Tema 3

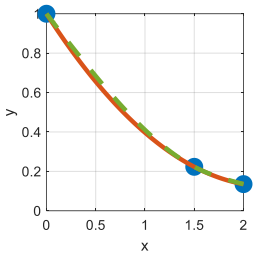
Métodos Numéricos Aplicados I

Interpolación

Índice

Esquema

INTERPOLACIÓN



$\boldsymbol{n+1}$ puntos

$\{(x_0, f(x_0)), (x_1, f(x_1)), \dots, (x_n, f(x_n))\}$

Polinomios de grado $\boldsymbol{n}$

- Newton
- Lagrange
- Hermite

Funciones a trozos

- Splines
  - Naturales
  - Restringidos

3.1 Introducción y objetivos

Cada 10 años, el Instituto Nacional de Estadística de España realiza el censo de población. Los datos del histórico desde el año 1900 se muestran en la Tabla 1 y se representan en la Figura 1.

Año	1900	1910	1920	1930	1940	1950
Población [MPersonas]	18.617	19.991	21.389	23.677	26.014	28.118
Año	1960	1971	1981	1991	2001	2011
Población [MPersonas]	30.583	33.956	37.743	39.434	40.847	46.816

Tabla 1: Datos de la evolución demográfica en España

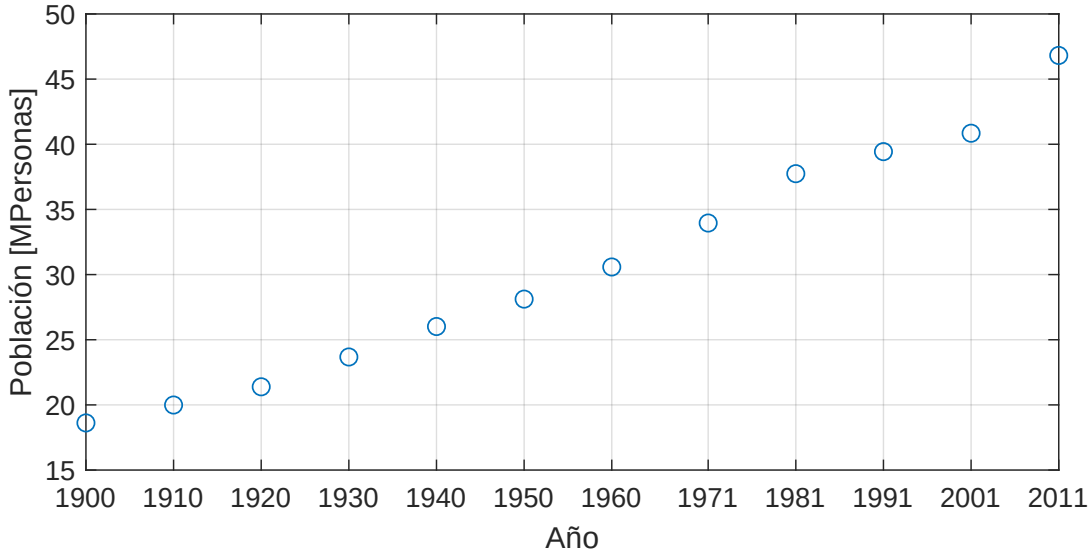


Figura 1: Representación de la evolución demográfica en España

El hecho de que la toma de datos tenga lugar cada 10 años (entre 1960 y 1971 pasaron

11 años) hace que no podamos tener datos más finos acerca de lo que ocurre a mitad de la década o en un año en concreto.

La interpolación consiste en obtener precisamente esos datos internos que no conocemos, a partir de una serie de aproximaciones. En caso de que quisiéramos saber qué población hubo antes de 1900 o habrá más allá de 2011 estaríamos hablando de extrapolación, puesto que serían puntos fuera del intervalo de nuestros datos.

Para obtener los puntos intermedios en los que no conocemos los datos, utilizaremos polinomios de la forma

$$p_n(x) = \sum_{i=0}^{n} a_i x^i, \quad n \geq 0, \quad a_i \in \mathbb{R}. \tag{1}$$

Generalmente se utilizan polinomios para la interpolación porque sus derivadas e integrales dan como resultado otros polinomios. Además, dada cualquier función definida en un intervalo cerrado, existe un polinomio que se ajusta cuanto se quiera a la función, tal y como indica en Teorema 1.

Teorema 1: Teorema de aproximación de Weierstrass

Sea $f \in \mathcal{C}[a, b]$. Entonces, para todo $\epsilon > 0$ existe un polinomio $p(x)$ tal que

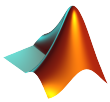
$$|f(x) - p(x)| < \epsilon, \quad \forall x \in [a, b].$$

A lo largo del tema veremos diferentes estrategias para la interpolación, basadas en polinomios de diferentes autores.

Los objetivos que trataremos de alcanzar en este tema serán los siguientes:

- Conocer los polinomios de interpolación de Newton, Lagrange y Hermite
- Implementar computacionalmente los polinomios de interpolación de Newton, Lagrange y Hermite

- Conocer los splines cúbicos
- Implementar computacionalmente los splines cúbicos naturales



Algunas funciones Matlab utilizadas en este tema

- `syms x`. Declara la variable `x` como simbólica.
- `coefs(p,'All')`. Obtiene los coeficientes de un polinomio `p` que es simbólico.
- `double(c)`. Transforma la variable `c` en una variable de tipo `double`.
- `subs(p,x,x0)`. La variable `p` simbólica depende de la variable `x` simbólica. La función reemplaza el valor de `x0` en `x`.

3.2 Interpolación de Newton

En este apartado trabajaremos con el polinomio de interpolación de Newton. Para comprender el polinomio de grado n, comenzaremos por la interpolación lineal, e iremos aumentando el orden del polinomio. En cualquier caso, en lugar de trabajar con (1) utilizaremos el polinomio genérico

$$p_n(x) = b_0+b_1(x-x_0)+b_2(x-x_0)(x-x_1)+\cdots+b_n(x-x_0)(x-x_1)\cdots(x-x_{n-1}). \quad (2)$$

Interpolación lineal

La interpolación que da lugar a un polinomio de primer grado requiere del conocimiento de dos puntos

$$(x_0, f(x_0)), \quad (x_1, f(x_1)).$$

La obtención de los coeficientes b_0 y b_1 de (2) pasa por forzar a que $p_1(x)$ pase por los dos puntos, es decir,

$$\begin{cases} p_1(x_0) &= f(x_0) = b_0, \\ p_1(x_1) &= f(x_1) = b_0 + b_1(x_1 - x_0), \end{cases}$$

de donde se obtiene que

$$b_0 = f(x_0), b_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0},$$

por lo que el polinomio de interpolación lineal de Newton es

$$p_1(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0). \tag{3}$$

A la expresión de b_1 se la denomina **diferencia dividida de primer orden** y también se escribe como

$$f[x_1, x_0] = \frac{f(x_1) - f(x_0)}{x_1 - x_0},$$

por lo que podemos reescribir (3) como

$$p_1(x) = f(x_0) + f[x_1, x_0](x - x_0).$$

Interpolación cuadrática

El polinomio cuadrático de interpolación de Newton pasa por los tres puntos

$$(x_0, f(x_0)), \quad (x_1, f(x_1)), \quad (x_2, f(x_2)).$$

Si hacemos que el polinomio pase por los puntos que conocemos, obtendremos la expresión

$$p_2(x) = f(x_0) + f[x_1, x_0](x - x_0) + f[x_2, x_1, x_0](x - x_0)(x - x_1), \tag{4}$$

donde $f[x_2, x_1, x_0]$ es la **diferencia dividida de segundo orden**, de expresión

$$f[x_2, x_1, x_0] = \frac{f[x_2, x_1] - f[x_1, x_0]}{x_2 - x_0}.$$

Interpolación general

El polinomio de grado n de interpolación de Newton pasa por los $n + 1$ puntos

$$(x_0, f(x_0)), (x_1, f(x_1)), \ldots (x_n, f(x_n)).$$

Las **diferencias divididas de orden** n se definen a partir de las diferencias divididas de orden $n - 1$ (ver cómo se definieron las de orden dos con respecto de las de primer orden) como

$$f[x_n, x_{n-1}, x_{n-2}, \cdots, x_1, x_0] = \frac{f[x_n, x_{n-1}, \cdots, x_1] - f[x_{n-1}, x_{n-2}, \cdots, x_1, x_0]}{x_n - x_0}.$$

Los coeficientes b_i son las diferencias divididas de orden i, por lo que el polinomio de interpolación de Newton de grado n será

$$\begin{aligned} p_n(x) &= f(x_0) + f[x_1, x_0](x - x_0) + \cdots + \\ &\quad + f[x_n, x_{n-1}, \cdots, x_1, x_0](x - x_{n-1})(x - x_{n-2}) \cdots (x - x_1)(x - x_0). \end{aligned} \tag{5}$$

El Teorema 2 recoge el error cometido al realizar la aproximación polinómica.

Teorema 2: Error en el polinomio de interpolación de Newton

Sean $a = x_0 < x_1 < \ldots < x_n = b$ y sea $f \in \mathcal{C}^{n+1}[a,b]$. Entonces, $\forall x \in [a,b]$ existe un $\xi(x) \in (a,b)$ tal que

$$f(x) = p_n(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)\cdots(x-x_n),$$

donde p_n viene determinado por (5).

Ejemplo 1. Dados los datos del censo de población de España desde 1971, calcula el polinomio de interpolación de Newton de mayor grado posible para estimar la población en el año 2005.

Los datos que disponemos son

Año	1971	1981	1991	2001	2011
Población [MPersonas]	33.956	37.743	39.434	40.847	46.816

Al disponer de 5 datos, obtendremos un polinomio de grado 4 de la forma

$$\begin{aligned} p_4(x) &= f(x_0) + f[x_1,x_0](x-x_0) + f[x_2,x_1,x_0](x-x_0)(x-x_1) + \\ &+ f[x_3,x_2,x_1,x_0](x-x_0)(x-x_1)(x-x_2) + \\ &f[x_4,x_3,x_2,x_1,x_0](x-x_0)(x-x_1)(x-x_2)(x-x_3). \end{aligned}$$

Calculamos los coeficientes y obtenemos

$$\begin{aligned} &f(x_0) = f(1971) = 33.956, \\ &f[x_1,x_0] = f[1981,1971] = 0.3787, \\ &f[x_2,x_1,x_0] = f[1991,1981,1971] = -0.01048, \\ &f[x_3,x_2,x_1,x_0] = f[2001,1991,1981,1971] = 0.000303, \\ &f[x_4,x_3,x_2,x_1,x_0] = f[2011,2001,1991,1981,1971] = 0.0000127. \end{aligned}$$

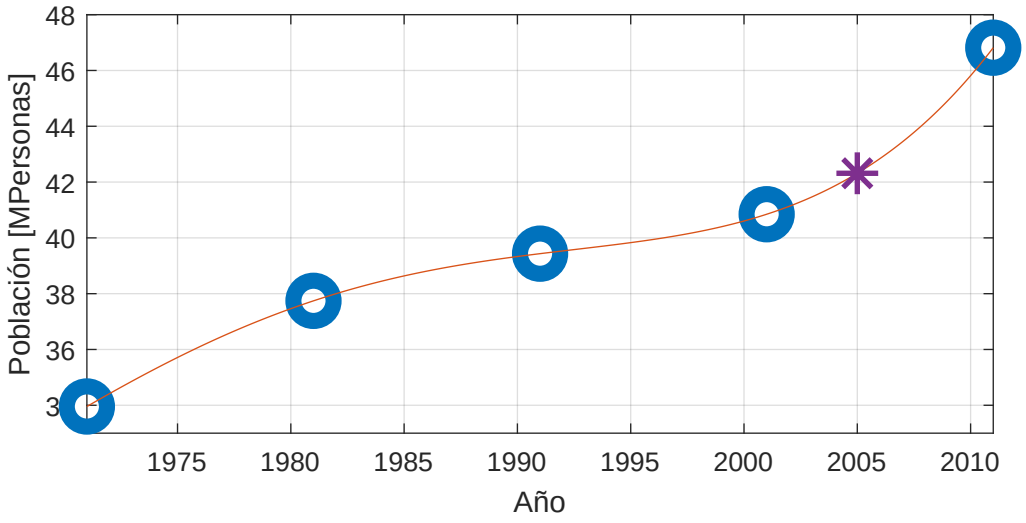
De modo que el polinomio es

$$\begin{aligned} p_4(x) &= 33.956 + 0.3787(x-1971) - 0.01048(x-1971)(x-1981) + \\ &\quad +0.000303(x-1971)(x-1981)(x-1991) + \\ &\quad 0.0000127(x-1971)(x-1981)(x-1991)(x-2001). \end{aligned}$$

Si evaluamos el polinomio en el año 2005, obtenemos

$$p(2005) = 42.315 \text{ MPersonas}.$$

A continuación se muestran en azul los datos, en naranja el polinomio de interpolación evaluado de forma continua y en morado el valor del año 2005 obtenido por interpolación.



Implementación computacional del polinomio de interpolación de Newton

A continuación vamos a plantear cómo debería ser un programa que obtenga el polinomio de interpolación de Newton.

- **Entrada** Los $n+1$ puntos conocidos $(x_0, f(x_0)), \ldots, (x_n, f(x_n))$. Las coordenadas

x_i de los puntos irán en un vector, y las coordenadas $f(x_i)$ de los puntos en otro vector.

- **Salida** Los $n+1$ coeficientes $b_0, \ldots, b_n$ y el polinomio $p(x)$.

La Figura 2 representa la estructura de cómo se generan las diferencias divididas en los diferentes puntos. Nótese que los coeficientes necesarios para generar el polinomio de interpolación de Newton están marcados en azul.

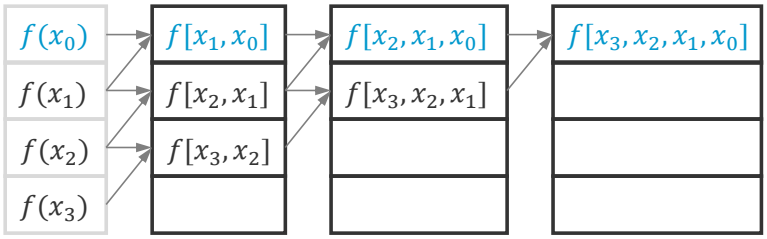


Figura 2: Generación de las diferencias divididas

Dada la estructura de la Figura 2, una posible implementación se basa en rellenar columnas paso a paso. En cada uno de los pasos introduciremos una nueva columna i que serán las diferencias divididas de orden $i-1$; los términos por debajo de la diagonal inferior no se tendrán en cuenta. Además, para cada paso i generaremos el sumando del polinomio de interpolación.

Los pasos se recogen a continuación.

- La primera columna de f_i contendrá los valores de entrada $f(x_i)$ y se inicializará el polinomio $p(x) = f(x_0)$.
- La segunda columna será la correspondiente a las diferencias divididas de primer orden, que se calculará a partir de la primera columna y los valores de x_i; el polinomio $p(x)$ se actualizará sumándole $f[x_1, x_0](x - x_0)$. Nótese que al tener el término $x - x_0$ estamos trabajando con una variable simbólica.
- La tercera columna será la correspondiente a las diferencias divididas de segundo orden, que se calculará a partir de la segunda columna y los valores de x_i; el

polinomio $p(x)$ se actualizará sumándole $f[x_2, x_1, x_0](x - x_0)(x - x_1)$

- En general, la columna i será la correspondiente a las diferencias divididas de orden $i - 1$, que se calculará a partir de la columna $i - 1$ y los valores de x_i; el polinomio $p(x)$ se actualizará sumándole $f[x_{i-1}, \ldots, x_0](x - x_0) \cdots (x - x_{i-2})$.

La Figura 3 presenta un diagrama de flujo para la implementación computacional del polinomio de interpolación de Newton.

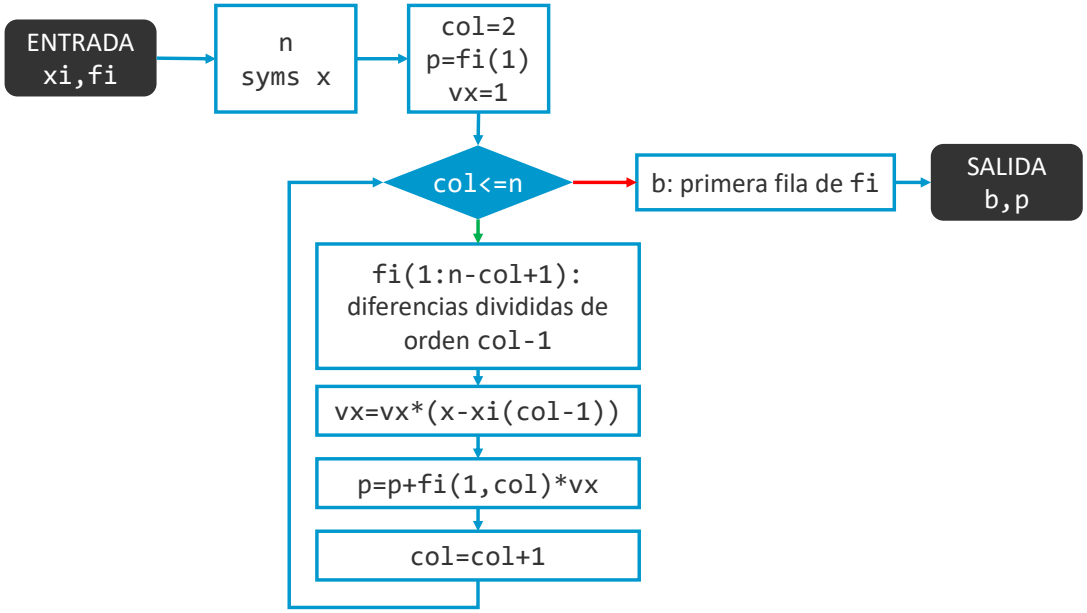


Figura 3: Diagrama de flujo para la obtención del polinomio de interpolación de Newton

Al programa que genere el polinomio de interpolación de Newton lo nombraremos `polinomioNewton.m`.

Ejemplo 2. Resolver el Ejemplo 1 computacionalmente.

Trabajamos con formato de representación `shortg`. Ejecutamos sobre la consola

```
>> xi=1971:10:2011;
>> fi=[33.956 37.743 39.434 40.847 46.816];
>> [b,p]=polinomioNewton(xi,fi)
```

Obtenemos los coeficientes b_i

```
b =
   33.956        0.3787      -0.01048      0.000303    ...
      1.2567e-05
```

y el polinomio $p_4(x)$

```
p =
(3787*x)/10000 - (131*(x - 1971)*(x - 1981))/12500 ...
   + (5589363454334033*(x - 1971)*(x - 1981)*(x - ...
   1991))/18446744073709551616 + ...
   (7418050683507677*(x - 1971)*(x - 1981)*(x - ...
   1991)*(x - 2001))/590295810358705651712 - ...
   7124617/10000
```

Si quiero obtener los coeficientes a_i del polinomio para que esté expresado en la forma

$$p_4(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4,$$

ejecuto

```
>> c=double(coeffs(p,'All'))
c =
   1.9309e+08   -3.9013e+05        295.58     ...
      -0.099527    1.2567e-05
```

Para conocer el valor del polinomio en el año 2005, ejecutamos

```
>> p2005=double(subs(p,x,2005))
p2005 =
        42.316
```

3.3 Interpolación de Lagrange

El polinomio de interpolación de Lagrange también es el único polinomio de grado n que pasa por los $n+1$ puntos, al igual que ocurría con la interpolación de Newton. Sin embargo, su generación difiere ligeramente.

Interpolación lineal

A partir de los dos puntos

$$(x_0, f(x_0)), \quad (x_1, f(x_1)),$$

se definen las funciones $L_0(x)$ y $L_1(x)$ tales que

$$\begin{cases} L_0(x_0) = 1, \\ L_0(x_1) = 0, \end{cases} \quad \begin{cases} L_1(x_0) = 0, \\ L_1(x_1) = 1, \end{cases}$$

de modo que

$$L_0(x) = \frac{x - x_1}{x_0 - x_1}, \quad L_1(x) = \frac{x - x_0}{x_1 - x_0}.$$

El polinomio de interpolación de Lagrange de primer orden viene dado por

$$p_1(x) = L_0(x)f(x_0) + L_1(x)f(x_1). \tag{6}$$

Interpolación general

El polinomio de grado n de interpolación de Lagrange pasa por los $n+1$ puntos

$$(x_0, f(x_0)), (x_1, f(x_1)), \dots (x_n, f(x_n)).$$

En este caso, tendremos n funciones $L_i(x)$, que denotaremos por $L_{n,i}(x)$. Estas funciones cumplen

$$\left\{\begin{array}{l} L_{n,0}(x_0) = 1, \\ L_{n,0}(x_1) = 0, \\ \vdots \\ L_{n,0}(x_{n-1}) = 0, \\ L_{n,0}(x_n) = 0, \end{array}\right. \quad \left\{\begin{array}{l} L_{n,1}(x_0) = 0, \\ L_{n,1}(x_1) = 1, \\ \vdots \\ L_{n,1}(x_{n-1}) = 0, \\ L_{n,1}(x_n) = 0, \end{array}\right. \quad \cdots \left\{\begin{array}{l} L_{n,n}(x_0) = 0, \\ L_{n,n}(x_1) = 0, \\ \vdots \\ L_{n,n}(x_{n-1}) = 0, \\ L_{n,n}(x_n) = 1, \end{array}\right.$$

es decir,

$$L_{n,i}(x_j) = \left\{\begin{array}{ll} 1, & i = j, \\ 0, & i \neq j, \end{array}\right.$$

lo que da lugar a las funciones

$$\begin{aligned} L_{n,i}(x) &= \frac{(x-x_0)(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)} = \\ &= \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{(x-x_j)}{(x_i-x_j)} \end{aligned} \tag{7}$$

Nótese que en la expresión anterior, en el numerador aparecen todos los términos $(x-x_j)$ a excepción del término $(x-x_i)$, y en el denominador aparecen todos los términos (x_i-x_j) a excepción del término (x_i-x_i).

De este modo, el polinomio de interpolación de Lagrange de grado n tiene la forma

$$\begin{aligned} p_n(x) &= L_{n,0}(x)f(x_0) + L_{n,1}(x)f(x_1) + \cdots + L_{n,n}(x)f(x_n) = \\ &= \sum_{i=1}^{n} L_{n,i}(x)f(x_i). \end{aligned} \tag{8}$$

El Teorema 3 recoge la expresión del error cometido al realizar una aproximación polinómica. Nótese que tiene la misma expresión que en el caso de la interpolación de Newton.

Teorema 3: Error en el polinomio de interpolación de Lagrange

Sean $a = x_0 < x_1 < \ldots < x_n = b$ y sea $f \in \mathcal{C}^{n+1}[a,b]$. Entonces, $\forall x \in [a,b]$ existe un $\xi(x) \in (a,b)$ tal que

$$f(x) = p_n(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)\cdots(x-x_n),$$

donde p_n viene determinado por (8).

Ejemplo 3. Dados los datos del censo de población de España desde 1971, calcula el polinomio de interpolación de Lagrange de mayor grado posible para estimar la población en el año 2005.

Los datos que disponemos son

Año	1971	1981	1991	2001	2011
Población [MPersonas]	33.956	37.743	39.434	40.847	46.816

Al disponer de 5 datos, obtendremos un polinomio de grado 4. Por simplicidad, las funciones $L_{n,i}(x) = L_{4,i}(x)$ se nombrarán como $L_i(x)$. el polinomio tendrá la forma

$$p_4(x) = L_0 f(x_0) + L_1 f(x_1) + L_2 f(x_2) + L_3 f(x_3) + L_4 f(x_4).$$

Calculamos las funciones $L_i(x), i = 0, 1, \ldots, 4$, y obtenemos

$$L_0(x) = \frac{(x-1981)(x-1991)(x-2001)(x-2011)}{(1971-1981)(1971-1991)(1971-2001)(1971-2011)} =$$

$$= \frac{(x-1981)(x-1991)(x-2001)(x-2011)}{240000},$$

$$
\begin{aligned}
L_1(x) &= \frac{(x-1971)(x-1991)(x-2001)(x-2011)}{(1981-1971)(1981-1991)(1981-2001)(1981-2011)} = \\
&= -\frac{(x-1971)(x-1991)(x-2001)(x-2011)}{60000}, \\
L_2(x) &= \frac{(x-1971)(x-1981)(x-2001)(x-2011)}{(1991-1971)(1991-1981)(1991-2001)(1991-2011)} = \\
&= \frac{(x-1971)(x-1991)(x-2001)(x-2011)}{40000}, \\
L_3(x) &= \frac{(x-1971)(x-1981)(x-1991)(x-2011)}{(2001-1971)(2001-1981)(2001-1991)(2001-2011)} = \\
&= -\frac{(x-1971)(x-1981)(x-1991)(x-2011)}{60000}, \\
L_4(x) &= \frac{(x-1971)(x-1981)(x-1991)(x-2001)}{(2011-1971)(2011-1981)(2011-1991)(2011-2001)} = \\
&= \frac{(x-1971)(x-1981)(x-1991)(x-2011)}{240000}.
\end{aligned}
$$

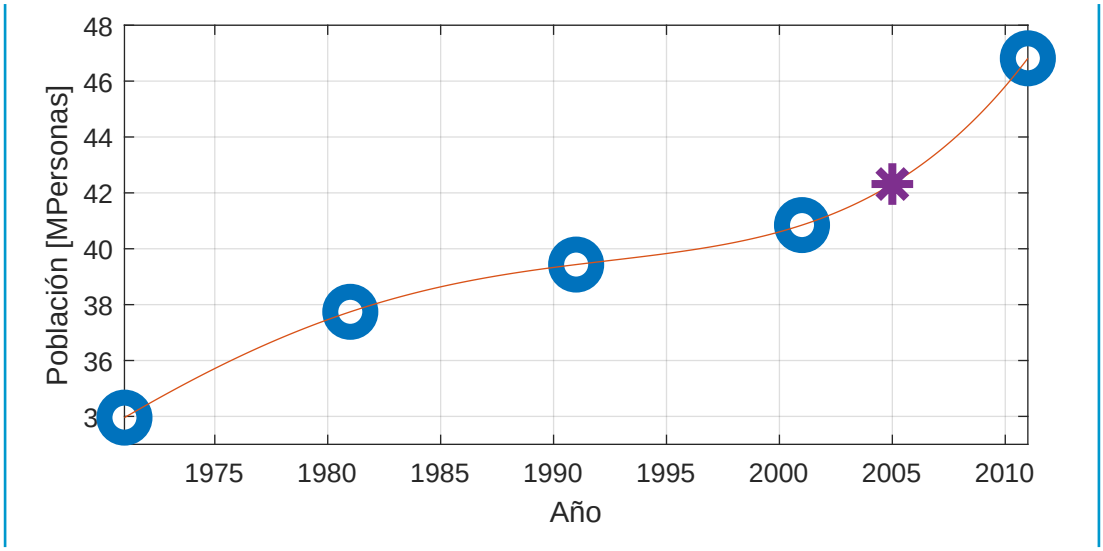
De modo que el polinomio es

$$
\begin{aligned}
p_4(x) = &\tfrac{33.956}{240000}(x-1981)(x-1991)(x-2001)(x-2011)+ \\
&+\tfrac{37.743}{60000}(x-1971)(x-1991)(x-2001)(x-2011)+ \\
&+\tfrac{39.434}{40000}(x-1971)(x-1981)(x-2001)(x-2011)+ \\
&+\tfrac{40.847}{60000}(x-1971)(x-1981)(x-1991)(x-2011)+ \\
&+\tfrac{46.816}{240000}(x-1971)(x-1981)(x-1991)(x-2001).
\end{aligned}
$$

Si evaluamos el polinomio en el año 2005, obtenemos

$$p_4(2005) = 42.316 \text{ MPersonas.}$$

A continuación se muestran en azul los datos, en naranja el polinomio de interpolación evaluado de forma continua y en morado el valor del año 2005 obtenido por interpolación.



Implementación computacional del polinomio de interpolación de Lagrange

En este apartado seguiremos las mismas pautas que en el correspondiente al polinomio de interpolación de Newton en cuanto a parámetros de entrada y salida. No obstante, ahora no vamos a tener los coeficientes b_i, por lo que el único parámetro de salida será $p_n(x)$.

La mayor complejidad del algoritmo reside en la obtención de las funciones $L_i(x), i = 0, 1, \ldots, n$. Una vez obtenidas, solo falta multiplicar cada una de ellas por $f(x_i)$ y sumar todos los términos.

La idea de la generación de cada función L_i se fundamente en el comentario posterior a (7). Si tomamos el vector

$$\begin{bmatrix} x_0 & x_1 & \cdots & x_{i-1} & x_i & x_{i+1} & \cdots & x_{n-1} & x_n \end{bmatrix}$$

y eliminamos el término x_i, será suficiente con generar un vector

$$x - \begin{bmatrix} x_0 & x_1 & \cdots & x_{i-1} & x_{i+1} & \cdots & x_{n-1} & x_n \end{bmatrix}$$

para el numerador, y otro

$$x_i - \begin{bmatrix} x_0 & x_1 & \cdots & x_{i-1} & x_{i+1} & \cdots & x_{n-1} & x_n \end{bmatrix}$$

para el denominador.

La Figura 4 presenta un diagrama de flujo para la implementación computacional del polinomio de interpolación de Lagrange.

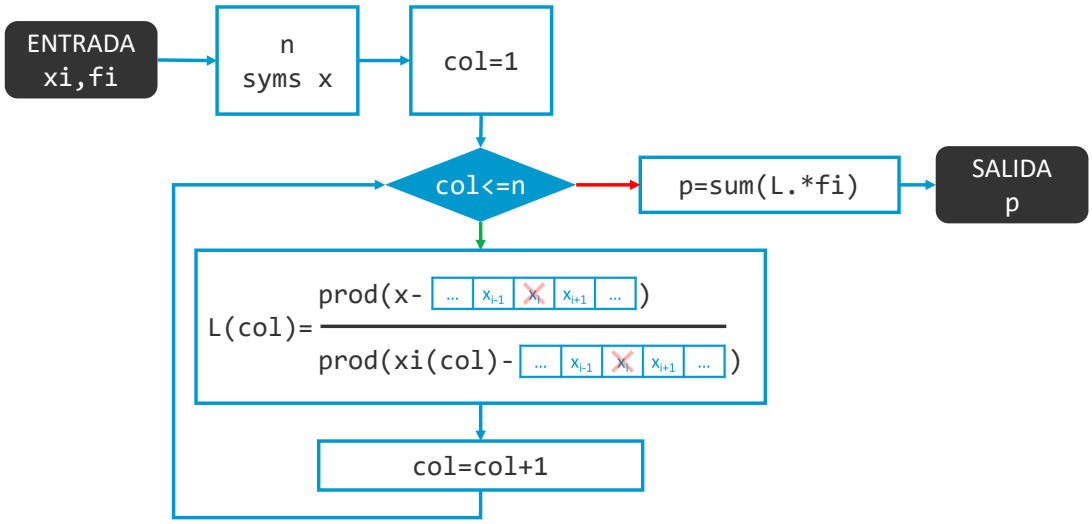


Figura 4: Diagrama de flujo para la obtención del polinomio de interpolación de Lagrange

Al programa que genere el polinomio de interpolación de Lagrange lo nombraremos `polinomioLagrange.m`.

Ejemplo 4. Resolver el Ejemplo 3 computacionalmente.

Trabajamos con formato de representación `shortg`. Ejecutamos sobre la consola

```
>> xi=1971:10:2011;
>> fi=[33.956 37.743 39.434 40.847 46.816];
>> [p]=polinomioLagrange(xi,fi)
p =
(1463*(x - 1971)*(x - 1981)*(x - 1991)*(x - ...
```

```
2001))/7500000 - (40847*(x - 1971)*(x - ...
1981)*(x - 1991)*(x - 2011))/60000000 + ...
(19717*(x - 1971)*(x - 1981)*(x - 2001)*(x - ...
2011))/20000000 - (12581*(x - 1971)*(x - ...
1991)*(x - 2001)*(x - 2011))/20000000 + ...
(8489*(x - 1981)*(x - 1991)*(x - 2001)*(x - ...
2011))/60000000
```

Si quiero obtener los coeficientes a_i del polinomio para que esté expresado en la forma

$$p_4(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4,$$

ejecuto

```
>> c=double(coeffs(p,'All'))
c =
   1.9309e+08  -3.9013e+05       295.58     ...
      -0.099527   1.2567e-05
```

Para conocer el valor del polinomio en el año 2005, ejecutamos

```
>> p2005=double(subs(p,x,2005))
p2005 =
       42.316
```

3.4 Interpolación de Hermite

Hasta ahora, para la generación de los polinomios de interpolación, hemos forzado que el polinomio que generemos pase por los puntos cuyos valores conocemos.

Si además del valor de la función en los puntos conocemos el valor de la derivada de la función sobre los puntos, podemos trabajar con el polinomio de interpolación de Hermite.

Teorema 4

Sean $f \in \mathcal{C}^1[a,b]$ y $x_0, \ldots, x_n \in [a,b]$. Entonces, el único polinomio que coincide con f y f' en los puntos $x_0, \ldots, x_n$ es el polinomio de Hermite de grado menor o igual a $2n+1$ dado por

$$H_{2n+1}(x) = \sum_{i=0}^{n} f(x_i) H_{n,i}(x) + f'(x_i) \hat{H}_{n,i}(x), \tag{9}$$

donde

$$H_{n,i} = \left[1 - 2(x - x_i) L'_{n,i}(x_i)\right] L^2_{n,i}(x), \hat{H}_{n,i} = (x - x_i) L^2_{n,i}(x),$$

siendo $L_{n,i}(x)$ las funciones de Lagrange determinadas en (7).

El procedimiento para la obtención del polinomio de Hermite se recoge en la Figura 5.

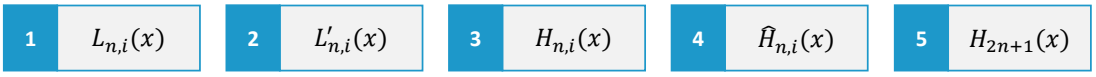


Figura 5: Pasos para la obtención del polinomio de Hermite

Teorema 5: Error en el polinomio de interpolación de Hermite

Sean $a = x_0 < x_1 < \ldots < x_n = b$ y sea $f \in \mathcal{C}^{n+1}[a,b]$. Entonces, $\forall x \in [a,b]$ existe un $\xi(x) \in (a,b)$ tal que

$$f(x) = H_{2n+1}(x) + \frac{f^{(2n+2)}(\xi(x))}{(2n+2)!}(x - x_0)^2 \cdots (x - x_n)^2,$$

donde H_{2n+1} viene determinado por (9).

Ejemplo 5. Las funciones de Bessel de primera especie $J_k(x)$ cumplen

$$\frac{d^k J_k(x)}{dx^k} = (-1)^k J_k(x).$$

Conociendo los datos

$\downarrow x, k \rightarrow$	0	1
0.0000	1.0000	0.0000
0.5000	0.9385	0.2423
1.0000	0.7652	0.4401

calcula el polinomio de interpolación de Hermite para obtener $J_0(0.75)$.

En primer lugar, observamos que

$$J_0'(x) = -J_1(x),$$

de modo que los valores que aparecen en la tabla son los de la función y su derivada. Por tanto, disponemos de los datos necesarios para obtener el polinomio de interpolación de Hermite.

Vemos que $n = 2$, por lo que obtendremos el polinomio $H_5(x)$. De aquí en adelante, en lugar de utilizar $L_{2,i}(x)$, $L'_{2,i}(x)$, $H_{2,i}(x)$ y $\hat{H}_{2,i}(x)$ utilizaremos la misma nomenclatura obviando el primer subíndice. Sigamos los pasos del procedimiento.

1. Cálculo de $L_i(x)$, $i = 0, 1, 2$.

$$\begin{aligned} L_0(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = 2x^2 - 3x + 1, \\ L_1(x) &= \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = -4x^2 + 4x, \\ L_2(x) &= \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = 2x^2 - x. \end{aligned}$$

2. Cálculo de $L_i'(x)$, $i = 0, 1, 2$.

$$L_0'(x) = 4x - 3, \quad L_1'(x) = -8x + 4, \quad L_2(x) = 4x - 1.$$

3. Cálculo de $H_i(x)$, $i = 0, 1, 2$.

$$\begin{aligned} H_0(x) &= [1 - 2(x-x_0)L_0'(x_0)]\, L_0^2(x) = 24x^5 - 68x^4 + 66x^3 - 23x^2 + 1, \\ H_1(x) &= [1 - 2(x-x_1)L_1'(x_1)]\, L_1^2(x) = 16x^4 - 32x^3 + 16x^2, \\ H_2(x) &= [1 - 2(x-x_2)L_2'(x_2)]\, L_2^2(x) = -24x^5 + 52x^4 - 34x^3 + 7x^2. \end{aligned}$$

4. Cálculo de $\hat{H}_i(x)$, $i = 0, 1, 2$.

$$\begin{aligned} \hat{H}_0(x) &= (x-x_0)L_0^2(x) = 4x^5 - 12x^4 + 13x^3 - 6x^2 + x, \\ \hat{H}_1(x) &= (x-x_1)L_1^2(x) = 16x^5 - 40x^4 + 32x^3 - 8x^2, \\ \hat{H}_2(x) &= (x-x_2)L_2^2(x) = 4x^5 - 8x^4 + 5x^3 - x^2. \end{aligned}$$

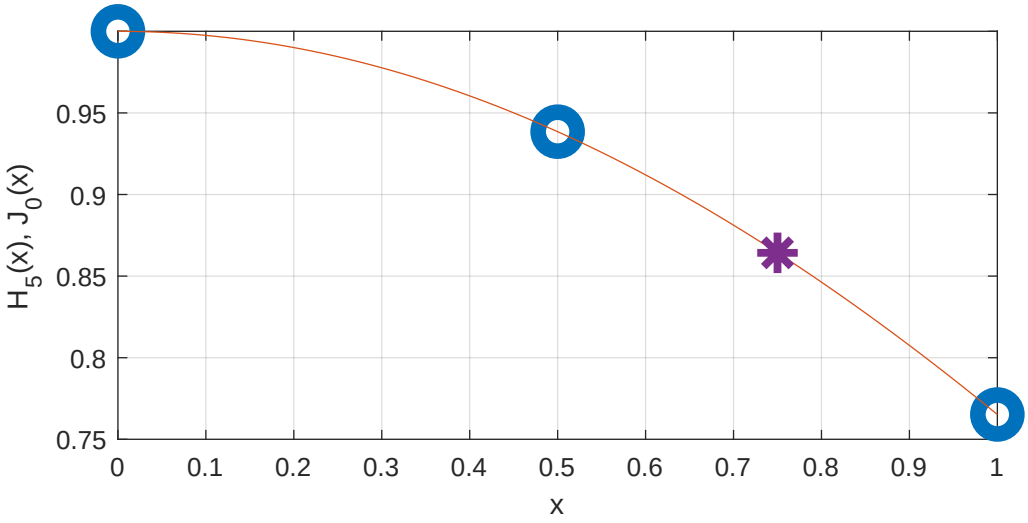
5. Obtención de $H_5(x)$

$$\begin{aligned} H_5(x) &= f(x_0)H_0(x) + f(x_1)H_1(x) + f(x_2)H_2(x) + \\ &\quad + f'(x_0)\hat{H}_0(x) + f'(x_1)\hat{H}_1(x) + f'(x_2)\hat{H}_2(x) = \\ &= -0.002x^5 + 0.0192x^4 - 0.0029x^3 - 0.2491x^2 + 1. \end{aligned}$$

Por tanto, el valor de $J_0(0.75)$ lo aproximamos por

$$J_0(0.75) \approx H_5(0.75) = 0.901461.$$

A continuación se muestran en azul los datos, en naranja el polinomio de interpolación evaluado de forma continua y en morado el valor del polinomio en $x = 0.75$.



Implementación computacional del polinomio de Hermite

Existe una estrecha relación entre el polinomio de interpolación de Hermite (9) y el polinomio de interpolación de Newton (5) a partir de su expresión en diferencias divididas. De hecho, se puede expresar como

$$H_{2n+1}(x) = f(z_0) + \sum_{k=1}^{2n+1} f[z_k, \ldots, z_0](x - z_0)(x - z_1) \cdots (x - z_{k-1}),$$

donde $z_{2i} = z_{2i+i} = x_i$ y $f[z_{2i+1}, z_{2i}] = f'(x_i)$.

La Figura 6 muestra la estructura de la generación de las diferencias divididas para el polinomio de Hermite. Nótese su relación con la Figura 2, que representaba el caso de

Newton.

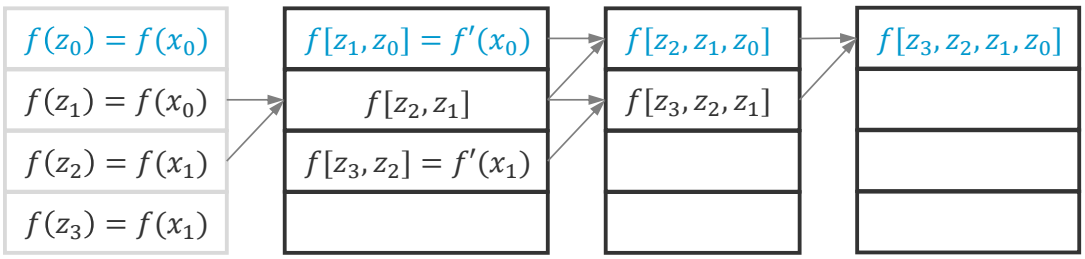


Figura 6: Generación de las diferencias divididas para el polinomio de Hermite

Por tanto, para la implementación computacional tomamos como referencia el correspondiente a Newton, puesto que utilizaba diferencias divididas. Las modificaciones a incluir consisten en la generación del vector z y en la sustitución de las diferencias divididas por las derivadas en los puntos adecuados.

La Figura 7 representa el flujograma de la implementación computacional del método de Hermite.

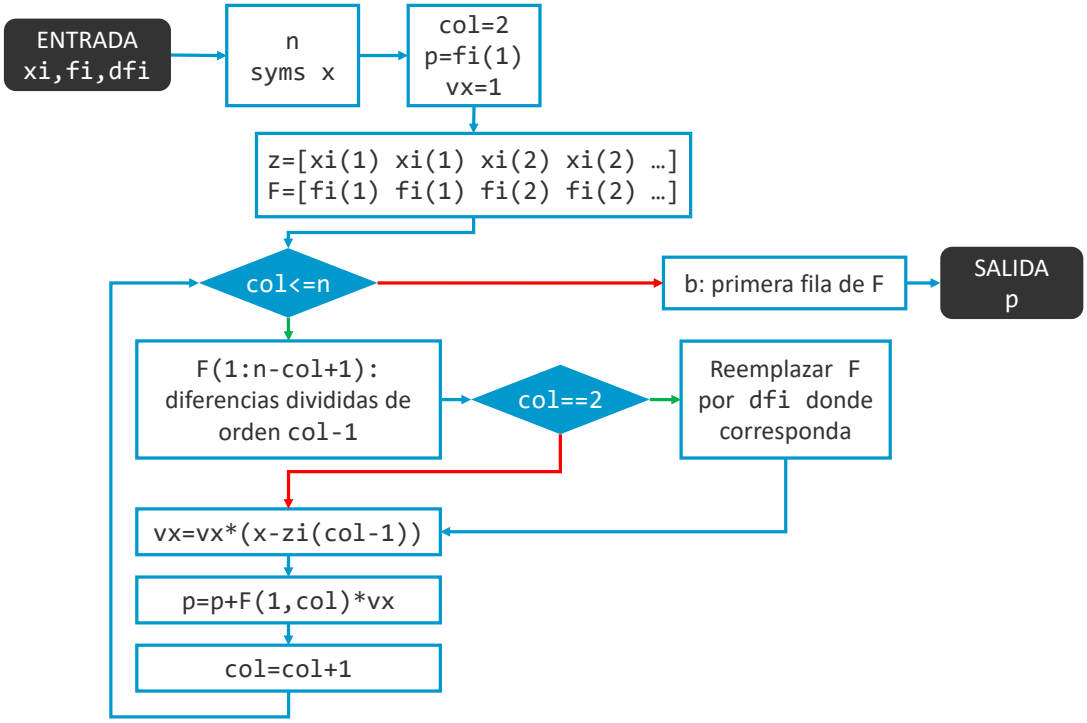


Figura 7: Diagrama de flujo para la obtención del polinomio de interpolación de Hermite

Al programa que genere el polinomio de interpolación de Hermite lo nombraremos `polinomioHermite.m`.

Ejemplo 6. Resolver el Ejemplo 5 computacionalmente.

Trabajamos con formato de representación `shortg`.

Ejecutamos sobre la consola

```
>> xi=[0 .5 1];
>> fi=[1 .9385 .7652];
>> dfi=[0 -.2423 -.4401];
>> p=polinomioHermite(xi,fi,dfi)
p =
 (37*x^2*(x - 1/2))/2500 + (19*x^2*(x - ...
    1/2)^2)/1250 - (123*x^2)/500 - ...
    (9007199254739*x^2*(x - 1)*(x - ...
    1/2)^2)/4503599627370496 + 1
```

Si quiero obtener los coeficientes a_i del polinomio para que esté expresado en la forma

$$p_5(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4,$$

ejecuto

```
>> c=double(coeffs(p,'All'))
c =
      1      -0.2491      -0.0029       0.0192 ...
                -0.002
```

Para conocer el valor del polinomio en punto 0.75, ejecutamos

```
>> p075=double(subs(p,x,.75))
p075 =
       0.86426
```

3.5 Splines

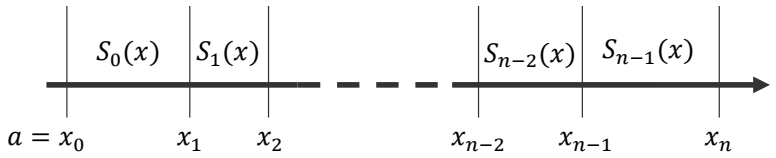
Hasta ahora hemos trabajado con polinomios que se ajustan a una serie de puntos o a una función. La idea de los splines consiste en utilizar un polinomio diferente para cada pareja de puntos consecutivos, de forma que dé lugar a una función a trozos, en la que en cada trozo hay un polinomio.

Los splines más habituales son los cúbicos, cuya forma se describe en la Definición 1.

Definición 1: Splines cúbicos

Sea f una función definida en $a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$, el **spline cúbico** $S(x)$ es una función que cumple las siguientes condiciones.

1. Coincide con la función f en los puntos x_i, es decir, $S(x_i) = f(x_i)$ y $S(x_{i+1}) = f(x_{i+1})$, con $i = 0, 1, \ldots, n-1$.
2. Se define por intervalos, de modo que $S(x) = S_i(x), x_i \leq x \leq x_{i+1}$.



3. Los splines en un intervalo coinciden en los puntos que comparten, de modo que

$$S_{i+1}(x_{i+1}) = S_i(x_{i+1}), \quad i = 0, 1, \ldots, n-2.$$

4. Las primeras derivadas de los splines en un intervalo coinciden en los puntos

que comparten, de modo que

$$S'_{i+1}(x_{i+1}) = S'_i(x_{i+1}), \quad i = 0, 1, \ldots, n-2.$$

5. Las segundas derivadas de los splines en un intervalo coinciden en los puntos que comparten, de modo que

$$S''_{i+1}(x_{i+1}) = S''_i(x_{i+1}), \quad i = 0, 1, \ldots, n-2.$$

6. Las condiciones de contorno pueden ser

   - naturales, si $S''(x_0) = S''(x_n) = 0$, o
   - restringidas, por ejemplo $S'(x_0) = f'(x_0)$ y $S'(x_n) = f'(x_n)$.

Los splines cúbicos van a tener la expresión genérica

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad i = 0, 1, \ldots, n-1. \qquad (10)$$

Bajo las condiciones de la Definición 1 se puede demostrar que la obtención de los $4n$ parámetros de (10) equivale a resolver el sistema de ecuaciones lineales

$$h_{i-1}c_{i-1} + 2(h_{i-1}+h_i)c_i + h_i c_{i+1} = \frac{3}{h_i}(a_{i+1}-a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1}), \quad i = 1, 2, \ldots, n-1, \qquad (11)$$

donde

$$h_i = x_{i+1} - x_i, a_i = f(x_i), b_i = \frac{1}{h_i}(a_{i+1} - a_i) - \frac{h_i}{3}(2c_i + c_{i+1}) \text{ y } d_i = \frac{c_{i+1} - c_i}{3h_i}.$$

Accede al vídeo: Sistema lineal para la obtención de los coeficientes de los splines cúbicos

Splines cúbicos naturales

Teorema 6

Sea f una función definida en los nodos $a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$, entonces f tiene un único spline S en los nodos, es decir, un spline que cumple $S''(a) = S''(b) = 0$.

Las condiciones naturales $S''(x_0) = 0$ y $S''(x_n) = 0$ implican que $c_0 = 0$ y $c_n = 0$, respectivamente. A partir del conocimiento de x_i y $f(x_i)$ obtenemos directamente los valores de h_i y a_i. Desarrollando (11) para cada uno de los valores de i, junto con las condiciones naturales $S''(x_0) = S''(x_n) = 0$, obtenemos el sistema $Ax = b$, donde

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ h_0 & 2(h_0+h_1) & h_1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix},$$

$$x = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} \text{ y } b = 3 \begin{bmatrix} 0 \\ \frac{a_2-a_1}{h_1} - \frac{a_1-a_0}{h_0} \\ \frac{a_3-a_2}{h_2} - \frac{a_2-a_1}{h_1} \\ \vdots \\ \frac{a_n-a_{n-1}}{h_{n-1}} - \frac{a_{n-1}-a_{n-2}}{h_{n-2}} \\ 0 \end{bmatrix}.$$

Se trata de un sistema tridiagonal. La resolución de este tipo de sistemas, para optimizar el aspecto computacional, se lleva a cabo a través del algoritmo de Crout.

El algoritmo de Crout resuelve el sistema de ecuaciones lineal tridiagonal $Ax = b$ descomponiendo la matriz A en una matriz triangular inferior L y una matriz triangular superior U, siguiendo los pasos:

1. Obtención de L y U a partir de A

2. Solución de $Lz = b$

3. Solución de $Ux = z$

A continuación se muestra la implementación del algoritmo en Matlab.


```
function x=Crout(dP,dS,dI,b)
% La función x=Crout(dP,dS,dI,b) obtiene la ...
   solución del sistema Ax=b utilizando el ...
   algoritmo de Crout. dP, dS y dI son las ...
   diagonales principal, superior e inferior de A.
n=length(dP);
% 1. Obtención de las matrices L y U tales que A = LU
l(1)=dP(1);
u(1)=dS(1)/l(1);
for i=2:n-1
    l(i)=dP(i)-dI(i-1)*u(i-1);
    u(i)=dS(i)/l(i);
end
l(n)=dP(n)-dI(n-1)*u(n-1);
% 2. Solución del sistema Lz = b
z(1)=b(1)/l(1);
for i=2:n
    z(i)=(1/l(i))*(b(i)-dI(i-1)*z(i-1));
end
% 3. Solución del sistema Ux = z
x(n)=z(n);
for i=n-1:-1:1
    x(i)=z(i)-u(i)*x(i+1);
end
x=x(:);
```

Una vez sabemos cómo resolver el sistema $Ax = b$, pasemos a la implementación del método para obtener los splines cúbicos naturales.

Los datos de entrada serán los puntos

$$(x_0, f(x_0)), (x_1, f(x_1)), \ldots (x_n, f(x_n)).$$

El polinomio que obtendremos podrá tener la forma (10) o la forma $p(x) = p_0 + p_1 x + p_2 x^2 + p_3 x^3$.

La Figura 8 muestra el flujograma de la obtención de los splines cúbicos naturales.

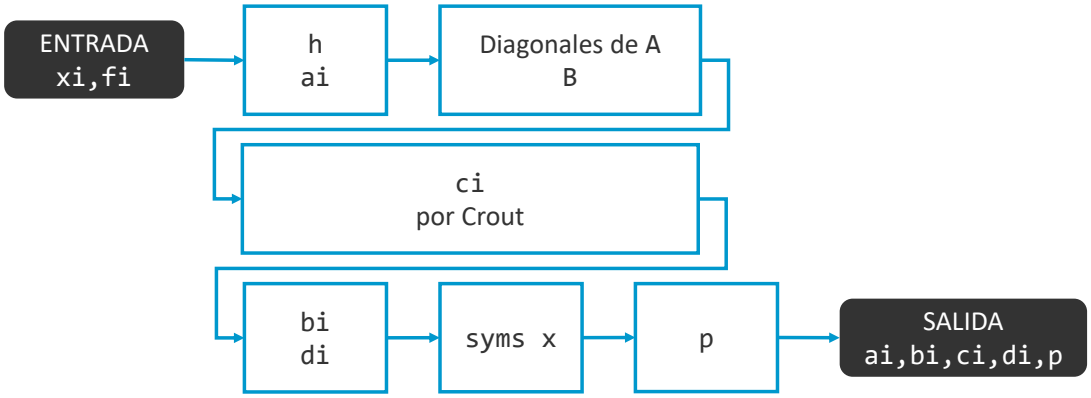


Figura 8: Diagrama de flujo para la obtención de splines cúbicos naturales

Teorema 7: Error en splines cúbicos naturales

Sean $a = x_0 < x_1 < \ldots < x_n = b$, $f \in \mathcal{C}^2[a, b]$ y $h = \text{máx}_i\{h_i\}$. Entonces,

$$|\epsilon(x)| \leq h^{3/2} \sqrt{\left(\int_a^b [f''(x)]^2 \, dx \right)}.$$

Ejemplo 7. Dados los datos del censo de población de España desde 1971, obtén los splines cúbicos naturales para estimar la población en el año 2005.

Ejecutamos sobre la consola

```
>> xi=1971:10:2011;
>> fi=[33.956 37.743 39.434 40.847 46.816];
>> [ai,bi,ci,di,p]=splineCubicoNatural(xi,fi)
ai =
   33.9560
   37.7430
   39.4340
   40.8470
   46.8160

bi =
    0.4247
    0.2867
    0.0720
    0.3563

ci =
         0
   -0.0138
   -0.0077
    0.0361
         0

di =
   -0.0005
    0.0002
    0.0015
   -0.0012

p =
    (1912775267450551*x)/4503599627370496 - ...
```

```
       (4244727573818253*(x - ...
       1971)^3)/9223372036854775808 - ...
       4521444778997554323 53/562949953421312000
    (10033*x)/35000 - (994858025113653*(x - ...
       1981)^2)/72057594037927936 + ...
       (7565800319374475*(x - ...
       1981)^3)/36893488147419103232 - 331328/625
 (1441*x)/20000 - (4412395301202439*(x - ...
    1991)^2)/5764607523034234 88 + ...
    (3361754404361469*(x - ...
    1991)^3)/2305843009213693952 - 2080351/20000
   (1559*x)/4375 + (650023835359643*(x - ...
      2001)^2)/18014398509481984 - ...
      (1386717515433905*(x - ...
      2001)^3)/1152921504606846976 - 23526827/35000
```

El polinomio resultante será

$$S(x) = \begin{cases} S_0(x), & x \in [1971, 1981], \\ S_1(x), & x \in [1981, 1991], \\ S_2(x), & x \in [1991, 2001], \\ S_3(x), & x \in [2001, 2011], \end{cases}$$

donde

$$\begin{aligned} S_0(x) &= 33.9560 + 0.4247(x - 1971) - 0.0005(x - 1971)^3, \\ S_1(x) &= 37.7430 + 0.2867(x - 1981) - 0.0138(x - 1981)^2 + \\ &\quad +0.0002(x - 1981)^3, \\ S_2(x) &= 39.4340 + 0.0720(x - 1991) - 0.0077(x - 1991)^2 + \\ &\quad +0.0015(x - 1991)^3, \\ S_3(x) &= 40.8470 + 0.3563(x - 2001) + 0.0361(x - 2001)^2 - \\ &\quad -0.0012(x - 2001)^3. \end{aligned}$$

Para conocer el valor del polinomio en el año 2005, buscamos el valor $S(2005) = S_3(2005)$, y ejecutamos

```
>> p2005=double(subs(p(4),x,2005))
p2005 =
    42.7727
```

A continuación se muestran en azul los datos, en naranja los splines cúbicos naturales y en morado el valor del año 2005 obtenido por splines.