

Aproximación de puntos vía curvas de Bézier  
Usos en el diseño y la optimización

TRABAJO FIN DE GRADO



FACULTAD DE CIENCIAS MATEMÁTICAS

GRADO EN INGENIERÍA MATEMÁTICA

Departamento de Álgebra, Geometría y Topología

**TUTORA:** Marina Logares Jiménez

**ESTUDIANTE:** Javier Sempere Hernández

### **Agradecimientos:**

A mi tutora Marina, por ayudarme y guiarme durante todo el curso para poder realizar este trabajo, brindándome consejos, apoyo e inspiración.

A mi familia, por cuidarme y quererme de forma incondicional. Sin ellos no estaría aquí.

A Laura, Miguel y Dani por apoyarme y acompañarme estos años de universidad.

A Miguel, por hacerme mejor persona y aguantarme. Ha sufrido este trabajo más que yo.

A Misifú, José Luis, Benito, Penélope y Petite. Por ser mi apoyo emocional durante la carrera.

### **RESUMEN:**

Las curvas de Bézier nacieron como herramienta al diseño en la industria automovilística, sin embargo hoy en día se están usando e investigando en áreas completamente diferentes. En este trabajo se introducirá las curvas de Bézier con un breve contexto histórico y se dividirá en dos partes principales. En la primera parte se estudiará cómo se puede generalizar las curvas mediante parámetros de forma y un parámetro fraccional, consiguiendo una herramienta gráfica muy versátil para la programación creativa. En la segunda parte se estudiará dos algoritmos que usan las curvas de Bézier para aproximar puntos sobre superficies. El objetivo, será explorar diferentes áreas de investigación en las que se usan estas curvas, que se introducen por primera vez en el grado en la asignatura de Geometría Diferencial y Aplicaciones.

### **ABSTRACT:**

Bézier curves originated as a design tool in the automotive industry, but nowadays they are being used and researched in completely different areas. This paper will introduce Bézier curves with a brief historical context and will be divided into two main parts. The first part will study how curves can be generalized using shape parameters and a fractional parameter, achieving a highly versatile graphical tool for creative programming. The second part will study two algorithms that use Bézier curves to approximate points on surfaces. The objective will be to explore different research areas where these curves are used, which are introduced for the first time in the degree course on Differential Geometry and Applications.

# Índice

<b>1</b>	<b>Introducción</b>	<b>4</b>
1.1	Curvas de Bézier . . . . .	4
1.2	Motivación y objetivos . . . . .	4
<b>2</b>	<b>Curvas de Bézier fraccionarias generalizadas</b>	<b>5</b>
2.1	Parámetros de forma . . . . .	5
2.1.1	Definición y Propiedades . . . . .	5
2.1.2	Continuidad paramétrica, $C^r$ . . . . .	8
2.1.3	Continuidad geométrica, $G^r$ . . . . .	11
2.2	Curvas fraccionarias generalizadas . . . . .	12
2.2.1	Definición y Propiedades . . . . .	12
2.2.2	Continuidad fraccionaria, $F^r$ . . . . .	14
2.3	Ejemplos en el diseño . . . . .	16
<b>3</b>	<b>Aproximación de puntos en superficies</b>	<b>18</b>
3.1	Conceptos básicos . . . . .	18
3.1.1	Parametrización y Plano Tangente . . . . .	18
3.1.2	Primera Forma Fundamental . . . . .	18
3.1.3	Mapa de Gauss y Segunda Forma Fundamental . . . . .	19
3.1.4	Símbolos de Christoffel . . . . .	19
3.1.5	Derivada Covariante . . . . .	20
3.1.6	Geodésicas y Exponencial de Riemann . . . . .	21
3.2	Aproximación de puntos en superficies . . . . .	22
3.2.1	Curvas de Bézier sobre superficies . . . . .	22
3.2.2	Algoritmo de interpolación . . . . .	23
3.2.3	Algoritmo de aproximación de puntos . . . . .	25
<b>4</b>	<b>Conclusiones</b>	<b>32</b>
<b>5</b>	<b>Apéndices</b>	<b>33</b>
5.1	Apéndice A . . . . .	33
5.2	Apéndice B . . . . .	34
5.3	Apéndice C . . . . .	35
<b>6</b>	<b>Referencias</b>	<b>38</b>

# 1 Introducción

## 1.1 Curvas de Bézier

A mediados del siglo pasado la industria automovilística buscaba no solo coches prácticos y de calidad, sino que además fueran atractivos visualmente para los ciudadanos de a pie. Por aquel entonces, se usaban plantillas para diferentes curvas y los diseñadores debían señalar qué plantillas habían usado para cada recorrido de la curva. Si se debía hacer alguna modificación, había que reproducir de nuevo el diseño. Esto resultaba en un trabajo tedioso y lento, y pronto se vio que las técnicas matemáticas podían implementarse informáticamente para mejorar el rendimiento del diseño. Un ingeniero francés llamado Pierre E. Bézier, que trabaja en una famosa compañía dedicada a la automoción, se planteó el problema del diseño asistido por ordenador y desarrolló la base matemática para dibujar mediante ordenadores las curvas que hoy llevan su nombre y son utilizadas en la mayoría de programas de diseño.

Las propiedades que tienen estas curvas las hacen muy útiles para todo tipo de tareas relacionadas con el diseño y la modelización de curvas y superficies. En este trabajo, nos centraremos en estudiar nuevas líneas de investigación que usan las curvas de Bézier como base aplicadas en áreas muy ajenas a su aplicación original, el diseño de automóviles.

## 1.2 Motivación y objetivos

La modelización de curvas y superficies es fundamental en la industria, la ciencia y el diseño. Construir formas flexibles y fácilmente manipulables satisface una necesidad en estos campos. La primera parte de este trabajo consistirá en el estudio de las curvas de Bézier generalizadas, que nos servirá para poder definir un marco teórico en el que poder construir una infinidad de formas que se ajusten a las necesidades del diseñador mediante una serie de parámetros que nos permitirán modificar la curva de Bézier tradicional. Se estudiará los efectos geométricos que estos parámetros tienen y las limitaciones de estas nuevas curvas en la aplicación práctica.

Primero se introducirán los **parámetros de forma**, que nos permitirán deformar la curva original sin necesidad de modificar los puntos de control y manteniendo la continuidad impuesta. Esta característica puede resultar de mucha utilidad cuando a partir de un diseño original se quiere hacer pequeñas variaciones sin necesidad de volver a construir las curvas/superficies de nuevo.

Después se definirá el llamado **parámetro fraccional**, que nos permitirá modificar la longitud de la curva y definir un nuevo tipo de continuidad: la **continuidad fraccional**. Esto permitirá obtener mayor flexibilidad para modelizar formas que mediante la definición clásica de curvas de Bézier se hubiese necesitado construir nuevas curvas o modificar completamente las ya existentes, siendo esto más ineficiente que cambiar unos pocos parámetros sin necesidad de preocuparnos por perder la continuidad con la que se trabajaba.

Por último, en los apéndices se explica brevemente algunos conceptos auxiliares como los splines suavizados o el algoritmo del descenso de gradiente, junto con código en Julia utilizado para obtener las gráficas.

## 2 Curvas de Bézier fraccionarias generalizadas

### 2.1 Parámetros de forma

#### 2.1.1 Definición y Propiedades

**Definición 1.** Sea la curva de Bézier definida por los  $n + 1$  puntos de control  $\mathbf{P}_i \in \mathbb{R}^m$ :

$$\alpha(t) = \sum_{i=0}^n B_{i,n}(t) \mathbf{P}_i$$

donde

$$B_{i,n}(t) = \binom{n}{i} (1-t)^{n-i} t^i, \quad t \in [0, 1]$$

Nos referiremos a esta curva como la curva de Bézier original.

Definimos la **Base de funciones Bernstein de grado  $n$  con  $n$  parámetros de forma** como:

$$\begin{aligned} \hat{B}_{i,n}(t) &= B_{i,n}(t) \left(1 + \frac{a_i}{n-i+1} (1-t) - \frac{a_{i+1}}{i+1} t\right), \quad t \in [0, 1] \\ a_0 = a_{n+1} &= 0 \quad -(n-i+1) < a_i < i, \quad i = 0, 1, \dots, n \end{aligned}$$

donde  $a_1, \dots, a_n$  son los parámetros de forma.

**Proposición 1.** La base de funciones  $\hat{B}_{i,n}(t)$  tiene las siguientes propiedades:

1.  $\hat{B}_{i,n}(t) \geq 0$ ,  $t \in [0, 1]$
2.  $\hat{B}_{i,n}(t) = \hat{B}_{n-i,n}(1-t)$ , cuando  $a_i = -a_{n-i+1}$
3.  $\hat{B}_{i,n}(t) = B_{i,n}(t)$  cuando  $a_i = 0$ ,  $i = 1, \dots, n$ .
4.  $\sum_{i=0}^n \hat{B}_{i,n}(t) = 1$

*Demostración.* Para 1,2 y 3, es inmediato usando la definición y sabiendo que  $B_{i,n}(t) \geq 0$  ya que  $t \in [0, 1]$  (ver [22]). Veamos 4:

Como  $\sum_{i=0}^n B_{i,n}(t) = 1$  puesto que

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

Entonces,

$$\begin{aligned} \sum_{i=0}^n \hat{B}_{i,n}(t) &= \sum_{i=0}^n B_{i,n}(t) + \sum_{i=0}^n \frac{a_i}{n-i+1} (1-t) - \sum_{i=0}^n \frac{a_{i+1}}{i+1} t = \\ &= 1 + \sum_{i=1}^n \frac{a_i}{n-i+1} (1-t) B_{i,n}(t) - \sum_{i=0}^{n-1} \frac{a_{i+1}}{i+1} t B_{i,n}(t) = \\ &= 1 + \sum_{i=1}^n \left( \frac{a_i}{n-i+1} (1-t) B_{i,n}(t) - \frac{a_i}{i} t B_{i-1,n}(t) \right) = \\ &= 1 + \sum_{i=1}^n a_i \left( \frac{1}{n-i+1} (1-t) B_{i,n}(t) - \frac{1}{i} t B_{i-1,n}(t) \right) \stackrel{(*)}{=} 1 \end{aligned}$$

En (\*) se ha usado:

$$\begin{aligned}
& \frac{1}{n-i+1}(1-t)B_{i,n}(t) - \frac{1}{i}tB_{i-1,n}(t) = \\
& = \frac{1}{n-i+1}(1-t)\binom{n}{i}t^i(1-t)^{n-i} - \frac{1}{i}t\frac{n}{i-1}t^{i-1}(1-t)^{n-i+1} = \\
& = \frac{1}{n-i+1}\binom{n}{i}t^i(1-t)^{n-i+1} - \frac{1}{i}\frac{n}{i-1}t^i(1-t)^{n-i+1} = \\
& = t^i(1-t)^{n-i+1}\left[\frac{1}{n-i+1}\frac{n!}{(n-i)!i!} - \frac{1}{i}\frac{n!}{(n-i+1)!(i-1)!}\right] = \\
& = t^i(1-t)^{n-i+1}\left[\frac{n!}{(n-i+1)!i!} - \frac{n!}{(n-i+1)!i!}\right] = 0
\end{aligned}$$

□

Veamos los cambios geométricos que ocurren en la base de Bernstein al variar los parámetros de forma:

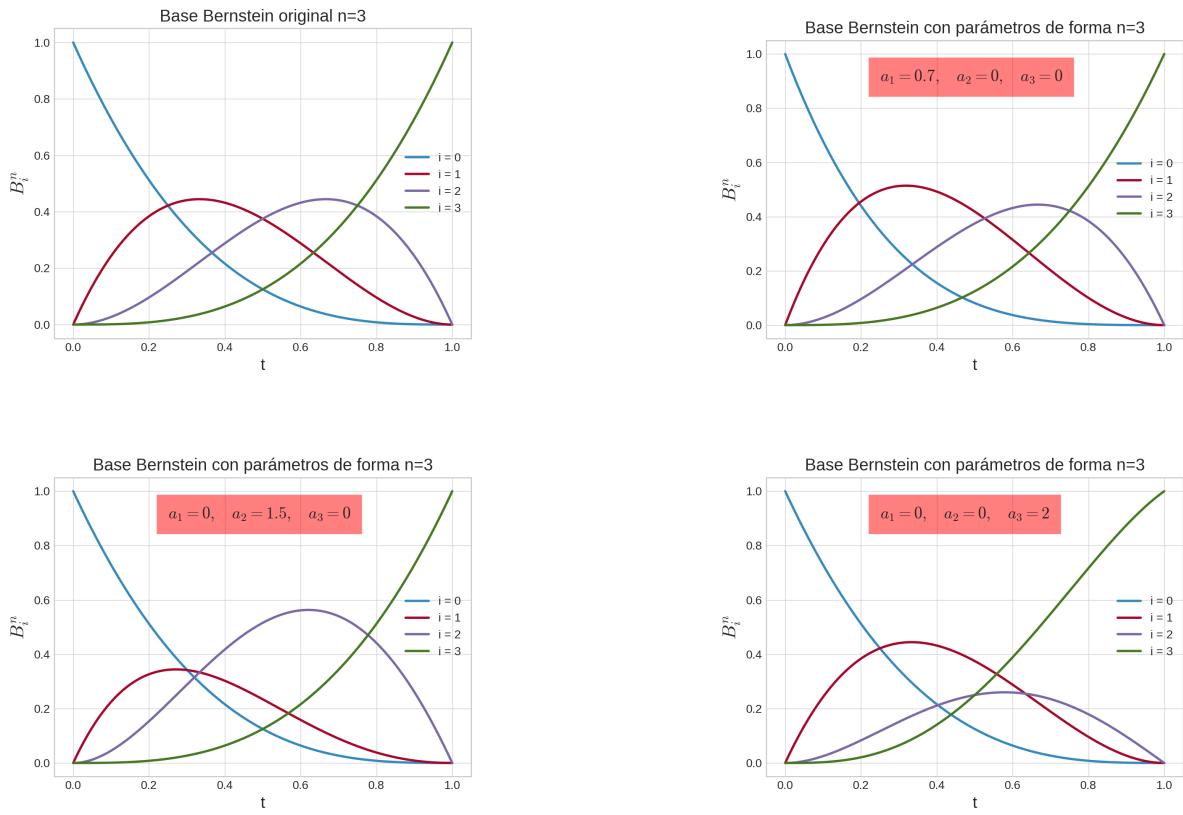


Figure 1: Cambios en la base de funciones Bernstein al variar los parámetros de forma para  $n = 3$

Observamos que perdemos la propiedad simétrica propia de las curvas de Bézier, donde si los puntos de control se determinaban en orden inverso la curva resultante tenía la misma forma, salvo que se recorría en sentido inverso.

**Definición 2.** Definimos la **Curva de Bézier de grado  $n$  con  $n$  parámetros de forma** como

$$\hat{\alpha}(t) = \sum_{i=0}^n \hat{B}_{i,n}(t)P_i, \quad t \in [0, 1] \quad (1)$$

Notemos que la cantidad de parámetros de forma nos lo determina el grado de la curva. Por las propiedades (1) que la base de funciones  $\hat{B}_{i,n}(t)$  hereda de  $B_{i,n}(t)$  se sigue que la nueva curva de Bézier cumple la propiedad del casco convexo ([22]). Es decir, si unimos dos puntos cualesquiera de la curva,

el segmento estará dentro del polígono de control.

Veamos el efecto que tiene en las curvas al variar los valores de los parámetros de forma. El desarrollo de una curva de grado  $n$  con  $n$  parámetros de forma, con  $\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n \in \mathbb{R}^m$  puntos de control es:

$$\begin{aligned}\hat{\alpha}(t) = & B_{0,n}(t) (1 + 0 - a_1 t) \mathbf{P}_0 + B_{1,n}(t) \left( 1 + a_1(1-t) - \frac{a_2}{2} t \right) \mathbf{P}_1 + \\ & + B_{2,n}(t) \left( 1 + \frac{a_2}{n-1}(1-t) - \frac{a_3}{3} t \right) \mathbf{P}_2 + \dots + \\ & + B_{i-1,n}(t) \left( 1 + \frac{a_{i-1}}{n-i+2}(1-t) - \frac{a_i}{i} t \right) \mathbf{P}_{i-1} + \\ & + B_{i,n}(t) \left( 1 + \frac{a_i}{n-i+1}(1-t) - \frac{a_{i+1}}{i+1} t \right) \mathbf{P}_i + \dots + \\ & + B_{n,n}(t) (1 + a_n(1-t) + 0) \mathbf{P}_n\end{aligned}$$

El término de la expresión sobre la que actúa el parámetro  $a_i$  es:

$$\begin{aligned}a_i \left( -\frac{B_{i-1,n}(t)}{i} t \mathbf{P}_{i-1} + \frac{B_{i,n}(t)}{n-i+1} (1-t) \mathbf{P}_i \right) = \\ = a_i \left( -\frac{\binom{n}{i-1} (1-t)^{n-i+1} t^{i-1}}{i} t \mathbf{P}_{i-1} + \frac{\binom{n}{i} (1-t)^{n-i} t^i}{n-i+1} (1-t) \mathbf{P}_i \right) = \\ a_i \left( \frac{n!}{(n-i+1)! i!} (1-t)^{n-i+1} t^i \right) (\mathbf{P}_i - \mathbf{P}_{i-1})\end{aligned}$$

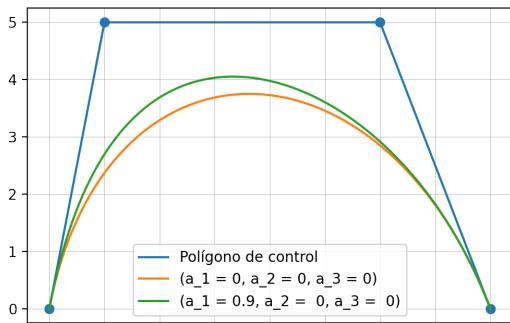
Supongamos que  $a_j = 0, \forall j \in \{0, 1, \dots, n\}, j \neq i$ , y cambiamos el valor del parámetro  $a_i$  de  $a_i^1$  a  $a_i^2$ , denotamos  $\Delta a_i = a_i^1 - a_i^2$ . La curva  $\hat{\alpha}(t)$  cambiará de  $\hat{\alpha}^1(t)$  a  $\hat{\alpha}^2(t)$ , igualmente denotamos  $\Delta \hat{\alpha}(t) = \hat{\alpha}^1(t) - \hat{\alpha}^2(t)$ . Se tiene que:

$$\Delta \hat{\alpha}(t) = \Delta a_i \frac{n!}{(n-i+1)! i!} (1-t)^{n-i+1} t^i (\mathbf{P}_i - \mathbf{P}_{i-1})$$

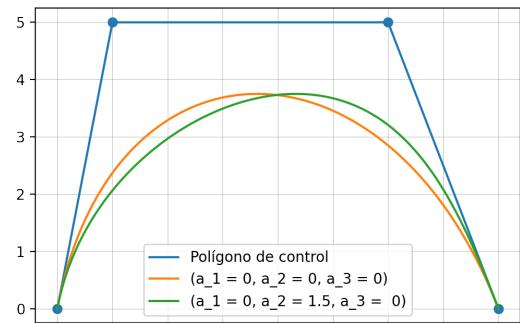
Esto tiene la siguiente interpretación geométrica: Fijando  $t_0 \in [0, 1]$ , el punto  $\hat{\alpha}(t_0) \in \mathbb{R}^m$  se desplaza en la dirección  $\overrightarrow{\mathbf{P}_{i-1} \mathbf{P}_i}$  con una magnitud de

$$\|\Delta \hat{\alpha}(t_0)\| = |\Delta a_i| \frac{n!}{(n-i+1)! i!} (1-t_0)^{n-i+1} t_0^i \|\overrightarrow{\mathbf{P}_{i-1} \mathbf{P}_i}\|$$

Veámoslo gráficamente:



(a) Dirección  $\overrightarrow{\mathbf{P}_0 \mathbf{P}_1}$



(b) Dirección  $\overrightarrow{\mathbf{P}_1 \mathbf{P}_2}$

Figure 2: Ejemplo con puntos de control:  $\mathbf{P}_0 = (0, 0), \mathbf{P}_1 = (1, 5), \mathbf{P}_2 = (6, 5), \mathbf{P}_3 = (11, 0)$

Veamos ahora cómo conectar dos curvas de Bézier con parámetros de forma.

### 2.1.2 Continuidad paramétrica, $C^r$

**Definición 3.** Sean dos curvas de Bézier  $\hat{\alpha}(t, a_1, \dots, a_n)$  y  $\hat{\beta}(t, b_1, \dots, b_m)$ ,  $t \in [0, 1]$  con  $n$  y  $m$  parámetros de forma respectivamente, ambas curvas de grado  $r+1$ , ( $n, m > r$ ). Diremos que  $\hat{\alpha}$  y  $\hat{\beta}$  son  $C^r$ -continuas si se cumple:

$$\hat{\alpha}^{(k)}(1, a_1, \dots, a_n) = \hat{\beta}^{(k)}(0, b_1, \dots, b_m), \forall k \in \{0, 1, \dots, r\}$$

donde  ${}^k$  denota la derivada  $k$ -ésima respecto la variable  $t$ .

Estudiemos más en detalle este tipo de continuidad para  $r < 3$ .

$$\hat{\alpha}(t, a_1, \dots, a_n) = \sum_{i=0}^n \hat{B}_{i,n}(t) \mathbf{P}_i = \sum_{i=0}^n B_{i,n}(t) \left[ 1 + \frac{a_i}{n-i+1}(1-t) - \frac{a_{i+1}}{i+1}t \right] \mathbf{P}_i$$

$$\hat{\beta}(t, b_1, \dots, b_m) = \sum_{j=0}^m \hat{B}_{j,m}(t) \mathbf{Q}_j = \sum_{j=0}^m B_{j,m}(t) \left[ 1 + \frac{b_j}{m-j+1}(1-t) - \frac{b_{j+1}}{j+1}t \right] \mathbf{Q}_j$$

#### Condiciones $C^0$ :

Queremos "unir" las dos curvas, es decir, que el final de la primera curva sea el principio de la segunda.

Observamos que:  $B_{i,n}(1) = \delta_{i,n}$  y  $B_{i,n}(0) = \delta_{i,0}$ , siendo  $\delta_{i,j}$  la Delta de Kronecker y recordemos que  $a_0 = a_{n+1} = b_0 = b_{m+1} = 0$  Se tiene que:

$$\hat{\alpha}(1, a_1, \dots, a_n) = \sum_{i=0}^n \delta_{i,n} \left[ 1 + \frac{a_i}{n-i+1}(1-1) - \frac{a_{i+1}}{i+1} \right] \mathbf{P}_i = \left[ 1 + 0 - \frac{a_{n+1}}{i+1} \right] \mathbf{P}_n = \mathbf{P}_n$$

$$\hat{\beta}(0, b_1, \dots, b_m) = \sum_{j=0}^m \delta_{j,0} \left[ 1 + \frac{b_j}{m-j+1}(1-0) - \frac{b_{j+1}}{j+1}0 \right] \mathbf{Q}_j = \left[ 1 + \frac{b_0}{m-0+1} - 0 \right] \mathbf{Q}_0 = \mathbf{Q}_0$$

Por tanto, la condición para la continuidad  $C^0$  es:

$$\mathbf{P}_n = \mathbf{Q}_0 \tag{2}$$

#### Condiciones $C^1$ :

Deben cumplirse las Condiciones  $C^0$ :  $\mathbf{P}_n = \mathbf{Q}_0$

Además debe cumplirse:  $\hat{\alpha}^{(1)}(1, a_1, \dots, a_n) = \hat{\beta}^{(1)}(0, b_1, \dots, b_m)$

$$\hat{\alpha}^{(1)}(t, a_1, \dots, a_n) = \sum_{i=0}^n \left[ \frac{d}{dt} (B_{i,n}(t)) \left[ 1 + \frac{a_i}{n-i+1}(1-t) - \frac{a_{i+1}}{i+1}t \right] - B_{i,n}(t) \left[ \frac{a_i}{n-i+1} + \frac{a_{i+1}}{i+1} \right] \right] \mathbf{P}_i$$

$$\hat{\beta}^{(1)}(t, b_1, \dots, b_m) = \sum_{j=0}^m \left[ \frac{d}{dt} (B_{j,m}(t)) \left[ 1 + \frac{b_j}{m-j+1}(1-t) - \frac{b_{j+1}}{j+1}t \right] - B_{j,m}(t) \left[ \frac{b_j}{m-j+1} + \frac{b_{j+1}}{j+1} \right] \right] \mathbf{Q}_j$$

Teniendo en cuenta que  $\frac{d}{dt} B_{i,n}(t) = n(B_{i-1,n-1}(t) - B_{i,n-1}(t))$ , se tiene que:

$$\frac{d}{dt} B_{i,n}(1) = n(\delta_{i-1,n-1} - \delta_{i,n-1}) = n(\delta_{i,n} - \delta_{i,n-1})$$

$$\frac{d}{dt} B_{i,n}(0) = n(\delta_{i-1,0} - \delta_{i,0})$$

Por tanto:

$$\begin{aligned}\hat{\alpha}^{(1)}(1, a_1, \dots, a_n) &= \\ &= \sum_{i=0}^n \left[ n(\delta_{i,n} - \delta_{i,n-1}) \left[ 1 + \frac{a_i}{n-i+1}(1-1) - \frac{a_{i+1}}{i+1}1 \right] - \delta_{i,n} \left[ \frac{a_i}{n-i+1} + \frac{a_{i+1}}{i+1} \right] \right] \mathbf{P}_i = \\ &= -n \left[ 1 - \frac{a_n}{n} \right] \mathbf{P}_{n-1} + [n-a_n] \mathbf{P}_n = \\ &= [a_n - n] \mathbf{P}_{n-1} + [n-a_n] \mathbf{P}_n\end{aligned}$$

$$\begin{aligned}\hat{\beta}^{(1)}(0, b_1, \dots, b_m) &= \\ &= \sum_{j=0}^m \left[ n(\delta_{j-1,0} - \delta_{j,0}) \left[ 1 + \frac{b_j}{m-j+1}(1-0) - \frac{b_{j+1}}{j+1}0 \right] - \delta_{j,0} \left[ \frac{b_j}{m-j+1} + \frac{b_{j+1}}{j+1} \right] \right] \mathbf{Q}_j = \\ &= \left[ -m \left( 1 + \frac{b_0}{m-j+1} \right) - (0+b_1) \right] \mathbf{Q}_0 + \left[ m \left( 1 + \frac{b_1}{n} \right) \right] \mathbf{Q}_1 = \\ &= -[m+b_1] \mathbf{Q}_0 + [m+b_1] \mathbf{Q}_1\end{aligned}$$

Entonces, debe cumplirse:

$$\mathbf{Q}_1 = \frac{[a_n - n] \mathbf{P}_{n-1} + [n-a_n] \mathbf{P}_n}{m+b_1} + \mathbf{Q}_0 \quad (3)$$

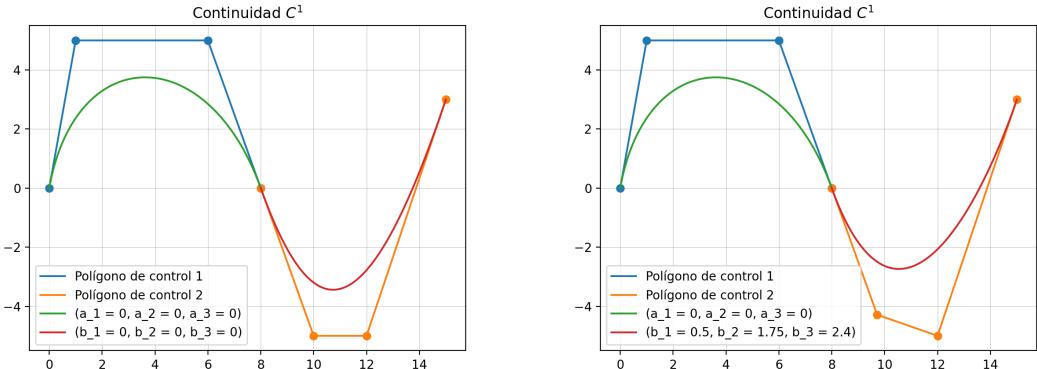


Figure 3: Ejemplo de continuidad  $C^1$ . Observamos que los dos primeros puntos de control de la segunda curva (rojo) están condicionados por los puntos de control de la primera curva (verde) y los parámetros de forma.

### Condiciones $C^2$ :

Deben cumplirse las condiciones de  $C^1$ :

$$\begin{aligned}\mathbf{Q}_0 &= \mathbf{P}_n \\ \mathbf{Q}_1 &= \frac{[a_n - n] \mathbf{P}_{n-1} + [n-a_n] \mathbf{P}_n}{m+b_1} + \mathbf{Q}_0\end{aligned}$$

Y además debe cumplirse:  $\hat{\alpha}^{(2)}(1, a_1, \dots, a_n) = \hat{\beta}^{(2)}(0, b_1, \dots, b_m)$ :

$$\begin{aligned}\hat{\alpha}^{(2)}(t, a_1, \dots, a_n) &= \\ &= \sum_{i=0}^n \left[ \frac{d^2}{dt^2} (B_{i,n}(t)) \left[ 1 + \frac{a_i}{n-i+1}(1-t) - \frac{a_{i+1}}{i+1}t \right] - 2 \frac{d}{dt} B_{i,n}(t) \left[ \frac{a_i}{n-i+1} + \frac{a_{i+1}}{i+1} \right] \right] \mathbf{P}_i\end{aligned}$$

$$\hat{\beta}^{(2)}(t, b_1, \dots, b_m) = \\ = \sum_{j=0}^m \left[ \frac{d^2}{dt^2} (B_{j,m}(t)) \left[ 1 + \frac{b_j}{m-j+1} (1-t) - \frac{b_{j+1}}{j+1} t \right] - 2 \frac{d}{dt} B_{j,m}(t) \left[ \frac{b_j}{m-j+1} + \frac{b_{j+1}}{j+1} \right] \right] \mathbf{Q}_j$$

Observamos que:

$$\frac{d^2}{dt^2} (B_{i,n}(t)) = n \left[ \frac{d}{dt} B_{i-1,n-1}(t) - \frac{d}{dt} B_{i,n-1}(t) \right] = n(n-1) [B_{i-2,n-2}(t) - 2B_{i-1,n-2}(t) + B_{i,n-2}(t)]$$

Por tanto:

$$\frac{d^2}{dt^2} B_{i,n}(1) = n(n-1) [\delta_{i,n} - 2\delta_{i,n-1} + \delta_{i,n-2}] \\ \frac{d^2}{dt^2} B_{i,n}(0) = n(n-1) [\delta_{i-2,0} - 2\delta_{i-1,0} + \delta_{i,0}]$$

$$\hat{\alpha}^{(2)}(1, a_1, \dots, a_n) = \\ = \sum_{i=0}^n \left[ n(n-1) [\delta_{i,n} - 2\delta_{i,n-1} + \delta_{i,n-2}] \left[ 1 - \frac{a_{i+1}}{i+1} \right] - 2n [\delta_{i,n} - \delta_{i,n-1}] \left[ \frac{a_i}{n-i+1} + \frac{a_{i+1}}{i+1} \right] \right] \mathbf{P}_i = \\ = [n(n-1 - a_{n-1})] \mathbf{P}_{n-2} + [-2(n-1)(n-a_n) + na_{n-1} + 2a_n] \mathbf{P}_{n-1} + [n(n-1) - 2na_n] \mathbf{P}_n$$

$$\hat{\beta}^{(2)}(0, b_1, \dots, b_m) = \\ = \sum_{j=0}^m \left[ m(m-1) [\delta_{j-2,0} - 2\delta_{j-1,0} + \delta_{j,0}] \left[ 1 + \frac{b_j}{m-j+1} \right] - 2m [\delta_{j-1,0} - \delta_{j,0}] \left[ \frac{b_j}{m-j+1} + \frac{b_{j+1}}{j+1} \right] \right] \mathbf{Q}_j = \\ = [m(m-1) + 2mb_1] \mathbf{Q}_0 + [-2(m-1)(m+b_1) - 2b_1 - mb_2] \mathbf{Q}_1 + [m(m-1 + b_2)] \mathbf{Q}_2$$

Entonces, debe cumplirse:

$$\mathbf{Q}_2 = \frac{1}{m(m-1+b_2)} [n(n-1 - a_{n-1}) \mathbf{P}_{n-2} + [(-2n+2)(n-a_n) + a_{n-1} + 2a_n] \mathbf{P}_{n-1} + \\ + [n(n-1) - 2na_n] \mathbf{P}_n - [m(m-1) + 2mb_1] \mathbf{Q}_0 - [-2(m-1)(m+b_1) - 2b_1 - mb_2] \mathbf{Q}_1]$$

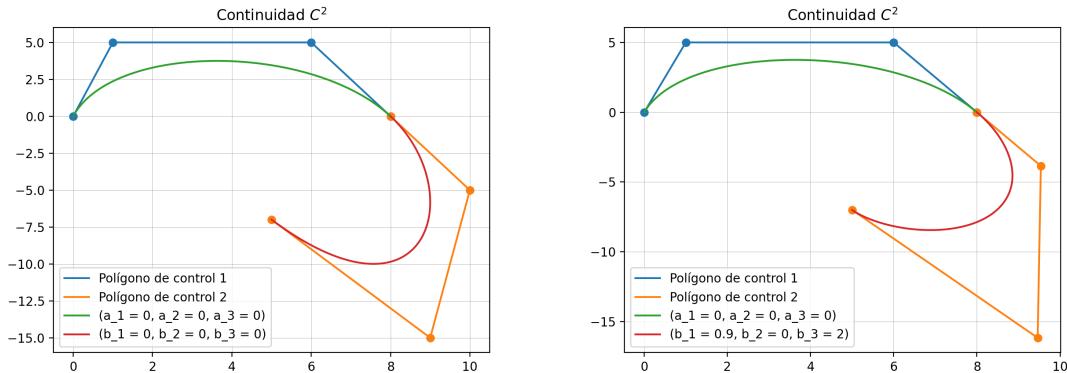


Figure 4: Ejemplo de continuidad  $C^2$ .

### 2.1.3 Continuidad geométrica, $G^r$

**Definición 4.** Sean dos curvas de Bézier  $\hat{\alpha}(t, a_1, \dots, a_n)$  y  $\hat{\beta}(t, b_1, \dots, b_m)$ ,  $t \in [0, 1]$  con  $n$  y  $m$  parámetros de forma respectivamente, ambas curvas de grado  $\geq r+1$ , ( $n, m > r$ ). Diremos que  $\hat{\alpha}$  y  $\hat{\beta}$  son  $\mathbf{G}^r$ -continuas si se cumple:

$$\begin{aligned}\hat{\alpha}(1, a_1, \dots, a_n) &= \hat{\beta}(0, b_1, \dots, b_m) \\ \hat{\alpha}^{(1)}(1, a_1, \dots, a_n) &= k_1 \hat{\beta}^{(1)}(0, b_1, \dots, b_m) \\ \hat{\alpha}^{(2)}(1, a_1, \dots, a_n) &= k_1^2 \hat{\beta}^{(2)}(0, b_1, \dots, b_m) + k_2 \hat{\beta}^{(1)}(0, b_1, \dots, b_m) \\ &\quad \dots \\ \hat{\alpha}^{(r)}(1, a_1, \dots, a_n) &= k_1^r \hat{\beta}^{(r)}(0, b_1, \dots, b_m) + k_2^{r-1} \hat{\beta}^{(r-1)}(0, b_1, \dots, b_m) + \dots + k_r \hat{\beta}^{(1)}(0, b_1, \dots, b_m) \\ k_1 > 0, k_i \in \mathbb{R}, i &= 2, \dots, r\end{aligned}$$

Estudiemos más en detalle este tipo de continuidad para  $r < 3$ . Usando el desarrollo de las condiciones para la continuidad paramétrica  $C^r$ , se sigue que:

**Condiciones  $\mathbf{G}^0$ :**

$$\mathbf{Q}_0 = \mathbf{P}_n$$

**Condiciones  $\mathbf{G}^1$ :**

$$\begin{aligned}\mathbf{Q}_0 &= \mathbf{P}_n \\ \mathbf{Q}_1 &= \frac{[a_n - n] \mathbf{P}_{n-1} + [n - a_n] \mathbf{P}_n}{k_1(m + b_1)} + \mathbf{Q}_0\end{aligned}$$

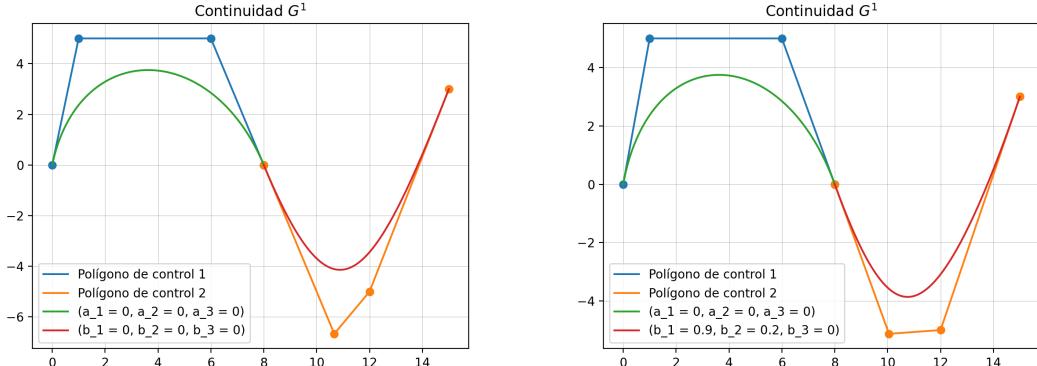


Figure 5: Ejemplo de continuidad  $G^1$  con  $k_1 = 0.75$

**Condiciones  $\mathbf{G}^2$ :**

$$\begin{aligned}\mathbf{Q}_0 &= \mathbf{P}_n \\ \mathbf{Q}_1 &= \frac{[a_n - n] \mathbf{P}_{n-1} + [n - a_n] \mathbf{P}_n}{k_1(m + b_1)} + \mathbf{Q}_0 \\ \mathbf{Q}_2 &= \frac{1}{k_1^2 m (m - 1 + b_2)} [n(n - 1 - a_{n-1}) \mathbf{P}_{n-2} + [(-2n + 2)(n - a_n) + a_{n-1} + 2a_n] \mathbf{P}_{n-1} + \\ &+ [n(n - 1) - 2na_n] \mathbf{P}_n - k_1^2 [m(m - 1) + 2mb_1] \mathbf{Q}_0 - k_1^2 [-2(m - 1)(m + b_1) - 2b_1 - mb_2] \mathbf{Q}_1 - \\ &- k_2(m + b_1)(\mathbf{Q}_1 - \mathbf{Q}_0)]\end{aligned}$$

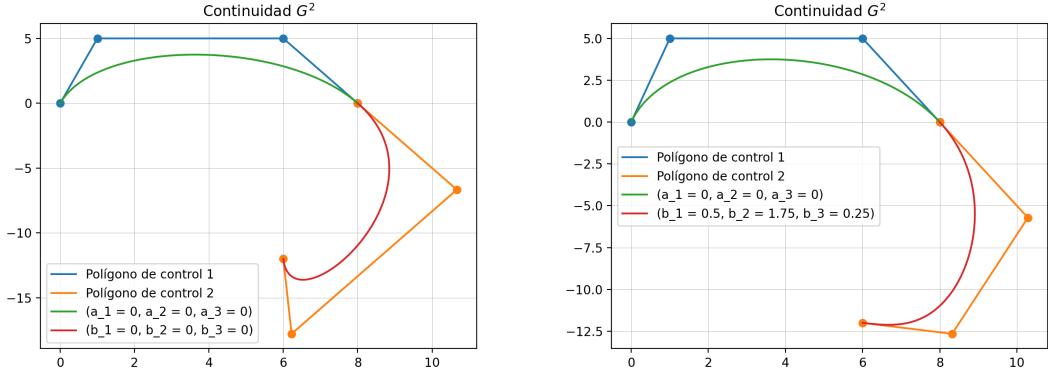


Figure 6: Ejemplo de continuidad  $G^2$  con  $k_1 = 0.75$  y  $k_2 = 0.75$

La implementación para garantizar la continuidad  $C^r$  y  $G^r$  es muy sencilla, ya que fijada la primera curva, solo se deben determinar los puntos de control necesarios de la segunda mediante unas fórmulas explícitas.

## 2.2 Curvas fraccionarias generalizadas

### 2.2.1 Definición y Propiedades

Como hemos visto, la continuidad paramétrica y geométrica tiene una limitación: permiten que las curvas estén conectadas en los puntos final e inicial. Sin embargo, podríamos querer conectar dos curvas en cualquier punto de la primera curva. En este caso tendríamos que usar un método de subdivisión para juntar la primera curva y conectar el punto final de la nueva curva con el inicial de la segunda. Vamos a solventar este problema introduciendo un nuevo parámetro, llamado **parámetro fraccional**, que denotaremos por  $v$ . Con él, podremos definir un nuevo tipo de continuidad.

**Definición 5.** Sea  $t \in [0, 1]$  y  $v \geq 0$ , definimos la **curva de Bézier fraccionaria generalizada de grado  $n$  con  $n$  parámetros** como:

$$f(t, v, a_1, \dots, a_n) = \sum_{i=0}^n \bar{B}_{i,n}(t, v, a_1, \dots, a_n) \mathbf{P}_i$$

con  $\mathbf{P}_i, i = 0, 1, \dots, n$  puntos de control,  $a_0 = a_{n+1} = 0$  donde:

$$\bar{B}_{i,n}(t, v, a_1, \dots, a_n) = B_{i,n}(t) \left( 1 + \frac{a_i}{n-i+1} (1 - \psi(v, t)) - \frac{a_{i+1}}{i+1} \psi(v, t) \right)$$

con

$$B_{i,n}(t) = \binom{n}{i} (1 - \psi(v, t))^{n-i} \psi(v, t)^i$$

$$\begin{aligned} \psi: \mathbb{R}^+ \times [0, 1] &\rightarrow [0, 1] \\ (v, t) &\mapsto \psi(v, t) \end{aligned}$$

En general, por [22], [18], [17] usaremos como definición de  $\psi$ :

$$\psi(v, t) = \frac{t^{v+1}}{\Gamma(v+2)}$$

Pero veremos que no es la única definición que podemos usar para conseguir los resultados que buscamos.

También, vamos a observar el comportamiento de  $\psi$  y el efecto geométrico en las curvas resultantes para diferentes valores de  $v$ .

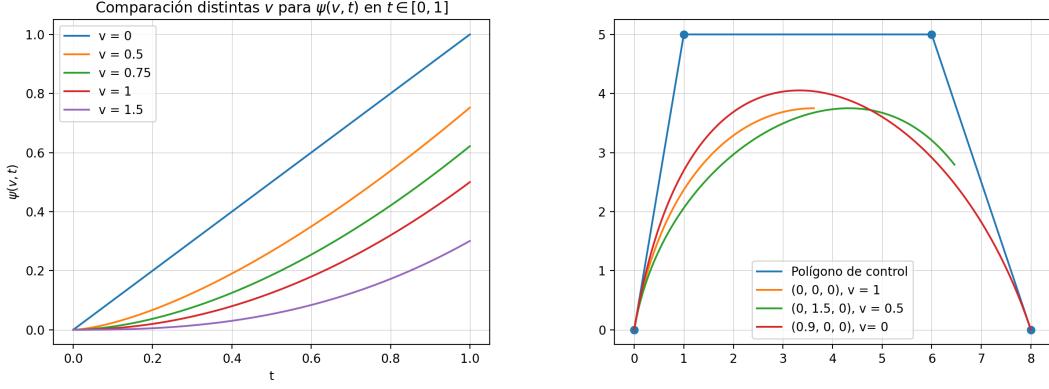


Figure 7: Efecto al variar  $v$  en la función  $\psi$  y el efecto geométrico que distintos valores del parámetro fraccional tiene en las curvas de Bézier

Como podemos ver, para  $v = 0$  se tiene que  $\phi(0, t) = t$  y obtenemos las curvas de Bézier con parámetros de forma que ya hemos definido en la sección anterior. Para  $v > 0$ , las curvas resultantes van cambiando en longitud puesto que  $\phi(v, 1) < 1$ .

Este efecto se consigue de forma similar con una función  $f(v, t)$  que cumpla:

- $f(v, t)$  convexa en  $t \in [0, 1], \forall v \geq 0$
- $f(v, 1) < 1, \forall v \geq 0$ , por esto, el parámetro 't' de la curva de Bézier orginal es menor que 1, por tanto se acorta la curva.
- $f(0, t) = t, \forall t \in [0, 1]$ , devuelve a la base de Bernstein original para  $v = 0$ .

Un ejemplo trivial sería  $f(v, t) = \frac{1}{v+1}t^{v+1}$ . Sin embargo, para el resto del trabajo continuaremos con la definición de  $\psi(v, t)$  anterior.

La curva de Bézier fraccionaria generalizada hereda todas las propiedades de la curva de Bézier clásica y tiene nuevas propiedades como el ajuste de la forma fijando los puntos de control y el acortamiento en longitud.

Veamos los cambios geométricos que ocurren en la base de Bernstein al variar el parámetro fraccional:

**Proposición 2.** *La Base de funciones de Bernstein generalizada cumple las siguientes propiedades:*

1. Para  $v = 0$  y  $a_i = 0, i = 1, \dots, n$ , obtenemos la base de funciones de Bernstein clásica.
2.  $\bar{B}_{i,n}(t) \geq 0, i = 0, \dots, n$
3.  $\sum_{i=0}^n \bar{B}_{i,n}(t) = 1$
4.  $\bar{B}_{i,n}(t) = \bar{B}_{n-i,n}(1-t), i = 0, \dots, n$  para  $a_i = -a_{n-i+1}$  y  $v = 0$

*Demostración.* La demostración completa se puede ver en [22], se basa en la proposición 1. Veamos una intuición:

Para 1, se usa directamente la definición para  $v = a_1, \dots, a_n = 0$ , obteniéndose  $\bar{B}_{i,n}(t) = B_{i,n}(t)$ .

Para 2, se tiene que para  $v \geq 0$  y  $t \in [0, 1]$ ,  $0 \leq \psi(v, t) \leq 1 \Rightarrow -1 \leq -\psi(v, t) \leq 0 \Rightarrow 0 \leq 1 - \psi(v, t) \leq 1$ . Como  $B_{i,n}(t) \geq 0$ , por definición se tiene que  $\bar{B}_{i,n}(t) \geq 0$ .

Para 3, se usa una demostración similar a la utilizada en la proposición 1.

Para 4, partiendo de las hipótesis  $a_i = -a_{n-i+1}$  y  $v = 0$ , obtenemos la base de Bernstein clásica obteniendo propiedad de simetría propia de la base de funciones  $B_{i,n}(t)$ .  $\square$

Una consecuencia de 2 y 3 en la proposición anterior es que la curva de Bézier fraccionaria generalizada con puntos de control  $P_i, i = 0, 1, \dots, n$  queda dentro del polígono de control, es decir, cumple la propiedad de casco convexo. [22]

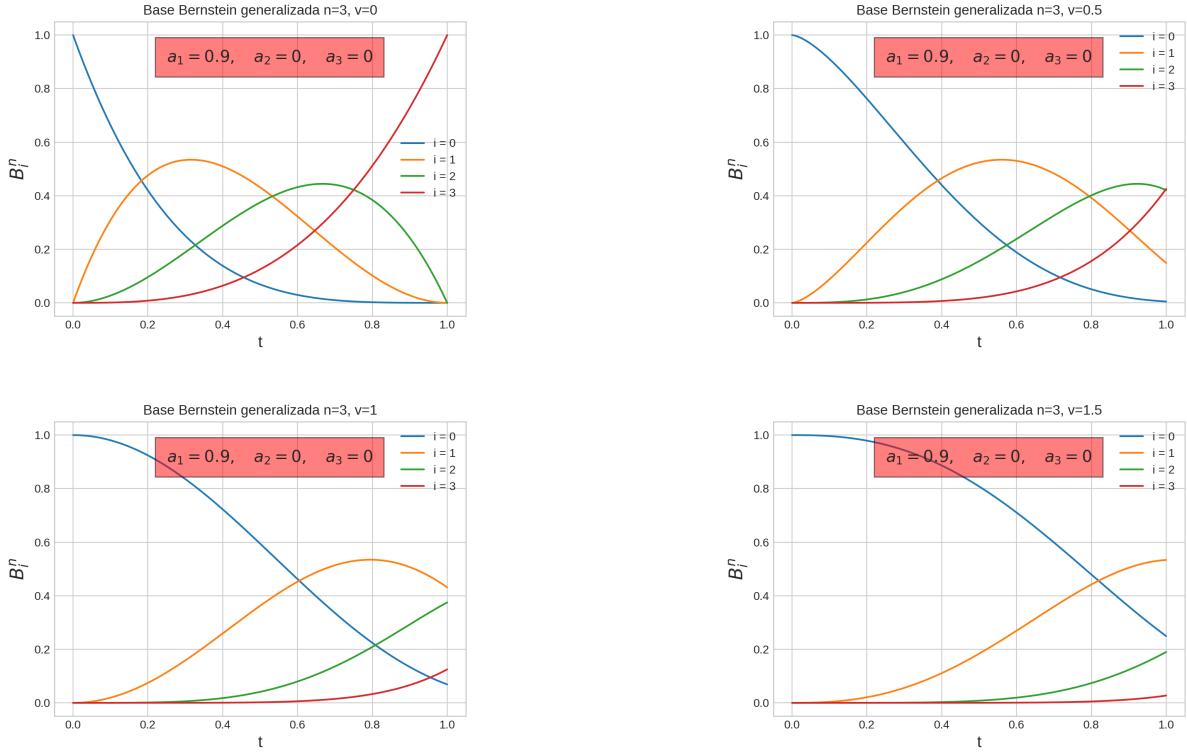


Figure 8: Base de funciones de Bernstein para unos parámetros de forma y diferentes valores del parámetro fraccional  $v$

### 2.2.2 Continuidad fraccionaria, $F^r$

A diferencia de la continuidad parámetrica y geométrica, la continuidad fraccionaria nos permitirá dar criterios para conectar la segunda curva en cualquier punto de la primera curva.

**Definición 6.** Sean

$$f(t, v_f, a_1, \dots, a_n) = \sum_{i=0}^n \bar{B}_{i,n}(t, v_f, a_1, \dots, a_n) \mathbf{P}_i, t \in [0, 1], n \geq 3$$

$$g(t, v_g, b_1, \dots, b_m) = \sum_{j=0}^m \bar{B}_{j,m}(t, v_g, b_1, \dots, b_m) \mathbf{Q}_j, t \in [0, 1], m \geq 3$$

Donde  $\bar{B}_{i,n}$  y  $\bar{B}_{j,m}$  son la base de funciones de Bézier generalizadas fraccionarias de grado  $n$  y  $m$ ,  $a_1, \dots, a_n$  y  $b_1, \dots, b_m$  los parámetros de forma y  $P_i, i = 0, \dots, n$ ,  $Q_j, j = 0, \dots, m$  los puntos de control de  $f$  y  $g$  respectivamente. Las dos curvas son  $F^r$  continuas si se cumple:

$$\begin{cases} f(1, v_f) = g(0, v_g) \\ f'(1, v_f) = \phi_1 g'(0, v_g) \\ f''(1, v_f) = \phi_1^2 g''(0, v_g) + \phi_2' g'(0, v_g) \\ \dots \\ f^{(r)} = \phi_1^r g^{(r)}(0, v_g) + \phi_2^{r-1} g^{(r-1)}(0, v_g) + \dots + \phi_r g'(0, v_g) \end{cases} \quad (4)$$

Donde el factor  $\phi_1 > 0$  y  $\phi_i \in \mathbb{R}$  para  $i = 2, \dots, r$

Veamos ejemplos de  $F^r$  para  $r < 3$ . Supondremos  $v_g = 0$  para simplificar el problema.

**Continuidad  $F^0$ :** Debe mantenerse la continuidad en cualquier punto de la curva  $f(t; v_f, a_1, \dots, a_n)$ . Solo involucra por tanto el primer punto de control de  $g$ , que siempre se interpola. Por ello, la condición impuesta es:

$$Q_0 = \sum_{i=0}^n \bar{B}_{i,n}(1; v_f, a_1, \dots, a_n) \mathbf{P}_i$$

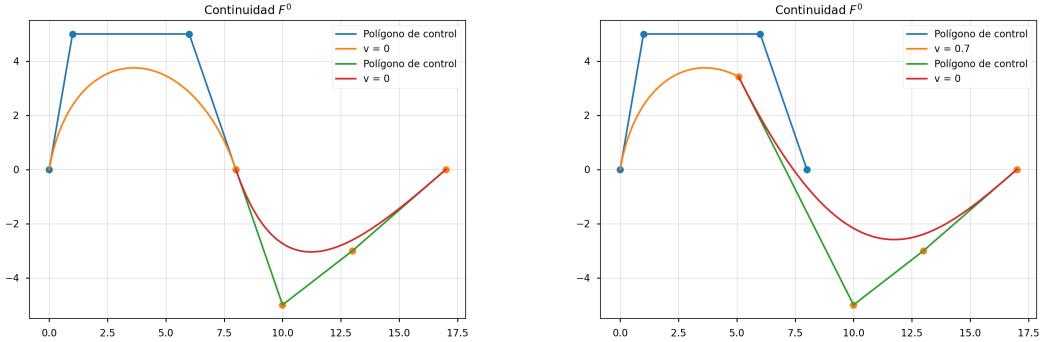


Figure 9: Continuidad  $F^0$  para los mismos parámetros de forma variando el parámetro  $v_f$

**Continuidad  $F^1$ :** Debe cumplir  $F^0$  y además ser tangente a cualquier punto de la primera curva salvo por el factor de escala  $\phi$ . Es decir, hay que resolver  $f(1, v_f) = g(0, v_g)$  y  $f'(1, v_f) = \phi g'(0, v_g)$ ,  $\phi > 0$ . Los puntos de control de  $g$  que quedan determinados por la continuidad son:

$$Q_0 = \sum_{i=0}^n \bar{B}_{i,n}(1; v_f, a_1, \dots, a_n) \mathbf{P}_i$$

$$Q_1 = \frac{1}{\phi(n+b_1)} \left[ \frac{d}{dt} \left( \sum_{i=0}^n \bar{B}_{i,n}(t, v_f, a_1, \dots, a_n) P_i \right) \Big|_{t=1} \right] + Q_0$$

Donde  $b_1$  es el primer parámetro de forma de la curva  $g$ .

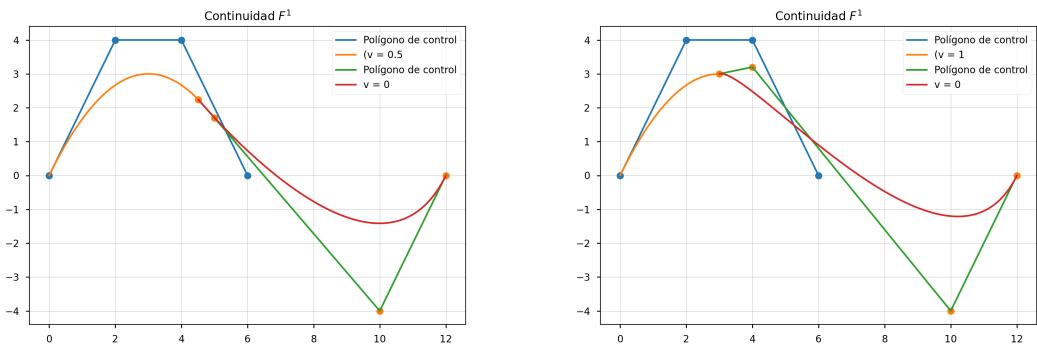


Figure 10: Continuidad  $F^1$  para los mismos parámetros de forma variando el parámetro  $v_f$

**Continuidad  $F^1$ :** Debe cumplir  $F^0$  y  $F^1$  y además debe tener la misma aceleración en cualquier punto de la primera curva salvo por los factores de escala  $\phi_1$  y  $\phi_2$ . Es decir, hay que resolver  $f(1, v_f) = g(0, v_g)$  y

$f'(1, v_f) = \phi_1 g'(0, v_g)$ ,  $\phi_1 > 0$  y también debe cumplir  $f''(1, v_f) = \phi_1^2 g''(0, v_g) + \phi_2 g'(0, v_g)$  con  $\phi_1 > 0$  y  $\phi_2 \in \mathbb{R}$ . Los puntos de control de  $g$  que quedan determinados por la continuidad son:

$$Q_0 = \sum_{i=0}^n \bar{B}_{i,n}(1; v_f, a_1, \dots, a_n) \mathbf{P}_i$$

$$Q_1 = \frac{1}{\phi(n+b_1)} \left[ \frac{d}{dt} \left( \sum_{i=0}^n \bar{B}_{i,n}(t, v_f, a_1, \dots, a_n) P_i \right) \Big|_{t=1} \right] + Q_0$$

$$Q_2 = \frac{1}{n+b_2-1} \left[ \frac{1}{\phi_1^2 n} \left( \frac{d^2}{dt^2} \left( \sum_{i=0}^n \bar{B}_{i,n}(t, v_f, a_1, \dots, a_n) P_i \right) \Big|_{t=1} \right. \right. + \right.$$

$$\left. \left. \left. + \phi_2(n+b_1)(Q_0 - Q_1) \right) - (n+2b_1-1)Q_0 + (2n+2b_1+b_2-2)Q_1 \right]$$

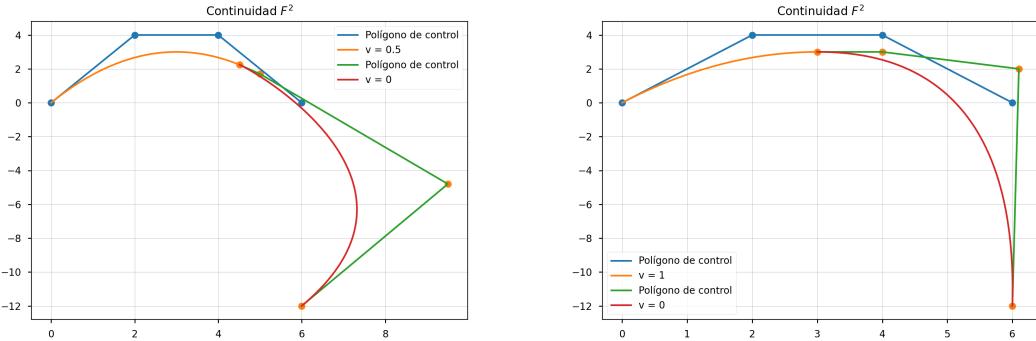


Figure 11: **Continuidad  $F^2$**  para los mismos parámetros de forma variando el parámetro  $v_f$

Donde  $b_1$  y  $b_2$  son los dos primeros parámetros de forma de  $g$ .

En [22] se puede consultar las fórmulas explícitas para dos curvas cúbicas.

**Nota:** Aunque para órdenes superiores se puede perder la intuición geométrica. La diferencia entre los tres tipos de continuidades se puede pensar como: la continuidad parámetrica garantiza la continuidad en el "punto de unión" de las curvas, es decir, en la intersección entre las curvas puede ser derivable pero puede ocurrir que visualmente las curvas tengan una unión "abrupta". Esto se solventa con la continuidad geométrica, que garantiza una unión más "suave". Para  $r = 1$  es muy intuitivo ya que coincide con la paramétrica excepto por un factor de escala. La continuidad fraccionaria es similar a la geométrica salvo que añade el parámetro fraccional para variar la longitud de las curvas.

### 2.3 Ejemplos en el diseño

Una aplicación de todo lo visto hasta ahora podría ser para diseñar gráficos y animaciones por ordenador. Actualmente ya existen muchas herramientas de diseño que permiten modelizar curvas y realizar animaciones sin dificultad y sin necesidad de conocer los fundamentos matemáticos que hay detrás. Una desventaja evidente de las curvas de Bézier fraccionarias generalizadas es que para poder variar la curva, hay que cambiar una serie de parámetros. Esto significa que se puede perder la intuición geométrica de la curva resultante, haciendo que estas curvas sean poco prácticas en el diseño gráfico asistido por programas gráficos. Sin embargo, puede encontrar su nicho en la programación creativa [23], donde las matemáticas y el código no son un fin en si mismo, sino un medio de expresión personal.

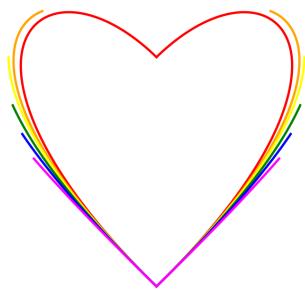


Figure 12: Ejemplo de diseño usando las curvas de Bézier fraccionarias generalizadas modificando los parámetros de forma y el parámetro fraccional.

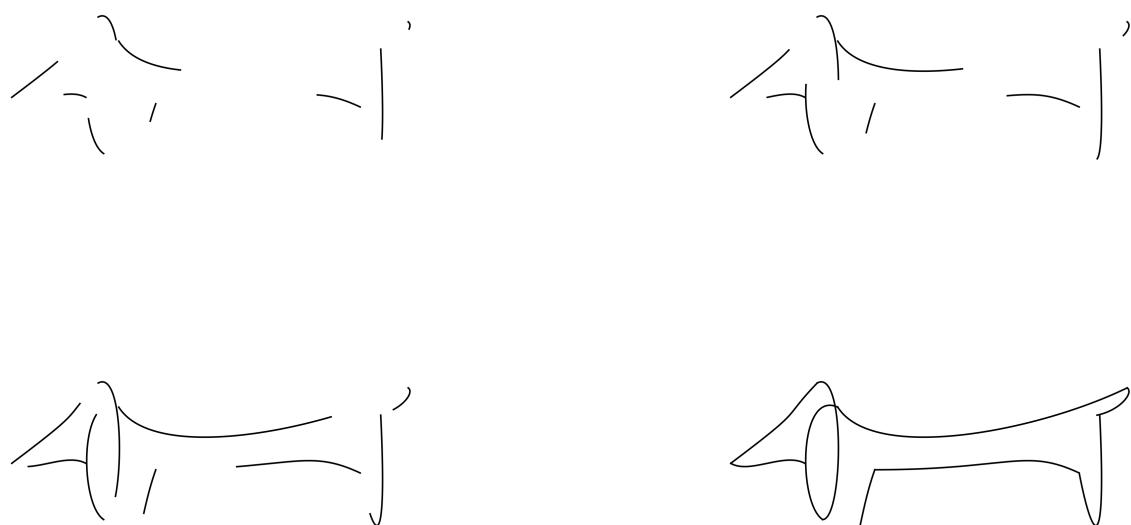


Figure 13: Diseño de Lump usando curvas de Bézier fraccionarias variando únicamente el parámetro fraccional. Basado en *Le Chien, Pablo Picasso*

### 3 Aproximación de puntos en superficies

Hemos visto una posible extensión de las curvas de Bézier al campo del diseño y la ingeniería, donde hemos sido capaces de generalizar la base de funciones de Bernstein para obtener variaciones en tamaño y forma de las curvas de Bézier originales.

Ahora vamos a usar las curvas de Bézier en un contexto muy diferente. Veamos cómo se han usado las curvas de Bézier en contextos ajenos al diseño (su aplicación original) y se están usando en el campo de la optimización para aproximar puntos. Nuestro objetivo será poder ajustar varias curvas de Bézier cumpliendo la continuidad  $C^1$  aproximando puntos (ligados a instantes temporales) sobre una superficie. La aproximación de puntos es un problema que recibe mucha investigación, por ejemplo, en anatomía computacional ([3]), algoritmos de generación de trayectorias para vehículos autónomos ([9]) o en diseño y gráficos por ordenador ([19]). Una introducción a la optimización sobre superficies se puede consultar en [8]. Existen numerosos trabajos que desarrollan algoritmos que consiguen este objetivo, por ejemplo: [20], [1], [12]. Primero, se repasarán algunos conceptos básicos vistos en la asignatura de Geometría Diferencial y Aplicaciones para poder fijar notación. Después, se presentará el problema formalmente y se introducirán algunos algoritmos desarrollados en [12] y [6], que nos permitirán obtener una aproximación en superficies regulares usando la librería Manopt.jl [5].

#### 3.1 Conceptos básicos

##### 3.1.1 Parametrización y Plano Tangente

Comencemos introduciendo algunos conceptos básicos de Geometría Diferencial que usaremos más adelante. Para ello, nos basaremos principalmente en [10].

**Definición 7.** Un subconjunto  $S \subset \mathbb{R}^3$  es una **superficie regular** si  $\forall p \in S, \exists V \subset \mathbb{R}^3, p \in V$  y existe  $\psi: U \rightarrow V \cap S, U = \dot{U} \subset \mathbb{R}^2$  tal que:

- $\psi$  es diferenciable
- $\psi$  es un homeomorfismo. Por tanto,  $\exists \psi^{-1}: V \cap S \rightarrow U$  continua.
- $\forall q \in U$ , la diferencial  $d\psi_q: \mathbb{R}^2 \rightarrow \mathbb{R}^3$  es biyectiva.

A  $\psi$  se le llama **parametrización**.

Sea  $\psi: U \subset \mathbb{R}^2 \rightarrow S$  una parametrización de una superficie regular  $S$  y sea  $q \in U$ . El subespacio vectorial de dimensión 2  $d\psi_q \subset \mathbb{R}^3$  coincide con el conjunto de vectores tangentes a  $S$  en  $\psi(q)$ . El plano  $d\psi_q$  que contiene a  $p = \psi(q)$  se denomina **plano tangente a  $S$  en  $p$**  y se denota por  $T_p S$ .  $\psi$  determina una base de  $T_p S$ :  $\{\frac{\partial \psi}{\partial u}(q), \frac{\partial \psi}{\partial v}(q)\}$ . Por notación,  $\frac{\partial \psi}{\partial u} = \psi_u$  y  $\frac{\partial \psi}{\partial v} = \psi_v$ .

##### 3.1.2 Primera Forma Fundamental

El producto escalar natural de  $\mathbb{R}^3$  induce en cada plano tangente  $T_p S \subset \mathbb{R}^3$  un producto escalar  $\langle \cdot, \cdot \rangle_p$ , que es una forma bilineal simétrica, al cual le corresponde una forma cuadrática  $I_p: T_p S \rightarrow \mathbb{R}$ ,  $I_p(w) = \langle w, w \rangle_p = |w|^2$ .

**Definición 8.**  $I_p$  se denomina la **primera forma fundamental de  $S$  en  $p$** . Podemos expresar  $I_p$  en la base  $\{\psi_u, \psi_v\}$  de  $T_p S$  asociada a una parametrización  $\psi(u, v)$  en  $p \in S$ :

$$\begin{cases} E = \langle \psi_u, \psi_u \rangle_p \\ F = \langle \psi_u, \psi_v \rangle_p \\ G = \langle \psi_v, \psi_v \rangle_p \end{cases} \quad (5)$$

La matriz asociada a  $I$  es:  $\begin{pmatrix} E & F \\ F & G \end{pmatrix}$

Sea  $p \in S$  superficie regular, y sea  $\psi: U \subset \mathbb{R}^2 \rightarrow S, \phi: V \subset \mathbb{R}^2 \rightarrow S$  dos parametrizaciones de  $S$  tales que  $p \in \psi(U) \cap \phi(V) = W$ . Entonces  $h = \psi^{-1} \circ \phi$  es un difeomorfismo. Es decir,  $h$  y  $h^{-1}$  son diferenciables.

### 3.1.3 Mapa de Gauss y Segunda Forma Fundamental

**Definición 9.** Dado  $p \in S$ , podemos elegir un vector normal unitario en cada punto de  $\psi(U)$  mediante la función diferenciable  $N: \psi(U) \rightarrow \mathbb{R}^3$

$$N(p) = \frac{\psi_u \times \psi_v}{|\psi_u \times \psi_v|}(p), \quad p \in \psi(U) \subset S$$

En general, si  $V$  es un conjunto abierto contenido en  $S$  y  $N: V \rightarrow \mathbb{R}^3$  es una función diferenciable que asocia a cada  $p \in V$  un vector normal unitario en  $p$ , decimos que  $N$  es un **campo diferenciable de vectores normales unitarios** sobre  $V$ .

Decimos que una superficie regular es **orientable** si admite un campo diferenciable de vectores normales unitarios definido sobre toda la superficie. La elección de ese campo  $N$  se llama **orientación** de  $S$ .

Sea  $S \subset \mathbb{R}^3$  una superficie con una orientación  $N$ . La función diferenciable  $N: S \rightarrow \mathbb{S}^2$  se llama **Mapa de Gauss**. Aquí,  $\mathbb{S}^2$  es la esfera unitaria.

Se tiene que  $dN_p: T_p(S) \rightarrow T_{N(p)}(\mathbb{S}^2)$  es una aplicación lineal ya que  $T_p(S)$  y  $T_{N(p)}(\mathbb{S}^2)$  son el mismo espacio vectorial. Además, se trata de una aplicación lineal autoadjunta ([10], pg 142), por tanto, tiene asociada una forma cuadrática  $Q$  en  $T_p S$  dada por  $Q(v) = \langle dN_p(v), v \rangle, v \in T_p S$ .

**Definición 10.** La forma cuadrática  $II_p$  definida en  $T_p S$  mediante  $II_p(v) = -\langle dN_p(v), v \rangle$  se llama **segunda forma fundamental de  $S$  en  $p$** . En la base  $\{\psi_u, \psi_v\}$ ,  $dN$  tiene asociada la matriz  $(a_{i,j})$ ,  $i, j = 1, 2$  donde

$$\begin{cases} a_{11} = \frac{fF - eG}{EG - F^2} \\ a_{12} = \frac{gF - fG}{EG - F^2} \\ a_{21} = \frac{eF - fE}{EG - F^2} \\ a_{22} = \frac{fF - gE}{EG - F^2} \end{cases} \quad (6)$$

Aquí,  $E, F, G$  son los calculados en (5) y

$$\begin{cases} e = \langle N, \psi_{uu} \rangle \\ f = \langle N, \psi_{vu} \rangle \\ g = \langle N, \psi_{vv} \rangle \end{cases}$$

con  $N = \frac{\psi_u \times \psi_v}{|\psi_u \times \psi_v|}$ . A (6) se les llama **ecuaciones de Weingarten**.

### 3.1.4 Símbolos de Christoffel

Sea  $S$  una superficie regular, orientable y orientada. Sea  $\psi: U \subset \mathbb{R}^2 \rightarrow S$  una parametrización en la orientación de  $S$ . Es posible asignar a cada punto de  $\psi(U)$  un triángulo dado por los vectores  $\psi_u$ ,  $\psi_v$  y  $N$ . Expresando las derivadas de los vectores  $\psi_u$  y  $\psi_v$  en la base  $\{\psi_u, \psi_v, N\}$  obtenemos:

$$\begin{cases} \psi_{uu} = \Gamma_{11}^1 \psi_u + \Gamma_{11}^2 \psi_v + eN \\ \psi_{uv} = \Gamma_{12}^1 \psi_u + \Gamma_{12}^2 \psi_v + fN \\ \psi_{vu} = \Gamma_{21}^1 \psi_u + \Gamma_{21}^2 \psi_v + fN \\ \psi_{vv} = \Gamma_{22}^1 \psi_u + \Gamma_{22}^2 \psi_v + gN \end{cases} \quad (7)$$

donde  $\Gamma_{ij}^k, i, j, k = 1, 2$  se llaman los **símbolos de Christoffel de  $S$  en la parametrización  $\psi$** . Puesto que  $\psi_{uv} = \psi_{vu}$  se tiene que  $\Gamma_{12}^1 = \Gamma_{21}^1$  y  $\Gamma_{12}^2 = \Gamma_{21}^2$ . Para calcularlos, tomamos producto escalar respecto  $\psi_u$  y  $\psi_v$  obteniendo:

$$\begin{cases} \Gamma_{11}^1 E + \Gamma_{11}^2 F = \langle \psi_{uu}, \psi_u \rangle \\ \Gamma_{11}^1 F + \Gamma_{11}^2 G = \langle \psi_{uu}, \psi_v \rangle \end{cases} \quad \begin{cases} \Gamma_{12}^1 E + \Gamma_{12}^2 F = \langle \psi_{uv}, \psi_u \rangle \\ \Gamma_{12}^1 F + \Gamma_{12}^2 G = \langle \psi_{uv}, \psi_v \rangle \end{cases} \quad \begin{cases} \Gamma_{22}^1 E + \Gamma_{22}^2 F = \langle \psi_{vv}, \psi_u \rangle \\ \Gamma_{22}^1 F + \Gamma_{22}^2 G = \langle \psi_{vv}, \psi_v \rangle \end{cases}$$

y si  $EG - F^2 \neq 0$  se tiene

$$\left( \begin{matrix} \Gamma_{11}^1 \\ \Gamma_{11}^2 \end{matrix} \right) = \frac{1}{EG - F^2} \left( \begin{matrix} \langle \psi_{uu}, \psi_u \rangle \\ \langle \psi_{uu}, \psi_v \rangle \end{matrix} \right), \quad \left( \begin{matrix} \Gamma_{12}^1 \\ \Gamma_{12}^2 \end{matrix} \right) = \frac{1}{EG - F^2} \left( \begin{matrix} \langle \psi_{uv}, \psi_u \rangle \\ \langle \psi_{uv}, \psi_v \rangle \end{matrix} \right), \quad \left( \begin{matrix} \Gamma_{22}^1 \\ \Gamma_{22}^2 \end{matrix} \right) = \frac{1}{EG - F^2} \left( \begin{matrix} \langle \psi_{vv}, \psi_u \rangle \\ \langle \psi_{vv}, \psi_v \rangle \end{matrix} \right)$$

### 3.1.5 Derivada Covariante

Un campo vectorial tangente en un abierto  $V \subset S$  de una superficie regular  $S$  es una correspondencia  $w$  que asigna a cada  $p \in V$  un vector  $w(p) \in T_p S$ . Se dice que  $w$  es diferenciable en  $p$  si dada una parametrización  $\psi(u, v)$  en  $p$ , las funciones  $a$  y  $b$  de  $w = a\psi_u + b\psi_v$  en la base  $\{\psi_u, \psi_v\}$  son diferenciables en  $p$ . Un ejemplo de un campo vectorial diferenciable a lo largo de una curva  $\alpha$  sería el campo  $\alpha'(t)$  de los vectores tangentes de  $\alpha$ .

**Definición 11.** Sea  $w$  un campo vectorial diferenciable en un conjunto abierto  $V \subset S$  y  $p \in V$ . Sea  $y \in T_p S$  y la curva parametrizada  $\alpha: (-\epsilon, \epsilon) \rightarrow V$  con  $\alpha(0) = p$  y  $\alpha'(0) = y$  y sea  $w(t), t \in (-\epsilon, \epsilon)$  la restricción del campo vectorial  $w$  a la curva  $\alpha$ . El vector que se obtiene al proyectar ortogonalmente  $\frac{dw}{dt}(0)$  en el plano  $T_p S$  se llama **derivada covariante en  $p$  del campo vectorial  $w$  relativo al vector  $y$** . Lo denotamos por  $\frac{Dw}{dt}(0)$  o bien  $D_y w(p)$

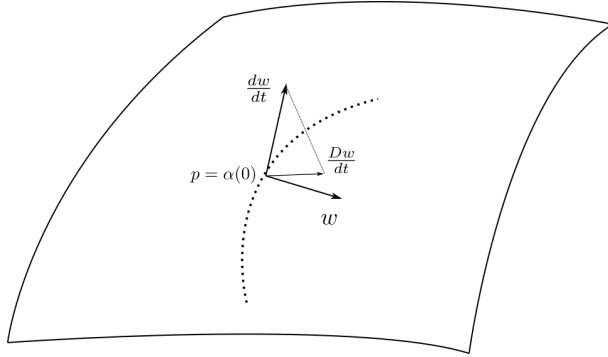


Figure 14: Derivada covariante

Vamos a ver que la derivada covariante en  $p$  no depende de la curva  $\alpha$ . De forma general,

$$w: S \rightarrow TS = \bigcup_{p \in S} T_p S$$

$$p \mapsto w_p \in T_p S$$

Consideremos la restricción de  $w$  a  $\alpha(t) = \psi(u(t), v(t))$ ,  $w(\alpha(t)) = a(t)\frac{\partial\psi(u(t), v(t))}{\partial u} + b(t)\frac{\partial\psi(u(t), v(t))}{\partial v} = a(t)\frac{\partial\psi}{\partial u} + b(t)\frac{\partial\psi}{\partial v} = w(t) \in T_{\alpha(t)} S$   
Por tanto

$$\frac{d}{dt}w(t) = a'(t)\frac{\partial\psi}{\partial u} + a(t)\left(\frac{\partial^2\psi}{\partial u^2}u'(t) + \frac{\partial^2\psi}{\partial u \partial v}v'(t)\right) + b'(t)\frac{\partial\psi}{\partial v} + b(t)\left(\frac{\partial^2\psi}{\partial v \partial u}u'(t) + \frac{\partial^2\psi}{\partial v^2}v'(t)\right)$$

Tenemos entonces

$$\begin{aligned} \frac{Dw}{dt} &= \left\langle \frac{dw(t)}{dt}, \frac{\partial\psi}{\partial u} \right\rangle \frac{\partial\psi}{\partial u} + \left\langle \frac{dw(t)}{dt}, \frac{\partial\psi}{\partial v} \right\rangle \frac{\partial\psi}{\partial v} = \\ &= a'(t)\frac{\partial\psi}{\partial u} + a(t)u'(t)\left\langle \frac{\partial^2\psi}{\partial u^2}, \frac{\partial\psi}{\partial u} \right\rangle \frac{\partial\psi}{\partial u} + a(t)v'(t)\left\langle \frac{\partial^2\psi}{\partial u \partial v}, \frac{\partial\psi}{\partial u} \right\rangle \frac{\partial\psi}{\partial u} + b(t)u'(t)\left\langle \frac{\partial^2\psi}{\partial v \partial u}, \frac{\partial\psi}{\partial u} \right\rangle \frac{\partial\psi}{\partial u} + \\ &\quad + b(t)v'(t)\left\langle \frac{\partial^2\psi}{\partial v^2}, \frac{\partial\psi}{\partial u} \right\rangle \frac{\partial\psi}{\partial u} + a(t)u'(t)\left\langle \frac{\partial^2\psi}{\partial u^2}, \frac{\partial\psi}{\partial v} \right\rangle \frac{\partial\psi}{\partial v} + a(t)v'(t)\left\langle \frac{\partial^2\psi}{\partial u \partial v}, \frac{\partial\psi}{\partial v} \right\rangle \frac{\partial\psi}{\partial v} + b'(t)\frac{\partial\psi}{\partial v} + \\ &\quad + b(t)u'(t)\left\langle \frac{\partial^2\psi}{\partial v \partial u}, \frac{\partial\psi}{\partial v} \right\rangle \frac{\partial\psi}{\partial v} + b(t)v'(t)\left\langle \frac{\partial^2\psi}{\partial v^2}, \frac{\partial\psi}{\partial v} \right\rangle \frac{\partial\psi}{\partial v} \end{aligned}$$

Y por (7) se tiene que:

$$\begin{aligned}\frac{Dw}{dt} &= \frac{\partial\psi}{\partial u}(a'(t) + a(t)u'(t)\Gamma_{11}^1 + a(t)v'(t)\Gamma_{12}^1 + b(t)u'(t)\Gamma_{12}^1 + b(t)v'(t)\Gamma_{22}^1) + \\ &\quad + \frac{\partial\psi}{\partial v}(b'(t) + a(t)u'(t)\Gamma_{11}^2 + a(t)v'(t)\Gamma_{12}^2 + b(t)u'(t)\Gamma_{12}^2 + b(t)v'(t)\Gamma_{22}^2)\end{aligned}$$

Es decir,  $\frac{Dw}{dt}$  depende solo de  $(u', v') = y$  y no de la curva  $\alpha$ . En particular, si  $S$  es un plano, podemos encontrar una parametrización tal que en (5):  $E = G = 1, F = 0$  y por tanto,  $\Gamma_{ij}^k = 0 \forall i, j, k$ . Esto es, la derivada covariante coincide con la derivada usual de vectores en el plano. (ver [10]).

### 3.1.6 Geodésicas y Exponencial de Riemann

Un campo vectorial  $w$  restringido a una curva parametrizada  $\alpha: I \rightarrow S$  se dice **paralelo** si  $\frac{Dw}{dt} = 0 \forall t \in I$ . En el caso de un plano, un campo paralelo a lo largo de una curva parametrizada se reduce a un campo constante a lo largo de la curva.

**Definición 12.** Sea  $\alpha: I \rightarrow S$  una curva parametrizada y  $w_0 \in T_{\alpha(t_0)}S$ ,  $t_0 \in I$ . Sea  $w$  el campo vectorial paralelo a lo largo de  $\alpha$  con  $w(t_0) = w_0$ . El vector  $w(t_1)$ ,  $t_1 \in I$  se llama **transporte paralelo de  $w_0$  a través de  $\alpha$  al punto  $t_1$**

**Definición 13.** Una curva parametrizada  $\gamma: I \rightarrow S$  se dice **geodésica** si el campo de sus vectores tangentes  $\gamma'(t)$  es paralelo a lo largo de  $\gamma$ , es decir,  $\frac{D\gamma'(t)}{dt} = 0$ . Un ejemplo de geodésicas son las líneas rectas en el plano o los grandes círculos en la esfera  $S^2$ . Decimos que una geodésica parametrizada uniendo dos puntos es **mínima** si su longitud es menor o igual a cualquier curva regular parametrizada a trozos uniendo estos dos puntos. Sea  $\gamma(0) = a$  y  $\gamma(1) = b$ , denotamos la geodésica mínima por  $g(t; a, b)$  y su longitud por la distancia  $d_S(a, b)$ .

**Definición 14.** Sea  $\gamma$  la geodésica que satisface  $\gamma(0) = p$  y  $\gamma'(0) = v$  y  $\gamma(1) = q \in S$ . Definimos la exponencial de Riemann como

$$\begin{aligned}exp_p: T_pS &\rightarrow S \\ v &\mapsto q = exp_p(v) = \gamma(1, v)\end{aligned}$$

Donde  $\gamma(1)$  lo hemos denotado por  $\gamma(1, v)$  para hacer explícito su dependencia con el vector  $v$  en  $\gamma'(0) = v$ .

**Nota:** Como trabajamos con superficies regulares, en particular con  $\mathbb{S}^2$ , la exponencial siempre estará bien definida y será un diferenciable. Por eso, el conjunto donde  $exp$  es inyectiva será todo  $S$ .

**Definición 15.** Sea  $S = \mathbb{S}^2$ , definimos el logaritmo de Riemann como la inversa de la exponencial:

$$\begin{aligned}log_a: S &\rightarrow T_pS \\ q &\mapsto v = log_p(q) = \gamma'(0)\end{aligned}$$

Sean  $p, q \in S = \mathbb{S}^2$ , podemos calcular la geodésica mínima entre  $p$  y  $q$  como

$$g(t; p, q) = exp_p(t log_p(q)), \quad t \in [0, 1]$$

Usaremos esto más adelante para el algoritmo De Casteljau cuando definamos las curvas de Bézier sobre  $S = \mathbb{S}^2$ .

**Definición 16.** Sea  $\gamma: [0, T] \rightarrow S$  una geodésica sobre  $S$  y sea  $h: [0, T] \times (-\epsilon, \epsilon) \rightarrow S$  una variación de  $\gamma$  tal que  $\forall s \in (-\epsilon, \epsilon)$ , la curva  $h_s(t) = h(t, s)$  para  $t \in [0, T]$  es una geodésica. Definimos el **campo de Jacobi** a lo largo de  $\gamma$  como  $\frac{\partial h}{\partial s}(t, 0) = J(t)$

### 3.2 Aproximación de puntos en superficies

El objetivo de esta sección es introducir un algoritmo que nos permita aproximar una curva de Bézier  $C^1$  a una serie de puntos sobre una superficie. Tendremos dos objetivos: minimizar la distancia de los puntos a la curva, y minimizar la curvatura de esta.

Para ello, la función a minimizar será

$$\min_{\beta \in \mathbf{B}} \int_{t_0}^{t_n} \left\| \frac{D^2 \beta(t)}{dt} \right\|_{\beta(t)}^2 dt + \lambda \sum_{i=0}^n d^2(\beta(t_i), d_i) \quad (8)$$

donde  $\mathbf{B}$  es el conjunto de curvas de Bézier admisibles sobre la superficie  $S$ ,  $\frac{D^2}{dt}$  es segunda derivada covariante y el parámetro  $\lambda$  determina el balance entre acercar la curva a los puntos y minimizar la curvatura. Si  $S = \mathbb{R}^n$ ,  $\mathbf{B}$ , por [13] (teorema 2.4), el conjunto de curvas cúbicas de Bézier engloba los splines de suavizado cúbicos (ver Apéndice 1). Por tanto, los puntos de control óptimos vendrán dados por los splines cúbicos. Además, la derivada covariante coincide con la derivada parcial usual en  $\mathbb{R}^n$ , ya que los símbolos de Christoffel se anulan [10]. Notemos que si  $\lambda \rightarrow \infty$ , sacrificamos la reducción de curvatura por la interpolación de puntos. Estudiaremos el problema general, propondremos un algoritmo de resolución y estudiaremos los resultados con un ejemplo básico, analizando la complejidad computacional del algoritmo y las dificultades que puedan presentarse.

#### 3.2.1 Curvas de Bézier sobre superficies

Recordemos que dados unos puntos de control  $b_0, \dots, b_K \in \mathbb{R}^n$   $K \in \mathbb{N}$ , la curva de Bézier  $\beta_K: [0, 1] \rightarrow \mathbb{R}^n$  de grado  $K$  se define como  $\beta(t) = \sum_{i=0}^K b_i B_{i,K}(t)$  donde  $B_{i,K}(t) = \binom{K}{i} t^i (1-t)^{K-i}$  es la base de polinomios de Bernstein usual. También, recordemos que  $\beta_K(0) = b_0$ ,  $\beta_K(1) = b_K$ ,  $\frac{d}{dt} \beta_K(0) = K(b_1 - b_0)$  y  $\frac{d}{dt} \beta_K(1) = K(b_K - b_{K-1})$ , es decir, se interpola el primer y último punto de control y la derivada en  $t = 0$  y  $t = 1$  es tangente al segmento que une los dos primeros puntos y los dos últimos respectivamente.

Para construir la curva de Bézier podemos usar el algoritmo de De Casteljau [7], que consiste en ir formando la curva a través de combinaciones convexas de los puntos de control.

---

#### Algoritmo 1 De Casteljau

---

**Requiere:**  $b_0, \dots, b_K$

▷ Puntos de control

$$g_i^0 := b_i \quad i = 0, \dots, K \\ g_j^k := (1-t)g_j^{k-1} + tg_{j+1}^{k-1} \quad k = 1, \dots, K \quad j = 0, \dots, K-k$$


---

Donde  $g_0^K = \beta_K(t)$ . Por ejemplo, para  $K = 2$  se tiene que

$$g_0^0 = b_0, \quad g_1^0 = b_1, \quad g_2^0 = b_2 \\ g_0^1 = (1-t)g_0^0 + tg_1^0, \quad g_1^1 = (1-t)g_1^0 + tg_2^0 \\ \beta_2(t) = g_0^2 = (1-t)g_0^1 + tg_1^1 \quad t \in [0, 1]$$

A la derecha podemos observar un ejemplo con puntos de control  $(0, 0)$ ,  $(1, 3)$  y  $(2, 1)$  donde se ha representado las combinaciones convexas que realiza el algoritmo para dibujar la curva de Bézier resultante. En particular, para  $t = 0.3$  (en azul),  $t = 0.55$  (en verde) y  $t = 0.9$  (en morado).

Veamos cómo exigir la continuidad a varias curvas de Bézier para  $t \in [0, N]$ .

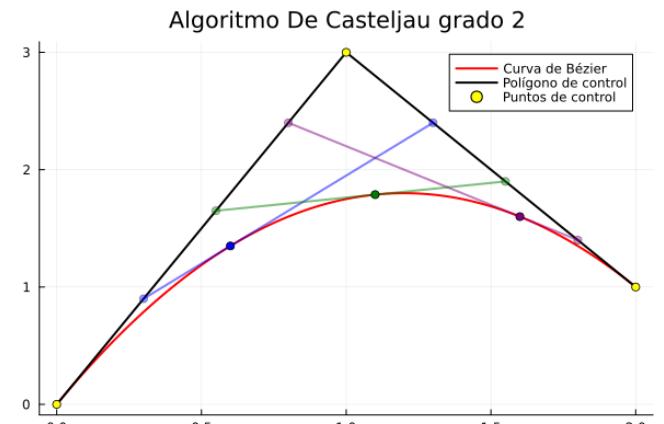


Figure 15: Ejemplo Algoritmo De Casteljau para  $K = 2$

**Definición 17.** Sea

$$\begin{aligned}\beta: [0, N] &\rightarrow \mathbb{R}^n \\ t &\mapsto \beta_{K_i}^i(t - i), \quad i = \lfloor t \rfloor\end{aligned}$$

Donde cada curva de Bézier  $\beta_{K_i}^i$  tiene puntos de control  $b_0^i, b_1^i, \dots, b_{K_i}^i$  para  $i = 0, \dots, N - 1$ . Diremos que  $\beta$  es **continua**  $C^0$  si el último punto de control de una curva  $\beta_{K_i}^i$  y el primer punto de control de  $\beta_{K_{i+1}}^{i+1}$  coinciden:  $b_{K_i}^i = b_0^{i+1} \quad i = 0, \dots, N - 2$ .

Denotemos por  $b_i^-$  el antepenúltimo punto de control de la  $(i-1)$ -ésima curva de Bézier y por  $b_i^+$  el segundo punto de control de la  $i$ -ésima curva de Bézier. Se dice que  $\beta$  es **diferenciable** ( $continua C^1$ ) en  $t = i$  si se cumple que  $p_i := b_{K_{i-1}}^{i-1} = b_{K_i}^i = \frac{K_{i-1}b_i^- + K_ib_i^+}{K_{i-1} + K_i}$ . Es decir, que  $p_i$  (primer punto de control de la  $i$ -ésima curva de Bézier, coincidente con el último punto de control de la  $(i-1)$ -ésima curva de Bézier),  $b_i^+$  y  $b_i^-$  estén alienados.

Hasta ahora, solo hemos recuperado conceptos ya conocidos (ver Continuidad  $C^0$  y  $C^1$  con parámetros de forma nulos). Sin embargo, ¿cómo podemos definir las curvas de Bézier sobre una superficie  $S$ ? Para ello, referimos a [6], [1], [20] y [12]. Consideremos los puntos de control  $b_0, \dots, b_K \in S$ . La curva de Bézier  $\beta_K(t): [0, 1] \rightarrow S$  la definimos directamente usando el algoritmo de De Casteljau modificado para superficies. Aquí, cambiamos las combinaciones convexas por geodésicas mínimas, que como hemos visto en la sección 3.1.6, podemos definir a partir de la exponencial y logaritmo Riemanniano:

---

### Algoritmo 2 De Casteljau sobre superficies

---

**Requiere:**  $b_0, \dots, b_K \in S$  ▷ Puntos de control  
 $g_i^0 := b_i \quad i = 0, \dots, K$   
 $g_j^k := \exp_{g_j^{k-1}}(t \log_{g_j^{k-1}} g_{j+1}^{k-1}) \quad k = 1, \dots, K \quad j = 0, \dots, K - k$

---

Se tiene que  $\beta_K(t) = g_K^0$ ,  $t \in [0, 1]$ . Además  $\beta_K(0) = b_0$ ,  $\beta_K(1) = b_K$ ,  $\frac{d}{dt}\beta_K(0) = K \log_{b_0} b_1$  y  $\frac{d}{dt}\beta_K(1) = -K \log_{b_K} b_{K-1}$  análogo a las propiedades vistas en  $\mathbb{R}^n$ .

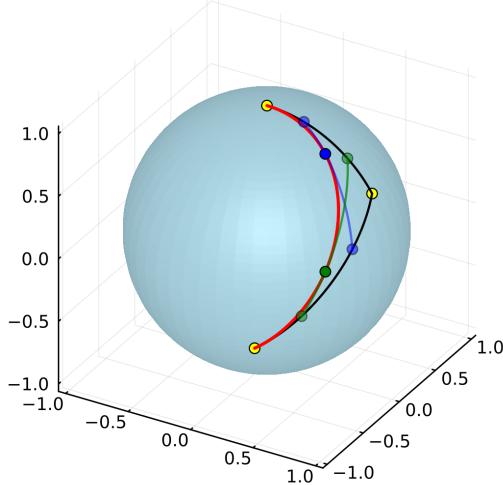


Figure 16: Ejemplo Algoritmo De Casteljau sobre  $\mathbb{S}^2$  para  $K = 2$

Una demostración de esto se puede ver en [12].

### 3.2.2 Algoritmo de interpolación

Tratemos primero el problema de interpolar los puntos, es decir, cuando  $\lambda \rightarrow \infty$ . Dados los puntos sobre la superficie  $d_0, \dots, d_n$  asociados a los tiempos  $t_0, \dots, t_n$  (por simplicidad,  $t_i = i \quad \forall i$ ), buscamos encontrar la

A la izquierda podemos ver un ejemplo del algoritmo con los puntos de control (en amarillo)  $(0, 0, 1)$ ,  $(0.84, 0, 0.54)$  y  $(0.37, -0.82, -0.41)$ . La línea roja es la curva de Bézier resultante, es decir,  $g_K^0$ . Se ha dibujado las geodésicas mínimas en diferentes instantes del algoritmo para visualizar su comportamiento, para  $t = 0.3$  (en azul) y para  $t = 0.8$  (en verde).

De forma análoga, podemos definir la curva de Bézier a trozos como

$$\begin{aligned}\beta: [0, N] &\rightarrow S \\ t &\mapsto \beta_{K_i}^i(t - i), \quad i = \lfloor t \rfloor\end{aligned}$$

y las condiciones para cumplir  $C^1$  son  $b_{K_i}^i = b_0^{i+1}, \quad i = 0, \dots, N - 2$  y  $\frac{1}{K_{i-1}} \log_{p_i} b_i^- = -\frac{1}{K_i} \log_{p_i} b_i^+ \quad i = 1, \dots, N - 1$ , donde  $p_i = g(\frac{K_{i-1}}{K_{i-1} + K_i}; b_i^-, b_i^+)$ .

curva de Bézier cúbica a trozos  $\beta \in C^1$  que minimice  $\int_{t_0}^{t_n} \left\| \frac{D^2 \beta(t)}{dt} \right\|_{\beta(t)} dt$  tal que  $\beta(t_i) = d_i \quad \forall i = 0, \dots, n$ . Ya hemos visto la interpolación de  $d_0$  y  $d_n$  están asegurada. Por condiciones de diferenciabilidad (restricciones no lineales), debemos encontrar  $b_i^+, b_{i+1}^- \in S, \quad i = 0, \dots, n-1$  tales que minimicen aceleración de la curva. Este es un problema de optimización sobre superficies que es computacionalmente muy costoso. Para simplificarlo, trabajaremos en  $\mathbb{R}^n$  para calcular los puntos de control  $b_i^+, b_{i+1}^-$ . Por [13], la curva de Bézier cúbica a trozos bajo condiciones de interpolación minimiza la aceleración cuadrática media y satisface las condiciones de diferenciabilidad. La curva será óptima en el espacio euclídeo, aunque no se puede garantizar la optimalidad en una superficie  $S$  regular cualquiera, es una aproximación computacionalmente eficiente [11].

Como ya sabemos, la derivada covariante coincide con la derivada usual en  $\mathbb{R}^n$  (ya que se anulan los símbolos de Christoffel [10]), por tanto buscamos minimizar  $\int_0^n \left\| \frac{d^2}{dt^2} \beta(t) \right\|^2$  donde  $\beta \in \{\beta_3(t-i), i = \lfloor t \rfloor\}$  con puntos de control:  $\{d_i, b_i^+, b_{i+1}^-, d_{i+1}\}$ . Por condiciones de diferenciabilidad, para el caso de curvas de Bézier de grado 3, tenemos que  $b_i^+ = 2d_i - b_i^-$ . Entonces, solo necesitamos encontrar  $b_0^+, b_i^- \in \mathbb{R}^n, \quad i = 1, \dots, n$ . Sea  $X = [b_0^+, b_1^-, \dots, b_n^-] \in \mathbb{R}^{(n+1) \times n}$  las variables,  $D = [d_0, \dots, d_n] \in \mathbb{R}^{(n+1) \times n}$  los datos del problema y  $A, C \in \mathbb{R}^{(n+1) \times (n+1)}$  matrices de coeficientes.

Calculando explícitamente la integral [12] para el caso de una curva de Bézier cúbica con puntos de control  $\{d_i, b_i^+, b_{i+1}^-, d_{i+1}\}$  y considerando las restricciones de diferenciabilidad ( $b_i^+ = 2d_i - b_i^-$ ), podemos resolver analíticamente  $n = \dim(\mathbb{R}^n)$  sistemas de ecuaciones (uno por cada componente). Denotando a  $\beta_3^i$  la  $i$ -ésima curva de Bézier de  $\beta$ , se tiene

$$\begin{aligned} \frac{\partial \beta_3^0}{\partial b_0^+} &= 6b_0^+ - 3b_1^- - 3d_0 \\ \frac{\partial \beta_3^0}{\partial b_1^-} &= -3b_0^+ + 6b_1^- - 3d_1 \end{aligned}$$

A partir de la segunda curva y hasta la penúltima, se deben tener en cuenta las condiciones de diferenciabilidad:

$$\begin{aligned} \frac{\partial \beta_3^i}{\partial b_i^-} &= 6b_i^- + 3b_{i+1}^- - 9d_i \\ \frac{\partial \beta_3^i}{\partial b_{i+1}^-} &= 3b_i^- + 6b_{i+1}^- - 6d_i - 3d_{i+1} \end{aligned}$$

Para  $i = 1, \dots, n-1$ . Entonces, separando las variables  $X = [b_0^+, b_1^-, \dots, b_n^-]$  y los datos  $D = [d_0, \dots, d_n]$  con sus respectivas matrices de coeficientes  $A$  y  $C$  (ver [12]) llegamos al sistema  $AX = CD$  con  $\det(A) \neq 0$ .

Se tiene entonces que  $X = A^{-1}CD = WD \Rightarrow x_i = \sum_{j=0}^n w_{ij} d_j \quad i = 0, \dots, n$ . Pero esto es la resolución en  $\mathbb{R}^n$ , para poder extrapolarlo a una superficie debemos generalizar las condiciones de optimalidad para poder calcular los puntos de control sobre una superficie  $S$ . Para ello, hacemos

$$x_i - d_{ref} = \sum_{j=0}^n w_{ij} (d_j - d_{ref}) \quad i = 0, \dots, n \quad \forall d_{ref}$$

Si nos fijamos, esto sería el "logaritmo riemanniano en un espacio euclídeo", en el sentido:  $\log_a b = b - a$ , es decir, una función que nos lleva  $x_i$  a  $d_{ref}$  en  $T_{d_{ref}} S$ .  $\chi_i = \log_{d_{ref}} x_i = \sum_{j=0}^n w_{ij} \log_{d_{ref}} d_j \quad i = 0, \dots, n \Rightarrow x_i = \exp_{d_{ref}} \chi_i$ .

Pero ¿cómo delegimos  $d_{ref}$ ? Deben ser tal que  $T_{d_{ref}} S$  sea una buena aproximación de  $S$  para  $x_i$ . Dado  $i \in \{1, \dots, n\}$ ,  $w_{ii} > w_{ij}$  por construcción de la matriz, para  $x_i = b_i^-$  el valor mejor aproximado es sobre el plano tangente es  $d_i$ . Es decir, para cada punto de control  $b_i^-$  aproximamos la solución del espacio euclídeo a  $S$  tomando como referencia los puntos (datos)  $d_i \in S$ . Para  $x_0 = b_0^+$  se toma  $d_{ref} = d_0$ . Como, por condiciones de diferenciabilidad,  $\frac{1}{K_{i-1}} \log_{p_i} b_i^- = -\frac{1}{K_i} \log_{p_i} b_i^+$  (y que el logaritmo es la función inversa de la exponencial), se tiene que  $b_i^+ = \exp_{d_i}(-\chi_i) \quad i = 1, \dots, n-1$ . Una vez obtenidos los puntos de control, se puede construir la curva de Bézier usando el algoritmo de De Casteljau para superficies.

---

**Algoritmo 3** Interpolación de puntos vía Curvas de Bézier  $C^1$ 


---

**Requiere:**  $d_0, \dots, d_n \in S, A, C \in \mathbb{R}^{n+1}$

$$s_0 = \dots = s_n = 0$$

$$W = A^{-1}C$$

**for**  $i=0:n$  **do**

$$d_{ref} = d_i$$

**for**  $j=0:n$  **do**

$$s_j = \log_{d_{ref}} d_j$$

**end for**

$$\bar{x} = \sum_{k=0}^n w_{ik} s_k$$

$$x = \exp_{d_{ref}} \bar{x}$$

**if**  $i \neq 0, i \neq n$  **then**

$$b_i^- = x$$

$$b_i^+ = \exp_{d_{ref}}(-\bar{x})$$

**else if**  $i = 0$  **then**

$$b_0^+ = x$$

**else**

$$b_n^- = x$$

**end if**

**end for**

$$\beta = \beta^i(t - i), \text{ con puntos de control: } d_i, b_i^+, b_{i+1}^-, d_{i+1}, \quad i = \lfloor t \rfloor, \quad t \in [0, n]$$


---

Vamos a ver un ejemplo con 4 puntos  $\{d_0, d_1, d_2, d_3\}$  sobre  $\mathbb{S}^2$ , el conjunto de curvas de Bézier se representará en rojo. Los puntos de control no se muestran ya que no tienen ningún interés en la interpolación de los puntos  $d_i$ , ya que son fijados directamente por el algoritmo.

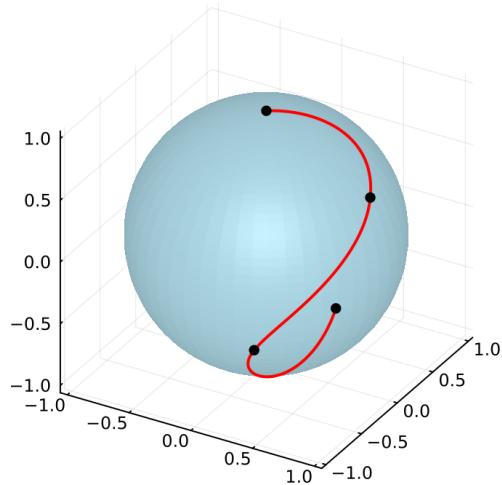


Figure 17: Interpolación de puntos usando curvas de Bézier cúbicas en  $\mathbb{S}^2$

La construcción se hace a través de curvas de Bézier cúbicas. La curva de Bézier resultante interporla los puntos  $d_i$  y además es  $C^1$ . Ya sabemos que las curvas de Bézier son diferenciables en su dominio  $[0, 1]$ . Solo tenemos que comprobar la diferenciabilidad en los puntos de unión de las diferentes curvas de Bézier cúbicas, por construcción, en  $t = i, i = 1, \dots, n - 1$ . Pero esto es inmediato, puesto que  $\log_{d_i} b_i^- = \log_{d_i} (\exp_{d_i} \bar{x}_i) = \bar{x}_i = -\log_{d_i} (\exp_{d_i} (-\bar{x}_i)) - \log_{d_i} (b_i^+)$ . Y como solo usamos curvas de grado 3,  $K_1 = K_2 = \dots = K_{n-1} = 3$ .

### 3.2.3 Algoritmo de aproximación de puntos

Volvamos a (8). En [4] se propone un esquema que aproxima la segunda derivada covariante, ya que calcular la segunda derivada covariante sobre una curva  $\beta: [0, 1] \rightarrow S$  es computacionalmente costoso. Es decir,

queremos aproximar el término

$$\int_{t_0}^{t_n} \left\| \frac{D^2\beta(t)}{dt} \right\|_{\beta(t)}^2 dt$$

mediante el esquema de diferencias finitas de segundo orden propuesto en [4]. Para ello, consideremos tres puntos sobre la superficie  $x, y, z \in S$  y el conjunto de "puntos medios" entre  $x$  y  $z$ :

$$C_{x,z} := \{c \mid c = g(\frac{1}{2}; x, z)\}$$

para toda geodésica  $g(\cdot; x, z)$  no necesariamente mínima. La diferencia finita de segundo orden sobre una superficie se define por

$$d_2[x, y, z] = \min_{c \in C_{x,z}} 2d_S(c, y)$$

**Nota:** En  $S = \mathbb{R}^n$  esto es equivalente a  $\|x - 2y + z\| = 2\|\frac{1}{2}(x, z) - y\|$ .

Suponiendo (para simplificar la notación) que  $t_0 = 0, \dots, t_N = N$  se tiene que  $\Delta t = 1 = \frac{1}{N}$  y la norma de la aceleración se approxima por:

$$\left\| \frac{D^2\beta(t)}{dt} \right\| \approx \frac{1}{\Delta t^2} d_2^2[\beta(i-1), \beta(i), \beta(i+1)], \quad i = 1, \dots, N-1 \quad (9)$$

Por la regla del trapecio (para integración numérica) obtenemos

$$\int_0^N \left\| \frac{D^2\beta(t)}{dt} \right\|_{\beta(t)}^2 dt \approx \sum_{i=1}^{N-1} \frac{\Delta t d_2^2[\beta(t_{i-1}), \beta(t_i), \beta(t_{i+1})]}{\Delta t^4} = \sum_{i=1}^{N-1} \frac{d_2^2[\beta(t_{i-1}), \beta(t_i), \beta(t_{i+1})]}{\Delta t^3} := A(\Omega) \quad (10)$$

donde  $\Omega$  es el conjunto de todos los puntos de control de  $\beta$ . El siguiente paso consiste en minimizar  $A(\Omega)$ . Para ello, se usará un algoritmo de Descenso de Gradiente sobre  $S^M$  donde  $M$  lo definiremos más tarde. El objetivo será conseguir calcular  $\nabla_{S,b_i} A(\Omega)$ .

Denotamos por  $D_x f[v](x_0) \in T_{f(x_0)} S$  la derivada direccional de  $f: S \rightarrow S$  evaluado en  $x_0 \in S$  en la dirección  $v \in T_x S$ . Si  $g: S \rightarrow S$ , se tiene que

$$D_x(f \circ g)[v](x) = D_{g(x)} f[D_x g[v](x)](f(g(x))) \quad (11)$$

donde  $D_x g[v](x) \in T_{g(x)} S$  y  $D_{g(x)} f[D_x g[v](x)](f(g(x))) \in T_{f(g(x))} S$ .

Por ([2], 3.31) Sea  $f: S \rightarrow \mathbb{R}$ ,  $x \in S$  y  $v \in T_x S$ , el gradiente  $\nabla_S f(x) \in T_x S$  es el vector tangente que cumple

$$\langle \nabla_S f(x), v \rangle_x = D_x f[v](x) \quad \forall v \in T_x S \quad (12)$$

Consideremos ahora una geodésica mínima entre  $x$  e  $y$ :  $g(t; x, y) \quad t \in [0, 1]$ . Sea  $v \in T_x S$  un vector tangente sobre  $x$  y consideremos una geodésica  $\gamma_{x,v}(s)$  tal que  $\gamma_{x,v}(0) = x$  y  $\frac{\partial}{\partial s} \gamma(0) = v \in T_x S$ .

Sea la variación  $h(t, s)$  de  $g$  con dirección de variación  $v \in T_x S$ :

$$h(t, s) := \exp_{\gamma_{x,v}(s)}(t \log_{\gamma_{x,v}(s)} y) \quad t \in [0, 1], s \in (-\epsilon, \epsilon) \quad (13)$$

Denotemos el campo de Jacobi por

$$J(t) = \left. \frac{\partial}{\partial s} h(t, s) \right|_{s=0} \in T_{g(t;x,y)} S \quad (14)$$

Obtenemos la relación ([6]).

$$D_x g(t; \cdot, y)[v] = \frac{\partial}{\partial s} h(t, s) = J(t) = J_{g,v} \quad (15)$$

donde se ha hecho explícito en la notación del campo de Jacobi la geodésica y la dirección de variación.

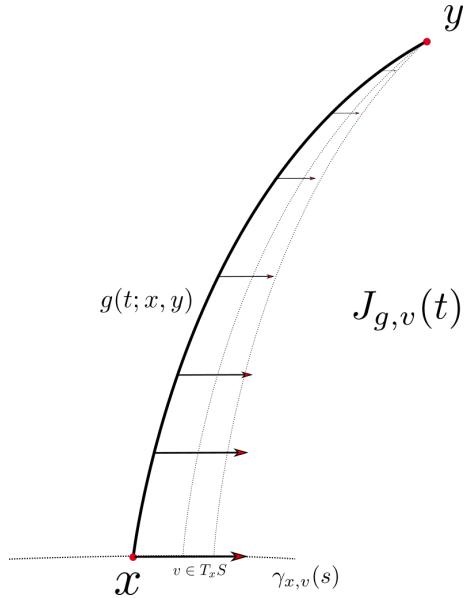


Figure 18: Representación del campo  $J_{g,v}(t)$

El interés por calcular las derivadas direccionales de las geodésicas mínimas mediante el campo de Jacobi es que en espacios simétricos [14], se pueden calcular con una fórmula cerrada [4], cuya implementación ya está incluida en la librería **Manopt.jl** del lenguaje de programación Julia [5].

Recordemos que en el algoritmo De Casteljau sobre superficies, la curva de Bézier se obtenía mediante una combinación de geodésicas. Sea  $g_1(t) = g(t; x, y)$  y  $g_2(t; g_1(t), z)$ . Por la regla de la cadena se tiene

$$D_x g_2(t)[v] = D_{g_1(t)} g(t; \cdot, z)[D_x g_1(t)[v]] \quad (16)$$

Y por (15) tenemos que

$$D_x g_2(t)[v] = J_{g_2, D_x g_1(t)[v]}(t)$$

Y si tuviéramos  $g_3(t) = g(t; z, g_1(t))$  sería

$$D_x g_3(t)[v] = D_{g_1(t)} g(t; z, \cdot)[D_x g_1(t)[v]] = J_{g_3, D_x g_1(t)[v]}(1-t)$$

Es decir, misma dirección que con  $g_2$  pero el campo de Jacobi se recorre al revés.

Sea  $\{e_1^1, \dots, e_m^1\}$  una base ortonormal de  $T_x S$  y  $\{e_1^2, \dots, e_m^2\}$  una base ortonormal de  $T_{g_1(t)} S$ , como  $v \in T_x S$ ,  $v = \sum_{l=1}^m v_l e_l^1$  y podemos expresar el campo de Jacobi a lo largo de la geodésica  $g_1(t)$  con dirección de variación  $e_l^1$  (el vector  $l$ -ésimo de la base ortonormal de  $T_x S$ ), se tiene  $J_{g_1, e_l^1}(t) = \sum_{j=1}^m \alpha_j e_j^2 \in T_{g_1(t)} S$ . Finalmente, como  $D_x g_1(t)[v] = J_{g_1, v}$ , se tiene

$$D_x g_2(t)[v] = \sum_{j=1}^m \sum_{l=1}^m J_{g_2, e_j^2}(t) \alpha_j v_l$$

Recordemos que el algoritmo de De Casteljau en superficies construía la curva de Bézier de grado  $K$   $\beta_k(t)$  (con  $k+1$  puntos de control) de forma recursiva usando geodésicas. Por ejemplo, para  $K=3$ , la curva de Bézier cúbica  $B_3(t)$  con puntos de control  $\{b_0, b_1, b_2, b_3\}$  se construía como:

Para  $k=0$ :  $g_0^0 = b_0$ ,  $g_1^0 = b_1$ ,  $g_2^0 = b_2$ ,  $g_3^0 = b_3$

Para  $k=1$ :  $g_0^1(t) = g(t; g_0^0, g_1^0)$ ,  $g_1^1(t) = g(t; g_1^0, g_2^0)$ ,  $g_2^1(t) = g(t; g_2^0, g_3^0)$

Para  $k=2$ :  $g_0^2(t) = g(t; g_0^1(t), g_1^1(t))$ ,  $g_1^2(t) = g(t; g_1^1(t), g_2^1(t))$

Para  $k=3$ :  $g_3^0(t) = g(t; g_0^2(t), g_1^2(t)) = \beta_3(t)$

Observemos que  $g_j^k(t) = g(t; g_j^{k-1}(t), g_{j+1}^{k-1}(t)) = \beta_k(t; b_j, \dots, b_{j+k})$ , es decir, se puede considerar una curva de Bézier de grado  $k$  con puntos de control  $\{b_j, \dots, b_{j+k}\}$ . Denotemos por

$$v_j^k(t) = D_x g_j^k[v](t)$$

la derivada direccional de  $g_j^k$  en el punto  $x \in \{b_j, \dots, b_{j+k}\}$  en la dirección  $v \in T_x S$ . Si  $b_i, i \neq j$ , se tiene  $D_{b_i} g_j^k[v](t) = 0$ .

Consideremos  $b_0$  y  $v \in T_{b_0} S$ , se tiene  $D_{b_0} g_0^0[v] = v$  y  $D_{b_0} g_j^0[v] = 0$  para  $j = 1, 2, 3$ .

Por otra parte,  $D_{b_0} g_0^1[v]$  es la derivada direccional de la geodésica  $g_0^1$  en el punto  $b_0$  en la dirección  $v$ . Por (15) se tiene que  $D_{b_0} g_0^1[v] = J_{g_0^1, v}(t) = v_0^1(t)$  y como  $g_1^1(t)$  y  $g_2^1(t)$  no dependen de  $b_0$ , entonces  $D_{b_0} g_j^1(t)[v] = 0$  para  $j = 1, 2$ .

Ahora, para  $k = 2$ , tenemos dos geodésicas:  $g_0^2(t)$  y  $g_1^2(t)$ . Por una parte,  $g_0^2(t)$  sí depende de  $b_0$  y como es su punto inicial se tiene que  $D_{b_0} g_0^2(t)[v] = D_{g_0^1(t)} g_0^2[D_{b_0} g_0^1(t)[v]] = J_{g_0^2, v_0^1(t)} = v_0^2(t)$ . Por otra parte, la otra geodésica  $g_1^2(t)$  no depende de  $b_0$  así que  $D_{b_0} g_1^2(t)[v] = 0$ .

Finalmente, para  $k = 3$  tenemos que  $\beta_3(t) = g(t; g_0^2(t), g_1^2(t))$ . Donde solo  $g_0^2(t)$  depende de  $b_0$ . Por tanto,  $D_{b_0} g_0^3(t)[v] = D_{g_0^2(t)} g_0^3(t)[D_{b_0} g_0^2(t)[v]] = J_{g_0^3, v_0^2(t)}$ .

En resumen, la derivada direccional de  $\beta_3(t)$  en el primer punto de control  $b_0$  con una dirección  $v \in T_{b_0} S$  se calcula a través de los campos de Jacobi de forma recursiva. Veamos ahora cómo sería para el segundo punto de control  $b_1$ .

Se abusa de notación y denotamos  $v \in T_{b_1} S$ .

$$\text{Para } k = 0: D_{b_1} g_0^0[v] = 0, \quad D_{b_1} g_1^0[v] = v, \quad D_{b_1} g_j^0[v] = 0, \quad j = 2, 3$$

Como  $b_1$  aparece en  $g_0^1(t)$  (ya que es la geodésica que va de  $b_0$  a  $b_1$ ) y en  $g_1^1(t)$  (ya que es la que va de  $b_1$  a  $b_2$ ). Se tiene que  $D_{b_1} g_0^1(t)[v] = J_{\bar{g}_0^1(t), v}(1-t) = v_0^1(t)$  ya que por definición se debe recorrer al revés, empezando por  $b_1$ . y  $D_{b_1} g_1^1(t)[v] = J_{g_1^1(t), v}(t) = v_1^1(t)$ . Como  $g_2^1(t)$  no depende de  $b_1$ ,  $D_{b_1} g_2^1(t)[v] = 0$ .

$$\begin{aligned} \text{Para } k = 1: D_{b_1} g_0^1(t)[v] &= J_{\bar{g}_0^1(t), v}(1-t) = v_0^1(t) \\ D_{b_1} g_1^1(t)[v] &= J_{g_1^1(t), v}(t) = v_1^1(t) \end{aligned}$$

Ahora bien, como  $g_0^2(t) = g(t; g_0^1(t), g_1^1(t))$  y tanto  $g_0^1$  como  $g_1^1(t)$  dependen de  $b_1$ , se tiene que:

$$\begin{aligned} D_{b_1} g_0^2(t)[v] &= D_{g_0^1(t)} g_0^2(t)[D_{b_1} g_0^1(t)[v]] + D_{g_1^1(t)} g_0^2(t)[D_{b_1} g_1^1(t)[v]] = \\ &= D_{g_0^1(t)} g_0^2(t)[v_0^1(t)] + D_{g_1^1(t)} g_0^2(t)[v_1^1(t)] = \\ &= J_{g_0^2(t), v_0^1(t)}(t) + J_{g_0^2(t), v_1^1(t)}(1-t) = v_0^2(t) \end{aligned}$$

Y  $D_{b_1} g_1^2(t)[v] = D_{g_1^1(t)} g_1^2(t)[D_{b_1} g_1^1(t)[v]] = J_{g_1^2(t), v_1^1(t)} = v_1^2(t)$  ya que  $g_1^2(t) = g(t; g_1^1(t), g_2^1(t))$  y solo  $g_1^1(t)$  depende de  $b_1$ . Entonces

$$\begin{aligned} \text{Para } k = 2: D_{b_1} g_0^2(t)[v] &= J_{g_0^2(t), v_0^1(t)}(t) + J_{g_0^2(t), v_1^1(t)}(1-t) = v_0^2(t) \\ D_{b_1} g_1^2(t)[v] &= J_{g_1^2(t), v_1^1(t)} = v_1^2(t) \end{aligned}$$

Por último,  $g_0^3(t) = g(t; g_0^2(t), g_1^2(t))$  donde tanto  $g_0^2$  como  $g_1^2$  dependen de  $b_1$ . Por el mismo razonamiento de antes:

$$\begin{aligned} \text{Para } k = 3: D_{b_1} g_0^3(t)[v] &= D_{g_0^2(t)} g_0^3(t)[v_0^2(t)] + D_{g_1^2(t)} g_0^3(t)[v_1^2(t)] = \\ &= J_{g_0^3(t), v_0^2(t)} + J_{g_0^3(t), v_1^2(t)}(1-t) = D_{b_1} \beta_3(t)[v] \end{aligned}$$

Vamos a ver la expresión general para calcular la derivada direccional de una curva de Bézier de forma recursiva sobre un punto de control. La demostración se puede encontrar en [6], que se basa en la misma idea que se ha desarrollado en el ejemplo anterior.

**Teorema 3.1.** Dada una geodésica  $g_i^k(t)$  del algoritmo de De Casteljau sobre superficies, con puntos de control  $\{b_i, \dots, b_{i+k}\}$ . La derivada direccional de  $g_i^k$  sobre un punto de control  $b_s \in \{b_i, \dots, b_{i+k}\}$  en la dirección  $v \in T_{b_s}S$  viene dada por

$$v_i^k(t) = D_{b_s} g_i^k(t)[v] = \begin{cases} J_{g_i^k(t), v_i^{k-1}(t)}(t), & \text{si } i = s \\ J_{g_i^k(t), v_i^{k-1}(t)}(t) + J_{\bar{g}_i^k(t), v_{i+1}^{k-1}}(1-t), & \text{si } i < s < i+k \\ J_{\bar{g}_i^k(t), v_{i+1}^{k-1}(t)}(1-t), & \text{si } s = i+k \end{cases}$$

¿Cómo garantizamos la continuidad  $C^1$ ?

Usando la misma notación que en la sección anterior, tenemos una sucesión  $\{\beta_{K_i}^i\}_i$  con  $i = 0, \dots, n-1$  donde la  $i$ -ésima curva de Bézier  $B_{K_i}^i$  de grado  $K_i$  tiene puntos de control  $\{b_0^i, \dots, b_{K_i}^i\}$ . Supondremos  $K_i = K \quad \forall i$ . Recordamos que habíamos denotado por  $p_i$  al último punto de control de la curva  $(i-1)$ -ésima y el primer punto de control  $i$ -ésima, que deben coincidir para obtener la continuidad  $C^0$ . Es decir,  $p_i = b_{K_i}^{i-1} = b_0^i$  donde la  $(i-1)$ -ésima curva de Bézier tiene puntos de control  $\{p_{i-1}, \dots, b_i^-, p_i\}$  y la  $i$ -ésima curva de Bézier tiene los puntos de control  $\{p_i, b_i^+, \dots, p_{i+1}\}$ .

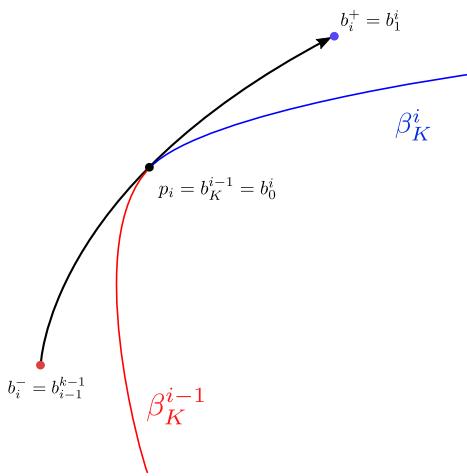


Figure 19: Continuidad  $C^1$

$\{p_i, \hat{g}(1; b_i^-, p_i, b_i^+), \dots, p_{i+1}\}$  donde  $b_i^-$  aparece en función de la geodésica  $\hat{g}$ , por eso, se debe aplicar la regla de la cadena:

$$D_{b_i^+} \beta_K^i(t-i; p_i, \cdot, \dots, p_{i+1}) [D_{b_i^-} \hat{g}(1)[v]]$$

Algo similar ocurre para  $p_i$ . En  $t \in (i-1, i]$ ,  $\beta_K^{i-1}$  tiene a  $p_i$  como su último punto de control directamente así que la derivada direccional en ese punto es inmediato. Sin embargo, para  $\beta_K^i$ ,  $p_i$  aparece en función de  $b_i^+ = \hat{g}(t; b_i^-, p_i, b_i^+)$ , así que habría que aplicar la regla de la cadena. Observamos que esta condición afecta a  $n-1$  curvas (desde la segunda a la última).

Volviendo a (10), ya tenemos todo lo necesario para poder calcular el gradiente en nuestra función objetivo:

**Teorema 3.2.** Sea  $S$  una superficie de dimensión  $m$ ,  $b_s$  uno de los puntos de control de la curva a trozos  $\beta(t)$  y  $\{e_1, \dots, e_m\}$  una base ortonormal de  $T_{b_s}S$ . El gradiente de (10) en  $b_s$  es

$$\nabla_x A(\Omega) = \frac{1}{\Delta t^3} \sum_{l=1}^m \sum_{i=1}^{n-1} \sum_{j=i-1}^{i+1} \langle \nabla d_{2,i}^2, D_{b_s} \beta^j[e_l] \rangle v_l \quad (17)$$

*Demostración.* Como  $\nabla_{b_s} A(\Omega) \in T_{b_s} S \Rightarrow \nabla_{b_s} A(\Omega) = \sum_{l=1}^m a_l(b_s) e_l$ . Sea  $v = \sum_{l=1}^m v_l e_l \in T_{b_s} S$ ,  $\langle \nabla_{b_s} A(\Omega), v \rangle = \sum_{l=1}^m a_l(b_s) v_l$ .

Por (10)

$$A(\Omega) = \sum_{i=1}^{n-1} \frac{d_2^2[\beta(t_{i-1}), \beta(t_i), \beta(t_{i+1})]}{\Delta t^3} = \sum_{i=1}^{n-1} \frac{d_{2,i}^2}{\Delta t^3} \quad (18)$$

Por propiedades del producto escalar,  $\langle \nabla_{b_s} A(\Omega), v \rangle = \frac{1}{\Delta t^3} \sum_{i=1}^{n-1} \langle \nabla_{b_s} d_{2,i}^2, v \rangle \stackrel{(12)}{=} \frac{1}{\Delta t^3} \sum_{i=1}^{n-1} D_{b_s} d_{2,i}^2 [v]$

Por definición,  $d_{2,i}^2$  depende de las curvas  $\beta(t_{i-1})$ ,  $\beta(t_i)$  y  $\beta(t_{i+1})$ . Por tanto, por (11):

$$D_{b_s} d_{2,i}^2 [v] = D_{\beta(t_{i-1})} d_{2,i}^2 [D_{b_s} \beta(t_{i-1})[v]] + D_{\beta(t_i)} d_{2,i}^2 [D_{b_s} \beta(t_i)[v]] D_{\beta(t_{i+1})} d_{2,i}^2 [D_{b_s} \beta(t_{i+1})[v]]$$

Y por (12),  $D_{\beta(t_j)} d_{2,i}^2 [D_{b_s} \beta(t_j)[v]] = \langle \nabla d_{2,i}^2, D_{b_s} \beta(t_j)[v] \rangle$

donde, obtenemos una expresión cerrada para  $\nabla d_{2,i}^2$  en [4] y por lo visto anteriormente en esta sección, se puede calcular  $D_{b_s} \beta(t_j)[v]$  a través de campos de Jacobi (con fórmula cerrada en [6]), de forma recursiva. Finalmente, como  $v = \sum_{l=1}^m v_l e_l \Rightarrow D_{b_s} \beta(t_j)[v] = \sum_{l=1}^m v_l D_{b_s} \beta(t_j)[e_l]$  y por tanto

$$D_{\beta(t_j)} d_{2,i}^2 [D_{b_s} \beta(t_j)[v]] = \sum_{l=1}^m v_l \langle \nabla d_{2,i}^2, D_{b_s} \beta(t_j)[e_l] \rangle$$

obteniendo

$$\langle \nabla_{b_s} A(\beta), v \rangle = \frac{1}{\Delta t^3} \sum_{l=1}^m v_l \sum_{i=1}^{n-1} \left( \langle \nabla d_{2,i}^2, D_{b_s} \beta(t_{i-1})[e_l] \rangle + \langle \nabla d_{2,i}^2, D_{b_s} \beta(t_i)[e_l] \rangle + \langle \nabla d_{2,i}^2, D_{b_s} \beta(t_{i+1})[e_l] \rangle \right)$$

quedando finalmente

$$\nabla_x A(\beta) = \frac{1}{\Delta t^3} \sum_{l=1}^m \sum_{i=1}^{n-1} \sum_{j=i-1}^{i+1} \langle \nabla d_{2,i}^2, D_{b_s} \beta^j[e_l] \rangle v_l \quad (19)$$

□

Por último, mencionar que el algoritmo implementado en la librería *Manopt.jl* [5], no tiene en cuenta la curva  $\beta: [0, n] \rightarrow S$ , solo considera todos los puntos de control  $\Omega = (p_0, b_0^+, \dots, b_n^-, p_n) \in S^M$ . Por las condiciones de continuidad  $C^1$ , todas las  $b_i^+$  dependen de  $b_{i-1}^-$  y  $p_i$ . Además que las  $p_i = b_{i-1}^- = b_i^+$  también están fijadas para conseguir  $C^0$ , por tanto desde la segunda curva de Bézier hasta la última, se debe considerar un punto de control menos a optimizar:  $K - 1$  (ya que hemos supuesto que todas las curvas tienen el mismo grado  $K$ ). Es decir,  $\beta$  debe ser determinada mediante  $(K + 1) + (n - 1) * (K - 1) = M$ , que son los puntos de control a optimizar.

Por [5] y ([11], definición 2.53), si  $F(\Omega) = A(\Omega) + \lambda \sum_{i=0}^n d^2(p_i, d_i)$  su gradiente es

$$\nabla_{b_s} F(\Omega) = \begin{cases} \nabla_{b_s} A(\Omega) - 2\lambda \log_{p_i} d_i, & \text{si } b_s = p_i \\ \nabla_{b_s} A(\Omega), & \text{en otro caso} \end{cases}$$

En Julia, podemos definir la función objetivo utilizando la función *cost\_L2\_acceleration\_bezier* y definir su gradiente mediante *grad\_L2\_acceleration\_bezier*.

Una intuición del algoritmo del descenso del gradiente se puede ver en el Apéndice B.

Por último, para definir un algoritmo de descenso por gradiente sobre  $\Omega$  se puede usar la función *gradient\_descent* dándole unos valores iniciales a los puntos de control y definiendo un tamaño de paso y un criterio de parada [11]. Finalmente, se puede exportar los resultados gráficamente a través de un archivo tipo *asymptote*.

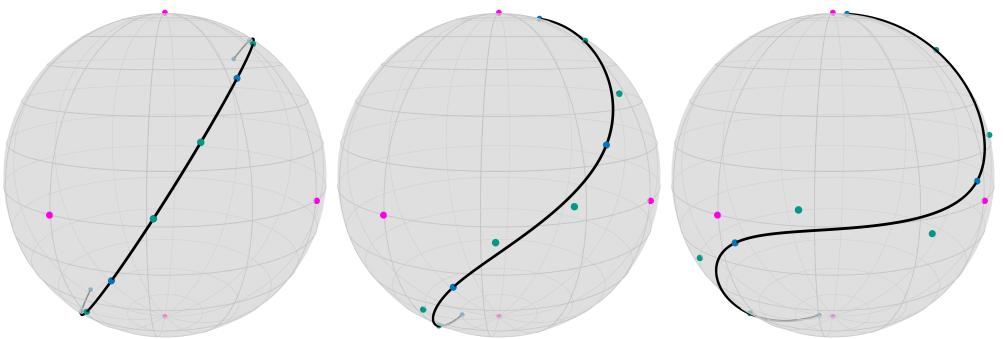


Figure 20: Ejemplo usando la librería Manopt.jl, los puntos  $d_i$  en morado y la curva de Bézier en negro, con sus puntos de control en verde (para los puntos interiores) y en azul los  $p_i$ . De izquierda a derecha, se ha aplicado el algoritmo para  $\lambda = 0, 3, 20$

## 4 Conclusiones

En este trabajo se ha estudiado cómo en diferentes ámbitos se pueden usar las curvas de Bézier. En el ámbito del diseño, generalizando su definición original mediante reparametrizaciones sencillas, se ha conseguido una familia de curvas con una gran versatilidad pudiendo modificar tanto la forma como la longitud sin necesidad de modificar los puntos de control. Aunque sus aplicaciones prácticas en el diseño pueden ser limitadas para los programas gráficos debido al gran número de parámetros involucrados, puede ser una herramienta muy útil en el campo de la programación creativa, donde la teoría matemática no solo no es un impedimento, sino que es el medio que se usa para poder crear arte.

Por otra parte, en el ámbito de la optimización, mediante conocimientos básicos obtenidos en las asignaturas de Geometría Diferencial y Aplicaciones, Álgebra Lineal y Métodos Numéricos, se ha podido estudiar diferentes algoritmos que usan las curvas de Bézier para interpolar y aproximar puntos sobre una superficie (en el trabajo, sobre  $\mathbb{S}^2$ ). Para ello, se ha tenido que ampliar algunos conceptos que no se habían estudiado en el grado como la derivada covariante, las geodésicas y los campos de Jacobi. Todo ello, desde un punto de vista fundamental usando notación muy básica utilizando la bibliografía típica de las asignaturas de Geometría del grado [10].

Se han conseguido entonces dos objetivos: aprender sobre diferentes usos de las curvas de Bézier viendo cómo una definición simple y elegante puede derivar en un área de investigación actualmente en desarrollo en ámbitos ajenos a los pensados originalmente. Y también, ha servido para poder conocer, aunque sea de forma superficial, diferentes campos en matemáticas como la optimización en espacios no euclídeos. Además de poder hacer uso de Julia, un lenguaje de programación relativamente poco conocido, que se ha estudiado en las asignaturas de Taller de Tecnomatemática y Simulación Numérica, aprendiendo sobre librerías en desarrollo como Manopt o Manifolds. Al ser aún código en desarrollo y pequeño, se ha podido estudiar el código fuente (libre) para poder comprender mejor cómo funcionan los algoritmos numéricos que implementan la teoría matemática estudiada. Siendo de mucha ayuda no solo para comprender bien cómo funcionan los algoritmos, sino además para aprender a leer, usar y desarrollar código abierto.

Una habilidad importante que me ha obligado a desarrollar este trabajo ha sido poder leer un gran número de artículos y poder hacerme una idea de lo que tratan, aún sin la necesidad de poder estudiarlo en profundidad. Esto me ha servido para poder usar diferentes artículos de diversos autores para saber qué algoritmos y métodos se están utilizando y qué ventajas o inconvenientes presentan, y así, poder elegir cuáles me gustaría probar a implementar en código y estudiar en detalle. Considero que haber estudiado dos cursos en común con el grado de Matemáticas me ha permitido conseguir una base matemática con la poder comprender áreas desconocidas para mí de las matemáticas aplicadas.

La teoría ha venido acompañada por numerosas gráficas que se han obtenido en su mayoría implementando los algoritmos desarrollados en los lenguajes de programación Python, Julia y R, haciendo uso de numerosas librerías de código abierto. Las gráficas que no se han obtenido mediante código, se han realizado en Inkscape, usando exclusivamente curvas de Bézier.

## 5 Apéndices

### 5.1 Apéndice A

En el contexto de la regresión no paramétrica, queremos estudiar la relación entre una serie de predictores  $x_1, \dots, x_p$  y una variable respuesta  $Y$ :

$$Y = f(x_1, \dots, x_p) + \epsilon$$

Donde no conocemos la forma de  $f$  y suponemos  $\epsilon \sim N(0, 1)$ . Por simplificar el ejemplo, supongamos que solo tenemos un predictor  $x$ , es decir,  $Y = f(x) + \epsilon$ . Un método para estudiar la relación es utilizando la regresión mediante splines suavizados [16].

Los **splines suavizados** se obtienen como la función  $s(x) \in C^2$  que minimiza la suma de cuadrados residual más una penalización que mide su rugosidad. La penalización se mide mediante el parámetro  $\lambda$ . Supongamos que tenemos los datos  $\{x_1, \dots, x_n\}$  asociados a  $\{y_1, \dots, y_n\}$ . La función objetivo a minimizar, una vez se ha fijado lambda, es:

$$\sum_{i=1}^n (y_i - s(x_i))^2 + \lambda \int s''(x)^2 dx$$

Como vemos, esta función objetivo es similar a (8), salvo que el parámetro  $\lambda$  en nuestro caso penaliza la distancia a los puntos  $x_i$  y no a la "rugosidad". Se supone que  $s(x)$  es un spline cúbico. Para  $\lambda = 0$  se consigue la interpolación y para  $\lambda \rightarrow \infty$  el spline tiende a una recta. Por ejemplo, usando el dataset *MultiKink*, podemos estudiar la relación entre la medida del grosor del pliegue de la piel en el triceps ( $Y$ ) de deportistas y su edad  $x$ . Vamos a usar varios valores de  $\lambda$  para ver el efecto en el spline resultante:

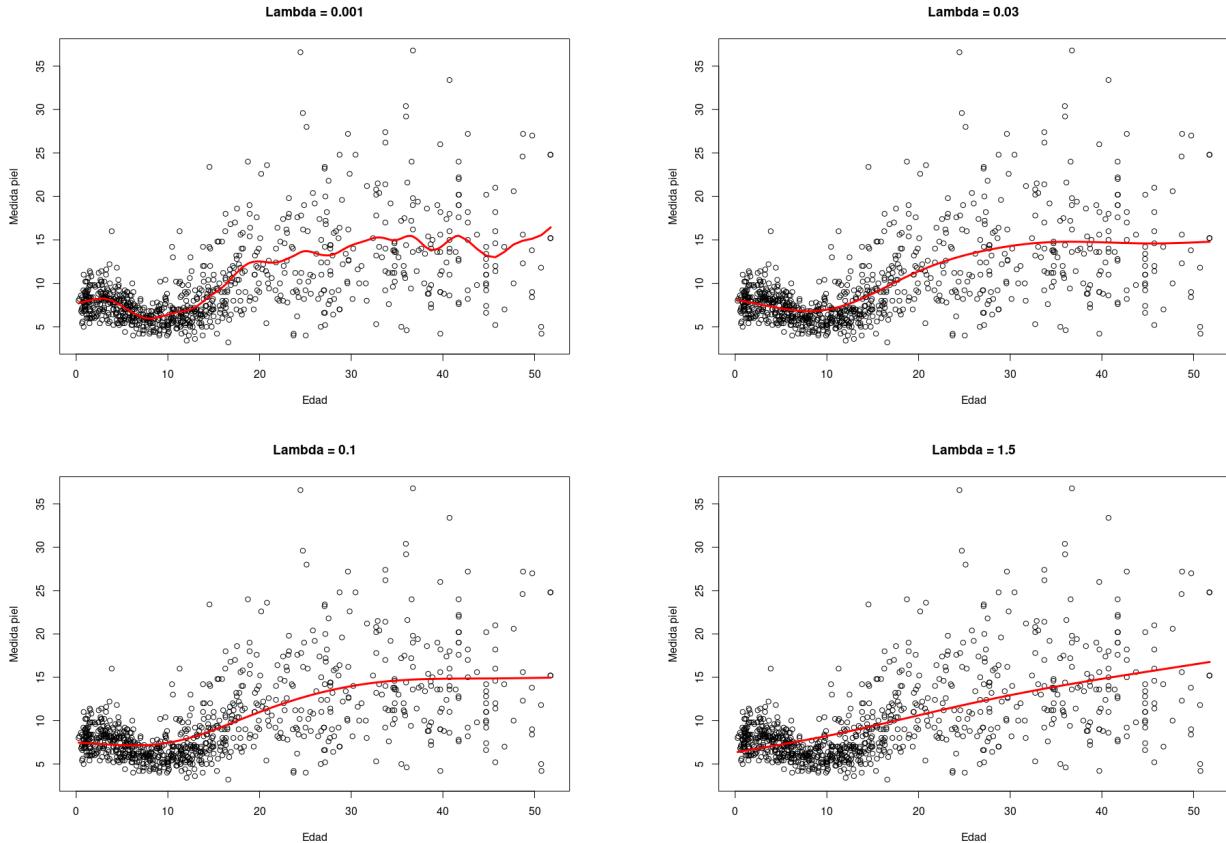


Figure 21: Spline suavizado (rojo) para diferentes valores de  $\lambda$

## 5.2 Apéndice B

Los métodos de búsqueda lineal en  $\mathbb{R}^n$  se basan en actualizar la fórmula

$$x_{k+1} = x_k + t_k \eta_k$$

donde  $\eta_k$  es la dirección de búsqueda y  $t_k \in \mathbb{R}_+$  es el tamaño del paso. Por ejemplo, el método del descenso del gradiente en  $\mathbb{R}^n$  utiliza como dirección  $-\nabla f$ . Es decir,  $x_{k+1} = x_k - t_k \nabla f(x_k)$  con  $k \geq 0$ . Obtenemos así una secuencia  $f(x_0) \geq f(x_1) \geq f(x_2) \geq \dots$ , donde se busca que la secuencia  $(x_k)_{k \geq 0}$  converja al mínimo local.

Por [2], sea  $F: S^M \rightarrow \mathbb{R}$ , su gradiente  $\nabla F$ ,  $x_0 \in S^M$ , el tamaño de paso  $s_k > 0$ ,  $k \in \mathbb{N}$ . El algoritmo de descenso de gradiente implementado en Manopt.jl utiliza

$$x_{k+1} = \exp_{x_k}(-s_k \nabla F(x_k))$$

hasta que se alcanzo un criterio de parada. Por ([2], capítulo 4), el tamaño del paso se da usando la condición de la búsqueda lineal de Armijo: sean  $\beta, \sigma \in (0, 1)$ ,  $\alpha > 0$  y sea  $m \in \mathbb{N}$  el entero positivo más pequeño que cumpla

$$F(x_k) - F(\exp_{x_k}(-\beta^m \alpha \nabla F(x_k))) \geq \sigma \beta^m \alpha \|\nabla F(x)\|_{x_k}$$

Se define el tamaño del paso como  $s_k := \beta^m \alpha$ . En Manopt.jl, está implementado en la función *ArmijoLInesearch* donde se ha usado utilizando los parámetros por defecto (retracción exponencial,  $\beta = 0.95$ ,  $\sigma = 0.1$  y paso inicial de 1) y un criterio de parada cuando la norma de la diferencia entre el valor actual y el anterior es menor a  $10^{-5}$ . El código que se ha utilizado para conseguir una de las figuras de (20) (cambiando los valores de lambda)

```
using Manopt, Manifolds, Random, Colors

geo_pts = collect(range(0.0, 1.0; length=101))
bezier_pts = collect(range(0.0, 3.0; length=201))
camera_position = (-1.0, -0.7, 0.3)

black = RGBA{Float64}(colorant"#000000")
TolVibrantBlue = RGBA{Float64}(colorant"#0077BB")
TolVibrantOrange = RGBA{Float64}(colorant"#EE7733")
TolVibrantMagenta = RGBA{Float64}(colorant"#EE3377")
TolVibrantCyan = RGBA{Float64}(colorant"#33BBEE")
TolVibrantTeal = RGBA{Float64}(colorant"#009988")

M = Sphere(2)
B = artificial_S2_composite_bezier_curve()
b = B[2].pts

curve_samples = collect(range(0.0, 3.0; length=151))
pB = get_bezier_points(M, B, :differentiable)
N = PowerManifold(M, NestedPowerRepresentation(), length(pB))

lambda = 20.0
d = get_bezier_junctions(M, B)
function F2(M, pB)
    return cost_L2_acceleration_bezier(
        M.manifold, pB, get_bezier_degrees(M.manifold, B), curve_samples, lambda, d
    )
end
```

```

function gradF2(M, pB)
    return grad_L2_acceleration_bezier(
        M.manifold, pB, get_bezier_degrees(M.manifold, B), curve_samples, lambda, d
    )
end
x1 = get_bezier_points(M, B, :differentiable)
pB_opt_appr = gradient_descent(
    N,
    F2,
    gradF2,
    x1;
    stepsize=ArmijoLinesearch(M; contraction_factor=0.5, sufficient_decrease=0.0001),
    stopping_criterion=StopWhenChangeLess(10.0^(-5)),
)
B_opt_appr = get_bezier_segments(M, pB_opt_appr, get_bezier_degrees(M, B), :differentiable)
asymptote_export_S2_signals(
    image_prefix * "/Bezier-Appr-Min.asy";
    curves=[de_casteljau(M, B_opt_appr, bezier_pts), de_casteljau(M, B, bezier_pts)],
    points=[
        d,
        get_bezier_junctions(M, B_opt_appr),
        get_bezier_inner_points(M, B_opt_appr),
    ],
    colors=Dict(
        :curves => [TolVibrantBlue, black],
        :points => [TolVibrantOrange, TolVibrantBlue, TolVibrantTeal]
        #:tectors => [TolVibrantCyan],
    ),
    camera_position=camera_position,
    #arrow_head_size=10.0,
    line_widths=[1.5, 0.75, 1.5],
    dot_size=4.0,
)
render_asymptote(image_prefix * "/Bezier-Appr-Min.asy"; render=2)

```

### 5.3 Apéndice C

El código en Julia utilizado para las figuras (15), (16) y (17) ha sido:

```

using Plots
using Manifolds

function sphere_point(u, v)
    x = cos(u) * sin(v)
    y = sin(u) * sin(v)
    z = cos(v)
    return [x, y, z]
end

function plot_sphere()
    u = range(0, 2*pi, length=50)
    v = range(0, pi, length=50)
    x = [sphere_point(ui, vi)[1] for ui in u, vi in v]
    y = [sphere_point(ui, vi)[2] for ui in u, vi in v]
    z = [sphere_point(ui, vi)[3] for ui in u, vi in v]

```

```

plot(x, y, z, st=:surface, color=:lightblue, legend=false, alpha=0.5)
end

"""
Dada una lista de puntos sobre la esfera, devuelve las
coordenadas en 3 listas diferentes: X, Y y Z
"""
function getXYZ(lista_puntos)
    x = []
    y = []
    z = []
    for i = 1:length(lista_puntos)
        x = [x; lista_puntos[i][1]]
        y = [y; lista_puntos[i][2]]
        z = [z; lista_puntos[i][3]]
    end
    return x, y, z
end

b = [sphere_point(0,0), sphere_point(0,1), sphere_point(2, -2)]
plot_sphere()
scatter3d!(getXYZ(b), color=:yellow)

"""
Dada una lista de puntos de control "b", calcular
el algoritmo De Casteljau sobre una superficie regular
"""
function DeCasteljauSuperficies(b, t)
    K = length(b) - 1
    x = copy(b)
    M = Sphere(2)
    for k in 1:K
        for j in 1:K-k+1
            x[j] = shortest_geodesic(M, x[j], x[j+1], t)
        end
    end
    return x[1]
end

bezier(t) = DeCasteljauSuperficies(b,t)
ts = [0:0.01:1;]
beta = bezier.(ts)
plot!(getXYZ(beta), color=:red, lw=:2)

g(t) = shortest_geodesic(M, sphere_point(0,0), sphere_point(0,1), t)
ts = [0:0.01:1;]
curva = g.(ts);
X, Y, Z = getXYZ(curva)
plot!(X, Y, Z, linewidth=1.5, color=:black)

g(t) = shortest_geodesic(M, sphere_point(0,1), sphere_point(2, -2), t)
ts = [0:0.01:1;]
curva = g.(ts);
X, Y, Z = getXYZ(curva)
plot!(X, Y, Z, linewidth=1.5, color=:black)

```

```

bezier(t) = DeCasteljauSuperficies(b,t)
ts = [0:0.01:1;]
beta = bezier.(ts)
plot!(getXYZ(beta), color=:red, lw=:2)

#####
t = 0.3
P1 = shortest_geodesic(M, sphere_point(0,0), sphere_point(0,1), 0.3)
P2 = shortest_geodesic(M, sphere_point(0,1), sphere_point(2, -2), 0.3)
scatter3d!([P1[1]], [P1[2]], [P1[3]], color=:blue, alpha=0.5)
scatter3d!([P2[1]], [P2[2]], [P2[3]], color=:blue, alpha=0.5)
g(t) = shortest_geodesic(M, P1, P2, t)
ts = [0:0.01:1;]
curva = g.(ts);
X, Y, Z = getXYZ(curva)
plot!(X, Y, Z, linewidth=1.5, color=:blue, alpha=0.5)
PM = g(0.3) #Punto de la curva
scatter3d!([PM[1]], [PM[2]], [PM[3]], color=:blue)

t = 0.7
P1 = shortest_geodesic(M, sphere_point(0,0), sphere_point(0,1), 0.7)
P2 = shortest_geodesic(M, sphere_point(0,1), sphere_point(2, -2), 0.7)
scatter3d!([P1[1]], [P1[2]], [P1[3]], color=:green, alpha=0.5)
scatter3d!([P2[1]], [P2[2]], [P2[3]], color=:green, alpha=0.5)
g(t) = shortest_geodesic(M, P1, P2, t)
ts = [0:0.01:1;]
curva = g.(ts);
X, Y, Z = getXYZ(curva)
plot!(X, Y, Z, linewidth=1.5, color=:green, alpha=0.7)
PM = g(0.7) #Punto de la curva

scatter3d!([PM[1]], [PM[2]], [PM[3]], color=:green, dpi=300)

```

El código para las figuras de la sección 2 requiere muchas líneas de código y se puede consultar en la carpeta CÓDIGOS en el link: [www.github.com/JSemperH/TFG](http://www.github.com/JSemperH/TFG).

El resto de figuras se han realizado utilizando el programa de código abierto Inkscape, usando únicamente curvas de Bézier.

## 6 Referencias

### References

- [1] P-A Absil, Pierre-Yves Gousenbourger, Paul Striewski, and Benedikt Wirth. Differentiable piecewise-bézier surfaces on riemannian manifolds. *SIAM Journal on Imaging Sciences*, 9(4):1788–1828, 2016.
- [2] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.
- [3] Antoine Arnould, Pierre-Yves Gousenbourger, Chafik Samir, Pierre-Antoine Absil, and Michel Canis. Fitting smooth paths on riemannian manifolds: endometrial surface reconstruction and preoperative mri-based navigation. In *Geometric Science of Information: Second International Conference, GSI 2015, Palaiseau, France, October 28-30, 2015, Proceedings 2*, pages 491–498. Springer, 2015.
- [4] Miroslav Bacák, Ronny Bergmann, Gabriele Steidl, and Andreas Weinmann. A second order nonsmooth variational model for restoring manifold-valued images. *SIAM Journal on Scientific Computing*, 38(1):A567–A597, 2016.
- [5] Ronny Bergmann. Manopt. jl: Optimization on manifolds in julia. *Journal of Open Source Software*, 7(70):3866, 2022.
- [6] Ronny Bergmann and Pierre-Yves Gousenbourger. A variational model for data fitting on manifolds by minimizing the acceleration of a bézier curve. *Frontiers in Applied Mathematics and Statistics*, 4:59, 2018.
- [7] Wolfgang Boehm and Andreas Müller. On de casteljau’s algorithm. *Computer Aided Geometric Design*, 16(7):587–605, 1999.
- [8] Nicolas Boumal. *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023.
- [9] Ji-wung Choi and Gabriel Hugh Elkaim. Bézier curves for trajectory guidance. In *World Congress on Engineering and Computer Science, WCECS*, pages 22–24. Citeseer, 2008.
- [10] Manfredo P Do Carmo. *Differential geometry of curves and surfaces: revised and updated second edition*. Courier Dover Publications, 2016.
- [11] Pierre-Yves Gousenbourger. *Interpolation and fitting on Riemannian manifolds*. PhD thesis, UCLouvain, ICTEAM institute, 2020.
- [12] Pierre-Yves Gousenbourger, Estelle Massart, and P-A Absil. Data fitting on manifolds with composite bézier-like curves and blended cubic splines. *Journal of Mathematical Imaging and Vision*, 61(5):645–671, 2019.
- [13] Peter J Green and Bernard W Silverman. *Nonparametric regression and generalized linear models: a roughness penalty approach*. Crc Press, 1993.
- [14] Sigurdur Helgason. *Differential geometry and symmetric spaces*, volume 341. American Mathematical Soc., 2001.
- [15] Wen Huang. *Optimization algorithms on Riemannian manifolds with applications*. PhD thesis, The Florida State University, 2013.
- [16] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.
- [17] Syed Ahmad Aidil Adha Said Mad, Md Yushalify Misro, Kenjiro T Miura, et al. Generalized riemann-liouville fractional bézier curve and its applications in engineering surface. *Alexandria Engineering Journal*, 65:585–606, 2023.
- [18] Syed Ahmad Aidil Adha Said Mad, Md Yushalify Misro Zain, and Kenjiro T Miura. Curve fitting using generalized fractional bézier curve. 2023.

- [19] Claudio Mancinelli, Giacomo Nazzaro, Fabio Pellacini, and Enrico Puppo. b/surf: Interactive bézier splines on surface meshes. *IEEE Transactions on Visualization and Computer Graphics*, 2022.
- [20] Tomasz Popiel and Lyle Noakes. Bézier curves and c<sub>2</sub> interpolation in riemannian manifolds. *Journal of Approximation Theory*, 148(2):111–127, 2007.
- [21] Syed Ahmad Aidil Adha Said Mad Zain and Md Yushalify Misro. Shape analysis and fairness metric of generalized fractional bézier curve. *Computational and Applied Mathematics*, 41(6):276, 2022.
- [22] Syed Ahmad Aidil Adha Said Mad Zain, Md Yushalify Misro, and Kenjiro T Miura. Generalized fractional bézier curve with shape parameters. *Mathematics*, 9(17):2141, 2021.
- [23] Dan Verständig. Code as you are?—über kreative praktiken des codings und deren bedeutung für subjektivierungsprozesse. *Praxistheoretische Perspektiven in der Medienpädagogik*, pages 87–110, 2020.