Interpolación polinómica

Contenidos

Polinomio interpolante

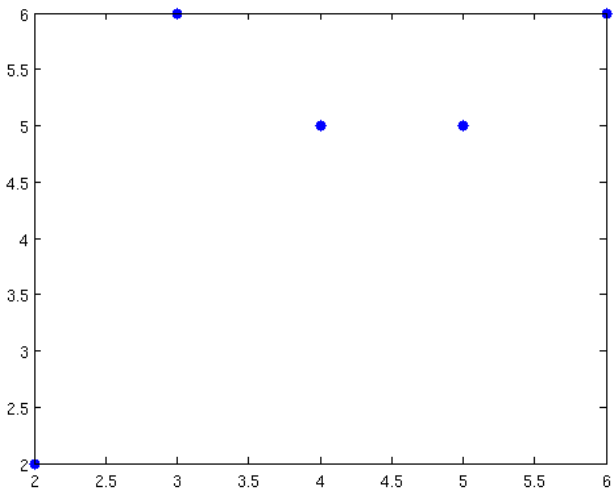
Problema Tenemos una serie de puntos $(x_0,y_0),(x_1,y_1),\ldots,(x_n,y_n)$ y queremos encontrar un polinomio que pase por todos ellos.

Por ejemplo, si tenemos el conjunto de puntos:

$$C=\{(2,2),(3,6),(4,5),(5,5),(6,6)\}$$

Dibujamos los puntos:

```
x=[2,3,4,5,6];
y=[2,6,5,5,6];
plot(x,y,'.','markersize',20)
```



Solución con la matriz de Vandermonde

Como tenemos 5 puntos podemos plantear 5 ecuaciones y necesitamos 5 incógnitas que serán los coeficientes del polinomio. Por lo tanto:

$$P_4(x)=a_0+a_1x+a_2x^2+a_3x^3+a_4x^4$$

es decir a_0,a_1,a_2,a_3,a_4, que son 5 incógnitas y el polinomio es, en principio, de grado 4. En general, para los puntos $(x_0,y_0),(x_1,y_1),\ldots,(x_n,y_n)$ el polinomio interpolante será de grado $\leq n$.

Las ecuaciones serían:

$$\begin{array}{ll} P_4(x_0)=y_0 & a_0+a_1x_0+a_2x_0^2+a_3x_0^3+a_4x_0^4=y_0 \\ P_4(x_1)=y_1 & a_0+a_1x_1+a_2x_1^2+a_3x_1^3+a_4x_1^4=y_1 \\ P_4(x_2)=y_2 & a_0+a_1x_2+a_2x_2^2+a_3x_2^3+a_4x_2^4=y_2 \\ P_4(x_3)=y_3 & a_3+a_1x_3+a_2x_3^2+a_3x_3^3+a_4x_3^4=y_3 \\ P_4(x_4)=y_4 & a_0+a_1x_4+a_2x_4^2+a_3x_4^3+a_4x_4^4=y_4 \end{array}$$

Y el sistema a resolver es:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & x_0^3 & x_0^4 \\ 1 & x_1 & x_1^2 & x_1^3 & x_1^4 \\ 1 & x_2 & x_2^2 & x_2^3 & x_2^4 \\ 1 & x_3 & x_3^2 & x_3^3 & x_3^4 \\ 1 & x_4 & x_4^2 & x_4^3 & x_4^4 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

La matriz de coeficientes del sistema se llama *Matriz de Vandermonde*.

Ejercicio 1

1. Escribir una función `V=Vandermonde(x)` que tenga como argumentos de entrada el vector `x` y como argumento de salida la matriz de Vandermonde `V`.
2. Calcular los coeficientes del polinomio resolviendo el sistema.
3. Dibujar el polinomio

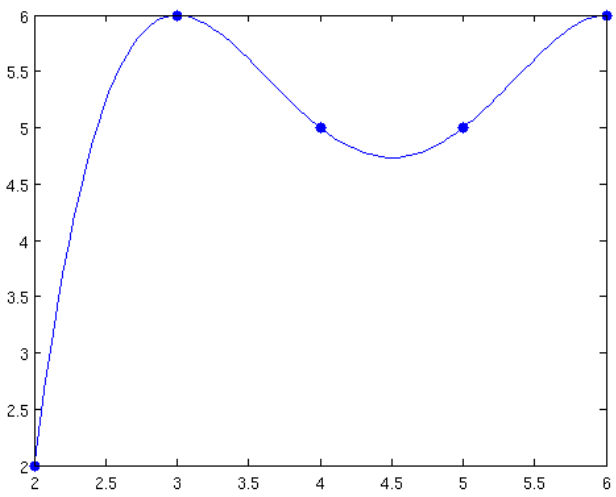
```
V=Vandermonde(x)
```

```
V =
           1           2           4           8          16
           1           3           9          27          81
           1           4          16          64         256
           1           5          25         125         625
           1           6          36         216        1296
```

```
a=V\y'
aa=a(end:-1:1)'  % trasponemos a
                 % y le damos la vuelta,
                 % empezando por el coeficiente de mayor grado
```

```
a =
  -75.0000
   81.0000
  -29.2500
    4.5000
   -0.2500
aa =
   -0.2500    4.5000  -29.2500   81.0000  -75.0000
```

```
xx=linspace(min(x),max(x));
yy=polyval(aa,xx);                  % polyval
                                    % Entrada:
                                    % aa -> los coeficientes del polinomio
                                    % de mayor a menor
                                    % xx -> una serie de puntos
                                    % Salida:
                                    % yy -> valor del polinomio en esos puntos
                                    %
plot(x,y,'.','markersize',20)       % dibujamos los puntos
hold on, plot(xx,yy)                % dibujamos el polinomio
```



Existe una función Matlab que calcula la Matriz de Vandermonde. El orden de los elementos es distinto al que nosotros usamos:

```
vander(x)
```

```
ans =
          16           8           4           2           1
          81          27           9           3           1
         256          64          16           4           1
         625         125          25           5           1
        1296         216          36           6           1
```

Pero este no es un método aconsejable porque la matriz de Vandermonde, en general, está mal condicionada y puede dar lugar a grandes errores en la solución del sistema.

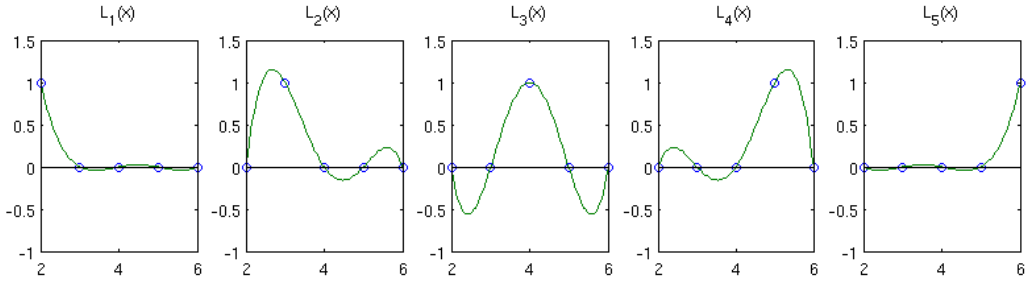
Interpolación mediante los polinomios fundamentales de Lagrange

Para cada $k = 0, 1, \ldots, n$, existe un único polinomio ℓ_k de grado $\leq n$ tal que $\ell_k\left(x_j\right) = \delta_{kj}$ (uno si $k = j$ y cero en caso contrario):

$$\ell_k\left(z\right) = \frac{(z - x_0)\ldots(z - x_{k-1})(z - x_{k+1})\ldots(z - x_n)}{(x_k - x_0)\ldots(x_k - x_{k-1})(x_k - x_{k+1})\ldots(x_k - x_n)}$$

$\ell_0, \ell_1, \ldots, \ell_n$ son los *polinomios fundamentales de Lagrange*. Los 5 *polinomios fundamentales de Lagrange* correspondientes al conjunto de puntos C serían:

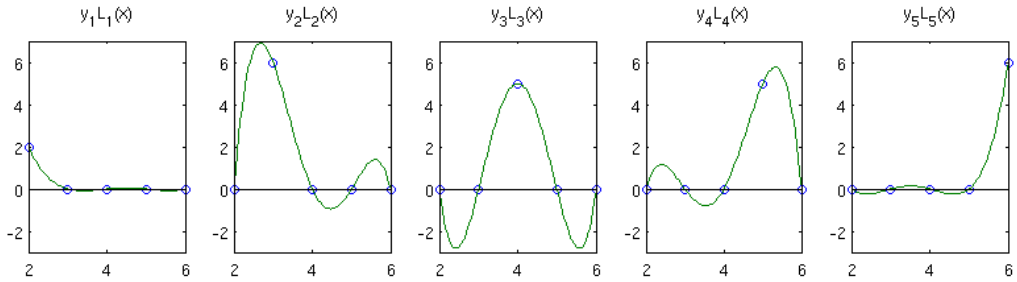
```
polinomios_fundamentales_de_Lagrange
```



Vemos que valen 1 en un nodo y 0 en todos los demás.

Si multiplicamos cada uno de los *polinomios fundamentales de Lagrange* por su correspondiente y_k tenemos $y_0\ell_0, y_1\ell_1, \ldots, y_n\ell_n$:

```
polinomios_fundamentales_de_Lagrange_por_y
```



Vemos que valen y_k en el nodo correspondiente y 0 en todos los demás.

El polinomio de interpolación de Lagrange en los puntos $x_{0,}x_1, \ldots, x_n$ relativo a los valores $y_0, y_1, \ldots, y_n$ es

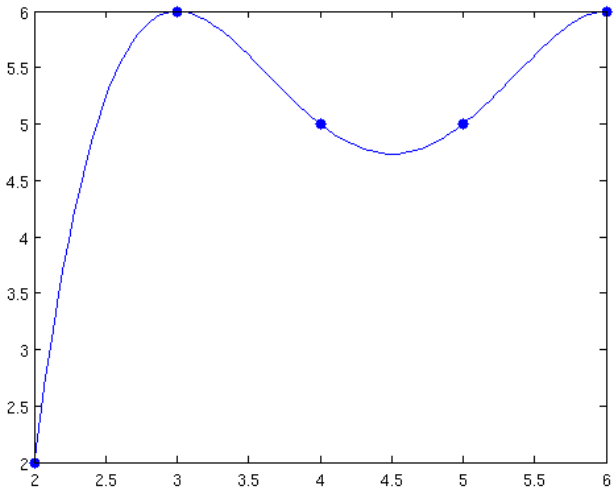
$$P_n\left(x\right) = y_0\ell_0\left(x\right) + y_1\ell_1\left(x\right) + \cdots + y_n\ell_n\left(x\right)$$

Y si sumamos estos cinco polinomios

$$P_4(x) = y_0\ell_0(x) + \cdots + y_4\ell_4(x)$$

obtenemos el polinomio interpolante de Lagrange

```
polinomio_interpolante_Lagrange
```



El inconveniente de este método es que si queremos incorporar un nodo nuevo tenemos que rehacer todos los cálculos.

Interpolación mediante diferencias divididas

Fórmula de Newton

El polinomio de interpolación de Lagrange en los puntos $x_0, x_1, \ldots, x_n$ relativo a los valores $y_0, y_1, \ldots, y_n$ es

$$P_n(x) = [y_0] + [y_0, y_1](x - x_0) + [y_0, y_1, y_2](x - x_0)(x - x_1) + \cdots + [y_0, y_1, \ldots, y_n](x - x_0)(x - x_1)\cdots(x - x_{n-1})$$

Tabla de diferencias divididas

Los coeficientes del polinomio anterior son la primera fila de la tabla:

$$\begin{array}{llllll}
x_0 & y_0 & [y_0,y_1] = \frac{y_0 - y_1}{x_0 - x_1} & [y_0,y_1,y_2] = \frac{[y_0,y_1] - [y_1,y_2]}{x_0 - x_2} & \cdots & [y_0,y_1,\ldots,y_n] = \frac{[y_0,\ldots,y_{n-1}] - [y_1,\ldots,y_n]}{x_0 - x_n} \\
x_1 & y_1 & [y_1,y_2] = \frac{y_1 - y_2}{x_1 - x_2} & [y_1,y_2,y_3] = \frac{[y_1,y_2] - [y_2,y_3]}{x_1 - x_3} & \cdots & \\
x_2 & y_2 & [y_2,y_3] = \frac{y_2 - y_3}{x_2 - x_3} & [y_2,y_3,y_4] = \frac{[y_2,y_3] - [y_3,y_4]}{x_2 - x_4} & \cdots & \\
\cdots & \cdots & \cdots & \cdots & \cdots & \\
x_{n-1} & y_{n-1} & [y_{n-1},y_n] = \frac{y_{n-1} - y_n}{x_{n-1} - x_n} & & & \\
x_n & y_n & & & &
\end{array}$$

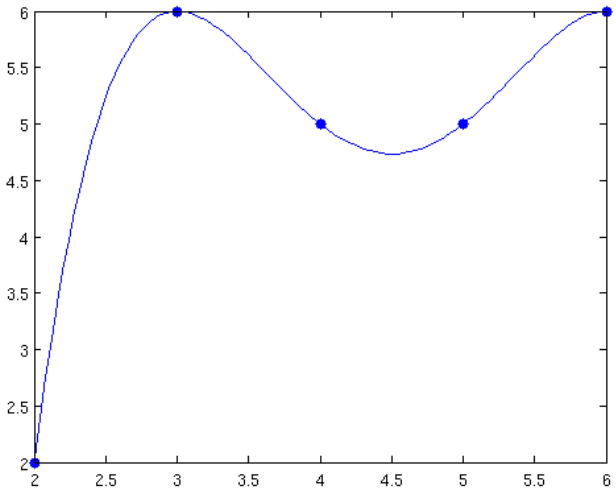
Con este método, si añadimos un nodo, no hay que rehacer todos los cálculos, sino que se añade una línea más a la tabla anterior en su parte inferior.

Ejercicio 2 Escribir una función `a=difdiv(x,y)` que calcule la matriz `a` que contiene la tabla de diferencias divididas de Newton para los puntos contenidos en `x`, `y`. No almacenar `x` en `a`. Utilizar la matriz `a` para calcular el polinomio interpolante en la forma de Newton. Dibujar el polinomio interpolante y los puntos.

```
x=[2,3,4,5,6];
y=[2,6,5,5,6];
a=difdiv(x,y)
```

```
a =
    2.0000    4.0000   -2.5000    1.0000   -0.2500
    6.0000   -1.0000    0.5000         0         0
    5.0000         0    0.5000         0         0
    5.0000    1.0000         0         0         0
    6.0000         0         0         0         0
```

```
xx=linspace(min(x),max(x));
yy=pol_newton(x,a,xx);
plot(x,y,'.','markersize',10)
hold on,plot(xx,yy)
```



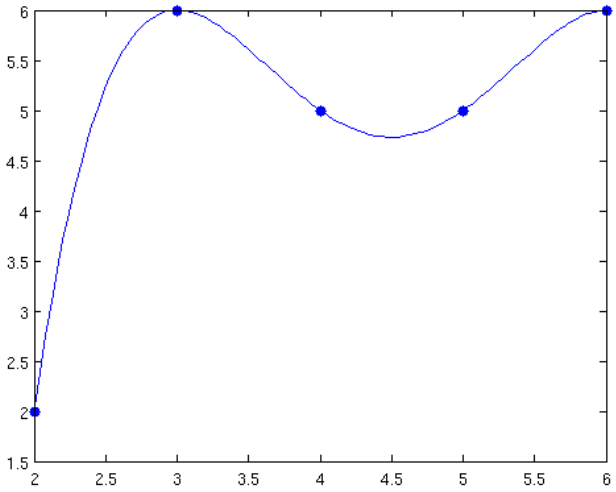
Interpolación con órdenes Matlab

El polinomio de interpolación se puede obtener con el comando de Matlab `polyfit` escogiendo como grado del polinomio el número de puntos menos uno. Nos da los coeficientes del polinomio interpolante en un vector, empezando por el de mayor grado:

```
x=[2,3,4,5,6];
y=[2,6,5,5,6];
pol=polyfit(x,y,length(x)-1);  % coeficientes del polinomio
```

Representamos el polinomio:

```
xx=linspace(min(x),max(x));
yy=polyval(pol,xx);
plot(x,y,'.','markersize',20)
hold on,plot(xx,yy)
```



Ejercicio 3 Escribir una función `cheby(f,a,b,n)` que interpole la función `f` en el intervalo `[a,b]` utilizando `n` nodos:

1. Utilizando nodos equiespaciados incluyendo los extremos del intervalo.
2. Utilizando nodos de Chebyshev:

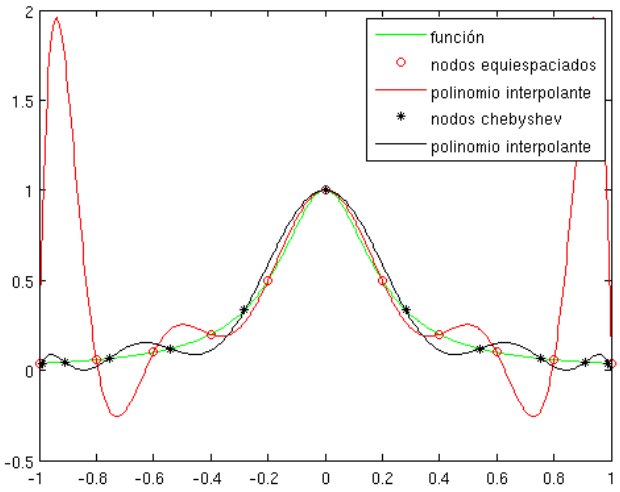
$$x_i^{(n)} = \cos \frac{(2i-1)\,\pi}{2n} \quad i = 1, 2, \ldots, n$$

En el intervalo $[-1, 1]$ con $n = 11$ nodos y la función:

$$f(x) = \frac{1}{1+25x^2}$$

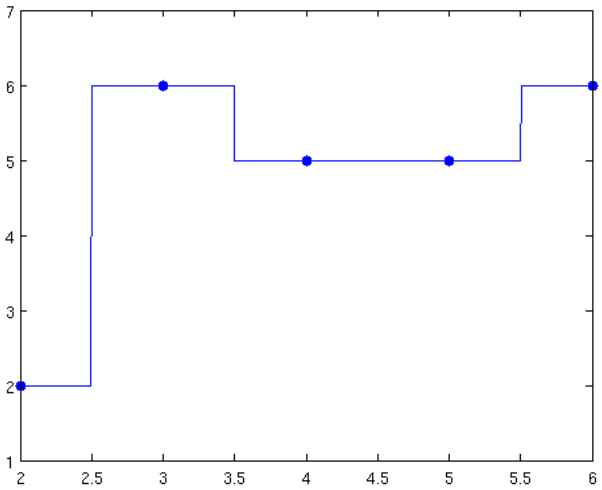
En ambos casos dibujar los nodos, la función y el polinomio interpolador.

```
f=@(x) (1)./(1+25*x.^2);
cheby(f,-1,1,11)
```



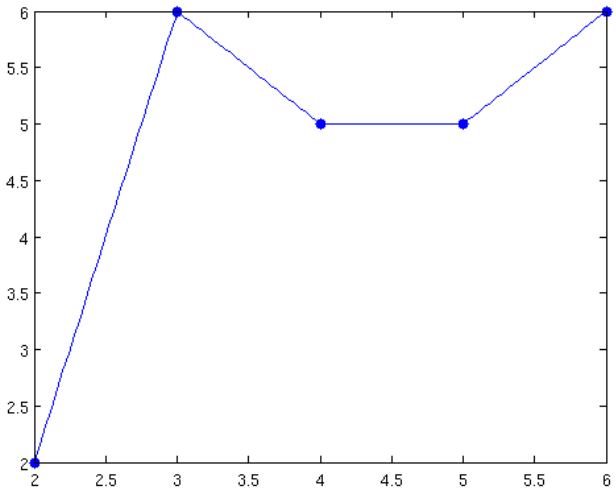
Podemos obtener la interpolación lineal a trozos con polinomios de grado cero con la orden `interp1` y la opción `nearest`

```
x=[2,3,4,5,6];
y=[2,6,5,5,6];
xx=linspace(min(x),max(x),1000);
yy=interp1(x,y,xx,'nearest');
plot(x,y,'.','markersize',20)
axis([2 6 1 7])
hold on, plot(xx,yy),hold off
```



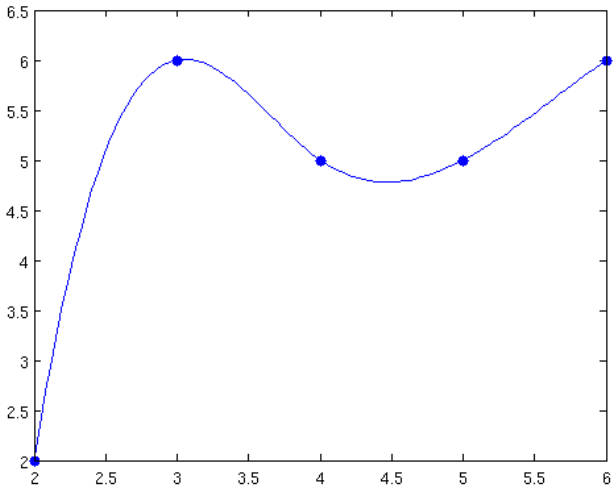
Y con polinomios de grado uno (o lo que es lo mismo, interpolación lineal a trozos) con la orden `interp1` y la opción `linear`

```
x=[2,3,4,5,6];
y=[2,6,5,5,6];
xx=linspace(min(x),max(x));
yy=interp1(x,y,xx,'linear');
plot(x,y,'.','markersize',20)
hold on, plot(xx,yy), hold off
```



Y la interpolación con splines (con la opción *not-a-knot*) con la orden

```
yy = spline(x,y,xx);
plot(x,y,'.','markersize',20)
hold on, plot(xx,yy), hold off
```



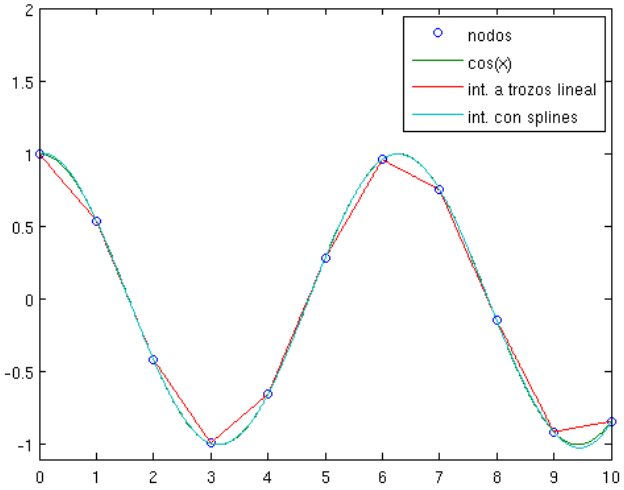
Ejercicio 4 Realizar una programa error_max que, para la función $\cos x$ en el intervalo $[0, 10]$ tomando como nodos $x = 0, 1, \ldots, 10$ dibuje los polinomios de interpolación:

1. Lineal a trozos.
2. Con splines.

y calcule, para los puntos `puntos=0:0.01:10` el error máximo en cada caso. ¿Qué interpolación da mayor error máximo?

```
error_max
```

```
Error interpolación lineal a trozos = 0.12207
Error interpolación con splines = 0.024833
```



Published with MATLAB® 7.14