1 *Formas de Lagrange, Newton y Splines*

Se conoce como problema de interpolación de Lagrange el que se plantea cuando se conoce un conjunto de puntos $\{x_0, x_1, x_2, \ldots x_n\}$ denominados soporte y el valor de la función en ellos $\{f(x_0), f(x_1), f(x_2), \ldots f(x_n)\}$, y se desea obtener un polinomio $P_n(x)$ que coincida con la función en los puntos del soporte.

La primera opción para resolver este problema es plantear directamente el sistema lineal de ecuaciones a que da lugar el conjunto de condiciones, esto es, $P_n(x_k)=f(x_k)$, k=0, 1, 2, … n, siendo $P_n(x) = a_n x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} \ldots + a_2 x^2 + a_1 x + a_0$. Esto conduce al sistema

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{pmatrix}$$

Pero este método da lugar a un sistema muy mal condicionado, con importantes errores numéricos. Su implementación en MATLAB se realiza mediante la instrucción *polyfit* tomando como grado del polinomio el número de datos menos uno.

`p=polyfit(x,y,n)` Obtiene el polinomio de grado n con ajuste mínimo cuadrático a los datos (x,y). Error si length(x) ≤n

`[p,S,m]=polyfit(x,y,n)` Realiza previamente un centrado y escalado de los datos x tomando respectivamente la media y la varianza devueltos en m, esto es, $\tilde{x} = \frac{x - \overline{x}}{s_x}$. Esto mejora las propiedades numéricas del polinomio y del algoritmo de ajuste.

Utilizamos las instrucciones anteriores para interpolar el polinomio x^4+2x-1 mediante 10 puntos equiespaciados en el intervalo [-3,3], y posteriormente representar los puntos y el interpolante.

```
» f=inline('x.^4+2*x-1');x=linspace(-3,3,10);y=f(x);
» p=polyfit(x,y,length(x)-1)
» X=linspace(-3,3);Y=f(X);P=polyval(p,X);
» plot(X,Y,X,P,x,y,'*');grid on;
» legend('funcion','Interpolante','datos')
```

A continuación comparamos las instrucciones anteriores usando la función $f(x) = x\sin(x^2)$ en el intervalo $[-\pi,\pi]$ tomando 8 puntos equiespaciados.

```
» f=inline('x.*sin(x.^2)');
```

```
» n=8;a=-pi;b=pi;
» x=linspace(a,b,n);y=f(x);
» p0=polyfit(x,y,n-1)
» [p1,s,m]=polyfit(x,y,n-1)
» X=linspace(a,b);Y=f(X);P0=polyval(p0,X);P1=polyval(p1,(X-m(1))/m(2));
» plot(X,Y,X,P0,X,P1,x,y,'*');grid on;
» legend('funcion','Interp','InterpN','datos')
```

Se pide:

1. Calcular el polinomio de interpolación de la función $f(x)=e^{-x^2}$ utilizando un soporte de 9 puntos equiespaciados en el intervalo [-2,2]. Representar la función y el polinomio de interpolación.

También se puede obtener el polinomio mediante la forma de Newton de diferencias divididas:

$$P_n(x)=\sum_{k=0}^{n} f[x_0,x_1,\cdots,x_k](x-x_0)(x-x_1)\cdots(x-x_{k-1}).$$

o calcularlo como combinación de los polinomios de Lagrange, definidos como

$$L_k^n(x)=\frac{(x-x_0)...(x-x_{k-1}).(x-x_{k+1})...(x-x_n)}{(x_k-x_0)...(x_k-x_{k-1}).(x_k-x_{k+1})...(x_k-x_n)} \text{ y así } P_n(x)=\sum_{k=0}^{n} f(x_k)L_k^n(x)$$

Las siguientes instrucciones obtienen una matriz cuyas filas son los coeficientes de los polinomios de interpolación de Lagrange a partir de un soporte de puntos X.

```
n=size(x);p=zeros(n);
for i=1:n
   q=x;q(i)=[];q=poly(q);p(i,:)=q/polyval(q,x(i));
end
```

Utilizando los datos del ejemplo $f(x)=x\sin(x^2)$, el polinomio de interpolación se obtiene mediante

```
» P=y*p
```

Se pide:

2. Representar los polinomios de Lagrange cuando se toma como soporte el {0, 1}. Verificar que la suma de ambos polinomios es una recta constante. ¿Cual es su valor?. Repetir lo anterior con {0,1,2}, {0,1,2,3} y {0,1,2,3,4}.
3. Comprobar, con el ejemplo $f(x)=e^{-x^2}$, que el polinomio obtenido usando los polinomios de Lagrange coincide con el obtenido inicialmente.

Independientemente de la forma de escritura y de los problemas de condicionamiento, los polinomios de grado elevado presentan fuertes oscilaciones. Esto origina que habitualmente el incremento de puntos de soporte empeore la función de interpolación en vez de mejorarla. Para solventar este problema, se define la interpolación segmentaria.

1.1 Interpolación Segmentaria

MATLAB tiene definida la función *interp1* que permite realizar interpolación segmentaria sobre un soporte determinado. Las funciones para interpolar de MATLAB se dan en la siguiente tabla:

interp1(x,y,xi,'método')	Obtiene los valores correspondientes a los puntos xi con el método seleccionado a partir de los datos (x,y). Los métodos posibles son: 'nearest': punto más cercano (grado 0) "linear": interpolación lineal (por defecto) 'spline': interpolación cúbica con splines 'pchip': Interpolación cúbica con polinomios de Hermite 'cubic': Ide´ntica a 'pchip'
spline(x,y,xi)	Lo mismo que interp1(x,y,xi,'spline'). Permite spline natutral o sujeta (cuando y tiene dos elementos más que x).
spline(x,yi)	Devuelve en forma especial los resultados (coeficientes, soporte, etc.) del interpolante
pchip(x,y,xi)	Lo mismo que interp1(x,y,xi,'pchip'). Permite spline natutral o sujeta (cuando y tiene dos elementos más que x).
pchip(x,y)	Devuelve en forma especial los resultados (coeficientes, soporte, etc.) del interpolante
interp2(x,y,z,xi,yi,'método')	Interpolación bidimensional
Interp3	Interpolación tridimensional
interpn	Interpolación n-dimensional

Sin embargo, las funciones anteriores no nos dan los coeficientes del polinomio de interpolación (salvo *splines* y *pchip* que lo da en una forma especial).

Utilizamos la interpolación segmentaria para interpolar la función $f(x) = x\sin\left(x^2\right)$

```
» f=inline('x.*sin(x.^2)');
» n=8;a=-pi;b=pi;
» x=linspace(a,b,n);y=f(x);
» X=linspace(a,b);Y=f(X);
» P0=interp1(x,y,X,'linear')
» P1=interp1(x,y,X,'spline')
» plot(X,Y,X,P1,X,P2,x,y,'*');grid on;
» legend('funcion','lineal','spline','datos')
```

Mostrémoslo con la función de Runge.

Se pide:

4. Verificar la oscilación de los polinomios de interpolación clásicos y compararla con las splines usando la función de Runge $f(x) = \dfrac{1}{1+x^2}$ cuando se toman 5 puntos equiespaciados en el intervalo [-5,5]. Representar la función frente a ambos interpolantes. Repetir tomando 9 y 17 puntos.

2 Anexo: Instrucciones necesarias

2.1 Manejo de polinomios

roots(v)	Raíces del polinomio cuyos coeficientes son las componentes del vector v
poly(v)	Polinomio cuyas raíces son las componentes del vector v
poly(A)	Polinomio característico de la matriz A
polyval(v,S)	Evalúa el polinomio definido por el vector v en S
polyvalm(v,S)	Evalúa matricialmente el polinomio v en S
conv(v,w)	Realiza el producto de los polinomios cuyos coeficientes son los elementos de los vectores v y w
[q,r]=deconv(v,w)	Cociente y resto de la división polinómica de v entre w
[n,d,q]=residue(v,w)	Cociente q y expansión del resto en fracciones simples, siendo n y d los respectivos numeradores y denominadores.
[v,w]=residue(n,d,q)	Realiza la operación inversa de la anterior, obteniendo el numerador y denominador a partir de la expansión en fracciones simples.

2.1.1 Derivación e integración

polyder(v)	Deriva el polinomio cuyos coeficientes vienen dados por el vector v
polyder(v,w)	Deriva el polinomio producto de v*w
[q,d]=polyder(v,w)	Deriva el polinomio cociente de v/ w
polyint(v)	Integra el polinomio cuyos coeficientes vienen dados por el vector v

2.1.2 Interpolación y ajuste

v=polyfit(x,y,n)	Obtiene el polinomio de grado n con ajuste mínimo cuadrático a los datos (x,y)

2.1.3 Operaciones simbólicas

sym2poly(s)	Convierte el polinomio simbólico s en un vector cuyas componentes son sus coeficientes
poly2sym(v)	Convierte el vector v en un polinomio cuyos coeficientes son las componentes del vector
collect(s,y)	Reune coeficientes en potencias de la variable y (x por defecto)
expand(s)	Expande polinomios y funciones trigonométricas, logarítmicas y potenciales
factor(s)	Factoriza una expresión simbólica
horner(s)	Realiza la representación encajada de Horner de un polinomio
[n,d]=numden(s)	Obtiene el numerados y denominador de un cociente de polinomios