Tema 3

Interpolación Polinomial

RESUMEN TEÓRICO

El problema general de la interpolación es el siguiente: dados $n+1$ puntos distintos $a \leq x_1 < x_2 < \cdots < x_{n+1} \leq b$ de un intervalo $[a,b]$, llamados nodos de la interpolación, y $n+1$ números reales $y_1, y_2, \cdots, y_{n+1}$, llamados valores de la interpolación, se trata de encontrar una función f, en una cierta clase prefijada de funciones $\mathcal{F}$, tal que $f(x_i) = y_i$ para $i = 1, 2, \cdots, n+1$.

El caso particular más conocido es el problema de la interpolación polinómica, en el que $\mathcal{F}$ es el conjunto de los polinomios de grado menor o igual que n.

3.1. La fórmula de interpolación de Lagrange.

Teorema *Dados $n+1$ puntos (x_i, y_i) $(i = 1, 2, \cdots, n+1)$ de un problema de interpolación, existe un único polinomio p de grado menor o igual que n tal que $p(x_i) = y_i$ para todo $i = 1, 2, \cdots, n+1$. Dicho polinomio p viene dado por*

$$p(x) = \sum_{i=1}^{n+1} y_i L_i(x), \qquad \textit{(fórmula de interpolación de Lagrange)}$$

donde $L_i(x)$ viene dado por

$$L_i(x) = \prod_{j=1, j\neq i}^{n+1} \frac{x - x_j}{x_i - x_j} = \frac{(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_{n+1})}{(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_{n+1})}.$$

Teorema (Error en la interpolación polinomial) *Sea f una función de clase $\mathcal{C}^{n+1}[a,b]$. Consideremos el problema de interpolación correspondiente a unos nodos $a \leq x_1 < x_2 < \cdots < x_{n+1} \leq b$ con valores $y_1 = f(x_1)$, $y_2 = f(x_2)$,$\cdots$,$y_{n+1} = f(x_{n+1})$ y sea p el correspondiente polinomio interpolador. Entonces para cada $x \in [a,b]$ existe un punto $\alpha \in (a,b)$ (que depende de x) que satisface*

$$f(x) = p(x) + \frac{\omega(x) f^{n+1)}(\alpha)}{(n+1)!},$$

siendo $\omega(x) = (x-x_1)(x-x_2)\cdots(x-x_{n+1})$.

La forma práctica de usar esta fórmula es acotar:

$$|\text{Error}(x)| = |f(x) - p(x)| \leq \frac{|\omega(x)| M_{n+1}}{(n+1)!},$$

donde M_{n+1} es una cota de $|f^{n+1)}|$ en $[a, b]$.

Hay que hacer una advertencia con respecto a esta fórmula: en el exterior del intervalo $[a, b]$ el valor de $|\omega(x)|$ crece muy rápidamente. En consecuencia el uso de interpolación para aproximar $f(x)$ fuera de ese intervalo (extrapolación) debe evitarse.

En el caso particular de que los nodos esten equiespaciados $x_i = x_1 + (i-1)h$ para $i = 1, \cdots, n+1$, y el polinomio $p(x)$ se utiliza sólo para interpolar en el intervalo $[x_1, x_{n+1}]$ se tiene:

$$|\text{Error}(x)| = |f(x) - p(x)| \leq \frac{M_{n+1}}{(n+1)!} h^{n+1} = C h^{n+1},$$

donde M_{n+1} es una cota de $|f^{n+1)}|$ en $[x_1, x_{n+1}]$.

De la expresión anterior se deduce que cuando h tiende a cero, el error tiende también a cero a la misma velocidad que h^{n+1}, lo que se expresa como $|\text{Error}(x)| = \mathcal{O}\left(h^{n+1}\right)$.

Comandos de Matlab:

- `polyfit(x,y,n)`

  nos da los coeficientes, en sentido descendente, del polinomio que interpola los nodos (x, y) donde $x = [x_1, x_2, \cdots, x_{n+1}]$ y $y = [y_1, y_2, \cdots, y_{n+1}]$.

- `polyval(p,x)`

  evalua el polinomio cuyos coeficientes vienen dados en sentido descendente por el vector $p = [a_n, a_{n-1}, \cdots, a_0]$ en los valores dados por el vector x.

- `conv(p,q)`

  calcula los coeficientes del producto de los polinomios cuyos coeficientes vienen dados en sentido descendente por los vectores p y q.

- `deconv(p,q)`

  calcula los coeficientes del cociente de los polinomios cuyos coeficientes vienen dados en sentido descendente por los vectores p y q.

- `poly(r)`

  calcula los coeficientes del polinomio cuyos raíces vienen dadas por el vector r.

3.2. Fórmulas de interpolación de Newton

Uno de los inconvenientes de la fórmula de interpolación de Lagrange es que no hay relación entre la construcción del polinomio $p_n(x)$ (polinomio de grado menor o igual que n que pasa por los puntos (x_i, y_i) con $i = 1, \cdots, n+1$) y la del polinomio $p_{n+1}(x)$ (polinomio de grado menor o igual que $n+1$ que pasa por los puntos (x_i, y_i) con $i = 1, \cdots, n+2$). Cada polinomio debe construirse individualmente y se necesitan muchas operaciones para calcular polinomios de grado elevado. Los métodos de diferencias divididas sirven para generar sucesivamente estos polinomios mediante un esquema recursivo.

La idea es expresar el polinomio $p_n(x)$ en la forma:

$$p_n(x) = a_0 + a_1(x - x_1) + a_2(x - x_1)(x - x_2) + \cdots + a_n(x - x_1)(x - x_2)\cdots(x - x_n)$$

para unas constantes apropiadas $a_0, a_1, \cdots, a_n$. Estas constantes se obtienen calculando las llamadas diferencias divididas.

3.2.1. Algoritmo de las diferencias divididas

El algoritmo para generar el polinomio de interpolación $p_n(x)$, para los datos $\{(x_i, y_i) : i = 1, 2, \cdots, n+1\}$, viene dado por

$$\begin{aligned} p_n(x) &= f[x_1] + f[x_1, x_2](x - x_1) + f[x_1, x_2, x_3](x - x_1)(x - x_2) + \cdots + \\ &\quad + f[x_1, x_2, \cdots, x_{n+1}](x - x_1)(x - x_2)\cdots(x - x_n) \end{aligned}$$

donde las diferencias divididas $f[x_1, x_2, \cdots, x_k]$ para $k = 1, 2, \cdots, n+1$, están definidas según el siguiente esquema recursivo:

$$(Ddiv) \quad \begin{cases} f[x_i] &:= y_i, \ i = 1, 2, \cdots, n+1 \\ f[x_i, x_{i+1}, \cdots, x_{i+k-1}, x_{i+k}] &:= \dfrac{f[x_{i+1}, \cdots, x_{i+k-1}, x_{i+k}] - f[x_i, x_{i+1}, \cdots, x_{i+k-1}]}{x_{i+k} - x_i}, \end{cases}$$

Para calcular las diferencias divididas necesarias para obtener el polinomio interpolador p, construimos la siguiente tabla:

x	$f(x)$	primeras diferencias divididas	segundas diferencias divididas	terceras diferencias divididas
x_1	$\boxed{f[x_1] := y_1}$			
x_2	$f[x_2] := y_2$	$\boxed{f[x_1, x_2] := \frac{f[x_2] - f[x_1]}{x_2 - x_1}}$		
x_3	$f[x_3] := y_3$	$f[x_2, x_3] := \frac{f[x_3] - f[x_2]}{x_3 - x_2}$	$\boxed{f[x_1, x_2, x_3] := \frac{f[x_2, x_3] - f[x_1, x_2]}{x_3 - x_1}}$	
x_4	$f[x_4] := y_4$	$f[x_3, x_4] := \frac{f[x_4] - f[x_3]}{x_4 - x_3}$	$f[x_2, x_3, x_4] := \frac{f[x_3, x_4] - f[x_2, x_3]}{x_4 - x_2}$	$\boxed{f[x_1, x_2, x_3, x_4] := \frac{f[x_2, x_3, x_4] - f[x_1, x_2, x_3]}{x_4 - x_1}}$

3.3. Nodos de Chebyshev

Como vimos anteriormente, el error depende de la función f, pero también de cómo se escojan los nodos. Para cada valor de n, ¿cuál será la mejor forma de elegir los nodos?. Aquella para la cual el valor máximo de $|\omega(x)|$ sea lo más pequeño posible:

$$\text{mín}\left\{\text{máx}\left\{|(x-x_1)(x-x_2)\cdots(x-x_{n+1})| : a \le x \le b\right\} : a \le x_1 < x_2 < \cdots < x_{n+1} \le b\right\}.$$

Para resolver esta cuestión podemos restringirnos, haciendo una traslación y un cambio de escala si es necesario, al intervalo $[-1,1]$. La solución viene dada mediante los polinomios de Chebyshev $\{T_{n+1}(x)\}$, donde $T_{n+1}(x) = \cos((n+1)\arccos(x))$. Recordemos que la función $T_{n+1}(x)$ es la solución polinómica de la ecuación diferencial $(1-x^2)y'' - xy' + (n+1)^2 y = 0$ cuyo coeficiente lider es 2^n. La forma más fácil de construirlos es mediante la relación de recurrencia:

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x), \quad (n = 1, 2, \cdots). \end{aligned}$$

Los polinomios de Chebyshev tienen muchas propiedades interesantes. La que a nosotros nos concierne es que el valor absoluto máximo de $2^{-n}T_{n+1}(x)$ para $x \in [-1,1]$ es 2^{-n} y que, además, es el polinomio con menor máximo absoluto de entre todos los polinomios con coeficiente lider igual a 1; en otras palabras, que $2^{-n}T_{n+1}(x)$ es la solución al problema anterior o, equivalentemente, que los mejores nodos posibles son las $n+1$ raíces simples de T_{n+1}:

$$x_i = \cos\left(\frac{(2i-1)\pi}{2(n+1)}\right), i = 1, 2, \cdots, n+1,$$

que se llaman nodos de Chebyshev del intervalo $[-1,1]$. Como ya hemos indicado, para trabajar en un intervalo cualquiera hay que hacer un cambio de variable lineal que nos lleve el intervalo $[-1,1]$ a nuestro intervalo de trabajo $[a,b]$:

$$x \in [-\pi, \pi] \to z(x) := \frac{b-a}{2}x + \frac{b+a}{2} \in [a,b],$$

entonces los nodos de Chebyshev en $[a,b]$ son los puntos $z(x_i)$ $(i = 1, 2, \cdots, n+1)$.

Observación: Interpolación de Hermite

Para intentar mejorar el fenómeno de Runge, vease problema 3-1, o eliminarlo se pensó en la posibilidad de que el polinomio interpolador no sólo coincidiese con la función en los valores que tomaba en los nodos, sino que también lo hagan los de su derivada primera en los nodos.

Al polinomio que interpola de esta forma se le denomina de Hermite, pero sin embargo, sigue manifestándose el fenómeno de Runge, es decir, se mejora el resultado en la parte central del intervalo, pero en los extremos la diferencia entre el polinomio interpolador y la función es considerable.

La manera de evitar el fenómeno de Runge es hacer una interpolación polinomial a trozos, es decir, lo que se conoce como una **interpolación por splines**.

3.4. Determinación de las funciones splines interpolantes. Propiedades de convergencia

Consideremos $\{x_1 = a < x_2 < ... < x_n < x_{n+1} = b\}$ un conjunto de nodos. La interpolación por splines no es más que tomar un conjunto de nodos en cada subintervalo de la forma $[x_i, x_{i+1}]$ y construir un polinomio de interpolación, de grado no superior a k (para un k prefijado) sobre dicho conjunto de nodos, por lo que el método se conoce también como interpolación polinomial a trozos.

Definicion. [Función spline interpolante] *Una **función spline interpolante** de grado k con nodos $\{x_1, x_2, ..., x_n, x_{n+1}\}$ es una función $S(x)$ formada por varios polinomios, cada uno de ellos definido sobre un subintervalo y que se unen entre sí bajo ciertas condiciones de continuidad. Las condiciones que debe cumplir $S(x)$ son las siguientes:*

1. *En cada intervalo $[x_i, x_{i+1}]$, $S(x)$ es un polinomio de grado $gr[S(x)] \leq k$.*
2. *$S(x)$ admite derivada continua de orden $k-1$ en $[x_1, x_{n+1}]$.*

En general pueden crearse funciones spline de grado k, pero la interpolación más frecuente es a través de funciones splines de grado 3, es decir, de **splines cúbicos**.

3.4.1. Splines cúbicos

Definicion. [Spline cúbico.]
*Dado el conjunto de nodos $\Delta = \{x_1 = a, x_2, ..., x_n, x_{n+1} = b\}$ del intervalo $[a, b]$, diremos que la función S_Δ es un **spline cúbico** asociado a Δ si cumple las siguientes condiciones:*

1. *La restricción S_i de S_Δ a cada intervalo $[x_i, x_{i+1})$, para $i = 1, 2, ..., n$ es un polinomio de grado no superior a tres.*
2. *$S(x) \in \mathcal{C}^2([a, b])$, es decir, S_Δ es una función continua, dos veces derivable y con derivadas continuas en el intervalo $[a, b]$.*

Definicion. [Spline cúbico interpolante.] *Diremos que S_Δ es un spline cúbico interpolante para el conjunto de nodos Δ, si :*

1. *S_Δ es un spline cúbico asociado a Δ.*
2. *$S_\Delta(x_i) = f(x_i) = y_i$ para $i = 1, 2, ..., n+1$, es decir, cumple las condiciones de interpolación.*

Antes de construir un spline cúbico vamos a ver cuántas condiciones ha de cumplir y cuántas incógnitas van a hacernos falta. Si en cada subintervalo de Δ intentamos construir un polinimio de grado tres que aproxime a la función, deberemos calcular cuatro incógnitas (los cuatro coeficientes del polinomio de grado tres) por subintervalo, es decir $4n$ incógnitas. Por otro lado, estos polinomios deben cumplir, en cada uno de los nodos las condiciones:

$$\left.\begin{array}{rcl} S_i(x_{i+1}) & = & S_{i+1}(x_{i+1}) \\ S_i'(x_{i+1}) & = & S_{i+1}'(x_{i+1}) \\ S_i''(x_{i+1}) & = & S_{i+1}''(x_{i+1}) \end{array}\right\} \quad i = 1, ..., n-1. \tag{3.1}$$

Es decir, se deben cumplir un total de $3(n-1)$ condiciones además de las $n+1$ de interpolación:

$$S_i(x_i) = f(x_i) = y_i \quad i = 1, 2, ..., n+1.$$

Dado que tenemos un total de $4n$ incógnitas para $4n-2$ condiciones, debemos imponer dos nuevas condiciones para poder determinar los coeficientes de la función spline. Dependiendo de las condiciones que impongamos, obtendremos un tipo de spline u otro.

- Si exigimos que la derivada segunda se anule en los extremos, es decir, si

  $$S''_\Delta(a) = S''_\Delta(b) = 0,$$

  diremos que $S_\Delta(x)$ es el **spline natural** asociado al conjunto de nodos Δ.

- Si exigimos que la derivada primera tome un determinado valor en los extremos, es decir, si

  $$S'_\Delta(a) = y'_1, \quad S'_\Delta(b) = y'_{n+1},$$

  diremos que $S_\Delta(x)$ es el **spline sujeto** asociado al conjunto de nodos Δ.

- Si, suponiendo que $y_1 = y_{n+1}$, exigimos que

  $$S'_\Delta(a) = S'_\Delta(b), \quad S''_\Delta(a) = S''_\Delta(b),$$

  diremos que se trata de un **spline periódico**.

Nos centraremos en el cálculo de los splines naturales y con el fin de simplificar la notación, llamaremos :

$$h_i = x_{i+1} - x_i, \quad i = 1, ..., n$$
$$M_i = S''_i(x_i), \quad i = 1, ..., n+1.$$

Los valores M_i se denominan **momentos** y determinarán completamente los splines cúbicos.

Observemos en primer lugar, que como en cada subintervalo $[x_i, x_{i+1}]$, $i = 1, \cdots, n$, el spline S_Δ es un polinimio de grado tres, su segunda derivada es una recta (un polinomio de grado uno). En consecuencia, al imponer las condiciones (3.1) sobre la igualdad de derivadas segundas en los nodos, obligamos a que la segunda derivada de la función spline $S''_\Delta(x)$ constituya un conjunto de rectas que se cortan en los nodos del conjunto elegido. Ahora bien, dado que cada recta queda determinada por dos puntos, podemos escribir el valor de las restricciones (3.1) sobre S''_i como

$$S''_i(x) = M_i \frac{x_{i+1} - x}{h_i} + M_{i+1} \frac{x - x_i}{h_i} \quad i = 1, \cdots, n.$$

Integrando respecto a x obtenemos el valor de la primera derivada del spline en este intervalo :

$$S'_i(x) = -\frac{M_i}{2} \frac{(x_{i+1} - x)^2}{h_i} + \frac{M_{i+1}}{2} \frac{(x - x_i)^2}{h_i} + A_i.$$

Volviendo a integrar respecto a x obtenemos:

$$S_i(x) = \frac{M_i}{6} \frac{(x_{i+1} - x)^3}{h_i} + \frac{M_{i+1}}{6} \frac{(x - x_i)^3}{h_i} + A_i(x - x_i) + B_i.$$

Si imponemos ahora las condiciones de interpolación:

$$S_i(x_i) = y_i, \quad S_i(x_{i+1}) = y_{i+1},$$

obtenemos

$$\frac{M_i}{6}h_i^2 + B_i = y_i \Rightarrow B_i = y_i - \frac{M_i}{6}h_i^2,$$

$$\frac{M_{i+1}}{6}h_i^2 + A_i h_i + B_i = y_{i+1} \Rightarrow A_i = \frac{y_{i+1}-y_i}{h_i} - \frac{h_i}{6}(M_{i+1} - M_i).$$

Podemos, por tanto, determinar los valores de las constantes A_i y B_i, que determinan el valor de $S_\Delta(x)$ en el intevalo $[x_i, x_{i+1}]$, $i = 1, \cdots, n$, en función de los momentos. Siendo:

$$\begin{aligned} S_i(x) &= \frac{M_i}{6h_i}(x_{i+1}-x)^3 + \frac{M_{i+1}}{6h_i}(x-x_i)^3 \\ &\quad + \left[\frac{y_{i+1}-y_i}{h_i} - \frac{h_i}{6}(M_{i+1}-M_i)\right](x-x_i) + y_i - \frac{M_i}{6}h_i^2, \end{aligned}$$

o en términos de $x - x_i$ y denotando $e_i = \frac{y_{i+1}-y_i}{h_i}$ resulta:

$$S_i(x) = \frac{M_{i+1}-M_i}{6h_i}(x-x_i)^3 + \frac{M_i}{2}(x-x_i)^2 + \left[e_i - \frac{h_i}{6}(M_{i+1}+2M_i)\right](x-x_i) + y_i.$$

El problema se reduce, por tanto, a calcular los momentos para cada uno de los intervalos, para lo que utilizaremos la única condición de (3.1) que no hemos utilizado:

$$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}).$$

Esta condición nos da, para cada $i = 1, ..., n-1$ una ecuación:

$$h_i M_i + 2(h_i + h_{i+1})M_{i+1} + h_{i+1}M_{i+2} = u_i,$$

donde hemos denotado $u_i = 6(e_{i+1} - e_i)$, para $i = 1, \cdots, n-1$. Si conocemos M_1 y M_{n+1} entonces obtenemos el sistema tridiagonal de $n-1$ ecuaciones con $n-1$ incógnitas:

$$\begin{pmatrix} 2(h_1+h_2) & h_2 & & & \\ h_2 & 2(h_2+h_3) & h_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & h_{n-1} \\ & & & h_{n-1} & 2(h_{n-1}+h_n) \end{pmatrix} \begin{pmatrix} M_2 \\ M_3 \\ \vdots \\ \vdots \\ M_n \end{pmatrix} = \begin{pmatrix} u_1 - h_1 M_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{n-2} \\ u_{n-1} - h_n M_{n+1} \end{pmatrix}.$$

En el caso del spline natural tenemos que $M_1 = S''(a) = 0$ y $M_{n+1} = S''(b) = 0$.

En el caso del spline sujeto $S_1'(x_1) = y_1'$ y $S_n'(x_{n+1}) = y_{n+1}'$, con y_1' y y_{n+1}' conocidos, se obtienen dos nuevas ecuaciones:

$$2h_1M_1 + h_1M_2 = 6(e_1 - y_1'), \quad h_nM_n + 2h_nM_{n+1} = 6(y_{n+1}' - e_n),$$

dando lugar a un sistema tridiagonal de $n+1$ ecuaciones con $n+1$ incógnitas:

$$\begin{pmatrix} 2h_1 & h_1 & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & h_{n-1} & 2(h_{n-1}+h_n) & h_n & \\ & & & h_n & 2h_n \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \\ \vdots \\ \vdots \\ M_n \\ M_{n+1} \end{pmatrix} = \begin{pmatrix} 6(e_1 - y_1') \\ u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{n-1} \\ 6(y_{n+1}' - e_n) \end{pmatrix}.$$

Ambos sistemas se pueden resolver por cualquiera de los métodos iterados estudiados ya que al ser la matriz del sistema diagonal dominante, tenemos la seguridad de que convergen y tienen una única solución.

El siguiente programa halla los coeficientes en potencias de $x - x_i$ para el caso spline natural.

```
function S=splinenatural(X,Y)
% Este programa calcula los coeficientes del trazador cubico que interpola los
% nodos (X,Y) y tiene derivadas segundas en los extremos nulas, es decir
% el spline natural.
%                            Datos de entrada
%      -X es un vector 1x(n+1) que contiene las abscisas
%      -Y es un vector 1x(n+1) que contine las ordenadas
%                            Datos de salida
%      - S: las filas de S son los coeficientes de las potencias de
%           (x-x(i)), en orden descendente, de cada uno de los polinomios
N=length(X)-1;
H=diff(X);
E=diff(Y)./H;
diagsupinf=H(2:N-1);
diagprinc=2*(H(1:N-1)+H(2:N));
U=6*diff(E);
% restricciones del spline natural
Minicial=0; Mfinal=0;
% construccion de la matriz de los coeficientes para el calculo de los
% momentos
A=diag(diagprinc)+diag(diagsupinf,1)+diag(diagsupinf,-1);
% construccion del vector independiente para el calculo de los momentos
B=U'; B(1)=B(1)-H(1)*Minicial; B(N-1)=B(N-1)-H(N)*Mfinal;
% resolvemos el sistema y hallamos los momentos
```

```
M=A\B;
% añadimos los dos momentos naturales extremos y lo escribimos como vector
% fila
M=[Minicial,M',Mfinal];
% calculo de los coeficientes del polinomio cubico i-esimo en potencias
% de (x-x_i)
for i=1:N
    S(i,1)=(M(i+1)-M(i))/(6*H(i));
    S(i,2)=M(i)/2;
    S(i,3)=E(i)-H(i)*(M(i+1)+2*M(i))/6;
    S(i,4)=Y(i);
end
```

Por último en el caso del spline periódico $S_1''(x_1) = S_n''(x_{n+1})$ y $S_1'(x_1) = S_n'(x_{n+1})$, se obtenien las ecuaciones:

$$M_{n+1} = M_1, \quad 2(h_1 + h_n)M_1 + h_1M_2 + h_nM_n = 6(e_1 - e_n).$$

El sistema de n ecuaciones con n incógnitas resultante, que en este caso no es tridiagonal ni diagonal dominante, es:

$$\begin{pmatrix} h_1 & 2(h_1+h_2) & h_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & h_{n-1} \\ h_n & 0 & & h_{n-1} & 2(h_{n-1}+h_n) \\ 2(h_1+h_n) & h_1 & & 0 & h_n \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ \vdots \\ M_{n-1} \\ M_n \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ \vdots \\ u_{n-1} \\ 6(e_1 - e_n) \end{pmatrix},$$

y un programa que calcula los coeficientes, en potencias de $x - x_i$, del spline periódico es

```
function S=splineperiod(X,Y)
% Este programa calcula los coeficientes del trazador cúbico que interpola los
% nodos (X,Y) con condiciones periodicas S'(a)=S'(b)y S''(a)=S''(b)
%                      Datos de entrada
%      -X es un vector 1x(n+1) que contiene las abscisas
%      -Y es un vector 1x(n+1) que contine las ordenadas
%                      Datos de salida
%      -S es una matriz cuyas filas son los coeficientes, en orden descendente,
%           de cada trazador cúbico
N=length(X)-1;
H=diff(X);
E=diff(Y)./H;
diagprinc=H(1:N);
diagsup1=2*(H(1:N-1)+H(2:N));
diagsup2=H(2:N-1);
```

```
U=6*diff(E);
% construccion de la matriz de los coeficientes para el calculo de los
% momentos
A=diag(diagprinc)+diag(diagsup1,1)+diag(diagsup2,2);
A(N,1)=2*(H(1)+H(N));
A(N-1,1)=H(N);
A(N,2)=H(1);
% construccion del vector independiente para el calculo de los momentos
B=[U';6*(E(1)-E(N))];
% resolvemos el sistema y hallamos los momentos M(1),...,M(N)
M=A\B;
% ya sabemos que M(N+1)=M(1) lo añadimos y lo escribimos como vector fila
M=[M',M(1)]
% calculo de los coeficientes del polinomio cubico i-esimo en potencias
% de (x-x_i)
for i=1:N
    S(i,1)=(M(i+1)-M(i))/(6*H(i));
    S(i,2)=M(i)/2;
    S(i,3)=E(i)-H(i)*(M(i+1)+2*M(i))/6;
    S(i,4)=Y(i);
end
```

Teorema. [Unicidad del spline cúbico natural] *Si f es una función definida en $[a,b]$, entonces f tiene un único spline cúbico natural interpolante , es decir, que cumple $S''(a) = S''(b) = 0$.*

Teorema. [Convergencia] *Sea $f \in \mathcal{C}^4([a,b])$ con $\max_{a\leq x\leq b} \mid f^{(4}(x) \mid= M$. Si S es el único interpolante cúbico natural de f con respecto a los nodos $a = x_1 < x_2 < ... < x_{n+1} = b$ que satisface $S''(a) = S''(b) = 0$, entonces existe una constante $A \in \mathbb{R}$ tal que*

$$\max_{a\leq x\leq b} \mid f(x) - S(x) \mid\leq AM \max_{1\leq j\leq n} (x_{j+1} - x_j)^4.$$

Comandos de Matlab:

- `spline(x,y)`

  Halla los trazadores cúbicos que interpolan los nodos (x_i, y_i) $i = 1, \cdots, n+1$ cuyas coordenadas estan dadas por los vectores x e y de dimensión $n+1$. Estos trazadores son los llamados *'not-a-knot'*, (no nodo) es decir, impone como condiciones adicionales que los trazadores cúbicos en los intervalos $[x_1, x_2]$ y $[x_2, x_3]$ sean el mismo y en los intervalos $[x_{n-1}, x_n]$ y $[x_n, x_{n+1}]$ también sean el mismo.

- `spline(x,y)`

  Para vectores $x = [x_1, x_2, \cdots, x_{n+1}]$ de dimensión $n+1$ e $y = [y'_1, y_1, \cdots, y_{n+1}, y'_{n+1}]$ de dimensión $n+3$ entonces calcula el trazador cúbico sujeto con $S'_1(x_1) = y'_1$ y $S'_n(x_{n+1}) = y'_{n+1}$ en los nodos $(x_1, y_1), \cdots, (x_{n+1}, y_{n+1})$ dados por todas las componentes del vector x y las componentes del vector y excepto la primera y la última.

- `yy = ppval(spline(x,y),xx)`

  Halla los valores que toma el trazador cúbico que interpola los nodos dados por los vectores x e y en los valores dados por el vector xx.

- `yy = spline(x,y,xx)`

  Tiene el mismo efecto que la orden `yy=ppval(spline(x,y),xx)`

- `y=ppval(mkpp(X,S),x)`

  Evalua en los valores dados por el vector x el trazador cúbico a trozos en los intervalos $[X(i), X(i+1)]$ dados en el vector X con coeficientes $S(i,:)$ obtenidos por cualquier función spline.

- `[BREAKS,COEFS,L,K,D]=UNMKPP(S)`

  Extrae información de un polinomio a trozos en concreto `BREAKS` muestra los nodos que limitan los trozos, `X(1),...,X(n+1)` y `COEFS` muestra el vector de coeficientes de cada uno de los trozos expresado en potencias de $x - X(i)$.

Referencias básicas: Burden y Faires [1985, Cap. 3]; Kincayd y Cheney [1994, 6.1-6.2 y 6.4]; Henrici [1972, Cap. 4 y 5]; Henrici [1982, Cap. 9]; Mathews y Fink [2000,Cap 4]; Nakamura [1997, Cap. 4 y 9]; Scheid y Di Contanzo [1991, Cap.9]; Stoer y Bulirsch [1980, 2.1 y 2.4].

Ejercicios Resueltos Tema 3

PROBLEMA 3-1: El fenómeno de Runge es un problema, debido a la falta de convergencia puntual de los polinomios interpoladores con respecto a la función considerada al aumentar el número de nodos, que puede darse cuando los nodos están igualmente espaciados. Para observar este fenómeno consideramos la función $f(x) = \dfrac{1}{1+12x^2}$. Sean $P_n(x)$ y $Q_n(x)$ los polinomios que interpolan a $f(x)$, en $n+1$ nodos igualmente espaciados y en los nodos de Chebyshev respectivamente, en el intervalo $[-1, 1]$.

(1) Hacer un gráfico comparativo de $f(x)$ y $P_n(x)$ para $n = 5, 10, 15, 20$. Lo mismo para $f(x)$ y $Q_n(x)$.

(2) Para cada uno de los valores $n = 2, 3, ..., 20$ estimar el error máximo

$$EP_n = max\{|f(x) - P_n(x)| : -1 \leq x \leq 1\},$$

aproximándolo mediante el cálculo de la diferencia $|f(x) - P_n(x)|$ en 201 puntos igualmente espaciados del intervalo $[-1, 1]$(máximo discreto). ¿Cómo se comporta EP_n conforme n crece?.

(3) Repetir el apartado (2) con

$$EQ_n = max\{|f(x) - Q_n(x)| : -1 \leq x \leq 1\}.$$

SOLUCIÓN PROBLEMA 3-1:

Contenido del fichero frunge.m (donde tenemos definida la función $f(x) = 1/(1+12x^2)$, que le llamaremos a partir de ahora frunge).

```
function y=frunge(x)
y=(1+12.*x.^2).^(-1);
```

Apartado (1)

El programa que sigue hace un gráfico en la esquina superior izquierda **subplot(2,2,1)**, **plot(x,y)** de la función $f(x) = 1/(1+12x^2)$ en $[-1,1]$ y de su polinomio de interpolación con 6 nodos igualmente espaciados en $[-1,1]$.

```
%numero de nodos es n+1
n=5;
%eleccion de n+1 puntos igualmente espaciados en [-1,1]
nodos=linspace(-1,1,n+1);
%valores de la funcion en los nodos
valores=frunge(nodos);
%polinomio de interpolacion
p=polyfit(nodos,valores,n);
%500 puntos del intervalo [-1,1] igualmente espaciados
t=linspace(-1,1,500);
%dibujo de la funcion, los nodos y su polinomio interpolador
subplot(2,2,1),plot(t,frunge(t),'b',nodos,valores,'r*',t,polyval(p,t),'k')
%ventana grafica, x en [-1.1,1.1] e y en [-0.2,1.2]
axis([-1.1,1.1,-0.2,1.2])
%le ponemos titulo
title('n=5')
```

El programa que sigue hace un gráfico en la esquina superior izquierda **subplot(2,2,2)**, **plot(x,y)** de la función $f(x) = 1/(1+12x^2)$ en $[-1,1]$ y de su polinomio de interpolación con 11 nodos igualmente espaciados en $[-1,1]$.

```
%numero de nodos es n+1
n=10;
%eleccion de n+1 puntos igualmente espaciados en [-1,1]
nodos=linspace(-1,1,n+1);
%valores de la funcion en los nodos
valores=frunge(nodos);
%polinomio de interpolacion
p=polyfit(nodos,valores,n);
%500 puntos del intervalo [-1,1] igualmente espaciados
t=linspace(-1,1,500);
%dibujo de la funcion, los nodos y su polinomio interpolador
subplot(2,2,2),plot(t,frunge(t),'b',nodos,valores,'r*',t,polyval(p,t),'k')
%ventana grafica, x en [-1.1,1.1] e y en [-0.2,1.2]
```

```
axis([-1.1,1.1,-0.2,1.2])
%le ponemos titulo
title('n=10')
```

De igual forma se hace para $n = 15$ (16 puntos), y para que el gráfico salga en la esquina inferior izquierda la instrucción **subplot(2,2,3), plot(x,y)**(**subplot(2,2,4), plot(x,y)**).

Los resultados aparecen representados en la figura 3.1

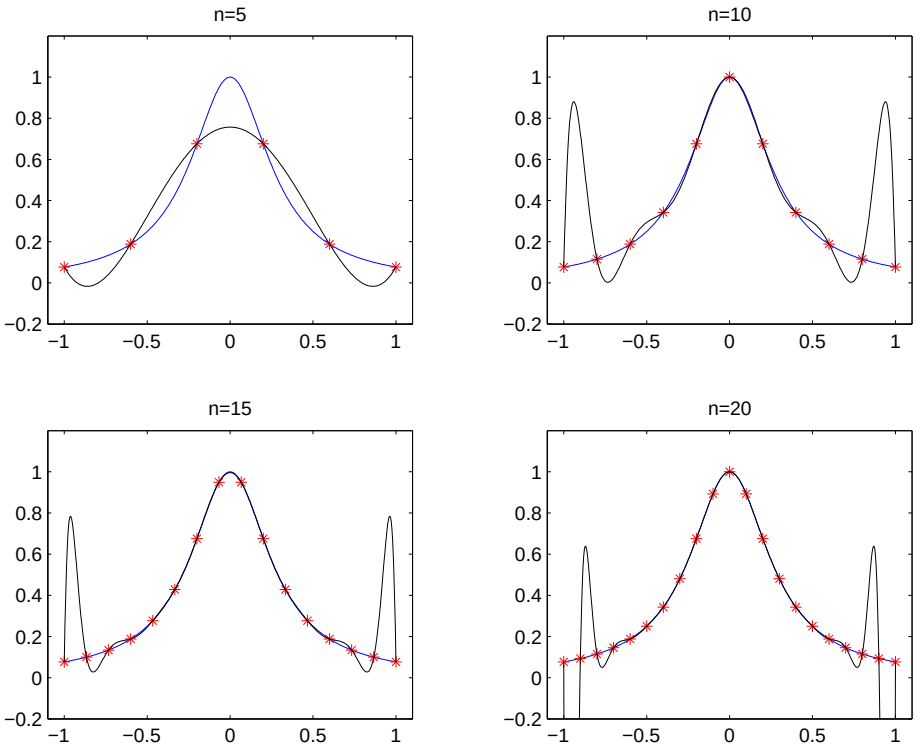

Figura 3.1: Gráfico de la función $f(x) = \frac{1}{1+12x^2}$ y de su polinomio de interpolación con $(6, 11, 16, 21)$ nodos igualmente espaciados en $[-1, 1]$.

La codificación que sigue hace un gráfico en la esquina superior izquierda de la ventana **subplot(2,2,1),plot(x,y)**, de la función $1/(1 + 12x^2)$ en $[-1, 1]$ y de su polinomio de interpolación con nodos de Chebyshev con 5 puntos. De igual forma se hace para $n = 10, 15$ y 20, dibujándolas en distintas esquinas **subplot(2,2,2)**, **plot(x,y)** para $n = 10$, **subplot(2,2,3)**, **plot(x,y)** para $n = 15$ y **subplot(2,2,4)**, **plot(x,y)** para $n = 20$.

```
%numero de nodos es n+1
n=5;
%calculo de los nodos de Chebyshev para el polinomio de grado 5 en [-1,1]
for i=1:n+1
    nodosche5(i)=cos(((2*i-1)*pi)/(2*(n+1)));
end
%valores de la funcion en los nodos.
valorche5=frunge(nodosche5);
```

```
%polinomio de interpolacion
p5=polyfit(nodosche5,valorche5,n);
%repetimos con 11 nodos
n=10;
for i=1:n+1
    nodosche10(i)=cos(((2*i-1)*pi)/(2*(n+1)));
end
valorche10=frunge(nodosche10);
p10=polyfit(nodosche10,valorche10,n);
%repetimos con 16 nodos
n=15;
for i=1:n+1
   nodosche15(i)=cos(((2*i-1)*pi)/(2*(n+1)));
end
valorche15=frunge(nodosche15);
p15=polyfit(nodosche15,valorche15,n);
%repetimos con 21 nodos
n=20;
for i=1:n+1
    nodosche20(i)=cos(((2*i-1)*pi)/(2*(n+1)));
end
valorche20=frunge(nodosche20);
p20=polyfit(nodosche20,valorche20,n);
%500 puntos del intervalo [-1,1] igualmente espaciados.
t=linspace(-1,1,500);
%dibujo de la funcion, los nodos y su polinomio interpolador
subplot(2,2,1),plot(t,frunge(t),'b',nodosche5,valorche5,'r*',t,polyval(p5,t),'k')
%ventana grafica, x en [-1.1,1.1] e y en [-0.2,1.2]
axis([-1.1,1.1,-0.2,1.2])
%le ponemos titulo
title('n=5')
%repetimos lo mismo en la segunda subventana grafica
subplot(2,2,2),plot(t,frunge(t),'b',nodosche10,valorche10,'r*',t,polyval(p10,t),'k')
axis([-1.1,1.1,-0.2,1.2])
title('n=10')
%repetimos lo mismo en la tercera subventana grafica
subplot(2,2,3),plot(t,frunge(t),'b',nodosche15,valorche15,'r*',t,polyval(p15,t),'k')
axis([-1.1,1.1,-0.2,1.2])
title('n=15')
%repetimos lo mismo en la cuarta subventana grafica
subplot(2,2,4),plot(t,frunge(t),'b',nodosche20,valorche20,'r*',t,polyval(p20,t),'k')
axis([-1.1,1.1,-0.2,1.2])
title('n=20')
```

Los resultados aparecen en la figura 3.2

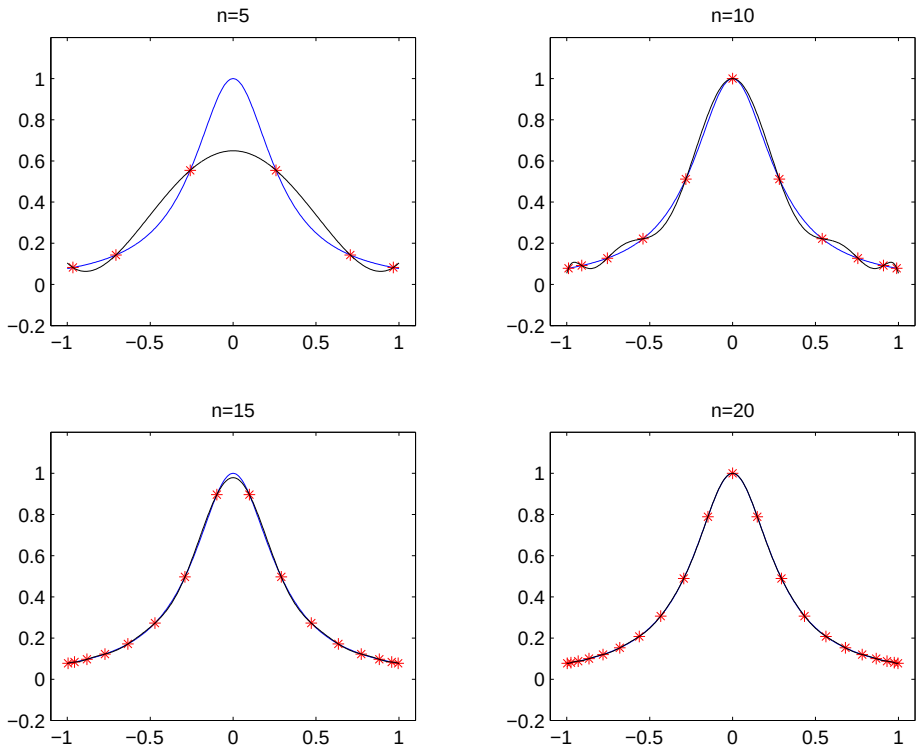

Figura 3.2: Gráfico de la función $f(x) = \frac{1}{1+12x^2}$ y de su polinomio de interpolación con $(6, 11, 16, 21)$ nodos de Chebyshev en $[-1, 1]$.

Apartado (2)

Gráfico del máximo discreto (201 puntos igualmente espaciados en $[-1, 1]$) de $|f(x)-p_n(x)|$ donde $f(x) = 1/(1 + 12x^2)$ y $p_n(x)$ es el polinomio de interpolación de grado n con $n + 1$ nodos igualmente espaciados.

```
%contador
 c=0;
 for n=2:20
 %define la abscisa del error
     c=c+1;
     absc(c)=n;
 %ponemos el maximo de |f(x)-p(x)| a cero
     Max(c)=0;
 %calculo del polinomio de interpolacion con n+1 nodos igualmente
 %espaciados en [-1,1]
     nodos=linspace(-1,1,n+1);
     valores=frunge(nodos);
     p=polyfit(nodos,valores,n);
 %calculo del maximo discreto de |frunge(k)-p(k)| tomando 201 puntos
 %igualmente espaciados en [-1,1]
     for k=-1:.01:1
```

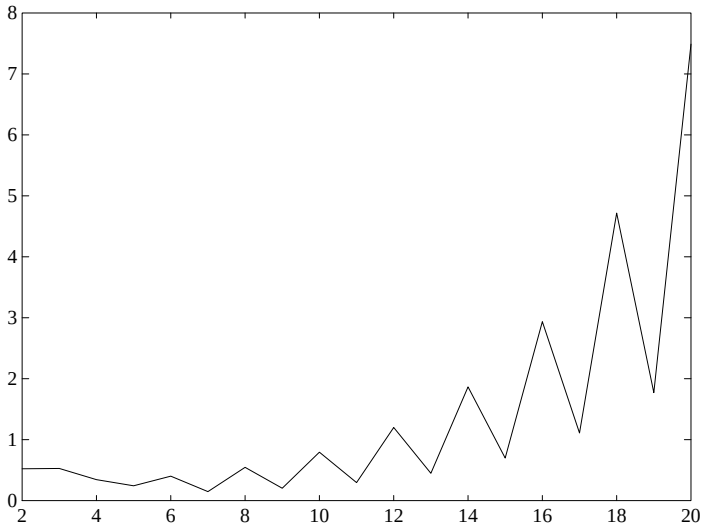

Figura 3.3: Error discreto (max. dis $|f(x) - p_n(x)|$) frente a n ($n+1$ nodos igualmente espaciados).

```
        if abs(frunge(k)-polyval(p,k))>Max(c)
           Max(c)=abs(frunge(k)-polyval(p,k));
        end
    end
end
%dibuja n frente al error cometido para dicho n
plot(absc,Max)
```

Los resultados aparecen representados en la figura 3.3

Apartado (3)

Gráfico del máximo discreto (201 puntos igualmente espaciado de $[-1, 1]$) de $|f(x)-q_n(x)|$ donde $f(x) = 1/(1 + 12x^2)$ y $q_n(x)$ es el polinomio de interpolación de grado n con $n + 1$ nodos de Chebyshev.

```
%contador
c=0;
for n=2:20
    %define la abscisa del error
    c=c+1;
    absc(c)=n;
    %ponemos el maximo de |f(x)-p(x)| a cero
    Max(c)=0;
    %calculo del polinomio de interpolacion con nodos de Chebishev
    %en [-1,1].
    for i=1:n+1
```

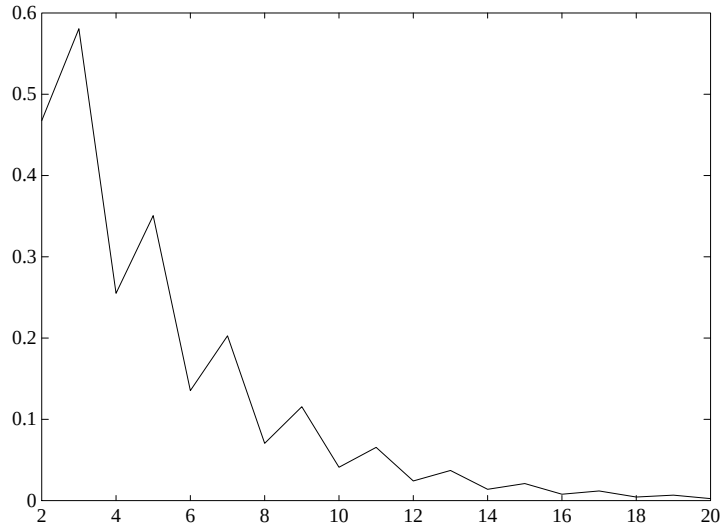

Figura 3.4: Error discreto (max. dis $|f(x) - q_n(x)|$) frente a n ($n+1$ nodos de Chebyshev).

```
        nodosche(i)=cos(((2*i-1)*pi)/(2*(n+1)));
    end
    valoresche=frunge(nodosche);
    q=polyfit(nodosche,valoresche,n);
    %calculo del maximo discreto de |frunge(k)-q(k)| tomando 201 puntos
    %igualmente espaciados en [-1,1]
    for k=-1:.01:1
        if abs(frunge(k)-polyval(q,k))>Max(c)
           Max(c)=abs(frunge(k)-polyval(q,k));
        end
    end
end
%dibuja n frente al error cometido para dicho n
plot(absc,Max)
```

Los resultados aparecen representados en la figura 3.4

PROBLEMA 3-2: Considera el problema de interpolación definido por los siguientes datos:

Nodos:	0	1	2	3	4	5	6	7	8	9	10
Valores:	0	0.8	0.9	0.6	0.4	0.6	0.2	0.6	0.9	0.3	0

(1) Calcula el *spline* cúbico interpolador de tipo periódico. Representar el polinomio interpolador y sus nodos

(2) Calcula el *spline* cúbico interpolador de tipo natural. Representar el polinomio interpolador y sus nodos

(3) Calcula el *spline* cúbico interpolador de tipo sujeto cuya derivada en el primer nodo es -1 y en el último nodo es 0. Representar el polinomio interpolador y sus nodos

(4) Calcula el *spline* cúbico interpolador de tipo no-nodo. Representar el polinomio interpolador y sus nodos

SOLUCIÓN PROBLEMA 3-2:

Apartado (1)

Para resolverlo escribimos en la ventana de comandos:

```
X=0:1:10;
Y=[0,0.8,0.9,0.6,0.4,0.6,0.2,0.6,0.9,0.3,0];
S=splineperiod(X,Y)
```

obteniéndose los siguientes coeficientes:

```
  -0.5708    0.9718    0.3990         0
   0.2105   -0.7407    0.6301    0.8000
   0.0287   -0.1091   -0.2196    0.9000
   0.1746   -0.0230   -0.3517    0.6000
  -0.4273    0.5010    0.1263    0.4000
   0.5344   -0.7809   -0.1536    0.6000
  -0.3105    0.8225   -0.1120    0.2000
  -0.1923   -0.1091    0.6014    0.6000
   0.2799   -0.6861   -0.1938    0.9000
   0.2727    0.1536   -0.7263    0.3000
```

Para representarlo creamos el fichero de función

```
trazadorcubicoperiod.m
```

cuyo contenido es el siguiente:

```
function y=trazadorcubicoperiod(X,Y,x)
% Aqui hallamos el valor en x del trazador cubico periodico
% determinado por los nodos (X,Y)
%                 Datos de entrada
```

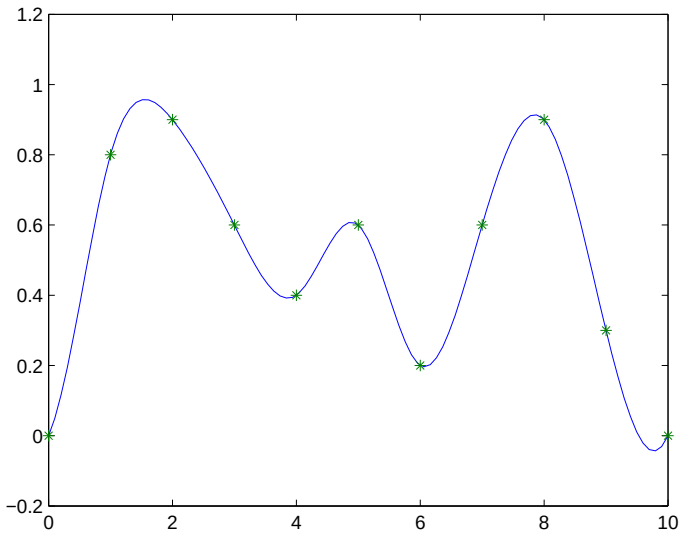

Figura 3.5: trazador cúbico periódico y sus nodos.

```
%      -X es un vector 1x(n+1) que contiene las abscisas
%      -Y es un vector 1x(n+1) que contine las ordenadas
%      -x es el valor donde queremos calcular su valor
%                  Datos de salida
%      -y es el valor de la función en x
% trazador cubico periodico
S=splineperiod(X,Y);
y=ppval(mkpp(X,S),x);
```

y al escribir

```
X=0:1:10;
Y=[0,0.8,0.9,0.6,0.4,0.6,0.2,0.6,0.9,0.3,0];
h=linspace(X(1),X(11),100);
yperiod=trazadorcubicoperiod(X,Y,h);
plot(h,yperiod,'-',X,Y,'*')
```

aparece la figura 3.5.

Apartado (2)

Para resolverlo escribimos el comando:

```
S=splinenatural(X,Y)
```

obteniendose los siguientes coeficientes:

```
  -0.1601         0    0.9601         0
   0.1005   -0.4803    0.4798    0.8000
   0.0582   -0.1789   -0.1794    0.9000
   0.1666   -0.0042   -0.3624    0.6000
  -0.4246    0.4956    0.1290    0.4000
   0.5318   -0.7782   -0.1536    0.6000
  -0.3025    0.8171   -0.1146    0.2000
  -0.2219   -0.0903    0.6122    0.6000
   0.3900   -0.7559   -0.2340    0.9000
  -0.1380    0.4140   -0.5760    0.3000
```

Para representarlo creamos el fichero de función

```
trazadorcubiconatural.m
```

cuyo contenido es el siguiente:

```
function y=trazadorcubiconatural(X,Y,x)
% Aqui hallamos el valor en x del trazador cubico natural
% determinado por los nodos (X,Y)
%                    Datos de entrada
%      -X es un vector 1x(n+1) que contiene las abscisas
%      -Y es un vector 1x(n+1) que contine las ordenadas
%      -x es el valor donde queremos calcular su valor
%                    Datos de salida
%      -y es el valor de la funcion en x
% trazador cubico natural
S=splinenatural(X,Y);
y=ppval(mkpp(X,S),x);
```

Escribiendo en la ventana de comandos

```
X=0:1:10;
Y=[0,0.8,0.9,0.6,0.4,0.6,0.2,0.6,0.9,0.3,0];
h=linspace(X(1),X(11),100);
ynatural=trazadorcubiconatural(X,Y,h);
plot(h,ynatural,'-',X,Y,'*')
```

aparece la figura 3.6

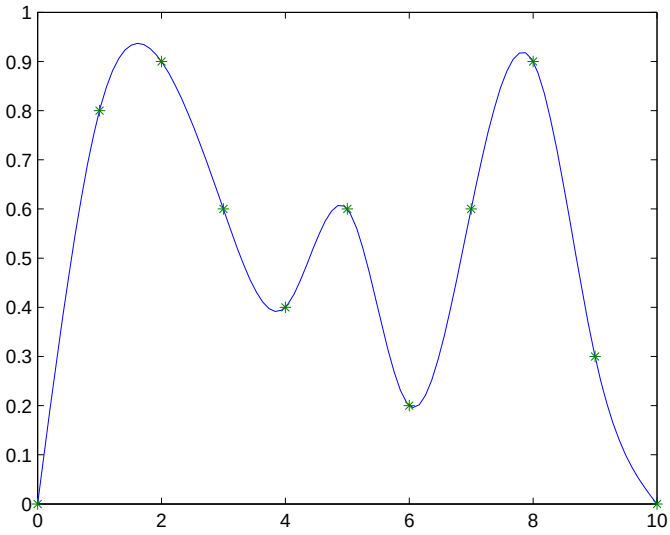

Figura 3.6: trazador cúbico natural y sus nodos.

Apartado (3)

Para resolverlo escribimos los comandos:

```
Y1=[-1,Y,0];
 S=spline(X,Y1);
 [BREAKS,COEFS,L,K,D] = UNMKPP(S);
```

obteniendose los siguientes coeficientes:

```
COEFS =

   -1.5950    3.3950   -1.0000         0
    0.4849   -1.3900    1.0050    0.8000
   -0.0448    0.0649   -0.3201    0.9000
    0.1942   -0.0695   -0.3247    0.6000
   -0.4321    0.5132    0.1190    0.4000
    0.5344   -0.7833   -0.1511    0.6000
   -0.3053    0.8198   -0.1145    0.2000
   -0.2132   -0.0960    0.6093    0.6000
    0.3581   -0.7357   -0.2225    0.9000
   -0.0194    0.3388   -0.6194    0.3000
```

Para representarlo escribimos

```
 X=0:1:10;
 Y=[0,0.8,0.9,0.6,0.4,0.6,0.2,0.6,0.9,0.3,0];
 Y1=[-1,Y,0];
 h=linspace(X(1),X(11),100);
```

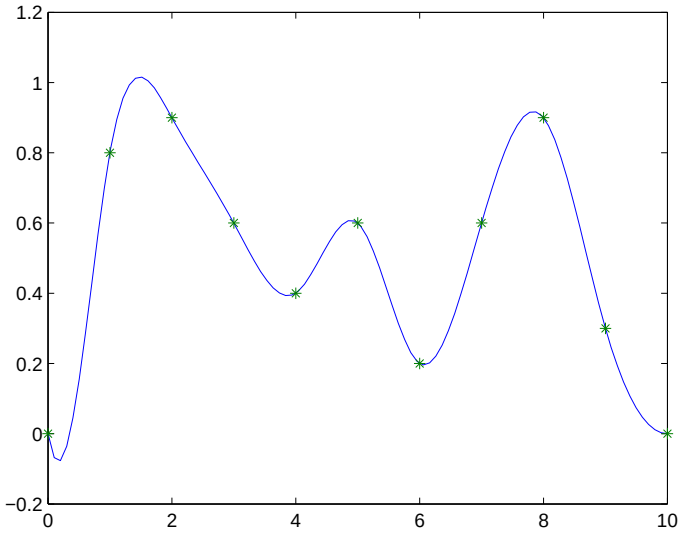

Figura 3.7: trazador cúbico sujeto y sus nodos.

```
ysujeto=spline(X,Y1,h);
plot(h,ysujeto,'-',X,Y,'*')
```

Los resultados aparecen representados en la figura 3.7

Apartado (4)

Para resolverlo escribimos los comandos:

```
S=spline(X,Y);
[BREAKS,COEFS,L,K,D] = UNMKPP(S);
```

obteniendose los siguientes coeficientes:

```
COEFS =

    0.0454   -0.4862    1.2408         0
    0.0454   -0.3500    0.4046    0.8000
    0.0730   -0.2138   -0.1592    0.9000
    0.1628    0.0051   -0.3679    0.6000
   -0.4241    0.4934    0.1307    0.4000
    0.5336   -0.7789   -0.1548    0.6000
   -0.3104    0.8220   -0.1116    0.2000
   -0.1920   -0.1092    0.6012    0.6000
    0.2784   -0.6852   -0.1932    0.9000
    0.2784    0.1500   -0.7284    0.3000
```

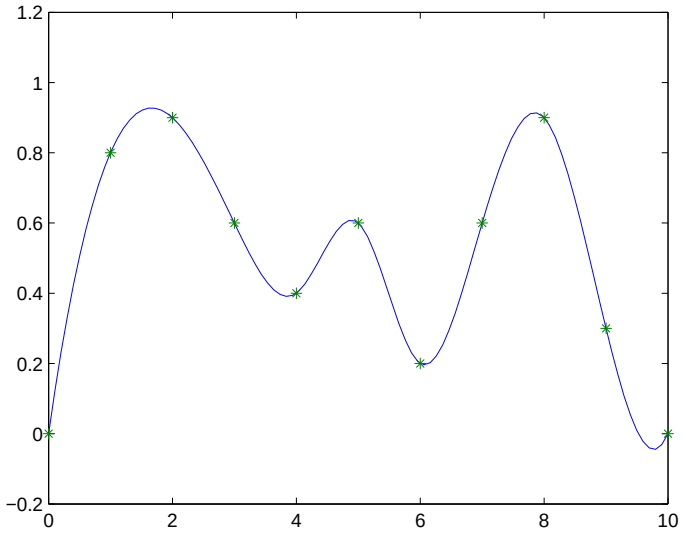

Figura 3.8: trazador cúbico no-nodo y sus nodos.

Para representarlo escribimos

```
X=0:1:10;
Y=[0,0.8,0.9,0.6,0.4,0.6,0.2,0.6,0.9,0.3,0];
h=linspace(X(1),X(11),100);
ynonodo=spline(X,Y,h);
plot(h,ynonodo,'-',X,Y,'*')
```

Los resultados aparecen representados en la figura 3.8

NOTA: Los *splines* que calcula **MATLAB** son los llamados *'not-a-knot'*, es decir, impone como condiciones adicionales que los trazadores cúbicos en los intervalos $[x_1, x_2]$ y $[x_2, x_3]$ sean el mismo y en los intervalos $[x_{n-1}, x_n]$ y $[x_n, x_{n+1}]$ también sean el mismo. Por esta razón los trazadores cúbicos en los intervalos $[0, 1]$ y $[1, 2]$, $S_1(x) = 0{,}0454x^3 - 0{,}4862x^2 + 1{,}2408x$, y $S_2(x) = 0{,}0454(x-1)^3 - 0{,}35(x-1)^2 + 0{,}4046(x-1) + 0{,}8$ coinciden. Y en los intervalos $[8, 9]$ y $[9, 10]$, $S_9(x) = 0{,}2784(x-8)^3 - 0{,}6852(x-8)^2 - 0{,}1932(x-8) + 0{,}9$, y $S_{10}(x) = 0{,}2784(x-9)^3 + 0{,}15(x-9)^2 - 0{,}7284(x-9) + 0{,}3$ también coinciden.