Curvas de Bézier y sus aplicaciones

Índice

Trabajo realizado por:

-Ignacio Aguilera Martos.
-Luis Balderas Ruiz.
-Ignacio Barragán Lozano.
-Alicia Rodríguez Gómez

1. Polinomios de Bernstein

Los polinomios de Bernstein, cuyo nombre hace referencia al matemático ucraniano Sergei Natanovich Bernstein, son llamados así porque son la base primordial de la demostración que Bernstein hizo sobre el "Teorema de Aproximación de Weirstrass". La esencia de dicha prueba es la construcción de una sucesión de polinomios que convergen uniformemente a una función continua. Más adelante se expondrá detalladamente dicha demostración.

Señalar que los polinomios de Bernstein tienen cierta conexión con la teoría de Probabilidad, con problemas sobre momentos, con la teoría de sumas de series divergentes. Además existen problemas asociados al análisis complejo, algunos aún sin ser resueltos, que tienen como base el comportamiento de los nombrados polinomios.

Estos polinomios son útiles herramientas matemáticas ya que se definen de forma sencilla. Además son rápidamente calculables en sistemas computacionales y permiten representar una gran variedad de funciones. Se pueden integrar y derivar con suma facilidad, y se utilizan para formar curvas que se aproximan con bastante exactitud a una función deseada.

a) Definición

Definimos el i-ésimo polinomio de Bernstein de grado n tal que:

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

Además, sea una función f(x) definida en el intervalo [0,1], la siguiente expresión es el Polinomio de Bernstein de orden n de f(x):

$B_n(x) = B_f^n(x) = \sum_{k=0}^{n} f\left(\frac{k}{n}\right) \binom{n}{k} x^k (1-x)^{n-k}$

Para los casos más sencillos, conocemos el polinomio de Bernstein. Para n=0, solo existe uno:

$B_0^0(t) = 1$

Para el caso de n=1, encontramos solo dos polinomios:

$B_0^1(t) = 1-t$, $B_1^1(t) = t$

También para n=2, sabemos que son tres polinomios:

$B_0^2(t) = 1-2t+t^2$ $B_1^2(t) = 2(t-t^2)$ $B_2^2(t) = t^2$

Y por último para el caso n=3, son tal que:

$B_0^3(t)$ =1-3t+3t²– t³ B_1^3(t)=3t-6t²+3t³ B_2^3(t)=3t²-3t³ $B_3^3(t)$ =t³

b) Propiedades

Existen ciertas propiedades que caracterizan a los polinomios de Bernstein. A continuación numeraremos algunas de ellas:

1- **Base de polinomios:** El conjunto de polinomios de Bernstein del mismo grado n forman una base del espacio vectorial de polinomios de grado menor o igual que n.

$$\{B_0^n(t), B_1^n(t), \dots, B_n^n(t)\}$$

2- **Partición de la unidad:** Para todo n mayor o igual a cero y para todo t perteneciente al intervalo cerrado [0,1] se verifica:

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

Demostración:

Supongamos que tenemos los polinomios de Bernstein de grado 2:

$$B_0^2(t) = t^2 - 2t + 1$$
$$B_1^2(t) = -2t^2 + 2t$$
$$B_2^2(t) = t^2$$

Ahora los sumamos todos:

$$B_0^2(t) + B_1^2(t) + B_2^2(t) = (t^2 - 2t + 1) + (-2t^2 + 2t) + (t^2) = 1$$

3- **No negatividad:** Todo polinomio de Bernstein es no negativo dentro del intervalo [0,1], para todo t perteneciente a dicho intervalo.

$$B_i^n(t) \in [0,1]$$

Demostración:

Supongamos que tenemos los polinomios de Bernstein de grado 2:

$$B_0^2(t) = t^2 - 2t + 1$$
$$B_1^2(t) = -2t^2 + 2t$$
$$B_2^2(t) = t^2$$

Observamos que en el intervalo [0,1] los polinomios son siempre positivos.

4- **Simetría:** La siguiente igualdad se verifica para todo i desde cero hasta n:

$$B_i^n(t) = B_{n-i}^n(1-t)$$

Demostración: La forma más rápida de demostrar dicha característica es la visual. Para ello existen numerables programas que a los cuales les

introducimos el polinomio de Bernstein de cualquier grado y nos muestras las curvas. En nuestro caso, la herramienta de trabajo es Octave. En la parte final del trabajo se representan, varias curvas y es ahí donde veremos claramente la propiedad de simetría.

5- <u>Condiciones en los extremos del intervalo:</u>

$$B_i^n(0) = \delta_i^0 \qquad\qquad B_i^n(1) = \delta_i^n$$

Para todo i ∈ {0,1,…, n} donde δ es la función delta de Kronecker.

6- **<u>Recursión:</u>** Los polinomios de Bernstein cumplen la siguiente fórmula recursiva:

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t),$$

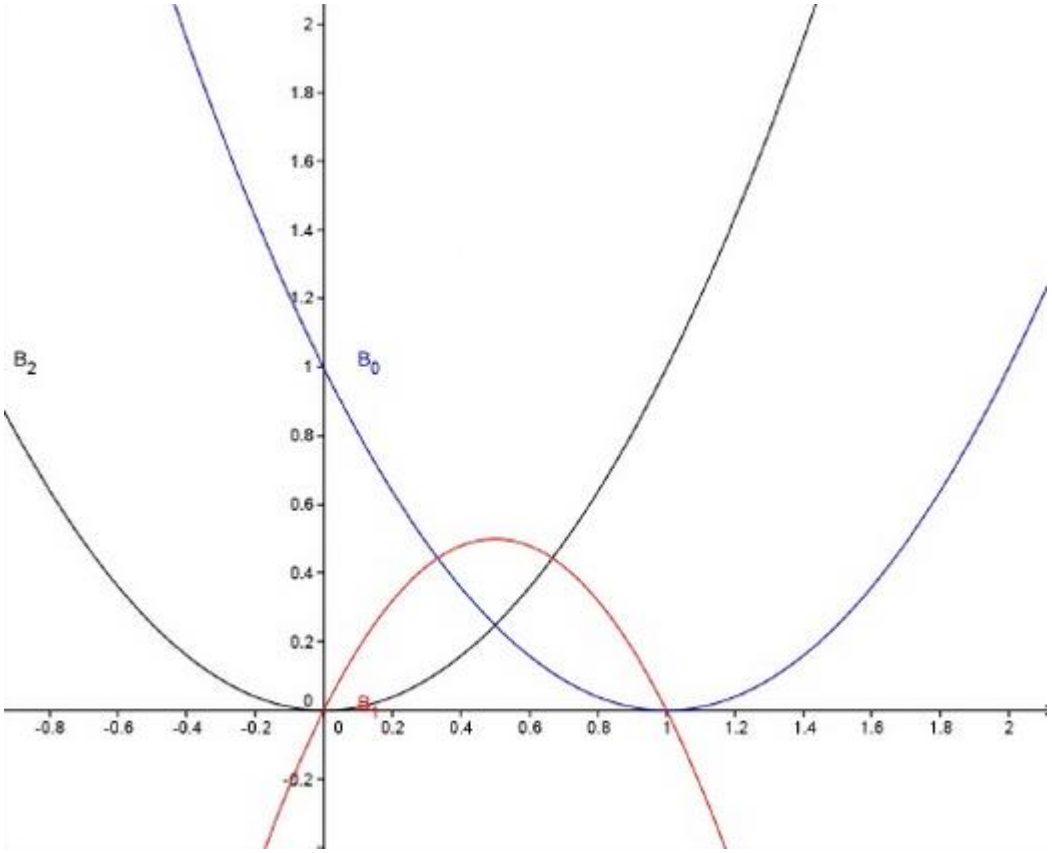
con $B_0^0(t) = 1$ y $B_j^n(t) = 0$ cuando j no pertenece {0,…,n}.

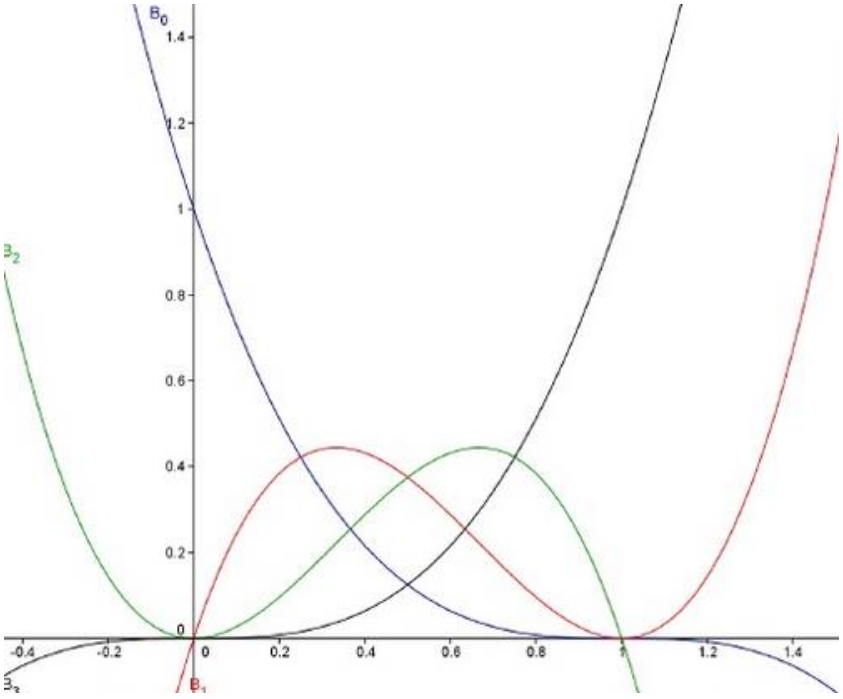
<u>Demostración:</u>

$$(1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t) =$$
$$(1-t)\binom{n-1}{i}t^i(1-t)^{n-1-i} + t\binom{n-1}{i-1}t^{i-1}(1-t)^{n-1-(i-1)} =$$
$$\binom{n-1}{i}t^i(1-t)^{n-i} + \binom{n-1}{i-1}t^i(1-t)^{n-i} =$$
$$\binom{n}{i}t^i(1-t)^{n-i} = B_i^n(t)$$

c) <u>Ejemplos</u>

Para hacer de todo el desarrollo teórico algo más fácil de comprender, utilizaremos una representación visual. Como antes hemos destacado que uno de los polinomios más sencillos de Bernstein son los de grado dos y tres. A continuación se visualizara una representación de los tres polinomios de Bernstein de grados dos y tres:


B2
B0
B1


B0
B2
B1

2. Curvas de Bézier

a) Introducción a las curvas de Bézier y el polígono de control

Las curvas de Bézier son un sistema desarrollado en los años 60 para el trazado de dibujos técnicos y diseño de aeronaves y automóviles. Su denominación se le atribuye a Pierre Bézier.

En la construcción de dichas curvas se utiliza una base distinta a la canónica {1,x,…,x^n} proporcionada por los polinomios de Bernstein: $\{B_0^n(t), \dots, B_n^n(t)\}$.

Siendo $B_i^n = \binom{n}{i} t^i (1-t)^{n-i}$ el polinomio i-ésimo de grado n de Bernstein. La ventaja de dicha base respecto a la canónica es que todos los polinomios presentan el mismo grado.

Por lo tanto podemos representar las curvas polinómicas de grado n como combinación de estos polinomios:

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

Donde los coeficientes c_i son los puntos del plano elegidos para dicha curva. Estos coeficientes $\{c_0, \dots, c_n\}$ forman lo que se llama el polígono de control de la curva de Bézier de grado n. Una curva de Bézier de dicho grado tiene por tanto un polígono de control de n+1 vértices.

b) Propiedades

1) **Interpolación de los extremos:**
   La curva de Bézier pasa por los vértices c_0 y c_n.

$$c(0) = \sum_{i=0}^{n} c_i \, B_i^n(0) = c_0 \quad c(1) = \sum_{i=0}^{n} c_i \, B_i^n(1) = c_n$$

2) **Simetría:**
   Los polígonos de control $\{c_0, \dots, c_n\}$ y $\{c_n, c_{n-1}, \dots, c_0\}$ definen la misma curva de Bézier como conjunto sólo que en el sentido contrario, es decir en el primer caso
   $c(0) = c_0$ y $c(1) = c_n$ y en el segundo caso $c(0) = c_n$ y $c(1) = c_0$. Esta propiedad se

debe a la simetría de los números combinatorios:

$$\binom{n}{i} = \frac{n!}{i!\,(n-i)!} = \binom{n}{n-i}$$

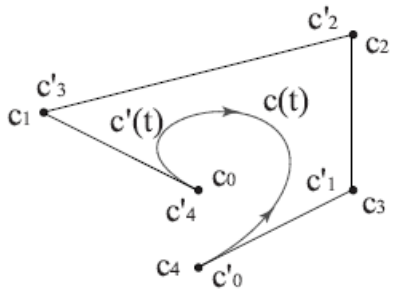
Lo que implica la relación entre los polinomios de Bernstein

$$B_i^n(1-t) = B_{n-i}^n(t)$$

Por tanto:

$$c(1-t) = \sum_{i=0}^{n} c_i\, B_i^n(1-t) = \sum_{i=0}^{n} c_i B_{n-i}^n\,(t) = \sum_{j=0}^{n} c_{n-j} B_j^n(t)$$

Tomando $j = n - i$.



3) <u>Invarianza afín:</u>

Aunque la curva esté definida en un principio en el intervalo [0,1] podemos trasladarla a cualquier otro intervalo mediante una aplicación afín:

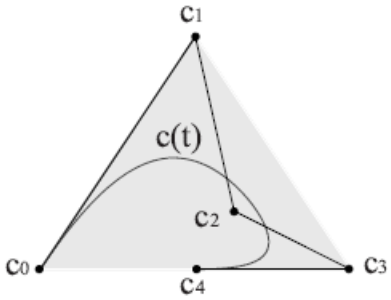
$$t(u) = \frac{u-a}{b-a} u \in [a, b]$$

De modo que u = a correspondería a t = 0 y u = b correspondería a t = 1. La curva con este nuevo dominio quedaría así:

$$c^*(u) = c\big(t(u)\big) = c\left(\frac{u-a}{b-a}\right) \quad u \in [a,b]$$

4) <u>Envolvente convexa:</u>

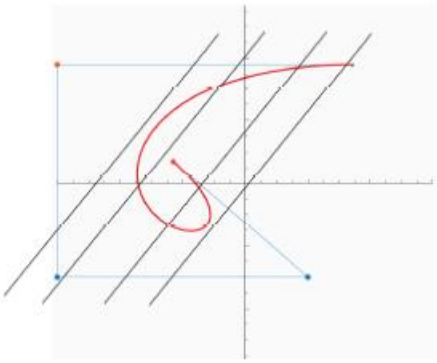
La envolvente convexa de un conjunto de puntos es el menor polígono que contiene a todos los puntos de dicho conjunto.



La curva de Bézier asociada a un polígono de control siempre está incluida en la envolvente convexa del conjunto de los vértices del polígono. Esto nos permitirá por ejemplo discernir si dos curvas de Bézier pueden cortarse observando si las envolventes convexas de ambos polígonos de control se cortan.

5) <u>Disminución de la variación:</u>

Si una recta r corta un polígono de control m veces, entonces la recta cortará a lo sumo la curva de Bézier m veces.



Esto es entendible gracias al aumento del grado que se le puede aplicar a una curva de Bézier. Dicho aumento de grado se obtiene mediante la multiplicación de la expresión de la curva por $1 = (1 - t + t)$:

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

$$= \sum_{i=0}^{n} c_i \frac{n!}{i!\,(n-i)!} \left(t^{i+1}(1-t)^{n-i} + t^i (1-t)^{n+1-i}\right) =$$

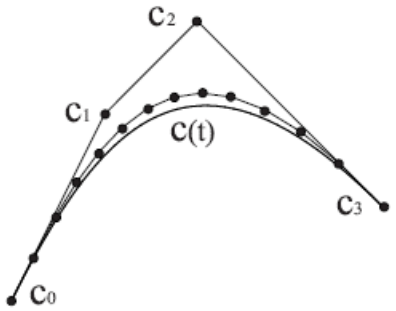
$$= \sum_{i=0}^{n} c_i \left(\frac{i+1}{n+1} B_{i+1}^{n+1}(t) + \frac{n+1-i}{n+1} B_i^{n+1}(t)\right) =$$

$$= \sum_{i=0}^{n+1} \left(\frac{i}{n+1} c_{i-1} + \frac{n+1-i}{n+1} c_i\right) B_i^{n+1}(t) = \sum_{i=0}^{n+1} c_i^1 B_i^{n+1}(t)$$

Identificando los nuevos vértices del polígono de control como $\{c_0^1, \dots, c_{n+1}^1\}$.

$$c_i^1 = \left(1 - \frac{i}{n+1}\right) c_i + \frac{i}{n+1} c_{i-1}$$

Iterando dicho proceso de elevación de grado con los nuevos vértices que se van obteniendo vamos redondeando el polígono de control. En el límite de dichas elevaciones de grado acabamos obteniendo la curva de Bézier (siendo esto un método muy poco eficiente de obtener la curva partiendo de una interpolación lineal).



Con ello al ir suavizando las esquinas del polígono de control vamos reduciendo la posibilidad de intersección de la recta r con la curva.

6) <u>Precisión lineal:</u>

Este proceso explicado anteriormente de elevación de grado puede ser utilizado para expresar el polinomio lineal t como un polinomio de grado n:

$$t = t(1 - t + t)^{n-1}$$

Así el polinomio t tiene por único polígono de control el correspondiente a $\{c_0 = 0 , \dots , c_1 = 1\}$ ya que:

$$c(t) = c_0 B_0^1(t) + c_1 B_1^1(t) = c_0(1-t) + c_1 t = t$$

La expresión del polinomio lineal t como polinomio de grado n quedaría de forma:

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

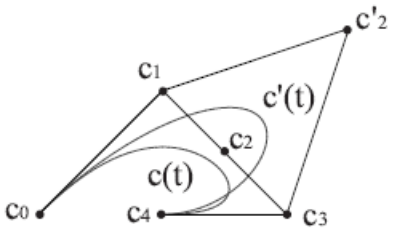
$$= \sum_{i=1}^{n} \frac{(n-1)!}{(i-1)!\,(n-i)!} t^{i}(1-t)^{n-i} = \sum_{i=1}^{n} \frac{i}{n} B_i^n(t) = \sum_{i=1}^{n} c_i^n B_i^n(t)$$

Siendo $c_i^n = \frac{i}{n}$.

A esta propiedad se la conoce como precisión lineal.

7) <u>Control pseudo-local:</u>

Si se desplaza un punto del polígono de control que forma la curva de Bézier conseguimos el desplazamiento de la curva en las proximidades de dicho punto movido, pero también en todo el resto de la curva, es por esto que se denomina a esta propiedad como control pseudo-local y no control local.



Esto se debe a que el máximo polinomio i-ésimo de Bernstein está en $t = \frac{i}{n}$:

$$0 = \frac{dB_i^n(t)}{dt} = \frac{n!}{i!\,(n-i)!}\,t^{i-1}(1-t)^{n-i-1}\{i(1-t) \overset{\Rightarrow}{-} (n-i)t\} nt = i$$

Por lo que la variación en el vértice i afecta más a los valores de c(t) en las proximidades del máximo de la curva.

c) Algoritmo de De Casteljau

Para la construcción de la curva de Bézier, a nivel práctico, los polinomios de Bernstein no nos son demasiado eficientes aunque nos hayan servido para dar las propiedades de las curvas de Bézier, por ello se suele utilizar el algoritmo recursivo dado por De Casteljau. La ventaja del algoritmo de De Casteljau es que sólo implica sumas y su sencillez de cálculo, por ello es un buen método de cálculo a nivel operacional para hallar curvas de Bézier.

Definición:

Dados n+1 puntos $\{c_0, \dots, c_n\}$ de R^2 se define la curva de Bézier asociada como:

$$c: [0,1] \to R^2$$

Dada por $c(t) = c_0^n(t)$ siendo $c_0^n(t)$ el resultado final del algoritmo recursivo definido por:

- $c_i^0(t) = c_i$
- $c_i^r(t) = (1-t)c_i^{r-1}(t) + tc_{i+1}^{r-1}(t)$

Para r = 1 , … , n e i = 0 , … , n-r.

Demostración:

Vamos a demostrar que dicho algoritmo recursivo equivale a la construcción de la curva de Bézier por polinomios de Bernstein.

$$c(t) = \sum_{i=0}^{n-r} c_i^r(t) B_i^{n-r}(t)$$

El caso r = 0 es trivial, suponiéndolo cierto para r-1 vamos a intentar probarlo:

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

Sustituimos en la fórmula general:

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

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

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

3. INTERPOLACIÓN Y OTRAS APLICACIONES

a) Interpolación de funciones polinómicas y funciones a trozos

Nuestro interés en estudiar los Polinomios de Bernstein se centra, entre otros aspectos, en la posibilidad de interpolar y/o aproximar funciones con una gran precisión. Partimos de la función más sencilla, f(x) = 1, así como f(x) = x, para encontrar un polinomio de Bernstein que se amolda tanto a ella, que da lugar a la misma función en el intervalo de trabajo [0,1]. Si tomamos la siguiente ecuación,

$$\sum_{i=0}^{k} B_{k,i}(x) = 1$$

y derivamos

$$\sum_{i=0}^{n} \binom{n}{i} x^{i-1}(1-x)^{n-i-1}[i(1-x) - (n-1)x] = 0$$

$$\sum_{i=0}^{n} \binom{n}{i} x^{i-1}(1-x)^{n-i-1}(i - nx) = 0$$

Si ahora multiplicamos por $x(1-x)$ tenemos

$$\sum_{i=0}^{n} \binom{n}{i} x^{i-1}(1-x)^{n-i-1}(i-nx) = 0$$

$$\Leftrightarrow \sum_{i=0}^{n} x^{i}(1-x)^{n-1} i = n \sum_{i=0}^{n} \binom{n}{i} x^{i}(1-x)^{n-i} x$$

$$\Leftrightarrow \sum_{i=0}^{n} \binom{n}{i} x^{i}(1-x)^{n-i} \frac{i}{n} = \sum_{i=0}^{n} \binom{n}{i} x^{i}(1-x)^{n-i} x$$

$$\Leftrightarrow \sum_{i=0}^{n} \binom{n}{i} x^{i}(1-x)^{n-i} \frac{i}{n} = x$$

encontrando el Polinomio de Bernstein para la función f(x) = x. Tomando esta última expresión y derivando podemos aproximar la función $f(x) = x^2$ de una manera totalmente exhaustiva, como se verá:

$$\sum_{i=0}^{n} \frac{i}{n} \binom{n}{i} x^{i-1}(1-x)^{n-i-1}(i-nx) = 1$$

y ahora multiplicando por $\frac{x(1-x)}{n}$se obtiene

$$\sum_{i=0}^{n} \frac{i}{n} \binom{n}{i} x^{i}(1-x)^{n-i} \left(\frac{i}{n} - x\right) = \frac{x(1-x)}{n}$$

$$\Leftrightarrow \sum_{i=0}^{n} \frac{i^2}{n^2} \binom{n}{i} x^{i}(1-x)^{n-i} - x^2 = \frac{x(1-x)}{n}$$

o bien

$$B_n(x^2; x) = x^2 + \frac{x(1-x)}{n}$$

todo ello en el intervalo [0,1]. Comprobando la siguiente operación:

$$\|x^2 - B_n(x^2, x)\| = \max_{0 \leqslant x \leqslant 1} |x^2 - B_n(x^2; x)| = \frac{1}{4n}$$

si $n \to \infty$

$$\|x^2 - B_n(x^2, x)\| \to 0$$

Por tanto, converge a 0 y el Polinomio de Bernstein aproxima tanto como se quiera a f(x)= x².

Encontramos, pues, una fuerte razón, para estudiar los Polinomios de Bernstein, esto es, interpolar y aproximar funciones polinómicas.

Del mismo modo que hemos estudiado el caso de aproximación de funciones, también podemos plantearnos el problema de funciones poligonales o lineales a trozos. Consideramos entonces un conjunto de vértices para la curva poligonal de forma (x_i, f_i)para i = 0,...,N donde por simplicidad asumimos que

$$0 = x_0 < x_1 < \ldots < x_N = 1$$

Estos puntos definen una interpolación lineal por trozos dado por

$$f(x) = \frac{f(x_i)(x_{i+1} - x) + f(x_{i+1})(x - x_i)}{x_{i+1} - x_i}$$

con $x \in [x_i, x_{i+1}]$, y esta función puede entonces aproximarse por polinomios de Bernstein. Nuevamente, el polinomio de Bernstein solo coincide en los puntos x_0 y x_Ny no interpola a ningún otro punto. Lo anterior es congruente con el teorema de aproximación en el sentido de que el polinomio de Bernstein provee una buena aproximación uniforme. Puede verse también, que si la función poligonal es cóncava entonces todos los polinomios de Bernstein asociados a ella están por debajo de la función y se aproximan de forma monótona.

b) Demostración del Teorema de Weierstrass y añadidos

Desde el punto de vista teórico, Bernstein utilizó sus polinomios para demostrar el Teorema de Aproximación de Weierstrass. Reza así:

Teorema de Aproximación de Weierstrass. *Si f(x) es una función continua en un intervalo [a,b] entonces para cualquier ε >0 existe un $n \in N$y un polinomio $p_n(x)$de grado n tal que*

$$|f(x) - p_n(x)| < \varepsilon$$

para todo $x \in [a, b]$.

A continuación, probamos la versión de Bernstein del mismo teorema.

Teorema. *Si f es continua en [0,1], entonces $B_n(f;\cdot)$converge uniformemente a f en [0,1].*

Demostración. Supongamos que f no es idénticamente cero y sea

$$M = \sup\{|f(x)|: x \in [0,1].\}$$

Consideremos $\varepsilon > 0$. A partir de que f es continua en un intervalo cerrado se sigue que f es uniformemente continua en [0,1], es decir, existe δ > 0 tal que si :

$$x, y \in [0,1] \text{y} |x - y| < \delta \quad \text{entonces}$$
$$|f(x) - f(y)| < \frac{\varepsilon}{2}$$

Sea $N = M/(\varepsilon\delta^2)^2$.Debemos demostrar que

$$|B_n(f,x) - f(x)| < \varepsilon \forall x \in [0,1] \quad \text{y } \forall n > N$$

Fijemos una $x \in [0,1]$y tomemos n > N. f(x) podemos desarrollarla como:

$$f(x) = \sum_{k=0}^{n} f(x) \binom{n}{k} x^k (1-x)^{n-k}$$

y además

$$|B_n(f;x) - f(x)| \leqslant \sum_{k=0}^{n} |f\left(\frac{k}{n}\right) - f(x)| \binom{n}{k} x^k (1-x)^{n-k}$$

Para estimar esta suma, dividimos el conjunto {0,1,2,...,n} en dos conjuntos A y B.

$$k \in A, \text{si} \quad |\frac{k}{n} - x| < \delta$$

$$k \in B, \text{si} \quad |\frac{k}{n} - x| \geq \delta$$

Para $k \in A$tenemos que

$$|f\left(\frac{k}{n}\right) - f(x)| < \varepsilon$$

según la primera consideración de la demostración y

$$\sum_{k\in A} |f\left(\frac{k}{n}\right) - f(x)| \binom{n}{k} x^k (1-x)^{n-k} \leq \sum_{k\in A} \frac{\varepsilon}{2} \binom{n}{k} x^k (1-x)^{n-k} \leq \frac{\varepsilon}{2}$$

Para $k \in B$, tenemos

$$|\frac{k - nx}{n}| \geq \delta \Rightarrow (k - nx)^2 \geq n^2\delta^2$$

y además

$$\sum_{k\in A} |f\left(\frac{k}{n}\right) - f(x)| \binom{n}{k} x^k (1-x)^{n-k} \leq 2M \sum_{k\in B} \binom{n}{k} x^k (1-x)^{n-k}$$
$$\leq \frac{2M}{n^2\delta^2} \sum_{k\in B} (k-nx)^2 \binom{n}{k} x^k (1-x)^{n-k}$$

por un resultado anterior

$$\frac{2M}{n^2\delta^2}\left(\frac{n}{4}\right) = \frac{M}{2n\delta^2} < \frac{M}{2N\delta^2} = \frac{\varepsilon}{2}$$

se sigue entonces que

$$|B_n(f;x) - f(x)| < \varepsilon$$

Esta prueba del Teorema de aproximación de Weierstrass demuestra que usando $B_n(f;x)$se produce una base para obtener buenas aproximaciones para funciones.

c) Uso en la ingeniería y diseño

Ahora que hemos desarrollado ampliamente las utilidades del polinomio de Bernstein, es hora de hablar acerca de las curvas de Bézier, las cuales hemos ido tratando a lo largo del trabajo. En este apartado, nos dedicamos a comentar sus aplicaciones más importantes. Entre ellas destaca el diseño de gráficos en ordenador. Las curvas de Bézier han sido ampliamente usadas en los gráficos generados por ordenador en el modelado de curvas suaves. Como la curva está completamente contenida en la envolvente convexa de los puntos de control, dichos puntos pueden ser visualizados gráficamente sobre el área de trabajo y usados para manipular la curva de una forma muy intuitiva. Las curvas cuadráticas y cúbicas son muy corrientes, siendo posible la aplicación de transformaciones afines sobre ellas. Las curvas de grados superiores son más difíciles de evaluar. Cuanto más complejas son las superficies que se necesitan, las curvas de bajo orden son menos apropiadas. Para garantizar la suavidad de las curvas el punto de control en el que se juntan dos curvas y el punto de control sobre cualquiera de los lados debe ser colineal. Esta opción está frecuentemente desactivada en programas como Adobe Illustrator o Inkscape. Estas curvas poli-Bézier pueden ser observadas en el formato de archivo SVG.

El método más simple para rasterizar una curva de Bézier es evaluarla en muchos

puntos espaciados, muy próximos entre sí, y escanearla aproximando la secuencia de segmentos lineales.

Esta manera de proceder no garantiza un resultado con la suficiente suavidad porque los puntos pueden estar espaciados demasiado separados. A la inversa, se pueden generar bastantes puntos de control en áreas donde la curva está cercana a la forma lineal. Un método adoptado, muy común, es la subdivisión recursiva, en el que los puntos de control de la curva son ajustados para ver si se aproxima a segmentos lineales sin pequeñas tolerancias. Si esto no se logra, la curva subdividida paramétricamente en dos segmentos $0 \leq t \leq 0.5 \text{y} 0.5 \leq t \leq 1 \text{y}$ el mismo procedimiento se aplica por recursivamente a cada mitad.

También hay métodos que usan la diferenciación, pero se debe tener cuidado y analizar los errores de propagación. Los métodos analíticos donde un desdoble es intersecado con cada línea escaneada hallando raíces de polinomios de grado tres (por segmentación cúbica) y con múltiples raíces, pero son frecuentes en la práctica.

Además de los gráficos informáticos, las últimas innovaciones dan lugar al uso de la curva de Bézier como método para sintetizar ondas sonoras que se pueden usar para crear sonidos (usando de nuevo curvas del mismo tipo). Es la llamada síntesis de Bézier.

Es notable el aporte práctico que estos de matemáticos hicieron al diseño computacional, al punto que todo lo que actualmente se entiende como "vectorial" en diseño, está de alguna manera relacionado con el trabajo de Bézier.

Otra línea está en el estudio de las llamadas "*splines*", que fueron descubierta por Isaac Schoenberg, y corresponden a una generalización de las curvas de Bézier. Análogamente, la generalización del algoritmo de De Casteljau para splines, corresponde al algoritmo de De Boor (desarrollado por Carl de Boor).

El resultado general, es, al menos en cuanto al diseño, un conjunto de herramientas que permite a diseñadores describir matemáticamente curvas, aunque tal matemática no sea visible. La gran ventaja de los gráficos vectoriales, es que si son construidos en unas determinadas dimensiones, son perfectamente ampliables sin "pixelar" los resultados, lo que se suele describir como ser independientes de la resolución.

4. Implementación en ordenador

Para realizar la implementación en ordenador de los algoritmos de construcción de las curvas de Bézier hemos usado el programa de cálculo numérico Octave. Hemos creado dos funciones, una que obtiene la expresión de la curva directamente de los polinomios de Bernstein pero que puede resultar poco eficiente para polígonos de control con muchos vértices y otra más eficiente que hace uso del algoritmo de De Casteljau. La primera función tiene un código muy simple:

function [bx,by] = bezierbernstein(x,y)

```
n = length(x);
bx = zeros(1,n);
by = zeros(1,n);
Bn = bernstein(n-1);
for i=1:n
bx = bx + x(i)*Bn(i,:);
by = by + y(i)*Bn(i,:);
endfor
endfunction
```

La función bernstein(n) devuelve una matriz con los polinomios de Bernstein de orden n. Esta función obtiene las coordenadas de la curva en función de x como la sumatoria de los polinomios de Bernstein del orden necesario por las coordenadas de los puntos de control. Es muy sencilla de programar en Octave ya que tenemos la ventaja de que todos los polinomios de Bernstein que se usan son del mismo grado, lo que evita discordancia en las dimensiones de los vectores.
La segunda función es un poco más compleja, sobre todo porque tienes que hacer coincidir las dimensiones de todos los polinomios:

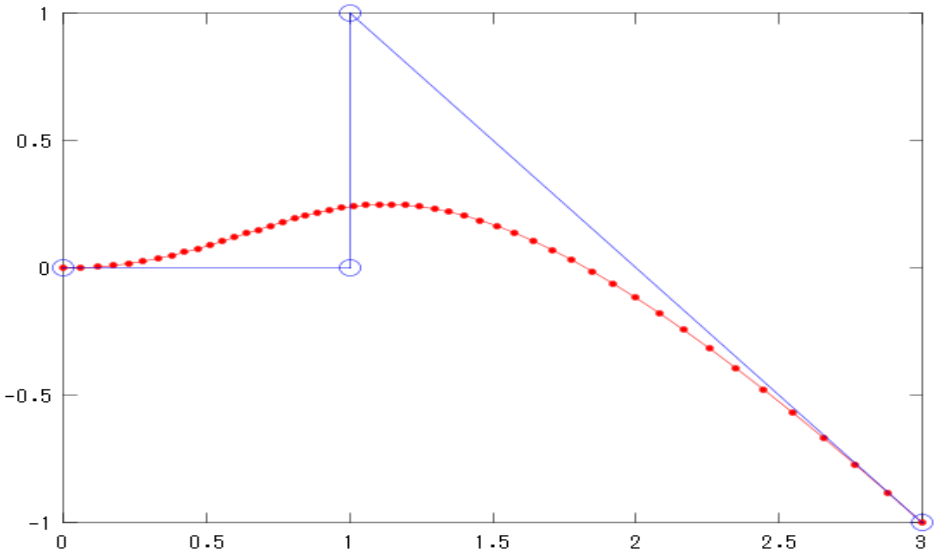
```
function [bx,by] = bezier(x,y)
  n = length(x);
  cx=zeros(n);
  cy=zeros(n);
  for z=1:n
       cx(z,n)=x(z);
       cy(z,n)=y(z);
  endfor
  for i=2:n+1
    for j=1:n-i+1
       cx(j,:) = (conv(cx(j,:),[-1 1]) + conv(cx(j+1,:),[1 0]))(2:n+1);
       cy(j,:) = (conv(cy(j,:),[-1 1]) + conv(cy(j+1,:),[1 0]))(2:n+1);
    endfor
  endfor
  bx=cx(1,:);
  by=cy(1,:);
endfunction
```

Esta función almacena en dos matrices distintas y por filas los polinomios que se van obteniendo. La función se sirve de que para calcular cada fila de la matriz solo se necesitan esa fila y la siguiente pero de la iteración anterior del bucle externo. Vamos a ver un ejemplo de uso de las funciones:

```
x = [0 1 1 3];
y = [0 0 1 -1];
[bx by] = bezier(x,y)
u=linspace(0,1,50);
v=polyval(bx,u);
```

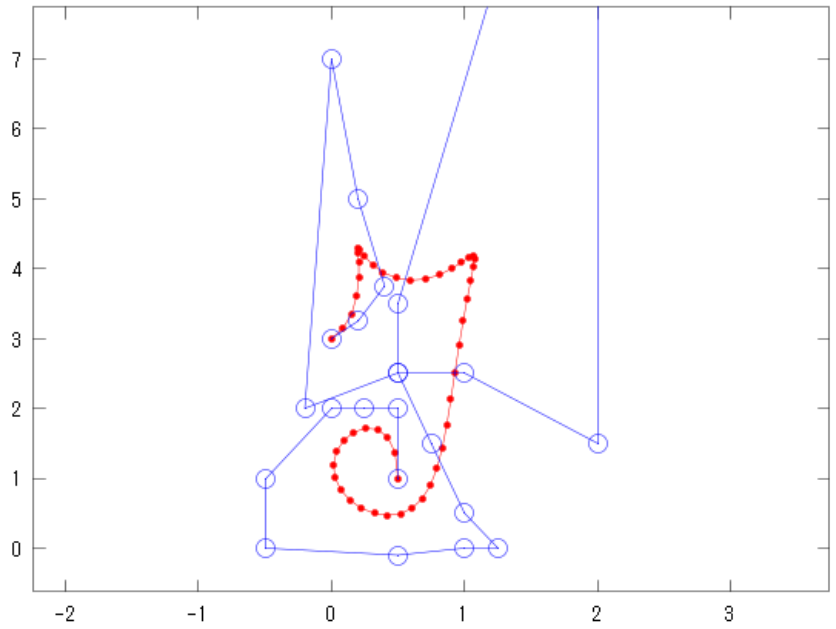
```
w=polyval(by,u);
plot(v,w,'-@r.',x,y,'-@bo')
```

Este código da como resultado la siguiente curva con su polígono de control.



La curva anterior se puede escribir como una función, veamos ahora un ejemplo de una curva de bezier que no es una función:

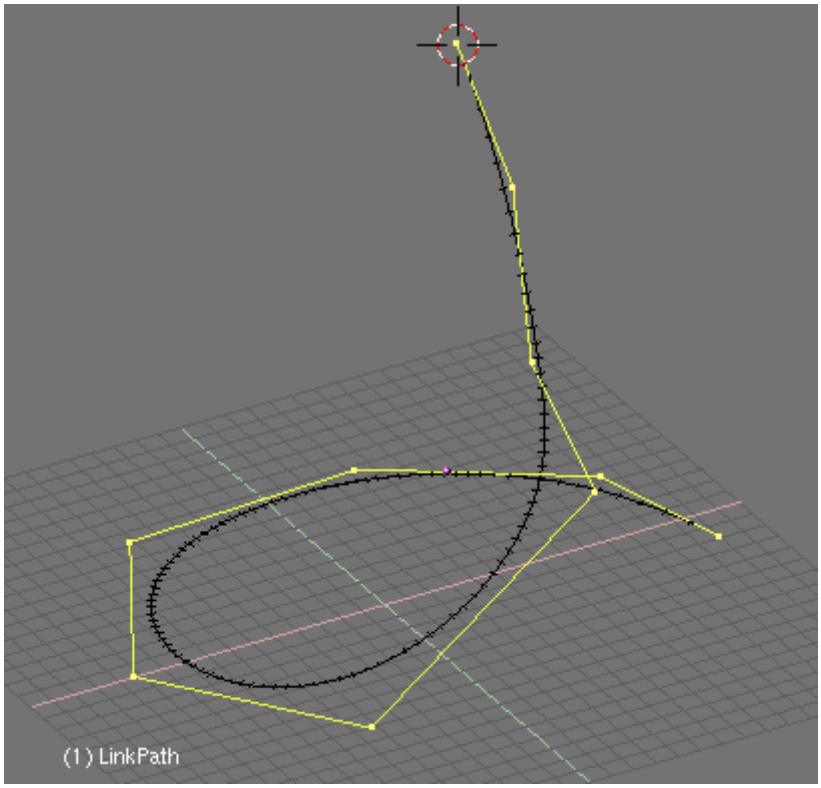
```
x = [0 0.2 0.4 0.2 0 -0.2 0.5 1 2 2 0.5 0.5 0.75 1 1.25 1 0.5 -0.5 -0.5 0 0.25 0.5 0.5];
y = [3 3.25 3.75 5 7 2 2.5 2.5 1.5 13 3.5 2.5 1.5 0.5 0 0 -0.1 0 1 2 2 2 1];
[bx by] = bezier(x,y)
u=linspace(0,1,50);
v=polyval(bx,u);
w=polyval(by,u);
plot(v,w,'-@r.',x,y,'-@bo')
```



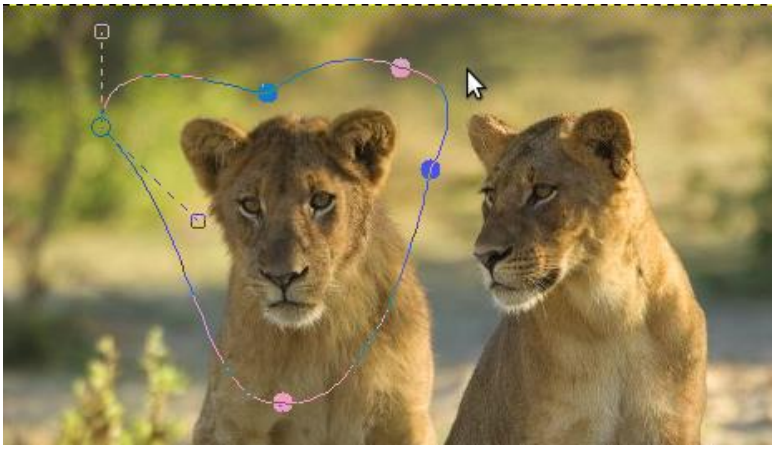
Cabe mencionar que ambas implementaciones dan lugar a la misma curva para el mismo polígono de control, como ya se había demostrado de forma teórica anteriormente.

Ejemplos en diseño gráfico

Las curvas de Bézier son muy usadas en el diseño gráfico básicamente para pintar curvas, tanto en dibujo en el plano como en modelado 3D. La mayoría de las aplicaciones más conocidas de diseño gráfico (Blender, Photoshop, Inkscape) incorporan una herramienta Bézier que permite dibujar curvas marcando los puntos de control. Lo que vemos aquí abajo se usó como base para el modelo de una cadena en Blender.



Otro ejemplo lo tenemos en el programa GIMP que las usa para hacer selecciones a partir de puntos seleccionados y de otros que el programa detecta automáticamente. Si nos fijamos la selección de aquí abajo tiene una forma parecida a la cara de la leona.



5. Bibliografía

a) http://es.wikipedia.org/wiki/Polinomio_de_Bernstein
b) http://delta.cs.cinvestav.mx/~mcintosh/comun/working/year2001/24may2001/node4.html
c) http://mcmai.izt.uam.mx/documentos/tesis/Gen.05-P/Loredo-CA-Tesis.pdf
d) http://www.miscelaneamatematica.org/Misc41/Meda_a.pdf
e) https://es.scribd.com/doc/95735336/5/Propiedades-polinomios-de-Bernstein
f) http://dcain.etsin.upm.es/~leonardo/tema2.pdf
g) http://www.uv.es/monterde/bezier2014_sesion2.pdf
h) http://www.etereaestudios.com/training_img/vectorial_tips/consejos.htm
i) An introduction to numerical analysis. Atkinson[ATK]
j) Análisis Numérico, RIchard L. Burden, J. Douglas Faires
k) http://www.ugr.es/~lorente/APUNTESMNM/capitulo5.pdf