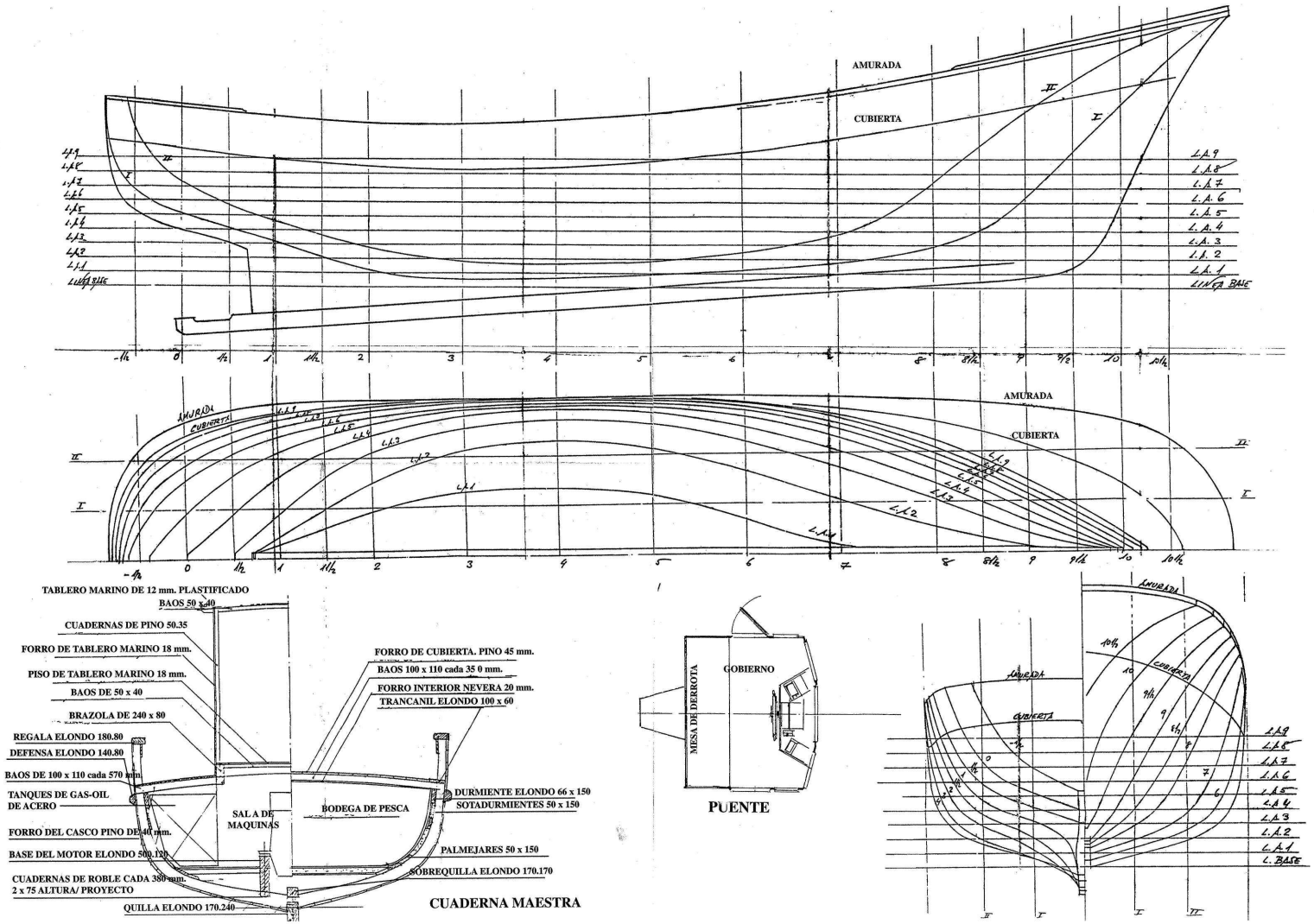
Curvas de Bézier

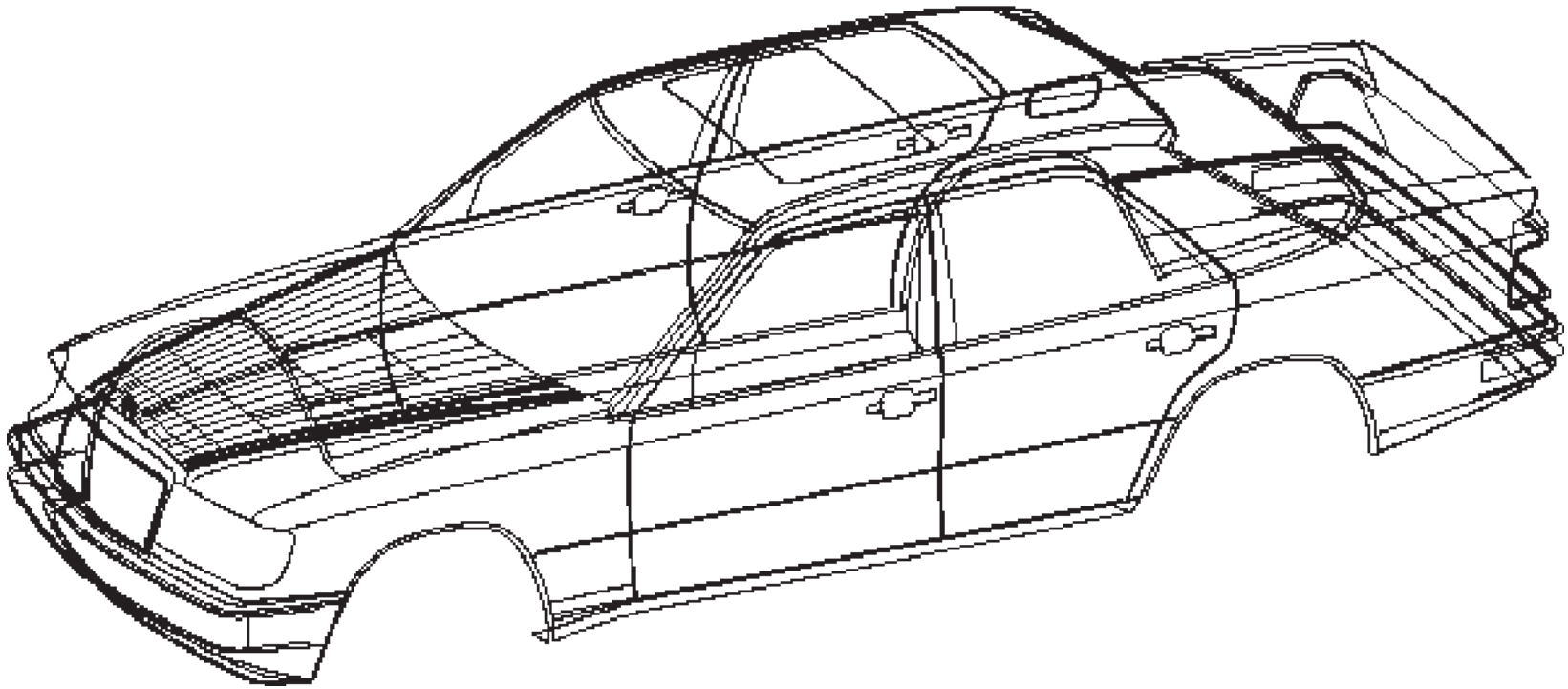
Leonardo Fernández Jambrina

Matemática Aplicada
E.T.S.I. Navales
Universidad Politécnica de Madrid

Plano de formas de un pesquero



Mercedes' bends



Analógico a digital

- Históricamente se trazaban los planos con junquillo (spline) en la industria naval, automovilística, aeronáutica...
- En la era de los ordenadores (1950-1960), ¿cómo digitalizar la información?
- Solución: Pierre Bézier (Renault), Paul de Casteljau (Citroën).

Estructura del curso

- Curvas polinómicas.
- Curvas racionales.
- Curvas *spline*.
- Superficies de Bézier.
- Generación de superficies.

Algunas referencias


G. Farin
Curves and Surfaces for CAGD: a Practical Guide.
Morgan Kaufmann Publishers, San Francisco, 2002.


J. Hoschek, D. Lasser
Fundamentals of Computer Aided Geometric Design.
AK Peters Ltd., Wellesley, 1993


D.F. Rogers, J .A. Adams
Mathematical Elements for Computer Graphics.
McGraw-Hill, New York, 1990


L. Fernández Jambrina
http://dcain.etsin.upm.es/˜leonardo


L. Fernández Jambrina
http://ocw.upm.es/matematica-aplicada

Esquema

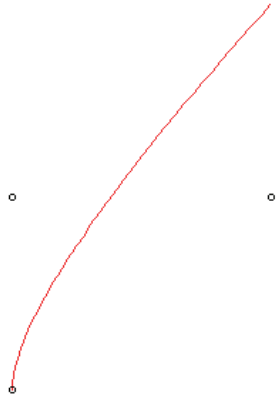
Curvas polinómicas

Representar curvas $c(t) = \sum_{i=0}^{n} a_i t^n, \quad t \in [0,1], \quad a_i \in \mathbb{R}^p$.

Problemas:

- Asimetría: a_0 es punto y el resto, vectores.
- Más asimetría: toda la información de la curva está concentrada en un entorno de a_0.
- Estabilidad numérica.

Comportamiento de los coeficientes bajo afinidades



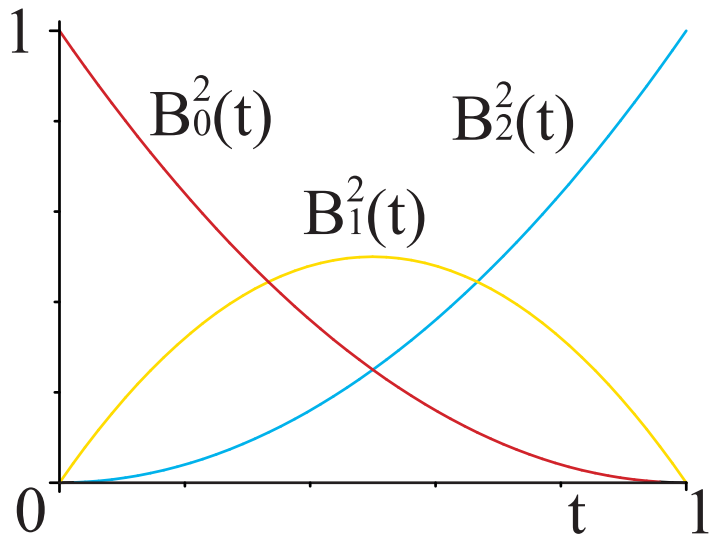
Bajo traslaciones.

Bajo rotaciones.

Polinomios de Bernstein

En vez de la base de polinomios, $\{1, t, \ldots, t^n\}$, usaremos la base de polinomios de Bernstein (coeficientes binomiales), $\left\{B_0^n(t), \ldots, B_n^n(t)\right\}$,

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



Polinomios de Bernstein

En vez de la base de polinomios, $\{1, t, \ldots, t^n\}$, usaremos la base de polinomios de Bernstein (coeficientes binomiales), $\left\{B_0^n(t), \ldots, B_n^n(t)\right\}$,

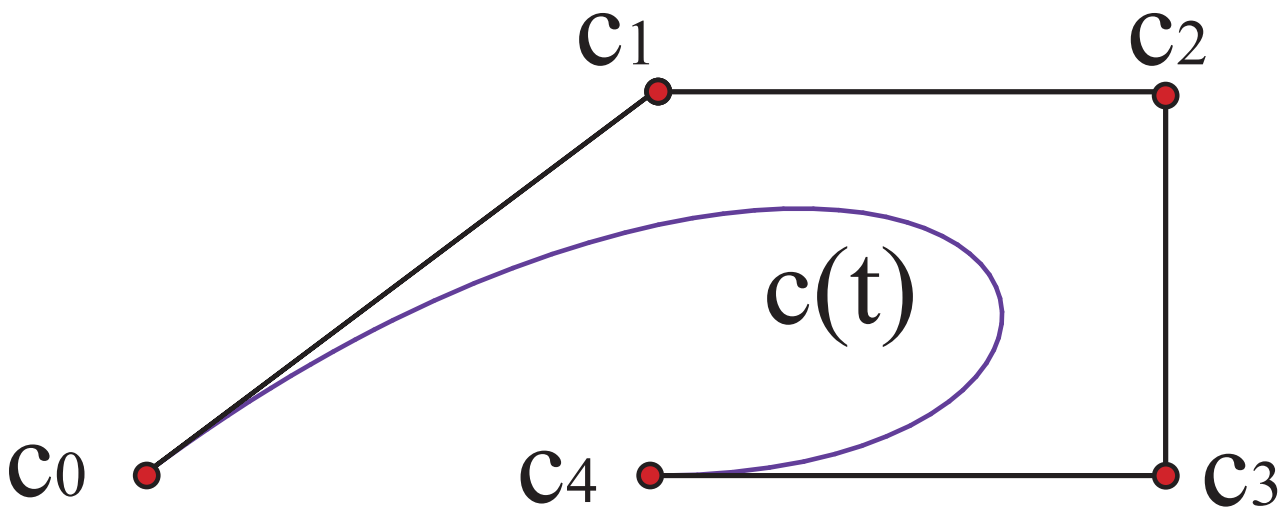
$$B_i^n(t) = \left(\begin{array}{c} n \\ i \end{array} \right) t^i (1-t)^{n-i}.$$

- Aparecen en la demostración del teorema de Weierstrass (aproximación uniforme de funciones continuas por polinomios).
- Partición de la unidad: $\sum_{i=0}^{n} B_i^n(t) = \big(t + (1-t)\big)^n = 1.$
- Origen: $B_0^n(0) = 1$, $B_i^n(0) = 0$, $i = 1, \ldots, n$.
- Final: $B_n^n(1) = 1$, $B_i^n(1) = 0$, $i = 0, \ldots, n-1$.

Curvas de Bézier

Representar curvas de grado n, $c(t) = \sum_{i=0}^{n} c_i B_i^n(t)$, $t \in [0,1]$, $c_i \in \mathbb{R}^p$,

$\{c_0, \ldots, c_n\}$ **polígono de control**.

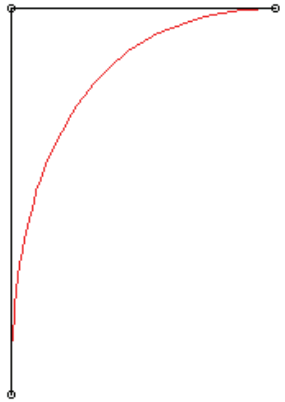


Curvas de Bézier

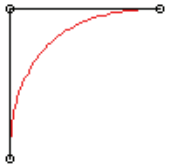
Representar curvas de grado n, $c(t) = \sum_{i=0}^{n} c_i B_i^n(t)$, $t \in [0,1]$, $c_i \in \mathbb{R}^p$, $\{c_0, \ldots, c_n\}$ **polígono de control**.

- Invariancia bajo transformaciones afines: para f afín,
  $f(c(t)) = \sum f(c_i) B_i^n(t)$, $\{c_0, \ldots c_n\} \xrightarrow{f} \{f(c_0), \ldots f(c_n)\}$.
- Extremos: $c_0 = c(0)$, $c_n = c(1)$.
- Envolvente convexa: la curva está contenida en el menor polígono convexo generado por $\{c_0, \ldots, c_n\}$.
- Derivadas: $c'(0) = n(c_1 - c_0)$, $c'(1) = n(c_n - c_{n-1})$,
  $c'(t) = n \sum \Delta c_i B_i^{n-1}(t)$, $\Delta c_i = c_{i+1} - c_i$.
- Control (cuasi)-local: moviendo un vértice se mueve la parte más próxima de la curva.

Comportamiento de los vértices bajo afinidades

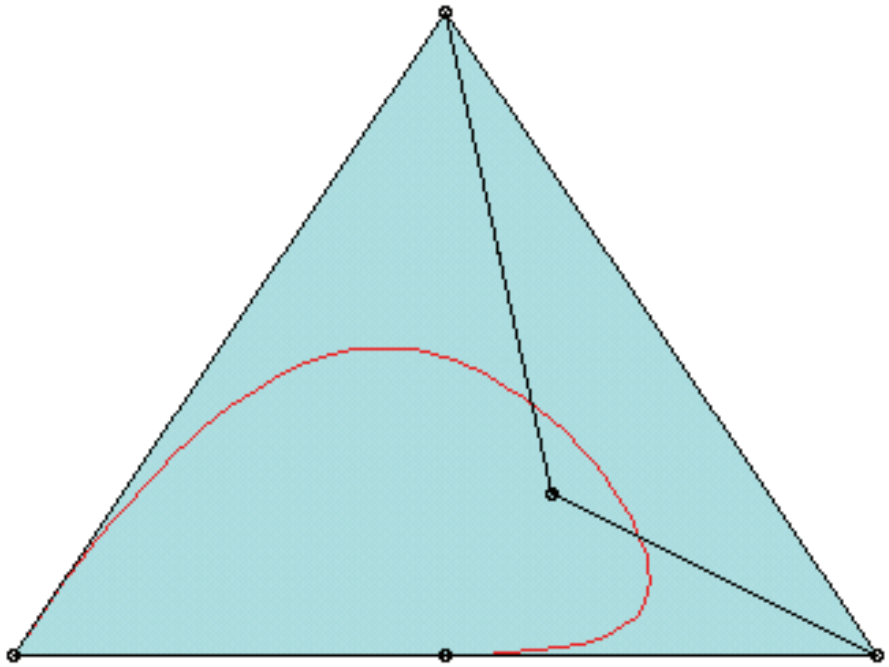


Bajo traslaciones.



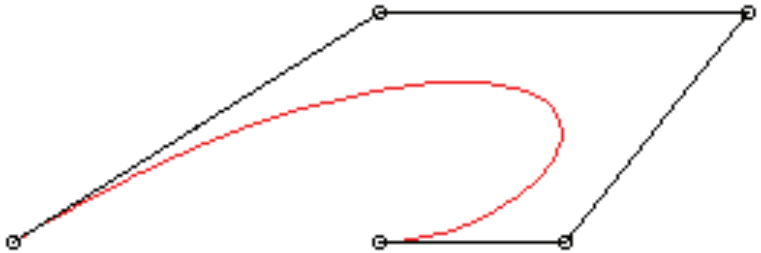
Bajo rotaciones.

Propiedad de la envolvente convexa del polígono



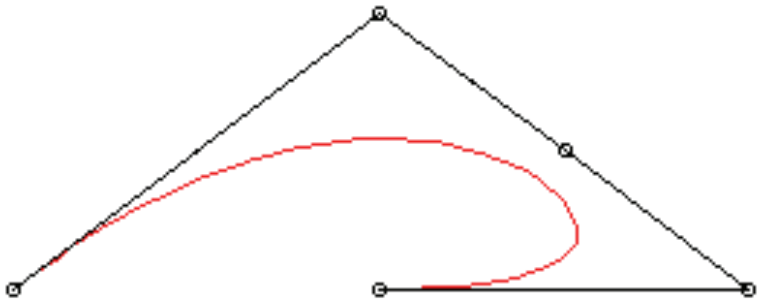
La curva se mantiene dentro de la envolvente convexa del polígono.

Comportamiento de los vértices extremos



La curva pasa por los vértices extremos del polígono y se mantiene tangente a los lados extremos del mismo.

Control cuasilocal



Moviendo un vértice del polígono se mueve la parte más próxima de la curva.

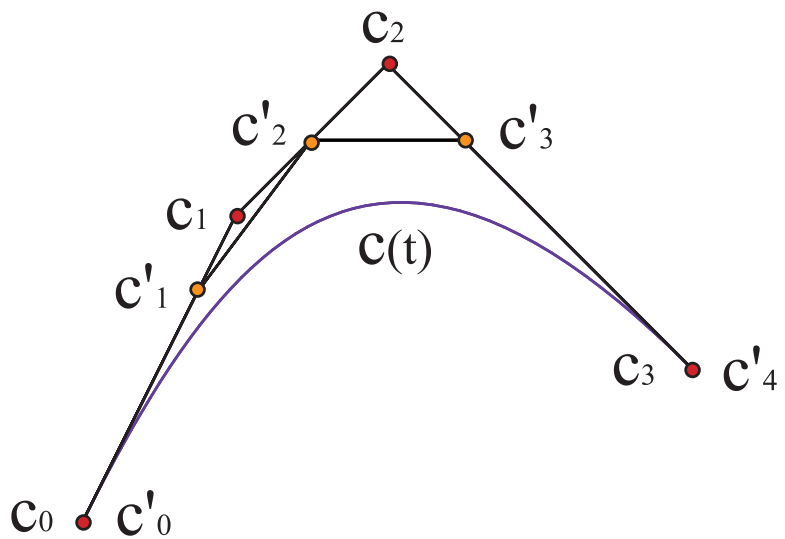
Elevación del grado

Una curva de Bézier de grado n se puede elevar formalmente de grado a $n+1$,

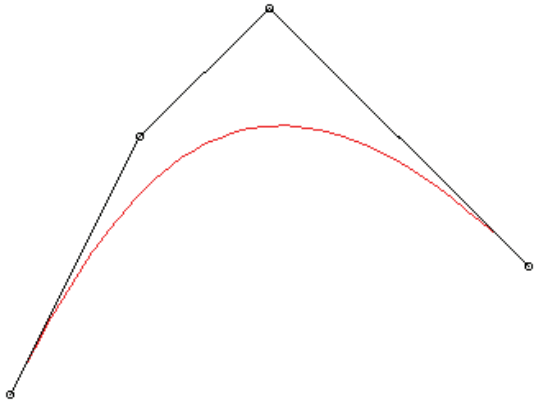
$$c(t) = c(t)\,((1-t)+t) = \sum_{i=0}^{n+1} c_i^1\, B_i^{n+1}(t),$$

reexpresándola con polinomios de Bernstein de grado $n+1$.

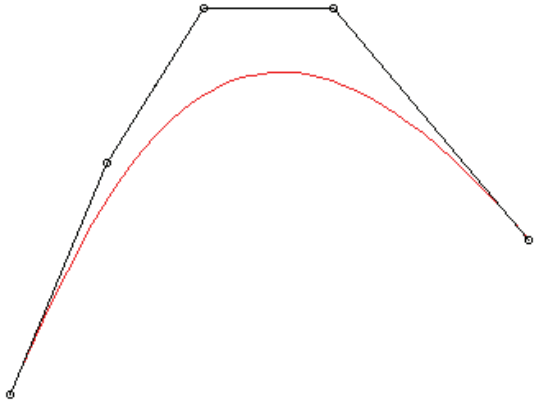
$$c_i^1 = \left(1 - \frac{i}{n+1}\right) c_i + \frac{i}{n+1} c_{i-1}, \quad i = 0, \ldots, n+1$$



Elevación del grado

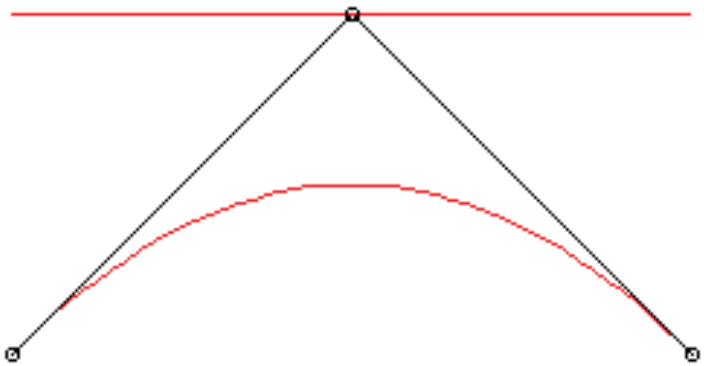


Elevación del grado.



Elevación sucesiva del grado.

Disminución de la variación



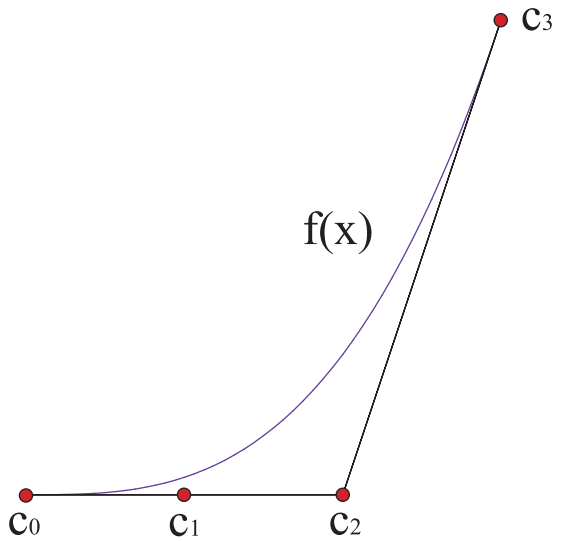
Una recta corta siempre a una curva en a lo sumo tantos puntos como al polígono.

Funciones o curvas

- Como caso particular de las curvas parametrizadas, podemos considerar las **gráficas** de funciones $f(t)$, $t \in [0,1]$.
- El intervalo $[0,1]$ no es problema, ya que mediante el cambio $t(u) = \frac{u-a}{b-a}$, podemos usar $u \in [a,b]$ tal que $t(a) = 0$, $t(b) = 1$.
- Son curvas parametrizadas de la forma $c(t) = (t, f(t))$.
- Si $f(t)$ es polinómica de grado n, tendrá un polígono de control $\{f_0, \ldots, f_n\}$.
- La coordenada x es un polinomio de grado uno, con lo cual su *polígono* será $\{0,1\}$.
- Elevando el grado a n, el polígono de x es $\{0, 1/n, \ldots, n/n\}$.
- Así pues, el polígono de la gráfica $c(t) = (t, f(t))$ es $\{c_0, \ldots, c_n\}$, con $c_i = (i/n, f_n)$. Es decir, los vértices tienen abscisas equiespaciadas.

Funciones o curvas

- Así pues, el polígono de la gráfica $c(t) = (t, f(t))$ es $\{c_0, \ldots, c_n\}$, con $c_i = (i/n, f_n)$. Es decir, los vértices tienen abscisas equiespaciadas.



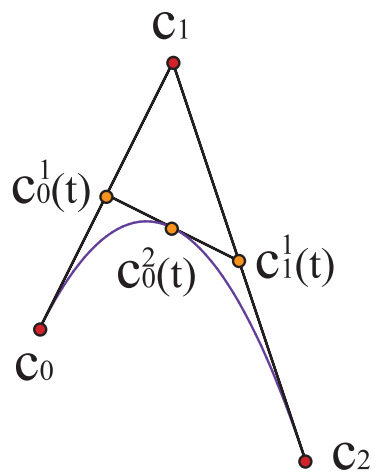
Algoritmo de De Casteljau

La formulación con polinomios de Bernstein es equivalente al siguiente algoritmo de n iteraciones:

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

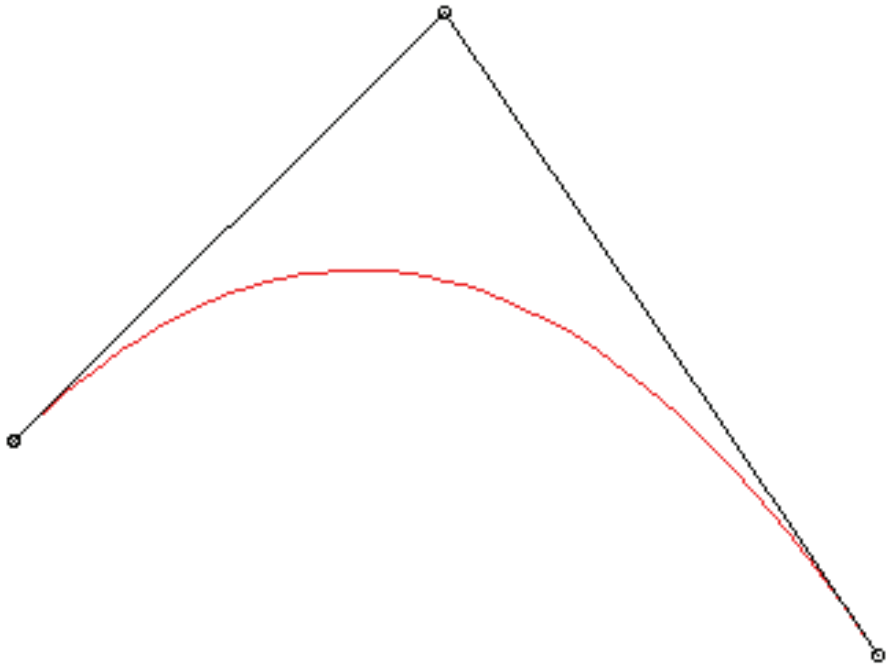
$$c_i^{r)}(t) := (1-t)c_i^{r-1)}(t) + tc_{i+1}^{r-1)}(t), \quad i = 0, \ldots, n-r, \quad r = 1, \ldots, n,$$

hasta llegar a un único punto: el valor $c(t) = c_0^{n)}(t)$.



- Derivada: $c'(t) = n\left(c_1^{n-1)}(t) - c_0^{n-1)}(t)\right)$.

Algoritmo de De Casteljau



Algoritmo de De Casteljau para una parábola.

Polarización de la parametrización

En vez de interpolar para un mismo t en los n pasos del algoritmo, interpolamos en el valor distinto t_i en cada paso, $c[t_1, \ldots, t_n]$.

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

Es una forma n-afín simétrica: para $\lambda + \mu = 1$,

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

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

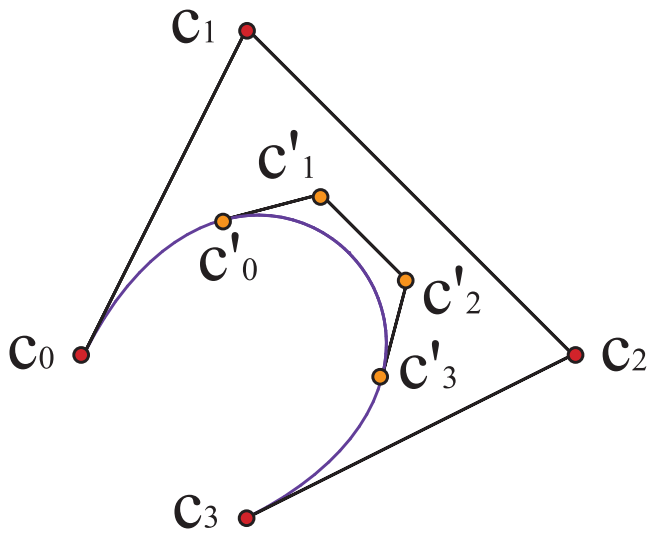
Propiedades de la polarización

- Reconstruye el polígono de control:

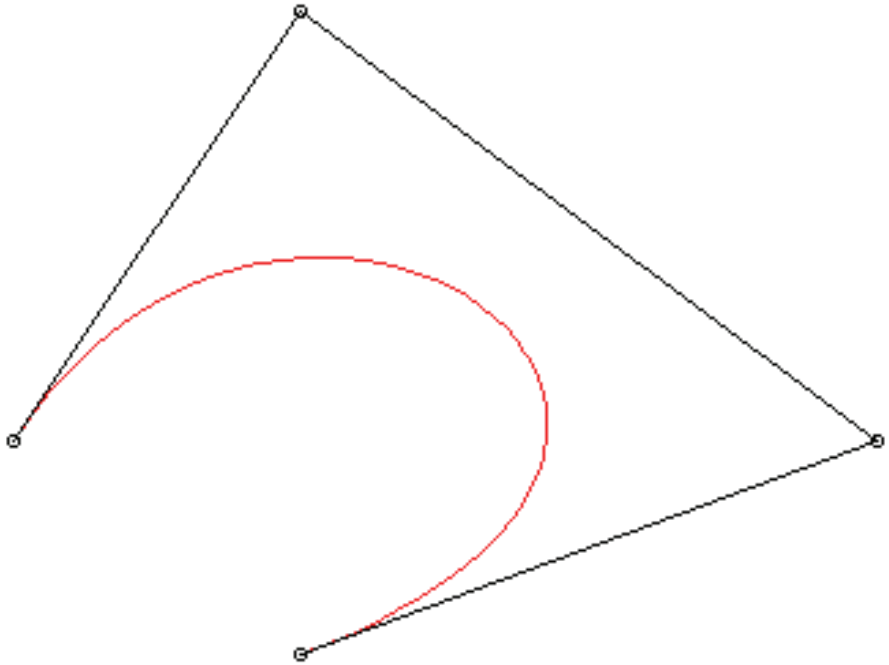
$$c[0^{<n-i>}, 1^{<i>}] = c_i, \qquad a^{<s>} = \underbrace{a, \ldots, a}_{s \text{ veces}}.$$

- Permite restringir la curva a un subintervalo $[a, b] \subset [0, 1]$, cuyo polígono es $\{\tilde{c}_0, \ldots, \tilde{c}_n\}$,

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



Restricción de curvas



La polar permite calcular el polígono de control de un tramo de una curva de Bézier conocida.

Derivación

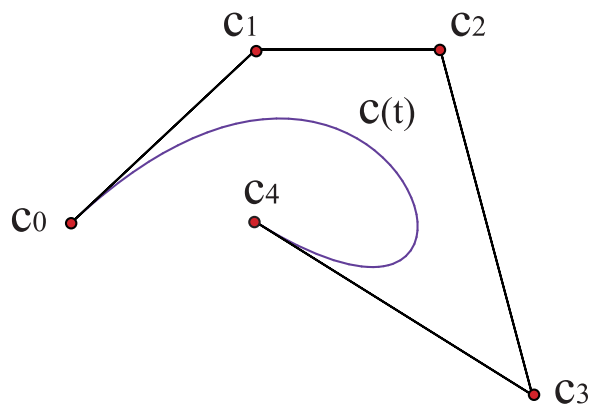
La derivada de una curva de Bézier es

$$\frac{\mathrm{d}c(t)}{\mathrm{d}t} = n\sum_{i=0}^{n-1} \Delta c_i B_i^{n-1}(t), \qquad \frac{\mathrm{d}^r c(t)}{\mathrm{d}t^r} = \frac{n!}{(n-r)!}\sum_{i=0}^{n-r} \Delta^r c_i B_i^{n-r}(t),$$

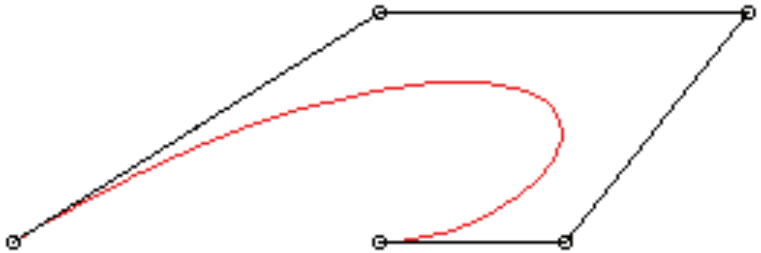
definiendo el vector diferencia como $\Delta c_i = c_{i+1} - c_i$.
La derivada es otra curva de polígono $\{n\Delta c_0, \ldots, n\Delta c_{n-1}\}$.
Como la curva pasa por los vértices inicial y final,

$$c'(0) = n\Delta c_0 = n(c_1 - c_0), \quad c'(1) = n\Delta c_{n-1} = n(c_n - c_{n-1}),$$

los vértices iniciales y finales indican las tangentes en los extremos.



Tangentes en los extremos



Las parejas de vértices de los extremos de la curva de Bézier determinan las tangentes en dichos extremos.

Uniones de curvas

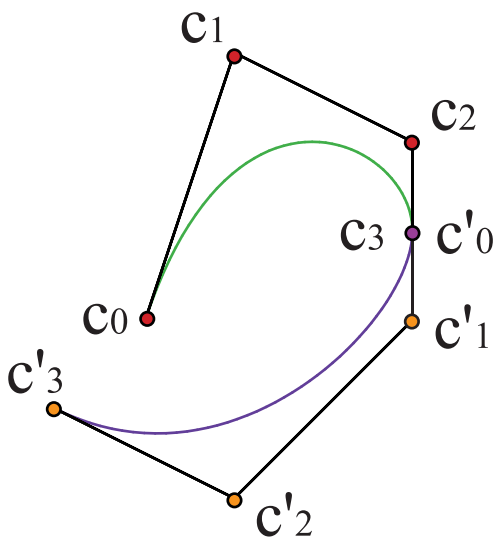
Para derivadas superiores,

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

con las diferencias de orden superior $\Delta^s c_i = \Delta^{s-1} c_{i+1} - \Delta^{s-1} c_i$.

Uniones de curvas

¿Cómo unimos dos curvas de grado n, definidas en intervalos adyacentes $[u_0, u_1]$, $[u_1, u_2]$ y polígonos $\{c_0, \ldots, c_n\}$, $\{\tilde{c}_0, \ldots, \tilde{c}_n\}$?



Uniones de curvas

¿Cómo unimos dos curvas de grado n, definidas en intervalos adyacentes $[u_0, u_1]$, $[u_1, u_2]$ y polígonos $\{c_0, \ldots, c_n\}$, $\{\tilde{c}_0, \ldots, \tilde{c}_n\}$?

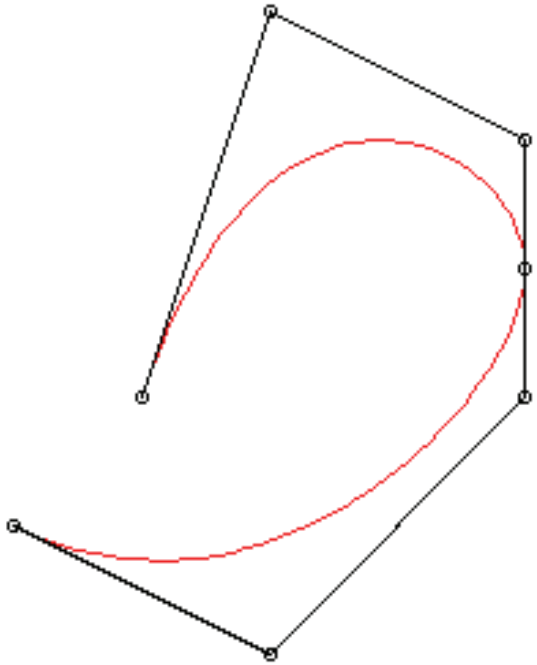
- Si queremos que sea continua, $c_n = \tilde{c}_0$.
- Si queremos que sea C^1,

$$\frac{\Delta c_{n-1}}{\Delta u_0} = \frac{\Delta \tilde{c}_0}{\Delta u_1}, \qquad \Delta u_0 = u_1 - u_0, \quad \Delta u_1 = u_2 - u_1.$$

- Si queremos que se a C^r (máximo $r = n - 1$),

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

Continuidad de las tangentes



Las parejas de vértices de los extremos de las curvas deben estar alineadas.

Interpolación

- Aunque este curso esté motivado por el modelado geométrico, las curvas de Bézier también tienen otros usos.
- Por ejemplo, el problema de *interpolar* una curva $c(t)$ de grado n que pase por $n+1$ puntos $a_0, \ldots, a_n$ en *instantes* $t_0, \ldots, t_n$,

$$a_0 = c(t_0), \quad \ldots \quad a_n = c(t_n).$$

- Supone resolver el sistema lineal $BX = A \quad (X = B^{-1}A)$,

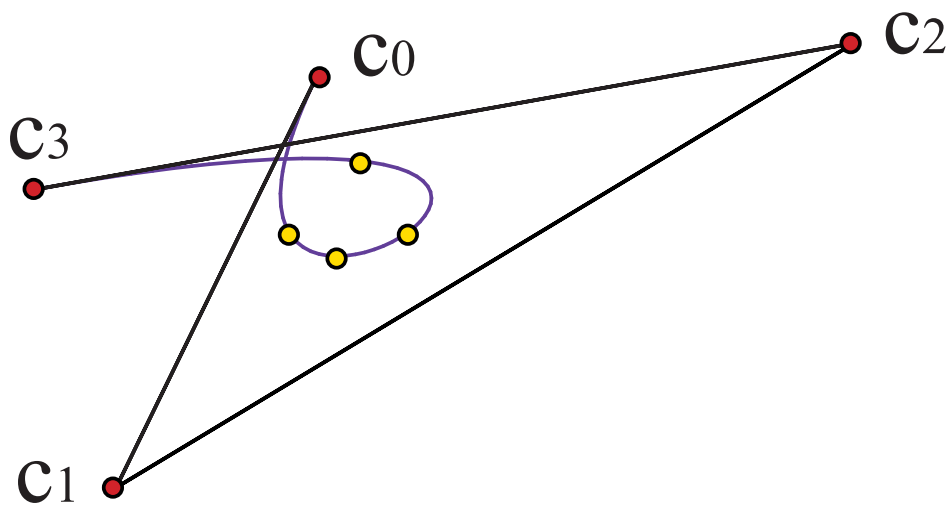
$$X = \begin{pmatrix} c_0 \\ \vdots \\ c_n \end{pmatrix}, \quad B = \begin{pmatrix} B_0^n(t_0) & \cdots & B_n^n(t_0) \\ \vdots & \ddots & \vdots \\ B_0^n(t_n) & \cdots & B_n^n(t_n) \end{pmatrix}, \quad A = \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix},$$

donde los puntos dato a_i e incógnita c_i están escritos como matrices fila.

Interpolación

- Aunque este curso esté motivado por el modelado geométrico, las curvas de Bézier también tienen otros usos.
- Por ejemplo, el problema de *interpolar* una curva $c(t)$ de grado n que pase por $n+1$ puntos $a_0, \ldots, a_n$ en *instantes* $t_0, \ldots, t_n$,

$$a_0 = c(t_0), \quad \ldots \quad a_n = c(t_n).$$



Aproximación

- Obviamente, no es necesario usar curvas de Bézier para estos problemas, pero es eficiente.
- También está el problema de *aproximar* una curva $c(t)$ de grado n que pase por $m+1$ puntos $a_0, \ldots, a_m$ en *instantes* $t_0, \ldots, t_m$, $m > n$.

$$a_0 = c(t_0), \quad \ldots \quad a_m = c(t_m).$$

- El sistema $BX = A$ de $m+1$ ecuaciones con $n+1$ incógnitas no tiene solución.
- Por eso se *aproxima* por una solución (mínimos cuadrados) del sistema $(B^t)BX = (B^t)A$ de $n+1$ ecuaciones con $n+1$.

Aproximación

- Obviamente, no es necesario usar curvas de Bézier para estos problemas, pero es eficiente.
- También está el problema de *aproximar* una curva $c(t)$ de grado n que pase por $m+1$ puntos $a_0, \ldots, a_m$ en *instantes* $t_0, \ldots, t_m$, $m > n$.

$$a_0 = c(t_0), \quad \ldots \quad a_m = c(t_m).$$