, procedemos a interpolar mediante **splines de Bézier cúbicos a trozos, con regularidad $\mathcal{C}^2$**.

Figura 5.9: Disposición inicial de los 39 nodos de Bézier.

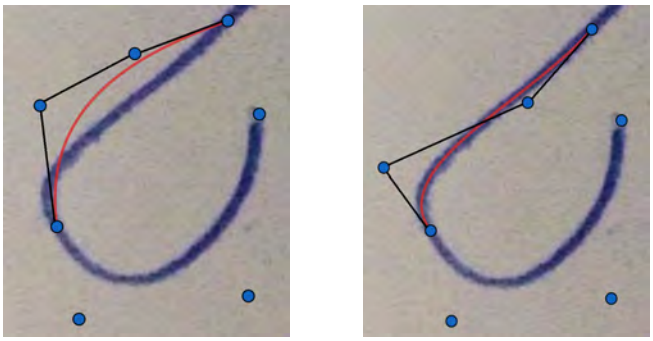


Figura 5.10: Efecto que tiene en los segmentos de curva de Bézier el desplazamiento de los nodos internos del correspondiente polígono para conseguir un mayor ajuste al trazo del dibujo.

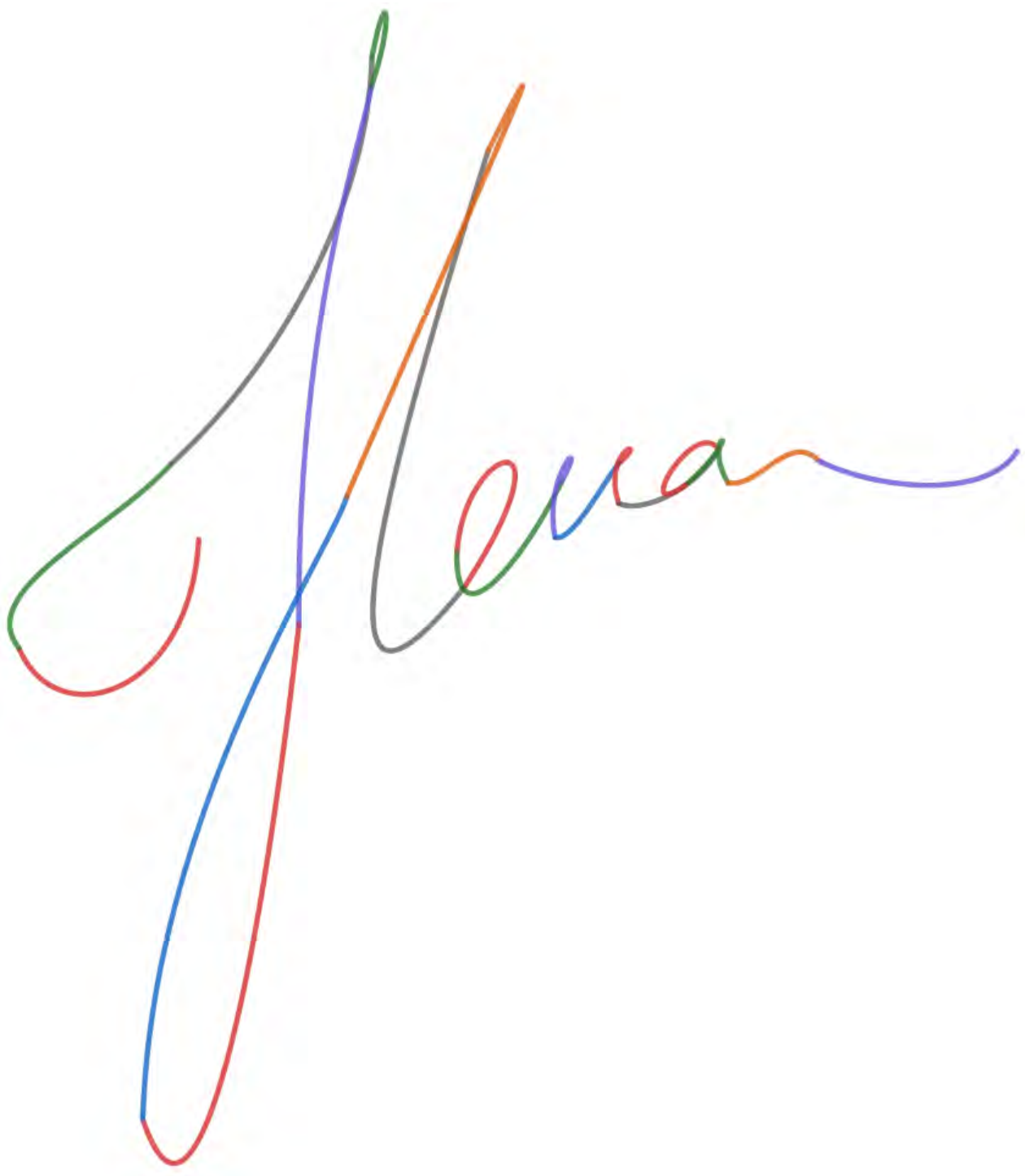


Figura 5.11: Interpolación mediante splines de Bézier cúbicos a trozos sin imponer condiciones de regularidad (observar los ángulos que se forman en los puntos en los que se produce cambio de color).

5.2.2. Aproximación mediante splines de Bézier cúbico a trozos imponiendo condiciones de regularidad $\mathcal{C}^2$

Partimos de los puntos azules $\mathbf{d}_{-1}, \mathbf{d}_0, \ldots, \mathbf{d}_L, \mathbf{d}_{L+1}$, y determinamos, de manera única, los nodos grises $\mathbf{b}_0, \ldots, \mathbf{b}_n$, mediante las condiciones establecidas en (4.4.7). El resultado final es el que se muestra en la Figura 5.12.

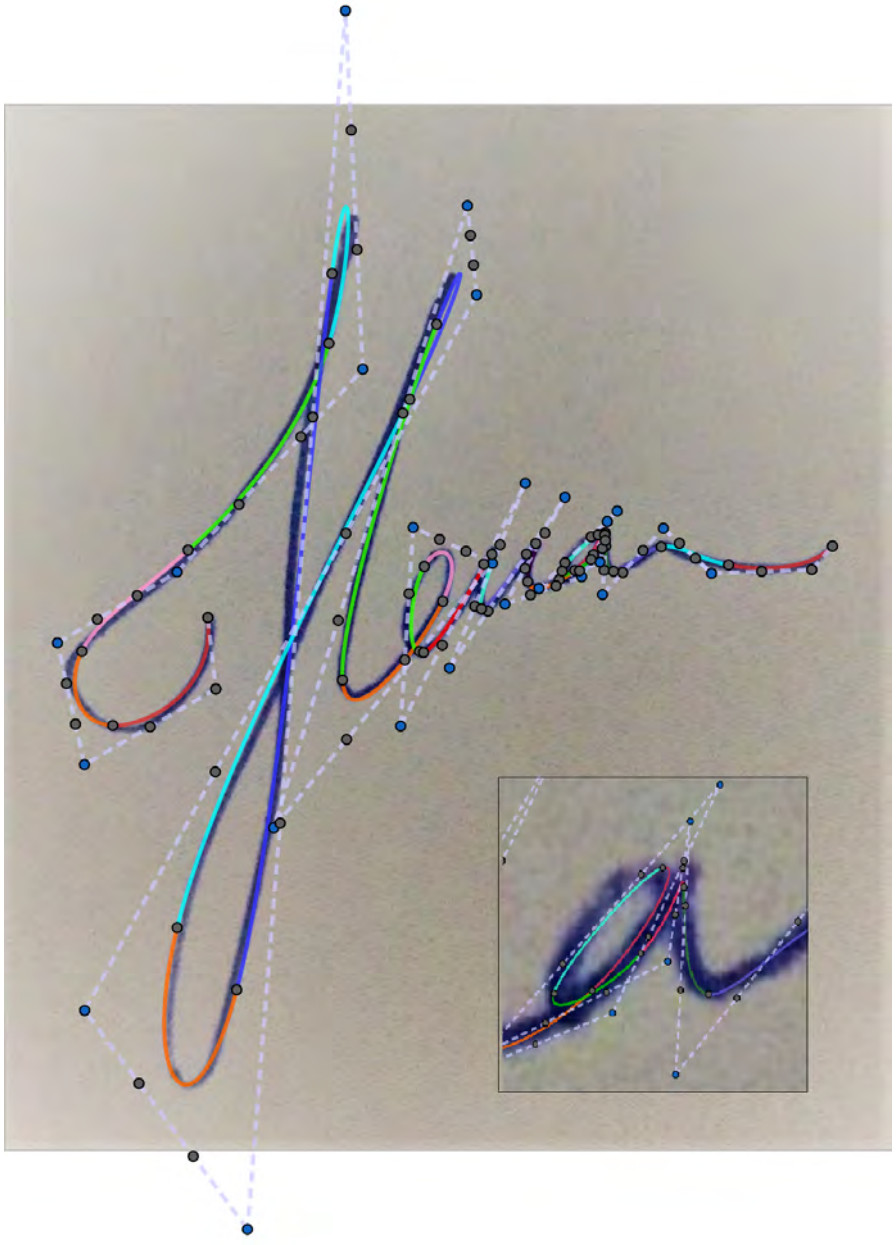


Figura 5.12: Imagen original junto al polígono y spline cúbico de Bézier a trozos con regularidad $\mathcal{C}^2$ que aproxima la firma.

5.2.3. Mediante funciones en MATLAB®

Una vez obtenida la aproximación descrita en la Subsección 5.2.2, hemos exportado el valor de los nodos $\mathbf{d}_{-1}, \mathbf{d}_0, \ldots, \mathbf{d}_L, \mathbf{d}_{L+1}$, que en la Figura 5.12 aparecen en color azul y ejecutamos la función `bspline_cubico.m` de MATLAB® (ver el código en la función 12 del Apéndice A) con dichos valores, para obtener la Figura 5.13, la cual muestra la misma aproximación que aparece en la Figura 5.12. La Figura 5.14 muestra esta misma aproximación pero mostrando también el polígono asociado al spline cúbico de Bézier con regularidad $\mathcal{C}^2$ que aproxima la firma.

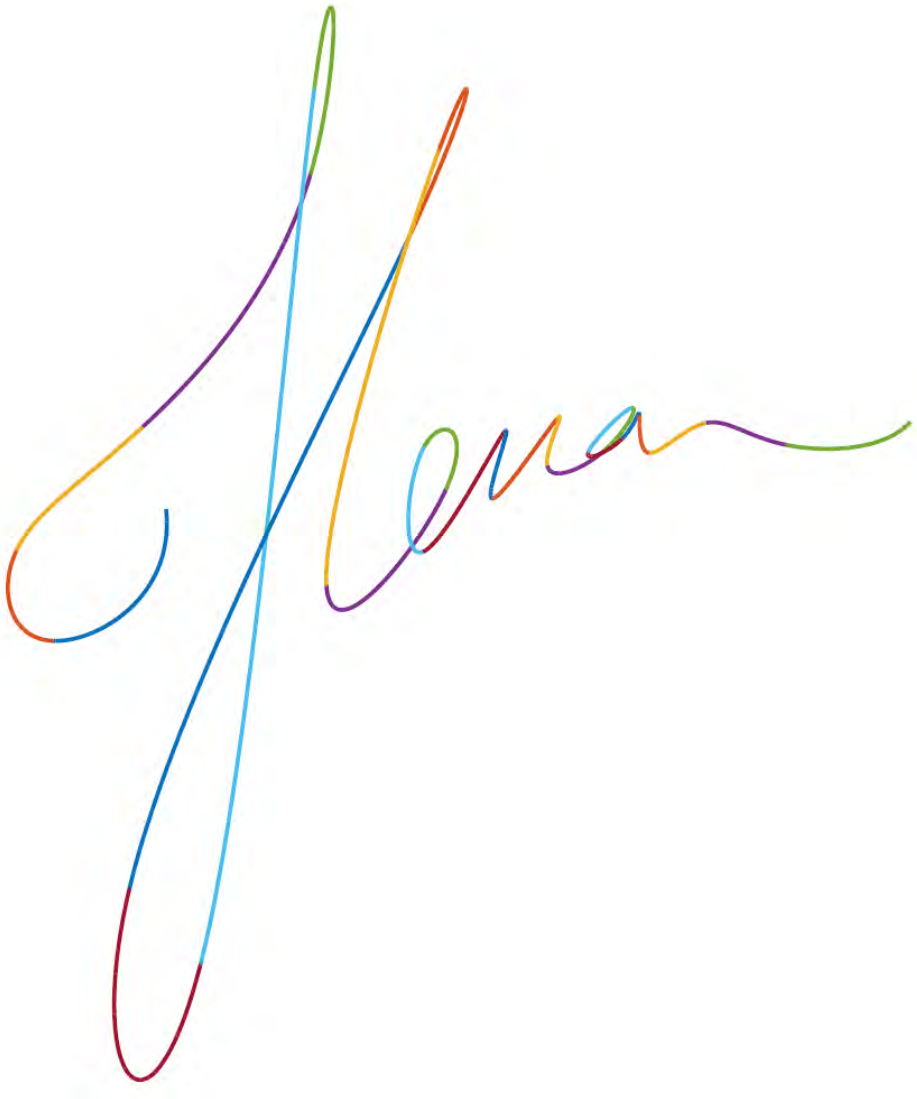


Figura 5.13: Spline cúbico de Bézier a trozos con regularidad $\mathcal{C}^2$ que aproxima la firma, empleando la función de MATLAB® `bspline_cubico.m` (ver Anexo A, Función 12).

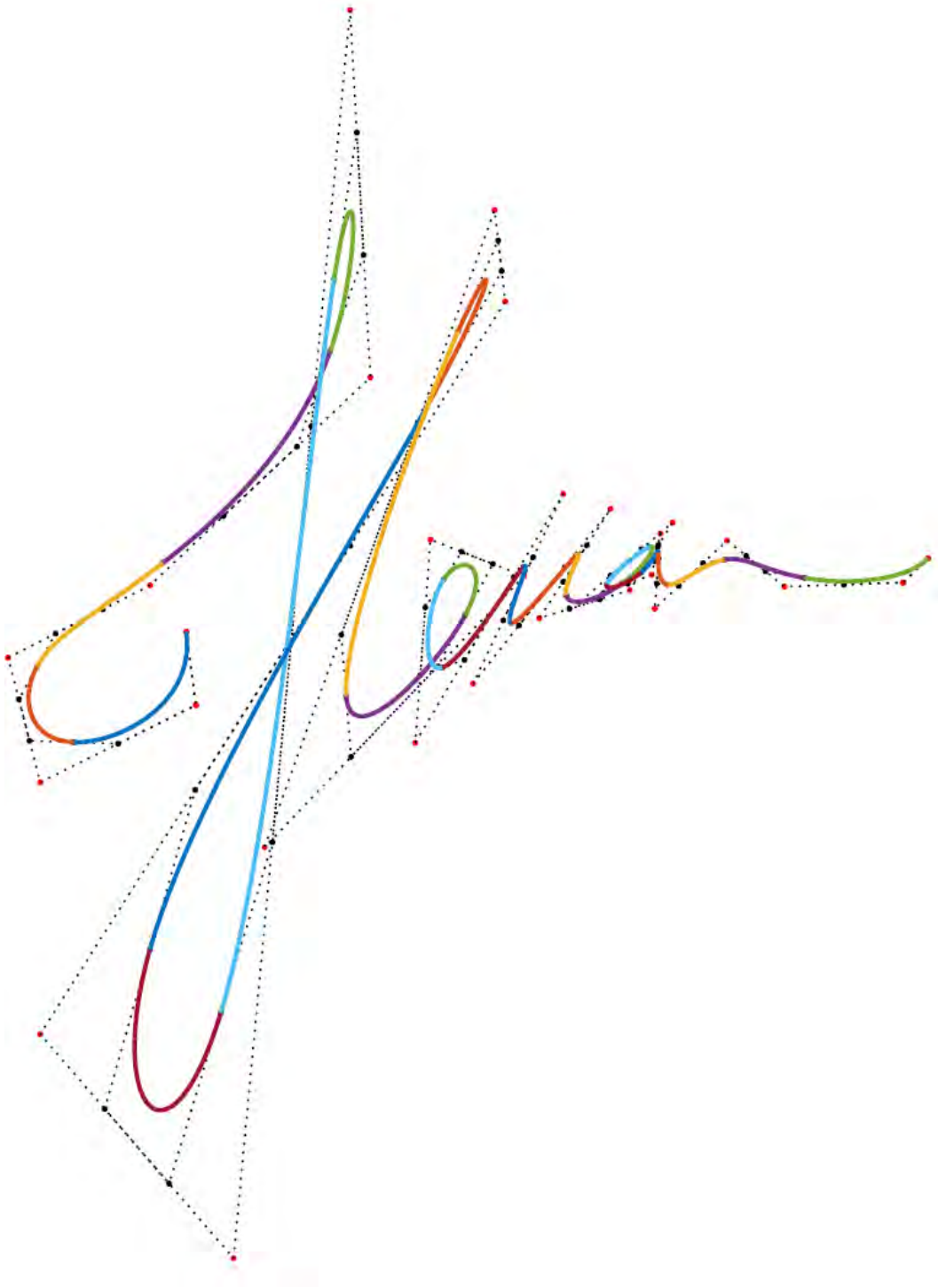


Figura 5.14: Polígono y spline cúbico de Bézier a trozos con regularidad $\mathcal{C}^2$ que aproximan la firma, empleando la Función de MATLAB® `bspline_cubico.m` (ver Anexo A, función 12). Misma aproximación que en la Figura 5.12

5.2.4. Mediante el programa InkScape

El programa InkScape está destinado explícitamente al diseño gráfico y emplea splines de Bézier cúbicos a trozos para aproximar dibujos. Estos vienen implícitamente definidos, y el diseñador tan solo tiene que escoger los nodos $\mathbf{d}_{-1}, \mathbf{d}_0, \ldots, \mathbf{d}_L, \mathbf{d}_{L+1}$ y ajustarlos adecuadamente al contorno del dibujo. Hemos exportado los nodos $\mathbf{d}_{-1}, \mathbf{d}_0, \ldots, \mathbf{d}_L, \mathbf{d}_{L+1}$, que han sido aproximados con GeoGebra en la Subsección 5.2.2 y ya empleados para hallar las Figuras 5.13 y 5.14, y los hemos introducido en el programa InkScape para obtener las Figuras 5.16 y 5.15, las cuales muestran la misma aproximación solo que en la segunda se ha dejado el polígono asociado al spline cúbico de Bézier que aproxima la firma de manera ilustrativa.

Figura 5.15: Spline cúbico de Bézier a trozos con regularidad $\mathcal{C}^2$ que aproxirma la firma mediante el programa de diseño gráfico InkScape. Como cabría esperar, se ha obtenido el mismo ajuste que en las Figuras 5.12, 5.13 y 5.14, al emplearse para los tres casos el mismo polígono $\mathbf{d}_{-1}, \ldots, \mathbf{d}_{L+1}$.

Figura 5.16: Misma aproximación que la que muestra la Figura 5.15 empleando también el programa de diseño gráfico InkScape pero mostrando junto al spline cúbico de Bézier a trozos con regularidad $\mathcal{C}^2$, el polígono de Bézier asociado.

Bibliografía

Cheney, E. W. (1982). *Introduction to Approximation Theory* (Second ed.). AMS Chelsea Pub.

Deuflhard, P., y Hohmann, A. (2003). *Numerical Analysis in Modern Scientific Computing: An Introduction* (Second ed.). Springer.

Eck, M. (1993). Degree reduction of Bézier curves. *Computer Aided Geometric Design*, *10*(3), 237 - 251.

Epperson, J. F. (1987). On the Runge Example. *The American Mathematical Monthly*, *94*(4), 329 - 341.

Farin, G. (1990). *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide* (Second ed.). Academic Press.

Farin, G. (1997). *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide* (Fourth ed.). Academic Press.

Apéndice A

Código de los programas más significativos en MATLAB®

1. Función que evalúa el polinomio de Bernstein $B_i^n(t;a,b)$ en $t \in [a,b]$ concreto.

```
function [r]=bernstein_pol(a,b,n,i,t);
    r = nchoosek(n,i)*power(b-a,-n)*power(b-t,n-i)*power(t-a,i);
end
```

2. Función que evalúa el polinomio cúbico $\mathbf{P}(t) = \sum_{i=0}^{3} \mathbf{b}_i B_i^3(t;a,b)$ en $t \in [a,b]$ concreto.

```
function [q] = bez_cub(b0,b1,b2,b3,t,a,b);
c = [b0,b1,b2,b3];
q = zeros(2,1);
for i = 0:3,
    q = q + c(:,i+1)*bernstein_pol(a,b,3,i,t);
end
end
```

3. Función que construye los polinomios parciales $\mathbf{b}_i^k(t;a,b),\ i = 0,\ldots,n-k$ siguiendo la fórmula de recurrencia (2.1.1) y devuelve el valor de $\mathbf{b}_0^n(t;a,b)$ para $t \in [a,b]$ concreto.

```
function [r]=valor_algoritmo_ab(c,t,a,b);
%c es la matriz que contiene los nodos de Bezier b0,...,bn
%t valor del parametro a evaluar (t en [a,b]).
n = size(c,2);
mx = zeros(n,n); %Matriz auxiliar para abscisas
my = zeros(n,n); %Matriz auxiliar para ordenadas
mx(:,1)=c(1,n:-1:1);
my(:,1)=c(2,n:-1:1);
lambdat = (t-a)/(b-a);
for j = 2:n
```

```
    for i = j:n
        mx(i,j) = (1-lambdat)*mx(i,j-1)+lambdat*mx(i-1,j-1);
        my(i,j) = (1-lambdat)*my(i,j-1)+lambdat*my(i-1,j-1);
    end
end
r = [mx(n,n);my(n,n)];

end
```

4. **Algoritmo de De Casteljau**. Función que dibuja $\mathbf{b}_0^n(t;a,b)$ en $[a,b]$ para un $n \in \mathbb{N}$ y $n+1$ nodos seleccionados por el usuario haciendo click en la pantalla.

```
function []=alg_casteljau(n,a,b);
figure(1)
clf
hold on
axis([-1 1 -1 1])
for k = 1:n+1,
    [c(1,k),c(2,k)] = ginput(1);
    plot(c(1,k),c(2,k),'*k','HandleVisibility','off');
end
t = linspace(a,b,1000);
h = zeros(2,length(t));

for i = 1:length(t)
    h(:,i)= valor_algoritmo_ab(c,t(i),a,b);
end
figure(1)
hold on
plot(c(1,:),c(2,:),'k*'); %Puntos de control
plot(c(1,:).',c(2,:).'); %Poligono de Bezier
plot(h(1,:),h(2,:),'r'); %Curva de Bezier
legend('Puntos de control','Poligono de Bezier','Curva de Bezier');
hold off
end
```

5. **Subdivisión.** Función que devuelve los $n+1$ coeficientes $\mathbf{c}_0,\ldots,\mathbf{c}_n$ correspondientes a la curva $\mathbf{c}(s)$ descrita en la Sección 3.1 a partir de $n+1$ nodos $\mathbf{b}_0,\ldots,\mathbf{b}_n$ seleccionados por el usuario haciendo click en la pantalla, para un $c \in [0,1]$ dado. Además, dibuja ambas curvas en una figura junto a sus nodos y polígonos.

```
function [c]=subdivision(ce,n);
figure(1)
clf
hold on
axis([-1 1 -1 1])
for k = 1:n+1,
    [b(1,k),b(2,k)] = ginput(1);
    plot(b(1,k),b(2,k),'*k','HandleVisibility','off');
end
```

```
c = zeros(2,n+1);
t = linspace(0,1,1000);
h = zeros(2,length(t));

for i=1:n+1,
    c(:,i) = valor_algoritmo_ab(b(:,1:1:i),ce,0,1);
end

for i = 1:length(t)
    h(:,i)= valor_algoritmo_ab(b,t(i),0,1);
    g(:,i)= valor_algoritmo_ab(c,t(i),0,1);
end

plot(b(1,:),b(2,:),'k*'); %Puntos de control
plot(b(1,:).',b(2,:).'); %Poligono de Bezier
plot(h(1,:),h(2,:),'m'); %Curva de Bezier

plot(c(1,:),c(2,:),'ko'); %Puntos de control
plot(c(1,:).',c(2,:).'); %Poligono de Bezier
plot(g(1,:),g(2,:),'g'); %Curva de Bezier
legend('Puntos de control curva inicial', 'Poligono de Bezier ...
   curva inicial','Curva de Bezier inicial','Puntos de control ...
   curva subdividida','Poligono de Bezier curva ...
   subdividida','Curva de Bezier subdividida');
title(strcat('Proceso de subdivision para c =  ', num2str(ce)));
hold off
end
```

6. **Elevación del grado**. Función que, dados los nodos $\mathbf{c}_0, \ldots, \mathbf{c}_n$ de una curva de Bézier de grado n, devuelve los nodos $\mathbf{d}_0, \ldots, \mathbf{d}_{n+1}$ correspondientes a la de grado $n+1$, definidos mediante la ecuación (3.2.1).

```
function [d]=subir_grado(c);
n = size(c,2);
d = zeros(2,n+1);
d(:,1)=c(:,1);
d(:,n+1)=c(:,n);
for i=2:n
    d(:,i)=(i/(n+1))*c(:,i-1)+(1-(i/(n+1)))*c(:,i);
end
```

7. Función empleada para obtener la Figura 3.2.

```
function []=convergencia_subir_grado(n,r);
figure(1)
clf
hold on
axis([-1 1 -1 1])
for k = 1:n+1,
    [c(1,k),c(2,k)] = ginput(1);
    plot(c(1,k),c(2,k),'*k','HandleVisibility','off');
```

```
end

plot(c(1,:),c(2,:),'k.'); %Puntos de control b_0,...,b_n
plot(c(1,:).',c(2,:).');  %Poligono de Bezier
legendInfo{1} = ['Puntos de control'];
legendInfo{2} = ['Poligono de Bezier'];
for i = 1:r,
    d = subir_grado(c);
    plot(d(1,:).',d(2,:).');  %Poligono de Bezier
    c = d;
    legendInfo{i+2} = ['Poligono de Bezier de grado n + ' num2str(i)];
end
legend(legendInfo);
hold off

end
```

8. **Regularidad $\mathcal{C}^1$**. Función que genera, a partir de la curva de Bézier definida en $[u_0, u_1]$ por los nodos $\mathbf{b}_0, \ldots, \mathbf{b}_n$ (escogidos por el usuario haciendo click en la pantalla), una curva definida en $[u_1, u_2]$ mediante los nodos $\mathbf{c}_0, \ldots, \mathbf{c}_n$ de tal manera que exista regularidad $\mathcal{C}^1$ en $\mathbf{b}_n = \mathbf{c}_0$.

```
function [b,c]=continuidad_C1(u0,u1,u2,n);
figure(1)
clf
hold on
axis([-1 1 -1 1]);
for k = 1:n+1,
    [b(1,k),b(2,k)] = ginput(1);
    plot(b(1,k),b(2,k),'*k','HandleVisibility','off');
end
c = zeros(2,n+1);
c(:,1)=b(:,n+1);

t0 = linspace(u0,u1,1000);
t1 = linspace(u1,u2,1000);

c(:,2) = b(:,n+1) + ((u2-u1)/(u1-u0))*(b(:,n+1)-b(:,n));
for k = 3:n+1
    [c(1,k),c(2,k)] = ginput(1);
    plot(c(1,k),c(2,k),'ko','HandleVisibility','off');
end
h0 = zeros(2,length(t0));
h1 = zeros(2,length(t1));

for i = 1:1000,
    h0(:,i)= valor_algoritmo_ab(b,t0(i),u0,u1);
    h1(:,i)= valor_algoritmo_ab(c,t1(i),u1,u2);
end

figure(1)
```

```
hold on
plot(b(1,:),b(2,:),'k*');  %Puntos de control
plot(c(1,:),c(2,:),'ko');  %Puntos de control
plot(b(1,:).',b(2,:).');   %Poligono de Bezier
plot(c(1,:).',c(2,:).');   %Poligono de Bezier

plot(h0(1,:),h0(2,:),'m');   %Curva de Bezier
plot(h1(1,:),h1(2,:),'g');   %Curva de Bezier

title('Spline cubico con regularidad C^{1}')
legend('Puntos control s_{0}','Puntos control s_{1}','Poligono ...
   Bezier s_{0}','Poligono Bezier s_{1}','Curva Bezier ...
   s_{0}','Curva Bezier s_{1}');
hold off

end
```

9. **Comprobación de la regularidad $\mathcal{C}^2$.** Dada una curva con regularidad $\mathcal{C}^1$, prueba si tiene o no regularidad $\mathcal{C}^2$. Muestra en una figura los dos valores de $\mathbf{d}$ definidos en 4.2.5. Se emplea para generar la Figura 4.3.

```
function [d1,d2]=comprobacion_C2(u0,u1,u2,n);
[c1,c2]=continuidad_C1(u0,u1,u2,n);
b = [c1,c2(:,2:n+1)]; %Quitamos el dato que esta repetido.
t1 = (u1-u0)/(u2-u0);

d1 = (b(:,n+2)-t1*b(:,n+3))/(1-t1); %Despejando d de la primera ...
   ecuacion
d2 = (b(:,n)-(1-t1)*b(:,n-1))/t1;   %Despejando d de la segunda ...
   ecuacion
t0 = linspace(u0,u1,1000);
t = linspace(u1,u2,1000);
h0 = zeros(2,length(t0));
h1 = zeros(2,length(t));

for i = 1:1000
    h0(:,i) = valor_algoritmo_ab(b(:,1:n+1),t0(i),u0,u1);
    h1(:,i)= valor_algoritmo_ab(b(:,n+1:size(b,2)),t(i),u1,u2);
end

figure(2)
clf
hold on
plot(c1(1,:),c1(2,:),'k*'); %Puntos de control
plot(c2(1,:),c2(2,:),'ko'); %Puntos de control
plot(c1(1,:).',c1(2,:).','HandleVisibility','off');  %Poligono de ...
   Bezier
plot(c2(1,:).',c2(2,:).','HandleVisibility','off');  %Poligono de ...
   Bezier
plot(d1(1),d1(2),'b*');
plot(d2(1),d2(2),'ro');
```

```
A1 = [d1(1) b(1,n)]; A2 = [d1(2) b(2,n)];
A3 = [d2(1) b(1,n+2)]; A4 = [d2(2) b(2,n+2)];
plot(A1,A2,'k--','HandleVisibility','off');
plot(A3,A4,'k--','HandleVisibility','off');
plot(h0(1,:),h0(2,:),'color',[0, 0.4470, ...
    0.7410],'HandleVisibility','off');  %Curva de Bezier
plot(h1(1,:),h1(2,:),'color',[0.8500, 0.3250, ...
    0.0980],'HandleVisibility','off');  %Curva de Bezier
legend('Puntos de control','Puntos de control','Punto d del lado ...
    izquierdo','Punto d del lado derecho');
hold off

end
```

10. **Regularidad $\mathcal{C}^2$.** Función que, dada una curva con nodos $\mathbf{b}_0, \ldots, \mathbf{b}_n$, (seleccionados desde el monitor por el usuario), elabora una curva con nodos $\mathbf{b}_n = \mathbf{c}_1, \ldots, \mathbf{c}_n$ que presenta regularidad $\mathcal{C}^2$. Se permite al usuario seleccionar los nodos $\mathbf{c}_3, \ldots, \mathbf{c}_n$ haciendo click en la pantalla (pues no hay restricciones para estos).

```
function [d2]=regularidad_C2(u0,u1,u2,n);
figure(1)
clf
hold on
axis([-1 1 -1 1]);
for k = 1:n+1,
    [b(1,k),b(2,k)] = ginput(1);
    plot(b(1,k),b(2,k),'*k','HandleVisibility','off');
end

c = zeros(2,n+1);
c(:,1)=b(:,n+1);

t0 = linspace(u0,u1,1000);
t = linspace(u1,u2,1000);

for i = 1:2,
    for j = 0:i
        c(:,i+1) = c(:,i+1)+ b(:,n-i+j+1)*bernstein_pol(u0,u1,i,j,u2);
    end
end

% Aunque de manera analoga podrian haberse establecido las ...
   condiciones:
% c(:,2) = b(:,n+1) + ((u2-u1)/(u1-u0))*(b(:,n+1)-b(:,n)); Para C1
% t1 = (u1-u0)/(u2-u0);
% d = (1/t1)*(b(:,n)-((1-t1)*b(:,n-1)));
% c(:,3) = (1/t1).*(c(:,2)-(1-t1)*(d));

%Damos el resto de nodos de c de manera arbitraria, pues no hay ...
   condiciones
%de regularidad para ellos.
```

```
for k = 4:n+1
    [c(1,k),c(2,k)] = ginput(1);
    plot(c(1,k),c(2,k),'ko','HandleVisibility','off');
end

for i = 1:1000
    h0(:,i) = valor_algoritmo_ab(b,t0(i),u0,u1);
    h1(:,i)= valor_algoritmo_ab(c,t(i),u1,u2);
end

plot(b(1,:),b(2,:),'k*'); %Puntos de control
plot(c(1,:),c(2,:),'ko'); %Puntos de control
plot(b(1,:).',b(2,:).','HandleVisibility','off');  %Poligono de Bezier
plot(c(1,:).',c(2,:).','HandleVisibility','off');  %Poligono de Bezier
plot(d(1),d(2),'g*');
A1 = [d(1) b(1,n)]; A2 = [d(2) b(2,n)];
A3 = [d(1) c(1,2)]; A4 = [d(2) c(2,2)];
plot(A1,A2,'k--'); plot(A3,A4,'k--');
plot(h0(1,:),h0(2,:),'color',[0, 0.4470, 0.7410]);  %Curva de Bezier
plot(h1(1,:),h1(2,:),'color',[0.8500, 0.3250, 0.0980]);  %Curva de ...
   Bezier
legend('Puntos de control','Puntos de control','Punto d (coinciden)');
hold off

end
```

11. **Spline cúbico de Bézier para L trozos.** Función que crea el spline cúbico de Bézier $\mathbf{s}(u)$ con L trozos, definido en $u_0 < u_1 < \cdots < u_L$ empleando los puntos $\mathbf{b}_0, \ldots, \mathbf{b}_{3L}$ introducidos por el usuario en la matriz $\mathbf{b}$.

```
function []=bez_cubatrozos(b,u);
%b es una matriz de dimensiones 2x(3L+1)
%u es un vector de dimension n+1, que tiene los valores u_0<...<u_L
L1 = (1/3)*(size(b,2)-1);
L2 = length(u)-1;
if (L1 == L2)
    L = L1;
    figure(1)
    hold on
    cont = 1;
    for i = 1:L,
        t(i,:) = linspace(u(i),u(i+1),1000);
        for j = 1:1000,
            q(i,j,:) = bez_cub(b(:,cont),b(:,cont+1),b(:,cont+2),...
            b(:,cont+3),t(i,j),u(i),u(i+1));
        end
        cont = cont + 3;
    end
    plot(b(1,:),b(2,:),'k*'); %Puntos de control
    plot(b(1,:).',b(2,:).');  %Poligono de Bezier
    legendInfo{1} = ['Puntos de control'];
```

```
        legendInfo{2} = ['Poligono de Bezier'];
        for k = 1:L,
            plot(q(k,:,1),q(k,:,2)); %Curva de Bezier
            legendInfo{k+2} = ['Trozo de curva de Bezier ' num2str(k)];
        end
        legend(legendInfo);
        hold off
    else
    error('Las dimensiones no encajan. Introduzca u y b con ...
       size(u)=[1,L+1] y size(b)=[2,3*L+1], L natural.');
    end
    end
```

12. **Spline cúbico de Bézier para L trozos con regularidad $\mathcal{C}^2$.** Función que crea el spline cúbico de Bézier $\mathbf{s}(u)$ con L trozos, definido en $u_0 < u_1 < \cdots < u_L$. Para ello, emplea los puntos $\mathbf{d}_{-1}, \ldots, \mathbf{d}_{L+1}$ introducidos por el usuario en la matriz $\mathbf{d}$.

```
function [] = bspline_cubico(d,u);
%d es una matriz de dimensiones 2xL+3, que tiene los nodos ...
   d_-1,...,d_L
%u es un vector de dimension n+1, que tiene los puntos u_0<...<u_L
L1 = size(d,2)-3;
L2 = length(u)-1;
if (L1 == L2)
    L = size(d,2)-3;
    figure(1)
    clf
    hold on
    b = zeros(2,3*L+1);
    Delta = zeros(L);
    for i = 1:L,
        Delta(i) = u(i+1)-u(i);
    end

    %Igualamos las primeras y ultimas condiciones.
    b(:,1) = d(:,1);
    b(:,2) = d(:,2);
    p = Delta(1)+Delta(2);
    b(:,3) = (Delta(2)/p)*d(:,2)+(Delta(1)/p)*d(:,3);
    r = Delta(L-1)+Delta(L);
    b(:,3*L-1) = (Delta(L)/r)*d(:,L+1)+(Delta(L-1)/r)*d(:,L+2);
    b(:,3*L) = d(:,L+2);
    b(:,3*L+1) = d(:,L+3);

    for i=2:L-1,
        q0 = Delta(i-1)+Delta(i);
        q1 = Delta(i+1)+Delta(i);
        q2 = Delta(i-1)+Delta(i)+Delta(i+1);
        b(:,3*i-1) = (q1/q2)*d(:,i+1) + (Delta(i-1)/q2)*d(:,i+2);
        b(:,3*i) = (Delta(i+1)/q2)*d(:,i+1) + (q0/q2)*d(:,i+2);
    end
```

```
    for i=1:L-1,
        q3 = Delta(i)+Delta(i+1);
        b(:,3*i+1) = (Delta(i+1)/q3)*b(:,3*i) + ...
            (Delta(i)/q3)*b(:,3*i+2);
    end

    cont = 1;
    for i = 1:L,
        t(i,:) = linspace(u(i),u(i+1),1000);
        for j = 1:1000,
            q(i,j,:) = bez_cub(b(:,cont),b(:,cont+1),b(:,cont+2),...
            b(:,cont+3),t(i,j),u(i),u(i+1));
        end
        cont = cont + 3;
    end

    plot(b(1,:),b(2,:),'k*'); %Puntos de control
    plot(b(1,:).',b(2,:).','LineWidth',1.5);  %Poligono de Bezier
    plot(d(1,:),d(2,:),'ro'); %Puntos de control
    plot(d(1,:).',d(2,:).','LineWidth',0.6);  %Poligono de Bezier

    legendInfo{1} = ['b_{0},...,b_{3L}'];
    legendInfo{2} = ['Poligono de Bezier'];
    legendInfo{3} = ['d_{-1}, . . . ,d_{L+1}'];
    legendInfo{4} = ['Poligono del Spline cubico'];

    for k = 1:L,
        plot(q(k,:,1),q(k,:,2));
        legendInfo{k+4} = ['Trozo de spline de Bezier s_' ...
            num2str(k)-1];
    end
    legend(legendInfo);
    hold off

else error('Las dimensiones no encajan. Introduzca u y d con ...
    size(u)=[1,L+1] y size(d)=[2,L+3], L natural.');
end
end
```