Curvas en el Diseño Geométrico

por

Alicia Cantón, Leonardo Fernández-Jambrina y María Jesús Vázquez-Gallo

En memoria de Omar Gil, que supo transmitir vida y matemática...

> «[El móvil] por fuera es sobrio y elegante, sugiere un fino control de los procesos de fabricación. Tal vez brazos de robot con programas llenos de geometría para gobernarlo».
>
> Extraído de un escrito inédito de Omar titulado «Mi móvil matemático» que se puede encontrar completo en su página `https://www.fing.edu.uy/~omargil/`.

Resumen. Las curvas, superficies y sólidos *spline* (NURBS) son el paradigma del diseño industrial hoy en día. Los matemáticos conocemos las funciones *spline* por su importancia en el cálculo numérico, pero también tienen su relevancia en el diseño libre, es decir, en el diseño de formas en la industria mediante el uso de una familia amplia y flexible de curvas que permitan amoldarse a las necesidades de la ingeniería. Es este atractivo enfoque geométrico el que queremos abordar en este artículo, para el cual no se necesitan matemáticas muy complicadas.

1. Introducción

Sin duda todos hemos oído hablar alguna vez del Diseño Asistido por Ordenador, más conocido como CAD, por sus siglas en inglés. Tal vez si hablamos de Diseño *Geométrico* Asistido por Ordenador [4], el acrónimo CAGD nos suene algo menos, pero lo podemos dejar en Diseño Geométrico. La idea es referirnos a las matemáticas, con especial énfasis en la geometría, que están detrás de las aplicaciones de diseño industrial (Rhinoceros, AutoCAD, Blender...) cuando tratamos de representar con ellas curvas, superficies o sólidos. Podemos ver un ejemplo de Rhinoceros en la figura 1.

Así pues, el diseño geométrico sería una disciplina que estudia la representación de las formas (curvas, superficies, sólidos) de objetos de interés industrial (piezas de maquinaria, cascos de barcos, fuselaje de aviones...) de manera matemática sencilla, pero eficiente y precisa, que permita trasladar a las oficinas técnicas las características del objeto para su manufactura.

Pero retrocedamos un poco en el tiempo [6]. La idea de almacenar la geometría del casco de un barco ya aparece en la época romana en la forma de plantillas de curvas para las cuadernas (secciones transversales) del casco, susceptibles de ser

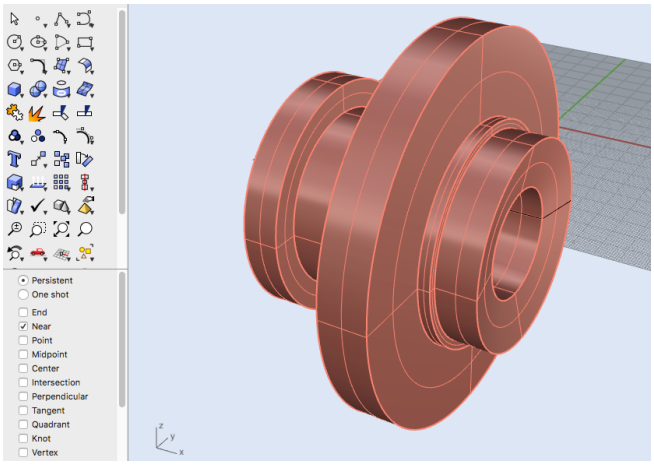

Figura 1: Diseño de una brida en Rhinoceros.

reutilizadas en el proceso de construcción. En la Venecia de los siglos XIII–XVI se mejoró la técnica, definiendo las cuadernas por medio de arcos circulares tangentes entre sí (biarcos). Los planos como tales aparecen en el siglo XVII en Inglaterra y, con ellos, probablemente el uso de junquillos (*splines* en inglés) o piezas de madera flexibles para el trazado de curvas (ver figura 2).

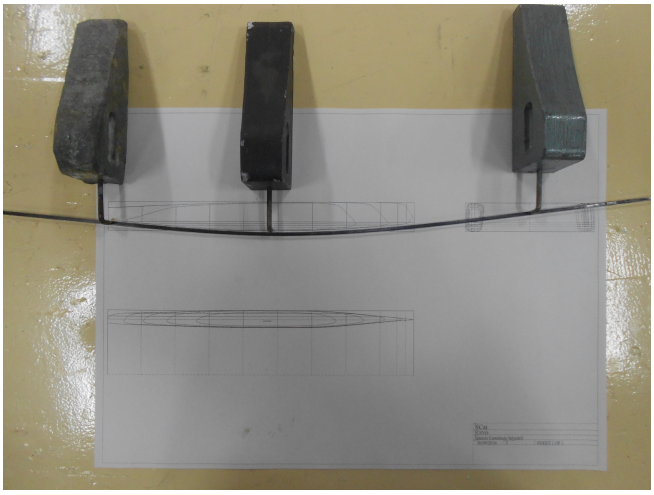

Figura 2: Junquillos.

De la industria naval saltamos a la aeronáutica: en 1944, R. Liming, quien por entonces trabajaba para la Aviación Norteamericana (NAA), planteó que era más eficiente almacenar la información de los planos en forma de números en vez de como

curvas dibujadas a mano. El advenimiento de los primeros ordenadores apoyaba la idea, pero extraer puntos de los planos e interpolar curvas por técnicas tradicionales no daba los resultados esperados.

Por acortar un poco la historia, la solución que hoy conocemos surge en la industria automovilística. En 1959 Citroën contrata al matemático Paul de Casteljau para abordar el problema de digitalizar planos. En vez de trazar curvas que pasaran por un conjunto de puntos, propone usar un *polígono de control*, formado por puntos próximos a la curva. Los puntos de la curva se obtienen a partir del algoritmo que lleva su nombre. Sin embargo, por la política de su empresa, estos trabajos se mantuvieron en secreto durante mucho tiempo.

En paralelo, Pierre Bézier, un ingeniero que trabajaba para la Renault, desarrolló ideas parecidas de manera independiente, pero, a diferencia del trabajo de De Casteljau, tuvo una gran difusión. Es por ello que conocemos estas construcciones como *curvas de Bézier*, aunque también hay precedentes en los trabajos de Pedro Miguel González-Quijano [12].

La idea en sí es sencilla: en lugar de almacenar los puntos de una curva con parametrización polinómica (lo que llamamos curva polinómica) como $c(t) = (1+t-t^2, 3-t+2t^2) = (1,3)+(1,-1)t+(-1,2)t^2$, podemos almacenar los *puntos* del plano $\{(1,3),(1,-1),(-1,2)\}$, correspondientes a las potencias de t en la parametrización. En realidad, no se usa la base monómica sino otra base, pero como ejemplo nos sirve. Estos puntos serían los *vértices* del *polígono de control* de la curva.

Para representar de manera exacta cónicas, podemos extender la construcción a curvas racionales, $c(t) = \left(\frac{1+t-t^2}{2+t+t^2}, \frac{3-t+2t^2}{2+t+t^2}\right)$. En este caso, además del polígono de control mencionado, guardaríamos los coeficientes de las potencias de t en el denominador, $\{2,1,1\}$, que llamaremos, una vez más en otra base, *pesos* de los vértices del polígono de control.

Finalmente, a nadie se le escapa que las curvas polinómicas y racionales no son el modo más eficiente de representar curvas complicadas, por lo que tendremos que acudir a curvas polinómicas o racionales a trozos o *spline*. Si la curva, por ejemplo, está definida en intervalos $[0,1/3]$, $[1/3,3/4]$, $[3/4,1]$, deberemos almacenar además los *nudos* de la partición, no uniforme en este caso, del intervalo $\{0,1/3,3/4,1\}$.

Llegado este punto, ya podemos avanzar que las curvas y superficies NURBS (acrónimo inglés de B-Splines Racionales No Uniformes) constituyen el paradigma del diseño geométrico. Es decir, se pueden plantear otras soluciones al problema de almacenar curvas, superficies y sólidos de interés industrial, pero, debido a que la mayoría de aplicaciones informáticas usan NURBS, es difícil que prosperen a corto plazo. De hecho, desde los años ochenta se ha consolidado el estándar IGES (acrónimo inglés de Especificación de Intercambio Inicial de Gráficos), del National Bureau of Standards, para intercambiar información, también geométrica, entre aplicaciones. Su última versión es de 1994. Hay otros estándares posteriores como STEP (por sus siglas en inglés de Estándar para Intercambio de Productos), pero IGES se mantiene entre los más empleados.

En la actualidad, aunque el origen del diseño geométrico como tal esté en la industria del automóvil, su uso se extiende a todas las ramas de la ingeniería y a

la arquitectura. De hecho, parte de nuestra investigación se dedica a la aplicación a la industria naval, en particular mediante el uso de superficies desarrollables [11]. Relacionado con esta referencia, uno de los proyectos más interesantes que han desarrollado investigadores de nuestro grupo ha sido modelar embarcaciones a partir de formas que la FAO (Organización de las Naciones Unidas para la Alimentación y la Agricultura) tiene en abierto para que países en vías de desarrollo potencien sus recursos pesqueros.

Una buena referencia para empezar es el libro de Gerald Farin [7], en la que nos hemos basado para escribir este artículo. Una referencia en castellano podría ser [10], con animaciones y ejemplos interactivos con GeoGebra.

2. Polinomios de Bernstein y curvas de Bézier

Descartadas las curvas poligonales por razones obvias, las primeras curvas candidatas para su uso en el diseño libre son las curvas con parametrización polinómica. Podemos parametrizar ingenuamente curvas de grado n como

$$c(t) = a_0 + a_1 t + \cdots + a_n t^n, \qquad t \in [0,1],$$

donde cada coeficiente $a_i \in \mathbb{R}^p$, con $p = 2$ o $p = 3$ según la curva sea plana o espacial.

Pese a su sencillez, esta representación en la base monómica $\{1, t, \ldots, t^n\}$ de los polinomios de grado máximo n no es conveniente por múltiples motivos. Para empezar, podemos ver los coeficientes a_i como términos de MacLaurin de la parametrización en torno a $t = 0$,

$$a_i = \frac{c^{(i)}(0)}{i!},$$

y, por tanto, como descriptores del comportamiento local de la curva en torno a su punto inicial. Pero entonces no serían adecuados para transmitir al diseñador el comportamiento global de la curva.

Además, mientras que $a_0 = c(0)$ es un punto, el resto de coeficientes son vectores. Esto se pone de manifiesto si realizamos una transformación afín de la curva. Por ejemplo, si trasladamos la curva por un vector $\mathbf{v}$, la nueva curva tendrá un coeficiente inicial $a_0 + \mathbf{v}$, mientras que el resto de coeficientes no se verán alterados. De nuevo poco evidente para nuestro diseñador, que es nuestro usuario.

Aparte, desde el punto de vista numérico, hay otras bases polinómicas más estables [9], como la base de polinomios de Bernstein en cuanto que el número de condición de las raíces simples de un polinomio es menor en esta última base [8]. Otras buenas propiedades se pueden encontrar en [5, 13, 14].

Por todo ello, merece la pena centrarnos en la base de los polinomios de Bernstein de grado n, $\{B_0^n(t), \ldots, B_n^n(t)\}$,

$$B_i^n(t) := \binom{n}{i} t^i (1-t)^{n-i}, \qquad \binom{n}{i} = \frac{n!}{i!\,(n-i)!},$$

cuyos coeficientes son los números combinatorios que aparecen en la fórmula del binomio de Newton,

$$(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} a^i b^{n-i}.$$

Una propiedad importante se pone de manifiesto tomando $a = t$, $b = 1 - t$ en la expresión del binomio de Newton,

$$1 = (t + 1 - t)^n = \sum_{i=0}^{n} B_i^n(t),$$

es decir, los polinomios de Bernstein de grado n forman una *partición de la unidad*. Veremos que muchas de las buenas propiedades de las curvas de Bézier se derivan de este hecho. Bernstein lo utilizó para dar una demostración constructiva del teorema de Weierstrass [1]. Este teorema establece que toda función continua en un intervalo $[a, b]$ puede ser aproximada uniformemente por polinomios.

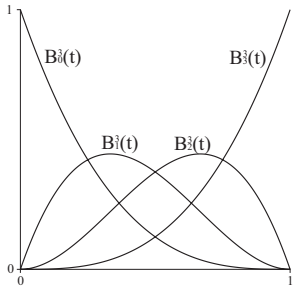

Figura 3: Polinomios de Bernstein de grado tres en $[0, 1]$.

Aquí vamos a representar las parametrizaciones de curvas polinómicas de grado máximo n como combinación de estos polinomios,

$$c(t) = \sum_{i=0}^{n} c_i B_i^n(t), \qquad t \in [0, 1], \tag{1}$$

donde, ahora sí, todos los coeficientes c_i son puntos de $\mathbb{R}^p$, que denominaremos *vértices del polígono de control*, $\{c_0, \ldots, c_n\}$, *de la curva de Bézier* $c(t)$. Estas curvas ya se han tratado en *La Gaceta* (ver [4, 15]) pero, por su interés, merece la pena ampliar la información.

Usando el intervalo $[0, 1]$ para el parámetro t se mantiene la positividad de los polinomios, pero se puede emplear otro intervalo sin más que realizar un cambio afín de parámetro,

$$t(u) = \frac{u - a}{b - a}, \quad u \in [a, b].$$

3. PROPIEDADES DE LAS CURVAS DE BÉZIER

Los vértices inicial y final del polígono de control tienen una interpretación directa. Tenemos que $B_i^n(0) = \delta_{i,0}$ y $B_i^n(1) = \delta_{i,n}$ (ver figura 3 para el caso $n = 3$). Por tanto, todo arco de curva de Bézier arranca en el vértice inicial c_0 y termina en el vértice final c_n,

$$c(0) = c_0, \qquad c(1) = c_n,$$

y en realidad son los únicos vértices del polígono de control por los que seguro pasa la curva de Bézier. De hecho, como vemos en la figura 4, el polígono de control parece una versión poligonal aproximada de la curva.

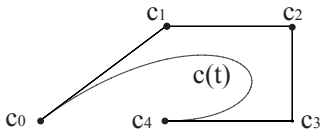

Figura 4: Una curva de Bézier pasa por los vértices inicial y final.

Como su nombre indica, el polígono de control sirve para controlar la forma de la curva. En realidad, el control no es local, ya que alterando un vértice se modifica la curva entera, aunque fundamentalmente la zona de la curva más próxima al vértice desplazado (ver figura 5). Una explicación heurística es que el máximo del polinomio i-ésimo de Bernstein se alcanza en $t = i/n$. Para un control verdaderamente local tendremos que esperar a las curvas *spline*.

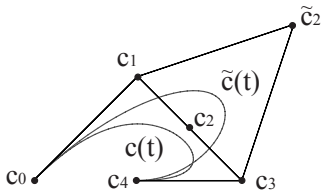

Figura 5: Control local: al mover el vértice c_2, se altera mayormente la parte más próxima de la curva.

El hecho de que la base de polinomios de Bernstein de grado n constituya una partición de la unidad tiene una consecuencia importante: la parametrización de las curvas de Bézier (1) es una combinación baricéntrica de los vértices del polígono de control de la curva, en la que los coeficientes del punto $c(t)$ son precisamente los polinomios de Bernstein evaluados en t. No solo eso; por la positividad de los polinomios de Bernstein en el intervalo $[0, 1]$, dichos coeficientes son positivos.

Una primera propiedad importante que se deduce de este hecho es que el arco de curva de Bézier está contenido en la envolvente convexa del polígono de control

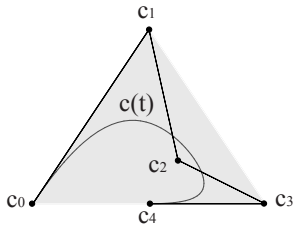

Figura 6: La curva está contenida en la envolvente convexa de su polígono de control.

(ver figura 6). Esto puede ser útil para tener una estimación primera de la posición de la curva.

Las transformaciones afines de $\mathbb{R}^p$ preservan las combinaciones baricéntricas. Si f es una transformación afín,

$$f(c(t)) = f\left(\sum_{i=0}^{n} c_i B_i^n(t)\right) = \sum_{i=0}^{n} f(c_i) B_i^n(t),$$

y, por tanto, el polígono de control de la curva transformada es la imagen del polígono de control de la curva primitiva, $\{f(c_0), \ldots, f(c_n)\}$. Por ello, para representar la imagen solo es preciso transformar el polígono de control de la curva (ver figura 7).

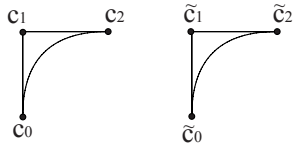

Figura 7: Para trasladar una curva de Bézier, basta trasladar su polígono de control.

Una parametrización de una curva polinómica de grado n se puede expresar formalmente como de grado $n+1$ de manera trivial en la base de potencias de t, ya que basta añadir el término $0t^{n+1}$. En el caso de parametrizaciones de curvas de Bézier es un poco más sofisticado. El interés de esta operación lo veremos en la siguiente sección, cuando mencionemos las curvas polinómicas a trozos, dado que es más sencillo si todos los tramos tienen el mismo grado, lo que motiva elevar formalmente el grado de algunos tramos.

En el caso de parametrizaciones de curvas de Bézier de grado n, podemos elevar el grado formalmente una unidad (ver figura 8) multiplicando la parametrización por $1 = (1 - t + t)$,

$$c(t) = \sum_{i=0}^{n+1} c_i^1 B_i^{n+1}(t),$$

tras una sencilla manipulación, identificando los nuevos vértices, c_i^1, como combinaciones baricéntricas de los vértices antiguos,

$$c_0^1 = c_0, \qquad c_i^1 = \left(1 - \frac{i}{n+1}\right) c_i + \frac{i}{n+1} c_{i-1}, \qquad c_{n+1}^1 = c_n.$$

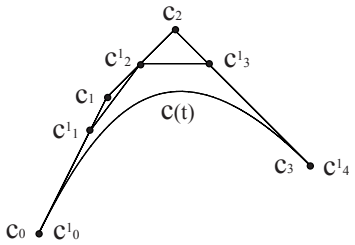

Figura 8: Los polígonos $\{c_0, \ldots, c_3\}$, $\{c_0^1, \ldots, c_4^1\}$ corresponden a la misma curva de Bézier.

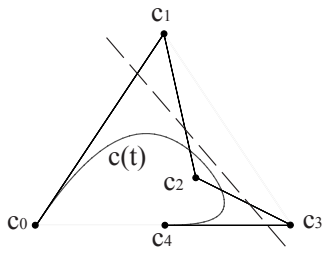

Figura 9: Disminución de la variación: la recta corta al polígono en cuatro puntos y en ninguno a la curva.

Otra propiedad que se demuestra con el procedimiento de elevación del grado es la llamada *disminución de la variación*. El número de puntos de intersección de una recta r con una curva de Bézier $c(t)$ es, *a lo sumo*, el número de puntos de intersección de la recta con el polígono de control de la curva, $\{c_0, \ldots, c_n\}$ (ver figura 9). Esto es así ya que elevar el grado es un proceso de recortado de esquinas del polígono de control, que disminuye las posibles intersecciones de la recta r con el polígono de control. Basta demostrar, pues, que la iteración reiterada de la elevación del grado tiene como límite la propia curva de Bézier [7].

4. Derivadas de curvas de Bézier

Las derivadas de la parametrización de una curva de Bézier se calculan de manera simple por las propiedades de los polinomios de Bernstein,

$$\frac{\mathrm{d}B_i^n(t)}{\mathrm{d}t} = n\left(B_{i-1}^{n-1}(t) - B_i^{n-1}(t)\right),$$

tomando nulos $B_{-1}^{n-1}(t)$, $B_n^{n-1}(t)$, con lo cual la derivada de la parametrización de una curva de Bézier es

$$\frac{\mathrm{d}c(t)}{\mathrm{d}t} = n\sum_{i=0}^{n} c_i \left(B_{i-1}^{n-1}(t) - B_i^{n-1}(t)\right) = n\sum_{i=0}^{n-1} \Delta c_i B_i^{n-1}(t),$$

definiendo el vector diferencia como $\Delta c_i := c_{i+1} - c_i$. Es importante, como veremos, el hecho de que el grado n aparezca como un factor global.

Por tanto, la derivada de una parametrización de una curva de Bézier de grado n es una curva vectorial de Bézier de grado $n-1$ y polígono de control dado por $\{n\Delta c_0, \ldots, n\Delta c_{n-1}\}$. En particular, en los extremos inicial y final del arco,

$$\left.\frac{\mathrm{d}c(t)}{\mathrm{d}t}\right|_{t=0} = n\Delta c_0 = n(c_1 - c_0), \quad \left.\frac{\mathrm{d}c(t)}{\mathrm{d}t}\right|_{t=1} = n\Delta c_{n-1} = n(c_n - c_{n-1}),$$

tenemos una interpretación de los lados inicial y final del polígono de control de la curva, que nos proporcionan la dirección de las tangentes en los extremos de la curva.

Iterando el proceso, podemos generalizar el resultado a derivadas superiores,

$$\frac{\mathrm{d}^k c(t)}{\mathrm{d}t^k} = \frac{n!}{(n-k)!}\sum_{i=0}^{n-k} \Delta^k c_i B_i^{n-k}(t),$$

definiendo las diferencias de orden superior de manera recurrente como $\Delta^k c_i = \Delta^{k-1} c_{i+1} - \Delta^{k-1} c_i$.

Supongamos que queremos unir dos arcos de curvas de Bézier (ver figura 10) de grado n, definidos en intervalos consecutivos $[u_0, u_1]$, $[u_1, u_2]$, y polígonos de control respectivos $\{c_0, \ldots, c_n\}$, $\{\tilde{c}_0, \ldots, \tilde{c}_n\}$. Si la curva compuesta es continua, tendremos que $c(u_1) = \tilde{c}(u_1)$, es decir, $c_n = \tilde{c}_0$.

Si queremos que la curva definida a trozos en el intervalo $[u_0, u_2]$ tenga una parametrización de clase C^1, tendremos que exigir, además,

$$\left.\frac{\mathrm{d}c(u)}{\mathrm{d}u}\right|_{u=u_1} = \left.\frac{\mathrm{d}\tilde{c}(u)}{\mathrm{d}u}\right|_{u=u_1},$$

lo que se traduce en una condición simple,

$$\frac{\Delta c_{n-1}}{\Delta u_0} = \frac{\Delta \tilde{c}_0}{\Delta u_1}.$$

Observamos que el grado n ha desaparecido, por lo cual nos queda una condición geométrica sobre los vértices de los extremos adyacentes de las curvas.

Esta expresión es bastante llamativa, ya que la condición sobre las derivadas de la parametrización de las curvas se traduce en una condición sobre las diferencias de los vértices de sus polígonos en la que no interviene el grado de las curvas, sino los tres vértices más cercanos a la unión y las longitudes de los intervalos donde están definidas.

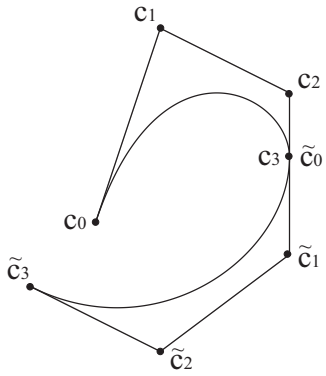

Figura 10: Los segmentos $\overline{\tilde{c}_0\tilde{c}_1}$, $\overline{c_2c_3}$ deben ser paralelos para que la curva compuesta tenga tangente continua.

Para que la curva sea variedad de clase C^1, no necesitamos que la parametrización sea C^1, sino que basta con la continuidad de la tangente en el punto de unión, para lo cual se precisa únicamente que el lado final del polígono de la primera curva y el lado inicial del polígono de control de la segunda curva sean paralelos. A esta condición a veces se la denomina G^1.

La condición para que la parametrización de la curva a trozos sea de clase C^k se deduce fácilmente,

$$\frac{\Delta^s c_{n-s}}{(\Delta u_0)^s} = \frac{\Delta^s \tilde{c}_0}{(\Delta u_1)^s}, \quad s = 0, \ldots, k.$$

Por su parte, la condición para que la parametrización de la curva sea además de clase C^2,

$$\frac{\Delta^2 c_{n-2}}{(\Delta u_0)^2} = \frac{\Delta^2 \tilde{c}_0}{(\Delta u_1)^2},$$

teniendo en cuenta la condición para que sea de clase C^1, tal como se observa en la figura 11, implica que las prolongaciones de los segmentos $\overline{c_{n-2}c_{n-1}}$ y $\overline{\tilde{c}_2\tilde{c}_1}$ se cortan para definir un punto d,

$$c_{n-1} + \frac{\Delta u_1}{\Delta u_0}\Delta c_{n-2} = d = \tilde{c}_1 - \frac{\Delta u_0}{\Delta u_1}\Delta \tilde{c}_1.$$

Otra manera de verlo es considerar los vértices $\{c_{n-2}, c_{n-1}, c_n\}$, $\{\tilde{c}_0, \tilde{c}_1, \tilde{c}_2\}$ como polígonos de control de dos parábolas, dado que las condiciones de derivabilidad no dependen del grado de la curva. En este caso, exigir que la curva compuesta sea de clase C^2 es lo mismo que exigir que sea una única parábola, cuyos dos tramos se obtienen por subdivisión del intervalo $[u_0, u_2]$ en dos subintervalos. La parábola tendrá un polígono de control $\{c_{n-2}, d, \tilde{c}_2\}$.

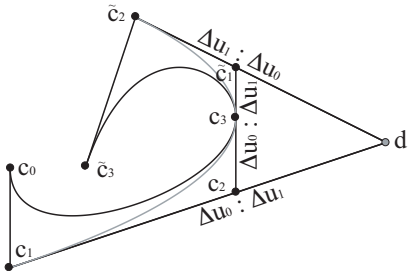

Figura 11: Curva cúbica de clase C^2 de dos tramos, con parábola auxiliar en gris.

5. Algoritmo de De Casteljau

La construcción anterior de las curvas de Bézier en términos de polinomios de Bernstein es útil para deducir propiedades, pero no es la manera en la que se enunciaron, ni la más eficiente.

El algoritmo para trazarlas, análogo al algoritmo de Horner para evaluación de polinomios en la base monómica, consiste en interpolar en el valor t los vértices del polígono de control, de modo que en cada iteración perdemos un vértice. Denotando por $c_i^{1)}(t)$ los vértices interpolados a valor t tras la primera iteración,

$$c_i^{1)}(t) := (1-t)c_i + tc_{i+1}, \quad i = 0, \ldots, n-1,$$

vamos repitiendo el proceso n veces,

$$c_i^{r)}(t) := (1-t)c_i^{r-1)}(t) + tc_{i+1}^{r-1)}(t), \quad i = 0, \ldots, n-r, \quad r = 1, \ldots, n,$$

para quedarnos con un vértice en la última iteración, que es justamente el punto de parámetro t de la curva de Bézier,

$$c(t) = c_0^{n)}(t).$$

Es fácil demostrar por recursividad que ambas construcciones de las curvas de Bézier son equivalentes [7]. En la figura 12 tenemos un ejemplo sencillo con una parábola.

El algoritmo de De Casteljau tiene varias ventajas. Como todos los factores son positivos, involucra solo sumas, lo que le dota de robustez en aritmética de coma flotante. Además, para calcular el punto $c_i^{r)}(t)$ necesitamos solo los puntos $c_i^{r-1)}(t)$, $c_{i+1}^{r-1)}(t)$, lo que nos permite ir sustituyendo en memoria el punto $c_i^{r-1)}(t)$ por el $c_i^{r)}(t)$ a medida que vamos calculando, con lo que no es preciso almacenar los valores previos.

Podemos ir un poco más allá del algoritmo y, en lugar de interpolar en cada iteración con el mismo valor de t, interpolar con un valor t_1 en la primera iteración, con un valor t_2 en la segunda, y así sucesivamente hasta interpolar con un valor t_n

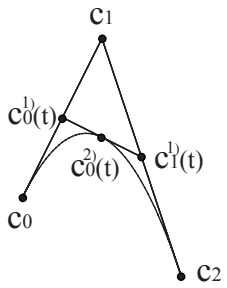

Figura 12: Algoritmo de De Casteljau para una parábola.

en la última iteración, obteniendo así un punto fuera de la curva que denotaremos $c[t_1, \ldots, t_n]$,

$$\begin{aligned} c_i^{1)}[t_1] &:= c_i^{1)}(t_1) = (1-t_1)c_i + t_1 c_{i+1}, \quad i = 0, \ldots, n-1, \\ c_i^{r)}[t_1, \ldots, t_r] &:= (1-t_r)c_i^{r-1)}[t_1, \ldots, t_{r-1}] + t_r c_{i+1}^{r-1)}[t_1, \ldots, t_{r-1}], \\ c[t_1, \ldots, t_n] &:= c_0^{n)}[t_1, \ldots, t_n], \quad i = 0, \ldots, n-r, \quad r = 1, \ldots, n. \end{aligned}$$

Esta construcción es la forma polar de la parametrización $c(t)$: es la única forma multiafín simétrica asociada a la parametrización,

$$c[t, \ldots, t] = c(t).$$

La forma polar (llamada a menudo *blossom* en el argot del diseño geométrico), es multiafín en el siguiente sentido: sea $\lambda + \mu = 1$ una combinación baricéntrica de números, entonces la forma polar

$$\begin{aligned} c[t_1, \ldots, t_{i-1}, \lambda t_i + \mu s_i, t_{i+1}, \ldots, t_n] = \lambda c[t_1, \ldots, t_{i-1}, t_i, t_{i+1}, \ldots, t_n] \\ + \mu c[t_1, \ldots, t_{i-1}, s_i, t_{i+1}, \ldots, t_n], \end{aligned}$$

es afín en cada una de las variables. La simetría es consecuencia del teorema de Menelao [7].

Una propiedad interesante de la forma polar es que permite reconstruir el polígono de control de la curva de Bézier. Es fácil comprobar que, si interpolamos i veces con el valor $t = 1$ y $n - i$ veces con el valor $t = 0$, obtenemos el vértice c_i del polígono de control,

$$c[0^{\langle n-i \rangle}, 1^{\langle i \rangle}] = c_i, \qquad \text{donde } a^{\langle s \rangle} := \underbrace{a, \ldots, a}_{s \text{ veces}}.$$

Esta propiedad es útil para hacer un *zoom* sobre la curva (ver figura 13) y quedarnos solo con la parte correspondiente al intervalo $[a, b] \subset [0, 1]$, cuya parametrización denotaremos por $\tilde{c}(\tilde{t})$,

$$\tilde{c}(\tilde{t}) = c(u(\tilde{t})), \qquad u(\tilde{t}) = a(1-\tilde{t}) + b\tilde{t}, \qquad \tilde{t} \in [0, 1].$$

La forma polar de la curva $\tilde{c}$,

$$\tilde{c}[\tilde{t}_1, \ldots, \tilde{t}_n] = c[a(1-\tilde{t}_1) + b\tilde{t}_1, \ldots, a(1-\tilde{t}_n) + b\tilde{t}_n],$$

nos proporciona los vértices de su polígono de control,

$$\tilde{c}_i = \tilde{c}[0^{\langle n-i\rangle}, 1^{\langle i\rangle}] = c[a^{\langle n-i\rangle}, b^{\langle i\rangle}].$$

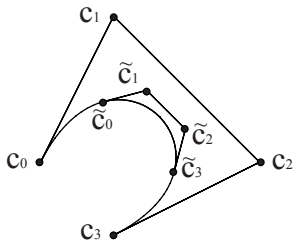

Figura 13: La forma polar nos permite obtener el polígono de una parte de una curva de Bézier.

6. Curvas racionales de Bézier

Las curvas polinómicas de Bézier se extienden fácilmente a curvas racionales. Basta considerar curvas polinómicas en $\mathbb{R}^{p+1}$ que no pasen por el origen y proyectarlas sobre un espacio afín con una parametrización racional. Esto no nos aportará muchos grados nuevos de libertad para el diseño de curvas, pero nos permite describir de manera exacta cónicas.

En $\mathbb{R}^{p+1} = \mathbb{R} \times \mathbb{R}^p$ podemos representar una curva de grado n por medio de un polígono de control, $\{\mathbf{c}_0, \ldots, \mathbf{c}_n\}$, donde $\mathbf{c}_i = (w_i, w_i c_i)$, con $c_i \in \mathbb{R}^p$ y $w_i \in \mathbb{R}$. La parametrización de la curva racional, $c(t)$, se obtiene a partir de la parametrización polinómica, $\mathbf{c}(t)$, por proyección (figura 14),

$$\mathbf{c}(t) = \sum_{i=0}^{n} \mathbf{c}_i B_i^n(t), \qquad c(t) = \frac{\sum_{i=0}^{n} w_i c_i B_i^n(t)}{\sum_{i=0}^{n} w_i B_i^n(t)}, \qquad t \in [0,1].$$

Describimos, pues, las curvas racionales con un polígono de control $\{c_0, \ldots, c_n\}$, como en el caso de curvas polinómicas, y unos parámetros adicionales, $\{w_0, \ldots, w_n\}$, que llamamos *pesos*, que tomaremos positivos, para mantener la propiedad de la envolvente convexa.

Esto es así porque seguimos usando combinaciones baricéntricas de los vértices del polígono de control, ya que, sumando los coeficientes de los vértices c_i,

$$\sum_{i=0}^{n} \frac{w_i B_i^n(t)}{\sum_{j=0}^{n} w_j B_j^n(t)} = 1.$$

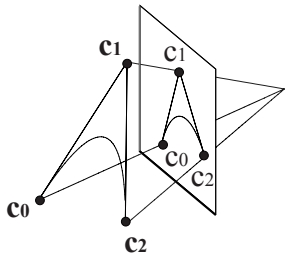

Figura 14: Proyección de una parábola de polígono $\{\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2\}$ en el espacio sobre una cónica de polígono $\{c_0, c_1, c_2\}$ en el plano.

El resto de propiedades que implicaba este hecho para las curvas de Bézier se siguen manteniendo. Algunas incluso mejoran, ya que las curvas racionales son invariantes bajo transformaciones proyectivas, no solo afines. Estas transformaciones son muy importantes para el diseño, ya que, además de las afines, incluyen los cambios de perspectiva.

Como la parametrización no cambia si multiplicamos todos los pesos por un mismo número, podemos usar este hecho para normalizar los pesos y escoger, por ejemplo, que el primer peso valga la unidad.

Además, si realizamos una transformación de Möbius del intervalo $[0, 1]$ sobre sí mismo,

$$t(u) = \frac{u}{(1-b)u + b}, \quad u \in [0, 1],$$

la parametrización sigue siendo racional y de grado n. El polígono de control se mantiene, pero la lista de pesos cambia a $\{\tilde{w}_0, \ldots, \tilde{w}_n\}$, donde $\tilde{w}_i = b^{n-i} w_i$. Podemos usar esta propiedad para conseguir que los pesos inicial y final sean iguales a la unidad, lo que nos deja tan solo con $n - 1$ pesos libres.

En el caso de curvas racionales de grado dos, recuperamos las parametrizaciones racionales de arcos de cónicas. Normalizando los pesos,

$$c(t) = \frac{(1-t)^2 c_0 + 2wt(1-t)c_1 + t^2 c_2}{(1-t)^2 + 2wt(1-t) + t^2},$$

es fácil ver que la parametrización no tiene polos para $0 < w < 1$, por lo que tendríamos un arco de elipse. Para $w = 1$ la parametrización es polinómica y tendremos un arco de parábola. Para $w > 1$, los polos son dos y tendremos un arco de hipérbola (ver figura 15), proporcionando una interpretación inmediata al peso de la cónica [2], que es útil asimismo para interpretar las cuádricas [3].

7. Curvas spline

Aunque las curvas polinómicas son un instrumento útil para el diseño, pecan de cierta rigidez, lo que puede provocar tramos de elevada curvatura u oscilaciones

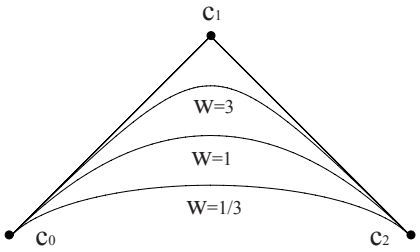

Figura 15: Elipse, $w = 1/3$, parábola, $w = 1$, hipérbola, $w = 3$.

bruscas si nos vamos a grados altos para representar formas complicadas. Aparte, nos gustaría mejorar el control local de la forma de las curvas.

Hemos visto que extender las curvas de Bézier a curvas racionales aporta pocos grados de libertad. Por ello, pasamos directamente a considerar las curvas polinómicas a trozos o *spline*.

A modo de ejemplo, estudiaremos cómo construir parábolas de clase C^1 a trozos, ya que es un caso sencillo, que nos permite entrever la generalidad de las curvas *spline*.

Consideremos una curva formada por N tramos parabólicos, que por tanto será plana. Tendremos una lista de vértices $\{c_0, \ldots, c_{2N}\}$, donde $\{c_0, c_1, c_2\}$ es el polígono de control del primer tramo, $\{c_2, c_3, c_4\}$ es el polígono del tramo segundo, $\{c_{2(i-1)}, c_{2i-1}, c_{2i}\}$, el del tramo i-ésimo, y $\{c_{2N-2}, c_{2N-1}, c_{2N}\}$, el del último tramo. Estamos ya implementando el hecho de que la curva es continua (ver figura 16).

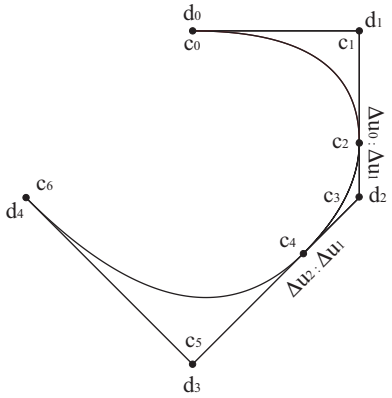

Figura 16: Parábola a trozos.

Cada polígono de control corresponderá a una parametrización en un intervalo. El primer tramo estará definido en un intervalo $[u_0, u_1]$, el segundo tramo, en $[u_1, u_2]$, el tramo i-ésimo, en $[u_{i-1}, u_i]$ y el último tramo, en $[u_{N-1}, u_N]$.

Para que la parametrización de la curva compuesta sea de clase C^1 en todo el intervalo $[u_0, u_N]$, exigimos que su derivada sea continua para cada valor $u = u_i$ en la unión de dos tramos,

$$\frac{\Delta c_{2i-1}}{\Delta u_{i-1}} = \frac{\Delta c_{2i}}{\Delta u_i}, \qquad i = 1, \ldots, N-1, \tag{2}$$

es decir, que los vectores $\mathbf{c}_{2i-1}\mathbf{c}_{2i}$ y $\mathbf{c}_{2i}\mathbf{c}_{2i+1}$ sean paralelos y en proporción $\Delta u_{i-1} : \Delta u_i$.

Es obvio que no podemos permitir a nuestro diseñador modificar *todos* los vértices, ya que perderíamos la derivabilidad. Solo deberemos permitirle modificar los vértices que no alteren dichas condiciones.

Podemos ver las condiciones (2) como definición de cada vértice extremo c_{2i} como combinación baricéntrica de los vértices interiores c_{2i-1}, c_{2i+1},

$$c_{2i} = \frac{\Delta u_i}{\Delta u_{i-1} + \Delta u_i} c_{2i-1} + \frac{\Delta u_{i-1}}{\Delta u_{i-1} + \Delta u_i} c_{2i+1}, \qquad i = 1, \ldots, N-1.$$

De esta forma, conocidos los vértices interiores, c_{2i-1}, de cada polígono y los *nudos* de la partición del intervalo, $\{u_0, \ldots, u_N\}$, podemos reconstruir los vértices de las uniones de tramos, c_{2i}, de los polígonos por la relación anterior, garantizando que la parametrización sea de clase C^1.

A los vértices impares habrá que añadir los vértices que no se obtienen de una condición de derivabilidad, c_0, c_{2N}, ya que no corresponden a ninguna unión de tramos.

Así pues, los únicos datos de la curva que debemos facilitar al diseñador para que los modifique libremente son los nudos de la partición del intervalo $\{u_0, \ldots, u_N\}$, y los vértices de los polígonos de control $\{c_0, c_1, \ldots, c_{2i-1}, \ldots, c_{2N-1}, c_{2N}\}$, un total de $N + 2$ vértices.

A este conjunto de vértices lo denominaremos *polígono B-spline* de la curva de clase C^1 polinómica a trozos. Denotaremos como $\{d_0, \ldots, d_{N+1}\}$ estos puntos,

$$d_0 := c_0, \qquad d_i := c_{2i-1},\ i = 1, \ldots, N, \qquad d_{N+1} := c_{2N}.$$

Usando la forma polar de la parametrización en cada tramo, tenemos que

$$d_0 = c[u_0, u_0], \quad d_i = c[u_{i-1}, u_i],\ i = 1, \ldots, N, \quad d_{N+1} = c[u_N, u_N].$$

Observamos que los vértices del polígono B-*spline* se obtienen evaluando la forma polar sobre parejas de nudos correlativos, salvo el primero y el último. Esto se puede arreglar definiendo *nudos auxiliares* $u_{-1} := u_0$, $u_{N+1} := u_N$, de modo que

$$d_i = c[u_{i-1}, u_i], \quad i = 0, \ldots, N+1,$$

y la lista de nudos queda así:

$$\{u_{-1}, u_0, u_1, u_2, \ldots, u_{N-2}, u_{N-1}, u_N, u_{N+1}\}.$$

Más interesante, dado que es el caso típico de las curvas polinómicas a trozos, es la construcción de una curva a trozos de clase C^2 de N tramos cúbicos, con vértices de los sucesivos polígonos de control $\{c_0, \ldots, c_{3N}\}$, de los cuales $\{c_{3(i-1)}, c_{3i-2}, c_{3i-1}, c_{3i}\}$ corresponden al tramo i-ésimo, definido en el intervalo $[u_{i-1}, u_i]$ (ver figura 17).

Ya sabemos que imponer que la curva sea de clase C^1 implica que

$$c_{3i} = \frac{\Delta u_i}{\Delta u_{i-1} + \Delta u_i} c_{3i-1} + \frac{\Delta u_{i-1}}{\Delta u_{i-1} + \Delta u_i} c_{3i+1}, \qquad i = 1, \ldots, N-1.$$

La condición de continuidad de las derivadas segundas, desarrollada en la sección 4, aplicada al nudo u_i nos proporciona como información que los vectores $\mathbf{c}_{3i-2}\mathbf{c}_{3i-1}$ y $\mathbf{c}_{3i-1}\mathbf{d}_i$ están en proporción $\Delta u_{i-1} : \Delta u_i$, igual que los vectores $\mathbf{d}_i\mathbf{c}_{3i+1}$ y $\mathbf{c}_{3i+1}\mathbf{c}_{3i+2}$.

Aplicando esta misma condición al nudo u_{i+1} obtenemos otra relación. Los vectores $\mathbf{c}_{3i+1}\mathbf{c}_{3i+2}$ y $\mathbf{c}_{3i+2}\mathbf{d}_{i+1}$ están en proporción $\Delta u_i : \Delta u_{i+1}$. Y del nudo u_{i-1} extraemos que $\mathbf{d}_{i-1}\mathbf{c}_{3i-2}$ y $\mathbf{c}_{3i-2}\mathbf{c}_{3i-1}$ guardan una proporción igual a $\Delta u_{i-2} : \Delta u_{i-1}$.

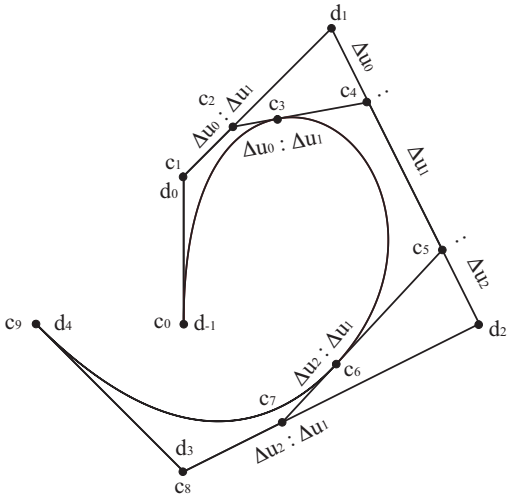

Figura 17: Cúbica a trozos.

Con toda esta información, estamos en condiciones de expresar los vértices c_{3i-1}, c_{3i+1} en términos de los nuevos puntos que hemos introducido,

$$c_{3i-1} = \frac{\Delta u_i}{\Delta u_{i-2} + \Delta u_{i-1} + \Delta u_i} d_{i-1} + \frac{\Delta u_{i-2} + \Delta u_{i-1}}{\Delta u_{i-2} + \Delta u_{i-1} + \Delta u_i} d_i,$$

$$c_{3i+1} = \frac{\Delta u_i + \Delta u_{i+1}}{\Delta u_{i-1} + \Delta u_i + \Delta u_{i+1}} d_i + \frac{\Delta u_{i-1}}{\Delta u_{i-1} + \Delta u_i + \Delta u_{i+1}} d_{i+1}.$$

Este resultado permite obtener todos los vértices de la curva compuesta a partir de los puntos $d_1, \ldots, d_{N-1}$ asociados a cada unión, imponiendo que la parametrización sea de clase C^2, salvo los vértices iniciales, c_0, c_1, y finales, c_{3N-1}, c_{3N}, que no corresponden a ninguna unión.

Denotamos, como en el caso de las parábolas, $d_{-1} := c_0$, $d_0 := c_1$, $d_N := c_{3N-1}$, $d_{N+1} := c_{3N}$, completando la lista de vértices.

Expresamos los nuevos vértices por medio de la forma polar. El nudo u_i está asociado a los vértices de los tramos cúbicos,

$$c_{3i-2} = c[u_{i-1}, u_{i-1}, \mathbf{u}_i], \quad c_{3i-1} = c[u_{i-1}, u_i, \mathbf{u}_i], \quad c_{3i} = c[u_i, u_i, \mathbf{u}_i],$$
$$c_{3i+1} = c[\mathbf{u}_i, u_i, u_{i+1}], \quad c_{3i+2} = c[\mathbf{u}_i, u_{i+1}, u_{i+1}],$$

y sobre la parábola auxiliar, con forma polar $b[t_1, t_2]$, que usamos para la continuidad de las derivadas segundas,

$$c_{3i-2} = b[u_{i-1}, u_{i-1}], \quad c_{3i-1} = b[u_{i-1}, u_i], \quad c_{3i} = b[u_i, u_i],$$
$$c_{3i+1} = b[u_i, u_{i+1}], \quad c_{3i+2} = b[u_{i+1}, u_{i+1}],$$

observando que la diferencia entre la cúbica y la parábola es simplemente evaluar la forma polar u_i.

Por tanto, $d_i = b[u_{i-1}, u_{i+1}] = c[u_{i-1}, \mathbf{u}_i, u_{i+1}]$, $i = 1, \ldots, N-1$, y

$$d_{-1} := c_0 = c[u_0, u_0, u_0], \quad d_0 := c_1 = c[u_0, u_0, u_1],$$
$$d_N := c_{3N-1} = c[u_{N-1}, u_N, u_N], \quad d_{N+1} := c_{3N} = c[u_N, u_N, u_N].$$

Si introducimos cuatro *nudos auxiliares*

$$u_{-2} := u_0, \quad u_{-1} := u_0, \quad u_{N+1} := u_N, \quad u_{N+2} := u_N,$$

todos los vértices se evalúan sobre ternas de nudos consecutivos,

$$d_i = c[u_{i-1}, u_i, u_{i+1}], \qquad i = -1, \ldots, N+1,$$

extraídas de $\{u_{-2}, u_{-1}, u_0, u_1, u_2, \ldots, u_{N-2}, u_{N-1}, u_N, u_{N+1}, u_{N+2}\}$.

Obviamente, la notación se corrige para comenzar todas las listas en el cero, y así codificar la curva cúbica compuesta por medio del *polígono de control B-spline* $\{d_0, \ldots, d_{N+2}\}$ y la sucesión de nudos $\{u_0, \ldots, u_N\}$, de manera que el diseñador pueda modificar la forma de la curva sin alterar sus propiedades de derivabilidad.

Estos ejemplos nos sugieren algunas propiedades de las curvas de grado n de clase C^{n-1} a trozos con N tramos:

- Se definen con un polígono B-*spline* de $n+N$ vértices $\{d_0, \ldots, d_{n+N-1}\}$ y una lista de $2n+N-1$ nudos $\{u_0, \ldots, u_{2n+N-2}\}$.
- La curva está definida en el intervalo de N tramos $[u_{n-1}, u_{n+N-1}]$.
- Los $n-1$ primeros nudos y los últimos son nudos auxiliares y se toman respectivamente iguales a u_{n-1} y a u_{n+N-1}. Es decir, la lista de nudos comienza con n nudos iguales $u_0 = \cdots = u_{n-1}$ y acaba con otros n nudos iguales $u_{n+N-1} = \cdots = u_{2n+N-2}$.
- Los vértices se definen a partir de la forma polar evaluada sobre listas de nudos correlativos, $d_i = c[u_i, \ldots, u_{i+n-1}]$.

La primera propiedad se comprueba rápidamente, ya que una curva continua de grado n y N tramos tiene polígonos de control por un total de $1+nN$ vértices, pero imponemos que la curva sea de clase C^{n-1}, lo que supone $n-1$ condiciones en las $N-1$ uniones de los tramos.

El resto de propiedades son consecuencia de definir los vértices como $d_i = c[u_i,\dots,u_{i+n-1}]$, a la vez que se mantienen los vértices primero y último,

$$d_0 = c_0 = c[u_{n-1}^{\langle n\rangle}], \qquad d_{n+N-1} = c_{nN} = c[u_{n+N-1}^{\langle n\rangle}],$$

lo que justifica la introducción de los nudos auxiliares.

Queda por ver cómo se expresa el algoritmo de De Casteljau con el polígono B-*spline* y la lista de nudos, sin pasar por los polígonos de control de cada tramo por medio de la forma polar. Es lo que se llama *algoritmo de De Boor*.

Esto es un poco tedioso, pero es simple aplicación reiterada de la propiedad de multiafinidad de la forma polar de una parametrización [7].

Para una curva de un único tramo, en la primera iteración, de cada par de vértices correlativos, $d_i = c[u_i,\dots,u_{i+n-1}]$, $d_{i+1} = c[u_{i+1},\dots,u_{i+n}]$, con $n-1$ nudos $\{u_{i+1},\dots u_{i+n-1}\}$ comunes desaparecen los nudos no comunes, u_i, u_{i+n}, y aparece u por interpolación,

$$u = \frac{u_{i+n}-u}{u_{i+n}-u_i}u_i + \frac{u-u_i}{u_{i+n}-u_i}u_{i+n},$$

para dar el nuevo vértice $d_i^{1)}(u)$, $i=0,\dots,n-1$, mediante

$$d_i^{1)}(u) = c[u_{i+1},\dots u_{i+n-1},u] = \frac{u_{i+n}-u}{u_{i+n}-u_i}d_i + \frac{u-u_i}{u_{i+n}-u_i}d_{i+1}.$$

El esquema se repite para el resto de iteraciones, $r=1,\dots,n$: del par de vértices

$$d_i^{r)} = c[u_{i+r},\dots,u_{i+n-1},u^{\langle r\rangle}], \qquad d_{i+1}^{r)} = c[u_{i+r+1},\dots,u_{i+n},u^{\langle r\rangle}],$$

desaparecen los nudos no comunes, u_{i+r}, u_{i+n}, y aparece u por interpolación,

$$u = \frac{u_{i+n}-u}{u_{i+n}-u_{i+r}}u_{i+r} + \frac{u-u_{i+r}}{u_{i+n}-u_{i+r}}u_{i+n},$$

para dar el nuevo vértice $d_i^{r+1)}(u)$, $i=0,\dots,n-r-1$,

$$\begin{aligned} d_i^{r+1)}(u) &= c[u_{i+r+1},\dots,u_{i+n-1},u^{\langle r+1\rangle}] \\ &= \frac{u_{i+n}-u}{u_{i+n}-u_{i+r}}d_i^{r)}(u) + \frac{u-u_{i+r}}{u_{i+n}-u_{i+r}}d_{i+1}^{r)}(u). \end{aligned}$$

Recapitulando, el *algoritmo de De Boor* no es más que la aplicación reiterada del algoritmo de De Casteljau (ver figura 18).

Para aplicar el algoritmo de De Boor a curvas compuestas de N tramos polinómicos de grado n, solo hay que identificar el subintervalo $[u_I, u_{I+1})$ al que pertenece el valor u del parámetro en el que estamos evaluando y los vértices $\{\tilde{d}_0 := d_{I-n+1},\dots,\tilde{d}_n := d_{I+1}\}$ y nudos $\tilde{u}_0 := u_{I-n+1},\dots,\tilde{u}_{2n-1} := u_{I+n}$ correspondientes a dicho tramo.

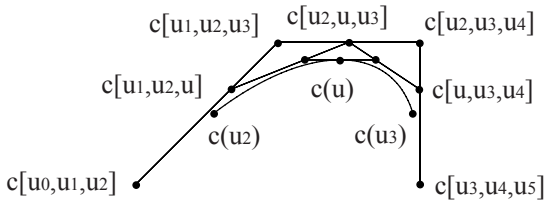

Figura 18: Algoritmo de De Boor para una cúbica.

8. PROPIEDADES DE LAS CURVAS SPLINE

El algoritmo de De Boor muestra que las parametrizaciones de las curvas *spline* siguen siendo combinaciones baricéntricas de los vértices del polígono B-*spline*, por lo que se mantienen las mismas propiedades de las curvas de Bézier, tales como la invariancia afín, la disminución de la variación o la propiedad de la envolvente convexa del polígono B-*spline*.

No solo eso, sino que además, como en cada tramo las curvas *spline* son curvas polinómicas, las propiedades que ya vimos para estas se verifican, no solo globalmente, sino en cada tramo concreto.

Por ejemplo, cada tramo de curva, digamos el i-ésimo, está contenido en la envolvente convexa de su polígono de control, $\{d_{i-1}, \ldots, d_{i+n-1}\}$. Lo mismo acontece con la propiedad de la disminución de la variación. Una recta corta al tramo i-ésimo de la curva como máximo en tantos puntos como al polígono de control de dicho tramo (figura 19).

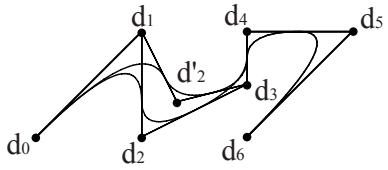

Figura 19: Control local: un vértice de un *spline* parabólico controla tres tramos de curva.

Pero hay otras propiedades que experimentan una mejora. Cada tramo de la curva *spline* depende tan solo de los $n+1$ vértices de su polígono de control. Así pues, no se ve alterado por cambios del resto de vértices. De otro modo, un vértice genérico d_i pertenece a los polígonos de control de todos los tramos desde el $(i-n+1)$-ésimo, $\{d_{i-n}, \ldots, d_i\}$, hasta el $(i+1)$-ésimo, $\{d_i, \ldots, d_{i+n}\}$. Un desplazamiento del vértice d_i modifica tan solo a un total de $n+1$ tramos de la curva, salvo que el vértice esté próximo al comienzo o al final del polígono, caso en el que controlará un número inferior de tramos. A esta nueva propiedad se la denomina *control local*.

Obviamente, si aun así la modificación de la curva es importante, se puede mitigar subdividiendo los tramos, de modo que el control afecte a una porción más peque-

ña de la curva. Esto se logra refinando la partición del intervalo adecuadamente y recalculando el polígono B-*spline* haciendo uso de la forma polar.

Vemos que el formalismo de Bézier y el *spline* son similares si empleamos la forma polar de la parametrización. El formalismo de Bézier usa vértices que se obtienen aplicando la forma polar sobre listas de nudos repetidos, mientras que el *spline* usa listas de nudos correlativos. Podemos pasar de uno a otro mediante la forma polar.

Las curvas que más se emplean en el diseño industrial son precisamente las curvas *spline* cúbicas, por todas las buenas propiedades que hemos expuesto y porque son las curvas alabeadas de grado más bajo. Tras esta revisión de las curvas en el diseño geométrico, quedaría ver el diseño de superficies, que excede ya los límites de este artículo, aunque en las referencias [7, 10] se puede ampliar en esta dirección.

Referencias

[1] S. N. Bernstein, Sur l'ordre de la meilleure approximation des fonctions continues par les polynômes de degré donné, *Mem. Acad. Roy. Belg.* **4** (1912), 1–104.

[2] A. Cantón, L. Fernández-Jambrina y M. E. Rosado, Geometric characteristics of conics in Bézier form, *Comput.-Aided Des.* **43** (2011), 1413–1421.

[3] A. Cantón, L. Fernández-Jambrina, M. E. Rosado y M. J. Vázquez-Gallo, Geometric elements and classification of quadrics in rational Bézier form, *J. Comput. Appl. Math.* **300** (2016), 400–419.

[4] J. Caravantes y L. González-Vega, Diseño Geométrico Asistido por Ordenador: Álgebra, Geometría y Computación, *Gac. R. Soc. Mat. Esp.* **11** (2008), 109–150.

[5] J. M. Carnicer y J. M. Peña, Shape preserving representations and optimality of the Bernstein basis, *Adv. Comput. Math.* **1** (1993), 173–196.

[6] G. Farin, A history of curves and surfaces in CAGD, *Handbook of Computer Aided Geometric Design*, 1–21, North-Holland, 2002.

[7] G. Farin, *Curves and surfaces for CAGD: A practical guide*, 5.ª ed., Morgan Kaufmann Publishers, 2002.

[8] R. Farouki, The Bernstein polynomial basis: A centennial retrospective, *Comput. Aided Geom. Design* **29** (2012), 379–419.

[9] R. Farouki y V. Rajan, On the numerical condition of polynomials in Bernstein form, *Comput. Aided Geom. Design* **4** (1987), 191–216.

[10] L. Fernández-Jambrina, *Curvas y Superficies en el Diseño Geométrico Asistido por Ordenador*, 2020. `https://dcain.etsin.upm.es/~leonardo/`

[11] L. Fernández-Jambrina y F. Pérez-Arribas, Developable surfaces bounded by NURBS curves, *J. Comput. Math.* **38** (2020), 693–709.

[12] M. A. Gil, Ingeniería y Matemática en España en la primera mitad del siglo XX: Pedro Miguel González-Quijano, *Gac. R. Soc. Mat. Esp.* **12** (2009), 751–772.

[13] J. M. PEÑA, B-splines and optimal stability, *Math. Comp.* **66** (1997), 1555–1560.

[14] J. M. PEÑA, On the optimal stability of bases of univariate functions, *Numer. Math.* **91** (2002), 305–318.

[15] C. VINUESA, Curvas peligrosas, *Gac. R. Soc. Mat. Esp.* **19** (2016), 151–167.

ALICIA CANTÓN, DPTO. DE MATEMÁTICA E INFORMÁTICA APLICADAS A LAS INGENIERÍAS CIVIL Y NAVAL, UNIVERSIDAD POLITÉCNICA DE MADRID
Correo electrónico: alicia.canton@upm.es

LEONARDO FERNÁNDEZ, DPTO. DE MATEMÁTICA E INFORMÁTICA APLICADAS A LAS INGENIERÍAS CIVIL Y NAVAL, UNIVERSIDAD POLITÉCNICA DE MADRID
Correo electrónico: leonardo.fernandez@upm.es

MARÍA JESÚS VÁZQUEZ-GALLO, DPTO. DE MATEMÁTICA E INFORMÁTICA APLICADAS A LAS INGENIERÍAS CIVIL Y NAVAL, UNIVERSIDAD POLITÉCNICA DE MADRID
Correo electrónico: mariajesus.vazquez@upm.es