Universidad Nacional de Colombia - Sede Medellín

Métodos Geométricos en Visualización Médica

Tesis presentada por Johan Heriberto Rúa Muñoz
para obtener el grado de Magister en Ciencias - Matemática Aplicada

Director
Marco Paluszny Kluczynsky

Facultad de Ciencias

Escuela de Matemáticas

2014

Índice general

1. Preliminares **4**
1.1. Curvas de Bézier 4
1.2. Curvas B-spline 6
1.2.1. Curvas B-spline Polinómicas 6
1.2.2. El Algoritmo de De Boor 7
1.2.3. Derivadas de las curvas B-spline 10
1.3. Superficies 13
1.3.1. Superficies de Bézier y B-spline 14

2. Superficies B-spline Desarrollables **16**
2.1. Superficies Regladas 16
2.1.1. Superficies Desarrollables 17

3. Aplicaciones **27**
3.1. Formato DICOM 27
3.2. Rodajas Curvas Construidas con Superficies B-spline Desarrollables 29
3.2.1. Volumen 29
3.2.2. Algoritmo 30

Conclusiones **33**

Códigos Matlab **34**
3.3. Polinomios de Bernstein 34
3.4. Curva de Bézier 35
3.5. Algoritmo de De Boor 36
3.6. Curva B-spline 39
3.7. Superficies de Bézier 46
3.8. Curva Generatriz Base 48
3.9. Superficies B-spline Desarrollables 49

Bibliografía **51**

Introducción

Las superficies desarrollables son un tipo especial de superficies regladas, tienen la propiedad que su curvatura gaussiana es nula y pueden ser aplicadas en el plano isométricamente, es decir sin distorsión. Esta propiedad es importante, por ejemplo en la fabricación de piezas curvas a partir de hojas planas por medio de un proceso sin estiramiento. Generalmente, se han empleado para diseñar piezas curvas de acero en la industria naval y, son de utilidad en la confección textil y de calzado [7]. Además son relevantes en el diseño arquitectónico [8].

El objetivo principal de este trabajo es desarrollar un método para construir superficies desarrollables que permitan la extracción de información a partir de volúmenes a lo largo de rodajas curvas. Una rodaja curva es una superficie desarrollable que yace en el interior del volumen. La rodaja se texturiza con la información del volumen y se utiliza la propiedad de desarrollabilidad para su despliegue isométrico en el plano.
El método considerado fue propuesto por Fernández Jambrina [9] para superficies con una curva base B-spline de grado arbitrario, sin embargo, para las aplicaciones tomaremos como base una curva B-spline cúbica.

La tesis consta de tres capítulos. En el primero se definen los conceptos necesarios para la representación de curvas y superficies de Bézier y B-splines.

En el capítulo dos se caracterizan las superficies desarrollables, se enuncia el resultado obtenido por Aumann [1] para el diseño de superficies desarrollables de Bézier y se describe una generalización de este resultado para superficies desarrollables tipo B-spline estudiado por Fernández Jambrina [9].

En el capítulo tres se estudian algunos conceptos de imágenes médicas. El enfoque es el despliegue de información médica en el área odontológica. El uso del software especializado *Matlab* para el manejo de volúmenes será la herramienta informática primaria.

Por último se anexan los códigos diseñados en *Matlab* para la implementación del método y las gráficas obtenidas a partir de tomografías de maxilares humanos. Los códigos que corresponden a las curvas y superficies desarrollables B-spline son válidos para curvas base de grado arbitrario.

En nuestro trabajo, consideramos la utilidad de las superficies B-spline descritas por Fernández Jambrina [9] en el contexto de visualización en el área odontológica.

Capítulo 1

Preliminares

En este capítulo se definen los conceptos necesarios sobre curvas y superficies para el desarrollo de los capítulos siguientes. En particular nos interesan las superficies desarrollables de Bézier y B-spline. En los textos [5], [6] se desarrolla detalladamente estas curvas y superficies.

1.1. Curvas de Bézier

Las curvas de Bézier polinómicas deben su popularidad a la facilidad de manipulación de su forma geométrica, es de notar que sólo se requiere conocimientos básicos de geometría, álgebra lineal y cálculo para su manejo. El desarrollo original de las curvas de Bézier se llevó a cabo en la industria del automóvil durante el período 1958-1960 por los franceses, Pierre Bézier en Renault y Paul de Casteljau en Citröen. El debate del desarrollo histórico de las curvas de Bézier se puede encontrar en [2], [3] y [10].
Una curva de Bézier de grado n se especifica por una secuencia de $n+1$ puntos que se llaman *puntos de control*. El polígono obtenido al unir los puntos de control con segmentos de línea en el orden prescrito se denomina el *polígono de control*.

Definición 1.1 *Dados $n+1$ puntos de control $\boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ la curva de Bézier de grado n se define como*

$$\boldsymbol{b}(t) = \sum_{i=0}^{n} \boldsymbol{b}_i B_i^n(t)$$

donde

$$B_i^n(t) = \begin{cases} \frac{n!}{i!(n-i)!}(1-t)^{n-i}t^i & 0 \leq i \leq n \\ 0 & \textit{en otro caso.} \end{cases}$$

A los $B_i^n(t)$ se les llama polinomios de Bernstein ó funciones básicas de Bernstein.

La figura 1.1 muestra las familias de los polinomios de Bernstein de grado dos y tres. La figura 1.2 ilustra una curva de Bézier de grado 2 con sus puntos de control y su polígono de control.

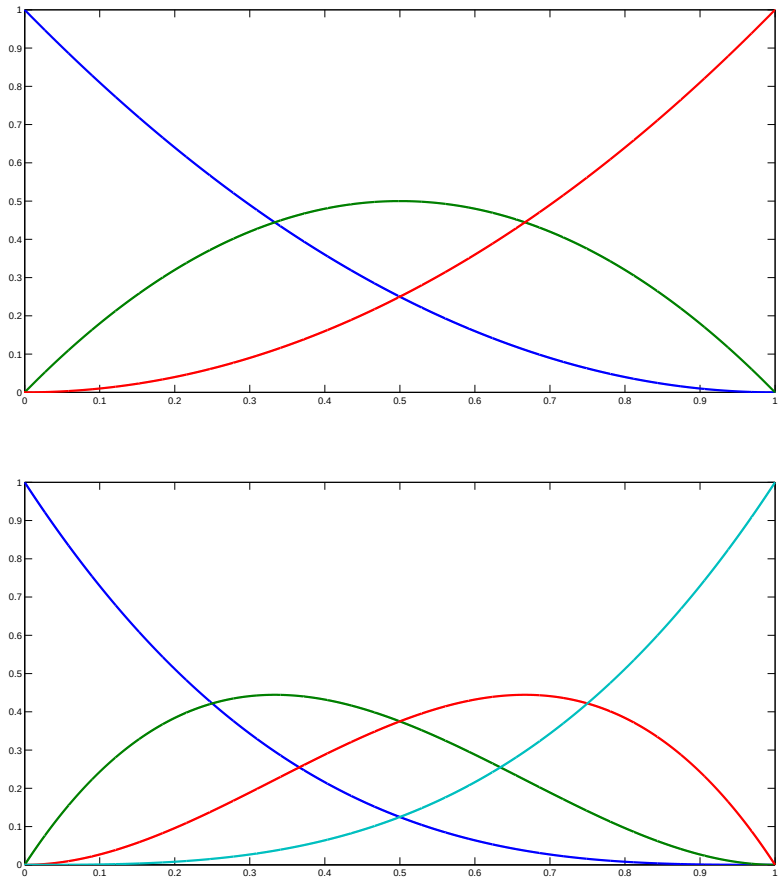

Figura 1.1 Polinomios de Bernstein de grados dos y tres.

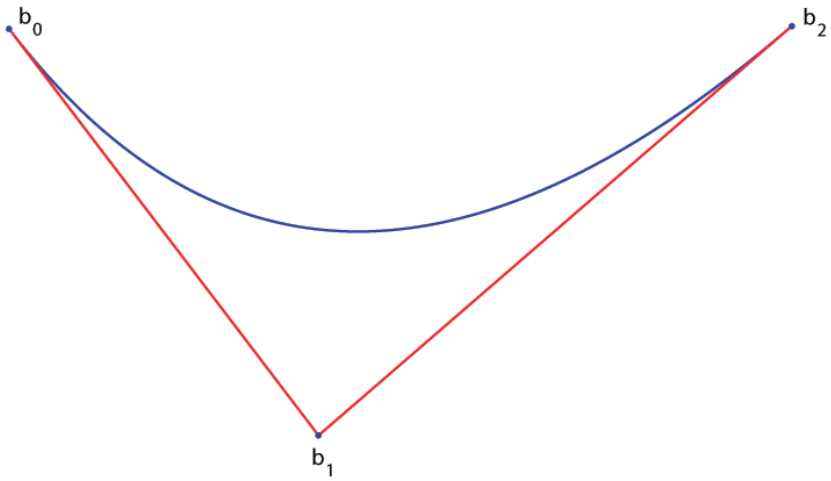

Figura 1.2 Curva cuadrática de Bézier con su polígono de control.

1.2. Curvas B-spline

Una curva B-spline es una curva polinómica a tramos y los tramos se conectan con cierto grado de diferenciabilidad. Los B-splines tienen propiedades análogas a las curvas de Bézier, teniendo a estas últimas como un caso particular. El desarrollo de los B-splines comienza con la publicación en 1946 del artículo de Schoenberg [15].

1.2.1. Curvas B-spline Polinómicas

Una curva B-spline definida en el intervalo $[a, b]$ se especifica con la siguiente información:

1. El *grado* d (*orden* $d+1$), de modo que cada segmento de la curva polinómica a tramos tiene grado d ó menos.

2. Una secuencia no decreciente de $m+1$ números reales $t_0, t_1, \ldots, t_m$, llamada el vector de nodos, de tal manera que $t_i \leq t_{i+1} \quad (i = 0, \ldots, m-1), t_d = a$ y $t_{m-d} = b$. Los nodos $t_0, t_1, \ldots, t_d$ y $t_{m-d}, t_{m-d+1}, \ldots, t_m$ se llaman nodos extremos, y los nodos $t_{d+1}, t_{d+2}, \ldots,\ t_{m-d-1}$ son llamados nodos interiores.

3. *Puntos de control* $\mathbf{b}_0, \ldots, \mathbf{b}_n$ con $n = m - (d+1)$.

Una curva B-spline se define en términos de las funciones básicas B-spline.

Definición 1.2 *Las funciones básicas B-spline de grado d, denotadas por $N_i^d(t)$, definidas por el vector de nodos $t_0, t_1, \ldots, t_m$ se definen de forma recursiva de la siguiente manera:*

$$N_i^0(t) = \begin{cases} 1, & \text{si} \qquad t \in [t_i, t_{i+1}) \\ 0, & \text{en otro caso.} \end{cases}$$

$$N_i^d(t) = \frac{t - t_i}{t_{i+d} - t_i} N_i^{d-1}(t) + \frac{t_{i+d+1} - t}{t_{i+d+1} - t_{i+1}} N_{i+1}^{d-1}(t),$$

para $i = 0, \ldots, n$ y $d \geq 1$.
Si el vector de nodos contiene valores de nodos repetidos, una división de la forma $N_i^{d-1}(t)/(t_{i+d} - t_i) = 0/0$ (para algún i) se pueden encontrar durante la recursión. Cada vez que esto ocurra, se asume que $0/0 = 0$.

Definición 1.3 *La curva B-spline de grado d (orden $d+1$) con los puntos de control $\boldsymbol{b}_0, \ldots, \boldsymbol{b}_n$ y nodos $t_0, \ldots, t_m$ se define en el intervalo $[a,b] = [t_d, t_{m-d}]$ por*

$$\mathbf{b}(t) = \sum_{i=0}^{n} \mathbf{b}_i N_i^d(t),$$

donde $N_i^d(t)$ son las funciones básicas B-spline de grado d.
La figura 1.3 es un ejemplo de una curva B-spline cúbica.

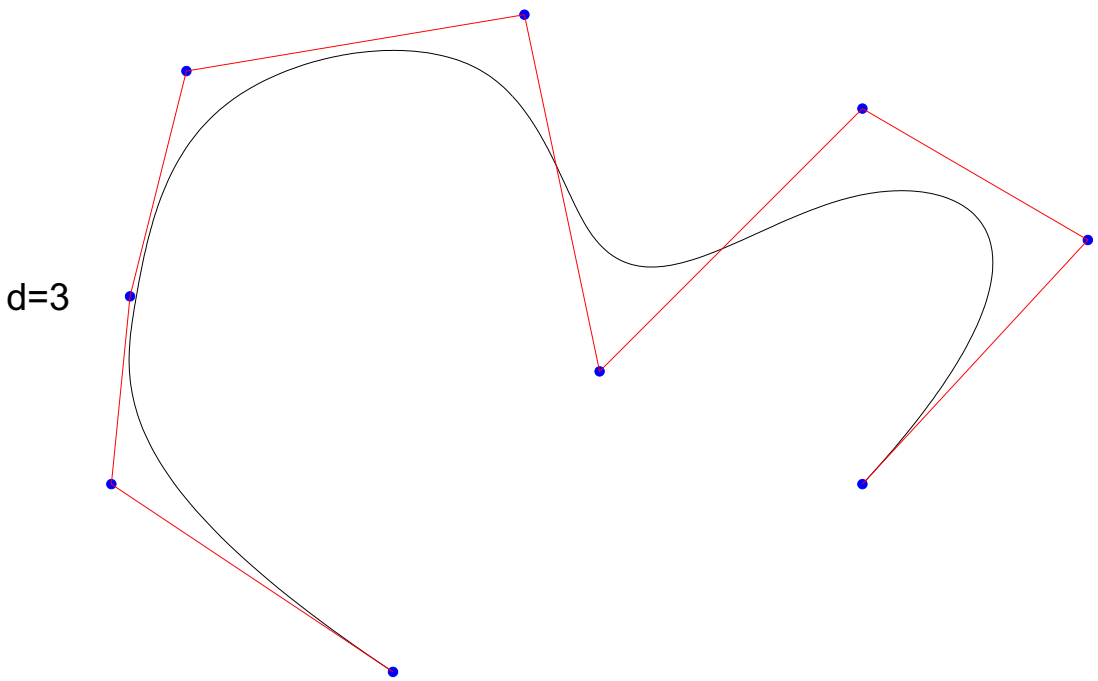

Figura 1.3 Ejemplo de una curva B-spline cúbica, con 9 puntos de control y secuencia de nodos 0,0,0,0,0.5,1,1.5,2,3,4,4,4,4.

1.2.2. El Algoritmo de De Boor

La evaluación de un punto de una curva B-spline se puede realizar usando un método conocido como el algoritmo de De Boor. Éste hace uso de la propiedad de las curvas B-spline llamada **control local** que establece: Si $\mathbf{b}(t)$ es una curva B-spline de grado d, cada segmento está determinado por $d+1$ puntos de control. Así, si $t \in [t_r, t_{r+1})$ con $r \in [d, m-d-1]$, tenemos que

$$\mathbf{b}(t) = \sum_{i=r-d}^{r} \mathbf{b}_i N_i^d(t).$$

Por lo tanto para evaluar $\mathbf{b}$(t) es suficiente evaluar $N_{r-d}^d(t), \ldots, N_r^d(t)$.
El algoritmo de De Boor se deduce entonces de la propiedad anterior y de la propiedad

de recursión de las funciones básicas B-spline. Supongamos que $t \in [t_r, t_{r+1})$ entonces

$$N_i^k(t) = \frac{t - t_i}{t_{i+k} - t_i} N_i^{k-1}(t) + \frac{t_{i+k+1} - t}{t_{i+k+1} - t_{i+1}} N_{i+1}^{k-1}(t).$$

lo anterior implica

$$\begin{aligned} \mathbf{b}(t) &= \sum_{i=r-k}^{r} \mathbf{b}_i N_i^k(t) \\ &= \sum_{i=r-k}^{r} \mathbf{b}_i \frac{t - t_i}{t_{i+k} - t_i} N_i^{k-1}(t) + \sum_{i=r-k}^{r} \mathbf{b}_i \frac{t_{i+k+1} - t}{t_{i+k+1} - t_{i+1}} N_{i+1}^{k-1}(t). \end{aligned}$$

Tomando $j = i + 1$ en la segunda sumatoria tenemos

$$\mathbf{b}(t) = \sum_{i=r-k}^{r} \mathbf{b}_i \frac{t - t_i}{t_{i+k} - t_i} N_i^{k-1}(t) + \sum_{j=r-k+1}^{r+1} \mathbf{b}_{j-1} \frac{t_{j+k} - t}{t_{j+k} - t_j} N_j^{k-1}(t).$$

Como $N_{r+1}^{k-1}(t) = N_{r-k}^{k-1}(t) = 0 \quad en \quad [t_r, t_{r+1})$,

$$\mathbf{b}(t) = \sum_{i=r-k+1}^{r} \left(\mathbf{b}_i \frac{t - t_i}{t_{i+k} - t_i} + \mathbf{b}_{i-1} \frac{t_{i+k} - t}{t_{i+k} - t_i} \right) N_i^{k-1}(t).$$

Sea

$$\mathbf{b}_i^1 = \mathbf{b}_{i-1} \frac{t_{i+k} - t}{t_{i+k} - t_i} + \mathbf{b}_i \frac{t - t_i}{t_{i+k} - t_i},$$

para $i = r - k + 1, \ldots, r$. De una manera similar, $N_i^{k-1}(t)$ se puede expresar en términos de las funciones básicas de grado $k - 2$ y así sucesivamente. Para una curva de grado d, el resultado es un procedimiento recursivo

$$\begin{aligned} \mathbf{b}_i^j(t) &= (1 - \alpha_i^j(t)) \mathbf{b}_{i-1}^{j-1}(t) + \alpha_i^j(t) \mathbf{b}_i^{j-1}(t), \\ \alpha_i^j(t) &= \frac{t - t_i}{t_{i+d-j+1} - t_i}, \end{aligned}$$

para $j = 1, \ldots, d, \quad i = r - d + j, \ldots, r$, donde $\mathbf{b}_i^0(t) = \mathbf{b}_i$, $\mathbf{b}_{-1} = 0$, y $\mathbf{b}_{m-d+1} = 0$. El j-ésimo paso expresa $\mathbf{b}$(t) en términos de las funciones básicas de grado $d - j$

$$\mathbf{b}(t) = \sum_{i=r-d+j}^{r} \mathbf{b}_i^j N_i^{d-j}(t).$$

Así, cuando $j = d$, el algoritmo determina el punto

$$\mathbf{b}(t) = \sum_{i=r}^{r} \mathbf{b}_i^d N_i^0(t) = \mathbf{b}_r^d$$

en la curva.
En resumen, para un determinado valor de t, el algoritmo de De Boor, genera una matriz triangular de puntos tales que $\mathbf{b}_r^d = \mathbf{b}(t)$.

$$\begin{array}{lllll} \mathbf{b}_{r-d}^0 & \mathbf{b}_{r-d+1}^0 & \dots & \dots & \mathbf{b}_r^0 \\ \mathbf{b}_{r-d+1}^1 & \dots & \dots & \mathbf{b}_r^1 & \\ \vdots & \dots & & & \\ \mathbf{b}_{r-1}^{d-1} & \mathbf{b}_r^{d-1} & & & \\ \mathbf{b}_r^d = & \mathbf{b}(t). & & & \end{array}$$

Las figuras 1.4 y 1.5 presentan ejemplos de curvas B-splines para diferentes conjuntos de puntos de control y nodos (ver al final del documento, donde se listan los códigos que fueron desarrollados para ilustrar los ejemplos de esta tesis).

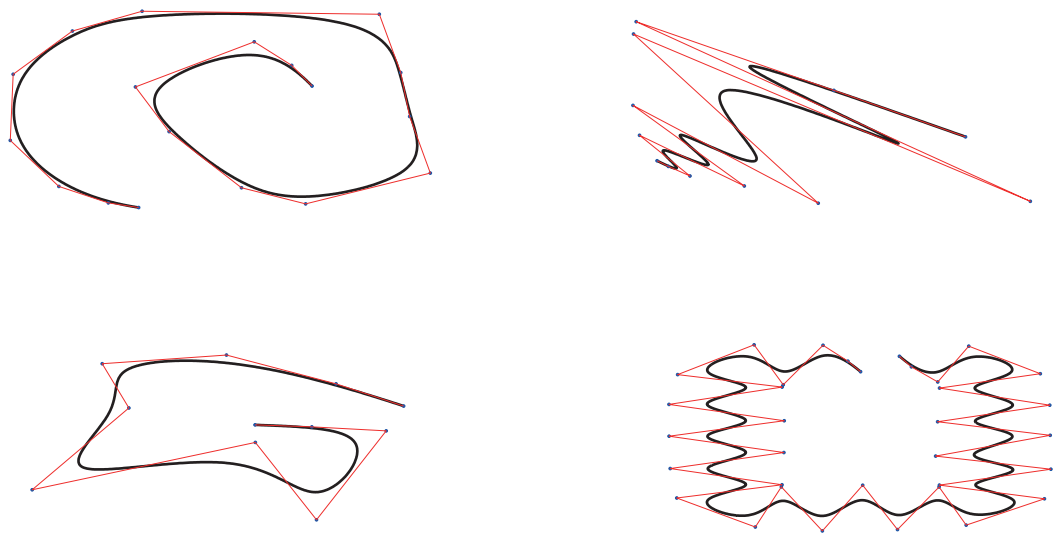

Figura 1.4 Curvas B-splines interpolantes [1] *con diferentes secuencias de nodos y grados.*

[1]Interpolante significa que el B-spline empieza en el primer punto de control $\mathbf{b}_0$ y termina en $\mathbf{b}_n$.

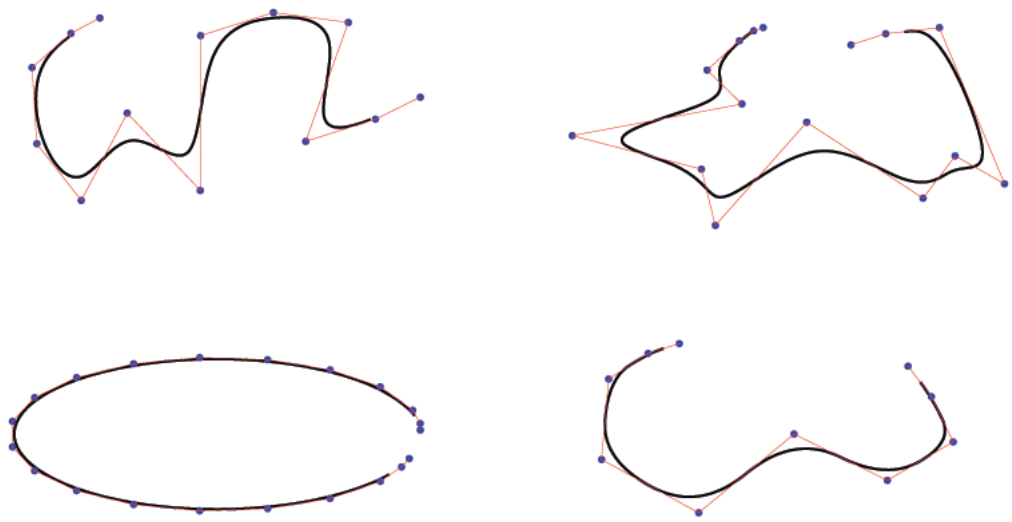

Figura 1.5 Curvas B-splines no interpolantes con diferentes secuencias de nodos y grados.

1.2.3. Derivadas de las curvas B-spline

Para determinar la derivada de una curva B-spline de grado d como un B-spline de grado *d-1*, lo primero es determinar las derivadas de las funciones básicas de grado d en términos de las funciones básicas de grado *d-1*.

Teorema 1.1 *La derivada de las funciones básicas B-spline de $N_i^d(t)$ de grado d se puede obtener en términos de las funciones básicas de grado d-1 de la siguiente manera:*

$$\frac{dN_i^d}{dt}(t) = \frac{d}{t_{i+d} - t_i} N_i^{d-1}(t) - \frac{d}{t_{i+d+1} - t_{i+1}} N_{i+1}^{d-1}(t).$$

Teorema 1.2 *La derivada de $\boldsymbol{b}(t) = \sum_{i=0}^{n} \boldsymbol{b}_i N_i^d(t)$ es*

$$\boldsymbol{b}'(t) = \sum_{i=0}^{n-1} \boldsymbol{b}_i^{(1)} N_i^{d-1}(t).$$

donde

$$\boldsymbol{b}_i^{(1)} = d \frac{\boldsymbol{b}_{i+1} - \boldsymbol{b}_i}{t_{i+d+1} - t_{i+1}},$$

y $N_i^{d-1}(t)$ son las funciones básicas de grado $d-1$.

Si consideramos la curva con $d = 3$, nodos $t_0 = 1.2, t_1 = 1.4, t_2 = 1.5, t_3 = 2.0, t_4 = 2.4, t_5 = 3.1, t_6 = 5.0, t_7 = 6.4, t_8 = 7.3$ y puntos de control $\mathbf{b}_0 = (2,1), \mathbf{b}_1 = (4,8), \mathbf{b}_2 = (5,-1), \mathbf{b}_3 = (3,-2), \mathbf{b}_4 = (2,-4)$. Tenemos que:

$$\mathbf{b}(t) = \sum_{i=0}^{4} \mathbf{b}_i N_i^3(t), \quad \text{con} \quad N_i^3(t) = \frac{t - t_i}{t_{i+3} - t_i} N_i^2(t) + \frac{t_{i+4} - t}{t_{i+4} - t_{i+1}} N_{i+1}^2(t)$$

Además,

$$\mathbf{b}'(t) = \sum_{i=0}^{3} \mathbf{b}_i^{(1)} N_i^2(t), \quad \text{con} \quad N_i^2(t) = \frac{t - t_i}{t_{i+2} - t_i} N_i^2(t) + \frac{t_{i+3} - t}{t_{i+3} - t_{i+1}} N_{i+1}^2(t)$$

donde

$$\begin{aligned}
\mathbf{b}_0^{(1)} &= 3\frac{\mathbf{b}_1 - \mathbf{b}_0}{t_4 - t_1} = 3\frac{(4,8) - (2,1)}{1} = (6,21) \\
\mathbf{b}_1^{(1)} &= 3\frac{\mathbf{b}_2 - \mathbf{b}_1}{t_5 - t_2} = 3\frac{(5,-1) - (4,8)}{1.6} = (1.875, -16.875) \\
\mathbf{b}_2^{(1)} &= 3\frac{\mathbf{b}_3 - \mathbf{b}_2}{t_6 - t_3} = 3\frac{(3,-2) - (5,-1)}{3} = (-2,-1) \\
\mathbf{b}_3^{(1)} &= 3\frac{\mathbf{b}_4 - \mathbf{b}_3}{t_7 - t_4} = 3\frac{(2,-4) - (3,-2)}{4} = (-0.75, -1.5).
\end{aligned}$$

En las figuras del 1.6 al 1.8 se ilustran las funciones básicas, la curva y sus vectores tangentes.

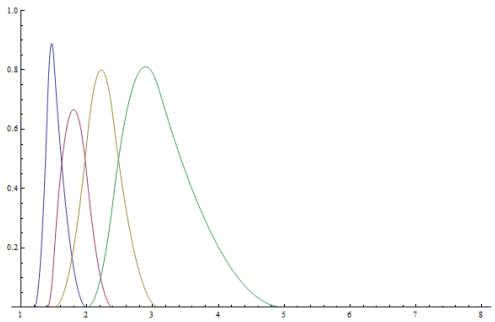

Figura 1.6 Funciones básicas $N_i^2(t), i = 0, \ldots, 3$.

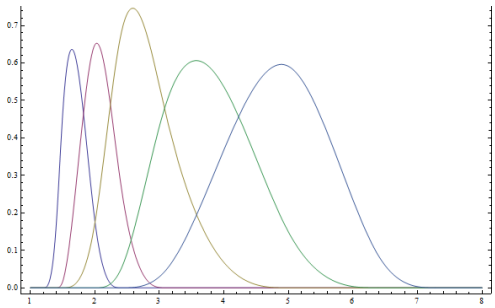

Figura 1.7 Funciones básicas $N_i^3(t), i = 0, \ldots, 4$.

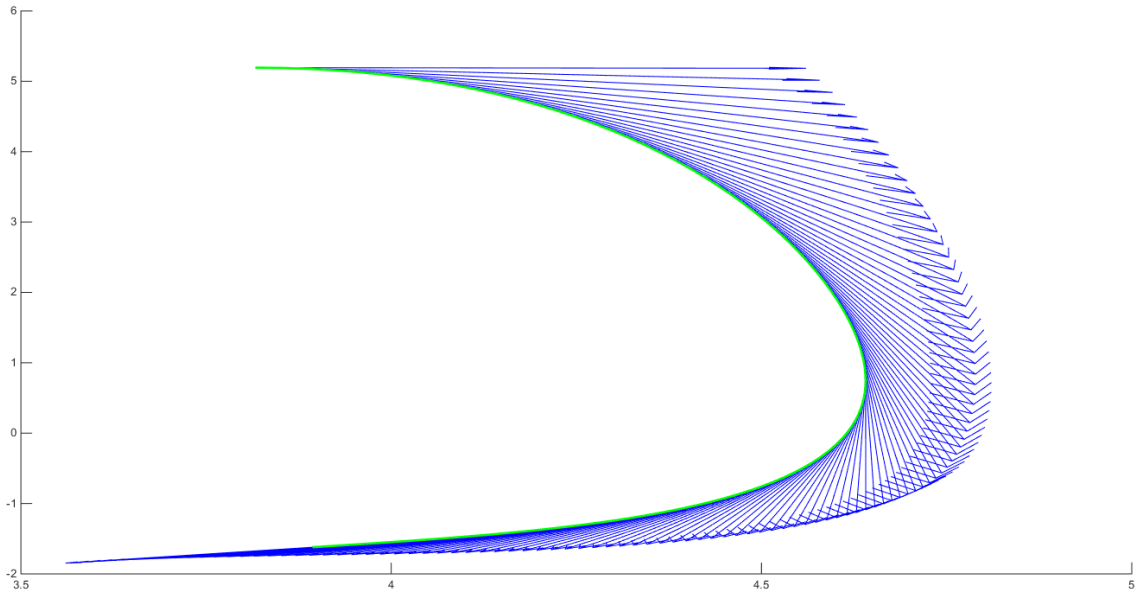

Figura 1.8 Curva B-spline del ejemplo y sus respectivos vectores tangentes.

1.3. Superficies

Las superficies tienen un papel fundamental en aplicaciones tales como los gráficos por computador, realidad virtual, juegos, y en el diseño asistido por computador. Las curvas conducen naturalmente al estudio de superficies. En particular las curvas de Bézier y las curvas B-spline tienen formas correspondientes de superficies.

Definición 1.4 *Sea U un subconjunto abierto de $\mathbb{R}^2$. Una superficie paramétrica es una función diferenciable $\mathbf{S} : U \subset \mathbb{R}^2 \to \mathbb{R}^3$. El conjunto $S = \mathbf{S}(U) \subset \mathbb{R}^3$ es llamado la superficie S o la traza de $\mathbf{S}$, y se dice que $\boldsymbol{S}$ parametriza S.*
Si V es un subconjunto cerrado de $\mathbb{R}^2$, decimos que $\boldsymbol{S} : V \to \mathbb{R}^3$ es una superficie paramétrica si existe un subconjunto abierto U tal que $V \subset U$, y una superficie paramétrica $\boldsymbol{S}_1 : U \to \mathbb{R}^3$ que satisface $\boldsymbol{S}(s,t) = \boldsymbol{S}_1(s,t)$ para todo $(s,t) \in V$.

Las coordenadas de un punto arbitrario de una superficie paramétrica S se expresan por medio de funciones de dos variables, por ejemplo,

$$\mathbf{S}(s,t) = (x(s,t), y(s,t), z(s,t)).$$

Las curvas $\mathbf{S}(s,t_0)$ y $\mathbf{S}(s_0,t)$, obtenidas mediante la fijación del valor de una de las variables, se llaman *s-curva* y *t-curva*, respectivamente.

La superficie paramétrica $\mathbf{S} : U \to \mathbb{R}^3$ se dice que es C^k-continua (o simplemente C^k) siempre que las funciones de coordenadas $x(s,t)$, $y(s,t)$ y $z(s,t)$ sean C^k-continuas en U. Si $|\mathbf{S}_s(s,t) \times \mathbf{S}_t(s,t)| \neq 0$, entonces la superficie se dice *regular* en (s,t), y $\mathbf{S}(s,t)$ se dice que es un *punto regular*. Si $\mathbf{S}(s,t)$ es regular para todo $(s,t) \in U$, entonces la superficie se dice *regular*. Si $|\mathbf{S}_s(s,t) \times \mathbf{S}_t(s,t)| = 0$, entonces $\mathbf{S}$ se dice que es *singular* en (s,t) y $\mathbf{S}(s,t)$ se denomina *punto singular*.

Una superficie paramétrica $\mathbf{S}$ definida en un conjunto cerrado V se dice que es C^k si existe un conjunto abierto U que contiene V, y una superficie paramétrica $\mathbf{S}_1$ continua C^k, definida en U, tal que $\mathbf{S}(s,t) = \mathbf{S}_1(s,t)$ para todo $(s,t) \in V$. Las derivadas parciales de $\mathbf{S}(s,t)$ en los puntos frontera de V se obtienen mediante la adopción de las derivadas de la asignación de extensión. Entonces $\mathbf{S}(s,t)$ es un punto ordinario (singular) si éste es un punto regular (singular) de $\mathbf{S}_1(s,t)$.
En un punto $\mathbf{p} = \mathbf{S}(s,t)$, tenemos que $\mathbf{S}_s(s,t)$ y $\mathbf{S}_t(s,t)$ son los vectores tangentes a la s-curva y a la t-curva, respectivamente. Si $\mathbf{p}$ es un punto regular de la superficie entonces $|\mathbf{S}_s(s,t) \times \mathbf{S}_t(s,t)| \neq 0$. Por lo tanto $\mathbf{S}_s(s,t)$ y $\mathbf{S}_t(s,t)$ son vectores linealmente indepen-

dientes, y un vector perpendicular a los dos es el *vector unitario normal* a la superficie, dado por

$$\mathbf{n}(s,t) = \frac{\mathbf{S}_s(s,t) \times \mathbf{S}_t(s,t)}{|\mathbf{S}_s(s,t) \times \mathbf{S}_t(s,t)|}.$$

1.3.1. Superficies de Bézier y B-spline

Sean $B_i^n(s)$ y $B_j^p(t)$ las funciones básicas de Bernstein de grado n y p en las variables s y t, respectivamente. Una superficie de Bézier con puntos de control $\mathbf{p}_{ij}$ $(0 \leq i \leq n, 0 \leq j \leq p)$ es la superficie paramétrica definida por

$$\mathbf{S}(s,t) = \sum_{i=0}^{n} \sum_{j=0}^{p} \mathbf{p}_{ij} B_i^n(s) B_j^p(t), \quad \text{para} \quad (s,t) \in [0,1] \times [0,1].$$

Las curvas parámetricas de una superficie de Bézier son curvas de Bézier. En particular, las s-curvas $\mathbf{S}(s,0), \mathbf{S}(s,1)$, y las t-curvas $\mathbf{S}(0,t), \mathbf{S}(1,t)$, son las curvas de Bézier que forman los cuatro bordes de la superficie.
Consideremos a continuacón las superficies B-spline. Sean $N_i^d(s)$ las funciones básicas B-spline de grado d con vector de nodos $s_0, s_1, \ldots, s_m$, y sean $N_j^e(t)$ las funciones básicas B-spline de grado e con vector de nodos $t_0, t_1, \ldots, t_q$. Una superficie B-spline con puntos de control $\mathbf{b}_{ij}$, con $0 \leq i \leq n = m - (d+1), \quad 0 \leq j \leq p = q - (e+1)$, se define por

$$\mathbf{S}(s,t) = \sum_{i=0}^{n} \sum_{j=0}^{p} \mathbf{b}_{ij} N_i^d(s) N_j^e(t), \quad \text{para} \quad (s,t) \in [s_d, s_{m-d}] \times [t_e, t_{q-e}].$$

Análogamente a las superfices de Bézier, las curvas parámetricas de una superficie B-spline son curvas B-spline. Así, las s-curvas $\mathbf{S}(s,t_e), \mathbf{S}(s,t_{q-e})$, y las t-curvas $\mathbf{S}(s_d,t), \mathbf{S}(s_{m-d},t)$ son las cuatro curvas borde de la superficie.

La figura 1.5 ilustra varios ejemplos de superficies de Bézier, sus polígonos de control están descritos al final del documento.

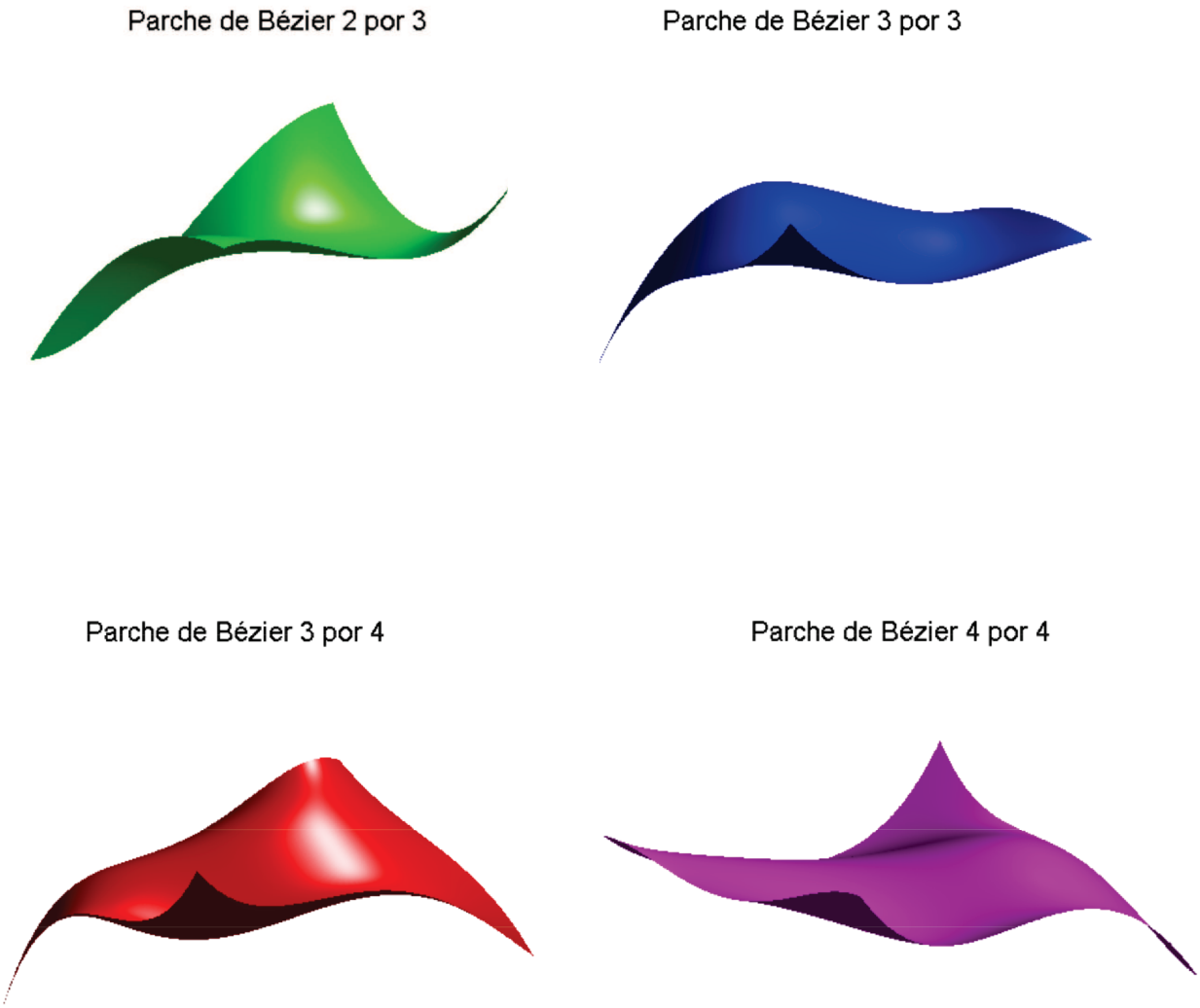

Figura 1.5 Ejemplos de superficies de Bézier.

La figura 1.6 muestra ejemplos de superficies B-spline [2]

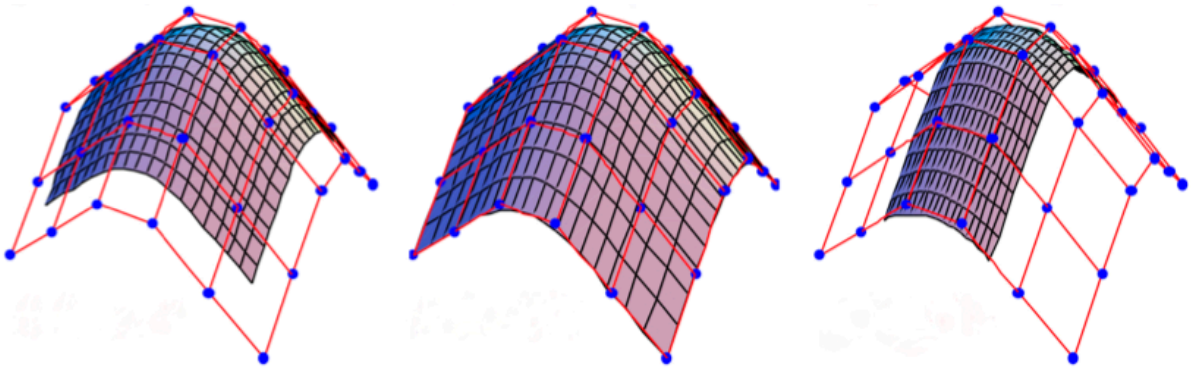

Figura 1.6 Ejemplos básicos de superficies B-spline.

[2] http://ocw.unican.es/ensenanzas-tecnicas/visualizacion-e-interaccion-grafica/material-de-clase-2/02-Bsplines.pdf

Capítulo 2

Superficies B-spline Desarrollables

Las superficies desarrollables son el subconjunto de las superficies regladas que pueden ser aplicadas isométricamente al plano. Debido a esta propiedad han sido objeto de estudio durante los últimos años para el diseño de diversas aplicaciones. Con el propósito de modelar superficies de interés en el contexto médico, hemos estudiado el algoritmo descrito por Fernández Jambrina [9] para la construcción de superficies B-spline desarrollables de grado arbitrario.

2.1. Superficies Regladas

Una *superficie reglada* es una superficie generada por el movimiento en el espacio de un segmento de recta llamado *generador*. Una curva que pasa a través de todos los generadores de la superficie es llamada una *directriz*. Lo anterior se expresa formalmente en la siguiente definición.

Definición 2.1 *Sean $I \subset \mathbb{R}$ un intervalo, $\mathbf{f} : I \to \mathbb{R}^3$ una curva paramétrica regular y $\mathbf{g} : I \to \mathbb{R}^3$ una función diferenciable con $\mathbf{g}(u) \neq 0$ para todo $u \in I$. Definimos una superficie reglada por*

$$\boldsymbol{S}(u, v) = \mathbf{f}(u) + v\mathbf{g}(u), \quad u \in I, v \in \mathbb{R},$$

donde $\mathbf{f}$ es una directriz o curva base de la superficie reglada y $\mathbf{g}(u)$ es un vector unitario, el cual da la dirección del generador en cada punto de la directriz.

Una superficie reglada también puede ser representada como los generadores que unen dos curvas espaciales de la siguiente forma:

$$\mathbf{S}(u, v) = (1 - v)\mathbf{c}(u) + v\mathbf{d}(u) \quad 0 \leq u, v \leq 1$$

donde $\mathbf{c}(u)$ y $\mathbf{d}(u)$ son directrices de la superficie. Estas dos representaciones son coiciden si se toma $\mathbf{f}(u) = \mathbf{c}(u)$ y la dirección de cada generador en u es $\mathbf{g}(u) = \mathbf{d}(u) - \mathbf{c}(u)$. La figura 2.3 muestra dos ejemplos de superficies regladas .

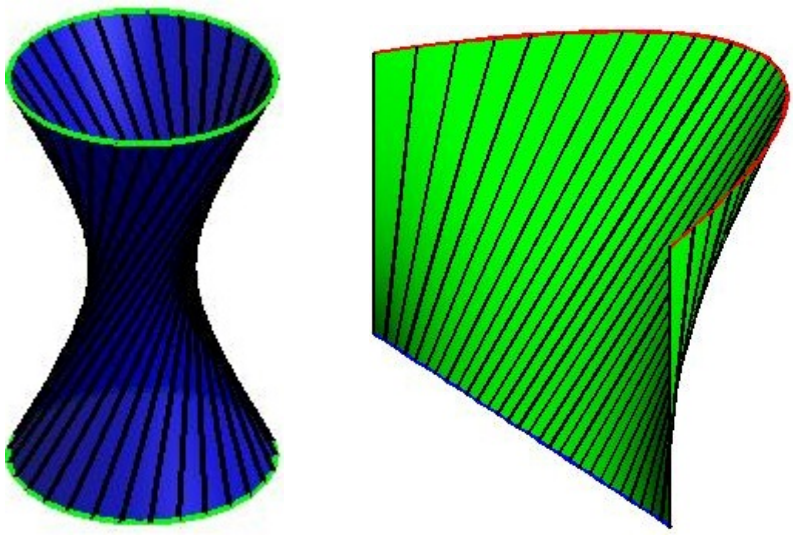

Figura 2.3 Superficies regladas: un hiperboloide de una hoja y un conoide [1]

2.1.1. Superficies Desarrollables

Definición 2.2 *Sea* $\mathbf{S} : U \to \mathbb{R}^3$ *una superficie paramétrica y sea* $\mathbf{p} = \mathbf{S}(u_0, v_0)$. *El subespacio vectorial generado por* $\mathbf{S}_u(u_0, v_0)$ *y* $\mathbf{S}_v(u_0, v_0)$ *es denominado plano tangente de* $\mathbf{S}$ *en* $\mathbf{p}$.

Una *superficie es desarrollable* si es una superficie reglada y posee el mismo plano tangente a lo largo de cada generador.

Teorema 2.1 *Una superficie* $\boldsymbol{S}$ *es desarrollable si y sólo si*

$$[(\mathbf{d}(u) - \mathbf{c}(u)) \times \mathbf{c}'(u)] \cdot \mathbf{d}'(u) = 0.$$

Prueba

Supongamos que $\mathbf{S}$ es una superficie desarrollable y tomemos dos puntos sobre un mismo generador evaluados en los valores del parámetro (u_0, v_1) y (u_0, v_2), con $v_1 \neq v_2$. La superficie es desarrollable si los planos tangentes en estos puntos coinciden. Los planos

[1] Un conoide es una superficie reglada cuyos generadores son paralelos a un plano director e intersectan una recta fija.

tangentes están definidos por los vectores $\mathbf{S}_u$ y $\mathbf{S}_v$, así tenemos que

$$\begin{aligned}
\mathbf{S}_u(u_0, v_1) &= (1 - v_1)\mathbf{c}'(u_0) + v_1\mathbf{d}'(u_0) && (2.1)\\
\mathbf{S}_u(u_0, v_2) &= (1 - v_2)\mathbf{c}'(u_0) + v_2\mathbf{d}'(u_0) && (2.2)\\
\mathbf{S}_v(u_0, v_1) &= \mathbf{S}_v(u_0, v_2) = \mathbf{d}(u_0) - \mathbf{c}(u_0). && (2.3)
\end{aligned}$$

Si los planos tangentes en los dos puntos coinciden, los vectores $\mathbf{S}_u(u_0, v_1)$, $\mathbf{S}_u(u_0, v_2)$ y $\mathbf{d}(u_0) - \mathbf{c}(u_0)$ deben ser coplanares. Por tanto, el determinante de la matriz formada por ellos es nulo, así

$$|\mathbf{S}_u(u_0, v_1)\ \mathbf{S}_u(u_0, v_2)\ (\mathbf{d}(u_0) - \mathbf{c}(u_0))| = 0. \qquad (2.4)$$

Usando la notación $\mathbf{S}_u(u_0, v_1) = \mathbf{S}_{u1}$, $\mathbf{S}_u(u_0, v_2) = \mathbf{S}_{u2}$

$$\mathbf{S}_{u1} \times \mathbf{S}_{u2} = (\mathbf{S}_{u1,y}\mathbf{S}_{u2,z} - \mathbf{S}_{u1,z}\mathbf{S}_{u2,y}, -(\mathbf{S}_{u1,x}\mathbf{S}_{u2,z} - \mathbf{S}_{u1,z}\mathbf{S}_{u2,x}), \mathbf{S}_{u1,x}\mathbf{S}_{u2,y} - \mathbf{S}_{u1,y}\mathbf{S}_{u2,x}).$$

Utilizando (2.1) y (2.2) tenemos que

$$\begin{aligned}
\mathbf{S}_{u1,y}\mathbf{S}_{u2,z} - \mathbf{S}_{u1,z}\mathbf{S}_{u2,y} &= [(1 - v_1)\mathbf{c}'_y + v_1\mathbf{d}'_y][(1 - v_2)\mathbf{c}'_z + v_2\mathbf{d}'_z]\\
&- [(1 - v_1)\mathbf{c}'_z + v_1\mathbf{d}'_z][(1 - v_2)\mathbf{c}'_y + v_2\mathbf{d}'_y]\\
&= (1 - v_1)(1 - v_2)\mathbf{c}'_y\mathbf{c}'_z + v_1(1 - v_2)\mathbf{c}'_z\mathbf{d}'_y + v_2(1 - v_1)\mathbf{c}'_y\mathbf{d}'_z\\
&+ v_1v_2\mathbf{d}'_y\mathbf{d}'_z - [(1 - v_1)(1 - v_2)\mathbf{c}'_y\mathbf{c}'_z + v_1(1 - v_2)\mathbf{c}'_y\mathbf{d}'_z\\
&+ v_2(1 - v_1)\mathbf{c}'_z\mathbf{d}'_y + v_1v_2\mathbf{d}'_y\mathbf{d}'_z]\\
&= (v_1 - v_1v_2 - v_2 + v_1v_2)\mathbf{c}'_z\mathbf{d}'_y + (v_2 - v_1v_2 - v_1 + v_1v_2)\mathbf{c}'_y\mathbf{d}'_z\\
&= (v_2 - v_1)(\mathbf{c}'_y\mathbf{d}'_z - \mathbf{c}'_z\mathbf{d}'_y).
\end{aligned}$$

Similarmente,

$$\mathbf{S}_{u1,x}\mathbf{S}_{u2,z} - \mathbf{S}_{u1,z}\mathbf{S}_{u2,x} = (v_2 - v_1)(\mathbf{c}'_x\mathbf{d}'_z - \mathbf{c}'_z\mathbf{d}'_x)$$

y

$$\mathbf{S}_{u1,x}\mathbf{S}_{u2,y} - \mathbf{S}_{u1,y}\mathbf{S}_{u2,x} = (v_2 - v_1)(\mathbf{c}'_x\mathbf{d}'_y - \mathbf{c}'_y\mathbf{d}'_x).$$

Por tanto,

$$\mathbf{S}_{u1} \times \mathbf{S}_{u2} = (v_2 - v_1)(\mathbf{c}' \times \mathbf{d}')$$

y así (2.4) se reduce a

$$[(\mathbf{d}(u) - \mathbf{c}(u)) \times \mathbf{c}'(u)] \cdot \mathbf{d}'(u) \quad = \quad 0 \tag{2.5}$$

para todo u, $v_1 \neq v_2$.
Recíprocamente, si la superficie satisface (2.5). Para todo u, $\mathbf{c}'(u)$, $\mathbf{d}'(u)$, $\mathbf{d}(u) - \mathbf{c}(u)$ pertenecen al mismo plano, entonces $\mathbf{S}_u(u, v)$, $\mathbf{S}_v(u, v)$ generan el mismo plano para u y para todo v. Por lo tanto el plano tangente a la superficie es el mismo para todos los puntos en un mismo generador, es decir, la superficie es desarrollable.
Así, concluimos que una superficie $\mathbf{S}$ es desarrollable si y sólo si $\mathbf{S}$ cumple (2.5).

□

A continuación se describe la clasificación de las superficies desarrollables considerando $\mathbf{c}(u)$ y $\mathbf{d}(u)$ como curvas de Bézier o B-splines. Sea $\mathbf{b}(u) = \mathbf{d}(u) - \mathbf{c}(u)$ y

$$\mathbf{S}(u, v) = \mathbf{c}(u) + v\mathbf{b}(u)$$

(i) *Superficies planas*: $\mathbf{S}$ define un plano si $\mathbf{c}(u)$ es una curva plana y para todo u, $\mathbf{b}(u)$ es un vector que yace sobre este plano.

(ii) *Superficies cilíndricas*: $\mathbf{S}$ es un cilindro si todos sus generadores son paralelos, esto es, si $\mathbf{b}'(u) \| \mathbf{b}(u)$. En este caso el vector $\mathbf{b}(u)$ es paralelo a una dirección constante $\mathbf{v}$ para todo u,

$$\mathbf{b}(u) = \lambda(u)\mathbf{v}, \quad \mathbf{S}(u, v) = \mathbf{c}(u) + v\lambda(u)\mathbf{v}.$$

Si consideramos $\mathbf{c}(u)$, $\mathbf{d}(u)$ como curvas de Bézier o curvas B-splines con polígonos de control $\{\mathbf{c_0}, \ldots, \mathbf{c_n}\}$ y $\{\mathbf{d_0}, \ldots, \mathbf{d_n}\}$ respectivamente, entonces es necesario y suficiente que los vectores $\mathbf{b}_0, \ldots, \mathbf{b}_n$ con $\mathbf{b}_i = \mathbf{d}_i - \mathbf{c}_i \quad i = 0, \ldots, n$ sean paralelos a la dirección $\mathbf{v}$.

Figura 2.4 Superficie cilíndrica.

(iii) *Superficies cónicas*: $\mathbf{S}$ es un cono si todos sus generadores pasan por un punto común. Así,

$$\mathbf{S}(u,v) = \mathbf{a} + (v+k)\mathbf{b}(u), \quad k > 0.$$

Si $\mathbf{c}(u)$, $\mathbf{d}(u)$ son curvas de Bézier o B-splines, con polígonos de control $\{\mathbf{c_0}, \ldots, \mathbf{c_n}\}$ y $\{\mathbf{d_0}, \ldots, \mathbf{d_n}\}$ respectivamente, entonces sus respectivos polígonos de control satisfacen

$$r(\mathbf{c}_i - \mathbf{c}_{i-1}) = \mathbf{d}_i - \mathbf{d}_{i-1}, \quad i = 1, \ldots, n, \quad r = 1 + 1/k.$$

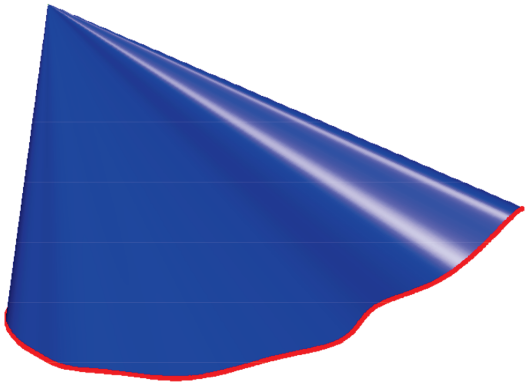

Figura 2.5 Superficie cónica.

(iv) *Superficies tangentes*: $\mathbf{S}$ define una superficie tangente si los generadores son las rectas tangentes de la curva $\mathbf{c}(u)$, por lo tanto

$$\mathbf{S}(u,v) = \mathbf{c}(u) + v\lambda(u)\mathbf{c}'(u).$$

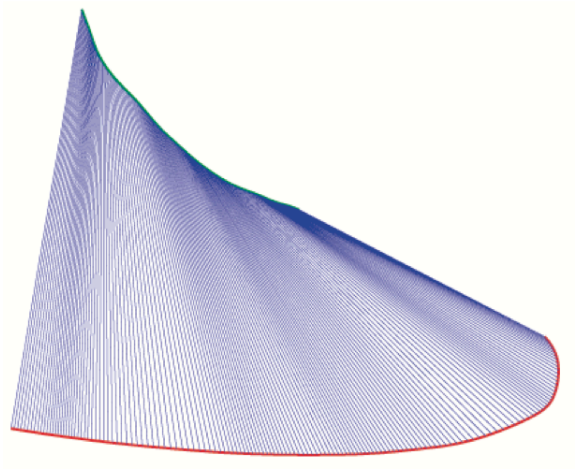

Figura 2.6 Superficie tangente.

A continuación consideramos el algoritmo de Aumann, el cual permite construir superficies desarrollables a partir de curvas de Bézier [1].

Teorema 2.2 *Sean λ, μ números reales, $\mathbf{d}_0$ un punto y $\mathbf{c}(u)$ una curva de Bézier de grado n con polígono de control $\{\mathbf{c_0}, \ldots, \mathbf{c_n}\}$. La superficie reglada $\mathbf{S}(u, v)$ dada por*

$$\mathbf{S}(u, v) = (1 - v)\mathbf{c}(u) + v\mathbf{d}(u) \quad 0 \leq u, v \leq 1 \tag{2.6}$$

es desarrollable si el polígono de control de la curva $\mathbf{d}(u)$ cumple la siguiente recursión:

$$\mathbf{d}_{i+1} = \mathbf{c}_i + \lambda(\mathbf{c}_{i+1} - \mathbf{c}_i) + \mu(\mathbf{d}_i - \mathbf{c}_i), \quad i = 0, 1, \ldots, n - 1. \tag{2.7}$$

- *La superficie $\mathbf{S}$ es un cilindro si y sólo si $\lambda = 1$.*
- *La superficie $\mathbf{S}$ es un cono si y sólo si $\lambda \neq 1$ y $\mu = 1$.*
- *La superficie $\mathbf{S}$ es una superficie tangente si y sólo si $\lambda \neq 1$ y $\mu \neq 1$.*

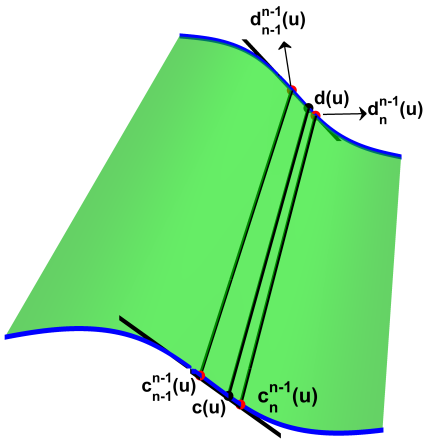

Figura 2.8 Interpretación de la condición de coplanaridad.

El resultado siguiente nos da condiciones necesarias y suficientes para que una superficie generada por dos curvas B-spline sea desarrollable.
Enunciamos el teorema en forma general, pero demostraremos los casos en los cuales las curvas son de uno y dos segmentos.

Proposición 2.1 *Sea $\mathbf{S}$ una superficie generada por dos curvas B-spline, $\mathbf{c}(u)$, $\mathbf{d}(u)$ de grado n con respectivos polígonos de control $\{\mathbf{c_0}, \ldots, \mathbf{c_L}\}$, $\{\mathbf{d_0}, \ldots, \mathbf{d_L}\}$ y secuencia de nodos $\{u_0, \ldots, u_K\}$, donde $K = L + n + 1$.*
$\mathbf{S}$ es una superficie desarrollable si y sólo para todo u, los cuatro penúltimos puntos $\mathbf{c}_{n-1}^{n-1}(u)$, $\mathbf{c}_n^{n-1}(u)$, $\mathbf{d}_{n-1}^{n-1}(u)$, $\mathbf{d}_n^{n-1}(u)$ de la evaluación del algoritmo de De Boor son coplanares.

Prueba

Supondremos por simplicidad que ambas curvas son B-splines que constan de un sólo un segmento y son de grado n, definidas por sus polígonos de control $\{\mathbf{c_0}, \ldots, \mathbf{c_n}\}$ y $\{\mathbf{d_0}, \ldots, \mathbf{d_n}\}$ y la misma secuencia de nodos $\{u_0, \ldots, u_{2n+1}\}$.

Sea $\mathbf{S}$ una superficie desarrollable, entonces:

Calculando $\mathbf{c}(u)$ y $\mathbf{d}(u)$, con $u \in [u_n, u_{n+1})$ con el algoritmo de De Boor tenemos que

$$\mathbf{c}_i^1 = \frac{u_{i+n} - u}{u_{i+n} - u_i}\mathbf{c}_{i-1} + \frac{u - u_i}{u_{i+n} - u_i}\mathbf{c}_i, \quad i = 1, \ldots, n,$$

$$\mathbf{c}_i^r = \frac{u_{i+n-r+1} - u}{u_{i+n-r+1} - u_i}\mathbf{c}_{i-1}^{r-1}(u) + \frac{u - u_i}{u_{i+n-r+1} - u_i}\mathbf{c}_i^{r-1}(u),$$
$$i = r, \ldots, n, \quad r = 1, \ldots, n,$$

$$\mathbf{c}(u) = \mathbf{c}_n^n = \frac{u_{n+1} - u}{u_{n+1} - u_n}\mathbf{c}_{n-1}^{n-1}(u) + \frac{u - u_n}{u_{n+1} - u_n}\mathbf{c}_n^{n-1}(u), \tag{2.8}$$

$$\mathbf{d}_i^1 = \frac{u_{i+n} - u}{u_{i+n} - u_i}\mathbf{d}_{i-1} + \frac{u - u_i}{u_{i+n} - u_i}\mathbf{d}_i, \quad i = 1, \ldots, n,$$

$$\mathbf{d}_i^r = \frac{u_{i+n-r+1} - u}{u_{i+n-r+1} - u_i}\mathbf{d}_{i-1}^{r-1}(u) + \frac{u - u_i}{u_{i+n-r+1} - u_i}\mathbf{d}_i^{r-1}(u),$$
$$i = r, \ldots, n, \quad r = 1, \ldots, n,$$

$$\mathbf{d}(u) = \mathbf{d}_n^n = \frac{u_{n+1} - u}{u_{n+1} - u_n}\mathbf{d}_{n-1}^{n-1}(u) + \frac{u - u_n}{u_{n+1} - u_n}\mathbf{d}_n^{n-1}(u), \tag{2.9}$$

Así,

$$\mathbf{d}(u) - \mathbf{c}(u) = \frac{u_{n+1} - u}{u_{n+1} - u_n}\left(\mathbf{d}_{n-1}^{n-1}(u) - \mathbf{c}_{n-1}^{n-1}(u)\right) + \frac{u - u_n}{u_{n+1} - u_n}\left(\mathbf{d}_n^{n-1}(u) - \mathbf{c}_n^{n-1}(u)\right). \tag{2.10}$$

Por otro lado, las derivadas de las curvas B-spline $\mathbf{c}(u)$, $\mathbf{d}(u)$ evaluadas en $u \in [u_n, u_{n+1})$ están dadas por:

$$\frac{d\mathbf{c}(u)}{du} = \frac{n}{u_{n+1} - u_n}\left(\mathbf{c}_n^{n-1}(u) - \mathbf{c}_{n-1}^{n-1}(u)\right) \tag{2.11}$$

$$\frac{d\mathbf{d}(u)}{du} = \frac{n}{u_{n+1} - u_n} \left(\mathbf{d}_n^{n-1}(u) - \mathbf{d}_{n-1}^{n-1}(u)\right). \tag{2.12}$$

Puesto que la superficie desarrollable generada por dos curvas paramétricas, $\mathbf{c}(u)$, $\mathbf{d}(u)$ está caracterizada por el hecho de que los vectores $\mathbf{c}'(u)$, $\mathbf{d}'(u)$, $\mathbf{d}(u) - \mathbf{c}(u)$ yacen sobre un plano para todo valor del parámetro u (ver teorema 2.1), entonces los puntos $\mathbf{c}_{n-1}^{n-1}(u)$, $\mathbf{c}_n^{n-1}(u)$, $\mathbf{d}_{n-1}^{n-1}(u)$, $\mathbf{d}_n^{n-1}(u)$ satisfacen la condición de coplanaridad.
Recíprocamente, como los vectores $\mathbf{c}'(u)$, $\mathbf{d}'(u)$, $\mathbf{d}(u) - \mathbf{c}(u)$ están dados por las ecuaciones 2.8-2.12 y los puntos $\mathbf{c}_{n-1}^{n-1}(u)$, $\mathbf{c}_n^{n-1}(u)$, $\mathbf{d}_{n-1}^{n-1}(u)$, $\mathbf{d}_n^{n-1}(u)$ son coplanares por hipótesis, entonces la condición de coplanaridad la satisfacen los vectores $\mathbf{c}'(u)$, $\mathbf{d}'(u)$, $\mathbf{d}(u) - \mathbf{c}(u)$. Por lo tanto, la superficie $\mathbf{S}$ es desarrollable (por el teorema 2.1).

Ahora consideremos una superficie $\mathbf{S}$ generada por dos curvas B-spline, $\mathbf{c}(u)$, $\mathbf{d}(u)$ compuestas cada una por dos segmentos de grado n, con polígonos de control $\{\mathbf{c_0}, \ldots, \mathbf{c_{n+1}}\}$, $\{\mathbf{d_0}, \ldots, \mathbf{d_{n+1}}\}$ y definidas en la misma secuencia de nodos $\{u_0, \ldots, u_{2n+2}\}$.
Para evaluar las curvas B-spline en el segundo segmento, con $u \in [u_{n+1}, u_{n+2})$, consideramos los puntos de control $\{\mathbf{c_1}, \ldots, \mathbf{c_{n+1}}\}$, $\{\mathbf{d_1}, \ldots, \mathbf{d_{n+1}}\}$ y la secuencia de nodos $\{u_1, \ldots, u_{2n+2}\}$. Reenumeramos estos puntos como $\{\mathbf{c_0} := \mathbf{c_1}, \ldots, \mathbf{c_n} := \mathbf{c_{n+1}}\}$, $\{\mathbf{d_0} := \mathbf{d_1}, \ldots, \mathbf{d_n} := \mathbf{d_{n+1}}\}$ y aplicamos el algoritmo de De Boor.
Las derivadas de las curvas B-spline $\mathbf{c}(u)$, $\mathbf{d}(u)$ en $u \in [u_{n+1}, u_{n+2})$ están definidas por:

$$\frac{d\mathbf{c}(u)}{du} = \frac{n}{u_{n+2} - u_{n+1}} \left(\mathbf{c}_n^{n-1}(u) - \mathbf{c}_{n-1}^{n-1}(u)\right)$$

$$\frac{d\mathbf{d}(u)}{du} = \frac{n}{u_{n+2} - u_{n+1}} \left(\mathbf{d}_n^{n-1}(u) - \mathbf{d}_{n-1}^{n-1}(u)\right),$$

donde $\mathbf{c}_{n-1}^{n-1}(u)$, $\mathbf{c}_n^{n-1}(u)$, $\mathbf{d}_{n-1}^{n-1}(u)$, $\mathbf{d}_n^{n-1}(u)$ son los penúltimos puntos del algoritmo de De Boor evaluado en $u \in [u_{n+1}, u_{n+2})$.
Análogamente a la prueba anterior, se concluye que $\mathbf{S}$ es una superficie desarollable si y sólo si los puntos $\mathbf{c}_{n-1}^{n-1}(u)$, $\mathbf{c}_n^{n-1}(u)$, $\mathbf{d}_{n-1}^{n-1}(u)$, $\mathbf{d}_n^{n-1}(u)$ con $u \in [u_n, u_{n+2})$ son coplanares.
Siguiendo un proceso similar a éste, la conclusión se extiende para curvas de N tramos.

□

La figura 2.8 ilustra la condición de coplanaridad para los puntos en el penúltimo paso del algoritmo de De Boor.
De lo anterior se sigue que:

$$\mathbf{d}_n^{n-1}(u) = (1 - \lambda - \mu)\mathbf{c}_{n-1}^{n-1}(u) + \lambda\mathbf{c}_n^{n-1}(u) + \mu\mathbf{d}_{n-1}^{n-1}(u),$$

donde los coeficientes λ, μ son funciones de u.
Reescribiendo esta expresión en una forma simétrica tenemos:

$$(1-\Lambda)\mathbf{c}_{n-1}^{n-1}(u) + \Lambda\mathbf{c}_n^{n-1}(u) = (1-\Gamma)\mathbf{d}_{n-1}^{n-1}(u) + \Gamma\mathbf{d}_n^{n-1}(u),$$
$$\Lambda = \frac{\lambda}{1-\mu}, \quad \Gamma = \frac{1}{1-\mu}.$$

En el trabajo de Fernández-Jambrina, (ver [9]), se presenta un algoritmo para construir mallas de control para superficies B-spline desarrollables de grado y número de tramos arbitrarios, este algoritmo puede ser considerado como una generalización del algoritmo de Aumann para construir superficies de Bézier desarrollables. A continuación el resultado.

Teorema 2.3 *Dos curvas B-spline* $\mathbf{c}(u)$, $\mathbf{d}(u)$ *con respectivos polígonos de control* $\{\mathbf{c_0}, \ldots, \mathbf{c_L}\}$, $\{\mathbf{d_0}, \ldots, \mathbf{d_L}\}$ *y secuencia de nodos* $\{u_0, \ldots, u_K\}$ *donde* $K = L + n + 1$, *definen una superficie desarrollable si existen* Λ^* *y* Γ^* *tales que:*

$$\frac{u_{i+n} - \Lambda^*}{u_{i+n} - u_i}\mathbf{c}_i + \frac{\Lambda^* - u_i}{u_{i+n} - u_i}\mathbf{c}_{i+1} = \frac{u_{i+n} - \Gamma^*}{u_{i+n} - u_i}\mathbf{d}_i + \frac{\Gamma^* - u_i}{u_{i+n} - u_i}\mathbf{d}_{i+1}, \tag{2.13}$$

para $i = 0, \ldots L-1$.

Corolario 2.1 *El caso* $\Lambda^* = \Gamma^*$ *corresponde a superficies cilíndricas y corresponde a superficies cónicas si los lados de los polígonos de control satisfacen,*

$$\Lambda^*(\mathbf{c}_{i+1} - \mathbf{c}_i) = \mathbf{d}_{i+1} - \mathbf{d}_i, \quad i = 0, \ldots, L-1. \tag{2.14}$$

La figura 2.9 muestra un ejemplo de una superficie B-spline desarrollable cilíndrica.

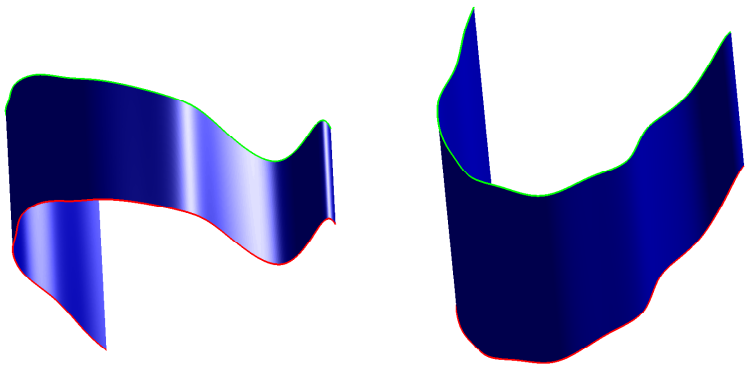

Figura 2.9 Dos vistas de una superficie B-spline cilíndrica.

Prueba del teorema 2.3

Se demostrará para el caso $n = 3$. Supongamos que $\mathbf{c}(u)$, $\mathbf{d}(u)$ son curvas B-spline cúbicas de un sólo segmento, definidas por sus polígonos de control $\{\mathbf{c_0}, \ldots, \mathbf{c_3}\}$ y $\{\mathbf{d_0}, \ldots, \mathbf{d_3}\}$ respectivamente, y la misma secuencia de nodos $\{u_0, \ldots, u_7\}$.

Calculando $\mathbf{c}(u)$ con el algoritmo de De Boor tenemos que:

$$\begin{aligned}
\mathbf{c}(u) \;=\; & \frac{u_3-u}{u_3-u_2}\frac{u_3-u}{u_3-u_1}\frac{u_3-u}{u_3-u_0}\mathbf{c}_0 + \frac{u_3-u}{u_3-u_2}\frac{u_3-u}{u_3-u_1}\frac{u-u_0}{u_3-u_0}\mathbf{c}_1 + \\
& \frac{u_3-u}{u_3-u_2}\frac{u_3-u}{u_3-u_1}\frac{u_4-u}{u_4-u_1}\mathbf{c}_1 + \frac{u_3-u}{u_3-u_2}\frac{u-u_1}{u_3-u_1}\frac{u-u_1}{u_4-u_1}\mathbf{c}_2 + \\
& \frac{u-u_2}{u_3-u_2}\frac{u_4-u}{u_4-u_2}\frac{u_4-u}{u_4-u_1}\mathbf{c}_1 + \frac{u-u_2}{u_3-u_2}\frac{u_4-u}{u_4-u_2}\frac{u-u_1}{u_4-u_1}\mathbf{c}_2 + \\
& \frac{u-u_2}{u_3-u_2}\frac{u-u_2}{u_4-u_2}\frac{u_5-u}{u_5-u_2}\mathbf{c}_2 + \frac{u-u_2}{u_3-u_2}\frac{u-u_2}{u_4-u_2}\frac{u-u_2}{u_5-u_2}\mathbf{c}_3
\end{aligned}$$

Así por la proposición anterior,

$$\begin{aligned}
(1-\Lambda)\mathbf{c}_0^2(u) + \Lambda\mathbf{c}_1^2(u) \;=\; & \frac{u_3-u}{u_3-u_1}\frac{u-u_3}{u_3-u_0}\mathbf{c}_0 + \frac{u_3-u}{u_3-u_1}\frac{u-u_0}{u_3-u_0}\mathbf{c}_1 + \frac{u_4-u}{u_4-u_1}\frac{u-u_1}{u_3-u_1}\mathbf{c}_1 + \\
& \frac{u-u_1}{u_3-u_1}\frac{u-u_1}{u_4-u_1}\mathbf{c}_2 - \Lambda\frac{u_3-u}{u_3-u_1}\frac{u_3-u}{u_3-u_0}\mathbf{c}_0 - \Lambda\frac{u_3-u}{u_3-u_1}\frac{u-u_0}{u_3-u_0}\mathbf{c}_1 - \\
& \Lambda\frac{u-u_1}{u_3-u_1}\frac{u_4-u}{u_4-u_1}\mathbf{c}_1 - \Lambda\frac{u-u_1}{u_3-u_1}\frac{u-u_1}{u_4-u_1}\mathbf{c}_2 - \Lambda\frac{u_4-u}{u_4-u_2}\frac{u_4-u}{u_4-u_1}\mathbf{c}_1 + \\
& \Lambda\frac{u_4-u}{u_4-u_2}\frac{u-u_1}{u_4-u_1}\mathbf{c}_2 + \Lambda\frac{u-u_2}{u_4-u_2}\frac{u_5-u}{u_5-u_2}\mathbf{c}_2 + \Lambda\frac{u-u_2}{u_4-u_2}\frac{u-u_2}{u_5-u_2}\mathbf{c}_3.
\end{aligned}$$

Por hipótesis para $i = 0$ tenemos que:

$$\begin{aligned}
\frac{u_3-\Lambda^*}{u_3-u_0}\mathbf{c}_0 + \frac{\Lambda^*-u_0}{u_3-u_0}\mathbf{c}_1 \;&=\; \frac{u_3-(1-\Lambda)u_2-\Lambda u_3}{u_3-u_0}\mathbf{c}_0 + \frac{(1-\Lambda)u_2+\Lambda u_3-u_0}{u_3-u_0}\mathbf{c}_1 \\
&=\; (1-\Lambda)\frac{u_3-u_2}{u_3-u_0}\mathbf{c}_0 + \frac{u_2-u_0}{u_3-u_0}\mathbf{c}_1 + \Lambda\frac{u_3-u_2}{u_3-u_0}\mathbf{c}_1.
\end{aligned}$$

Pero esto es,

$$\frac{u_3-\Lambda^*}{u_3-u_0}\mathbf{c}_0 + \frac{\Lambda^*-u_0}{u_3-u_0}\mathbf{c}_1 \;=\; (1-\Lambda)\mathbf{c}_0^2(u) + \Lambda\mathbf{c}_1^2(u) \quad \text{para} \quad u \in [u_1, u_2).$$

Similarmente,

$$\begin{aligned}
\frac{u_4-\Lambda^*}{u_4-u_1}\mathbf{c}_1 + \frac{\Lambda^*-u_1}{u_4-u_1}\mathbf{c}_2 \;&=\; (1-\Lambda)\mathbf{c}_0^2(u) + \Lambda\mathbf{c}_1^2(u) \quad \text{para} \quad u \in [u_2, u_3), \quad i = 1, \\
\frac{u_5-\Lambda^*}{u_5-u_2}\mathbf{c}_2 + \frac{\Lambda^*-u_2}{u_5-u_2}\mathbf{c}_3 \;&=\; (1-\Lambda)\mathbf{c}_0^2(u) + \Lambda\mathbf{c}_1^2(u) \quad \text{para} \quad u \in [u_3, u_4), \quad i = 2.
\end{aligned}$$

Análogamente,

$$\frac{u_3-\Gamma^*}{u_3-u_0}\mathbf{d}_0+\frac{\Gamma^*-u_0}{u_3-u_0}\mathbf{d}_1 \quad = \quad (1-\Gamma)\mathbf{d}_0^2(u)+\Gamma\mathbf{d}_1^2(u) \quad \text{para} \quad u\in[u_1,u_2),, \quad i=0,$$

$$\frac{u_4-\Gamma^*}{u_4-u_1}\mathbf{d}_1+\frac{\Gamma^*-u_1}{u_4-u_1}\mathbf{d}_2 \quad = \quad (1-\Gamma)\mathbf{d}_0^2(u)+\Gamma\mathbf{d}_1^2(u) \quad \text{para} \quad u\in[u_2,u_3), \quad i=1,$$

$$\frac{u_5-\Gamma^*}{u_5-u_2}\mathbf{d}_2+\frac{\Gamma^*-u_2}{u_5-u_2}\mathbf{d}_3 \quad = \quad (1-\Gamma)\mathbf{d}_0^2(u)+\Gamma\mathbf{d}_1^2(u) \quad \text{para} \quad u\in[u_3,u_4), \quad i=2.$$

Así por la proposición anterior $\mathbf{c}_0^{n-1}(u)$, $\mathbf{c}_1^{n-1}(u)$, $\mathbf{d}_0^{n-1}(u)$, $\mathbf{d}_1^{n-1}(u)$ son coplanares.
Por lo tanto, $\mathbf{c}(u)$ y $\mathbf{d}(u)$ definen una superficie desarrollable.

$\square$

Capítulo 3

Aplicaciones

La finalidad de este capítulo es describir la estructura interna y externa en forma simultánea de las piezas dentales de un maxilar superior humano, extrayendo información a lo largo de rodajas curvas a partir de volúmenes tomográficos. En general, cuando se despliega en forma plana una superficie curva, ésta puede estirarse en algunas zonas de la rodaja, lo cual genera el problema de la deformación, como se ilustra en el trabajo de Saroul [12][13][14]. Este problema de deformación lo podemos evitar usando superficies desarrollables.
Es también interesante observar que el despliegue en tres dimensiones de una sección espacial curvilínea permitirá la inspección simultánea del conjunto de piezas dentales, la cual puede ser útil desde el punto de vista terapéutico.

3.1. Formato DICOM

DICOM (Digital Imaging and Communications in Medicine) es el formato estándar internacional para el intercambio de imágenes médicas. Este formato lo constituye un archivo con la información de la imagen que describe e información del paciente. Actualmente se está incrementando su uso en otros dominios médicos tales como la oftalmología y la odontología.
La figura 3.1 muestra dos ejemplos de imágenes DICOM.

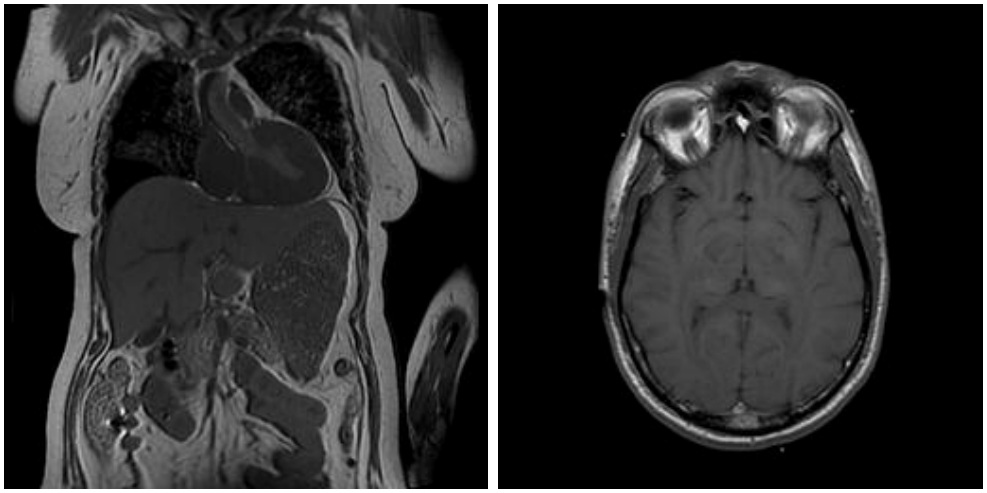

Figura 3.1 Imágenes de TC y RM [1]: RM coronal del cuerpo de un hombre y TC axial de la cabeza de una mujer.

Para el análisis de las imágenes médicas que se consideran en esta tesis, se utilizó el software matemático Matlab. Este software tiene las instrucciones *dicomread* y *dicominfo* que permiten leer los archivos en formato DICOM.
Si tenemos el archivo **imagen.dcm**, la instrucción **M=dicomread('imagen.dcm')** genera la matriz M de números con la codificación de las intensidades en los píxeles de la imagen. La instrucción **dicominfo('imagen.dcm')** permite visualizar la información adicional sobre los datos del paciente y de la imagen, tales como posición y orientación.
La figura 3.2 ilustra la codificación de colores en una imagen médica.

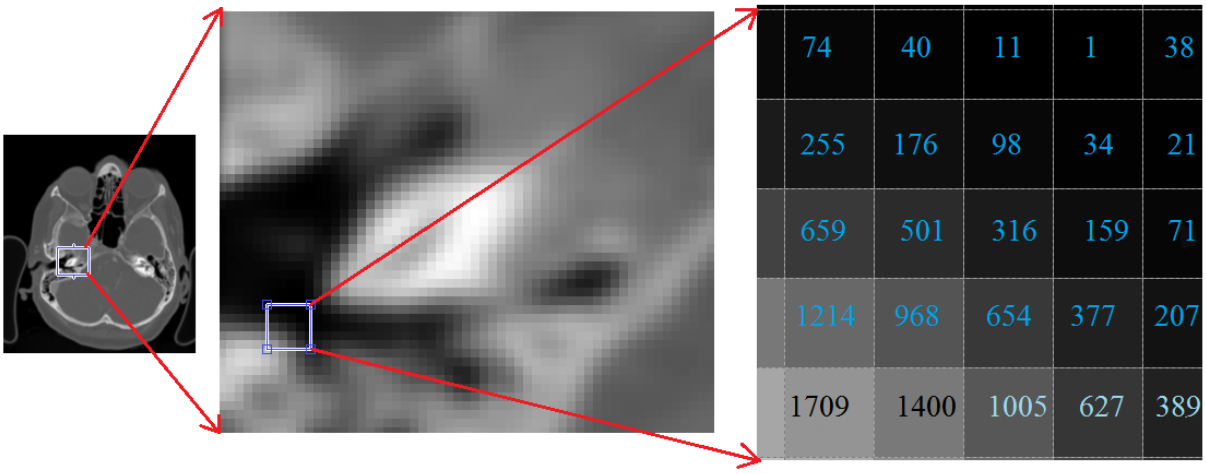

Figura 3.2 Codificación de colores en una imagen médica.

[1]TC (Computed Tomography) Tomografía Computarizada, RM (Magnetic Resonance) Resonancia Magnética

3.2. Rodajas Curvas Construidas con Superficies B-spline Desarrollables

En esta sección consideramos un volumen de imágenes médicas en formato DICOM, las cuales contienen información de la estructura ósea de un maxilar humano. Utilizaremos la construcción de superficies B-spline desarrollables del capítulo anterior para modelar rodajas curvas que permitan visualizar la información de interés contenida en el volumen.

3.2.1. Volumen

La secuencia de imágenes que estamos considerando contiene información (en términos de los niveles de gris) de los puntos del volumen que contiene el maxilar.
Las superficies regladas por su propiedad de ser consideradas como la superficie generada por el movimiento de un segmento de recta, hace que su programación sea fácil de implementar algoritmicamente, además su implementación es estable numéricamente.
El despliegue de estas superficies sobre el volumen requiere del proceso de texturización. Este proceso consiste en asignar a cada punto sobre la superficie el correspondiente nivel de gris de acuerdo a su ubicación en el volumen. Sin embargo, no todos los puntos pertenecen a dicho volumen, por lo tanto para asignarle su valor es necesario ubicarlo espacialmente y buscar los ocho puntos del volumen más cercanos a él, su nivel de gris correspondiente se calcula entonces como una combinación convexa de los niveles de gris de los puntos hallados.
Sabemos que una superficie reglada tiene la forma

$$\mathbf{S}(u,v) = (1-v)\mathbf{c}(u) + v\mathbf{d}(u) \quad 0 \leq u, v \leq 1$$

donde $\mathbf{c}(u)$ y $\mathbf{d}(u)$ son directrices de la superficie.
Escogiéndo un polígono de control adecuado y una secuencia de nodos uniforme para determinar la curva $\mathbf{c}(u)$, construiremos la curva $\mathbf{d}(u)$ utilizando el algoritmo explicado en el capítulo anterior para generar la superficie desarrollable. La escogencia de este tipo de superficies sobre las superficies regladas, es porque nos permitirán la inspección visual de la información sobre una región que contiene información de los dientes contenidos en el maxilar y su despliegue sin deformación.
La figura 3.3 ilustra una rodaja curva 3D para dos dientes.

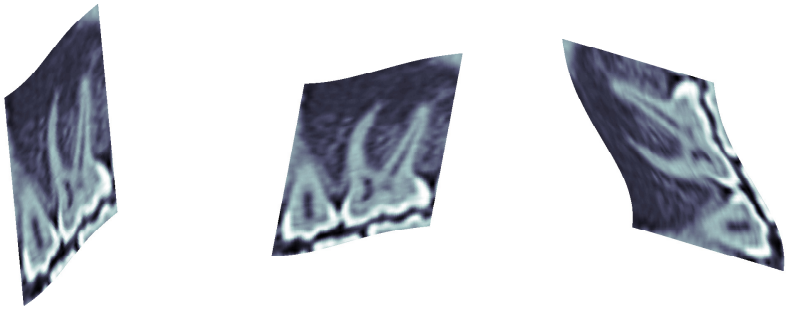

Figura 3.3 Rodaja curva 3D para dos piezas dentales del maxilar superior.

3.2.2. Algoritmo

Para la construcción de las superficies B-spline desarrollables la generatriz $\mathbf{c}(u)$ es considerada como una curva B-spline cúbica C^2. Los pasos a seguir son:

- Se realiza una inspección visual de las imágenes del volumen DICOM, con el fin de determinar la secuencia de imágenes donde están contenidos los dientes. La primera imagen sobre la corona dental y la última sobre la raíz.

- Para determinar la generatriz $\mathbf{c}(u)$ de la superficie desarrollable, seleccionamos una secuencia $\mathbf{c}_0, \ldots, \mathbf{c}_n$ de puntos de control en la primera imagen[2] sobre la corona de los dientes de interés y calculamos su B-spline cúbico C^2.

- El primer punto de control $\mathbf{d}_0$ de $\mathbf{d}(u)$ se toma sobre la última imagen y preferiblemente sobre la raíz de alguno de los dientes.

- Escogemos los parámetros Λ^* y Γ^* del Teorema 2.3 para determinar el tipo de superficie[3].

- Los puntos de control $\mathbf{d}_1, \ldots, \mathbf{d}_n$ de la curva generatriz $\mathbf{d}(u)$ se calculan a partir del teorema 2.3.

[2]En nuestro caso tomamos todos los $\mathbf{c_i}$ en una sola imagen, esto en general no es necesario.

[3]Para nuestros ejemplos: superficie cilíndrica figura 3.5, $\Lambda^* = \Gamma^* = 2000$. Figura 3.6, Superficie cónica $\Lambda^* = 2000$ y $\Gamma^* = 0.5$, superficie tangente $\Lambda^* = 94$ y $\Gamma^* = 15$.

- Finalmente, trazamos y texturizamos la superficie desarrollable generada por las dos curvas $\mathbf{c}(u)$ y $\mathbf{d}(u)$.

La figura 3.4 muestra una vista 3D del maxilar superior, utilizando el proceso descrito anteriormente.

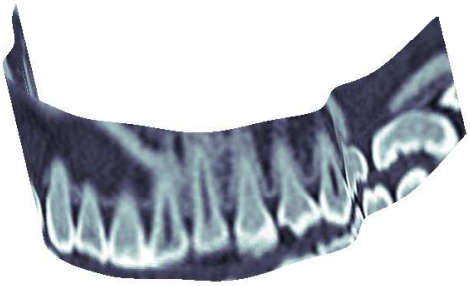

Figura 3.4 Vista 3D de una rodaja curva para las piezas dentales del maxilar superior.

La figura 3.5 ilustra diferentes vistas 3D del maxilar superior e inferior.
Las figuras 3.6 ilustra una superficie cónica y una tangente del maxilar superior.

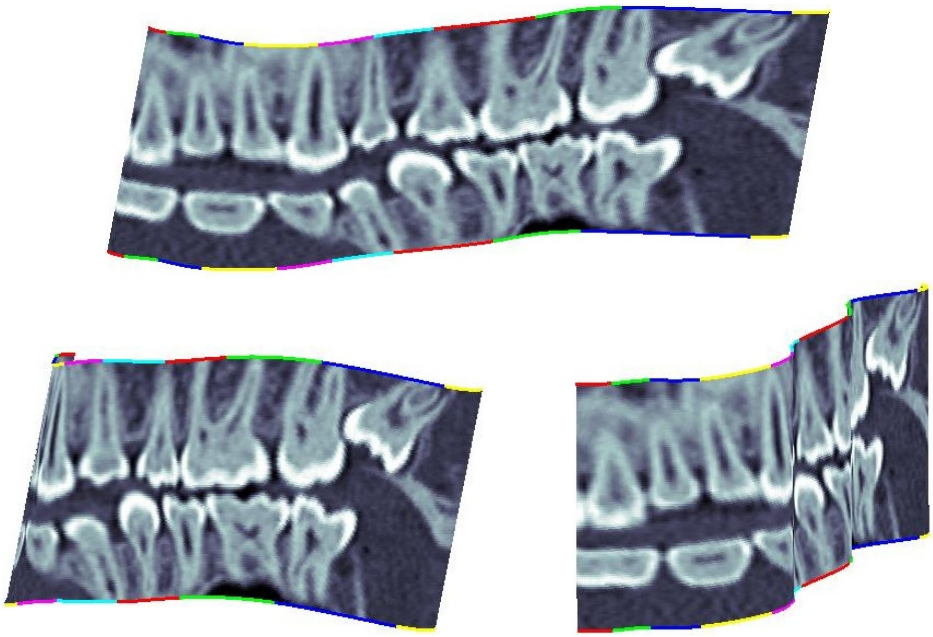

Figura 3.5 Diferentes vistas de una rodaja curva 3D para algunas piezas dentales.

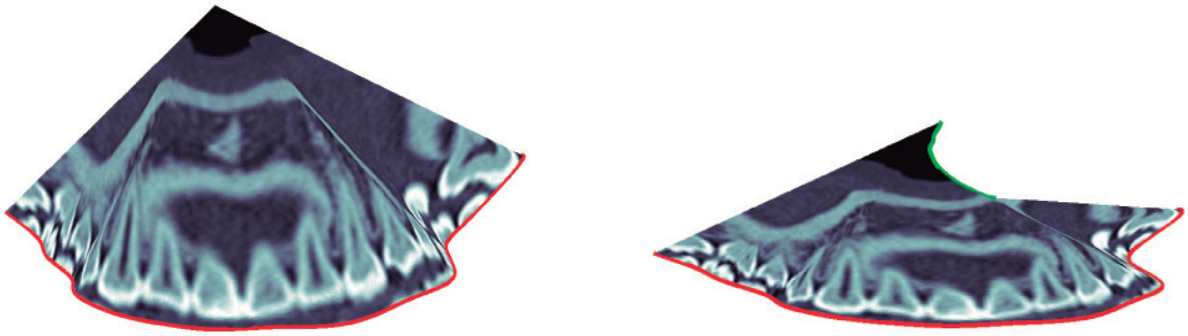

Figura 3.6 Superficie B-spline cónica (izquierda) y tangente (derecha) para el maxilar superior.

Observación

Para las aplicaciones ninguna de las superficies es adecuada para desplegar toda la información dental, sin embargo, las superficies cilíndricas y cónicas permiten una mejor visualización local de la región de interés.

Conclusiones

1. Se definen los conceptos para curvas de Bézier y curvas B-spline. Estas curvas son útiles en las aplicaciones por la facilidad para la manipulación de su forma. Además estas curvas poseen una serie de propiedades matemáticas que nos facultan para la realización de cálculos y aproximaciones numéricas. El algoritmo de De Boor es estable numéricamente y permite la evaluación de forma rápida y precisa.

2. Las superficies desarrollables poseen la propiedad que pueden ser aplicadas al plano sin ninguna distorsión métrica o equivalentemente son superficies regladas con curvatura gaussiana cero, por esta razón han sido ampliamente utilizadas en diversas aplicaciones. En esta tesis se diseñaron superficies generadas por dos curvas de Bézier o B-spline.

3. Se presentó una caracterización de las superficies B-spline desarrollables en términos de los puntos de control de las dos curvas B-spline que las generan. Ésta es la propuesta de Fernández Jambrina [9], la cual provee un algoritmo simple para construir su polígono de control. La malla de control se genera a partir del polígono de control de una de las curvas B-spline y el primer punto de control de la segunda curva. Este algoritmo puede ser considerado como una generalización del algoritmo de Aumann [1] para construir superficies de Bézier desarrollables.

4. El método para diseñar superficies B-spline desarrollables ha sido utilizado para describir simultáneamente la estructura de un conjunto de piezas dentales de un maxilar humano, extrayendo la información de un volumen tomográfico. Finalmente, se presentaron algunos ejemplos de visualizaciones 3D de superficies desarrollables cilíndricas construidas para piezas dentales del maxilar superior.

Códigos Matlab

En este capítulo se presentan los algoritmos diseñados para generar los gráficos de la tesis.

3.3. Polinomios de Bernstein

```
%Los polinomios de Bernstein son:
%Grado 2: B0,2(t)=(1 - t)^2, B1,2(t)=2*(1 - t)^t, B2,2(t)=t^2;
%Grado 3: B0,3(t)=(1 - t)^3, B1,3(t)=3*(1 - t)^2*t, B2,3(t)=3*(1 - t)*t^2,
    B3,3(t)=t^3

t=linspace(0,1,500);
B02=(1 - t).^2
B12=2*(1 - t).*t
B22=t.^2
%Gráfica polinomios Bernstein Grado 2
figure(1)
plot(t,B02,t,B12,t,B22,'-','LineWidth',2.3)
title('Polinomios Bernstein Grado 2')
B03=(1 - t).^3;
B13=3.*((1 - t).^2).*t;
B23=3.*(1 - t).*(t.^2);
B33=t.^3;
%Gráfica polinomios Bernstein Grado 3
figure(2)
plot(t,B03,t,B13,t,B23,t,B33,'-','LineWidth',2.3)
title('Polinomios Bernstein Grado 3')
```

3.4. Curva de Bézier

```
%%Gráfica Curva de Bézier Grado 2

%Para evaluar la curva, se genera un vector aleatorio para los respectivos
%puntos, sin embargo éstos valores pueden ser cualquier número real.
x=rand(1,3);
y=rand(1,3);
X= x(1).*((1 - t).^2) + 2.*x(2).*((1 - t).*t) + x(3).*(t.^2);
Y= y(1).*((1 - t).^2) + 2.*y(2).*((1 - t).*t) + y(3).*(t.^2);
figure
hold on
plot(X,Y,'-','LineWidth',2.3), axis off
plot(x,y,'r-','LineWidth',2.3)
plot(x,y,'bo','MarkerSize',5,'MarkerFaceColor','b')
```

3.5. Algoritmo de De Boor

```
%Versión 1
function vdBoor = Eval_dBoor(PB, orden, nodos, t)
% Algoritmo de De Boor para un B-spline abierto.

%% Datos de Entrada:
    %PB es una matriz n-filas, 2-columnas con las coordenadas de los
    %puntos de De Boor

    %orden es un entero positivo que determina el grado de la curva.
    %grado = orden - 1.


    %nodos es un vector fila con los valores de los nodos. Los valores son
    %una sucesión de numeros reales creciente.
    %El numero de nodos debe ser igual al numero de puntos de De Boor mas
       el
    %orden de la curva.
    %Si un nodo tiene multiplicidad mayor a uno, esta debe ser menor al
    %orden de la curva.

    %t es el parámetro a evaluar

%% Datos de salida:
    %vdBoor es el punto de la curva correpondiente al parametro t

%% Inicio del programa
    %ubicar el nodo a_i tal que t en [a_i, a_i+1)

 indice = find(t>=nodos,1,'last');
 if (t==nodos(end))
    indice = size(nodos,2) - sum(t==nodos);
 end

 npb = size(PB,1);
 grado = orden - 1;
 pbx = zeros(npb,npb);
 pbx(:,1) = PB(:,1);
 pby = zeros(npb,npb);
 pby(:,1) = PB(:,2);

 if ( (indice <= grado))  || (indice >= ( (npb + 1) ) )
   vdBoor=[0 0 0];
```

```
else
  for j=2:orden
   for i=indice - orden + j:indice
       alfa_1 = t - nodos(1,i);
       alfa_2 = nodos(1,i + orden - j + 1) - nodos(1,i);
       if ((alfa_1 == 0) || (alfa_2 == 0))
           alfa = 0;
       else
           alfa = alfa_1 / alfa_2;
       end
       pbx(i,j) = ((1 - alfa)*pbx(i-1,j-1)) + (alfa*pbx(i,j-1));
       pby(i,j) = ((1 - alfa)*pby(i-1,j-1)) + (alfa*pby(i,j-1));
    end
   end
    vdBoor = [pbx(indice,orden) pby(indice,orden)];
end

%Versión 2
%% Datos de Entrada:
   %PB es una matriz n-filas, 2-columnas con las coordenadas de los
   %puntos de De Boor

   %orden es un entero positivo que determina el grado de la curva.
   %grado = orden - 1.


   %nodos es un vector fila con los valores de los nodos. Los valores son
   %una sucesion de numeros reales no decreciente. a_{1} <= ... a_{i}
      <=...

   %El numero de nodos es igual al numero de puntos de De Boor mas el
   %orden de la curva.

   %Si un nodo tiene multiplicidad mayor a uno, esta debe ser menor al
   %orden de la curva.

   %t es el parámetro a evaluar

   %indice es el escalar tal que t en [a_{i}, a_{i+1}), indice = i.

%% Datos de salida:

   %vdBoor es el punto de la curva correpondiente al parametro t.
```

```
%% Inicio del programa

    % Formar las matrices con los puntos de De Boor para aplicar el
        algoritmo.
npb = size(PB,1);
pbx = zeros(npb,npb);
pby = zeros(npb,npb);
pbx(:,1) = PB(:,1);
pby(:,1) = PB(:,2);

    % Aplicación del algoritmo de De Boor.

for j=2:orden
    for i=indice - orden + j:indice
        alfa_1 = t - nodos(1,i);
        alfa_2 = nodos(1,i + orden - j + 1) - nodos(1,i);
        if ((alfa_1 == 0) || (alfa_2 == 0))
            alfa = 0;
        else
            alfa = alfa_1 / alfa_2;
        end
            pbx(i,j) = ((1 - alfa)*pbx(i-1,j-1)) + (alfa*pbx(i,j-1));
            pby(i,j) = ((1 - alfa)*pby(i-1,j-1)) + (alfa*pby(i,j-1));
    end
end
vdBoor = [pbx(indice,orden) pby(indice,orden)];
```

3.6. Curva B-spline

Existen dos versiones del código para generar una curva B–**spline**. Ambas usan la respectiva versión de Eval_DeBoor. Sin embargo para la segunda versión el cálculo de los índices de los nodos se realiza en un programa aparte llamado indices.m, que a la vez nos permite tener control sobre el número de segmentos de la curva.\\

```
% Versión 1
function [nodos,orden, p_spl]=spline(PB)
%Programa para graficar un B-Spline con n puntos de De Boor.

%% Datos de Entrada:
    %PB es una matriz n-filas, 2-columnas con las coordenadas de los
    %puntos de De Boor

    %orden es un entero positivo que determina el grado de la curva.
    %grado = orden - 1.


    %nodos es un vector fila con los valores de los nodos. Los valores son
    %una sucesión de numeros reales creciente.
    %El numero de nodos debe ser igual al numero de puntos de De Boor mas
       el
    %orden de la curva.
    %Si un nodo tiene multiplicidad mayor a uno, esta debe ser menor al
    %orden de la curva.

    %t es un vector de parametros evaluar

    %Funcion Eval_dBoor para calcular los puntos de la curva

%Salida: Grafico del B-Spline


%% Inicio del programa

 clear all
 clc
PB = rand(10,2).*10;

%Definir el orden
 orden = 4;
```

```
 % Definir el vector de nodos

% nodos=[0 0 0 0 1 1 1 1 2 2 3 3 3 3];
% nodos=[0 0 0 1 2 3 4 5 6 7 8 8 8 8];
 nodos = 1:10;

% Definir el vector de parametros
 t=nodos(1):0.01:nodos(end);

 %% Usar el algoritmo de De Boor para calcular los puntos de la curva B-
     spline.
 p_spl1=zeros(size(t,2),2);
 for i = 1:size(t,2)
         p_spl1(i,:) = Eval_dBoor(PB,orden,nodos,t(i));
 end

 a=find(p_spl1(:,1));
 b=find(p_spl1(:,2));
 p_spl=[p_spl1(a,1) p_spl1(b,2)];

 %% Graficar el poligono de De Boor, los puntos de control y la curva.
 figure
 hold on
 plot(p_spl(:,1),p_spl(:,2),'-g');
 plot(PB(:,1),PB(:,2),'bo','MarkerSize',5,'MarkerFaceColor','b');
 plot(PB(:,1),PB(:,2),'r-');

% Versión 2
%%%Inicio del programa
clear all, clc
tic
    % Definir los puntos de De Boor

        PB = rand(9,2).*100;

    % Definir el orden

        orden = 4;

    % Definir el vector de nodos. El numero de nodos es igual al numero de
    % puntos de De Boor mas el orden.

        % nodos = [-10 -8 -5 0.5 1 1 1 1 2.5 3 5 7 9];
```

```
        %nodos = [-10  -8  -5  0.5  0.5  0.5  3  5  7  9  11  14  15];
        %nodos = [-10  -8  -5  -1  0.5  0.5  3  5  7  9  11  14  15];
        nodos = [1 1 1 1 2 3 4 5 6 7 7 7 7];
        %nodos = 1:13;

    %Definir una matriz de parametros a evaluar en el dominio de la curva.

        npb = size(PB,1);
        parametros = indices(nodos, npb, orden);

%%Evaluación de los parámetros usando el algoritmo de De Boor

        %evaluación del primer segmento

        for i = 1 : size(parametros(1,:),2) - 1
            p_spl(i,:) = Eval_dBoor1(PB, orden, nodos, parametros(1,i),
                parametros(1,end));
        end

        %evaluación de los segmentos restantes y creación de la matriz de
        %puntos de la curva

        for j = 2 : size(parametros(:,end),1)
            for i = 1 : size(parametros(1,:),2) - 1
                p_spl1(i,:) = Eval_dBoor1(PB, orden, nodos, parametros(j,i)
                    , parametros(j,end));
            end
            p_spl = [p_spl;p_spl1];
        end


%%Graficar el poligono de De Boor, los puntos de control y la curva.
   hold on
   plot(p_spl(:,1),p_spl(:,2),'k-','MarkerSize',8); %,'LineWidth',2);
   plot(PB(:,1),PB(:,2),'bo','MarkerSize',5,'MarkerFaceColor','b');
   plot(PB(1,1),PB(1,2),'rs','MarkerSize',5,'MarkerFaceColor','b');
   plot(PB(end,1),PB(end,2),'bs','MarkerSize',5,'MarkerFaceColor','b');
   %plot(PB(:,1),PB(:,2),'r-');
toc

function P = indices(nodos, npb, orden)
%Función para generar una matriz de parámetros con sus respectivos indices
%en el dominio del b-spline.
```

```
%% Datos de Entrada:
% nodos es el vector de nodos.

% npb es el número de puntos de De Boor.

% grado es el grado de la curva.

%% Datos de Salida:
% P matriz de parámetros a evaluar por segmento. La última columna
% corresponde al indice por segmento.

%%
        [d(1,:) d(2,:)]=hist(nodos,unique(nodos));
        d(3,:)=cumsum(d(1,:));
        indices_dominio = find((d(3,:) >= orden) & (d(3,:) <= npb));

 % La matriz d tiene 3 filas. En la primera, cada posición
 % corresponde al número de repeticiones de cada nodo. La segunda
 % corresponde al conjunto de nodos sin repeticiones. En la tercera, cada
 % posición indica el indice para el cual un parámetro t, pertenece a
 %[a_{i}, a_{i+1}).

 %%
        for i = indices_dominio(1) : indices_dominio(end)
            P(i - (indices_dominio(1) - 1),:) = linspace(d(2,i), d(2,i+1)
                ,500);
        end

 % La matriz P tiene tantas filas como segmentos tenga el b-spline en el
 % dominio en el cual está definido. Por cada segmento se evaluaran 500
 % valores. La última columna corresponde al valor de indice de cada
 % segmento.

        P = [P (d(3,indices_dominio(1:end)))'];
```

```
Ejemplo 1
P =

   -1.2177    1.4737
   -1.4838    1.7232
   -1.9164    2.5941
   -2.3410    5.0375
   -2.3162    8.5543
   -1.7975   10.8508
```

```
   -1.1876   11.9022
    0.8871   11.7405
    1.0749    8.6450
    1.1521    6.3109
    1.3329    3.3040
    0.2423    1.6661
   -0.3186    2.5230
   -0.9515    5.5050
   -1.2466    7.8781
   -0.2072   10.2805
    0.1210    9.0293
    0.2984    7.9298

nodos=[  0    0    0    0    1    2    3    4    5    6    7
   8    9   10   11   12   13   14   15   15   15   15]

Ejemplo 2
P =

    1.2292    3.1123
    2.3480    1.3297
    4.4722   -1.6050
   -0.5011   10.9982
    9.8243   -4.7359
   -1.1444   20.2375
   17.0942  -10.0664
   -1.0847   42.3690
   37.9652   -9.4593
   -0.8259   46.1885
   18.6341   24.9053
   31.5931   10.5043

nodos=[0    0    0    0    1    2    3    4    5
    6    7    8    9    9    9    9]

Ejemplo 3
P =

    3.2000   -0.2000
    2.0167    0.3889
    0.0995    1.1551
   -2.0731    0.9240
   -1.6068   -0.2508
   -3.2999   -2.4211
```

```
    0.6063    -1.1650
    1.6746    -3.2193
    2.8954    -0.8576
    1.5945    -0.7528
    0.6000    -0.7000

nodos=[0     0     0     0     1     2     3     4     5     6     7     8
       8     8     8]

Ejemplo 4
P =

    2.5025     4.4966
    2.3353     4.8243
    2.0091     5.3170
    1.4861     4.0936
    1.1040     5.3292
    0.1009     4.4071
    1.4713     4.0219
   -0.0131     3.4849
    1.4994     2.9780
   -0.0116     2.5016
    1.4953     1.9951
    0.0034     1.4975
    1.4700     0.9946
    0.0895     0.5847
    1.1205    -0.3132
    1.4617     0.9313
    1.9999    -0.4325
    2.5297     0.9812
    2.9868    -0.3909
    3.5322     0.9268
    3.8876    -0.2558
    4.9145     0.5626
    3.5359     0.9849
    4.9996     1.4774
    3.4866     1.8822
    4.9902     2.5274
    3.5129     2.9474
    4.9850     3.4596
    3.5379     3.9910
    4.8605     4.4348
    3.9207     5.2902
    3.5144     4.1810
```

```
    3.1679      4.6498
    3.0116      4.9764

nodos=[0          0          0          0          1          2          3          4          5
          6          7          8          9          10         11         12         13
   14         15         16         17         18         19         20         21         22
   23         24         25         26         27         28         29         30         31
   31         31         31]
```

3.7. Superficies de Bézier

```
%Bezier Patch
%BPm,n(u,v)= Bm(u)*Bmn*Bn(v)
%Bm(u) vector fila que contiene los polinomios de Bernstein de grado m
%evaluados en u.
%Bn(v) vector columna que contiene los polinomios de Bernstein de grado n
%evaluados en v.
%Bmn matriz de orden m*n que contiene los puntos de control, llamada
%también red de control del parche.
%BPmn parche de bezier o producto tensorial de grado m*n evaluado en las
%variables u y v
%El producto en la formula es producto matricial.

%Definición de Parámetros y Polinomios de Bernstein

t=linspace(0,1,500);
s=linspace(0,1,500);
p=linspace(0,1,500);

%Polinomios Bernstein Grado 2
B02=(1 - t).^2;
B12= 2.*(1 - t).*t;
B22= t.^2;
%Polinomios Bernstein Grado 3
B03=(1 - s).^3;
B13=3.*((1 - s).^2).*s;
B23=3.*(1 - s).*(s.^2);
B33=s.^3;
%Polinomios Bernstein Grado 4
B04=(1 - p).^4;
B14=4.*((1 - p).^3).*p;
B24=6.*((1 -p).^2).*(p.^2);
B34=4.*(1 - p).*(p.^3);
B44=p.^4;
%Parche Bézier grado 2 por 3
B2=[B02' B12' B22'];
B3=[B03; B13; B23; B33];
x=repmat([1 2 3 4],3,1);
y=repmat([1;2;3],1,4);
z=rand(3,4);
PBX23=B2*x*B3;
PBY23=B2*y*B3;
PBZ23=B2*z*B3;
```

```
subplot(2,2,1);
surf(PBX23,PBY23,PBZ23), shading('interp');
title('Parche de Bézier 2 por 3');

%Parche Bézier grado 3 por 3
x=repmat([1 2 3 4],4,1);
y=repmat([1;2;3;4],1,4);
z=rand(4,4);
PBX33=(B3')*x*B3;
PBY33=(B3')*y*B3;
PBZ33=(B3')*z*B3;
subplot(2,2,2);
surf(PBX33,PBY33,PBZ33), shading('interp');
title('Parche de Bézier 3 por 3');

%Parche Bézier grado 3 por 4
x=repmat([1 2 3 4 5],4,1);
y=repmat([1;2;3;4],1,5);
z=rand(4,5);
B4=[B04; B14; B24; B34; B44];
PBX34=(B3')*x*B4;
PBY34=(B3')*y*B4;
PBZ34=(B3')*z*B4;
subplot(2,2,3);
surf(PBX34,PBY34,PBZ34), shading('interp');
title('Parche de Bézier 3 por 4');

%Parche Bézier grado 4 por 4
x=repmat([1 2 3 4 5],5,1);
y=repmat([1;2;3;4;5],1,5);
z=rand(5,5);
PBX44=(B4')*x*B4;
PBY44=(B4')*y*B4;
PBZ44=(B4')*z*B4;
subplot(2,2,4);
surf(PBX44,PBY44,PBZ44), shading('interp');
title('Parche de Bézier 4 por 4');
```

3.8. Curva Generatriz Base

```
function d=puntosd(T,C,G,L,d0,orden)
%puntosd es una función para calcular los puntos de control de la segunda
    directriz de
%e la superficie desarrollable.

%Datos de entrada
%T matriz de nodos
%L, G parámetros del resultado principal de Fernández Jambrina, teorema
    3.6.
%Estos parámetros determinan el tipo de superficie a trazar.
%d0 punto de control inicial de la curva d.
%orden de la curva, orden=grado + 1.

%Datos de salida:
%d puntos de control de la curva d.
n=orden-1;
L=size(C,1)-1;
d=zeros(L+1,3);
d(1,:)=d0;

for i=1:L
d(i+1,:)=(((T(i+n)-L)/(T(i+n)-T(i)))*C(i,:)+((L-T(i))/(T(i+n)-T(i)))*C(i
    +1,:)-((T(i+n)-G)/(T(i+n)-T(i)))*d(i,:))*((T(i+n)-T(i))/(G-T(i)));
end
```

3.9. Superficies B-spline Desarrollables

```
%Algoritmo para trazar la superficie B-spline desarrollable y texturizarla.

examples_splines;
nodos=[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15]
PB =
   [-2.2339     8.3509     2.5600
   -2.4006     9.3957     2.3450
   -2.3709    11.3267     2.0159
    1.2051    10.1227     3.0469
    0.9860    15.6913     1.9367
    1.1898    15.7434     2.7065
    1.2711    18.5277     3.1373
    1.6518    21.3388     2.0844
    1.9055    22.6085     2.5652
    2.3100    25.4903     2.8950
    2.5205    27.0643     2.6550
    2.6242    27.8830     2.4500]

[nodos,orden,c_c]=spline(PB);

if example==1
 d0=[352.5153   221.0032   119.5693];
else
 d0=[224.5402    59.6016    40.1615];
end

G_S=100; L_S=100;
d=puntosd(nodos,PB,G_S,L_S,d0,orden);
[nodos,orden,c_d]=spline(d);

figure
[x,y,z]=puntosrecta(c_c,c_d);
hold on
Textura

%% Algoritmo para obtener un reglado entre dos curvas param\'{e}tricas

function [x,y,z]=puntosrecta(c1,c2)

% Datos de entrada
% Dos curvas c1 y c2 con la misma parametrización
```

```
delta=.01;
t=0:delta:1;

for i=1:size(t,2)
    for j=1:size(c1,2)
        rectas(i,:,j)=(1-t(1,i))*c2(:,j) + t(1,i)*c1(:,j);
    end
end

x(:,:)=rectas(:,:,1);
y(:,:)=rectas(:,:,2);
z(:,:)=rectas(:,:,3);
```

Bibliografía

[1] Aumann G., A simple algorithm for designing developable Bézier surfaces, *Computer Aided Geometric Design*, Vol. 20, No. 8-9, pp 601-619, 2003.

[2] Bézier P., Style, mathematics and NC, *Computer-Aided Design*, Vol. 22 No. 9, pp 523-526, 1990.

[3] Davis, P., B-splines and geometric design, *SIAM News*, Vol. 29, No. 5, 1996.

[4] Deserno T., Biomedical Image Processing, *Springer*, 2011.

[5] Farin G., Curves and Surfaces for CGDA: A Practical Guide, *Academic Press*, 5th. Edition, 2002.

[6] Rogers D. F., An Introduction to Nurbs, *Morgan Kaufmann Publishers*, 2001.

[7] Chalfant J. S., Maekawa T., Design for manufacturing using B-spline developable surface, *Journal of Ship Production*, No. 42, pp 291-298.

[8] Pottman H., Architectural Geometry, *Bentley Institute Press*, 2007.

[9] Fernández Jambrina L., B-spline control nets for developable surfaces, *Computer Aided Geometric Design*, Vol. 24, No. 4, pp 189-196, 2007.

[10] Forrest A R., Interactive interpolation and aproximation by Bézier polynomials, *Computer-Aided Design*, Vol. 22 No. 9, pp 527-537, 1990. Originalmente publicado en *The Computer Journal* Vol.15 No. 1, pp 71-79, 1972.

[11] Lane J. M y Riesenfeld R. F., A Geometric proof for the variation diminishing property of B-spline approximation, *Journal of Approximation Theory*, Vol. 37, No. 1, pp 1-4, 1983.

[12] Saroul L., Surface extraction and flattening for anatomical visualization, Tesis Doctoral, Ecole Federale Polytechnique de Lausanne, 2006.

[13] Saroul L., Figueiredo O. y Hersch R., Distance preserving flattening of surface sections, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 12, No. 1, pp 26-35, 2006.

[14] Saroul L., Gerlach S. y Hersch R., Exploring curved anatomic structures with surface sections, IEEE Visualization, 2003.

[15] Schoenberg I., Contributions to the problem of aproximation of equidistant data by analytic functions, *Quart. Appl. Math.* Vol. 4, pp 45-99, 1946.