Universidad Politécnica de Cartagena

industriales
etsii UPCT

Generación de Curvas mediante Algoritmos Matemáticos de Interpolación y de Aproximación

Titulación: Ingeniería en Organización Industrial

Alumno/a: Antonio José Vilar Ballester

Director/a/s: Juan Carlos Trillo Moya

Cartagena, 25 de Julio de 2017

Dedicado:

A mis dos hijos Álvaro y Noa, que son ahora el motor de mi vida.

A mi mujer Virginia, por aguantarme.

A mis padres, que siempre han estado y están sin esperar nada a cambio.

A mis hermanos, por estar siempre tan unidos a pesar de tener cada uno su vida.

Y en especial a mi abuela Angelita, por esos veranos de estudios en los que me daba de comer.

Tabla de contenido

Índice de figuras

1. Introducción brazo robótico

1.1 Antecedentes

A medida que se ha ido mejorando la tecnología, se han desarrollado máquinas especializadas para distintas tareas. Sin embargo ninguna de estas máquinas tenía la versatilidad del brazo humano, y no podía alcanzar objetos alejados y colocarlos en la posición deseada. La mayoría de los robots actuales son utilizados en la industria, los cuales están formados por uno o dos brazos.

La palabra robot se define como una máquina controlada por un ordenador y programada para moverse, manipular objetos y realizar trabajos a la vez que interacciona con su entorno. Los robots son capaces de realizar tareas repetitivas de forma más rápida, barata y precisa que los seres humanos.

Los brazos robóticos han tratado de asemejar el movimiento del brazo humano, por lo que se han construido en base a las articulaciones de éstos. A través de un software el ordenador controla el robot rotando varios motores individuales (algunos brazos de robot más grandes utilizan sistemas hidráulicas). A diferencia de los motores ordinarios, este tipo de motores se mueven en incrementos exactos. Esto permite al ordenador mover el brazo de una forma muy precisa, repitiendo exactamente el mismo movimiento una y otra vez. El robot usa sensores de movimiento para asegurarse que se mueve justamente lo necesario.

En este proyecto enfoca vamos a de determinar el desplazamiento que debe tener un brazo robótico y cómo simularlo a través de una interfaz gráfica creada con el programa Matlab.
En el movimiento del brazo robótico, la posición queda descrita por un conjunto de parámetros de configuración, determinados por los ángulos de sus articulaciones y la posición de las mismas en el espacio utilizando marcos de referencia previamente definidos. En consecuencia, es habitual que la manipulación de objetos se base en la planificación de trayectorias en el espacio articular, que es computacionalmente mucho menos costosa que la planificación en el espacio cartesiano. Dichas trayectorias suelen expresarse como secuencias de configuraciones muy próximas entre sí, y el cálculo de las mismas es complicado.

Aplicaciones:

- Utilización en el sector industrial como por ejemplo; en moldeo por inyección, CNC, empaquetado y paletizado, montaje, supervisión de máquinas, etc.

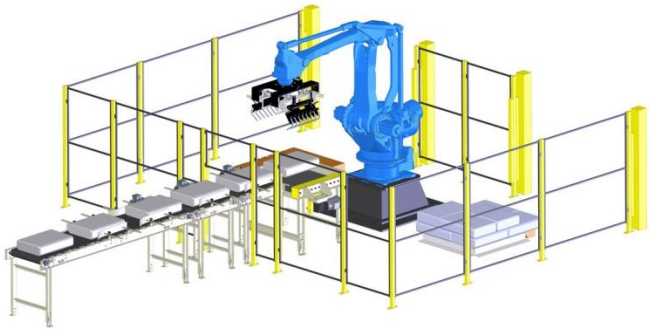

Figura 1.1 Brazo robótico, paletizador.

(Fuente: ihopereal.com)

Figura 1.2 Brazo robótico, cadena de montaje

(Fuente: www.emaze.com).

- Utilización en infraestructuras para mejorar la calidad de entornos abiertos: así, por ejemplo se aplicarían para procedimientos de limpieza en vías públicas, mantenimiento de zonas verdes, en sistemas de seguridad ciudadana, desactivación de explosivos y/o detección de los mismos, etc.

Figura 1.3 Brazo robótico, antiexplosivos.

(Fuente: www.popsci.com).

- Utilización en entornos cerrados habituales. En este ámbito de aplicación se encuentra un amplio sector poblacional, en el que se incluyen niños, ancianos, personas con problemas cognitivos y discapacitados, susceptibles de precisar ayudas más o menos permanentes. Su papel podrá ser asistencial, de rehabilitación, de teleasistencia, etc. De esta forma, tareas como limpiarse las gafas, comer o lavarse los dientes podrán ser ejecutadas por el dispositivo robótico.

Figura 1.4 Brazo robótico, rehabilitación.

(Fuente: roboticaensalud.blogspot.com.es).

1.2 Estructura del proyecto

En este apartado explicaremos como está estructurado el proyecto capítulo por capítulo.

En el Capítulo 2, daremos una explicación matemática de los splines cúbicos interpolantes.

En el Capítulo 3, la explicación matemática será de los splines cúbicos suavizantes [4].

En el Capítulo 4, hablaremos cómo se generan curvas a través de los splines cúbicos [3].

En el Capítulo 5, se explica cómo manejar la interfaz gráfica creada.

En el Capítulo 6, se exponen dos casos reales en los que se puede utilizar la interfaz gráfica creada.

En el Capítulo 7, explicamos las conclusiones tras la realización del proyecto.

En el Capítulo 8, enumeramos la bibliografía utilizada en el proyecto.

En el Capítulo 9, se muestra el código de programación de Matlab de la interfaz gráfica.

2. Splines cúbicos interpolantes

2.1 Planteamiento del problema

El problema de interpolación que planteamos a continuación consiste en construir en el intervalo [a,b] una función suave $\sigma(x)$ que coincide en los nodos del retículo w con una función dada f(x), la cual no posee la suavidad exigida.
Supongamos que en un intervalo [a,b] está definido un retículo

$$w: a = t_1 < t_2 < \cdots < t_{m-1} < t_m = b.$$

Consideremos la colección de números

$$y_1, y_2, \dots, y_{m-1}, y_m.$$

Construimos una función $\sigma(x)$ suave en el intervalo [a,b], que en los nodos del retículo w toma los valores dados, es decir:

$$\sigma(t_i) = y_i \ \forall \ i = 1, \dots, m-1, m.$$

Está claro que planteado así, este problema tendrá infinitas soluciones diferentes pero, imponiendo a la función $\sigma(x)$ condiciones adicionales, es posible lograr que el problema tenga solución única.

2.2 Definición

Se llama Spline cúbico interpolante S(x) en un retículo w a la función que:

1) En cada uno de los intervalos $[t_i, t_{i+1}] \ \forall \ i = 1, .., m-1$ es un polinomio de tercer grado,

$$S(x) = S_i(x) = a_0^{(i)}(x - t_i) + a_2^{(i)}(x - t_i)^2 + a_3^{(i)}(x - t_i)^3.$$

   En todo el intervalo, el Spline es un polinomio de tercer grado que se define mediante cuatro coeficientes. En total hay (m-1) intervalos, por lo que, para definir completamente el Spline es necesario hallar 4(m-1) números:

$$a_1^{(i)}, a_2^{(i)}, a_3^{(i)}, \qquad \forall \ i = 1, \dots, m-1.$$

2) Es dos veces diferenciable con continuidad en el intervalo [a,b], es decir, pertenece a la clase $C^2[a, b]$.

   Esta condición significa continuidad en la función $S(x)$ y de sus derivadas $S'(x)$ y $S''(x)$ en todos los nodos internos del retículo w. Como el número de nodos internos es m-2, entonces tenemos 3(m-2) condiciones más.

3) Satisface las condiciones

$$S(t_i) = f(t_i) = y_i \qquad \forall\ i = 1, \dots, m.$$

Junto con las condiciones de continuidad, tenemos 3(m-2)+,=4m-6 condiciones (ecuaciones).

2.3 Condiciones de contorno

Para tener las $4(m-1) = 4m-4$ condiciones necesarias para la definición unívoca del Spline, nos faltan dos ecuaciones, las cuales se formulan como restricciones sobre los valores del Spline y/o sus derivadas en los extremos del intervalo [a,b].
Generalmente para la construcción de un Spline cúbico interpolante se utilizan condiciones de entre los cuatro tipos siguientes:

A) Condiciones de primer tipo: Se dan los valores que debe tomar la derivada primera de $S(x)$ en los extremos del intervalo [a,b]:

$$S'(a) = f'(a) \quad ; \quad S'(b) = f'(b)$$

B) Condiciones de segundo tipo: Se dan los valores que debe tomar la derivada segunda de $S(x)$ en los extremos del intervalo [a,b]:

$$S''(a) = f''(a) \quad ; \quad S''(b) = f''(b)$$

C) Condiciones de tercer tipo: Condiciones periódicas

$$S'(a) = S'(b) \quad ; \quad S''(a) = S''(b)$$

Es natural imponer este tipo de condiciones cuando la función a interpolar es periódica de periodo $T = b - a$.

D) Condiciones de cuarto tipo:

$$S'''(y, t_2 - 0) = S'''(y, t_2 + 0)$$

$$S'''(y, t_{m-1} - 0) = S'''(y, t_{m-1} + 0)$$

En los puntos interiores del retículo, la derivada tercera de $S(x)$ es, en general, discontinua. Sin embargo, el número de puntos de discontinuidad puede ser reducido con ayuda de este tipo de condiciones.

En este caso, el Spline obtenido es tres veces diferenciable con continuidad en los puntos $t_2\ y\ t_{m-1}$ y acercándose por la izquierda (-0) o acercándose por la derecha (+0).

Teorema: El Spline cúbico interpolante que satisface las condiciones

$$S(t_i) = y_i\ \forall\ i = 1, \dots, m$$

y una condición de contorno cualquiera dentro de los cuatro tipos enumerados existe y es único.
En el apartado siete de este capítulo comprobaremos que este teorema es cierto y que además pueden tomarse condiciones de contorno mixtas (una en cada extremo), exceptuando

para el caso de condiciones de tercer tipo y para el caso de condiciones de cuarto tipo con menos de cuatro nodos; existiendo el Spline cúbico interpolante con solución única.

2.4 Elección de las condiciones de contorno

La elección de las condiciones de contorno es uno de los problemas principales en la interpolación, y adquiere una importancia especial cuando es necesario garantizar una precisión alta del Spline S(x) en las proximidades de los extremos del intervalo [a,b]. Las condiciones de contorno ejercen una influencia visible en el comportamiento del Spline cerca de los extremos a y b, y dicha influencia se va atenuando según nos alejamos de ellos. La elección de las condiciones de contorno depende, a menudo, de la existencia de datos adicionales sobre el comportamiento del Spline.

Si se conocen los valores de la derivada primera $f'(x)$ en los extremos del intervalo [a,b], es decir, se conoce la dirección de la tangente de la curva en los extremos; entonces es lógico utilizar condiciones de contorno de primer tipo.

Si por el contrario, lo que se conocen son los valores de la derivada segunda $f''(x)$; entonces es lógico utilizar condiciones de contorno de segundo tipo.

Si existe la posibilidad de elegir entre condiciones de primer y segundo tipo, se dará preferencia a las primeras.

Si la función es periódica, se deben elegir condiciones de contorno de tercer tipo.

En caso de no existir información adicional sobre el comportamiento de la función, se pueden utilizar las condiciones naturales de contorno:

$$S''(a) = 0 \quad ; \quad S''(b) = 0$$

Con estas condiciones la precisión en la aproximación puede disminuir bruscamente cerca de los extremos del intervalo [a,b].

Otra opción es utilizar las condiciones de contorno de primer o de segundo tipo con valores aproximados; esto quiere decir, que se tomarán sus aproximaciones en diferencias, en vez de los valores exactos de sus derivadas. La experiencia muestra que, en ocasiones, elegir las condiciones de contorno de cuarto tipo da buenas aproximaciones.

2.5 Elección de nodos de interpolación

Si la derivada tercera $f'''(x)$ tiene discontinuidades en algunos puntos del intervalo, para mejorar la aproximación, estos puntos deben ser incluidos entre los nodos de interpolación.

Si la derivada segunda $f''(x)$ es discontinua, entonces es necesario tomar medidas especiales para evitar oscilaciones del Spline cerca de los puntos de discontinuidad. Generalmente, los nodos de interpolación se eligen de tal manera que los puntos de la discontinuidad de la derivada segunda estén contenidos en un intervalo $[t_i, t_{i+1}]$ tal que: $h_i = \alpha \min\{h_{i-1}, h_{i+1}\}$ donde $\alpha \ll 1$. El número α puede ser elegido empíricamente, pero a menudo es suficiente tomar $\alpha = 0{,}01$.

Cuando las discontinuidades se presentan en la primera derivada $f'(x)$, existen varias técnicas para superar las dificultades que surgen. Una de las técnicas más simples consiste en partir el intervalo de aproximación en intervalos de continuidad de la derivada y construir en cada uno de ellos un Spline.

2.6 Construcción del Spline cúbico interpolante

Vamos a interpolar una función $f(x)$ $t_i \; \forall \; i = 1, \dots, m$ sobre unos nodos mediante un Spline cúbico interpolante, siendo $t_1 = a, t_m = b$. Denotamos por $y_i = f(t_i)$ los valores conocidos de la función en los nodos. Denominamos $h_i = t_{i+1} - t_i$ a los diferentes espaciados entre los nodos. Sea $S_i(x)$ la restricción en la función Spline $S(x)$ al intervalo$[t_i, t_{i+1}] \; \forall \; i = 1, \dots, m-1$. Como la función $S(x)$ está definida en [a,b] como: $S(x) = S_i(x), si \; x \in [t_i, t_{i+1}]$, bastará conocer cada uno de los trozos cúbicos $S_i(x)$.

Recodamos que es un polinomio cúbico que satisface las condiciones:

$$S_i(t_i) = y_i \quad \forall \; i = 1, \dots, m$$

y la condición $S_i(x) \; \in \; C^2[a, b]$, es decir:

$$S_i'(t_i) = S_{i+1}'(t_i) \; \forall \; i = 1, \dots, m-1,$$

$$S_i''(t_i) = S_{i+1}''(t_i) \; \forall \; i = 1, \dots, m-1,$$

Al ser $S_i(x)$ un polinomio de grado tres, su derivada será de grado dos y su segunda derivada será de grado uno.

Si denotamos por $z_i \; \forall \; i = 1, \dots, m$ los valores que toma $S_i''(t_i)$ para cada i, entonces para que las derivadas segundas peguen bien entre los diferentes trozos de la función $S''(x)$ tendrá que ser del estilo de la siguiente figura:

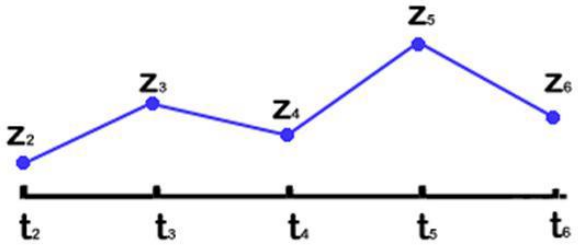

Figura 2.1 Derivadas segundas

Para que esto sea así, basta con construir cada trozo $S_i''(x)$ lineal de manera que $S_i''(t_i) = z_i \; y \; S_{i+1}''(t_{i+1})$. Usando la interpolación de Lagrange tenemos:

$$S_i''(x) = z_{i+1}\frac{x - t_i}{h_i} + z_i\frac{t_{i+1} - x}{h_i}.$$

De esta expresión, si integramos dos veces, obtenemos:

$$S_i(x) = \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \frac{z_i}{6h_i}(t_i - x)^3 + C_i(x - t_i) + D_i(t_{i+1} - x).$$

De las condiciones $S(t_i) = y_i \ \forall \ i = 1, \dots, m$ sabemos que: $S(t_i) = y_i \ y \ S_i(t_{i+1}) = y_{i+1}$

A partir de estas condiciones podemos deducir el valor de C_i y D_i, llegando hasta la expresión:

$$S_i(x) = \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \frac{z_i}{6h_i}(t_{i+1} - x)^3 + \left(\frac{y_{i+1}}{h_i} - \frac{z_{i+1}h_i}{6}\right)(x - t_i) + \left(\frac{y_i}{h_i} - \frac{z_i h_i}{6}\right)(t_{i+1} - x).$$

Con esta expresión de $S_i(x)$, para cada intervalo $[t_i, t_{i+1}]$ se garantiza la continuidad de S(x) y que coincidirá con $f(t_i)$ en los puntos $t_i \ \forall = 1, \dots, m$.
Queda por satisfacer la continuidad de la derivada primera de S(x):

$$S_i'(t_i) = S_{i-1}'(t_i) \ \forall \ i = 2, \dots, m - 1$$

Calculamos $S_i'(t_i)$ y $S_{i-1}'(t_i)$:

$$S'_i(x) = \frac{z_{i+1}}{2h_i}(x - t_i)^2 + \frac{z_i}{2h_i}(t_{i+1} - x)^2 + \left(\frac{y_{i+1}}{h_i} - \frac{z_{i+1}h_i}{6}\right) - \left(\frac{y_i}{h_i} - \frac{z_i h_i}{6}\right).$$

$$S'_{i-1}(x) = \frac{z_i}{2h_i}(x - t_{i-1})^2 + \frac{z_{i-1}}{2h_{i-1}}(t_i - x)^2 + \left(\frac{y_i}{h_{i-1}} - \frac{z_i h_{i-1}}{6}\right) - \left(\frac{y_{i-1}}{h_{i-1}} - \frac{z_{i-1}h_{i-1}}{6}\right).$$

Imponiendo $S_i'(t_i) = S_{i-1}'(t_i)$, simplificando la expresión y teniendo en cuenta que i=2,…, m-1, llegamos al sistema de ecuaciones lineales:

$$h_{i-1}z_{i-1} + 2(h_{i-1} + h_i)z_i + h_i z_{i+1} = \frac{6}{h_i}(y_{i+1} - y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1}).$$

Si definimos:

$$b_i = \frac{6}{h_i}(y_{i+1} - y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1}) \ \forall \ i = 2, \dots, m - 1,$$

Entonces el sistema queda:

$$\begin{cases} h_1 z_1 + 2(h_1 + h_2)z_2 + h_2 z_3 = b_2 \\ h_2 z_2 + 2(h_2 + h_3)z_3 + h_3 z_4 = b_3 \\ \dots \\ h_{m-2}z_{m-2} + 2(h_{m-2} + h_{m-1})z_{m-1} + h_{m-1}z_m = b_{m-1} \end{cases}$$

Se trata de (m-2) ecuaciones lineales con m incógnitas y por tanto nos quedan dos variables libres. Para obtenerlas tenemos que aplicar condiciones de contorno.

Como hemos explicado en el apartado de “Elección de las condiciones de contorno”, éstas se escogen dependiendo de los datos adicionales que tengamos sobre el comportamiento de la

función f(x). A continuación vamos a detallar la construcción de las dos ecuaciones necesarias para poder resolver el sistema, según el tipo de condiciones de contorno que utilicemos, pudiendo estas ser distintas en cada extremo.

A) En caso de conocer condiciones de primer tipo, es decir, los valores de la primera derivada de S(x) en los extremos del intervalo [a,b]; se obtendrían las dos ecuaciones que nos faltan para completar el sistema del siguiente modo:

- Obtenemos la derivada del trozo de Spline correspondiente al extremo.
- Evaluamos esta ecuación en punto extremo.
- Y lo igualamos al valor conocido de la primera derivada.

1) Para el extremo izquierdo (a):

$$S'_i(t_i) = -\frac{z_1}{2h_i}(t_2 - t_1)^2 + \frac{y_2}{h_1} - \frac{z_2 h_1}{6} - \frac{y_1}{h_1} + \frac{z_1 h_1}{6} = f'(a).$$

Ordenando las componentes:

$$-\frac{h_1}{3}z_1 - \frac{h_1}{6}z_2 = f'(a) - \frac{y_2 - y_1}{h_1}.$$

2) Para el extremo derecho (b):

$$S'_{m-1}(t_m) = \frac{z_m}{2h_{m-1}}(t_m - t_{m-1})^2 + \frac{y_m}{h_{m-1}} - \frac{z_m h_{m-1}}{6} - \frac{y_{m-1}}{h_{m-1}} + \frac{z_{m-1}h_{m-1}}{6} = f'(b).$$

Ordenando las componentes:

$$\frac{h_{m-1}}{3}z_m - \frac{h_{m-1}}{6}z_{m-1} = f'(b) - \frac{y_m - y_{m-1}}{h_{m-1}}.$$

B) Si utilizamos condiciones de segundo tipo, es decir, damos los valores de la segunda derivada en los extremos del intervalo $[a, b]$; básicamente se han de dar los valores de z_1 y z_m:

$$f''(a) = z_1 \; ; \; f''(b) = z_m.$$

C) En caso de que la función a suavizar sea periódica de periodo $T = b - a$, en particular, $f(a) = f(b)$ y por tanto $S_1(a) = S_{m-1}(b)$; se utilizarán condiciones de tercer tipo, las cuales darán las siguientes ecuaciones:

1) $S'_1(t_1) = S'_{m-1}(t_m)$:

$$\frac{h_1}{3}z_1 + \frac{h_1}{6}z_2 + \frac{h_{m-1}}{6}z_{m-1} + \frac{h_{m-1}}{3}z_m += \frac{y_{m-1} - y_m}{h_{m-1}} + \frac{y_2 - y_1}{h_1}.$$

2) $S_1''(t_1) = S_{m-1}''(t_m)$:

$$z_1 - z_m = 0$$

D) Si deseamos utilizar condiciones de cuarto tipo, con continuidad también en la derivada tercera de S(x) en los extremos, de forma que $S_1 = S_2$, $S_{m-2} = S_{m-1}$ sabiendo que se satisface $S_i(x) \in C^2[a,b]$; entonces debemos igualar $S_{i-1}'''(t_i)$ a $S_i'''(t_i) \quad \forall \quad i = 2, m-1$. Los Splines así construidos se suelen denominar *Splines no nodo*.

Sabemos que $S_i'''(x)$ es una recta que pasa por los puntos (t_{i-1}, z_{i-1}) y (t_i, z_i), entonces su pendiente y por tanto derivada será un coeficiente tal que:

$$\frac{z_{i+1} - z_1}{t_i - t_{i-1}} = \frac{z_{i+1} - z_i}{h_i}$$

1) En el extremo izquierdo:

$$S_1'''(t_2) = S_2'''(t_2)$$

$$\frac{z_2 - z_1}{h_1} = \frac{z_3 - z_2}{h_2}$$

Ordenando las componentes:

$$-h_2 z_1 + (h_2 + h_1) z_2 - h_1 z_3 = 0$$

2) En el extremo derecho:

$$S_{m-1}'''(t_{m-1}) = S_{m-2}'''(t_{m-1})$$

$$\frac{z_m - z_{m-1}}{h_{m-1}} = \frac{z_m - z_{m-2}}{h_{m-2}}$$

Ordenando las componentes:

$$-h_{m-1} z_{m-2} + (h_{m-1} + h_{m-2}) z_{m-1} - h_{m-2} z_m = 0$$

No siempre conocemos los mismos datos en ambos extremos del intervalo [a,b], por lo que podemos tener diferentes condiciones en cada extremo, excepto en las condiciones de tercer tipo que exigen que la función sea periódica de periodo T=b-a y en particular $f(a) = f(b)$.

Recordamos que las condiciones de primer tipo (primera derivada conocida) tienen preferencia sobre las de segundo tipo (segunda derivada conocida) y que en caso de no conocer los valores de estas derivadas, se pueden aproximar. Incluso si tampoco se pueden aproximar, podemos asumir $z_1 = z_m = 0$, en detrimento de la precisión de aproximación en caso de que la segunda derivada no fuera nula; los Splines cúbicos así construidos se denominan *naturales*. Optar por usar condiciones de cuarto tipo cuando se desconocen las demás suele dar buenos resultados.

2.7 Comprobación de sistema compatible determinado

2.6.1 Introducción

En este capítulo nos vamos a centrar en demostrar que cada uno de los sistemas que se forman incluyendo las condiciones de contorno es compatible determinado y por tanto no tendremos problemas a la hora de obtener el Spline correspondiente.

Para abreviar se utilizarán las siguientes denominaciones:

1: condiciones de primer tipo (se conoce el valor de la primera derivada en el extremo).
2: condiciones de segundo tipo (se conoce el valor de la segunda derivada en el extremo).
3: condiciones de tercer tipo (existe periocidad).
4: condiciones de cuarto tipo (existe tercera derivada continua en el extremo).
CI: condición de contorno en el extremo izquierdo.
CD: condición de contorno en el extremo derecho.

Si construimos el sistema completo Az=B para que éste tenga solución compatible determinada sólo habrá que demostrar que la matriz A es invertible, es decir, que el determinante de A no es nulo.

Formalmente, se dice que la matriz A de orden m es *estrictamente diagonal dominante* cuando se satisface:

$$|a_{i,i}| > \sum_{\substack{j=1 \\ j\neq i}}^{m} |a_{i,j}| \quad \forall\, i = 1, \dots, m$$

Enunciado del lema de Hadamard: Si $A = \left((a_{i,j})i,j\right)\ \forall i = 1, \dots, m$ es una matriz de estrictamente diagonal dominante, entonces A es invertible.

Demostración: Por contrarrecíproco. Supongamos que A no es invertible, entonces su núcleo no se reduce a cero, existe entonces un vector:

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} \neq 0$$

tal que AX=0. Entonces, se tiene que:

$$\forall\, i =, \dots, m; \qquad \sum_{j=1}^{m} a_{i,j} x_j = 0$$

Como $X \neq 0$, existe $x_{i_0} \neq 0$ tal que $|x_{i_o}| max\{|x_i|\} \quad \forall\, i = 1, .. m.$

Tenemos:

$$-a_{i_o,i_0}x_{i_0} = \sum_{\substack{j=1 \\ j\neq i_0}}^{m} a_{i_0,j}x_{j,}$$

de donde:

$$\left|a_{i_o,i_0}x_{i_0}\right| \leq \sum_{\substack{j=1 \\ j\neq i_0}}^{m} \left|a_{i_0,j}x_j\right|,$$

y como:

$$\forall\, j = 1, \dots, m\,; \quad \frac{|x_j|}{|x_{i_0}|} \leq 1$$

se obtiene:

$$\left|a_{i_o,i_0}\right| \leq \sum_{\substack{j=1 \\ j\neq i_0}}^{m} \left|a_{i_0,j}\right| \frac{|x_j|}{|x_{i_0}|} \leq \sum_{\substack{j=1 \\ j\neq i_0}}^{m} \left|a_{i_0,j}\right|$$

Finalmente:

$$\left|a_{i_o,i_0}\right| \leq \sum_{\substack{j=1 \\ j\neq i_0}}^{m} \left|a_{i_0,j}\right|$$

contradicción con la que culmina la demostración.

2.6.2 Caso CI=1, CD=1

Las condiciones en los extremos son de primer tipo, por lo que se conocen las primeras derivadas y tenemos las dos siguientes ecuaciones adicionales:

$$-\frac{h_1}{3}z_1 - \frac{h_2}{6}z_2 = f'(a) - \frac{y_2 - y_1}{h_1},$$

$$-\frac{h_{m-1}}{3}z_m - \frac{h_{m-1}}{6}z_{m-1} = f'(b) - \frac{y_m - y_{m-1}}{h_{m-1}}.$$

Por tanto la matriz quedará tal que:

$$\begin{pmatrix} \frac{-h_1}{3} & \frac{-h_1}{6} & 0 & 0 & \dots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & \ddots & 0 \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & h_{m-2} & 2(h_{m-2}+h_{m-1}) & h_{m-1} \\ 0 & \dots & 0 & 0 & \frac{h_{m-1}}{6} & \frac{h_{m-1}}{3} \end{pmatrix}$$

Como se puede comprobar fácilmente la matriz es estrictamente diagonal dominante y por tanto será invertible.

2.6.3 Caso CI=1, CD=2

La condición en el extremo izquierdo es de primer tipo y en el extremo derecho es de segundo tipo, por lo que se conoce la primera derivada en un extremo y la segunda derivada en el otro, así que tenemos las dos siguientes ecuaciones adicionales:

$$-\frac{h_1}{3}z_1 - \frac{h_2}{6}z_2 = f'(a) - \frac{y_2 - y_1}{h_1},$$

$$z_m = f''(b).$$

Por lo tanto la matriz se quedará tal que:

$$\begin{pmatrix} \frac{-h_1}{3} & \frac{-h_1}{6} & 0 & 0 & \dots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & \ddots & 0 \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & h_{m-2} & 2(h_{m-2}+h_{m-1}) & h_{m-1} \\ 0 & \dots & 0 & 0 & 0 & 1 \end{pmatrix}$$

Como se puede comprobar la matriz será invertible, puesto que es estrictamente diagonal dominante.

2.6.4 Caso CI=1, CD=4

La condición en el extremo izquierdo es de primer tipo y en el extremo derecho es de cuarto tipo, por lo que se conoce la primera derivada en un extremo y que el otro extremo también es continua su tercera derivada, así que tenemos las dos siguientes ecuaciones adicionales:

$$-\frac{h_1}{3}z_1 - \frac{h_2}{6}z_2 = f'(a) - \frac{y_2 - y_1}{h_1},$$

$$-h_{m-1}z_{m-2} + (h_{m-1}h_{m-2})z_{m-1} - h_{m-2}z_m = 0.$$

Por tanto la matriz quedará tal que:

$$\begin{pmatrix} \frac{-h_1}{3} & \frac{-h_1}{6} & 0 & 0 & \dots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & \ddots & 0 \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & h_{m-2} & 2(h_{m-2}+h_{m-1}) & h_{m-1} \\ 0 & \dots & 0 & -h_{m-1} & h_{m-2}+h_{m-1} & -h_{m-2} \end{pmatrix}$$

Si la última fila la sustituimos por la penúltima menos la última, obtendremos la siguiente matriz equivalente:

$$\sim \begin{pmatrix} \frac{-h_1}{3} & \frac{-h_1}{6} & 0 & 0 & \dots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & \ddots & 0 \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & h_{m-2} & 2(h_{m-2}+h_{m-1}) & h_{m-1} \\ 0 & \dots & 0 & h_{m-2}+h_{m-1} & h_{m-2}+h_{m-1} & h_{m-2}+h_{m-1} \end{pmatrix}$$

Ahora a la columna le restamos la columna y a su vez, a la columna le restamos la columna multiplicada por un coeficiente. Por lo que nos quedará la siguiente matriz equivalente:

$$\sim \begin{pmatrix} \frac{-h_1}{3} & \frac{-h_1}{6} & 0 & 0 & \dots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & \ddots & 0 \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & h_{m-2}-h_{m-1} & 2(h_{m-2}+h_{m-1})-\mu h_{m-1} & h_{m-1} \\ 0 & \dots & 0 & 0 & (1-\mu)(h_{m-2}+h_{m-1}) & h_{m-2}+h_{m-1} \end{pmatrix}$$

Con 0 < μ < 1 esta matriz cumple con el enunciado cuando también $\mu < {}^{h_{m-2}}/_{h_{m-1}}$ ya que

$$|h_{m-2} + h_{m-1}| = h_{m-2} + h_{m-1} > (1-\mu)(h_{m-2} + h_{m-1})$$

Y

$$|2(h_{m-2} + h_{m-1} - \mu h_{m-1})| = 2(h_{m-2} + h_{m-1}) - \mu h_{m-1} > h_{m-2} + 2h_{m-1} \geq |h_{m-2} - h_{m-1}| + |h_{m-1}|.$$

Así se observa fácilmente que es una matriz estrictamente diagonal dominante y por consiguiente el sistema tendrá solución única.

2.6.5 Caso CI=3, CD=3

Para este caso las ecuaciones adicionales se obtienen considerando que la función a suavizar es periódica de periodo $T = b - a$, de modo que aplicando las condiciones de tercer tipo se obtendrán las siguientes ecuaciones:

$$z_1 - z_m = 0,$$

$$\frac{h_1}{3}z_1 + \frac{h_1}{6}z_2 + \frac{h_{m-1}}{6}z_{m-1} + \frac{h_{m-1}}{3}z_m = \frac{y_{m-1} - y_m}{h_{m-1}} + \frac{y_2 - y_1}{h_1}.$$

Construyendo el sistema con estas ecuaciones, la matriz quedará tal que:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \dots & -1 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & \ddots & 0 \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & h_{m-2} & 2(h_{m-2}+h_{m-1}) & h_{m-1} \\ \frac{h_1}{3} & \frac{h_1}{6} & \dots & 0 & \frac{h_{m-1}}{6} & \frac{h_{m-1}}{3} \end{pmatrix}$$

Si la última columna la sustituimos por su suma con la primera, tendremos la siguiente matriz equivalente:

$$\sim \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & \ddots & h_1 \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & h_{m-2} & 2(h_{m-2}+h_{m-1}) & h_{m-1} \\ \frac{h_1}{3} & \frac{h_1}{6} & \dots & 0 & \frac{h_{m-1}}{6} & \frac{h_{m-1}+h_1}{3} \end{pmatrix}$$

Finalmente, sustituimos la última fila por ella misma menos la primera multiplicada por $\frac{h_1}{3}$, de modo que nos quedará la siguiente matriz equivalente:

$$\sim \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & \ddots & h_1 \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & h_{m-2} & 2(h_{m-2}+h_{m-1}) & h_{m-1} \\ 0 & \frac{h_1}{6} & \dots & 0 & \frac{h_{m-1}}{6} & \frac{h_{m-1}+h_1}{3} \end{pmatrix}$$

y esta matriz es claramente estrictamente diagonal dominante, por lo que será invertible y el correspondiente sistema tendrá solución única.

2.6.6 Otros casos

Tenemos seis posibles combinaciones además de las ya estudiadas, pero, puesto que serán las mismas sólo que variando una de las dos ecuaciones frontera, se desarrollarán de igual modo demostrando que son matrices estrictamente diagonales dominantes y según el enunciado del lema de Hadamard serán por tanto, invertibles y su sistema tendrá solución compatible determinada.

Es decir, utilizando correctamente las condiciones de contorno podremos construir el Spline a partir del sistema construido según se ha explicado en este capítulo.

3. Splines cúbicos suavizantes

3.1 Planteamiento del problema

Consideramos el retículo

$$a = x_1 < x_2 < \cdots < x_{m-1} < x_m = b$$

Y la colección de números

$$y_1, y_2, \dots, y_{m-1}, y_m$$

Puede ser que los valores y_i en el arreglo

$$(x_i, y_i), \quad \forall \quad i = 1, \dots, m$$

Contengan cierto error. Esto significa para todo $i = 0,1, \dots, m$ existe un intervalo

$$(c_I, d_I) \text{ ó } bien \quad (y_{I-\delta}, y_{I+\delta})$$

tal que cualquier número perteneciente a él puede ser tomado como valor de y_i. Los valores y_i pueden ser, por ejemplo, los resultados (que contienen un error aleatorio) de las mediciones de cierta función $y(x)$ para los valores dados de la variable x. No es conveniente utilizar interpolación para construir la función $y(x)$ a partir de estos valores experimentales, por cuanto la función interpolante reproducirá las oscilaciones condicionadas por la componente aleatoria en el arreglo $\{z_i\}$.

En este caso es más apropiado el método de suavización, uno de cuyos objetivos es disminuir la aleatoriedad en el resultado de las mediciones. Habitualmente, en tales problemas se pide hallar una función cuyos valores para $x = x_i$, $i = 1,2, \dots, m$,pertenezcan a los intervalos correspondientes y que tenga, además propiedades bastante buenas (por ejemplo, derivadas primera y segunda continuas, su gráfico no es muy encorvado, es decir no tiene oscilaciones muy fuertes, etcétera).

Un problema similar tiene lugar también al construir, a partir de un arreglo (exacto)

$$(x_i, y_i), \quad \forall \quad i = 1, \dots, m,$$

una función que pase no por los puntos dados, sino cerca de ellos y que, además, varíe de manera suficientemente suave. En otras palabras, es como si la función buscada suavizara el arreglo sin interpolarlo.

3.2 Definición

Consideremos un retículo

$$\omega \ : \ a = x_1 < x_2 < \cdots < x_{m-1} < x_m = b,$$

y dos colecciones de números

$$y_1, y_2, \dots, y_{m-1}, y_m$$

Se denomina spline cúbico suavizante en un retículo ω a una función S(x) que:

1) En todo el intervalo

$$[x_i, x_{i+1}], \qquad \forall \quad i = 1, \dots, m-1,$$

es un polinomio de tercer grado

$$S(x) = S_i(x) = a_0^i + \ a_1^i \ (x - x_i) + \ a_2^i \ (x - x_i)^2 + \ a_3^i \ (x - x_i)^3 \ . \qquad (3.1)$$

En todo el intervalo, el spline es un polinomio de tercer grado que se define mediante cuatro coeficientes. En total hay $(m-1)$ intervalos, por lo que, para definir completamente el Spline es necesario hallar $4(m-1)$ números:

$$a_0^{(i)}, a_1^{(i)}, a_2^{(i)}, a_3^{(i)}, \quad \forall\, i = 1, \dots, m-1.$$

2) Es dos veces diferenciable con continuidad en el intervalo [a,b], es decir, pertenece a la clase C^2 [a,b].

Esta condición significa continuidad de la función $S(x)$ y de sus derivadas $S'(x)$ y $S''(x)$ en todos los nodos internos del retículo w. Como el número de nodos internos es $m-2$, entonces tenemos $3(m-2)$ condiciones.

3) En ella alcanza su mínimo el funcional

$$J(f) = \int_a^b (f''(x))^2 dx \ + \ \sum_{i=0}^{m} \frac{1}{\rho_i} (f(x_i) - y_i)^2, \qquad (3.2)$$

donde y_i y $\rho_i > 0$ son números dados. A los números ρ_i se les denomina pesos.

4) Satisface condiciones de contorno de uno de los tres tipos siguientes, explicadas a continuación.

3.3 Condiciones de contorno

Las condiciones de contorno se dan en forma de restricciones sobre los valores del spline y de sus derivadas en todos los nodos fronterizos del retículo ω.

A) Condiciones de contorno de primer tipo: Las derivadas primeras de $S(x)$ son conocidas en los extremos del intervalo [a,b]:

$$S'(a) = z_1' \quad S'(b) = z_m'. \tag{3.3}$$

B) Condiciones de contorno de segundo tipo: Las derivadas segundas de $S(x)$ son conocidas en los extremos del intervalo [a,b]:

$$S''(a) = z_1'' \quad S''(b) = z_2''. \tag{3.4}$$

C) Condiciones de contorno de tercer tipo:

$$S(a) = S(b),\ S'(a) = S'(b),\ S''(a) = S''(b). \tag{3.5}$$

Estas condiciones se denominan periódicas.

Teorema: El Spline cúbico $S(\mathrm{x})$ que minimiza el funcional y satisface las condiciones de contorno de uno de los tres tipos indicados está definido unívocamente.

Por último el spline cúbico que minimiza el funcional J(f) y satisfice las condiciones de contorno de i-ésimo tipo se denomina spline suavizador de i-ésimo tipo.

3.4 Elección de las condiciones de contorno

La elección de las condiciones de contorno es uno de los problemas principales en los problemas de interpolación y aproximación mediante splines, y adquiere una importancia especial cuando es necesario garantizar una precisión alta del spline $S(\mathrm{x})$ en las proximidades de los extremos del intervalo $[a, b]$.

El efecto sobre la función spline en función de las condiciones de contorno elegidas, disminuirá conforme estemos más cerca de los puntos intermedios a interpolar, sin embargo nuestra función dependerá en los extremos en gran medida de las condiciones elegidas. La elección de las condiciones de contorno depende, a menudo, de la existencia de datos adicionales sobre el comportamiento de la función spline.

Si se conocen los valores de la derivada primera $f'(x)$ en los extremos del intervalo $[a, b]$, es decir, se conoce la dirección de la tangente de la curva en los extremos; entonces es lógico utilizar condiciones de contorno de primer tipo.

Si por el contrario, lo que se conocen son los valores de la derivada segunda $f''(x)$; entonces es lógico utilizar condiciones de contorno de segundo tipo.

Si existe la posibilidad de elegir entre condiciones de primer y segundo tipo, se dará preferencia a las primeras.

Si la función es periódica, se deben elegir condiciones de contorno de tercer tipo.

En caso de no existir información adicional sobre el comportamiento de la función, se pueden utilizar las condiciones naturales de contorno:

$$S''(a) = 0 \quad ; \quad S''(b) = 0.$$

Con estas condiciones la precisión en la aproximación puede disminuir bruscamente cerca de los extremos del intervalo $[a, b]$.

Otra opción es utilizar las condiciones de contorno de primer o de segundo tipo con valores aproximados; esto quiere decir, que se tomarán sus aproximaciones en diferencias, en vez de los valores exactos de sus derivadas.

3.5 Elección de los coeficientes de peso

La elección de los coeficientes de peso ρ_i del funcional permite controlar, en cierta medida, las propiedades de los splines suavizantes.

Para ilustrar cómo influye este parámetro de peso o también llamado de suavizado, consideraremos una serie de puntos pertenecientes a un retículo, los cuales serán ajustados mediante 6 valores distintos de ρ.

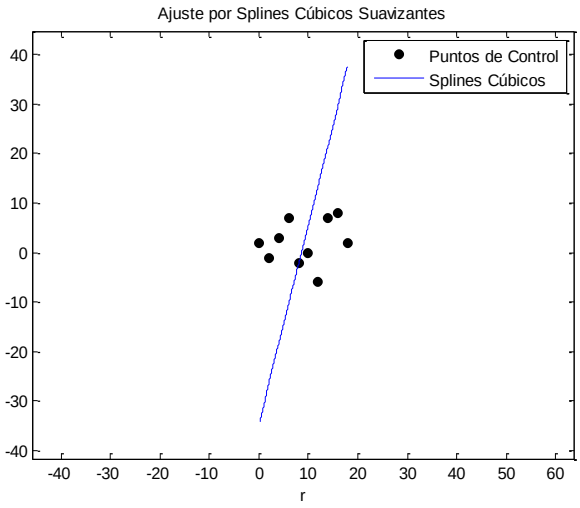

Figura 3.1 Spline cúbico suavizante para ρ = ∞.

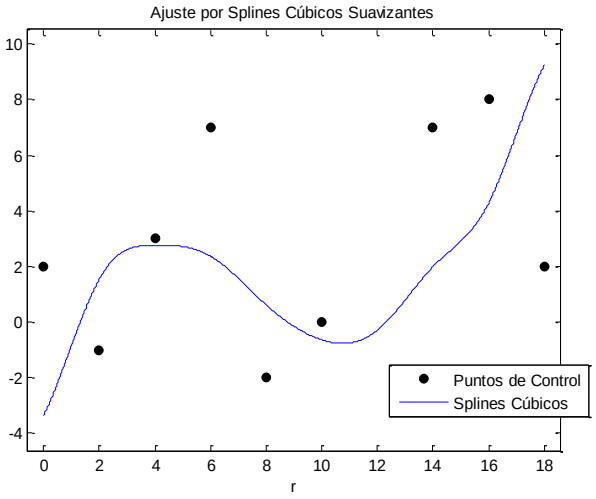

Figura 3.2 Spline cúbico suavizante para ρ = 10.

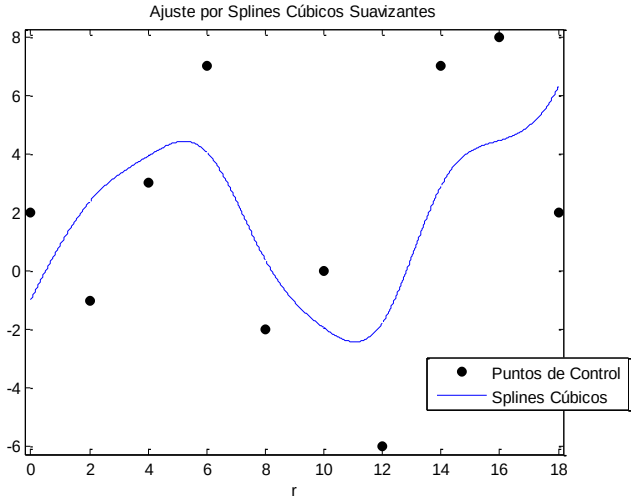

Figura 3.3 Spline cúbico suavizante para ρ = 1.5.

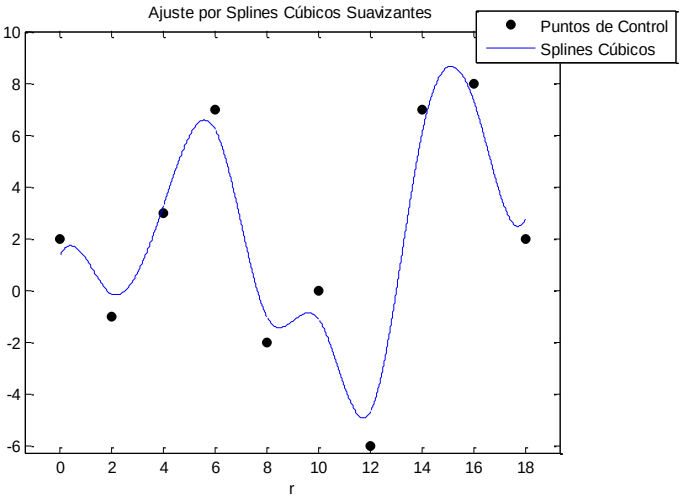

Figura 3.4 Spline cúbico suavizante para ρ = 0.1.

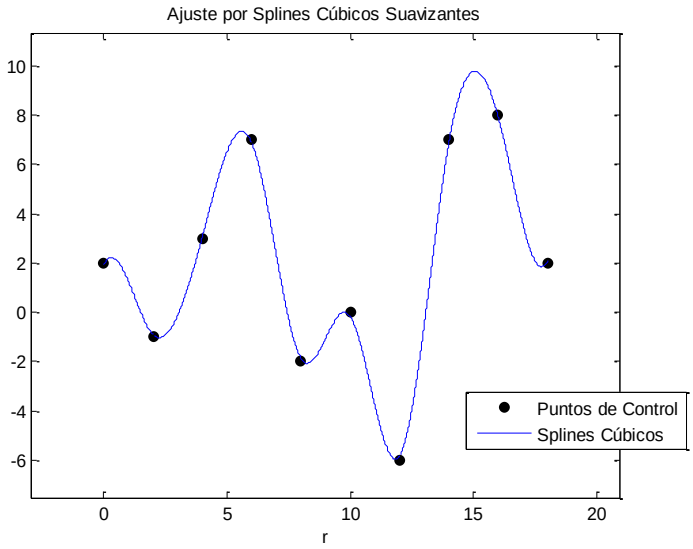

Figura 3.5 Spline cúbico suavizante para ρ = 0.01.

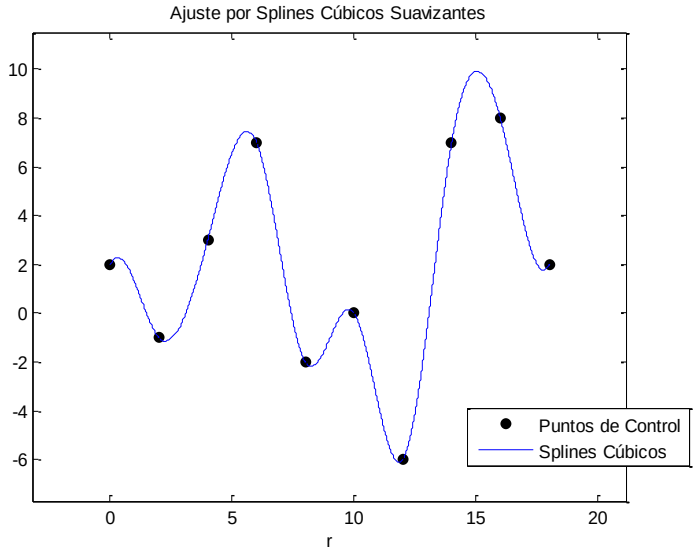

Figura 3.6 Spline cúbico suavizante para ρ = 0.

Vemos en las figuras 4.1-4.6 como al disminuir el valor de los pesos (se ha tomado el mismo valor del peso para todos los puntos), la función spline se pega más a los puntos de control.

Como se puede apreciar en la figura 4.6 si se cumple que $\rho_i = 0$ para todo i, entonces $z_i = y_i$, y el spline suavizador resulta ser un spline interpolante. Esto se traduce en que cuanto mayor es la precisión con que están dadas las magnitudes y_i, tanto menores deben ser los coeficientes de peso respectivos. Si es necesario que el spline pase por el punto (x_k, y_k), el coeficiente de peso ρ_i , $\forall\ i = 1, \dots, m$, correspondiente se deba hacer igual a cero. Sin embargo si se cumple que $\rho_i = \infty$, el ajuste se convierte en una recta de regresión. Con esto queda ilustrado como la bondad del ajuste depende del parámetro ρ.

En los cálculos prácticos lo más importante es la elección de los números ρ_i. Notar que los pesos pueden ser diferentes para puntos distintos. Así por ejemplo en el caso práctico naval que expondremos exigiremos que $\rho_1 = \rho_m = 0$, es decir, que los pesos en los extremos sean cero y así la curva pase por estos puntos, pero los pesos serán distintos de cero en los demás puntos de control.

Definimos $\Delta_i$como el error de medición de la magnitud y_i. Entonces es lógico exigir que

$$|S(x_i) - y_i| \leq \Delta_i$$

El caso más sencillo de elección de los coeficientes de peso ρ_i puede ser

$$\rho_i = c\Delta_i$$

donde c es cierta constante suficientemente pequeña.

3.6 Construcción del spline cúbico suavizante

En cada intervalo $[x_i, x_{i+1}]$,$\forall\ i = 1, \dots, m-1$, se busca un polinomio de grado 3 que satisfaga:

$$S_i(x_i) = z_i, \quad S_i(x_{i+1}) = z_{i+1}$$
$$S_i''(x_i) = \eta_i, \quad S_i''(x_{i+1}) = \eta_{i+1}$$

Se requiere también una condición de continuidad en la primera derivada, pero ésta será impuesta posteriormente. La expresión

$$S(x) = S_i(x) = z_i(1-t) + z_{i+1}t - \frac{h_i^2}{6}t(1-t)[(2-t)\eta_i + (1+t)\eta_{i+1}], \qquad (3.6)$$

ofrece un polinomio de grado 3 que cumple con las condiciones dichas.

Como comprobación, partimos de que la derivada segunda de un polinomio de grado 3 es un polinomio de grado 1, y en este caso debe de satisfacer $S_i''(x_i) = \eta_i, \quad S_i''(x_{i+1}) = \eta_{i+1}$, como ya se ha citado anteriormente. Así si consideramos que

$$t = \frac{x - x_i}{h_i} \quad \text{y} \quad x_{i+1} = x_i + h_i,$$

la derivada segunda del polinomio es:

$$S_i''(x) = (1-t)\eta_i + t\eta_{i+1}.$$

Si integramos el polinomio anterior y realizando un cambio de variable:

$$\int S_i''(x)dx = \int S_i''(t)h_i dt = h_i\left(-\eta_i\frac{(1-t)^2}{2} + \frac{t^2}{2}\eta_{i+1}\right) + C,$$

donde el cambio de variable ha sido,

$$\frac{x-x_i}{h_i} = t \rightarrow dx = h_i dt,$$

Si volvemos a integrar el polinomio:

$$S_i(x) = \int S_i'(x)dx = \int S_i'(t)h_i dt = \frac{h_i^2}{2}\int(-\eta_i(1-t)^2 + t^2\eta_{i+1})dt) + h_i tC + D,$$

$$S_i(x) = \frac{h_i^2}{6}[(1-t)^3\eta_i + t^3\eta_{i+1}] + h_i tC + D. \quad (3.7)$$

Para las condiciones establecidas; debe ser

$$x = x_i \rightarrow S_i(x_i) = z_i \text{ , con t=0,}$$

$$x = x_{i+1} \rightarrow S_i(x_{i+1}) = z_{i+1} \text{ , con t=1,}$$

$$\frac{h_i^2}{6}[\eta_i] + D = z_i \text{ -> } D=z_i-\frac{h_i^2}{6}[\eta_i],$$

al sustituir el valor de D,

$$\frac{h_i^2}{6}[\eta_{i+1}] + h_i C + D = z_{i+1} \text{ -> } h_i C=z_{i+1} - \frac{h_i^2}{6}[\eta_{i+1}] + z_i-\frac{h_i^2}{6}[\eta_i],$$

y por último C

$$C = \frac{z_{i+1}-z_i}{h_i} + \frac{h_i}{6}(\eta_i - \eta_{i+1}).$$

Sustituyendo los valores de las constantes de integración C y D de la ecuación 4.7,

$$h_i tC + D = (z_{i+1}-z_i)t + \frac{h_i^2}{6}(\eta_i - \eta_{i+1})t + z_i - \frac{h_i^2}{6}\eta_i.$$

Nuestro polinomio queda de la forma:

$$S_i(x) = \frac{h_i^2}{6}[(1-t)^3\eta_i + t^3\eta_{i+1}] + (z_{i+1}-z_i)t + \frac{h_i^2}{6}(\eta_i - \eta_{i+1})t + z_i - \frac{h_i^2}{6}\eta_i. \quad (4.8)$$

La forma lineal la podemos reescribir como

$$h_i tC + D - [z_i(1-t) + z_{i+1}t] + [z_i(1-t) + z_{i+1}t] =$$
$$= -\eta_{i+1}\frac{h_i^2}{6}t + \frac{h_i^2}{6}\eta_i t - \frac{h_i^2}{6}\eta_i + (1-t)z_i + t(z_i+1).$$

Con las siguientes observaciones para los términos que acompañan a η_i

$$-t(1-t)(2-t) = -(t-t^2)(2-t) = -(2t-2t^2-t^2+t^3) = -2t+3t^2-t^3 \text{ (I)}$$
$$(1-t)^3 = -t^3+3t^2-3t+1, \text{ (II)}$$

y la diferencia de I y II,

$$(-2t + 3t^2 - t^3) - (-t^3 + 3t^2 - 3t + 1) = -t + 1,$$

sustituyendo:

$$(-t + 1)\frac{h_i^2}{6}\eta_i. \tag{3.9}$$

Los términos que acompañan a η_{i+1} son

$$-t(1-t)(1+t) = -t(1-t^2) = -t + t^3 \quad \text{(I)}$$
$$t^3 \quad \text{(II)},$$

y la diferencia de I y II,

$$(-t + t^3) - t^3 = t,$$

y de nuevo incluyendo los coeficientes

$$t\ \frac{h_i^2}{6}\eta_{i+1}. \tag{3.10}$$

Volviendo a la ecuación 3.8, y sustituyendo en ella las expresiones anteriores, 3.9 y 3.10, se obtiene el spline suavizante de la forma

$$S(x) = S_i(x) = z_i(1-t) + z_{i+1}t - \frac{h_i^2}{6}t(1-t)[(2-t)\eta_i + (1+t)\eta_{i+1}]. \tag{3.11}$$

3.7 Obtención de los coeficientes del spline cúbico suavizante

Vamos a trabajar en el intervalo

$$[x_i, x_{i+1}] \qquad \forall \quad i = 1, \dots, m-1.$$

Si introducimos las notaciones

$$S(x_i) = z_i, \qquad S''(x_i) = \eta_i, \quad \forall \quad i = 1,2, \dots, m,$$

donde z_i y η_i son $2m + 2$ valores desconocidos.

Como ya se demostró anteriormente el spline cúbico suavizante se busca de la forma:

$$S(x) = S_i(x) = z_i(1-t) + z_{i+1}t - \frac{h_i^2}{6}t(1-t)[(2-t)\eta_i + (1+t)\eta_{i+1}],$$

donde

$$h_i = x_{i+1} - x_i\ ,\ t = \frac{x - x_i}{h_i},$$

y los números z_i, η_i, i=0,1,…,m, son la solución del sistema de ecuaciones algebraicas lineales

determinado por las condiciones impuestas.

La función $S_i(x)$ es continua en todo el intervalo [a,b]: para los dos primeros nodos, t tomará el valor t=0 y t=1 respectivamente, sustituyendo en la fórmula 4.11, obtenemos:

$$S_i(x_i) = z_i,$$
$$S_i(x_{i+1}) = z_{i+1.}$$

Así los números z_i y η_i, deben ser elegidos de acuerdo a que el spline tenga derivada primera continua en el intervalo [a,b], para ello vamos a calcular la derivada primera de la función del spline suavizante, $S_i(x)$

En el en el intervalo $[x_{i-1}, x_i]$:

$$S'(x) = S'_{i-1}(x) = \frac{-z_{i-1} + z_i}{h_{i-1}} - \frac{h_{i-1}}{6}[(2 - 6t + 3t^2)\eta_{i-1} + (1 - 3t^2)\eta_i].$$

En el punto $x_i - 0$ (para t=1), tendremos

$$S'(x_i - 0) = S'_{i-1}(x_i) = \frac{-z_{i-1}+z_i}{h_{i-1}} + \frac{h_{i-1}}{6}[\eta_{i-1} + 2\eta_i]. \qquad (3.12)$$

Calculamos la derivada primera del spline el en el intervalo $[x_i, x_{i+1}]$:

$$S'(x) = S'_i(x) = \frac{-z_i + z_{i+1}}{h_i} - \frac{h_i}{6}[(2 - 6t + 3t^2)\eta_i + (1 - 3t^2)\eta_{i+1}].$$

En el punto $x_i + 0$ (para t=0) tendremos

$$S'(x_i + 0) = S'_i(x_i) = \frac{-z_i+z_{i+1}}{h_i} - \frac{h_i}{6}(2\eta_i + \eta_{i+1}). \qquad (3.13)$$

Con la condición de continuidad para la primera derivada del spline en los puntos interiores del retículo ω,

$$S'(x_i - 0) = S'(x_i + 0), \qquad \forall \qquad i = 1, \dots, m - 1,$$

obtendremos m-1 relaciones tal que

$$\frac{-z_{i-1}+z_i}{h_{i-1}}+\frac{h_{i-1}}{6}[\eta_{i-1}+2\eta_i]=\frac{-z_i+z_{i+1}}{h_i}-\frac{h_i}{6}(2\eta_i+\eta_{i+1}),$$
$$\frac{-z_i+z_{i+1}}{h_i}+\frac{z_{i-1}-z_i}{h_{i-1}}=\frac{h_{i-1}}{6}(\eta_{i-1}+2\eta_i)+\frac{h_i}{6}(2\eta_i+\eta_{i+1}).$$

Simplificando

$$\frac{z_{i-1}}{h_{i-1}}-\left(\frac{1}{h_i}+\frac{1}{h_{i-1}}\right)z_i+\frac{z_{i+1}}{h_i}=\frac{1}{6}(h_{i-1}\eta_{i-1}+2(h_{i-1}+h_i)\eta_i+h_i\eta_{i+1}), \qquad (3.14)$$

obteniendo m-1 ecuaciones.

De la condición de continuidad de la primera derivada del spline en los puntos interiores del retículo ω,

$$\frac{z_1}{h_1}-\left(\frac{1}{h_2}+\frac{1}{h_1}\right)z_2+\frac{z_3}{h_2}=\frac{1}{6}[h_1\eta_1+2(h_1+h_2)\eta_2+h_2\eta_3],$$
$$\vdots$$
$$\frac{z_{i-1}}{h_{i-1}}-\left(\frac{1}{h_i}+\frac{1}{h_{i-1}}\right)z_i+\frac{z_{i+1}}{h_i}=\frac{1}{6}[h_{i-1}\eta_{i-1}+2(h_{i-1}+h_i)\eta_i+h_i\eta_{i+1}],$$
$$\vdots$$
$$\frac{z_{m-2}}{h_{m-2}}-\left(\frac{1}{h_{m-2}}+\frac{1}{h_{m-1}}\right)z_{m-1}+\frac{z_m}{h_{m-1}}=\frac{1}{6}[h_{m-2}\eta_{m-2}+2(h_{m-2}+h_{m-1})\eta_{m-1}+h_{m-1}\eta_m],$$
$$\forall \qquad i=2,\dots,m-1.$$

Pudiendo obtenerse la siguiente relación matricial:

$$A\eta=6Hz, \qquad (4.15)$$

dónde:

$$A=\begin{bmatrix} h_1 & 2(h_1+h_2) & h_2 & 0 & \dots & 0 \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & h_{m-2} & 2(h_{m-2}+h_{m-1}) & h_{m-1} \end{bmatrix}$$

$$H=\begin{bmatrix} \frac{1}{h_1} & (-\frac{1}{h_1}-\frac{1}{h_2}) & \frac{1}{h_2} & 0 & \dots & 0 \\ 0 & \frac{1}{h_2} & (-\frac{1}{h_2}-\frac{1}{h3}) & \frac{1}{h_3} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \frac{1}{h_{m-2}} & (-\frac{1}{h_{m-2}}-\frac{1}{h_{m-1}}) & \frac{1}{h_{m-1}} \end{bmatrix}$$

Quedando el sistema de ecuaciones incompleto, aún faltan ecuaciones.

Para la condición de minimización del funcional $J(S)=\int_a^b (S''(x))^2 dx + \sum_{i=0}^{m} \frac{1}{\rho i}(S(x_i)-y_i)^2$, calculamos el primer término

$$\int_a^b (S''(x))^2 dx = \int_0^1 [(1-t)\eta_i + t\eta_{i+1}]^2 h_i dt$$

$$t=\frac{x-x_i}{h_i} \rightarrow dx = h_i dt$$

Habiendo realizado el cambio de variable y desarrollando el binomio,

$$h_i \int_0^1 (1-t)^2 {\eta_i}^2 + t^2 {\eta_{i+1}}^2 + 2t(1-t)\eta_i\eta_{i+1} dt,$$

e integrando la expresión anterior,

$$\left[h_i\left[-\frac{(1-t)^3}{3}\right]_0^1 {\eta_i}^2 + \left[\frac{t^3}{3}\right]_0^1 {\eta_{i+1}}^2 + \left[t^2-\frac{2t^3}{3}\right]_0^1 \eta_i\eta_{i+1}\right] = h_i\left[\frac{1}{3}{\eta_i}^2 + \frac{1}{3}{\eta_{i+1}}^2 + \frac{1}{3}\eta_i\eta_{i+1}\right].$$

Así obtenemos el primer miembro de la expresión del funcional:

$$\int_a^b (S''(x))^2 dx = h_i\left[\frac{1}{3}{\eta_i}^2 + \frac{1}{3}{\eta_{i+1}}^2 + \frac{1}{3}\eta_i\eta_{i+1}\right]. \quad (3.16)$$

Vamos a desarrollar el funcional $J(s)$ mediante la técnica de los multiplicadores de Lagrange, la

variable auxiliar λ_1, impondrá que se satisfaga la condición de contorno en el extremo izquierdo. La variable λ_m por su parte impondrá que se cumpla la condición de contorno del extremo derecho. Los valores λ_i , $i = 2, \dots, m-1$ impondrán que se satisfagan las condiciones debidas a la continuidad de la primera derivada.

Derivando el funcional respecto de λ , $\frac{dJ_a(s)}{d\lambda}$, se obtiene el sistema de ecuaciones

$$\tilde{A}\eta = 6\hat{H}z + F, \tag{3.17}$$

donde $\tilde{A}, \hat{H}$ y F , dependerán de las condiciones de contorno elegidas.

Despejando η en la ecuación 3.17:

$$\eta = \tilde{A}^{-1}(6\,\hat{H}z + F). \tag{3.18}$$

donde necesitamos que $\tilde{A}$ sea invertible.

Derivando el funcional auxiliar J_a respecto de η, $\frac{dJ_a(s)}{d\eta}$, obtenemos m ecuaciones más, que expresadas matricialmente quedan:

$$U\,\eta = V\lambda. \tag{3.19}$$

Al despejar λ en la ecuación 3.18,

$$\lambda = V^{-1}U\eta, \tag{3.20}$$

donde nuevamente pedimos que V sea invertible.

Sustituyendo el valor anterior de η de la ecuación 3.18 en la ecuación 3.20,

$$\lambda = V^{-1}U\tilde{A}^{-1}(6\,\hat{H}z + F). \tag{3.21}$$

Al definir $M = V^{-1}U\tilde{A}^{-1}$, y sustituyendo:

$$\lambda = 6M\hat{H}z + MF. \tag{3.22}$$

Derivando el funcional respecto de z, obtenemos

$$z = Y + \frac{1}{2}\rho\, J\lambda. \tag{3.23}$$

Y al sustituir el valor de λ de la ecuación 3.22:

$$z = Y + \frac{1}{2}\rho\, J\left(6M\hat{H}z + MF\right) = Y + \frac{1}{2}K_1 z + K_{2,} \tag{3.24}$$

donde K_1, K_2son dos matrices auxiliares, introducidas para simplificar la notación del sistema

resultante,

$$\mathrm{K}_1 = \rho\,\mathrm{JM6\hat{H}},$$

$$\mathrm{K}_2 = \rho\,\mathrm{JMF}.$$

Y definiendo una tercera matriz auxiliar C

$$\mathrm{C} = \mathrm{Id} - \frac{1}{2}\mathrm{K}_1,$$

el sistema resultante quedará

$$\mathrm{Cz} = \mathrm{Y} + \mathrm{K}_2. \tag{3.25}$$

Para poder resolver este sistema de ecuaciones es necesario definir las matrices anteriores, que vendrán dadas en función de las condiciones de contorno elegidas, las cuales ya se comentó que se elegirían en función de los datos adicionales que tengamos sobre el comportamiento de la función.

Notar que este sistema siempre tendrá solución única para cualquier valor de ρ salvo para un número finito de valores de ρ que tienen que ver con los valores propios de la matriz C. En particular, para valores de ρ suficientemente pequeños (caso que nos es más interesante en la práctica) siempre hay solución única.

3.8 Sistema de ecuaciones final en función de las condiciones de contorno

A continuación se va a detallar cuáles serán las dos ecuaciones que se podrán obtener en función de las condiciones de contorno, las cuales incluso podrán ser distintas en cada extremo.

En caso de conocer condiciones de primer tipo, es decir, los valores de la primera derivada de $\mathrm{S(x)}$ en los extremos del intervalo $[\mathrm{a}, \mathrm{b}]$; se obtendrían las dos ecuaciones que nos faltan para completar el sistema del siguiente modo:

- Obtenemos la derivada del trozo de Spline correspondiente al extremo.
- Evaluamos esta ecuación en punto extremo.
- Y lo igualamos al valor conocido de la primera derivada.

1) Para el extremo izquierdo (a):

$$S_1'(x_1) = \frac{-z_1 + z_2}{h_1} - \frac{h_1}{6}(2\eta_1 + \eta_1) = f'(a),$$

$$\frac{-z_1 + z_2}{h_1} = f'(a) + \frac{h_1}{6}(2\eta_1 + \eta_1).$$

2) Para el extremo derecho (b):

$$S'_{m-1}(x_m) = \frac{-z_{m-1} + z_m}{h_{m-1}} + \frac{h_{m-1}}{6}[\eta_{m-1} + 2\eta_m] = f'(b),$$

$$\frac{-z_{m-1} + z_m}{h_{m-1}} = f'(b) - \frac{h_{m-1}}{6}[\eta_{m-1} + 2\eta_m]$$

Si utilizáramos condiciones de segundo tipo, es decir, damos los valores de la segunda derivada en los extremos del intervalo $[a, b]$; básicamente se han de dar los valores de η_1 y η_m:

$$f''(a) = \eta_1\,, \quad f''(b) = \eta_m.$$

En caso de que la función a suavizar sea periódica, o de tercer tipo, de periodo $T = b - a$, en particular, $f(a) = f(b)$ y por tanto $S_1(a) = S_{m-1}(b)$; se utilizarán condiciones de tercer tipo, las cuales darán las siguientes ecuaciones:

La primera ecuación adicional sería $S'_1(t_1) = S'_{m-1}(t_m)$, y sustituyendo:

$$\frac{-z_1 + z_2}{h_1} - \frac{h_1}{6}(2\eta_1 + \eta_1) = \frac{-z_{m-1} + z_m}{h_{m-1}} + \frac{h_{m-1}}{6}[\eta_{m-1} + 2\eta_m],$$

$$\frac{-z_1 + z_2}{h_1} - \frac{-z_{m-1} + z_m}{h_{m-1}} = \frac{h_{m-1}}{6}[\eta_{m-1} + 2\eta_m] + \frac{h_1}{6}(2\eta_1 + \eta_1).$$

La segunda ecuación obtenida sería $S''_1(t_1) = S''_{m-1}(t_m)$, en nuestro caso de suavización

$$\eta_1 - \eta_m = 0.$$

No siempre conocemos los mismos datos en ambos extremos del intervalo $[a, b]$, por lo que podemos tener diferentes condiciones en cada extremo, excepto en las condiciones de tercer tipo que exigen que la función sea periódica de periodo T=b-a y en particular $f(a) = f(b)$.

Por último antes de empezar a estudiar los diferentes casos en los que nos podremos encontrar a la hora de aproximar mediante splines suavizantes, recordamos que las condiciones de primer tipo tienen preferencia sobre las de segundo tipo, pudiendo aproximar estos valores.

A partir de aquí estudiaremos los casos más representativos, los cuales necesitan ser estudiados individualmente para poder demostrar que el sistema de ecuaciones resultante tiene solución y además es única, es decir vamos a demostrar que las matrices $\tilde{A}$ y V son invertibles en todos los casos.

Para estudiar la invertibilidad de dichas matrices vamos a hacer uso constantemente e un

resultado que asegura que una matriz estrictamente diagonal dominante es invertible.

3.9 Comprobación de sistema compatible determinado

3.9.1 Introducción

Una vez construido un sistema de ecuaciones, es muy importante comprobar si el sistema es compatible determinado, con única solución, pudiendo obtenerse ésta y así poder obtener el spline correspondiente.

Si construimos el sistema completo $Az = B$ para que éste tenga solución compatible determinada sólo habrá que demostrar que la matriz A es invertible, es decir, que el determinante de A no es nulo.

En matemáticas, en particular en álgebra lineal, una matriz cuadrada A de orden n se dice que es invertible, no singular, no degenerada o regular si existe otra matriz cuadrada de orden n, llamada matriz inversa de A y representada como A^{-1}, tal que:

$$A \cdot A^{-1} = A^{-1}A = I_n,$$

donde I_n es la matriz identidad de orden n y el producto utilizado es el producto de matrices usual.

Existe un resultado que asegura que todo matriz estrictamente diagonal dominante es invertible.

Formalmente, se dice que la matriz A de orden n es estrictamente diagonal dominante cuando se satisface:

$$|a_{i,i}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{i,j}| \qquad \forall\, i = 1, \dots, n.$$

El resultado, conocido como lema de Hadamard, se prueba de la siguiente manera:

Si la matriz A no fuese invertible, la ecuación $\mathrm{Az} = 0$ admitiría una solución no nula. Entonces existe:

$$\mathrm{z} = [\mathrm{z}_{1,} \dots \dots, \mathrm{z}_{\mathrm{n}}]^{\mathrm{T}},$$

una solución. Podemos suponer sin pérdida de generalidad que

$$\max_{1 \leq \mathrm{i} \leq \mathrm{n}} |\mathrm{z}_{\mathrm{i}}| = 1.$$

Sea r un índice para el que es $|\mathrm{z}_{\mathrm{r}}| = 1$. Tomando módulos en la ecuación

$$\mathrm{a}_{\mathrm{r}1}\mathrm{z}_1 + \cdots + \mathrm{a}_{\mathrm{rr}}\mathrm{z}_{\mathrm{r}} + \cdots + \mathrm{a}_{\mathrm{rn}}\mathrm{z}_{\mathrm{n}} = 0.$$

Se concluye que

$$|a_{rr}| \leq \sum_{i \neq r} |a_{r1}||z_1| \leq \sum_{i \neq r} |a_{r1}|.$$

Desigualdad que contradice la hipótesis de que la matriz A es estrictamente diagonal dominante. Esto permite dar por demostrado que una matriz estrictamente diagonal dominante es invertible.

Nos vamos a centrar en buscar el sistema de ecuaciones final para los casos particulares más representativos (los demás casos se deducen fácilmente a partir de éstos) y además demostraremos que cada uno de los sistemas que se forman, incluidas las condiciones de contorno, son compatibles determinados lo cual nos garantizará que la función spline se pueda construir.

Para abreviar se utilizarán las siguientes denominaciones:

1: condiciones de primer tipo (se conoce el valor de la primera derivada en el extremo).
2: condiciones de segundo tipo (se conoce el valor de la segunda derivada en el extremo).
3: condiciones de tercer tipo (existe periocidad).
CI: condición de contorno en el extremo izquierdo.
CD: condición de contorno en el extremo derecho.

3.9.2 Caso CI=1, CD=1.

Las condiciones en los extremos son de primer tipo, por lo que se conocen las primeras derivadas y tenemos las dos siguientes ecuaciones adicionales:

$$\frac{-z_1 + z_2}{h_1} = f'(a) + \frac{h_1}{6}(2\eta_1 + \eta_2),$$

$$\frac{-z_{m-1} + z_m}{h_{m-1}} = f'(b) - \frac{h_{m-1}}{6}[\eta_{m-1} + 2\eta_m].$$

Al desarrollar el funcional para las condiciones de contorno establecidas

$$J_a(S) = h_1\left[\frac{1}{3}{\eta_1}^2 + \frac{1}{3}{\eta_2}^2 + \frac{1}{3}\eta_1\eta_2\right] + \cdots + h_{m-1}\left[\frac{1}{3}{\eta_{m-1}}^2 + \frac{1}{3}{\eta_m}^2 + \frac{1}{3}\eta_{m-1}\eta_m\right] + \frac{1}{\rho_1}(z_1 - y_1)^2 + \cdots + \frac{1}{\rho_m}(z_m - y_m)^2 + \lambda_1\left(\frac{-z_1+z_2}{h_1} - \frac{h_1}{6}(2\eta_1 + \eta_2) - f'(a)\right) + \lambda_2\left[\left(\frac{z_1}{h_1} - \left(\frac{1}{h_2} + \frac{1}{h_1}\right)\right)z_2 + \frac{z_3}{h_2} - \frac{1}{6}(h_1\eta_1 + 2(h_1 + h_2)\eta_2 + h_2\eta_3)\right] + \cdots + \lambda_i\left[\left(\frac{z_{i-1}}{h_{i-1}} - \left(\frac{1}{h_i} + \frac{1}{h_{i-1}}\right)\right)z_i + \frac{z_{i+1}}{h_i} - \frac{1}{6}(h_{i-1}\eta_{i-1} + 2(h_{i-1} + h_i)\eta_i + h_i\eta_{i+1})\right] + \cdots + \lambda_{m-1}\left[\left(\frac{z_{m-2}}{h_{m-2}} - \left(\frac{1}{h_{m-2}} + \frac{1}{h_{m-1}}\right)\right)z_{m-1} + \frac{z_m}{h_{m-1}} - \frac{1}{6}(h_{m-2}\eta_{m-2} + 2(h_{m-2} + h_{m-1})\eta_{m-1} + h_{m-1}\eta_m)\right] + \lambda_m\left(\frac{-z_{m-1}+z_m}{h_{m-1}} + \frac{h_{m-1}}{6}[\eta_{m-1} +\right.$$

$2\eta_m] - f'(b)\Big),$

$$\forall \quad i = 3, \dots, m-2.$$

Operando el funcional $\frac{dJ_a(s)}{d\lambda}$ se obtiene el sistema de ecuaciones inicial ampliado

$$\tilde{A}\eta = 6\hat{H}z + F,$$

donde las matrices,

$$\tilde{A}=\begin{bmatrix} \frac{h_1}{3} & \frac{h_1}{6} & 0 & 0 & \dots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & \ddots & \vdots \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 0 & h_{m-2} & 2(h_{m-2}+h_{m-1}) & h_{m-1} \\ 0 & 0 & 0 & 0 & -\frac{h_{m-1}}{6} & -\frac{h_{m-1}}{3} \end{bmatrix}$$

$$\hat{H}=\begin{bmatrix} -\frac{1}{h_1} & \frac{1}{h_1} & 0 & 0 & 0 & 0 \\ \frac{1}{h_1} & (-\frac{1}{h_1}-\frac{1}{h_2}) & \frac{1}{h_2} & 0 & \dots & \vdots \\ 0 & \frac{1}{h_2} & (-\frac{1}{h_2}-\frac{1}{h_3}) & \frac{1}{h_3} & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 0 & \frac{1}{h_{m-2}} & (-\frac{1}{h_{m-2}}-\frac{1}{h_{m-1}}) & \frac{1}{h_{m-1}} \\ 0 & 0 & 0 & 0 & -\frac{1}{h_{m-1}} & \frac{1}{h_{m-1}} \end{bmatrix}$$

$$F=\begin{bmatrix} f'(a) \\ 0 \\ \vdots \\ 0 \\ f'(b) \end{bmatrix}$$

Como se puede comprobar fácilmente la matriz A es estrictamente diagonal dominante y por tanto será invertible.

Realizando la siguiente operación al funcional $\frac{dJ_a(s)}{d\eta}$, obtenemos m ecuaciones más,

$$\frac{2}{3}h_1\eta_1 + \frac{h_1}{3}\eta_2 - \frac{h_1}{3}\lambda_1 - \frac{h_1}{6}\lambda_2 = 0,$$

$$\frac{h_1}{3}\eta_1 + \frac{2}{3}(h_1+h_2)\eta_2 + \frac{h_2}{3}\eta_3 - \frac{h_1}{6}\lambda_1 - \frac{1}{3}(h_1+h_2)\lambda_2 - \frac{h_2}{6}\lambda_3 = 0,$$

$$\vdots$$

$$\frac{h_i}{3}\eta_{i-1} + \frac{2}{3}(h_{i-1}+h_i)\eta_i + \frac{h_i}{3}\eta_{i+1} - \frac{1}{3}(h_{i-1}+h_i)\lambda_i - \frac{h_i}{6}\lambda_{i+1} = 0,$$

$$\vdots$$

$$\frac{h_{m-3}}{3}\eta_{m-3} + \frac{2}{3}(h_{m-3}+h_{m-2})\eta_{m-2} + \frac{h_{m-2}}{3}\eta_{m-1} - \frac{1}{3}(h_{m-3}+h_{m-2})\lambda_{m-2} - \frac{h_{m-3}}{6}\lambda_{m-2} = 0,$$

$$\frac{h_{m-2}}{3}\eta_{m-2} + \frac{2}{3}(h_{m-2}+h_{m-1})\eta_{m-1} + \frac{h_{m-1}}{3}\eta_m - \frac{1}{3}(h_{m-2}+h_{m-1})\lambda_{m-1} - \frac{h_{m-2}}{6}\lambda_{m-1} + \frac{h_{m-1}}{6}\lambda_m = 0,$$

$$\frac{2}{3}h_{m-1}\eta_m + \frac{h_{m-1}}{3}\eta_{m-1} + \frac{h_{m-1}}{3}\lambda_m - \frac{h_{m-1}}{6}\lambda_{m-1} = 0,$$

$$\forall \qquad i = 3, \dots, m-1.$$

Que también puede ser escrito matricialmente como

$$U\,\eta = V\lambda,$$

donde

$$\mathrm{U}=\begin{bmatrix} \frac{2h_1}{3} & \frac{h_1}{3} & 0 & 0 & \dots & 0 \\ \frac{h_1}{3} & \frac{2(h_1+h_2)}{3} & \frac{h_2}{3} & 0 & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \frac{h_{m-2}}{3} & \frac{2(h_{m-2}+h_{m-1})}{3} & \frac{h_{m-1}}{3} \\ 0 & 0 & 0 & 0 & \frac{h_{m-1}}{3} & \frac{2h_{m-1}}{3} \end{bmatrix}$$

$$\mathrm{V}=\begin{bmatrix} \frac{h_1}{3} & \frac{h_1}{6} & 0 & 0 & \dots & 0 \\ \frac{h_1}{6} & \frac{(h_1+h_2)}{3} & \frac{h_2}{6} & 0 & \ddots & 0 \\ 0 & \frac{h_2}{6} & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \frac{h_{m-2}}{6} & \frac{2(h_{m-2}+h_{m-1})}{3} & -\frac{h_{m-1}}{6} \\ 0 & 0 & 0 & 0 & \frac{h_{m-1}}{6} & -\frac{h_{m-1}}{3} \end{bmatrix}$$

La matriz V se observa fácilmente que es estrictamente diagonal dominante, y por lo tanto es invertible.

Derivando el funcional respecto de los z_i, $\frac{dJ_a(s)}{dz}$,

$$\frac{2}{\rho_1}(z_1 - y_1) - \frac{\lambda_1}{h_1} + \frac{\lambda_2}{h_1} = 0,$$
$$\frac{2}{\rho_2}(z_2 - y_2) + \frac{\lambda_1}{h_1} - \left(\frac{1}{h_1} + \frac{1}{h_2}\right)\lambda_2 + \frac{\lambda_3}{h_2} = 0,$$
$$\vdots$$
$$\frac{2}{\rho_{i+1}}(z_i - y_i) - \left(\frac{1}{h_{i-1}} + \frac{1}{h_i}\right)\lambda_i + \frac{\lambda_{i+1}}{h_i} = 0,$$
$$\vdots$$
$$\frac{2}{\rho_{m-2}}(z_{m-2} - y_{m-2}) - \left(\frac{1}{h_{m-3}} + \frac{1}{h_{m-2}}\right)\lambda_{m-2} + \frac{\lambda_{m-3}}{h_{m-3}} = 0,$$
$$\frac{2}{\rho_{m-1}}(z_{m-1} - y_{m-1}) - \frac{\lambda_m}{h_{m-1}} - \left(\frac{1}{h_{m-2}} + \frac{1}{h_{m-1}}\right)\lambda_{m-1} + \frac{\lambda_{m-2}}{h_{m-2}} = 0,$$
$$\frac{2}{\rho_m}(z_m - y_m) + \frac{\lambda_m}{h_{m-1}} + \frac{\lambda_{m-1}}{h_{m-1}} = 0,$$
$$\forall \qquad i = 3, \dots, m-1.$$

Donde despejando z,

$$z_1 = y_1 + \frac{\rho_1}{2}\left(+\frac{\lambda_1}{h_1} - \frac{\lambda_2}{h_1}\right),$$
$$z_2 = y_2 + \frac{\rho_2}{2}\left(-\frac{\lambda_1}{h_1} + \left(\frac{1}{h_1} + \frac{1}{h_2}\right)\lambda_2 - \frac{\lambda_3}{h_2}\right),$$
$$\vdots$$
$$z_i = y_i + \frac{\rho_i}{2}\left(\left(\frac{1}{h_{i-1}} + \frac{1}{h_i}\right)\lambda_i - \frac{\lambda_{i+1}}{h_i}\right),$$
$$\vdots$$
$$z_{m-2} = y_{m-2} + \frac{\rho_{m-2}}{2}\left(\left(\frac{1}{h_{m-3}} + \frac{1}{h_{m-2}}\right)\lambda_{m-2} - \frac{\lambda_{m-3}}{h_{m-3}}\right),$$
$$z_{m-1} = y_{m-1} + \frac{\rho_{m-1}}{2}\left(\frac{\lambda_m}{h_{m-1}} + \left(\frac{1}{h_{m-2}} + \frac{1}{h_{m-1}}\right)\lambda_{m-1} - \frac{\lambda_{m-2}}{h_{m-2}}\right),$$
$$z_m = y_m + \frac{\rho_m}{2}\left(-\frac{\lambda_{m-1}}{h_{m-1}} - \frac{\lambda_m}{h_{m-1}}\right),$$
$$\forall \qquad i = 3, \dots, m-1.$$

Matricialmente puede ser expresado como,

$$Z = Y + \frac{1}{2}\rho\, \mathrm{J}\, \lambda,$$

donde,

$$\mathrm{J}=\begin{bmatrix} \frac{1}{h_1} & -\frac{1}{h_1} & 0 & 0 & \dots & 0 \\ -\frac{1}{h_1} & \frac{1}{h_1}+\frac{1}{h_2} & -\frac{1}{h_2} & \ddots & \ddots & \vdots \\ 0 & \frac{1}{h_2} & \frac{1}{h_2}+\frac{1}{h_3} & \ddots & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & -\frac{1}{h_{m-2}} & 0 \\ \vdots & \ddots & \ddots & -\frac{1}{h_{m-2}} & \frac{1}{h_{m-2}}+\frac{1}{h_{m-1}} & \frac{1}{h_{m-1}} \\ 0 & \dots & 0 & 0 & -\frac{1}{h_{m-1}} & -\frac{1}{h_{m-1}} \end{bmatrix}$$

3.9.3 Caso CI=2, CD=2.

Las condiciones en los extremos son de segundo tipo, por lo que se conocen las segundas derivadas y tenemos las dos siguientes ecuaciones adicionales:

$$f''(a) = \eta_1,$$
$$f''(b) = \eta_m.$$

Al desarrollar el funcional para las condiciones de contorno establecidas

$$J_a(s) = h_1\left[\frac{1}{3}{\eta_1}^2 + \frac{1}{3}{\eta_2}^2 + \frac{1}{3}\eta_1\eta_2\right] + \cdots + h_{m-1}\left[\frac{1}{3}{\eta_{m-1}}^2 + \frac{1}{3}{\eta_m}^2 + \frac{1}{3}\eta_{m-1}\eta_m\right] + \frac{1}{\rho_1}(z_1 - y_1)^2 + \cdots + \frac{1}{\rho_m}(z_m - y_m)^2 + \lambda_1(\eta_1 - f''(a)) + \cdots + \lambda_i\left[\left(\frac{z_{i-1}}{h_{i-1}} - \left(\frac{1}{h_i} + \frac{1}{h_{i-1}}\right)\right)z_i + \frac{z_{i+1}}{h_i} - \frac{1}{6}(h_{i-1}\eta_{i-1} + 2(h_{i-1} + h_i)\eta_i + h_i\eta_{i+1})\right] + \cdots + \lambda_{m-1}\left[\left(\frac{z_{m-2}}{h_{m-2}} - \left(\frac{1}{h_{m-2}} + \frac{1}{h_{m-3}}\right)\right)z_{m-2} + \frac{z_m}{h_{m-1}} - \frac{1}{6}(h_{m-2}\eta_{m-2} + 2(h_{m-2} + h_{m-1})\eta_{m-1} + h_{m-1}\eta_m)\right] + \lambda_m(\eta_m - f''(b))$$

$$\forall \quad i = 2, \dots, m-1.$$

Operando el funcional $\frac{\mathrm{dJ_a(s)}}{\mathrm{d}\lambda}$ se obtiene el sistema de ecuaciones ampliado

$$\tilde{\mathrm{A}}\eta = 6\hat{\mathrm{H}}z + F,$$

donde

$$\tilde{A}=\begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & \ddots & \vdots \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 0 & h_{m-2} & 2(h_{m-2}+h_{m-1}) & h_{m-1} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\tilde{H}=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{h_1} & (-\frac{1}{h_1}-\frac{1}{h_2}) & \frac{1}{h_2} & 0 & \dots & \vdots \\ 0 & \frac{1}{h_2} & (-\frac{1}{h_2}-\frac{1}{h_3}) & \frac{1}{h_3} & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 0 & \frac{1}{h_{m-2}} & (-\frac{1}{h_{m-2}}-\frac{1}{h_{m-1}}) & \frac{1}{h_{m-1}} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$F=\begin{bmatrix} f''(a) \\ 0 \\ \vdots \\ 0 \\ f''(b) \end{bmatrix}$$

Y como se puede observar con facilidad la matriz A estrictamente diagonal dominante, y por lo tanto invertible.

Realizando la siguiente operación al funcional $\frac{dJ_a(s)}{d\eta}$, obtenemos m ecuaciones más,

$$\frac{2}{3}h_1\eta_1+\frac{h_1}{3}\eta_2+\lambda_1-\frac{h_1}{6}\lambda_2=0,$$

$$\vdots$$

$$\frac{h_i}{3}\eta_{i-1}+\frac{2}{3}(h_{i-1}+h_i)\eta_i+\frac{h_i}{3}\eta_{i+1}-\frac{1}{3}(h_{i-1}+h_i)\lambda_i-\frac{h_i}{6}\lambda_{i+1}=0,$$

$$\vdots$$

$$\frac{h_i}{3}\eta_{m-2}+\frac{2}{3}(h_{m-2}+h_{m-1})\eta_{m-1}+\frac{h_{m-1}}{3}\eta_m-\frac{1}{3}(h_{m-2}+h_{m-1})\lambda_{m-1}-\frac{h_{m-2}}{6}\lambda_{m-1}=0,$$

$$\frac{2}{3}h_{m-1}\eta_m+\frac{h_{m-1}}{3}\eta_{m-1}+\lambda_m-\frac{h_{m-1}}{6}\lambda_{m-1}=0,$$

$$\forall \quad i=2,\dots,m-1.$$

Matricialmente puede ser expresado como

$$U\,\eta=V\lambda,$$

donde

$$U=\begin{bmatrix} \frac{2h_1}{3} & \frac{h_1}{3} & 0 & 0 & \dots & 0 \\ \frac{h_1}{3} & \frac{2(h_1+h_2)}{3} & \frac{h_2}{3} & 0 & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \frac{h_{m-2}}{3} & \frac{2(h_{m-2}+h_{m-1})}{3} & \frac{h_{m-1}}{3} \\ 0 & 0 & 0 & 0 & \frac{h_{m-1}}{3} & \frac{2h_{m-1}}{3} \end{bmatrix}$$

$$V=\begin{bmatrix} -1 & \frac{h_1}{6} & 0 & 0 & \dots & 0 \\ 0 & \frac{(h_1+h_2)}{3} & \frac{h_2}{6} & 0 & \ddots & 0 \\ 0 & \frac{h_2}{6} & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \frac{h_{m-2}}{6} & \frac{2(h_{m-2}+h_{m-1})}{3} & 0 \\ 0 & 0 & 0 & 0 & \frac{h_{m-1}}{6} & -1 \end{bmatrix}$$

Siendo la matriz V invertible, ya que su determinante es distinto de cero, como se comprueba fácilmente de desarrollar dicho determinante por la primera columna y posteriormente por la última. La matriz a la que se llega resulta ser estrictamente diagonal dominante y por tanto invertible y por tanto con determinante distinto de cero.

Derivando el funcional respecto de los z_i, $\frac{dJ_a(s)}{dz}$,

$$\frac{2}{\rho_1}(z_1-y_1)+\frac{\lambda_2}{h_1}=0,$$

$$\vdots$$

$$\frac{2}{\rho_i}(z_i - y_i) - (\frac{1}{h_{i-1}} + \frac{1}{h_i})\lambda_i + \frac{\lambda_{i+1}}{h_i} = 0,$$

$$\vdots$$

$$\frac{2}{\rho_{m-1}}(z_{m-1} - y_{m-1}) - (\frac{1}{h_{m-2}} + \frac{1}{h_{m-1}})\lambda_{m-1} + \frac{\lambda_{m-2}}{h_{m-2}} = 0,$$

$$\frac{2}{\rho_m}(z_m - y_m) + \frac{\lambda_{m-1}}{h_{m-1}} = 0,$$

$$\forall \qquad i = 2, \dots, m-1.$$

Despejando z,

$$z_1 = y_1 + \frac{\rho_1}{2}\left(-\frac{\lambda_2}{h_1}\right),$$

$$\vdots$$

$$z_i = y_i + \frac{\rho_i}{2}\left(\left(\frac{1}{h_{i-1}} + \frac{1}{h_i}\right)\lambda_i - \frac{\lambda_{i+1}}{h_i}\right),$$

$$\vdots$$

$$z_{m-1} = y_{m-1} + \frac{\rho_{m-1}}{2}\left(\left(\frac{1}{h_{m-2}} + \frac{1}{h_{m-1}}\right)\lambda_{m-1} - \frac{\lambda_{m-2}}{h_{m-2}}\right),$$

$$z_m = y_m + \frac{\rho_m}{2}\left(-\frac{\lambda_{m-1}}{h_{m-1}}\right),$$

$$\forall \qquad i = 2, \dots, m-1.$$

Matricialmente,

$$Z = Y + \frac{1}{2}\rho \, \mathrm{J} \, \lambda,$$

donde,

$$\mathrm{J} = \begin{bmatrix} 0 & -\frac{1}{h_1} & 0 & 0 & \dots & 0 \\ 0 & \frac{1}{h_1} + \frac{1}{h_2} & -\frac{1}{h_2} & \ddots & \ddots & \vdots \\ 0 & \frac{1}{h_2} & \frac{1}{h_2} + \frac{1}{h_3} & \ddots & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & -\frac{1}{h_{m-2}} & 0 \\ \vdots & \ddots & \ddots & \frac{1}{h_{m-2}} & \frac{1}{h_{m-2}} + \frac{1}{h_{m-1}} & 0 \\ 0 & \dots & 0 & 0 & \frac{1}{h_{m-1}} & 0 \end{bmatrix}.$$

3.9.4 Caso CI=3, CD=3

Para este caso las ecuaciones adicionales se obtienen considerando que la función a suavizar es periódica de periodo $\mathrm{T} = \mathrm{b} - \mathrm{a}$, de modo que aplicando las condiciones de tercer tipo se obtendrán las siguientes ecuaciones:

$$\eta_1 - \eta_\mathrm{m} = 0,$$

$$6\left(\frac{-\mathrm{z}_1 + \mathrm{z}_2}{\mathrm{h}_1} - \frac{-\mathrm{z}_{\mathrm{m}-1} + \mathrm{z}_\mathrm{m}}{\mathrm{h}_{\mathrm{m}-1}}\right) = \mathrm{h}_1(2\eta_1 + \eta_2) + \mathrm{h}_{\mathrm{m}-1}[\eta_{\mathrm{m}-1} + 2\eta_\mathrm{m}].$$

Y necesitamos que los datos cumplan las condiciones de compatibilidad

$$\rho_1 = \rho_\mathrm{m},$$

$$\mathrm{y}_1 = \mathrm{y}_\mathrm{m}.$$

Al desarrollar el funcional para las condiciones de contorno establecidas:

$$J_a(S) = h_1\left[\tfrac{1}{3}{\eta_1}^2 + \tfrac{1}{3}{\eta_2}^2 + \tfrac{1}{3}\eta_1\eta_2\right] + \cdots + h_{m-1}\left[\tfrac{1}{3}{\eta_{m-1}}^2 + \tfrac{1}{3}{\eta_m}^2 + \tfrac{1}{3}\eta_{m-1}\eta_m\right] + \frac{1}{\rho_1}(z_1 - y_1)^2 + \cdots + \frac{1}{\rho_m}(z_m - y_m)^2 + \lambda_1((\eta_1 - \eta_m)) + \lambda_2\left[\left(\frac{z_1}{h_1} - \left(\frac{1}{h_2} + \frac{1}{h_1}\right)\right)z_2 + \frac{z_3}{h_2} - \frac{1}{6}\left(h_1\eta_1 + 2(h_1 + h_2)\eta_2 + h_2\eta_3\right)\right] + \lambda_{i+1}\left[\left(\frac{z_i}{h_i} - \left(\frac{1}{h_{i+1}} + \frac{1}{h_i}\right)\right)z_{i+1} + \frac{z_{i+2}}{h_{i+1}} - \frac{1}{6}\left(h_i\eta_i + 2(h_i + h_{i+1})\eta_{i+1} + h_{i+1}\eta_{i+2}\right)\right] + \cdots + \lambda_{m-1}\left[\left(\frac{z_{m-2}}{h_{m-2}} - \left(\frac{1}{h_{m-2}} + \frac{1}{h_{m-3}}\right)\right)z_{m-2} + \frac{z_m}{h_{m-1}} - \frac{1}{6}\left(h_{m-2}\eta_{m-2} + 2(h_{m-2} + h_{m-1})\eta_{m-1} + h_{m-1}\eta_m\right)\right] + \lambda_m\left(\frac{-z_1 + z_2}{h_1} - \frac{-z_{m-1} + z_m}{h_{m-1}} - \frac{h_1}{6}(2\eta_1 + \eta_2) - \frac{h_{m-1}}{6}\left[\eta_{m-1} + 2\eta_m\right]\right) + \lambda_{m+1}(z_1 - z_m)$$

$$\forall \quad i = 2,3,\ldots,m-3.$$

Operando el funcional $\frac{\mathrm{dJ_a(s)}}{\mathrm{d}\lambda}$ se obtiene el sistema de ecuaciones:

$$\tilde{\mathrm{A}}\eta = 6\ddot{\mathrm{H}}z + F,$$

donde

$$\tilde{A}=\begin{bmatrix} 1 & 0 & 0 & 0 & \dots & -1 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & \ddots & \vdots \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & h_{m-2} & 2(h_{m-2}+h_{m-1}) & h_{m-1} \\ 2h_1 & h_1 & \dots & 0 & h_{m-1} & 2h_{m-1} \end{bmatrix}$$

$$\hat{H}=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{h_1} & (-\frac{1}{h_1}-\frac{1}{h_2}) & \frac{1}{h_2} & 0 & \dots & \vdots \\ 0 & \frac{1}{h_2} & (-\frac{1}{h_2}-\frac{1}{h3}) & \frac{1}{h_3} & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 0 & \frac{1}{h_{m-2}} & (-\frac{1}{h_{m-2}}-\frac{1}{h_{m-1}}) & \frac{1}{h_{m-1}} \\ -\frac{1}{h_1}-\frac{1}{h_{m-1}} & \frac{1}{h_1} & 0 & \dots & \frac{1}{h_{m-1}} & 0 \end{bmatrix}$$

La matriz $\tilde{A}$ es invertible ya que su determinante es distinto de cero. Esto se comprueba sumando a la última columna la primera y desarrollando el determinante por la primera fila, ya que se llega a una matriz estrictamente diagonal dominante que como se sabe tiene determinante distinto de cero.

Realizando la siguiente operación al funcional $\frac{dJ_a(s)}{d\eta}$, obtenemos m ecuaciones más

$$\frac{2}{3}h_1\eta_1+\frac{h_1}{3}\eta_2+\lambda_1-\frac{h_1}{6}\lambda_2-\lambda_m\frac{h_1}{3}=0,$$

$$\frac{h_1}{3}\eta_1+\frac{2}{3}(h_1+h_2)\eta_2+\frac{h_2}{3}\eta_3-\frac{1}{3}(h_1+h_2)\lambda_2-\frac{h_2}{6}\lambda_3-\lambda_m\frac{h_1}{6}=0,$$

$$\vdots$$

$$\frac{h_{i-1}}{3}\eta_{i-1}+\frac{2}{3}(h_{i-1}+h_i)\eta_i+\frac{h_i}{3}\eta_{i+1}-\frac{h_{i-1}}{6}\lambda_{i-1}-\frac{1}{3}(h_{i-1}+h_i)\lambda_i-\frac{h_i}{6}\lambda_{i+1}=0,$$

$$\vdots$$

$$\frac{h_{m-2}}{3}\eta_{m-2}+\frac{2}{3}(h_{m-2}+h_{m-1})\eta_{m-1}+\frac{h_{m-1}}{3}\eta_m-\frac{h_{m-2}}{6}\lambda_{m-2}-\frac{1}{3}(h_{m-2}+h_{m-1})\lambda_{m-1}-\lambda_m\frac{h_{m-1}}{6}=0,$$

$$-\lambda_1+\frac{2}{3}h_{m-1}\eta_m+\frac{h_{m-1}}{3}\eta_{m-1}-\frac{h_{m-1}}{6}\lambda_{m-1}-\lambda_m\frac{h_{m-1}}{3}=0,$$

$$\forall \qquad i = 3, \dots, m-2.$$

Matricialmente puede ser expresado como

$$U\,\eta = V\lambda,$$

donde

$$\mathrm{U}=\begin{bmatrix} \frac{2h_1}{3} & \frac{h_1}{3} & 0 & 0 & \dots & 0 \\ \frac{h_1}{3} & \frac{2(h_1+h_2)}{3} & \frac{h_2}{3} & 0 & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \frac{h_{m-2}}{3} & \frac{2(h_{m-2}+h_{m-1})}{3} & \frac{h_{m-1}}{3} \\ 0 & 0 & 0 & 0 & \frac{h_{m-1}}{3} & \frac{2h_{m-1}}{3} \end{bmatrix}$$

$$\mathrm{V}=\begin{bmatrix} -1 & \frac{h_1}{6} & 0 & \dots & 0 & \frac{h_1}{3} \\ 0 & \frac{(h_1+h_2)}{3} & \frac{h_2}{6} & 0 & \ddots & \frac{h_1}{6} \\ \vdots & \frac{h_2}{6} & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \frac{h_{m-2}}{6} & \frac{2(h_{m-2}+h_{m-1})}{3} & \frac{h_{m-1}}{6} \\ 1 & 0 & 0 & 0 & \frac{h_{m-1}}{6} & \frac{h_{m-1}}{3} \end{bmatrix}$$

La matriz V es invertible ya que su determinante es distinto de cero. Esto se comprueba sumando a la última fila la primera y desarrollando el determinante por la primera columna, ya que se llega a una matriz estrictamente diagonal dominante que como se sabe tiene determinante distinto de cero.

Derivando el funcional, respecto de los z_i, $\frac{dJ_a(s)}{dz}$,

$$\frac{2}{\rho_1}(z_1 - y_1) + \frac{\lambda_2}{h_1} - \frac{\lambda_m}{h_1} + \lambda_{m+1} = 0,$$

$$\frac{2}{\rho_2}(z_2 - y_2) - (\frac{1}{h_1} + \frac{1}{h_2})\lambda_2 + \frac{\lambda_3}{h_2} + \frac{\lambda_m}{h_1} = 0,$$

$$\vdots$$

$$\frac{2}{\rho_i}(z_i - y_i) + \lambda_{i-1}\frac{1}{h_{i-1}} - (\frac{1}{h_{i-1}} + \frac{1}{h_i})\lambda_i + \frac{\lambda_{i+1}}{h_i} = 0,$$

$$\vdots$$

$$\frac{2}{\rho_{m-1}}(z_{m-1} - y_{m-1}) - (\frac{1}{h_{m-2}} + \frac{1}{h_{m-1}})\lambda_{m-1} + \frac{\lambda_{m-2}}{h_{m-2}} + \lambda_m\frac{1}{h_{m-1}} = 0,$$

$$\frac{2}{\rho_m}(z_m - y_m) + \frac{\lambda_{m-1}}{h_{m-1}} - \frac{\lambda_m}{h_{m-1}} - \lambda_{m+1} = 0,$$

$$\forall \qquad i = 3, \dots, m-2.$$

De la primera y la última ecuación y de la condición $z_1 = z_m$, se obtiene el sistema de ecuaciones

$$\frac{2}{\rho_1}(z_1 - y_1) + \frac{\lambda_2}{h_1} - \frac{\lambda_m}{h_1} + \lambda_{m+1} = 0,$$

$$\frac{2}{\rho_m}(z_m - y_m) + \frac{\lambda_{m-1}}{h_{m-1}} - \frac{\lambda_m}{h_{m-1}} - \lambda_{m+1} = 0,$$

$$z_1 = z_m,$$

Restando a la primera ecuación la segunda, sustituyendo la tercera y teniendo en cuenta las condiciones de compatibilidad se obtiene el valor de λ_{m+1}

$$2\lambda_{m+1} = \frac{\lambda_2 - \lambda_m}{h_1} - \frac{\lambda_{m-1} - \lambda_m}{h_{m-1}},$$

$$\lambda_{m+1} = \frac{1}{2}\frac{\lambda_2 - \lambda_m}{h_1} - \frac{1}{2}\frac{\lambda_{m-1} - \lambda_m}{h_{m-1}}.$$

Sustituyendo este valor de λ_{m+1}en el sistema de ecuaciones obtenido derivando el funcional respecto de los z_i

$$\frac{2}{\rho_1}(z_1 - y_1) + \frac{3}{2}\frac{\lambda_2 - \lambda_m}{h_1} - \frac{1}{2}\frac{\lambda_{m-1} - \lambda_m}{h_{m-1}} = 0,$$

$$\frac{2}{\rho_2}(z_2 - y_2) - (\frac{1}{h_1} + \frac{1}{h_2})\lambda_2 + \frac{\lambda_3}{h_2} + \frac{\lambda_m}{h_1} = 0,$$

$$\vdots$$

$$\frac{2}{\rho_i}(z_i - y_i) + \lambda_{i-1}\frac{1}{h_{i-1}} - (\frac{1}{h_{i-1}} + \frac{1}{h_i})\lambda_i + \frac{\lambda_{i+1}}{h_i} = 0,$$

$$\vdots$$

$$\frac{2}{\rho_{m-1}}(z_{m-1} - y_{m-1}) - (\frac{1}{h_{m-2}} + \frac{1}{h_{m-1}})\lambda_{m-1} + \frac{\lambda_{m-2}}{h_{m-2}} + \lambda_m\frac{1}{h_{m-1}} = 0,$$

$$\frac{2}{\rho_m}(z_1 - y_m) + \frac{3}{2}\frac{\lambda_{m-1} - \lambda_m}{h_{m-1}} - \frac{1}{2}\frac{\lambda_2 - \lambda_m}{h_1} = 0,$$

$$\forall \qquad i = 3, \dots, m-2.$$

Despejando z,

$$z_1 = y_1 - \frac{\rho_1}{2}\left(\frac{3}{2}\frac{\lambda_2 - \lambda_m}{h_1} - \frac{1}{2}\frac{\lambda_{m-1} - \lambda_m}{h_{m-1}}\right),$$

$$z_2 = y_2 - \frac{\rho_2}{2}\left(-\left(\frac{1}{h_1} + \frac{1}{h_2}\right)\lambda_2 + \frac{\lambda_3}{h_2} + \frac{\lambda_m}{h_1}\right),$$

$$\vdots$$

$$z_i = y_i - \frac{\rho_i}{2}\left(-\left(\frac{1}{h_{i-1}} + \frac{1}{h_i}\right)\lambda_i + \frac{\lambda_{i+1}}{h_i} + \frac{\lambda_{i-1}}{h_{i-1}}\right),$$

$$\vdots$$

$$z_{m-1} = y_{m-1} - \frac{\rho_{m-1}}{2}\left(-\left(\frac{1}{h_{m-2}} + \frac{1}{h_{m-1}}\right)\lambda_{m-1} + \frac{\lambda_{m-2}}{h_{m-2}} + \lambda_m \frac{1}{h_{m-1}}\right),$$

$$z_m = z_1 = y_m - \frac{\rho_m}{2}\left(\frac{3}{2}\frac{\lambda_{m-1} - \lambda_m}{h_{m-1}} - \frac{1}{2}\frac{\lambda_2 - \lambda_m}{h_1}\right),$$

$$\forall \quad i = 3, \dots, m-2.$$

Matricialmente puede expresarse como

$$Z = Y + \frac{1}{2}\rho \, \mathrm{J} \, \lambda,$$

donde

$$\mathrm{J} = \begin{bmatrix} 0 & -\frac{1}{2h_1} & 0 & \dots & -\frac{1}{h_{m-1}} & -\frac{1}{h_1} + \frac{1}{h_{m-1}} \\ 0 & \frac{1}{h_1} + \frac{1}{h_2} & -\frac{1}{h_2} & \ddots & \ddots & -\frac{1}{h_1} \\ 0 & \frac{1}{h_2} & \frac{1}{h_2} + \frac{1}{h_3} & \ddots & 0 & \vdots \\ 0 & \ddots & \ddots & \ddots & -\frac{1}{h_{m-2}} & \vdots \\ \vdots & \ddots & \ddots & \frac{1}{h_{m-2}} & \frac{1}{h_{m-2}} + \frac{1}{h_{m-1}} & -\frac{1}{h_{m-1}} \\ 0 & -\frac{1}{2h_1} & \dots & 0 & -\frac{1}{h_{m-1}} & \frac{1}{h_1} + \frac{1}{h_{m-1}} \end{bmatrix}.$$

3.9.5 Otros casos.

Existen otras posibles combinaciones además de las ya estudiadas en este capítulo, sin embargo, puesto que serán las mismas sólo que variando una de las dos ecuaciones frontera, se desarrollarán de igual modo demostrando que son matrices estrictamente diagonales dominantes y según el enunciado del lema de Hadamard serán por tanto, invertibles y su sistema tendrá solución compatible determinada.

4. Splines cúbicos para generación de curvas

Proporcionaremos un conjunto de puntos conocidos, que serán una serie de coordenadas, los cuales denominaremos puntos de control. Estos puntos de control son los que servirán de directrices y con los que después ajustaremos funciones polinómicas continuas mediante una de las siguientes formas que se pueden ver representadas en las figuras 4.2, 4.3 y 4.4.

4.1 Interpolación de splines cúbicos

La interpolación de splines cúbicos tiene como idea central usar segmentos de polinomios de grado 3 entre pares coordenados de datos y unir cada uno de ellos adecuadamente para generar una curva suave.

El proceso es una **interpolación** del conjunto de puntos de control cuando las secciones polinómicas se ajustan de modo que la curva pasa a través de cada punto de control; lo que proporciona condiciones. Las restantes ecuaciones para la construcción del spline mediante interpolación serán los valores de las derivadas menores del spline en los extremos del segmento analizado, y las condiciones de contorno o de frontera.

En la Figura 4.1 vemos un ejemplo de ajuste con splines de grado 1. Vemos que cada segmento entre dos puntos está formado por una línea recta.

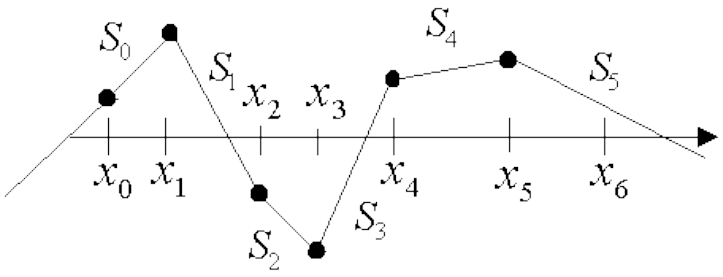

Figura 4.1 Representación de una curva que realiza una interpolación.

(Fuente: users.dsic.upv.es/asignaturas/eui/cnu/libro/tema7/tema76).

Una función spline está formada por varios polinomios cada uno definido en un intervalo y que se unen entre si bajo ciertas condiciones de continuidad.

Condiciones:

1. El trazador se ajusta a cada uno de los puntos.
2. El trazador es continuo.
3. La pendiente es continua.

4. La curvatura es continua.

En la Figura 4.2 vemos en trazo continuo (azul) la función original de la que vienen los datos, y en trazo discontinuo (rojo) la interpolación por splines cúbicos.

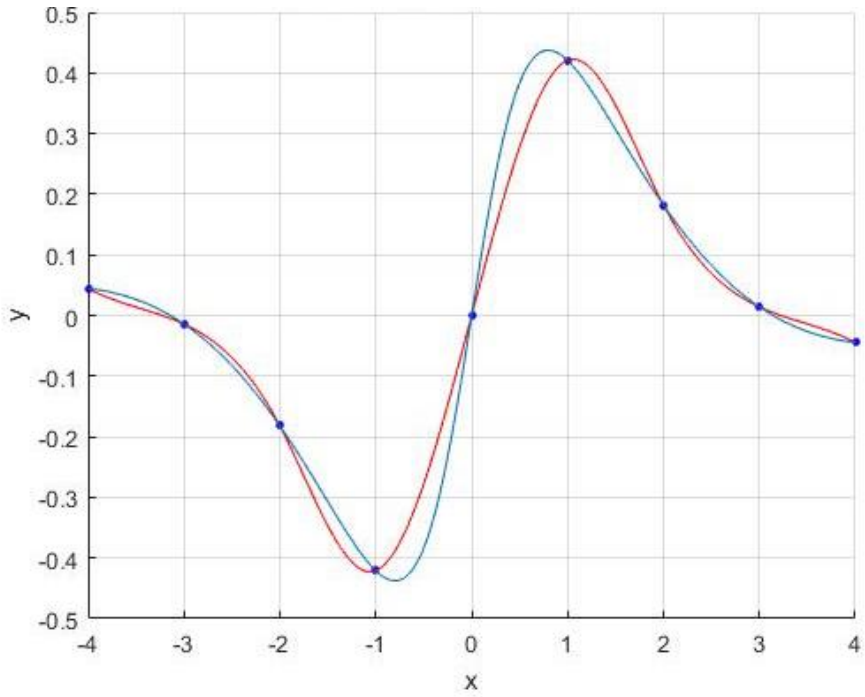

Figura 4.2 Interpolación por splines en Matlab.

(Fuente: www.sc.ehu.es/sbweb/fisica3/datos/regresion/interpolacion).

4.2 Suavizamiento de splines cúbicos

El suavizamiento de splines cúbicos tiene como objetivo encontrar una función que pueda interpolar los nodos o solo seguir la tendencia de ellos.

La curva realiza un **suavizamiento** al conjunto de puntos de control cuando los polinomios se ajustan a la trayectoria general marcada por los puntos de control sin pasar necesariamente a través ellos.

La medida de esta cercanía se puede definir de diferentes maneras, lo que genera una gran variedad de splines suavizantes.

En la Figura 4.3 y 4.4 se observa gráficamente el efecto del suavizamiento de la curva.

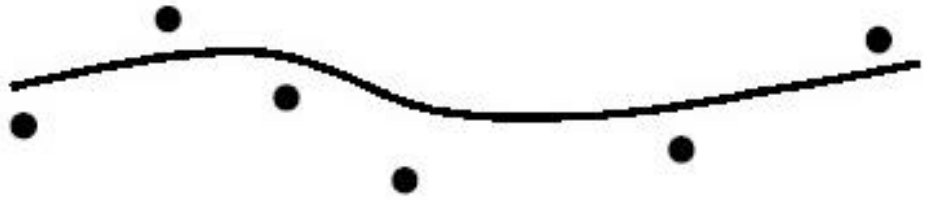

Figura 4.3 Representación de una curva que realiza un suavizamiento.

(Fuente: bibliografía [4]).

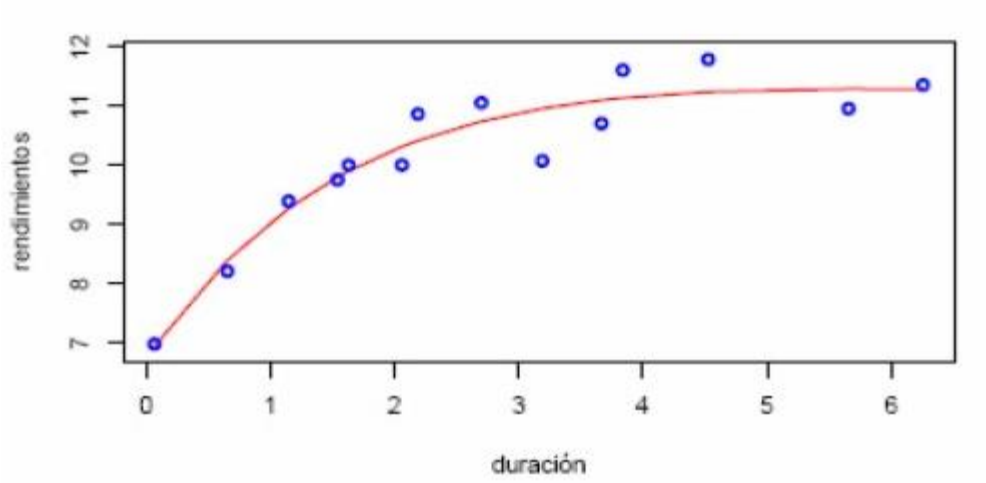

Figura 4.4 Suavizamiento por splines.

(Fuente: www.scielo.org.co).

4.3 Generación de Curvas Planas

Vamos a explicar el proceso a través de un ejemplo.

Partimos de las coordenadas de los puntos de control.

$$X = (1, 3, 4, 5, 7, 10)$$

$$Y = (5, 1, 3, -1, 4, 2)$$

En la Figura 4.5 se pueden observar los puntos representados en el plano para los cuales vamos a hacer para una curva spline. En este caso elegiremos splines cúbicos interpolantes:

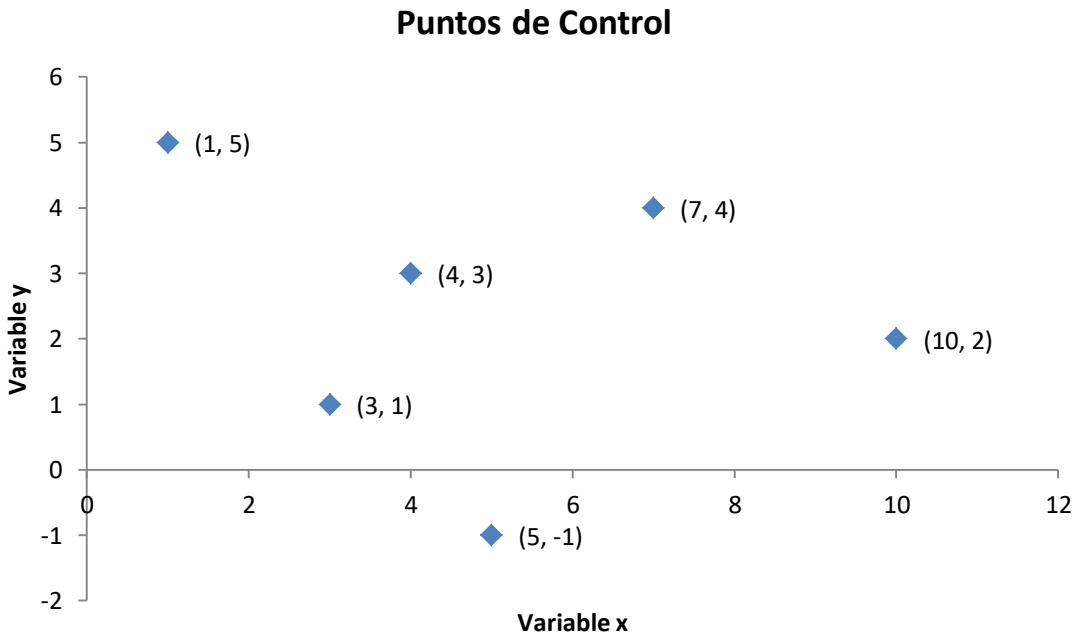

Figura 4.5 Representación de los puntos de control en el plano

A partir de estos puntos de control (X_n, Y_n) en vez de usar un solo polinomio para interpolar los datos, vamos a usar polinomios entre los puntos de control y así unir cada uno de los valores interpolados adecuadamente para conseguir un mejor ajuste de los datos.

El proceso se realiza interpolando para splines cúbicos primero en la variable 'x' y después en la variable 'y' utilizando una variable independiente auxiliar 't' que representa el tiempo. Así consideramos las funciones X(t), Y(t), y a cada una de ellas le aplicamos splines separadamente.

Para X, siendo:

$t = 1 \qquad X(1) = 1$

$t = 2 \qquad X(2) = 3$

$t = 3 \qquad X(3) = 4$

$t = 4 \qquad X(4) = 5$

$t = 5 \qquad X(5) = 7$

$t = 6 \qquad X(6) = 10$

Aplicamos splines cúbicos interpolantes para aproximar la función $X(t)$ y evaluamos en un mallado fino intermedio.

Definimos un espaciado 'h' y el mallado $t_j = t_{j-1} + h$ con $t_0 = 1$ y evaluamos el spline en t_j obteniendo $x_j = S_x(t_j)$, es decir $x_j = valores\ de\ las\ x\ interpoladas$.

En la Figura 4.6 podemos ver los valores de X interpolados entre cada punto de control para t=1, 2, 3, 4, 5 y 6.

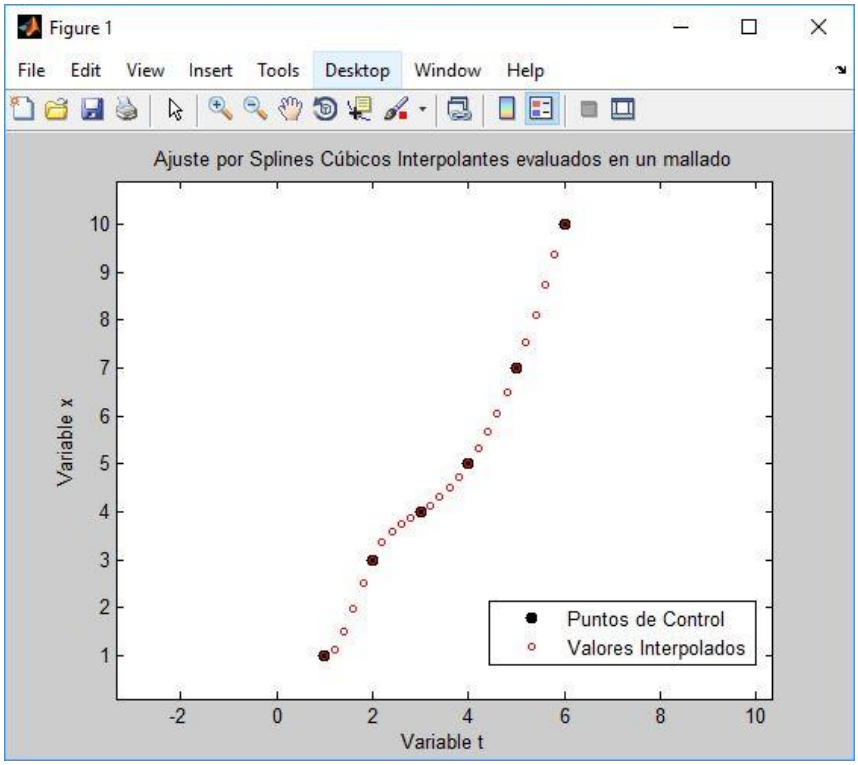

Figura 4.6 Valores de 'x' para cada punto 't'.

Y en la Figura 4.7 las funciones de los splines cúbicos que aproximan $x(t)$.

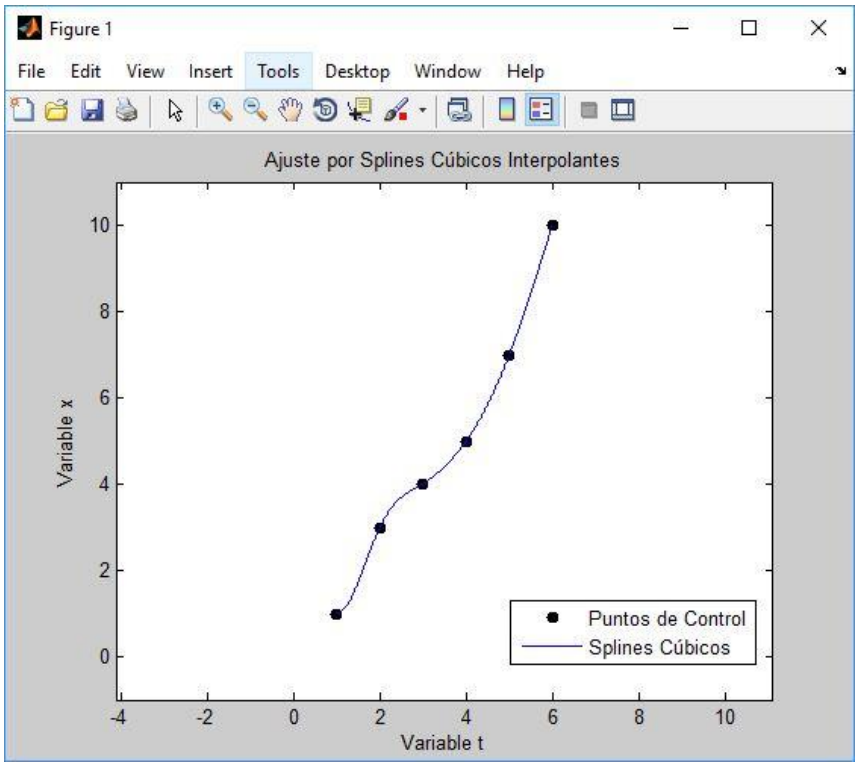

Figura 4.7 Representación función splines que aproximan x(t).

Para Y: Hacemos exactamente lo mismo que con X, obteniendo en nuestro ejemplo los valores $y_j = S_y(t_j)$ representados en la Figura 4.8.

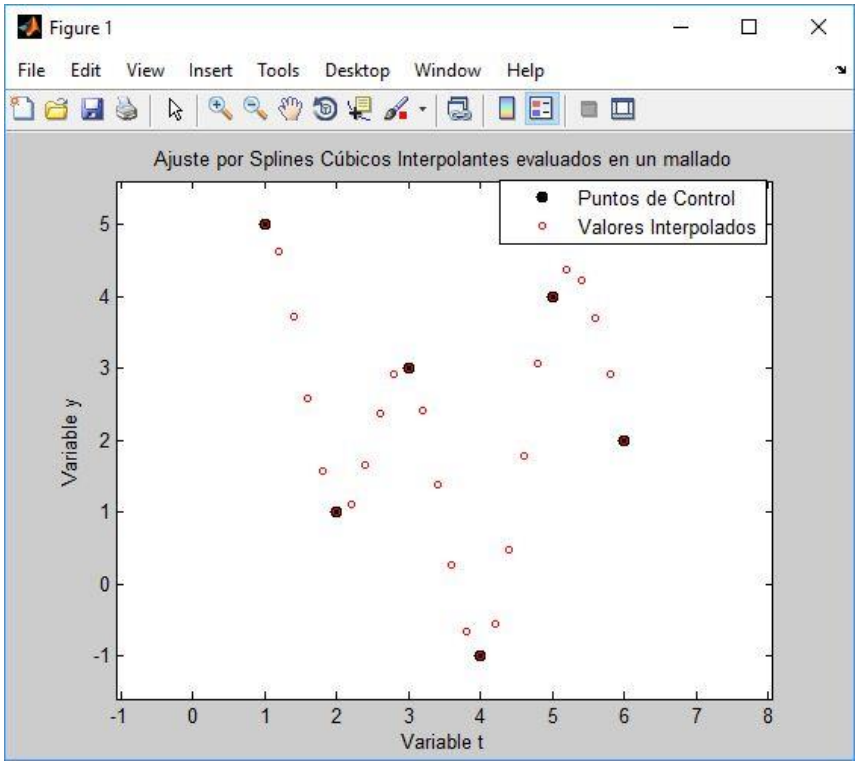

Figura 4.8 Valores de 'y' para cada punto 't'.

Las funciones splines correspondientes se ven en la Figura 4.9.

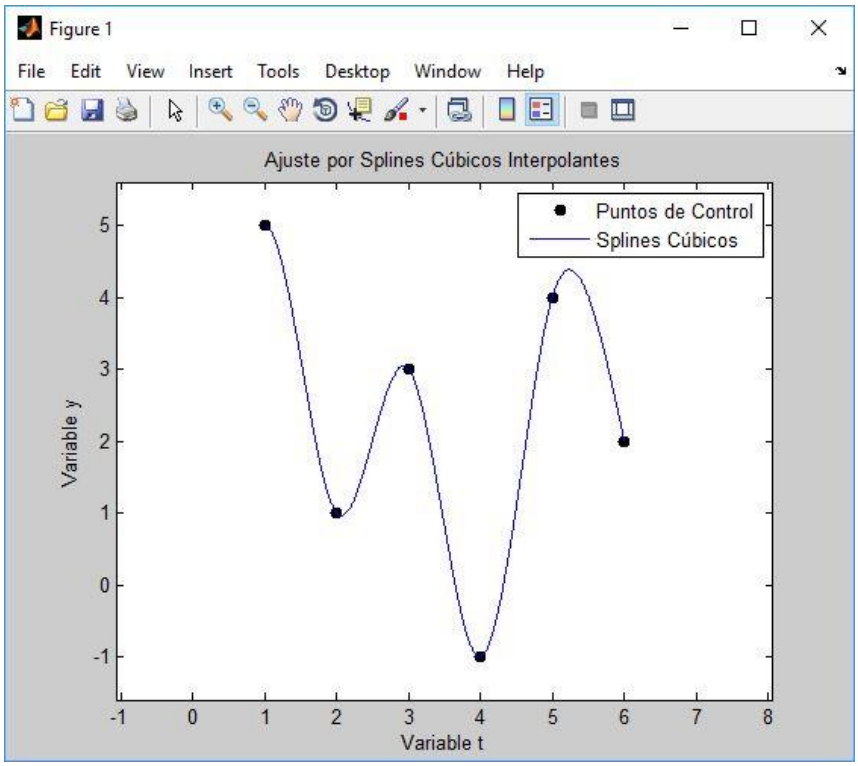

Figura 4.9 Representación función splines que aproximan y(t).

Tras calcular los valores interpolados y las funciones splines para X e Y, generamos la curva compuesta por las coordenadas (x_j, y_j), como puede observarse en la Figura 4.10.

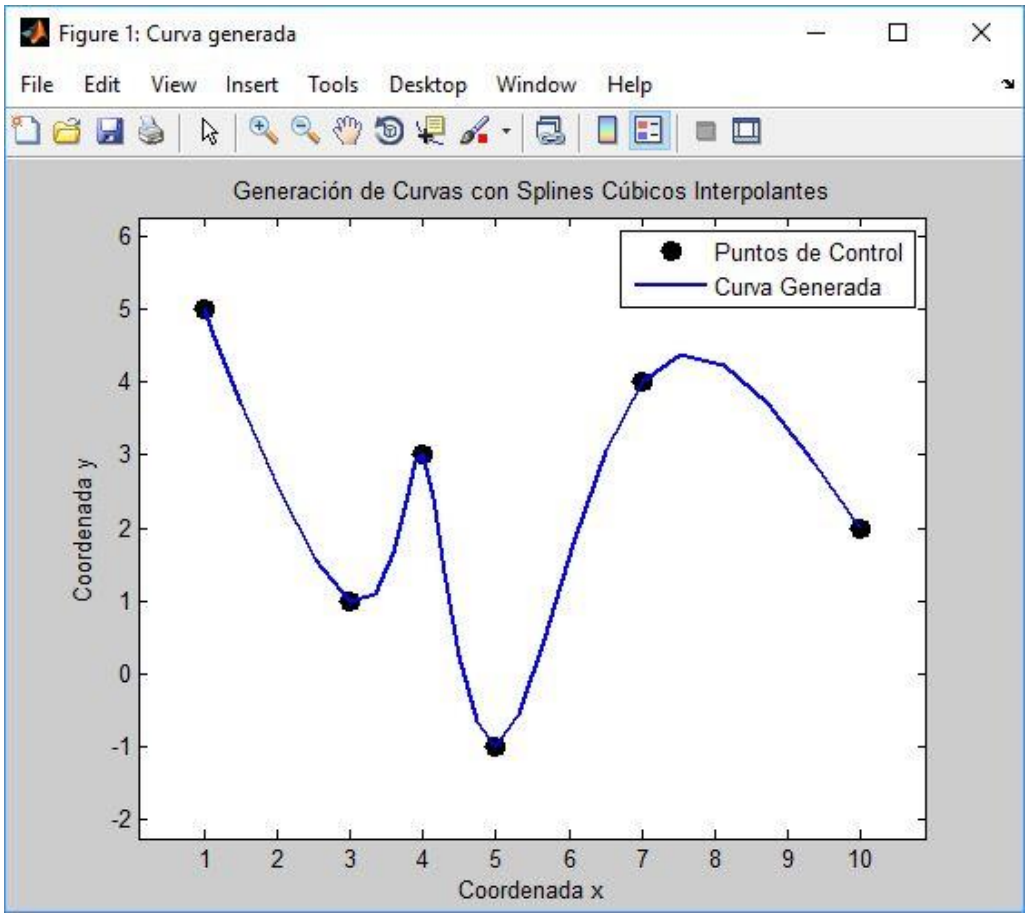

Figura 4.10 Representación curva (xj,yj) .

4.4 Generación de Curvas en el Espacio.

Para generar curvas en el espacio simplemente operamos como en el apartado anterior pero en este caso con $x(t), y(t), z(t)$.

Vamos a montar el proceso con el siguiente ejemplo donde aplicamos splines cúbicos suavizantes para aproximar los siguientes datos.

$$X = (1,2,7,10,11)$$

$$Y = (1,3,8,9,11)$$

$$Z = (1,2,3,4,11)$$

En la Figura 4.11 podemos ver los valores de X interpolados entre cada punto de control para t=1, 2, 3, 4 y 5.

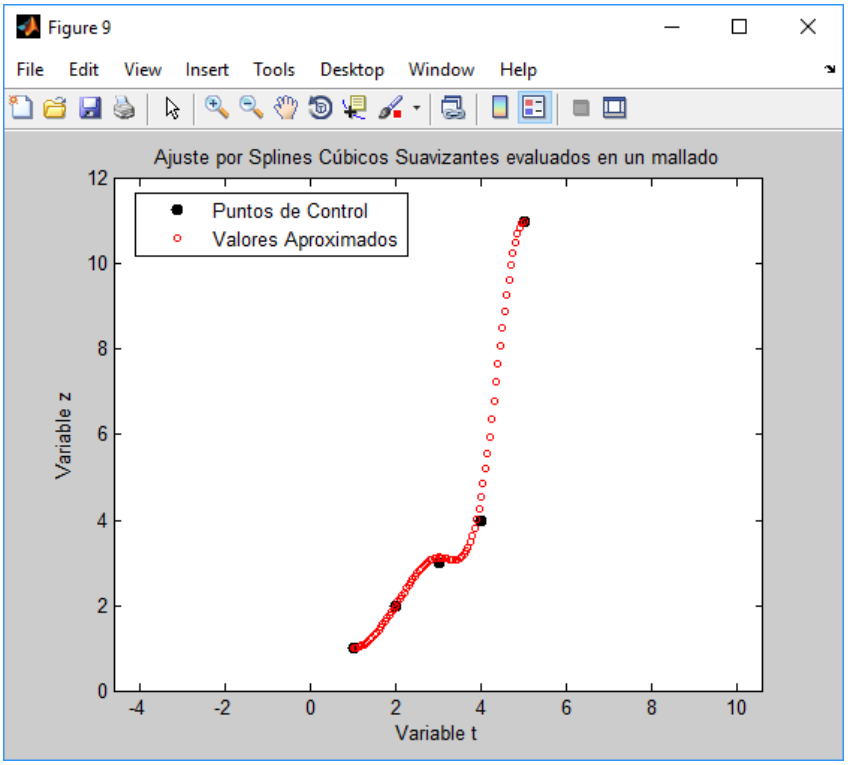

Figura 4.11 Valores de 'z' para cada punto 't'.

Y en la Figura 4.12 las funciones de los splines cúbicos que aproximan $z(t)$.

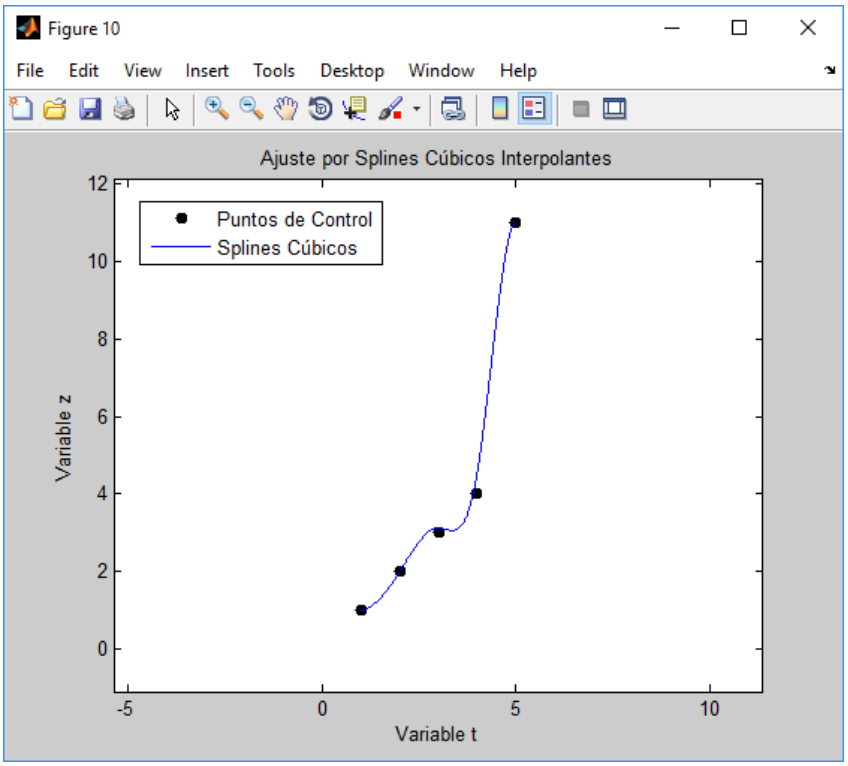

Figura 4.12 Representación función splines que aproximan z(t)

Tras calcular los valores interpolados y las funciones splines para X, Y y Z, generamos la curva compuesta por las coordenadas (x_j, y_j, z_j), como puede observarse en la Figura 4.13.

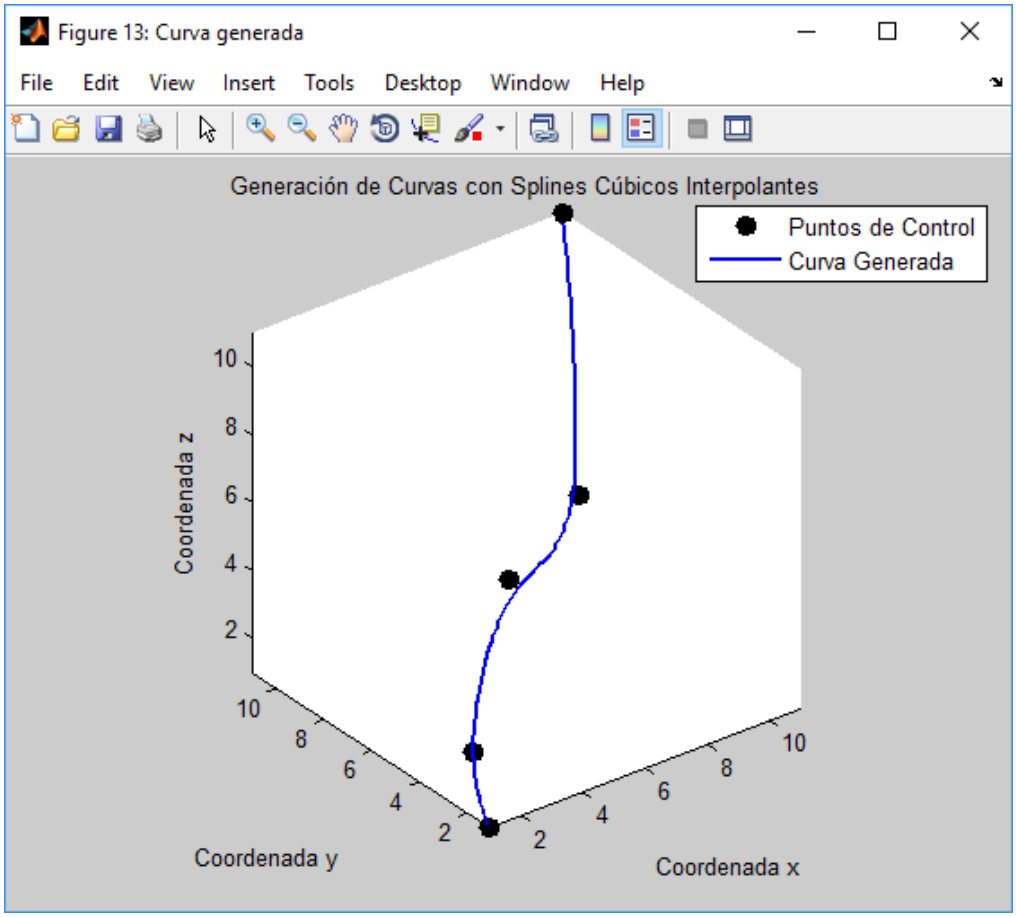

Figura 4.13 Representación curva (xj,yj,zj) en el espacio.

5. Interfaz gráfica

Para el desarrollo de la interfaz gráfica se ha utilizado Matlab, para ello se ha elegido el entorno de trabajo de Matlab Guide (GUI), el cual es muy intuitivo como podremos ver a continuación. Con esta interfaz cualquier usuario tiene la posibilidad de resolver problemas de aproximación mediante el uso de splines cúbicos.

5.1 Ejecución de la Interfaz Gráfica.

Para ejecutar la interfaz gráfica, el usuario deberá realizar los siguientes pasos:

I. *Abrir* el programa MATLAB.

II. Seleccionar el *directorio de trabajo* de Matlab, en nuestro caso se llama "GeneracionCurvas". En la barra superior de Matlab debe aparecer, por ejemplo, la secuencia que se muestra en la Figura 5.1:

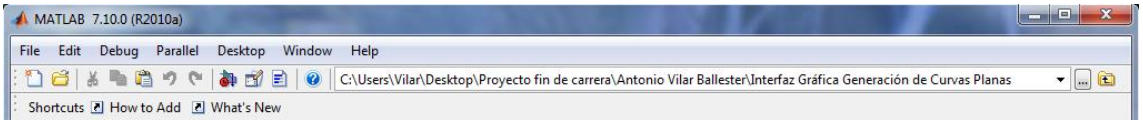

Figura 5.1 Directorio de trabajo de Matlab para ejecutar la Interfaz.

III. Escribir el nombre de la interfaz en la línea de comandos (pantalla principal de Matlab), como se muestra en la Figura 5.2. En nuestro caso habrá que escribir ''GeneracionCurvas''.

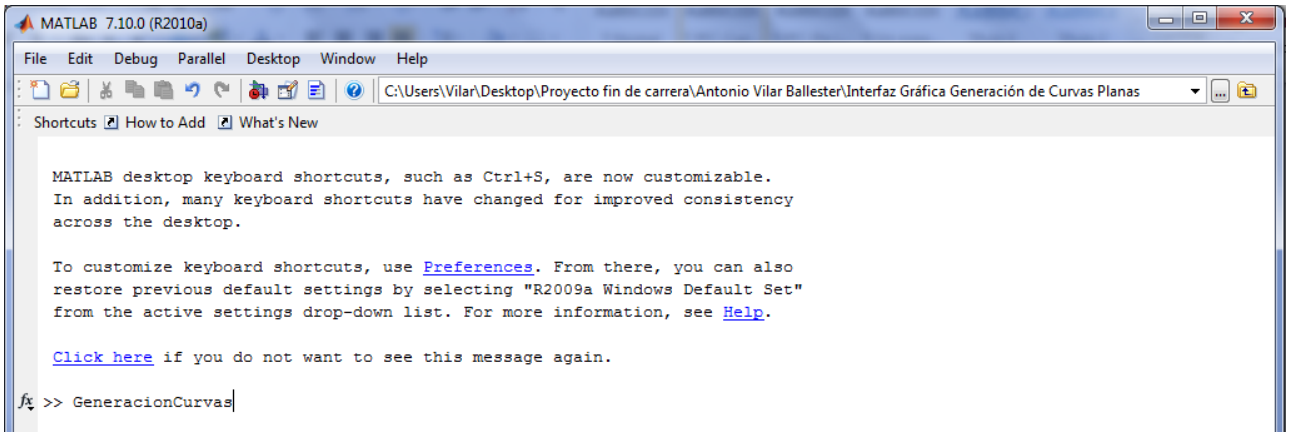

Figura 5.2 Ejecución de la interfaz gráfica.

Y nos redireccionará a la interfaz siguiente, en donde tenemos dos opciones:

- Splines Cúbicos Interpolantes.
- Splines Cúbicos Suavizantes.

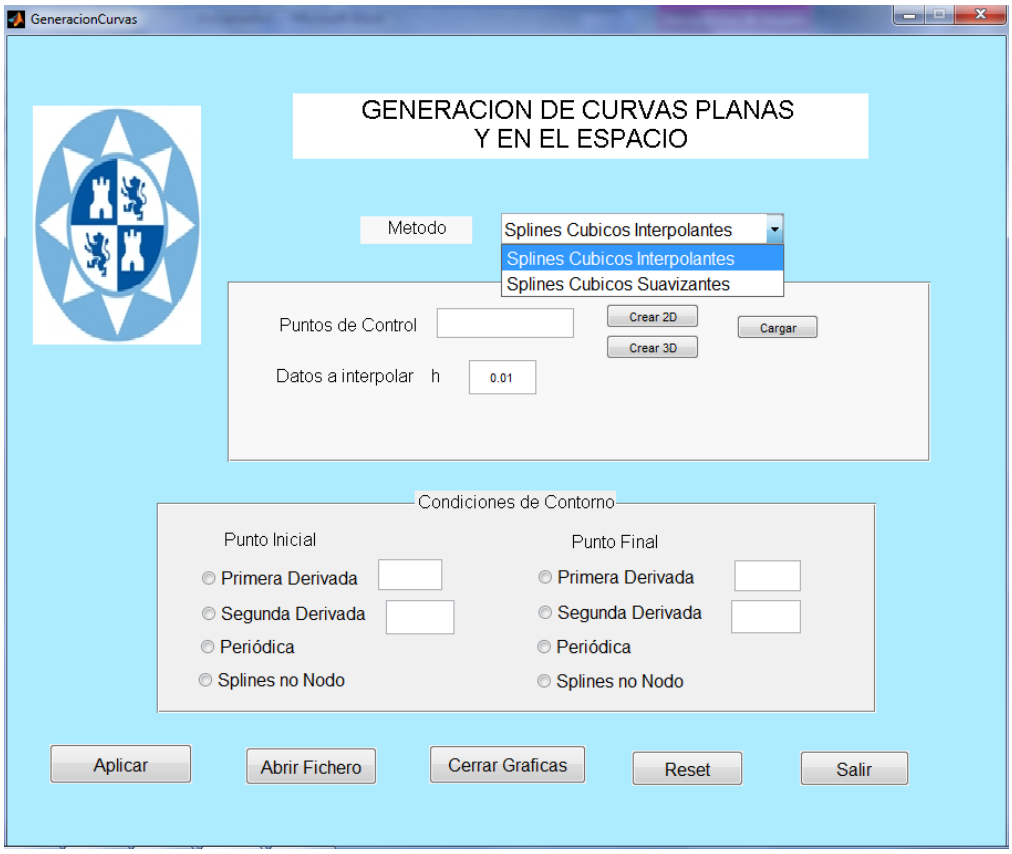

Figura 5.3 Pantalla principal de la interfaz gráfica.

En el caso de elegir "Splines Cúbicos Interpolantes", nos saldrá la interfaz tal y como nos la muestra la figura anterior.

A continuación explicaremos cómo ejecutar cada una de las opciones, splines cúbicos interpolantes o splines cúbicos suavizantes.

5.2 INTERFAZ DE SPLINES CÚBICOS INTERPOLANTES

Vamos a explicar cada uno de los paneles a complementar antes de la ejecución del programa. Empezaremos por el panel de carga de datos iniciales.

5.2.1 Panel de carga de datos iniciales.

En este panel debemos especificar las coordenadas de los nodos y las abscisas de los datos a interpolar.

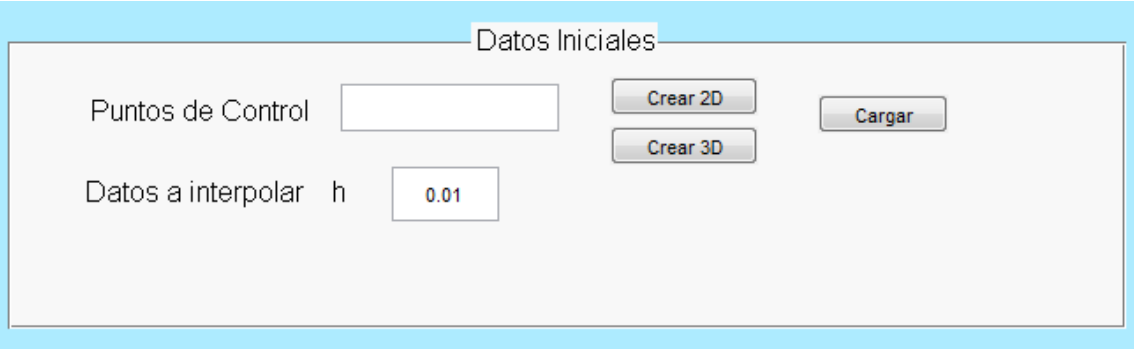

Figura 5.4 Panel de carga de los datos iniciales.

En los datos iniciales nos dan la opción de crear la gráfica final tanto en 2D como 3D según los datos que nosotros introduzcamos o bien cargar un archivo en el cual ya vengan los datos deseados.

Al pulsar el botón "Crear 2D" nos saldrán los siguientes datos por defecto, los cuales podremos modificar para introducir los datos que queremos tanto para el eje de abscisas como para el de ordenadas.

```
function [x,y,z]=crear_datos2D()
% completar las abscisas de los puntos de control
x=[1,2,7,10,11];
% completar las ordenadas de los puntos de control
y=[1,3,8,9,15];
% coordenada z de los puntos de control
z=zeros(size(x));
```

Y lo mismo con el botón "3D", añadiéndose los datos en la coordenada z:

```
function [x,y,z]=crear_datos3D()
% completar las abscisas de los puntos de control
x=[1,2,7,10,11];

% completar las ordenadas de los puntos de control
y=[1,3,8,9,15];

% coordenada z de los puntos de control
z=[1,2,3,4,5];
```

5.2.2 Panel de condiciones de contorno

En este panel es donde se han de especificar las condiciones de contorno en cada extremo.

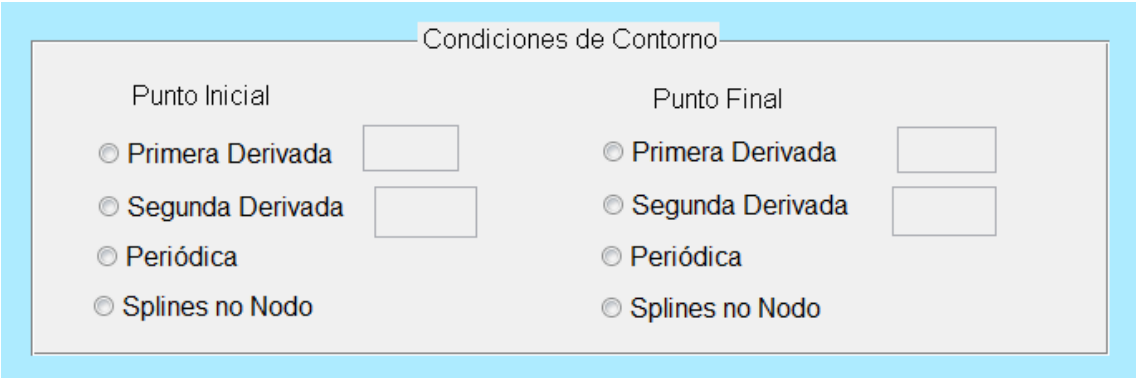

Figura 5.5 Panel de condiciones de contorno.

En cada extremo podemos utilizar una condición de contorno distinto, con la salvedad de que en el caso de seleccionar periódica, esta condición corresponde a todo el Spline, por lo que si seleccionas en un extremo "Periódica" en el otro se seleccionará automáticamente esta misma condición. Por la misma razón al activar otro tipo de condición en el panel se desactivarán las dos "Periódica", evitando así cometer el error de seleccionar que la curva es periódica sólo en un extremo, lo que no tiene sentido.

Si conocemos el valor de la primera derivada en el extremo, que corresponde al giro o pendiente de la curva en el mismo, es aconsejable utilizar este dato. Al seleccionar como condición "Primera derivada" se activará el cuadro de texto y podremos escribir el valor numérico. En cambio, si lo que conocemos es el valor de la segunda derivada en los extremos, se selecciona como condición ésta y habrá que completar el cuadro de texto que se activa a su derecha.

Si por cualquier razón se quedara un cuadro de texto con algún valor numérico desactivado porque se ha decidido más tarde utilizar otro tipo de condición, no hay que preocuparse por él, puesto que no se usará este dato a la hora de construir el Spline.

La condición de "Splines no nodo" hace referencia a la condición de cuarto tipo en la que la tercera derivada en el extremo también es continua. Por lo que para usarla en ambos extremos mínimo tendrán que existir cuatro nodos, ya que en caso de ser tres nodos la condición en ambos extremos sería la misma y sólo formaría una de las dos ecuaciones adicionales que se necesitan en la construcción del Spline.

Generalmente este último tipo de condiciones dan buenos resultados en caso de no conocer los valores de las derivadas primera o segunda. Aunque también podemos hacer uso de los Splines cúbicos naturales si indicamos como 0 el valor de las segundas derivadas en cada extremo; esto suele darnos resultado cuando no tenemos ninguna otra posibilidad, pero en

caso de que el valor de dicha derivada no sea próximo a 0 la precisión del Spline se verá mermada. Si no conocemos el valor exacto de las derivadas pero sí podemos aproximarlo, será beneficioso para la suavidad y precisión de la curva el utilizar dichos valores aproximados.

Si podemos elegir entre aportar los valores de la primera o de la segunda derivada a la construcción del Spline, se aconseja especificar los valores de la primera derivada.

5.2.3 Botones de acción

Los botones de acción son aquellos que llevan a cabo distintas acciones en el programa.

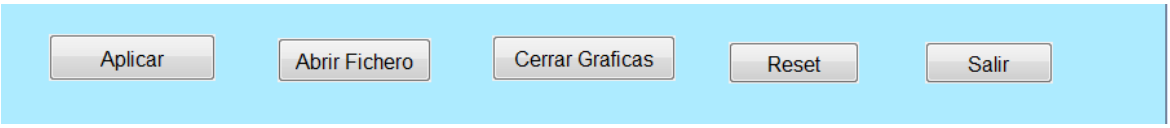

Figura 5.6 Botones de acción.

- pulsar este botón da la resolución del problema dibujando las gráficas correspondientes.
- **Abrir Fichero:** abre en el editor de Matlab los ficheros creados al aplicar el programa anteriormente.
- **Cerrar Gráficas:** cierra todas las gráficas que se hubieran abierto al aplicar el programa excepto la última, que es con la que trabajaremos habitualmente.
- **Reset:** borra todos los datos introducidos en la interfaz y desactiva todas las opciones que se hubieran marcado.
- **Salir:** cierra la interfaz gráfica.

5.2.4 Ejemplo de salida de datos

Vamos a realizar un ejemplo en 3D que así engloba las dos formas de resolver el problema, en 2D y 3D.

Al pinchar el botón de crear 3D nos aparecerá la siguiente ventana con unos datos para el eje de abscisa, el eje de ordenadas y para el eje z por defecto. Estos datos se pueden dejar tal y como están o si lo preferimos introducir nosotros los datos que deseemos.

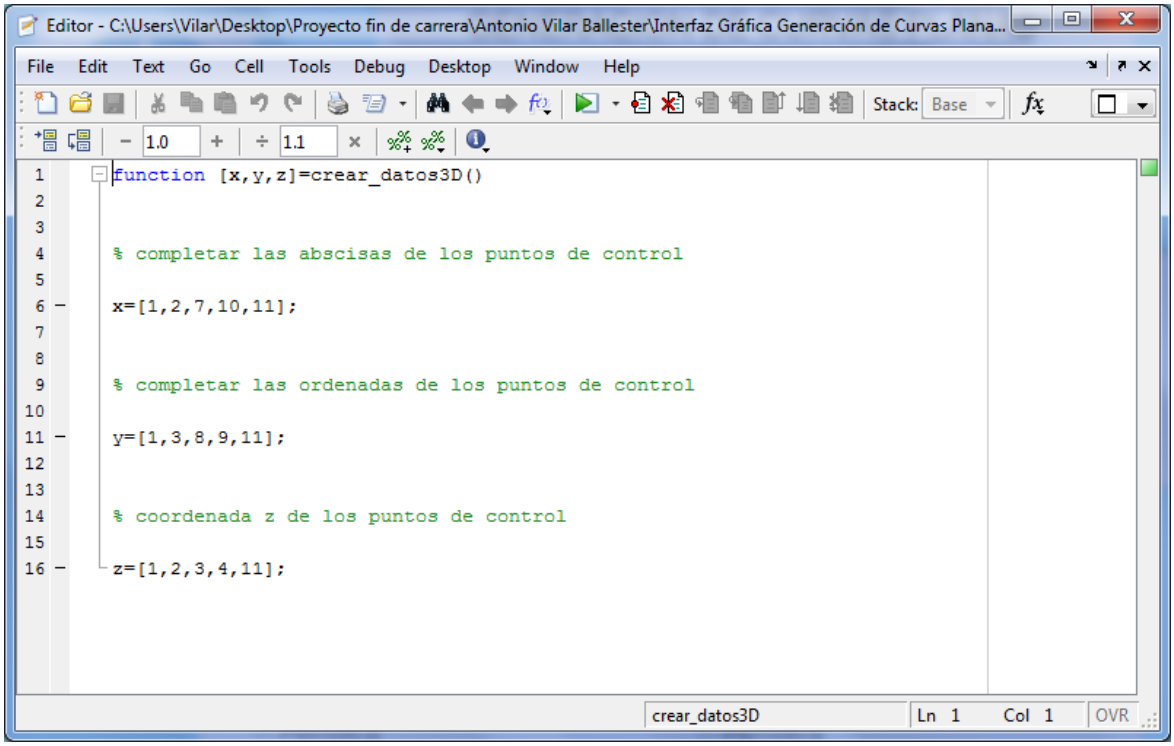

Figura 5.7 Datos ejes x, y , z.

En nuestro caso dejaremos los datos que están por defecto.

A continuación indicaremos las condiciones de contorno, en este ejemplo conocemos el valor de la primera derivada en el extremo izquierdo (1) y el valor de la primera derivada en el extremo derecho (1).

Figura 5.8 Interfaz con datos del ejemplo.

Tras introducir todos los datos en la interfaz pinchamos en el botón “Aplicar” y se mostrarán todas las gráficas.

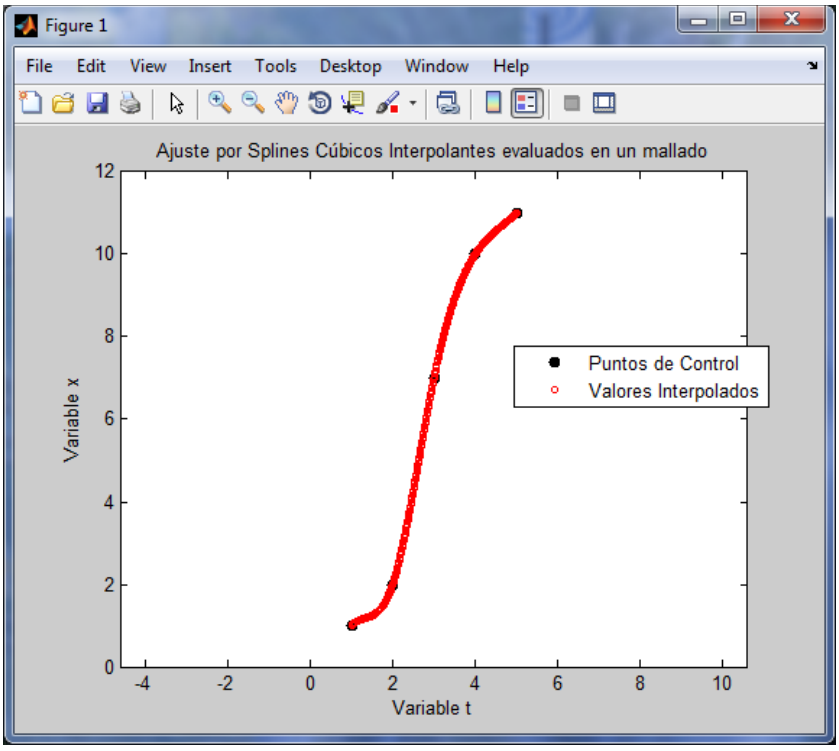

Figura 5.9 Valores interpolados, variable ‘x’.

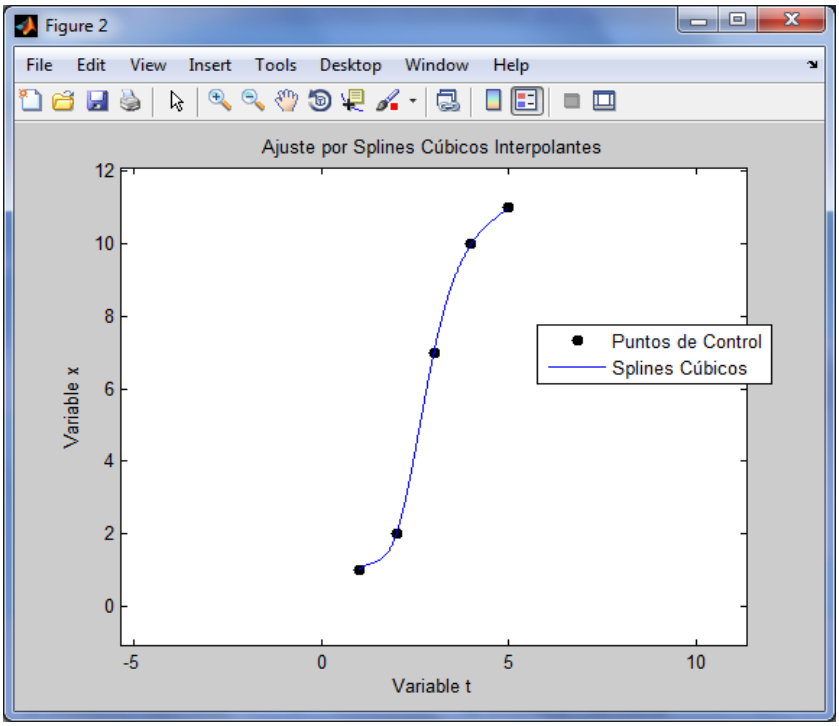

Figura 5.10 Splines cúbicos interpolantes, variable 'x'.

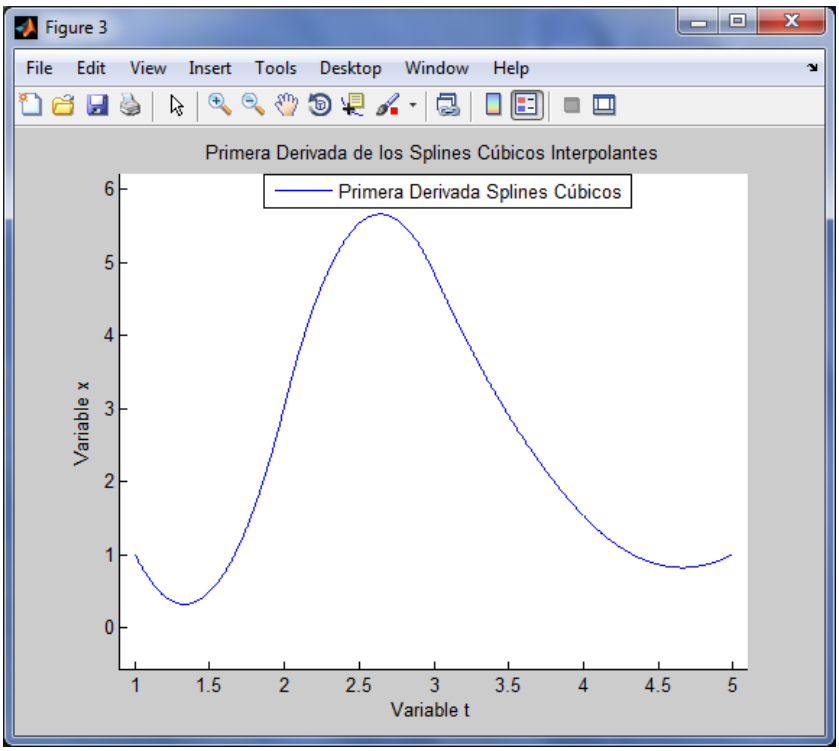

Figura 5.11 Primera derivada splines cúbicos, variable 'x'.

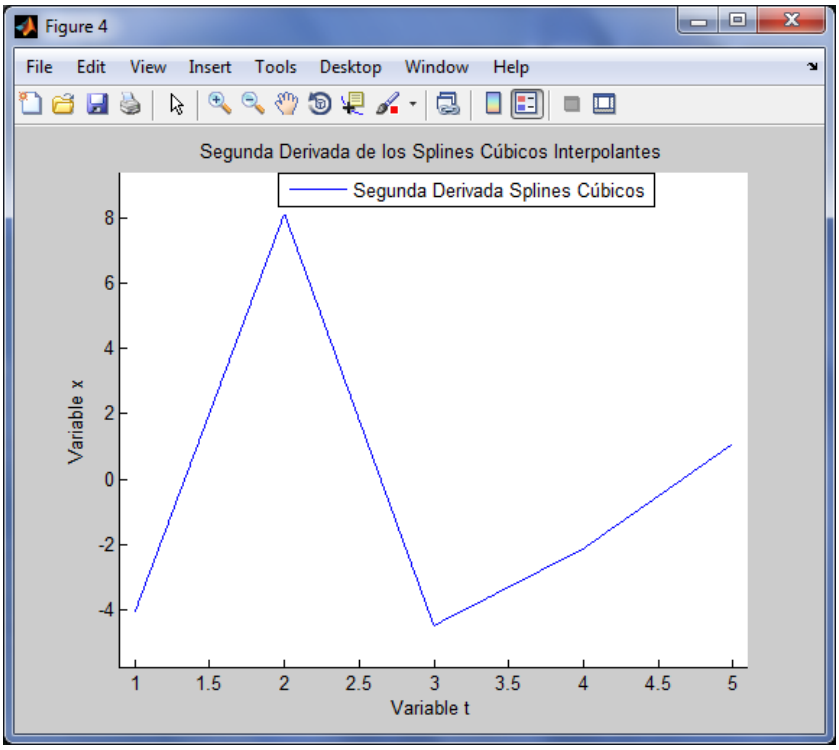

Figura 5.12 Segunda derivada splines cúbicos, variable 'x'.

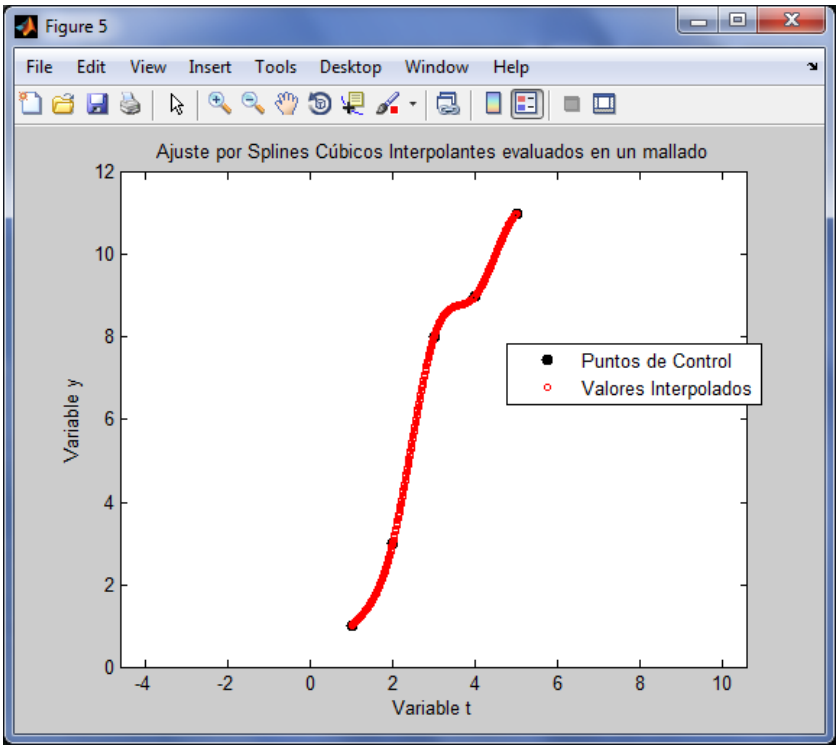

Figura 5.13 Valores interpolados, variable 'y'.

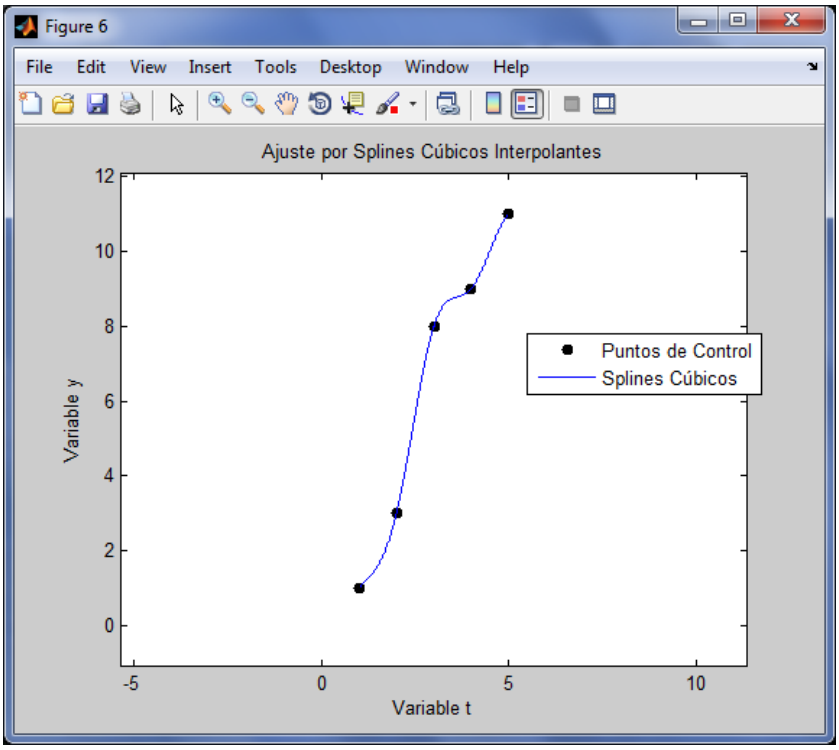

Figura 5.14 Splines cúbicos, variable 'y'.

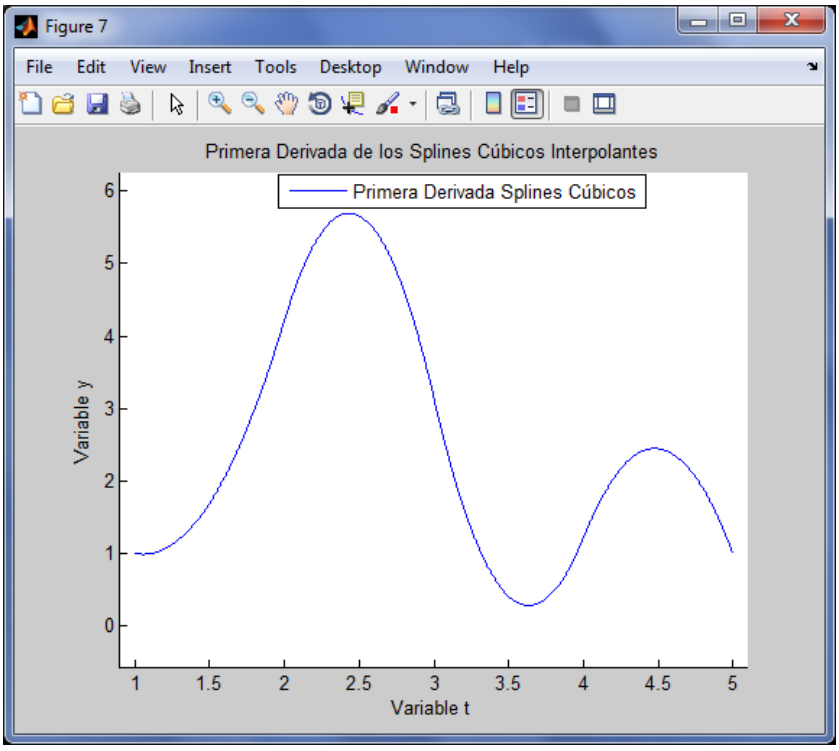

Figura 5.15 Primera derivada splines cúbicos, variable 'y'.

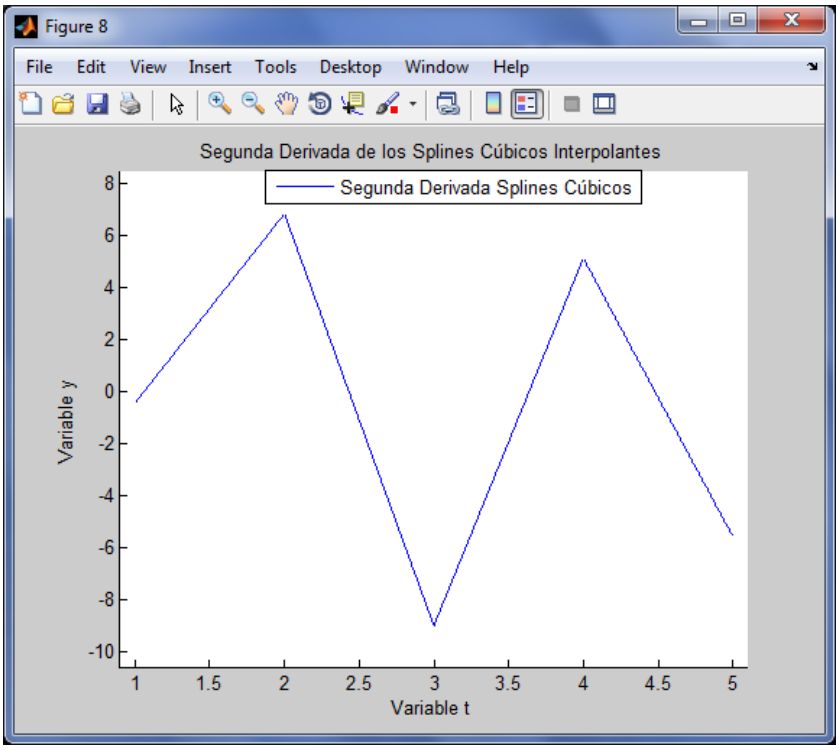

Figura 5.16 Segunda derivada splines cúbicos, variable 'y'.

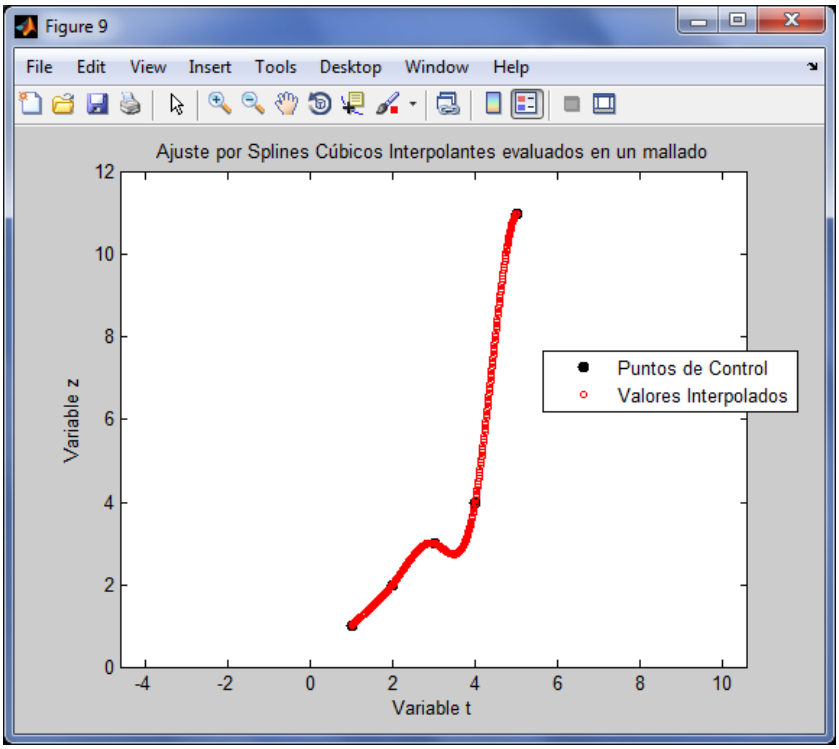

Figura 5.17 Valores interpolados, variable 'z'.

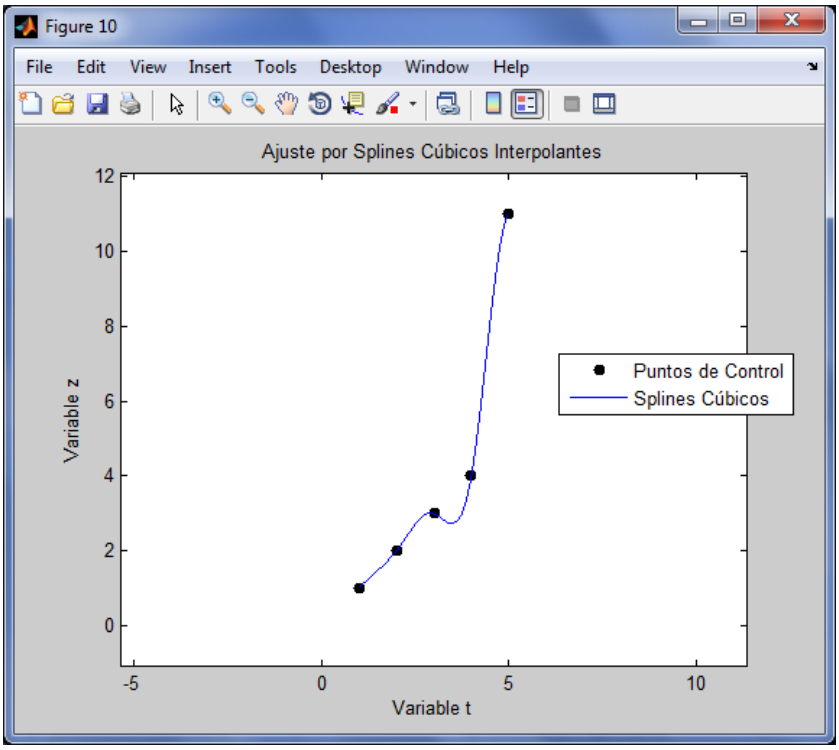

Figura 5.18 Splines cúbicos, variable 'z'.

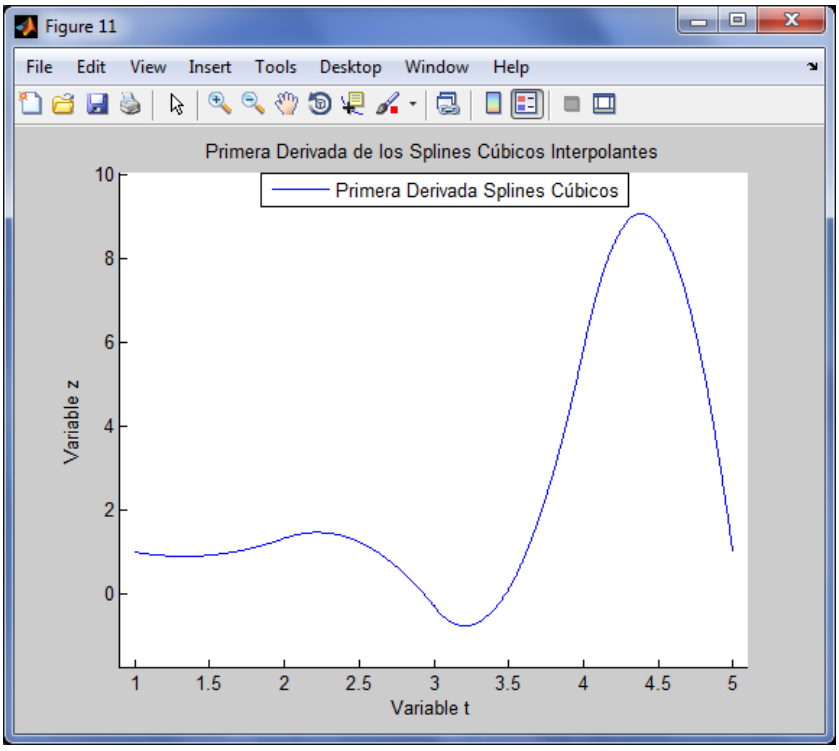

Figura 5.19 Primera derivada splines cúbicos, variable 'z'.

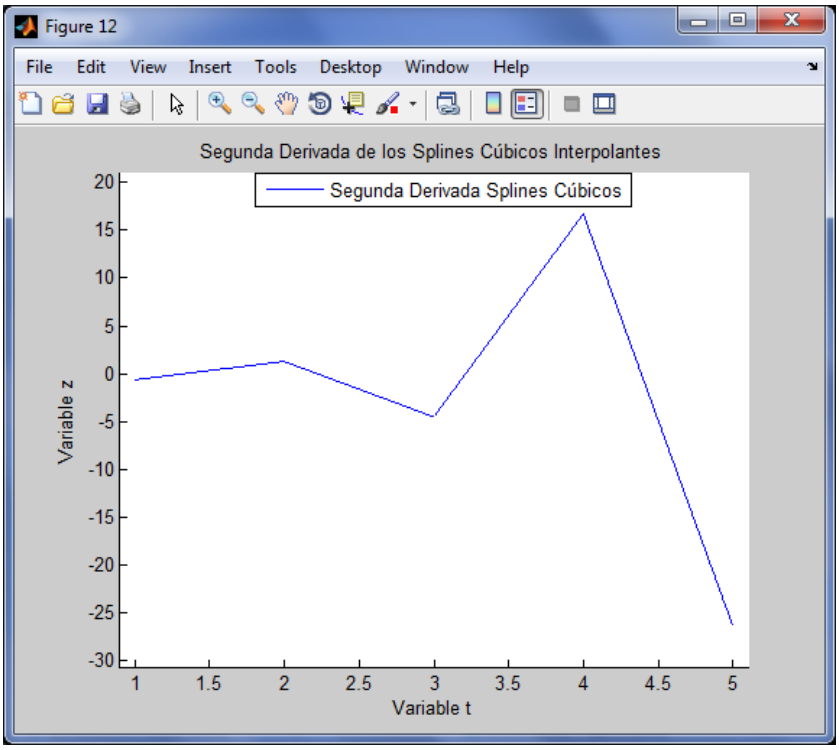

Figura 5.20 Segunda derivada splines cúbicos, variable 'z'.

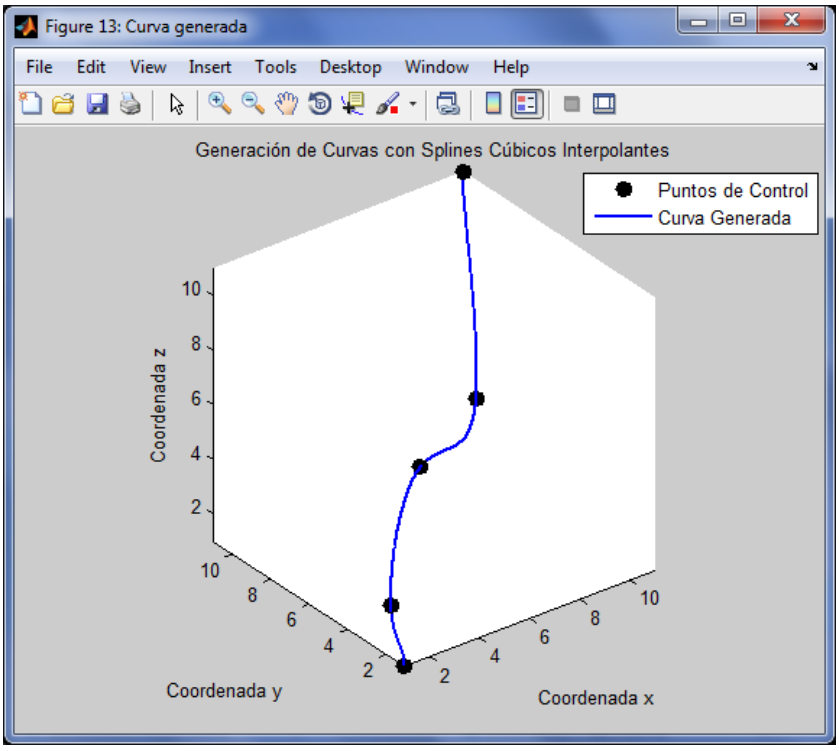

Figura 5.21 Curva generada 3D.

Si queremos las expresiones de los polinomios que forman el spline y los valores interpolados presionamos el botón “Abrir Fichero”, y nos aparecerán dos archivos en el editor de Matlab con los datos deseados.

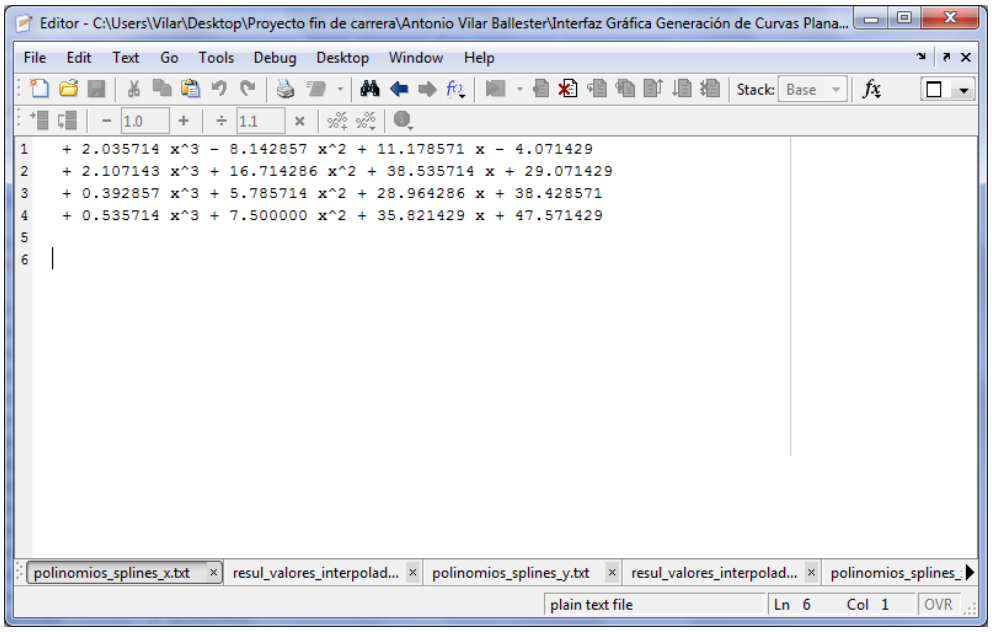

Figura 5.22 Expresiones polinomios que forman el spline.

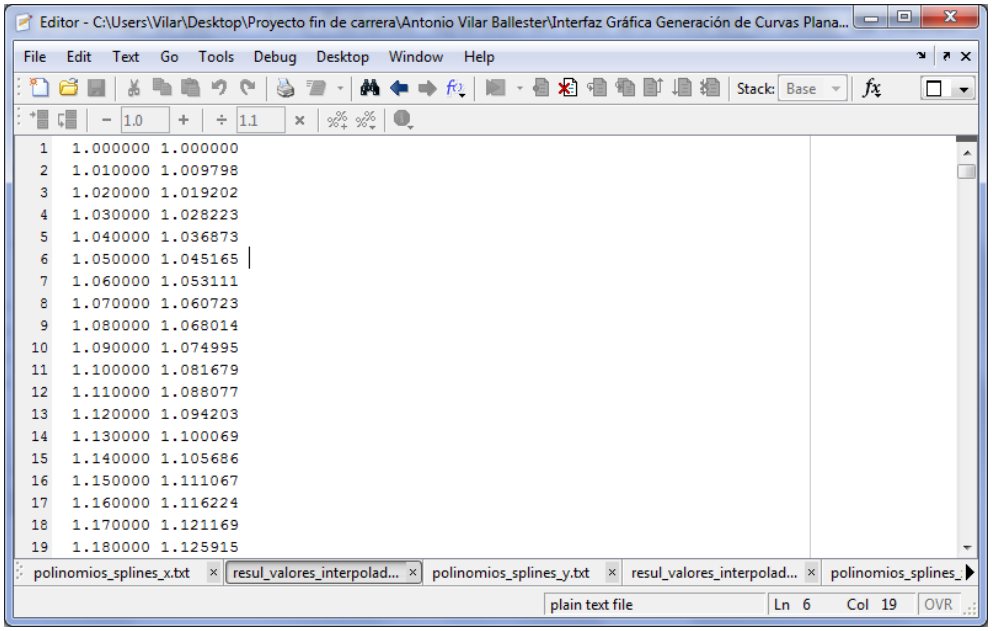

Figura 5.23 Valores interpolados.

Por último si no queremos seguir trabajando con la interfaz pulsamos el botón “Salir” y se cerrará la interfaz.

6. Casos prácticos

A continuación desarrollaremos dos casos prácticos en los cuales usaremos los splines cúbicos:

- Representación gráfica del casco de un buque
- Control del robot IRB120

6.1 Casco de un buque

6.1.1 Introducción

En este caso práctico utilizaremos los splines cúbicos suavizantes para obtener una curva suave, la cual constituye parte de un plano de formas que representa el casco de un buque. Para ello utilizaremos la interfaz gráfica que hemos desarrollado para obtener las gráficas de su primera y segunda derivada, las cuales nos permitirán analizar la forma y la suavidad de la curva calculada.

6.1.2 Datos de entrada

Vamos a partir de una cartilla de trazado de la cual sepamos que corresponde a un casco alisado previamente. En el siguiente caso tenemos el plano de formas de un casco ya alisado, por lo que, por ejemplo, seleccionaremos la octava sección para reconstruir la gráfica de la sección usando la aplicación desarrollada en Matlab. Utilizando el software de diseño Rhinoceros, conseguimos los valores de los puntos disponibles de la cuaderna. En la Figura 6.1 se ve representada de color azul la cuaderna seleccionada.

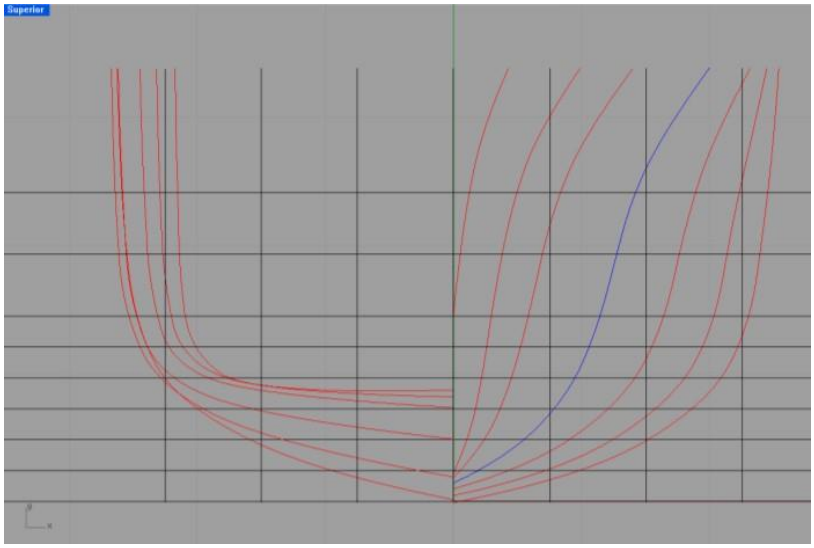

Figura 6.1 Caja de cuadernas.

A continuación en la Figura 6.2 obtenemos los puntos de intersección de la curva y sus correspondientes coordenadas, a través de sus propiedades.

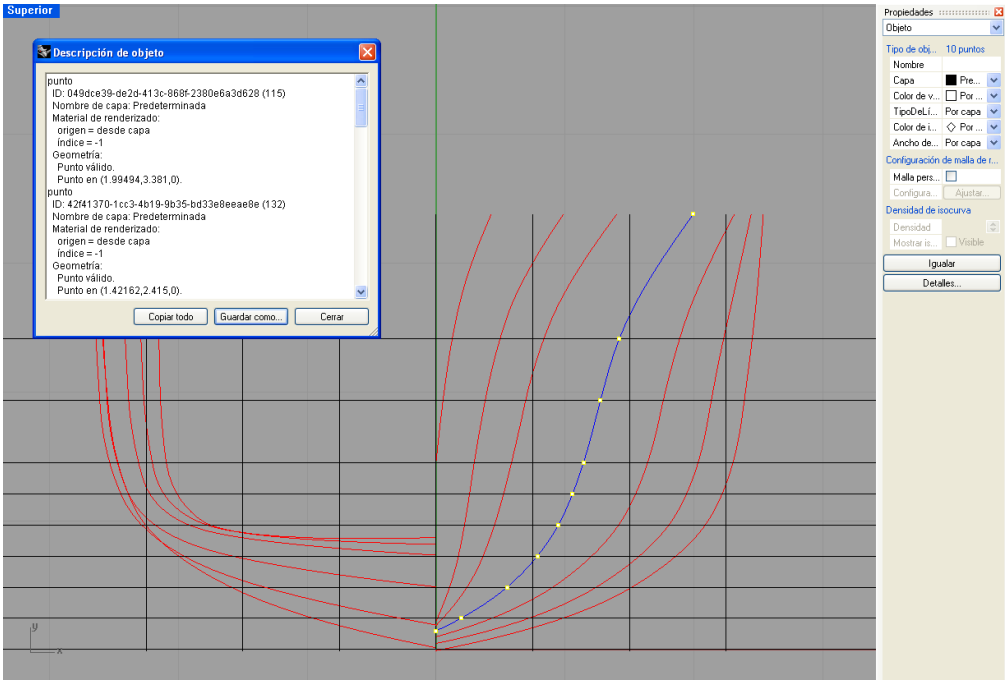

Figura 6.2 Propiedades de los puntos.

Escogemos diez puntos para definir la cuaderna escogida y extraemos los datos de las coordenadas de forma que nos sean útiles a la hora de indicarlos en la interfaz, que son:

- Eje de abscisas: 0, 0.1919, 0.5497, 0.7847, 0.9479, 1.0584, 1.1433, 1.2743, 1.4216 y 1.9949.
- Eje de ordenadas: 0.1451, 0.2415, 0.483, 0.7245, 0.966, 1.2075, 1.499, 1.932, 2.415 y 3.381

6.1.3 Resolución del problema

Utilizamos la interfaz gráfica desarrollada, introduciendo los datos obtenidos en el punto anterior y obtenemos las gráficas buscadas.

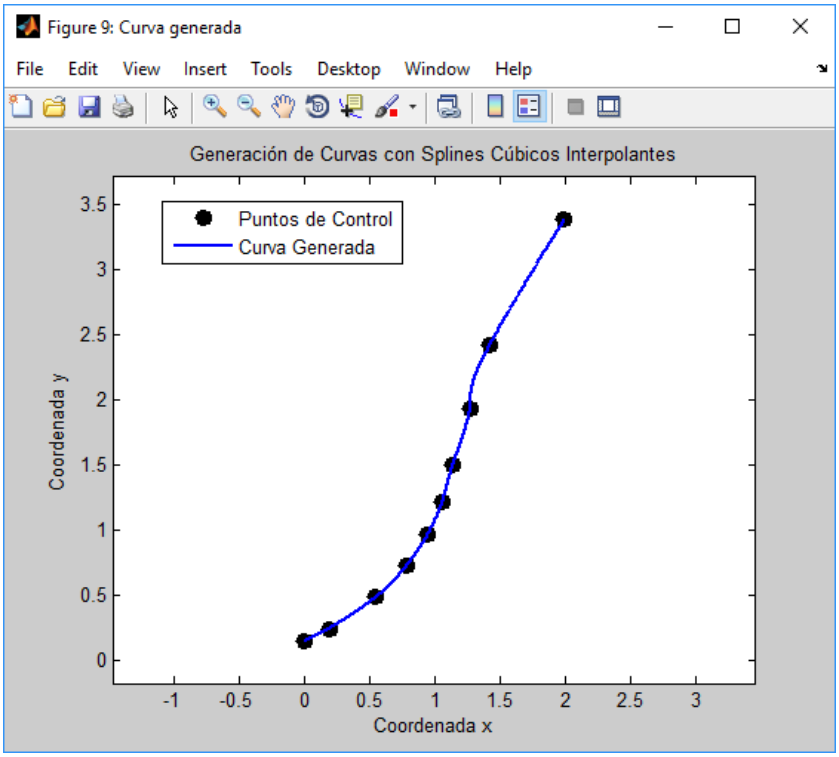

Figura 6.3 Resolución problema.

6.2 Brazos robóticos

6.2.1 Introducción

En este ejemplo se explica una posible aplicación de los programas desarrollados con Matlab para comunicar el brazo robótico IRB120 de ABB.
Para ello hace falta un socket de comunicación que se encargará de enviar y procesar los datos. Se implantará en Matlab una serie de interfaces de comunicación con el robot y una aplicación final.
La primera, será una interfaz gráfica realizada a través de la herramienta GUIDE. La segunda interfaz será a través de la creación de clases en Matlab y para la tercera interfaz se usa la herramienta Simulink.

Y la aplicación final se basará en las interfaces creadas anteriormente.

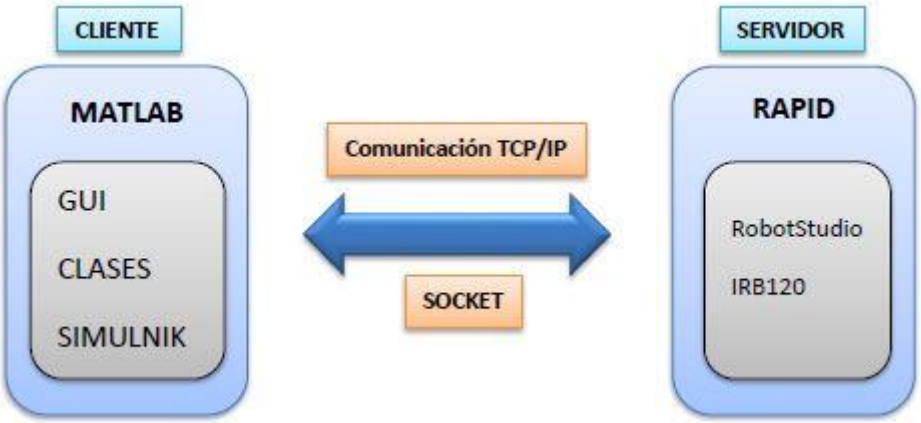

Figura 6.4 Esquema de los bloques principales.

(Fuente: bibliografía [5].

6.2.2 Brazo robótico IRB120

El brazo a utilizar es el IRB120 de ABB más pequeño y útil para muchas aplicaciones ya que pesa solamente 25 kg y puede manipular hasta 3 kg, con un área de trabajo de 580mm. Posee seis ejes, los tres primeros servirán para establecer la posición del efecto final y los tres últimos para determinar la orientación del mismo.

Figura 6.5 Robot IRB120.

(Fuente: new.abb.com/es).

Figura 6.6 Robot IRB120 en un entorno industrial

(Fuente: new.abb.com/es).

7. Conclusión

El objetivo de este Proyecto Fin de Carrera era generar curvas en el espacio con trayectorias suaves y que no contengan giros bruscos para poder aplicarlas posteriormente en el sector industrial, como en el recorrido de un brazo robótico.

Este tipo de curvas se pueden obtener a través de la definición de los splines cúbicos interpolantes o suavizantes.

A través del software Matlab hemos trabajado en el entorno Guide (GUI) desarrollando una interfaz gráfica la cual nos permite obtener las gráficas buscadas.

Aunque Matlab incluye un programa propio para la construcción de splines cúbicos, con la interfaz tenemos la posibilidad de realizar cambios y adecuarlos según nuestras necesidades.

Tras la realización del proyecto he adquirido conocimientos como el cálculo de curvas a través de los splines cúbicos interpolantes y suavizantes. Me ha permitido ampliar mis capacidades en el manejo de programación en Matlab así como en la creación de una interfaz gráfica.

8. Bibliografía

- *[1] Métodos numéricos con Matlab.*

  Cordero Barbero, Alicia; Molada Martínez, Eulalia; Hueso Pagoaga, Jose Luis; Torregrosa Sánchez, Juan Ramón.

  Editorial Universidad Politécnica de Valencia.

- *[2] Matemáticas Asistidas por Computador. [Apuntes]*

  Escudero Vergara, Antonio.

  Departamento de Matemática Aplicada y Estadística (ETSINO-Universidad Politécnica de Cartagena).

- *[3] Splines Cúbicos Interpolantes en el Diseño Naval. [3]*

  Gallego Valdellós, Irene.

  Universidad Politécnica de Cartagena. Cartagena (2012).

- *[4] Splines: Curvas y Superficies. Introducción al dibujo de curvas de aproximación e interpolación por computador.*

  González Morcillo, Carlos.

- *[5] Desarrollo de una interfaz para el control del robot IRB120 desde Matlab.*

  Gutiérrez Corbacho, Azahara.

  Universidad de Alcalá (2014).

- *[6] Splines Cúbicos Suavizantes en el Diseño Naval.*

  Roncer Peña, Blanca.

  Universidad Politécnica de Cartagena. Cartagena (2013).

9. Anexo

9.1 Código Matlab para la interfaz gráfica.

```
function varargout = GeneracionCurvas(varargin)
% GENERACIONCURVAS M-file for GeneracionCurvas.fig
%      GENERACIONCURVAS, by itself, creates a new GENERACIONCURVAS or
raises the existing
%      singleton*.
%
%      H = GENERACIONCURVAS returns the handle to a new
GENERACIONCURVAS or the handle to
%      the existing singleton*.
%
%      GENERACIONCURVAS('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in GENERACIONCURVAS.M with the given
input arguments.
%
%      GENERACIONCURVAS('Property','Value',...) creates a new
GENERACIONCURVAS or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before GeneracionCurvas_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to GeneracionCurvas_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GeneracionCurvas

% Last Modified by GUIDE v2.5 16-Jul-2015 10:30:54

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @GeneracionCurvas_OpeningFcn, ...
    'gui_OutputFcn',  @GeneracionCurvas_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT


% --- Executes just before GeneracionCurvas is made visible.
function GeneracionCurvas_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GeneracionCurvas (see VARARGIN)

% Choose default command line output for GeneracionCurvas
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GeneracionCurvas wait for user response (see UIRESUME)
% uiwait(handles.figure1);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%% CENTRAR INTERFAZ GRÁFICA EN PANTALLA %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

set( handles.output,'Units', 'pixels' );
screenSize = get(0, 'ScreenSize');

position = get( handles.output,'Position');
position(1) = (screenSize(3)-position(3))/2;
position(2) = (screenSize(4)-position(4))/2;
set( handles.output,'Position', position );



% --- Outputs from this function are returned to the command line.
function varargout = GeneracionCurvas_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%% INCLUIMOS EL ESCUDO DE LA UPCT %%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Importamos imagen *.jpg
escudo_upct=imread('escudo_upct.jpg');
axes(handles.foto);
imagesc(escudo_upct); axis off
 % --- Executes on selection change in metodo.
function metodo_Callback(hObject, eventdata, handles)
% hObject    handle to metodo (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns metodo
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
metodo



switch get(handles.metodo,'Value')
    case 1 % splines cúbicos interpolantes

        set(handles.text_pesos,'Visible','off');
        set(handles.edit_pesos,'Visible','off');
        set(handles.pushbutton_crear_pesos,'Visible','off');
        set(handles.pushbutton_cargar_pesos,'Visible','off');
        set(handles.radiobuttonIniSp,'Visible','on');
        set(handles.radiobuttonFSp,'Visible','on');

    case 2 % splines cúbicos suavizantes

        set(handles.text_pesos,'Visible','on');
        set(handles.edit_pesos,'Visible','on');
        set(handles.pushbutton_crear_pesos,'Visible','on');
        set(handles.pushbutton_cargar_pesos,'Visible','on');
        set(handles.radiobuttonIniSp,'Visible','off');
        set(handles.radiobuttonFSp,'Visible','off');

end



% --- Executes during object creation, after setting all properties.
function metodo_CreateFcn(hObject, eventdata, handles)
% hObject    handle to metodo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in pushbutton_aplicar.
function pushbutton_aplicar_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_aplicar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% LEEMOS LOS DATOS INICIALES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Recogemos los nodos y llamamos al correspondiente programa
nodos= get(handles.edit_puntos,'String');
nodos=fliplr(deblank(fliplr(deblank(nodos)))); %Eliminamos huecos en
blanco

% Comprobamos que no está vacío
if isempty(nodos)==1
    uiwait(msgbox('Debe introducir los puntos de control iniciales',
'Mensaje de error',...
        'error','modal'))
    return;
end

% Eliminamos la extensión del archivo para poder llamar a la función
nodos= strtok(nodos,'.');

% Ejecutamos la función
met=str2func(nodos);
[x,y,z]=met();


% Definimos los puntos donde interpolar-aproximar

n=length(x);

t=1:n;

h=get(handles.edit_h,'String');
h=sscanf(h, '%f');

% Comprobamos que no está vacío
if isempty(h)==1
    uiwait(msgbox('Debe introducir un valor para el espaciado h.',
'Mensaje de error',...
        'error','modal'))
    return;
end


th=1:h:n;


% Si trabajamos con splines cúbicos suavizantes leemos los pesos
asociados
% a cada punto de control

if get(handles.metodo,'Value')==2

    pesos= get(handles.edit_pesos,'String');
    pesos=fliplr(deblank(fliplr(deblank(pesos)))); %Eliminamos huecos
en blanco

    % Comprobamos que no está vacío
    if isempty(pesos)==1
        uiwait(msgbox('Debe introducir los pesos asociados a los
puntos de control iniciales', 'Mensaje de error',...
            'error','modal'))
        return;
```

```
    end

    % Eliminamos la extensión del archivo para poder llamar a la
función
    pesos= strtok(pesos,'.');

    % Ejecutamos la función
    met=str2func(pesos);
    p=met();


    % Comprobamos que los valores de los pesos son positivos y el
número de
    % pesos coincide con el número de nodos

    np=length(p);

    if all(p>=0)==0 | n~=np
        uiwait(msgbox('Debe introducir un valor de peso positivo para
cada nodo.', 'Mensaje de error',...
            'error','modal'))
        return;
    end



end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONDICIONES DE CONTORNO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Inicializamos algunas variables por defecto
S1a=0; S2a=0; S1b=0; S2b=0;
CI=0; CD=0;

% Frontera Izquierda
if get(handles.radiobuttonIni1D,'Value')==1
    CI=1;
    % Obtenemos la primera derivada al inicio
    aux = get(handles.editIni1D,'String');
    S1a=sscanf(aux, '%f');

    % Comprobamos que no está vacío
    if isempty(S1a)==1
        uiwait(msgbox('Debe introducir un valor para la primera
derivada en la frontera inicial.', 'Mensaje de error',...
            'error','modal'))
        return;
    end

elseif get(handles.radiobuttonIni2D,'Value')==1
    CI=2;
    % Obtenemos la primera derivada al inicio
    aux = get(handles.editIni2D,'String');
    S2a=sscanf(aux, '%f');

    % Comprobamos que no está vacío
```

```
    if isempty(S2a)==1
        uiwait(msgbox('Debe introducir un valor para la segunda
derivada en la frontera inicial.', 'Mensaje de error',...
            'error','modal'))
        return;
    end
elseif get(handles.radiobuttonIniP,'Value')==1
    CI=3;
    CD=3;
elseif get(handles.radiobuttonIniSp,'Value')==1

    CI=4;
end

% Frontera derecha
if get(handles.radiobuttonF1D,'Value')==1
    CD=1;
    % Obtenemos la primera derivada al final
    aux = get(handles.editF1D,'String');
    S1b=sscanf(aux, '%f');

    % Comprobamos que no está vacío
    if isempty(S1b)==1
        uiwait(msgbox('Debe introducir un valor para la primera
derivada en la frontera final.', 'Mensaje de error',...
            'error','modal'))
        return;
    end

elseif get(handles.radiobuttonF2D,'Value')==1
    CD=2;
    % Obtenemos la primera derivada a la derecha
    aux = get(handles.editF2D,'String');
    S2b=sscanf(aux, '%f');

    % Comprobamos que no está vacío
    if isempty(S2b)==1
        uiwait(msgbox('Debe introducir un valor para la segunda
derivada en la frontera final.', 'Mensaje de error',...
            'error','modal'))
        return;
    end
elseif get(handles.radiobuttonFSp,'Value')==1

    CD=4;
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% LLAMADA A LA FUNCIÓN QUE CALCULA LOS SPLINES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Número de puntos en la tabla
m=length(t);

% Comprobamos que las entradas son correctas

if m<3
    uiwait(msgbox('El número de puntos de control debe ser mayor que
```

```
2', 'Mensaje de error',...
        'error','modal'));
    return;
elseif CI==0 | CD==0
    uiwait(msgbox('Falta especificar alguna de las condiciones de
frontera', 'Mensaje de error',...
        'error','modal'));
    return;
elseif CI==3 & CD~=3
    uiwait(msgbox('Si CI=3 entonces CD debe ser también 3', 'Mensaje
de error',...
        'error','modal'));
    return;
elseif CD==3 & CI~=3
    uiwait(msgbox('Si CD=3 entonces CI debe ser también 3', 'Mensaje
de error',...
        'error','modal'));
    return;
elseif CD==3 & y(1)~=y(m)
    uiwait(msgbox('Para que la función sea periódica debe valer lo
mismo en los extremos', 'Mensaje de error',...
        'error','modal'));
    return;
elseif m==3 & CI==4 & CD==4
    uiwait(msgbox('Con 3 nodos la condición CI=4 ya implica CD=4 y nos
falta una condición', 'Mensaje de error',...
        'error','modal'));
    return;
end

if get(handles.metodo,'Value')==1   % Splines Cúbicos Interpolantes

    % LLamada para la coordenada x


[Six]=Splines_Interpolantes(t,x,CI,S1a,S2a,CD,S1b,S2b,'s',th,1,1,1,1,'
Variable x');
    !move polinomios_splines_interpolantes.txt
polinomios_splines_interpolantes_x.txt
    !move resul_valores_interpolados.txt
resul_valores_interpolados_x.txt


    % LLamada para la coordenada y


[Siy]=Splines_Interpolantes(t,y,CI,S1a,S2a,CD,S1b,S2b,'s',th,1,1,1,1,'
Variable y');
    !move polinomios_splines_interpolantes.txt
polinomios_splines_interpolantes_y.txt
    !move resul_valores_interpolados.txt
resul_valores_interpolados_y.txt


    if all(~z)   % tenemos una curva en el plano


        % Dibujamos en el plano los puntos de control y la curva
generada que
        % pasa por ellos
```

```
        figure('Name','Curva generada')


b1=plot(x,y,'ok','MarkerSize',6,'LineWidth',3,'MarkerFaceColor','k');

        hold on

        b2=plot(Six,Siy,'-b','LineWidth',2);


        Mx=max([x,Six]); mx=min([x,Six]);
        My=max([y,Siy]); my=min([y,Siy]);

        axis([mx-0.1*(Mx-mx) Mx+0.1*(Mx-mx)...
            my-0.1*(My-my) My+0.1*(My-my)]);
        axis equal;

        legend([b1,b2],'Puntos de Control','Curva Generada');
        title('Generación de Curvas con Splines Cúbicos
Interpolantes');
        xlabel('Coordenada x');
        ylabel('Coordenada y');

    else

        % LLamada para la coordenada z


[Siz]=Splines_Interpolantes(t,z,CI,S1a,S2a,CD,S1b,S2b,'s',th,1,1,1,1,'
Variable z');
        !move polinomios_splines_interpolantes.txt
polinomios_splines_interpolantes_z.txt
        !move resul_valores_interpolados.txt
resul_valores_interpolados_z.txt

        % Dibujamos en el espacio los puntos de control y la curva
generada que
        % pasa por ellos

        figure('Name','Curva generada')


b1=plot3(x,y,z,'ok','MarkerSize',6,'LineWidth',3,'MarkerFaceColor','k'
);

        hold on

        b2=plot3(Six,Siy,Siz,'-b','LineWidth',2);


        Mx=max([x,Six]); mx=min([x,Six]);
        My=max([y,Siy]); my=min([y,Siy]);
        Mz=max([z,Siz]); mz=min([z,Siz]);

        axis([mx-0.1*(Mx-mx) Mx+0.1*(Mx-mx)...
            my-0.1*(My-my) My+0.1*(My-my),mz-0.1*(Mz-mz) Mz+0.1*(Mz-
mz)]);
        axis equal;
```

```
        legend([b1,b2],'Puntos de Control','Curva Generada');
        title('Generación de Curvas con Splines Cúbicos
Interpolantes');
        xlabel('Coordenada x');
        ylabel('Coordenada y');
        zlabel('Coordenada z');



    end

elseif get(handles.metodo,'Value')==2  % Splines Cúbicos Suavizantes

    % LLamada para la coordenada x

    cd('./Splines Suavizantes');


[Six]=Splines_Suavizantes(t',x',CI,S1a,S2a,CD,S1b,S2b,'s',th,1,1,1,1,p
,'Variable x');
    !move polinomios_splines_suavizantes.txt
../polinomios_splines_suavizantes_x.txt
    !move resul_valores_aproximados.txt
../resul_valores_aproximados_x.txt


    % LLamada para la coordenada y



[Siy]=Splines_Suavizantes(t',y',CI,S1a,S2a,CD,S1b,S2b,'s',th,1,1,1,1,p
,'Variable y');



    !move polinomios_splines_suavizantes.txt
../polinomios_splines_suavizantes_y.txt
    !move resul_valores_aproximados.txt
../resul_valores_aproximados_y.txt

    cd ..


    if all(~z)   % tenemos una curva en el plano


        % Dibujamos en el plano los puntos de control y la curva
generada que
        % pasa por ellos

        figure('Name','Curva generada')


b1=plot(x,y,'ok','MarkerSize',6,'LineWidth',3,'MarkerFaceColor','k');

        hold on
```

```
        b2=plot(Six,Siy,'-b','LineWidth',2);


        Mx=max([x,Six]); mx=min([x,Six]);
        My=max([y,Siy]); my=min([y,Siy]);

        axis([mx-0.1*(Mx-mx) Mx+0.1*(Mx-mx)...
            my-0.1*(My-my) My+0.1*(My-my)]);
        axis equal;

        legend([b1,b2],'Puntos de Control','Curva Generada');
        title('Generación de Curvas con Splines Cúbicos
Interpolantes');
        xlabel('Coordenada x');
        ylabel('Coordenada y');

    else

        % LLamada para la coordenada z

         cd('./Splines Suavizantes');


[Siz]=Splines_Suavizantes(t',z',CI,S1a,S2a,CD,S1b,S2b,'s',th,1,1,1,1,p
,'Variable z');
        !move polinomios_splines_suavizantes.txt
../polinomios_splines_suavizantes_z.txt
        !move resul_valores_aproximados.txt
../resul_valores_aproximados_z.txt

        cd ..

        % Dibujamos en el espacio los puntos de control y la curva
generada que
        % pasa por ellos

        figure('Name','Curva generada')


b1=plot3(x,y,z,'ok','MarkerSize',6,'LineWidth',3,'MarkerFaceColor','k'
);

        hold on

        b2=plot3(Six,Siy,Siz,'-b','LineWidth',2);


        Mx=max([x,Six]); mx=min([x,Six]);
        My=max([y,Siy]); my=min([y,Siy]);
        Mz=max([z,Siz]); mz=min([z,Siz]);

        axis([mx-0.1*(Mx-mx) Mx+0.1*(Mx-mx)...
            my-0.1*(My-my) My+0.1*(My-my),mz-0.1*(Mz-mz) Mz+0.1*(Mz-
mz)]);
        axis equal;

        legend([b1,b2],'Puntos de Control','Curva Generada');
        title('Generación de Curvas con Splines Cúbicos
Interpolantes');
```

```
        xlabel('Coordenada x');
        ylabel('Coordenada y');
        zlabel('Coordenada z');



    end



end


% --- Executes on button press in pushbutton_fichero.
function pushbutton_fichero_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_fichero (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

edit polinomios_splines_x.txt;
edit resul_valores_interpolados_x.txt;
edit polinomios_splines_y.txt;
edit resul_valores_interpolados_y.txt;
edit polinomios_splines_z.txt;
edit resul_valores_interpolados_z.txt;

% --- Executes on button press in pushbutton_reset.
function pushbutton_reset_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Borramos lo escrito en los edit
set(handles.edit_puntos,'String','');
set(handles.edit_h,'String','0.01');
set(handles.editIni1D,'String','');
set(handles.editIni2D,'String','');
set(handles.editF1D,'String','');
set(handles.editF2D,'String','');

% Desactivamos los edit de las condiciones de contorno
set(handles.editIni1D,'Enable','off');
set(handles.editIni2D,'Enable','off');
set(handles.editF1D,'Enable','off');
set(handles.editF2D,'Enable','off');

% Desactivamos los radiobutton

set(handles.radiobuttonIni1D,'Value',0);
set(handles.radiobuttonIni2D,'Value',0);
set(handles.radiobuttonIniP,'Value',0);
set(handles.radiobuttonIniSp,'Value',0);
set(handles.radiobuttonF1D,'Value',0);
set(handles.radiobuttonF2D,'Value',0);
set(handles.radiobuttonFP,'Value',0);
set(handles.radiobuttonFSp,'Value',0);

% Escogemos el primer método de interpolación-aproximación por defecto
```

```
set(handles.metodo,'Value',1);


% Cerramos las gráficas
allPlots=findall(0,'Type','figure','FileName',[]);
delete(allPlots);


% --- Executes on button press in pushbutton_graficas.
function pushbutton_graficas_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_graficas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% Cerramos las gráficas
allPlots=findall(0,'Type','figure','Name',[]);
delete(allPlots);



% --- Executes on button press in pushbutton_salir.
function pushbutton_salir_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_salir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

salir=questdlg('¿Desea salir del programa?','Salida del
Programa','Si','No','No');
switch salir
    case 'Si'
        close all;
    case 'No'
        return;
end



function edit_puntos_Callback(hObject, eventdata, handles)
% hObject    handle to edit_puntos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_puntos as text
%        str2double(get(hObject,'String')) returns contents of
edit_puntos as a double


% --- Executes during object creation, after setting all properties.
function edit_puntos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_puntos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
```

```
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton_creardatos.
function pushbutton_creardatos_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_creardatos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

edit crear_datos2D.m

set(handles.edit_puntos,'String','crear_datos2D.m');


% --- Executes on button press in pushbutton_creardatos3D.
function pushbutton_creardatos3D_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_creardatos3D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


edit crear_datos3D.m

set(handles.edit_puntos,'String','crear_datos3D.m');


% --- Executes on button press in pushbutton_cargardatos.
function pushbutton_cargardatos_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_cargardatos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

cd('Datos Iniciales');

[filename,pathname,filterindex]=uigetfile( ...
    { '*.m', 'Archivo que contiene los datos donde interpolar
(*.m)'},'Seleccione un archivo');

if filterindex == 1
    set(handles.edit_datos,'String',filename);
end

cd ..

function edit_h_Callback(hObject, eventdata, handles)
% hObject    handle to edit_h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_h as text
%        str2double(get(hObject,'String')) returns contents of edit_h
as a double


% --- Executes during object creation, after setting all properties.
function edit_h_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_h (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton_crear_pesos.
function pushbutton_crear_pesos_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_crear_pesos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

edit crear_pesos.m

set(handles.edit_pesos,'String','crear_pesos.m');


% --- Executes on button press in pushbutton_cargar_pesos.
function pushbutton_cargar_pesos_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_cargar_pesos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

cd('Pesos');

[filename,pathname,filterindex]=uigetfile( ...
    { '*.m', 'Archivo que contiene los pesos correspondientes a cada
punto de control'},'Seleccione un archivo');

if filterindex == 1
    set(handles.edit_pesos,'String',filename);
end

cd ..



% --- Executes on button press in radiobuttonIni1D.
function radiobuttonIni1D_Callback(hObject, eventdata, handles)
% hObject    handle to radiobuttonIni1D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobuttonIni1D


if get(handles.radiobuttonIni1D,'Value')==1
    set(handles.editIni1D,'Enable','on');
    set(handles.editIni2D,'Enable','off');
    set(handles.radiobuttonIni2D,'Value',0);
    set(handles.radiobuttonIniP,'Value',0);
    set(handles.radiobuttonIniSp,'Value',0);
    set(handles.radiobuttonFP,'Value',0);
```

```
else
    set(handles.editIni1D,'Enable','off');
end

 function editIni1D_Callback(hObject, eventdata, handles)
% hObject    handle to editIni1D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editIni1D as text
%        str2double(get(hObject,'String')) returns contents of
editIni1D as a double


% --- Executes during object creation, after setting all properties.
function editIni1D_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editIni1D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in radiobuttonF1D.
function radiobuttonF1D_Callback(hObject, eventdata, handles)
% hObject    handle to radiobuttonF1D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobuttonF1D


if get(handles.radiobuttonF1D,'Value')==1
    set(handles.editF1D,'Enable','on');
    set(handles.editF2D,'Enable','off');
    set(handles.radiobuttonF2D,'Value',0);
    set(handles.radiobuttonFP,'Value',0);
    set(handles.radiobuttonFSp,'Value',0);
    set(handles.radiobuttonIniP,'Value',0);
else
    set(handles.editF1D,'Enable','off');
end



function editF1D_Callback(hObject, eventdata, handles)
% hObject    handle to editF1D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editF1D as text
%        str2double(get(hObject,'String')) returns contents of editF1D
as a double
 % --- Executes during object creation, after setting all properties.
```

```
function editF1D_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editF1D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in radiobuttonIni2D.
function radiobuttonIni2D_Callback(hObject, eventdata, handles)
% hObject    handle to radiobuttonIni2D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobuttonIni2D

if get(handles.radiobuttonIni2D,'Value')==1
    set(handles.editIni2D,'Enable','on');
    set(handles.editIni1D,'Enable','off');
    set(handles.radiobuttonIni1D,'Value',0);
    set(handles.radiobuttonIniP,'Value',0);
    set(handles.radiobuttonIniSp,'Value',0);
    set(handles.radiobuttonFP,'Value',0);
else
    set(handles.editIni2D,'Enable','off');
end




% --- Executes on button press in radiobuttonIniP.
function radiobuttonIniP_Callback(hObject, eventdata, handles)
% hObject    handle to radiobuttonIniP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobuttonIniP


if get(handles.radiobuttonIniP,'Value')==1
    set(handles.editIni1D,'Enable','off');
    set(handles.editIni2D,'Enable','off');
    set(handles.radiobuttonFP,'Value',1);
    set(handles.radiobuttonIni1D,'Value',0);
    set(handles.radiobuttonIni2D,'Value',0);
    set(handles.radiobuttonIniSp,'Value',0);
    set(handles.editF1D,'Enable','off');
    set(handles.editF2D,'Enable','off');
    set(handles.radiobuttonF1D,'Value',0);
    set(handles.radiobuttonF2D,'Value',0);
    set(handles.radiobuttonFSp,'Value',0);

else
    set(handles.radiobuttonFP,'Value',0);
```

```
end


% --- Executes on button press in radiobuttonIniSp.
function radiobuttonIniSp_Callback(hObject, eventdata, handles)
% hObject    handle to radiobuttonIniSp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobuttonIniSp


if get(handles.radiobuttonIniSp,'Value')==1
    set(handles.editIni1D,'Enable','off');
    set(handles.editIni2D,'Enable','off');
    set(handles.radiobuttonIni1D,'Value',0);
    set(handles.radiobuttonIni2D,'Value',0);
    set(handles.radiobuttonIniP,'Value',0);
    set(handles.radiobuttonFP,'Value',0);
end



% --- Executes on button press in radiobuttonF2D.
function radiobuttonF2D_Callback(hObject, eventdata, handles)
% hObject    handle to radiobuttonF2D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobuttonF2D

if get(handles.radiobuttonF2D,'Value')==1
    set(handles.editF2D,'Enable','on');
    set(handles.editF1D,'Enable','off');
    set(handles.radiobuttonF1D,'Value',0);
    set(handles.radiobuttonFP,'Value',0);
    set(handles.radiobuttonFSp,'Value',0);
    set(handles.radiobuttonIniP,'Value',0);
else
    set(handles.editF2D,'Enable','off');
end



% --- Executes on button press in radiobuttonFP.
function radiobuttonFP_Callback(hObject, eventdata, handles)
% hObject    handle to radiobuttonFP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobuttonFP



if get(handles.radiobuttonFP,'Value')==1
    set(handles.editF1D,'Enable','off');
    set(handles.editF2D,'Enable','off');
    set(handles.radiobuttonIniP,'Value',1);
```

```
    set(handles.radiobuttonF1D,'Value',0);
    set(handles.radiobuttonF2D,'Value',0);
    set(handles.radiobuttonFSp,'Value',0);
    set(handles.editIni1D,'Enable','off');
    set(handles.editIni2D,'Enable','off');
    set(handles.radiobuttonIni1D,'Value',0);
    set(handles.radiobuttonIni2D,'Value',0);
    set(handles.radiobuttonIniSp,'Value',0);
else
    set(handles.radiobuttonIniP,'Value',0);
end



% --- Executes on button press in radiobuttonFSp.
function radiobuttonFSp_Callback(hObject, eventdata, handles)
% hObject    handle to radiobuttonFSp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobuttonFSp


if get(handles.radiobuttonFSp,'Value')==1
    set(handles.editF1D,'Enable','off');
    set(handles.editF2D,'Enable','off');
    set(handles.radiobuttonF1D,'Value',0);
    set(handles.radiobuttonF2D,'Value',0);
    set(handles.radiobuttonFP,'Value',0);
    set(handles.radiobuttonIniP,'Value',0);
end



function editIni2D_Callback(hObject, eventdata, handles)
% hObject    handle to editIni2D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editIni2D as text
%        str2double(get(hObject,'String')) returns contents of
editIni2D as a double


% --- Executes during object creation, after setting all properties.
function editIni2D_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editIni2D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function editF2D_Callback(hObject, eventdata, handles)
% hObject    handle to editF2D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editF2D as text
%        str2double(get(hObject,'String')) returns contents of editF2D
as a double


% --- Executes during object creation, after setting all properties.
function editF2D_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editF2D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_pesos_Callback(hObject, eventdata, handles)
% hObject    handle to edit_pesos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_pesos as text
%        str2double(get(hObject,'String')) returns contents of
edit_pesos as a double


% --- Executes during object creation, after setting all properties.
function edit_pesos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_pesos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

9.2 Código Matlab construcción Splines cúbicos.

```
function
[Si]=Splines_Interpolantes(t,y,CI,S1a,S2a,CD,S1b,S2b,opx,x,opg1,opg2,o
pg3,opg4,mensaje)
% Esta función busca hallar las ecuaciones que definen el spline y
dibujar
% las gráficas de la curva, su primera y segunda derivada, asi como
los
% valores obtenidos al interpolar y su gráfica. Creando ficheros que
% recojan la información.
%
%
[Si]=Splines_Interpolantes(t,y,CI,S1a,S2a,CD,S1b,S2b,opx,x,opg1,opg2,o
pg3,opg4,mensaje)
%
% Variables de entrada:
% t:abscisas de la tabla de valores de los nodos
% y:ordenadas de la tabla de valores de los nodos
% CI:condición de contorno en el extremo izquierdo
% CD:condición de contorno en  el extremo derecho
%
% Posibles condiciones de contorno:
% 1- De primer tipo: valores de la primera derivada
% 2- De segundo tipo: valores de la segunda derivada
% 3- De tercer tipo: periódicas de periodo T=b-a
%    S'(a)=S'(b); S''(a)=S''(b). Si CI=3->CD=3
% 4- De cuarto tipo: Tres veces diferenciable en los puntos t2 y/o tm-
1
%
% S1a:valor de la primera derivada en el extremo izquierdo
% S2a:valor de la segunda derivada en el extremo izquierdo
% S1b:valor de la primera derivada en el extremo derecho
% S2b:valor de la segunda derivada en el extremo derecho
%
% opx:parámetro que nos indica si debemos calcular o no los valores
interpolados
% 's':se calculan; 'n': no se calculan
% x:retículo de puntos donde queremos evaluar la función polinómica a
trozos
% si opgI=1 entonces dibujará la gráfica
% opg1:parámetro que nos indica si debemos dibujar o no la gráfica de
los valores interpolados
% opg2:parámetro que nos indica si debemos dibujar o no la gráfica de
los splines
% opg3:parámetro que nos indica si debemos dibujar o no la gráfica de
las primeras derivadas
% opg4:parámetro que nos indica si debemos dibujar o no la gráfica de
las segundas derivadas
% mensaje: cadena de caracteres que se escribirá en las gráficas para
%          identificar la variable que se interpola
%
% Variables de salida:
% Si vector que contiene los valores interpolados de las ordenadas en
las
%    abscisas dadas por el vector x
%
%
%EJEMPLO:
%t=[1,3,4,6]
%y=[2,-1,3,1]
%CI=2 ; CD=1
%S1a=0; S2b=0 (los ceros no se van a utilizar, da igual el dato)
```

```
%S2a=2; S1b=2
%opx='s'
%x=[0:0.2:6]
%
%Splines_Interpolantes([1,3,4,6],[2,-
1,3,1],2,0,2,1,2,2,'s',[0:0.2:6],1,1,1,1,'Variable y')


% Número de puntos en la tabla
m=length(t);


% Definimos el vector h de distancias entre las abscisas
h=diff(t,1);

% Inicializamos
z=zeros(1,m);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Cálculo de las derivadas segundas
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Definimos la matriz de coeficientes del sistema lineal
A=zeros(m,m);
% Definimos el término independiente
B=zeros(m,1);



% Definimos la primera ecuación del sistema dependiendo de CI

if CI==1
    A(1,1)=-h(1)/3;
    A(1,2)=-h(1)/6;
    B(1)=S1a+(y(1)-y(2))/h(1);
elseif CI==2
    B(1)=S2a;
    A(1,1)=1;
elseif CI==3
    % primera ecuación
    A(1,1)=h(1)/3; A(1,2)=h(1)/6; A(1,m-1)=h(m-1)/6; A(1,m)=h(m-1)/3;
    B(1)=(y(m-1)-y(m))/h(m-1)+(y(2)-y(1))/h(1);
    % última ecuación
    A(m,1)=1; A(m,m)=-1;
elseif CI==4
    A(1,1)=-h(2); A(1,2)=h(1)+h(2); A(1,3)=-h(1);
end


% Definimos las m-2 ecuaciones intermedias


for i=2:m-1
    A(i,i-1)=h(i-1);
    A(i,i)=2*(h(i-1)+h(i));
    A(i,i+1)=h(i);
end
```

```
% A continuación definimos la matriz B

for i=2:m-1
    B(i)=6/h(i)*(y(i+1)-y(i))-6/h(i-1)*(y(i)-y(i-1));
end



% Definimos la última ecuación del sistema dependiendo de CD

if CD==1
    A(m,m-1)=h(m-1)/6; A(m,m)=h(m-1)/3;
    B(m)=S1b+(y(m-1)-y(m))/h(m-1);
elseif CD==2
    B(m)=S2b;
    A(m,m)=1;
elseif CD==4
    A(m,m-2)=-h(m-1); A(m,m-1)=h(m-1)+h(m-2); A(m,m)=-h(m-2);
end


% Resolución del sistema


if abs(det(A))<=10^(-14)
    uiwait(msgbox('El problema puede no tener solución, la matriz está
cerca de no ser invertible',...
        'Mensaje de error','error','modal'));
    return;
else
    z(1:m)=A\B;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Evaluación de los splines en x
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if opx=='s'

    % Encuentra el trozo polinómico a evaluar según cada entrada de x
    if x(1)==t(1)
        minim=y(1); maxim=y(1);
        Si(1)=y(1);
    else
        ind=find((x(1)>t)==0);
        ind=ind(1)-1;
        ind=ind(1);
        % Aplica la fórmula para calcular S_ind(x)
        Si(1)=z(ind+1)/(6*h(ind))*(x(1)-t(ind))^3+...
            z(ind)/(6*h(ind))*(t(ind+1)-x(1))^3+(y(ind+1)/h(ind)-...
            z(ind+1)*h(ind)/6)*(x(1)-t(ind))+(y(ind)/h(ind)-...
            z(ind)*h(ind)/6)*(t(ind+1)-x(1));
        minim=Si; maxim=Si;
    end

    for i=2:length(x)
        ind=find((x(i)>t)==0);
        ind=ind(1)-1;
        ind=ind(1);
```

```
        % Aplica la fórmula para calcular S_ind(x)
        Si(i)=z(ind+1)/(6*h(ind))*(x(i)-t(ind))^3+...
            z(ind)/(6*h(ind))*(t(ind+1)-x(i))^3+(y(ind+1)/h(ind)-...
            z(ind+1)*h(ind)/6)*(x(i)-t(ind))+(y(ind)/h(ind)-...
            z(ind)*h(ind)/6)*(t(ind+1)-x(i));
        if Si(i)<minim
            minim=Si(i);
        elseif Si(i)>maxim
            maxim=Si(i);
        end
    end

    if opg1==1
        figure;
        %Dibuja los puntos de control en la gráfica
        b1=plot(t,y,'ko','MarkerSize',2,'LineWidth',3);
        hold on;
        %Dibuja los valores interpolados en la gráfica
        b2=plot(x,Si,'ro','MarkerSize',3);
        hold on;
        axis([t(1)-0.1*(t(2)-t(1)) t(length(t))+...
            0.1*(t(2)-t(1)) min(min(y),minim)-...
            0.1*(max(y)-min(y)) max(max(y),maxim)+0.1*(max(y)-
min(y))]);
        axis equal;
        legend([b1,b2],'Puntos de Control','Valores Interpolados');
        title('Ajuste por Splines Cúbicos Interpolantes evaluados en
un mallado');
        xlabel('Variable t');
        ylabel(mensaje);
    end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Polinomios S_i(x)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

syms r;



for i=1:m-1
    S(i)=z(i+1)/(6*h(i))*(r-t(i))^3+z(i)/(6*h(i))*(t(i+1)-r)^3+...
        (y(i+1)/h(i)-z(i+1)*h(i)/6)*(r-t(i))+(y(i)/h(i)-...
        z(i)*h(i)/6)*(t(i+1)-r);
end



if opg2==1
    %Dibuja los puntos de control en la gráfica
    figure;
    b1=plot(t,y,'ko','MarkerSize',2,'LineWidth',3);
    hold on;

    %Dibuja cada uno de los polinomios
    Ms=0;
    ms=0;
    for i=1:m-1
```

```
        b3=ezplot(S(i),[t(i),t(i+1)]);
        auxX=t(i):(t(i+1)-t(i))/10:t(i+1);
        auxY=subs(S(i),auxX); MauxY=max(auxY); mauxY=min(auxY);
        if MauxY>Ms
            Ms=MauxY;
        end
        if mauxY<ms
            ms=mauxY;
        end
    end
    axis([t(1)-0.1*(t(2)-t(1)) t(length(t))+...
        0.1*(t(2)-t(1))  ms-0.1*(Ms-ms)  Ms+0.1*(Ms-ms)]);
    axis equal;
    legend([b1,b3],'Puntos de Control','Splines Cúbicos');
    title('Ajuste por Splines Cúbicos Interpolantes');
    xlabel('Variable t');
    ylabel(mensaje);
end


if opg3==1
    %Dibuja cada una de las primeras derivadas
    figure;
    hold on
    Ms=0;
    ms=0;
    S1=diff(S,1);
    for i=1:m-1
        b4=ezplot(S1(i),[t(i),t(i+1)]);
        auxX=t(i):(t(i+1)-t(i))/10:t(i+1);
        auxY=subs(S1(i),auxX); MauxY=max(auxY); mauxY=min(auxY);
        if MauxY>Ms
            Ms=MauxY;
        end
        if mauxY<ms
            ms=mauxY;
        end
    end
    axis([t(1)-0.1*(t(2)-t(1)) t(length(t))+...
        0.1*(t(2)-t(1))  ms-0.1*(Ms-ms)  Ms+0.1*(Ms-ms)]);
    legend(b4,'Primera Derivada Splines Cúbicos');
    title('Primera Derivada de los Splines Cúbicos Interpolantes');
    xlabel('Variable t');
    ylabel(mensaje);
end


if opg4==1
    %Dibuja cada una de las segundas derivadas
    figure;
    hold on
    S2=diff(S,2);
    Msd=subs(S2(1),t(1));
    msd=subs(S2(1),t(1));
    for i=1:m-1
        b5=ezplot(S2(i),[t(i),t(i+1)]);
        aux=subs(S2(i),t(i+1));
        if aux>Msd
            Msd=aux;
        elseif aux<msd
            msd=aux;
        end
```

```
    end

    if Msd>msd
       axis([t(1)-0.1*(t(2)-t(1)) t(length(t))+...
        0.1*(t(2)-t(1)) msd-0.1*(Msd-msd)  Msd+0.1*(Msd-msd)]);
    end
    legend(b5,'Segunda Derivada Splines Cúbicos');
    title('Segunda Derivada de los Splines Cúbicos Interpolantes');
    xlabel('Variable t');
    ylabel(mensaje);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  SALIDA DE RESULTADOS A UN FICHERO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%  Se abre o crea un archivo y se escribe en él para los resultados de
valores interpolados

if opx=='s'
    fid=fopen('resul_valores_interpolados.txt','w');
    for i=1:length(x)
        fprintf(fid,'%f %f \n',x(i),Si(i));
    end
    fclose(fid);
end

%  Se abre o crea un archivo y se escribe en él para los resultados de
los polinomios del Spline

fid=fopen('polinomios_splines_interpolantes.txt','w');
signos='----';
for i=1:length(t)-1
    coefi=sym2poly(expand(S(i)));
    traslacion=4-length(coefi);
    if traslacion>0
        coefi=[zeros(1,traslacion),coefi];
    end
    for j=1:4
        if coefi(j)>=0
            signos(j)='+';
        end
    end
    fprintf(fid,' %c %f x^3 %c %f x^2 %c %f x %c %f  \n',...
        signos(1),abs(coefi(1)),signos(2),abs(coefi(2)),signos(3),...
        abs(coefi(3)),signos(4),abs(coefi(4)));
end
fprintf(fid,'\n');
fclose(fid);
```