4. Interpolación y ajuste de datos

4.1. Interpolación

4.1.1. Interpolación de Lagrange

El problema de interpolación de Lagrange es el siguiente: dada una tabla con $(n+1)$ puntos, hallar un polinomio del menor grado posible que los interpole. O sea, dada la tabla

x	x_1	x_2	.....	x_{n+1}
y	y_1	y_2	.....	y_{n+1}

hallar

$$p_n(x) = a_1x^n + a_2x^{n-1} + \ldots + a_nx + a_{n+1}$$

de tal manera que

$$p_n(x_i) = y_i \ \forall i = 1, \ldots, n+1. \tag{1}$$

Teorema 4.1 *Si los x_i son distintos, entonces existe un único Polinomio Interpolador de Lagrange (PIL).*

Cómo calcular el PIL.

1ª Manera: Resolver un sistema lineal. Si desarrollamos las $n+1$ ecuaciones que hay en (1), tenemos

$$\left\{\begin{array}{l} a_1x_1^n + a_2x_1^{n-1} + \cdots + a_nx_1 + a_{n+1} = y_1 \\ a_1x_2^n + a_2x_2^{n-1} + \cdots + a_nx_2 + a_{n+1} = y_2 \\ \qquad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_1x_{n+1}^n + a_2x_{n+1}^{n-1} + \cdots + a_nx_{n+1} + a_{n+1} = y_{n+1} \end{array}\right.$$

Esto es un sistema de $(n+1)$ ecuaciones con $(n+1)$ incógnitas: $a_1, \ldots, a_{n+1}$. En forma matricial es

$$\begin{pmatrix} x_1^n & x_1^{n-1} & \cdots & x_1 & 1 \\ x_2^n & x_2^{n-1} & \cdots & x_2 & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n+1}^n & x_{n+1}^{n-1} & \cdots & x_{n+1} & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \cdots \\ a_{n+1} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_{n+1} \end{pmatrix}$$

O, de manera reducida

$$Xa = y$$

X es la matriz de Vandermonde y se sabe que si $x_i \neq x_j$ para $i \neq j$, entonces

$$\det(X) = -\prod_{1 \leq i < j \leq n+1} (x_j - x_i) \neq 0,$$

luego el sistema tiene solución única.

Ejemplo 4.1 *Hallar el PIL de la siguiente tabla*

x	-1	0	1
y	1	0	0

Tenemos 3 *puntos, luego* $n = 2$. *Por tanto* $p(x) = a_1x^2 + a_2x + a_3$. *Las tres ecuaciones son*

$$\begin{cases} p(-1) = 1 \\ p(0) = 0 \\ p(1) = 0 \end{cases}$$

o de forma expandida

$$\begin{cases} a_1(-1)^2 + a_2(-1) + a_3 = 1 \\ a_1 0^2 + a_2 0 + a_3 = 0 \\ a_1 1^2 + a_2 1 a_3 = 0 \end{cases}$$

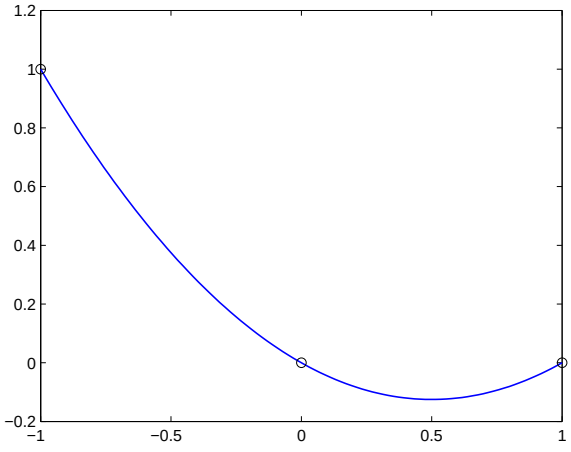
Matricialmente

$$\begin{pmatrix} 1 & -1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

De la segunda ecuación, se tiene $a_3 = 0$. *Teniendo esto en cuenta y sumando las dos primeras se tiene que* $a_1 = 1/2$. *Por tanto* $a_2 = -1/2$. *Así que*

$$p(x) = \frac{x^2}{2} - \frac{x}{2}$$

```
>> format compact
>> x=[-1, 0 ,1]';
>> y=[1, 0, 0]';
>> % Planteamos y resolvemos
>> % el sistema nosotros
>> X=[x.^2 x ones(3,1)];
>> a=X\y
a =
    0.5000
   -0.5000
         0
>> % O usamos comandos de
>> % Matlab
>> a=polyfit(x,y,2)
a =
    0.5000   -0.5000         0
>> xp=linspace(-1,1,100);
>> yp=polyval(a,xp);
>> plot(xp,yp,'b-',x,y,'ok')
```



Nota: En general, la matriz de Vandermonde puede resultar muy mal condicionada. Resolver directamente el sistema puede dar resultados muy alejados de la solución real.

2ª Manera: Usar polinomios de base Para interpolar $(n+1)$ puntos buscamos un polinomio de grado n, precisamente porque viene dado por $(n+1)$ parámetros: $a_1, \ldots, a_{n+1}$. El espacio de

todos los polinomios de grado n, $\mathbb{P}_n[x]$, se puede por tanto poner en correspondencia biyectiva con el espacio euclídeo $\mathbb{R}^{n+1}$:

$$\begin{array}{ccc} \mathbb{P}_n[x] & \leftrightarrow & \mathbb{R}^{n+1} \\ p(x) & \rightsquigarrow & (a_1, \ldots, a_{n+1}) \end{array}$$

Recordemos que $\mathbb{R}^{n+1}$ es un espacio vectorial de dimensión $(n+1)$ y que tiene una base que llamamos canónica que es la que sigue las direcciones de los ejes. Pero siempre que tengamos una colección de $(n+1)$ vectores linealmente independientes tendremos una base $\vec{v}_1, \ldots, \vec{v}_{n+1}$, que nos permite construir cualquier vector como combinación lineal: $\vec{v} = \alpha_1 \vec{v}_1 + \cdots + \alpha_{n+1} \vec{v}_{n+1}$.

Gracias a esa correspondencia biyectiva entre los polinomios y el espacio euclídeo, $\mathbb{P}_n[x]$ también tiene estructura de espacio vectorial de dimensión $(n+1)$. Ahora los elementos del espacio no son vectores, sino polinomios, pero podemos aplicar los mismos trucos. Vamos a buscar $(n+1)$ polinomios $\ell_1(x), \ldots, \ell_{n+1}(x)$ linealmente independientes. Estos formarán una base, y así podremos construir cualquier polinomio –en concreto el PIL– como una combinación adecuada de ellos. En el espacio euclídeo, la elección más natural es usar los ejes. Ahora, una elección que parece bastante natural es la siguiente:

Para $j = 1, \ldots, n+1$, sea $\ell_j(x)$ el polinomio que interpola la tabla

x	x_1	x_2	$\ldots$	x_j	$\ldots$	x_{n+1}
y	0	0	0	1	0	0

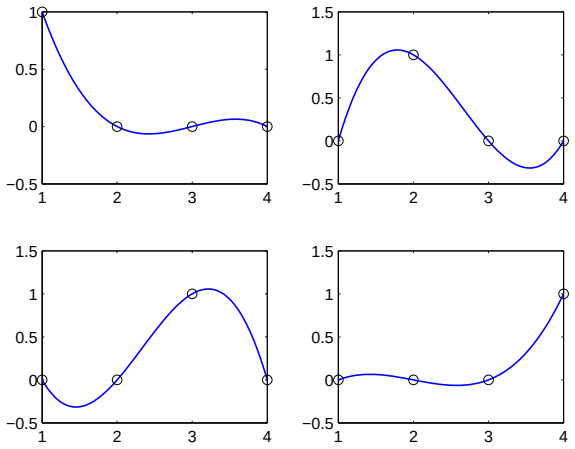
Así que

$$\ell_j(x_i) = \delta_{i,j} = \begin{cases} 0 & \text{si} \quad i \neq j \\ 1 & \text{si} \quad i = j \end{cases}$$

Hay una expresión para $\ell_j(x)$:

$$\ell_j(x) = \prod_{\substack{i=1 \\ i \neq j}}^{N} \left(\frac{x - x_i}{x_j - x_i} \right)$$

Por ejemplo, los polinomios de base para los nodos $x = 1, 2, 3, 4$ son



Una vez conocidos los $\ell_j(x)$ es trivial que el PIL de la tabla

x	x_1	x_2	.....	x_{n+1}
y	y_1	y_2	.....	y_{n+1}

es

$$p(x) = y_1\ell_1(x) + y_2\ell_2(x) + \cdots + y_{n+1}\ell_{n+1}(x)$$

Otras maneras de calcularlo: Existen otras técnicas mejores: fórmulas de Newton, diferencias divididas, ...

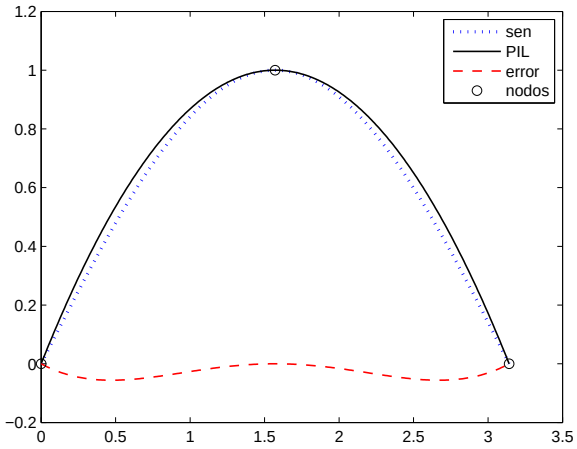
Estudio del error: Calcular el PIL requiere por lo general métodos directos, por lo que en teoría siempre encontramos la solución correcta. Los errores debidos al redondeo no son demasiado importantes a no ser que haya nodos que estén muy próximos entre sí, en cuyo caso, como ya hemos indicado, la matriz de Vandermonde está mal condicionada.

Si los datos y_i se obtienen como valores de una función $f(x)$, i.e. $y_i = f(x_i)$, entonces seguramente estaremos cometiendo un error al asignar el valor $p(x)$ a la función fuera de los nodos.

x	x_1	x_2	.....	x_{n+1}
y	$f(x_1)$	$f(x_2)$	.....	$f(x_{n+1})$

$$f(x) = p_n(x) + E_n(x),$$

siendo $E_n(x) = f(x) - p_n(x)$ el error cometido, que en principio es desconocido.



Teorema 4.2 *Si $f \in \mathcal{C}^{n+1}[a,b]$, y todos los $x_i \in [a,b]$, entonces existe $c \in [a,b]$ tal que*

$$E_n(x) = \frac{1}{(n+1)!}\prod_{i=1}^{n+1}(x - x_i)\, f^{(n+1)}(c).$$

Este teorema no nos permite encontrar el error de manera exacta, ya que es un resultado de existencia. Nos dice que existe un c pero no nos dice como calcularlo. De todas maneras, nos va a permitir acotar el error. Por un lado, la derivada $(n+1)$ de f tiene que ser continua para

aplicar el teorema y además el intervalo es compacto. Por el Teorema de Weierstrass, se tiene que existe $M_{n+1} > 0$ tal que

$$|f^{(n+1)}(c)| \leq M_{n+1}.$$

Así que el máximo error que vamos a cometer está relacionado con el

$$\max_{x\in[a,b]} \left| \prod_{i=1}^{n+1} (x - x_i) \right|$$

Veamos un caso particular: nodos equiespaciados. Sean $x_1 = a$, $h = (b-a)/n$ y $x_i = a+(i-1)h$ para $i = 1 : n+1$. Se tiene que

$$|E_1(x)| \leq \frac{h^2}{8} M_2$$

$$|E_2(x)| \leq \frac{h^3}{9\sqrt{3}} M_3$$

$$|E_3(x)| \leq \frac{h^4}{24} M_4$$

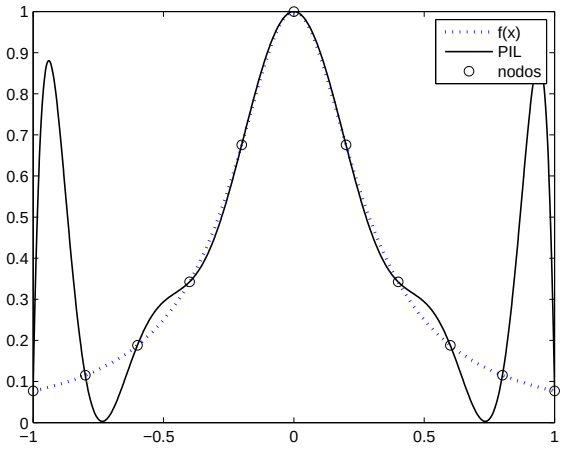
Ejemplo 4.2 *Para una función como* $\operatorname{sen}(x)$ *sabemos que* $M_{n+1} \leq 1$ *siempre, por tanto, cuántos más nodos utilicemos, menor será el error.*

Pero en general, M_{n+1} va a crecer más rápido de lo que decrece h^n y aparece el fenómeno de **oscilación polinomial**.

Ejemplo 4.3 *Interpolemos la función de Runge*

$$f(x) = \frac{1}{1 + 12x^2}$$

con 11 nodos equiespaciados en $[-1, 1]$. *El resultado es el siguiente*



Este efecto hace que la interpolación de Lagrange sea una técnica apenas usada para problemas reales de ajuste de curvas. Su principal aplicación hoy en día es la derivación de otro tipo de métodos de ajuste e interpolación.

Se abre ahora un abanico de posibilidades.

1. Cambiar los nodos: buscar nodos no equiespaciados que hagan más pequeña la cantidad máx $\left|\prod_{i=1}^{n+1}(x-x_i)\right|$ u otra que pudiera resultar de interés.

   Esto normalmente lo podremos hacer cuando la función sea conocida (por ejemplo, para integrar) o tengamos la oportunidad de elegir en que nodos queremos evaluar la función (por ejemplo, muestreo de algún tipo de señal).

2. No usar polinomios de grado muy alto:

   a) Interpolación polinomial a trozos.

   b) Método de mínimos cuadrados.

3. Usar otro tipo de funciones

   - Funciones trigonométricas: análisis de Fourier.
   - Wavelets
   - Otras funciones: mínimos cuadrados no lineales.
   - ...

Una posible elección de nodos: nodos de Chebyshev. Los nodos de Chebyshev $\hat{x}_1, \ldots, \hat{x}_{n+1}$ son aquellos que hacen mínima la cantidad

$$\max_{-1\leq x\leq 1}\left|\prod_{i=1}^{n+1}(x-\hat{x}_i)\right|$$

(resuelven un problema de tipo *minimax*). Estos nodos son las raíces del $(n+1)$-ésimo polinomio de Chebyshev, que se define de manera recurrente como

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_k(x) &= 2xT_{k-1}(x) - T_{k-2}(x) \text{ para } k \geq 2 \end{aligned}$$

Se demuestra que

$$T_{n+1}(x) = \cos\left((n+1)\arc\cos(x)\right)$$

y por tanto

$$\hat{x}_i = \cos\left(\frac{(2i-1)\pi}{2(n+1)}\right),\ i = 1:n+1$$

Por ejemplo, para 11 nodos ($n = 10$) los nodos de Chebyshev son

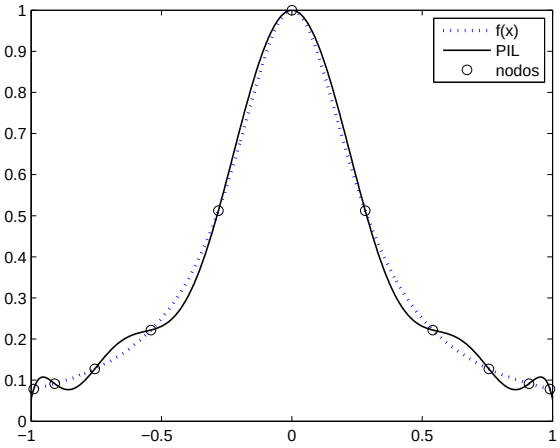
$$\hat{x} = [-0.9898,\ -0.9096,\ -0.7557,\ -0.5406,\ -0.2817,\ \ldots$$

$$\ldots,\ 0.0000,\ 0.2817,\ 0.5406,\ 0.7557,\ 0.9096,\ 0.9898]$$

Para encontrar los nodos en un intervalo $[a, b]$, simplemente aplicamos el cambio de variable

$$x = \frac{a+b}{2} + \frac{b-a}{2}\hat{x}$$

Interpolemos de nuevo $f(x) = 1/(1+12x^2)$ en $[-1, 1]$ con los nodos de Chebyshev:



4.1.2. Interpolación a trozos.

Dada la tabla

x	x_1	x_2	.....	x_{n+1}
y	y_1	y_2	.....	y_{n+1}

(en la que suponemos $x_1 < x_2 < \ldots < x_{n+1}$) escogeremos $m < n$ tal que $n = km$ para algún $k \in \mathbb{N}$ y la subdividiremos en k tablas

x	x_1	...	x_{m+1}
y	y_1	...	y_{m+1}

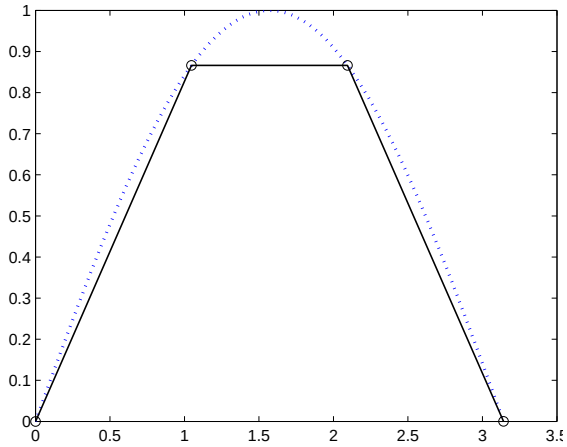
x	x_{m+1}	...	x_{2m+1}
y	y_{m+1}	...	y_{2m+1}

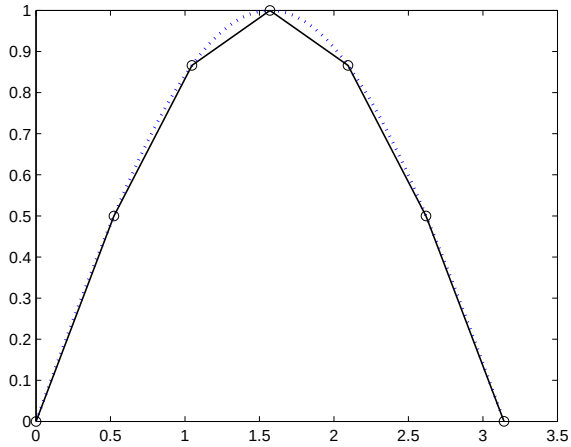
...

x	$x_{(k-1)m+1}$	...	x_{km+1}
y	$y_{(k-1)m+1}$	...	y_{km+1}

y buscaremos para cada una de estas tablas un polinomio de grado m. Finalmente tendremos una función definida en k trozos.

Interpolación lineal a trozos: Si escogemos $m = 1$, estaremos buscando una línea poligonal que una los puntos 2 a 2. Esta opción es rápida, sencilla, puede manejar muchos puntos y es fácilmente generalizable a dimensiones mayores. Es lo que en ingeniería se llama "mirar en un tabla". Por otro lado, es visualmente pobre, da curvas no derivables y con pocos puntos los resultados son poco fiables.





En el caso de estar aproximando una función $f \in C^2([a,b])$, cuando aumentamos el número de puntos, los resultados mejoran bastante rápido. Recordemos que el error máximo en cada subintervalo $[x_i, x_{i+1}]$ es del orden de h^2, siendo $h = x_{i+1} - x_i$. Como $h = 1/n$, tenemos que, llamando $P_1(f)$ a la función interpolante, el error máximo cometido es:

$$E_\infty = \text{máx}\{|f(x) - P_1(f(x))| : x \in [a,b]\} = O(h^2) = O(1/n^2)$$

Esto quiere decir que con (aproximadamente) el doble de puntos (exactamente, al pasar de $n+1$ puntos a $2n+1$ puntos) el error se divide (aproximadamente) por 4. En el ejemplo anterior, el error con 4 puntos es 0.134 y el error con 7 puntos es 0.0329. El factor de reducción es $0.134/0.0329 = 4.07$. Tenemos la siguiente tabla de error

n	3	6	12	24
E_∞	1.34e-1	3.29e-2	8.48e-3	2.13e-3
FR	-	4.07	3.88	3.98

Splines: Es otro tipo de interpolación polinomial a trozos, donde en cada trozo no vamos a buscar el PIL, sino que vamos a imponer más condiciones. Dada la tabla (ordenada)

x	x_1	x_2	.....	x_{n+1}
y	y_1	y_2	.....	y_{n+1}

buscamos una función $s(x) : [x_1, x_{n+1}] \to \mathbb{R}$ que cumpla

1. $s(x)$ restringido al subintervalo $[x_k, x_{k+1}]$ es un polinomio de grado 3, que llamaremos $s_k(x)$ para $k = 1 : n$.

$$s_k(x) = a_1^k x^3 + a_2^k x^2 + a_3^k x + a_4^k \text{ para } k = 1 : n$$

2. $s(x_k) = y_k$ para $k = 1 : n+1$ (INTERPOLACIÓN)
3. $s_k(x_{k+1}) = s_{k+1}(x_{k+1})$ para $k = 1 : n-1$ (CONTINUIDAD)
4. $s_k'(x_{k+1}) = s_{k+1}'(x_{k+1})$ para $k = 1 : n-1$ (DERIVABILIDAD)
5. $s_k''(x_{k+1}) = s_{k+1}''(x_{k+1})$ para $k = 1 : n-1$ (CURVATURA)

La primera condición nos dice que tenemos que determinar $4n$ parámetros, pero sólo hemos impuesto $4n-2$ condiciones. Dependiendo de como añadamos las dos condiciones que quedan, tendremos un tipo de spline u otro. Entre otras condiciones, podemos destacar:

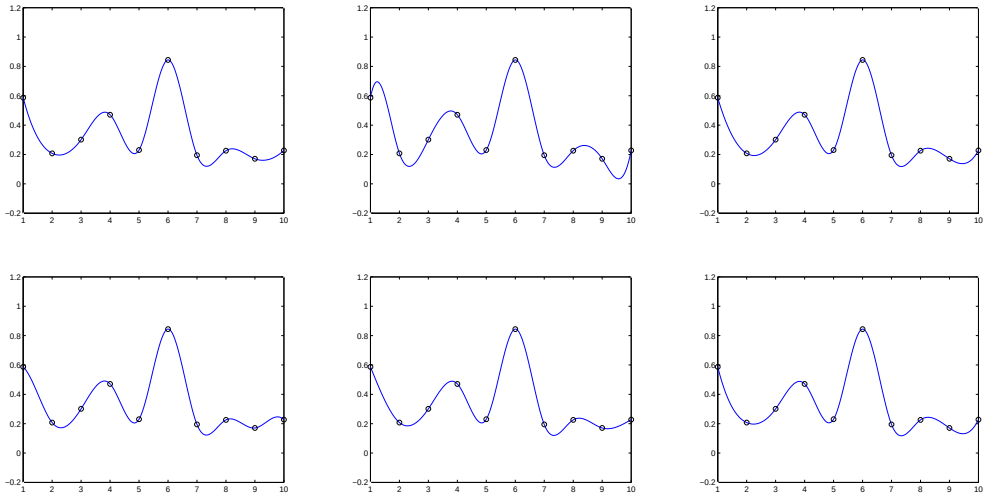
0. El que viene por defecto en MATLAB: $s_1'(x_1) = d_1$, $s_n'(x_{n+1}) = d_{n+1}$. (d_1 y d_2 los escoge MATLAB según las cúbicas que interpolan los 4 nodos de cada extremo).
1. Sujeto o completo: $s_1'(x_1) = d_1$, $s_n'(x_{n+1}) = d_{n+1}$. (d_1 y d_2 los escogemos).
2. Extrapolado (eliminar un nodo, not-a-knot): $s_1'''(x_2) = s_2'''(x_2)$, $s_{n-1}'''(x_n) = s_n'''(x_n)$.

3. Periódico: $s_1'(x_1) = s_n'(x_{n+1})$, $s_1''(x_1) = s_n''(x_{n+1})$.

4. Natural o variacional: $s''(x_1) = 0$, $s''(x_{n+1}) = 0$.

5. Imponer una condición sobre las derivadas segundas: $s''(x_1) = s_1$, $s''(x_{n+1}) = s_{n+1}$.

Ejemplo 4.4 *Se muestran a continuación los 6 tipos de splines descritos para el mismo conjunto de datos.*



Ejemplo 4.5 *El spline completo $s(x)$ que interpola a $f(x) = 1/(1+12x^2)$ con 11 nodos equiespaciados en $[-1, 1]$ es visualmente indistinguible de la misma función. De hecho*

$$\max_{-1\le x\le 1} |f(x) - s(x)| \simeq 3 \times 10^{-3} \qquad \max_{-1\le x\le 1} \frac{|f(x) - s(x)|}{|f(x)|} \simeq 6 \times 10^{-3}.$$

Nota: Si cambiamos la condición 5 por las siguiente:

5'. $s_k'(x_{k+1}) = d_{k+1}$ para $k = 1 : n - 1$

(donde los valores d_{k+1} de las pendientes en los nodos interiores son fijos), obtenemos la llamada interpolación cúbica de Hermite a trozos (`pchip` en MATLAB). Con esta técnica, perdemos la continuidad de la curvatura, pero a cambio ganamos, con una *adecuada elección* de los valores de las pendientes, una cierta "conservación de la monotonía". Esto quiere decir que si $y_k < y_{k+1}$, entonces $s_k(x)$ es creciente, y viceversa: si $y_k > y_{k+1}$, entonces $s_k(x)$ es decreciente. Los splines no cumplen esta propiedad en general.

4.2. Método de los mínimos cuadrados (Least Squares Method)

Supongamos que queremos determinar una función $y = f(x)$ que modelice de manera adecuada algún fenómeno, del que disponemos de m observaciones:

x	x_1	x_2	.....	x_m
y	y_1	y_2	.....	y_m

La función f la buscaremos como combinación lineal de un conjunto de n de funciones conocidas, y en principio "sencillas" de manejar (polinomios, trigonométricas, exponenciales, ...). Llamemos por ahora a estas funciones $\phi_1(x), \ldots, \phi_n$. La función f será de la forma:

$$f(x) = a_1\phi_1(x) + \cdots + a_n\phi_n(x)$$

Así que si queremos que se cumpla que $y = f(x)$ para cada uno de los datos de la tabla, queremos que se cumplan la siguiente ecuación para todo $i = 1 : m$:

$$\phi_1(x_i)a_1 + \ldots + \phi_n(x_i)a_n = y_i$$

Si llamamos $x_{i,j} = \phi_j(x_i)$ para $1 \leq i \leq m$ y $1 \leq j \leq n$ y denotamos X la matriz de diseño $(x_{i,j})$, $a = (a_1, \ldots, a_n)^T$ el vector de incógnitas e $y = (y_1, \ldots, y_m)^T$ el vector de datos, tenemos el siguiente sistema de ecuaciones

$$Xa = y$$

El caso más general en las aplicaciones es que $m > n$ (tenemos muchos datos y una familia con unas pocas funciones). Supondremos además que X tiene rango máximo ($rg(X) = n$) y que $rg([X, y]) = n + 1$. (Normalmente los valores y_i vienen de algún experimento. El error experimental hace que esta hipótesis se satisfaga de manera natural en problemas aplicados.) Así las cosas, el sistema $Xa = y$ es incompatible.

Ejemplo 4.6 *Recta de regresión: en este caso $\phi_1(x) = x$ y $\phi_2(x) = 1$. Buscamos una recta de la forma $y = a_1x + a_2$ de tal manera que*

$$y_i = a_1x_i + a_2 \;\forall\, i = 1 : m$$

Matricialmente, este sistema es de la forma

$$\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_m & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

Ejemplo 4.7 *Un muelle está colgado verticalmente dentro de un cilindro abierto por los extremos a altura 0, por encima de la apertura superior. La longitud $\ell > 0$ del muelle es desconocida porque su extremo inferior está dentro del cilindro y es inalcanzable. Se desea determinar ℓ, y de paso la constante del muelle k usando la ley de Hooke:*

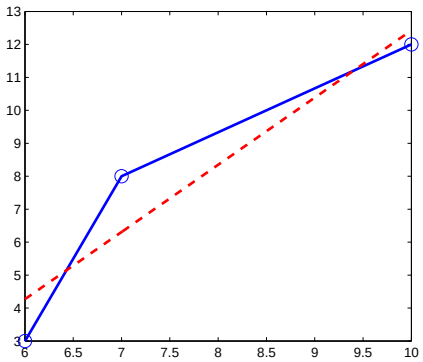
$$F = k(h - \ell) = kh - k\ell$$

donde F es la fuerza aplicada al muelle (podemos colgar una pesa en el extremo inferior lanzándola llena de pegamento, por ejemplo) y h es la distancia desde el extremo superior del muelle.

Se dispone de los siguienes datos

h	6	7	10
F	3	8	12

Obviamente se ha cometido algún error experimental, ya que esos datos no están en línea recta.



Sean $a_1 = k$ y $a_2 = -k\ell$. El sistema resultante de escribir la ley de Hooke para esos datos es incompatible:

$$\begin{cases} 6a_1 + a_2 &= 3 \\ 7a_1 + a_2 &= 8 \\ 10a_1 + a_2 &= 12 \end{cases}$$

Cuando el sistema es incompatible, da igual como escojamos los parámetros $a_1, \ldots, a_n$, nunca podremos hacer que todas las componentes del vector residuo $r = Xa - y$,

$$r_i = -y_i + \sum_{j=1}^{n} x_{i,j} a_j, \ i = 1 : m,$$

sean cero simultáneamente. En lugar de eso, vamos a intentar que los residuos sean lo más "pequeños" posible. Hay muchas maneras de entender "pequeño", sobre todo cuando hay que medir varias cosas a la vez como es nuestro caso. Nosotros vamos a buscar coeficientes $(a_1, \ldots, a_n)^T$ que hagan mínima la suma de los cuadrados de los residuos:

$$\text{mín} \sum_{i=1}^{m} r_i^2$$

Esta cantidad es el cuadrado de la norma euclídea del vector residuo, así que formulamos el problema de mínimos cuadrados como un problema de optimización en $\mathbb{R}^n$:

$$\text{Encontrar } a = (a_1, \ldots, a_n)^T \text{ tal que se haga mínima } \|Xa - y\|^2$$

Hay varias maneras de resolver este problema de optimización: se pueden escribir las derivadas parciales e igualar a cero, se pueden aprovechar las propiedades de los espacios con producto escalar, se pueden escribir las ecuaciones de Euler para problemas de optimización en espacios normados, ... De cualquiera de las maneras, se llega a que el vector de incógnitas a satisface el siguiente sistema lineal de ecuaciones compatible y determinado que se llaman ecuaciones normales o ecuaciones de Gauss:

$$X^T X a = X^T y$$

Aunque es muy tentador resolver este sistema (es compatible determinado, es de tamaño n, que se supone que no es muy grande, la matriz es simétrica y definida positiva), numéricamente es un sistema peor, ya que

$$\text{cond}(X^T X) = \text{cond}(X)^2$$

y es un sistema peor condicionado que el original.

Ejemplo 4.8 *Continuando con el ejemplo del muelle, la matriz de diseño y el vector de datos son*

$$X = \begin{pmatrix} 6 & 1 \\ 7 & 1 \\ 10 & 1 \end{pmatrix}, \; y = \begin{pmatrix} 3 \\ 8 \\ 12 \end{pmatrix}$$

con lo que las ecuaciones normales son de la forma

$$\begin{pmatrix} 185 & 23 \\ 23 & 3 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 194 \\ 23 \end{pmatrix}$$

Se tiene que $\mathrm{cond}(X) = 36.84$*, pero* $\mathrm{cond}(X^T X) = 1.357 \times 10^3$*.*

Para hacer esto de manera conveniente con un ordenador, se utiliza la llamada descomposición QR de la matriz X. Existen matrices $Q \in \mathbb{R}^{m \times m}$ y $R \in \mathbb{R}^{m \times n}$ donde Q es ortogonal ($Q^{-1} = Q^T$), y R es triangular superior y $\mathrm{rg}(R) = n$ tales que

$$X = QR$$

Así

$$X^T X a = X^T y \iff (QR)^T (QR) a = (QR)^T y \iff R^T Q^T Q R a = R^T Q^T y$$

Como $Q^T = Q^{-1}$, se tiene que $Q^T Q R = R$.

$$R^T R a = R^T Q^T y$$

Como R^T es triangular inferior, las últimas $m - n$ ecuaciones ahora son de la forma $0 = 0$. Ya tenemos un sistema cuadrado. Como $\mathrm{rg}(R^T) = \mathrm{rg}(R) = n$ podemos simplificar R^T por la izquierda de la ecuación. Así que nos queda el sistema

$$Ra = Q^T y$$

donde sólo nos hemos quedado con las primeras n ecuaciones a cada lado:

$$R(1:n, :)a = Q(:, 1:n)^T y$$

Este es un sistema triangular superior. Además $\mathrm{cond}(R) = \mathrm{cond}(A)$. Para calcular Q se utilizan las matrices llamadas reflectores de Householder. En la práctica, los algoritmos para resolver sistemas lineales mediante mínimos cuadrados no necesitan calcular toda la matriz Q, sino solamente sus n primeras columnas, e incluso no es necesario calcular estas explícitamente (análogamente a lo que ocurre con la matriz L de la descomposición LU y el método de Gauss.)

Ejemplo 4.9 *La matriz del sistema del muelle y su descomposición QR son*

$$X = \begin{pmatrix} 6 & 1 \\ 7 & 1 \\ 10 & 1 \end{pmatrix} = \begin{pmatrix} -0.4411 & -0.6777 & -0.5883 \\ -0.5147 & -0.3460 & 0.7845 \\ -0.7352 & 0.6488 & -0.1961 \end{pmatrix} \begin{pmatrix} -13.6015 & -1.6910 \\ 0 & -0.3749 \\ 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} -0.4411 & -0.6777 \\ -0.5147 & -0.3460 \\ -0.7352 & 0.6488 \end{pmatrix} \begin{pmatrix} -13.6015 & -1.6910 \\ 0 & -0.3749 \end{pmatrix}$$

El sistema triangular a resolver es

$$\begin{pmatrix} -13.6015 & -1.6910 \\ 0 & -0.3749 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} -14.2632 \\ 2.9847 \end{pmatrix}$$

4.3. Aproximación trigonométrica

Nos planteamos ahora aproximar una función $f : \mathbb{R} \to \mathbb{R}$. Supondremos que la función f cumple que existe $T > 0$:

$$f(t+T) = f(t) \ \forall\, t \in \mathbb{R}$$

Vamos a aproximarla mediante una combinación lineal de funciones trigonométricas:

$$f(t) \approx s(t) = \frac{a_0}{2} + \sum_{j=1}^{N} (a_j \cos(2\pi jt/T) + b_j \operatorname{sen}(2\pi jt/T)),$$

siendo $N \in \mathbb{N}$, a_0, (a_j) y (b_j) coeficientes reales adecuados. La aproximación la entenderemos en el sentido de los mínimos cuadrados. Al no tratarse de un conjunto discreto de datos, la norma no la podemos tomar en $\mathbb{R}^m$, como en la sección anterior. Manejaremos la norma análoga en el espacio de las funciones. Buscamos coeficientes que hagan mínima la expresión

$$\int_0^T \left(f(t) - \frac{a_0}{2} - \sum_{j=1}^{N} (a_j \cos(2\pi jt/T) + b_j \operatorname{sen}(2\pi jt/T)) \right)^2 dt$$

La solución a este problema viene dada por los llamados coeficientes de Fourier:

$$a_j = \frac{2}{T} \int_0^T f(t) \cos(2\pi jt/T) dt$$

$$b_j = \frac{2}{T} \int_0^T f(t) \operatorname{sen}(2\pi jt/T) dt$$

para todo $j \geq 0$ (definimos $b_0 = 0$ por conveniencia para el desarrollo posterior).

El cálculo de los coeficientes requiere el conocimiento de la función y la evaluación de las integrales. Veamos como realizarlo numéricamente de manera sencilla. Como pasa muchas veces en matemáticas, el camino más corto para resolver un problema de números reales, pasa por los números complejos.

Utilizando la fórmula $e^{i\theta} = \cos\theta + i \operatorname{sen}\theta$ y definiendo para $j \geq 0$

$$c_j = (a_j - ib_j)/2, \quad \text{y} \quad c_{-j} = (a_j + ib_j)/2$$

se tiene que

$$f(t) \approx s(t) = \sum_{j=-N}^{N} c_j e^{2\pi ijt/T}.$$

Además se tienen las siguientes fórmulas. Para $j \geq 0$

$$a_j = c_j + c_{-j}, \quad \text{y} \quad b_j = i(c_j - c_{-j})$$

y para todo $j \in \mathbb{Z}$:

$$c_j = \frac{1}{T} \int_0^T f(t) e^{-2\pi ijt/T} dt \tag{2}$$

Para calcular una aproximación de esa integral, subdividimos el intervalo $[0,T]$ en n partes iguales de tamaño $h = T/n$ y tomamos los puntos $t_k = kh$ para $k = 0 : n-1$ (como f es periódica $f(T) = f(0)$ y no estamos perdiendo información por no llegar hasta el final del intervalo). Una de las maneras más sencillas de aproximar la integral, es cambiar la función en cada subintervalo por un valor constante. (Si tuviéramos una función que tomase valores reales positivos, estaríamos cambiando el cálculo del área bajo la curva por el cálculo del área de un rectángulo: ¡base por altura! La base es h, la altura, el valor constante elegido.) El valor más adecuado suele ser el del punto medio, pero en este caso, gracias a la periodicidad, podemos tomar el valor que se toma en el extremo izquierdo del subintervalo ($t_k = kh$). Se tiene así que

$$c_j \approx c_j^n = \frac{1}{T}\sum_{k=0}^{n-1} h f(t_k) e^{-2\pi i j k h/T}$$

Teniendo en cuenta que $h = T/n$ y denotando $y_k = f(t_k)$ y $\omega = e^{-2\pi i/n}$, se tiene para $j \in \mathbb{Z}$

$$c_j^n = \frac{1}{n}\sum_{k=0}^{n-1} f(t_k) e^{-2\pi i j k/n} = \frac{1}{n}\sum_{k=0}^{n-1} \omega^{jk} y_k$$

Nótese ahora que $\omega^j = \omega^{j+n}$ para todo $j \in \mathbb{Z}$, por lo que ¡sólo tengo n valores diferentes! Las aproximaciones $c_j \approx c_j^n$ son válidas solamente para $-n/2 \le j < n/2$.

Aprovechamos de nuevo que $\omega^j = \omega^{j+n}$ para "reordenar" los cálculos e intentar escribirlos de forma matricial: para $-n/2 \le j < 0$ usaremos la fórmula correspondiente a $n + j$. Además, multiplicamos toda la ecuación por n, con lo cual en vez de calcular c_j^n para $-n/2 \le j < n$, calcularemos para $j = 0 : n-1$

$$Y_j = \begin{cases} nc_j^n & \text{si} \quad 0 \le j < n/2 \\ nc_{j-n}^n & \text{si} \quad n/2 \le j < n-1 \end{cases}$$

Esquemáticamente para $n = 7$ y $n = 6$ la correspondencia sería

$$\begin{array}{ccccccc} Y_0 & Y_1 & Y_2 & Y_3 & Y_4 & Y_5 & Y_6 \\ 7c_0^7 & 7c_1^7 & 7c_2^7 & 7c_3^7 & 7c_{-3}^7 & 7c_{-2}^7 & 7c_{-1}^7 \end{array}$$

$$\begin{array}{cccccc} Y_0 & Y_1 & Y_2 & Y_3 & Y_4 & Y_5 \\ 6c_0^6 & 6c_1^6 & 6c_2^6 & 6c_{-3}^6 & 6c_{-2}^6 & 6c_{-1}^6 \end{array}$$

Así las cosas, definiendo la matriz F de tamaño n como

$$F_{j+1,k+1} = \omega^{jk}$$

para $0 \le j, k \le n-1$, se tiene la fórmula

$$Y = Fy$$

Esta transformación se llama transformada discreta de Fourier del vector y (*Discrete Fourier Transform, DFT*).

Dándole la vuelta a las relaciones entre Y y c, tenemos que

$$c_j^n = \begin{cases} Y_j/n & \text{si} \quad 0 \le j < n/2 \\ Y_{j+n}/n & \text{si} \quad -n/2 \le j \le -1 \end{cases}$$

Cuando hacemos el análisis de una señal, tanto o más interesante que el cálculo en sí de los coeficientes a, b, c o Y, es detectar las frecuencias presentes en la señal. Si una frecuencia

$$f = \frac{j}{T} \tag{3}$$

está presente, entonces a_j o b_j serán distintos de cero. Luego ambos $|c_j|^2 = |c_{-j}|^2 = (a_j^2+b_j^2)/4 \neq 0$. Esto se detecta al hacer la transformada discreta de Fourier de la señal porque ambos $|Y_j|^2/n^2 = |Y_{n-j}|^2/n^2 \neq 0$. (Nótese que la mitad del resultado de la transformada discreta "sobra".) Nótese también que tomando n puntos, solamente podremos detectar frecuencias estrictamente menores que la frecuencia de Nyquist $n/(2T)$.

La cantidad $|Y_j|^2/n^2$ se llama potencia de la frecuencia correspondiente, y la cantidad Y^HY/n^2 es la potencia de la señal. Se tiene que

$$\frac{1}{T}\int_0^T f^2(t)dt = \lim_{n\to\infty} \frac{Y^HY}{n^2}$$

Teniendo en cuenta que la primera fila y la primera columna de F están formada por unos, el coste computacional de calcular la DFT es $2n^2 - 3n + 1$ operaciones con complejos (sumas y multiplicaciones). Aprovechando el hecho de que $e^{-2\pi i/n} = \left(e^{-2\pi i/(2n)}\right)^2$, James W. Cooley y John W. Tuckey crearon en 1965 un algoritmo que calculaba la transformada para un vector con un número par $n = 2m$ de términos combinando las transformadas para las componentes que ocupaban los lugares impares y pares respectivamente del vector original, que tienen tamaño m:

$$\begin{array}{l} U = DFT(y_{impar}) \\ V = DFT(y_{par}) \\ Y_k = U_k + \omega^k V_k \text{ para } k = 0 : n/2 - 1 \\ Y_{n/2+k} = U_k - \omega^k V_k \text{ para } k = 0 : n/2 - 1 \end{array}$$

Ojo a la extraña notación:

$$y_{impar} = (y_0, y_2, \ldots, y_{n-1})^T \; y_{par} = (y_1, y_3, \ldots, y_{n-1})$$

El coste de este procedimiento es aproximadamente la mitad del original y el algoritmo se conoce como **Transformada Rápida de Fourier** (*Fast Fourier Transform, FFT*). Si $n = 2^p$, se puede repetir el truco subdividiendo hasta tener que hacer transformadas de vectores de tamaño 1, con lo cual sólo hay que tener en cuenta el coste de las recombinaciones. El coste computacional de la FFT, medido en sumas y multiplicaciones complejas es

$$\frac{3n}{2}\log_2 n - \frac{n}{2} + 1$$

Versiones modernas y refinadas de la FFT aceleran el cálculo de la DFT para cualquier n.

La cantidad

$$Fs = \frac{n}{T} = \frac{1}{h}$$

se llama frecuencia de muestreo (*sampling frequency*). En la práctica, esta es la cantidad importante. Nos dice el número de muestras que tomamos por segundo. En muchas aplicaciones, el periodo de la función es desconocido, pero se sabe su orden de magnitud, que puede ser muy pequeño (el T que corresponde a la corriente eléctrica doméstica es $1/50$; el de las ondas de televisión está entre $1/(3 \times 10^8)$ y $1/(3 \times 10^7)$; el que corresponde al espectro audible, entre $1/20000$ y $1/20$.) Así que se muestrea la señal durante un tiempo T suficientemente grande, que puede cumplir la condición de periodicidad $f(t) = f(t+T)$ o no cumplirla, y en las fórmulas se usa que

$$T = \frac{n}{Fs} \tag{4}$$

para reemplazar T.

Ejemplo 4.10 *En el siguiente ejemplo se muestrea una señal de periodo conocido en un número suficiente de puntos y se detecta su frecuencia.*

```
% Ejemplo 1:
% Periodo conocido
% No usamos la frecuencia de muestreo
T=1/50;
n=8;
h=T/n;
t =(0:n-1)*h;
senial = @(x) sin(2*pi*x*100) + 0.5*cos(2*pi*x*50);
y = senial (t);
Y = fft (y);
potencia = abs(Y.*Y)/n^2;
f =(0:n-1)/T;
plot([0 1/h],[0,0],'k-', [f;f],[0*potencia;potencia],'c-',...
    f,potencia,'b.','markersize',16)
xlabel('Frecuencia')
ylabel('Potencia')
umbral=max(potencia(1:n/2+1))/10;
j = find(potencia(1:n/2+1)>=umbral); % Buscamos los ''picos'' de la señal
j = j - 1; % Restamos 1, porque la teoría está hecha para vectores 0:n-1
frecuencias_detectadas = j/T;
for k = frecuencias_detectadas
    fprintf('La señal tiene una componente de frecuencia %4.1f hertzios.\n',k);
end
```

La salida es

```
La señal tiene una componente de frecuencia 50.0 hertzios.
La señal tiene una componente de frecuencia 100.0 hertzios.
```

Ejemplo 4.11 *En el siguiente ejemplo se muestrea una señal durante un tiempo T mayor que su periodo, pero que cumple que $f(t+T) = f(t)$. Hay que tomar un número suficiente de puntos mediante una frecuencia de muestreo adecuada. En las fórmulas, se usa Fs en lugar de T o h siempre que se puede.*

```
% Ejemplo 2:
% Usamos un T grande, pero multiplo exacto del periodo
% Introducimos la frecuencia de muestreo
%
T=2; % Tiempo de muestro
Fs=256; % Muestras por segundo
n=T*Fs; % Muestras totales
t =(0:n-1)/Fs; % Puntos de muestreo
senial = @(x) sin(2*pi*x*100)+0.5*cos(2*pi*x*50);
y = senial (t);
Y = fft (y);
potencia = abs(Y.*Y)/n^2;
f =(0:n-1)*Fs/n;
plot([0 Fs],[0,0],'k-', [f;f],[0*potencia;potencia],'c-',...
    f,potencia,'b.','markersize',16)
xlabel('Frecuencia')
ylabel('Potencia')
umbral=max(potencia(1:n/2+1))/10;
j = find(potencia(1:n/2+1)>=umbral); % Buscamos los ''picos'' de la señal
j = j - 1; % Restamos 1, porque la teoría está hecha para vectores 0:n-1
frecuencias_detectadas = j*Fs/n;
for k = frecuencias_detectadas
    fprintf('La señal tiene una componente de frecuencia %4.1f hertzios.\n',k);
end
```

La salida es

```
La señal tiene una componente de frecuencia 50.0 hertzios.
La señal tiene una componente de frecuencia 100.0 hertzios.
```

Ejemplo 4.12 *En el siguiente ejemplo se muestrea una señal durante un tiempo T sin especificar, mayor que su periodo, y que NO cumple que $f(t+T) = f(t)$. Hay que tomar un número suficiente de puntos mediante una frecuencia de muestreo adecuada. En las fórmulas, se usa Fs en lugar de T o h siempre que se puede.*

```
% Ejemplo 3:
% No usamos T
% Utilizamos la frecuencia de muestreo
% Y muestreamos un número prefijado de puntos
%
Fs=512;
n=1024;
t =(0:n-1)/Fs;
senial = @(x) sin(2*pi*x*97.12)+0.5*cos(2*pi*x*41.2);
y = senial (t);
Y = fft (y);
potencia = abs(Y.*Y)/n^2;
f =(0:n-1)*Fs/n;
plot([0 Fs],[0,0],'k-', [f;f],[0*potencia;potencia],'c-',...
    f,potencia,'b.','markersize',16)
xlabel('Frecuencia')
ylabel('Potencia')
umbral=max(potencia(1:n/2+1))/10;
j = find(potencia(1:n/2+1)>=umbral); % Buscamos los ''picos'' de la señal
j = j - 1; % Restamos 1, porque la teoría está hecha para vectores 0:n-1
frecuencias_detectadas = j*Fs/n;
fprintf('\n')
for k = frecuencias_detectadas
    fprintf('La señal tiene una componente de frecuencia %4.1f hertzios.\n',k);
end
```

La salida es

```
La señal tiene una componente de frecuencia 41.0 hertzios.
La señal tiene una componente de frecuencia 97.0 hertzios.
```

Este es el ejemplo más cercano a la realidad. Aparecen coeficientes distintos de cero cuyas frecuencias no estaban presentes en la señal original. Se hace más difícil detectar los picos.

En la figura 1 se aprecia muy bien que la segunda mitad del vector transformado "sobra". Si nos fiásemos de esa parte, aparecerían frecuencias espurias, que no aparecen en la señal original y que dependen de la frecuencia de muestreo, o del número de datos usado. En general, sólo se representa hasta la frecuencia de Nyquist $f = Fs/2 = (n/2) * Fs/n$. Además, es normal representar la potencia en decibelios. Para pasar de unidades de potencia a decibelios, hay que escoger una potencia de referencia P_r y usar la fórmula

$$dB = 10 \log_{10} \left(\frac{P}{P_r} \right)$$

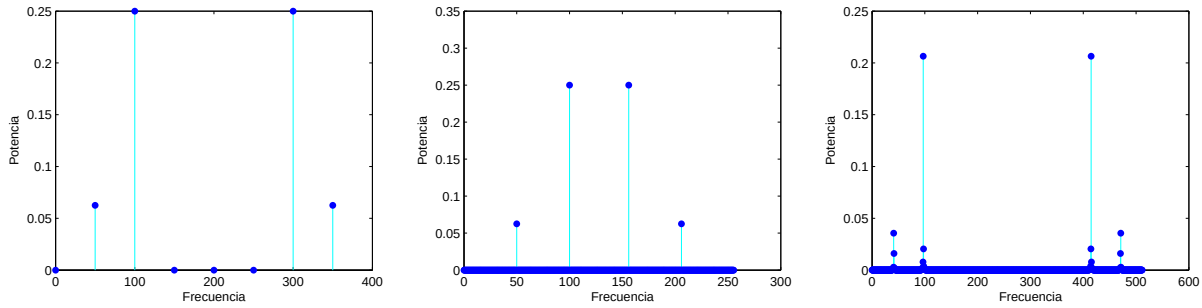
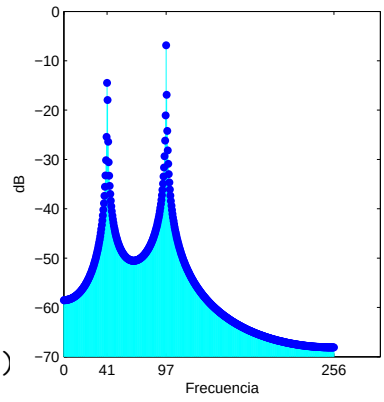


Figura 1: Izquierda: ejemplo 1. Centro: ejemplo 2. Derecha: ejemplo 3.

Ejemplo 4.13 *Añadiendo las siguientes líneas al ejemplo 4.12, obtenemos la gráfica en decibelios hasta la frecuencia de Nyquist.*

```
f =(0:n/2)*Fs/n;
Nyquist = Fs/2;
potref = 1;
dB=10*log10(potencia(1:n/2+1)/potref);
plot([0 Nyquist],[0,0],'k-', [f;f],...
    [-70*ones(size(dB));dB],'c-',...
    f,dB,'b.','markersize',16)
xlabel('Frecuencia')
ylabel('dB')
set(gca,'xtick',[0 frecuencias_detectadas Nyquist])
```



Ejercicios

Ejercicio 4.1 *Estima el condicionamiento (en la norma infinito, por ejemplo) de la matriz de Vandermonde correspondiente a los nodos* $x = 10^{-t}(0, 1, 2)$ *para* $t \in \mathbb{N}$.

Ejercicio 4.2 *Para los nodos* $x = (-1, 0, 1)$ *calcula y representa los polinomios de basa de Lagange* $\ell_i(x)$ *para* $i = 1 : 3$ *que cumplen que* $\ell_i(x_j) = \delta_{i,j}$. *Utilízalos para calcular el polinomio que interpola la tabla*

x	-1	0	1
y	0	-1	2

Ejercicio 4.3 *Calcular la cúbica* $y = ax^3 + bx^2 + cx + d$ *que pasa por los puntos* $(0, 0)$, $(1, 1)$, $(2, 2)$ *y* $(3, 2)$.

Ejercicio 4.4 *Calcular la recta* $y = cx + d$ *que mejor ajusta puntos* $(0, 0)$, $(1, 1)$, $(2, 2)$ *y* $(3, 2)$ *en el sentido de los mínimos cuadrados. Plantéese un sistema incompatible y resuélvanse las ecuaciones normales. Calcúlese el condicionamiento de la matriz* 2×2 *que se usa para resolver el sistema.*

Ejercicio 4.5 *Llamamos F^n a la matriz que realiza la transformada discreta de Fourier con n puntos de muestreo:*

$$F^n_{j,k} = \omega_n^{(j-1),(k-1)},\ j,k = 1:n,$$

donde $w_n = exp(-2\pi i/n)$ es la raíz enésima de la unidad. Escribe explícitamente F^1, F^2 y F^4.

Ejercicio 4.6 *Tomemos la señal*

$$s(t) = \text{sen}(2\pi t)$$

La muestreamos hasta $T = 1$ con una frecuencia de muestreo $Fs = n = 4$.

a) Escribe los puntos de muestreo t_k, $k = 0:3$ y los valores de la señal discreta y_k.

b) Calcula la transformada de Fourier discreta del vector $y = (y_0, y_1, y_2, y_3)^T$

b1) Usando directamente F^4

b2) Usando al menos una vez el algoritmo de transformada rápida de Fourier y la matriz F^2.

c) Identifica la frecuencia de Nyquist y representa el diagrama frecuencia–potencia hasta esa frecuencia.

d) Recupera las aproximaciones de los coeficientes c_j, a_j y b_j de la teoría clásica de Fourier a partir de los Y_j.

Ejercicios complementarios

Ejercicio 4.7 *La siguiente tabla nos muestra valores de las funciones de Bessel de primera clase de órdenes* 0 *y* 1*.*

x	$J_0(x)$	$J_1(x)$
0.0	1	0
0.2	0.99	0.0995
0.4	0.9604	0.196
0.6	0.912	0.2867
0.8	0.8463	0.3688
1.0	0.7652	0.4401
1.2	0.6711	0.4983
1.4	0.5669	0.5419
1.6	0.4554	0.5699
1.8	0.34	0.5815
2.0	0.2239	0.5767
2.2	0.1104	0.556
2.4	0.002508	0.5202
2.6	−0.0968	0.4708
2.8	−0.185	0.4097
3.0	−0.2601	0.3391
3.2	−0.3202	0.2613
3.4	−0.3643	0.1792
3.6	−0.3918	0.09547
3.8	−0.4026	0.01282
4.0	−0.3971	−0.06604
4.2	−0.3766	−0.1386
4.4	−0.3423	−0.2028
4.6	−0.2961	−0.2566
4.8	−0.2404	−0.2985
5.0	−0.1776	−0.3276

a) Utilizando interpolación lineal a trozos, calcula $J_0(1.7)$ *y* $J_0(3.9)$*. Compara con los valores (exactos hasta la cuarta cifra)* $J_0(1.7) = 0.3980$ *y* $J_0(3.83) = -0.4028$*.*

Aunque los valores obtenidos son bastante aproximados, en $x = 3.83$ *se observa un efecto raro: la función es decreciente entre* 3.80 *y* 3.83*, pero a nosotros nos sale creciente.*

b) Usando que $J_1(x) = -J_0'(x)$ *obtén el polinomio* $p(x)$ *de grado 3 que cumple que* $p(x_i) = J_0(x_i)$ *y* $p'(x_i) = J_0'(x_i)$ *(*$i = 1, 2$*) donde* $x_1 = 3.8$ *y* $x_2 = 4$ *son los dos nodos más cercanos a* $x = 3.83$*.*

c) Calcula $p(3.83)$*. ¿Ha desaparecido el efecto?*

d) $p(x)$ *alcanza un mínimo relativo en esa zona. ¿En que punto? ¿Cuánto vale? Si lo has hecho bien, el valor coincide al menos hasta la cuarta cifra con el valor del mínimo relativo de* $J_0(x)$ *en esa zona.*

e) Si no tuviésemos los valores de la derivada, podríamos haber obtenido un polinomio de grado 3 interpolando los dos nodos a la izquierda y los dos nodos a la derecha de 3.83*. Calcula este polinomio y repite los apartados c) y d).*

f) A la vista de la tabla, es fácil ver que debe haber una solución de $J_0(x) = 0$ *en* $[2.4, 2.6]$. *Para calcularla, podríamos usar la tabla para calcular algún polinomio que interpolase a la función en puntos cercanos y después resolver la ecuación polinómica* $p(x) = 0$. *Hay una manera muy sencilla de evitar la resolución de la ecuación: cambiar el punto de vista y darle la vuelta a la tabla. Por ejemplo, en lugar de interpolar*

x	2.2	2.4	2.6
y	0.1104	0.002508	−0.0968

interpola

y	0.1104	0.002508	−0.0968
x	2.2	2.4	2.6

y obtén un polinomio $q(y)$. *Finalmente calcula* $\hat{x} = q(0)$. *Compara con la solución real* $x = 2.4048$.

Ejercicio 4.8 *Desde tiempos de Newton se sabe que en un campo gravitatorio de intensidad* g (m/sg^2), *si se deja caer un cuerpo a velocidad* v_0 (m/sg) *desde una altura* h (m), *el espacio* s (m) *y el tiempo* t (sg) *se relacionan mediante la ecuación*

$$\left(\frac{g}{2}\right) t^2 + v_0 t + h = s$$

Para medir la gravedad se deja caer un peso. A altura $h = 100\,m$ *se pone el cronómetro a cero y se miden los tiempos con un cronómetro que mide hasta las décimas cada 25 metros hasta que toca el suelo. Resulta la siguiente tabla de datos.*

$s(m)$	75	50	25	0
$t(sg)$	2.2	3.2	3.9	4.5

Hay que calcular g *y* v_0. *Para ello (utilizando 4 cifras decimales para todos los cálculos):*

1. *Plantea un sistema lineal* $Ax = b$ *de 4 ecuaciones y 2 incógnitas* $x = [g/2, v_0]^T$ *con los datos del problema.*
2. *Para buscar su solución de mínimos cuadrados, plantea las ecuaciones normales* $Nx = c$.
3. *Resuelve* $Nx = c$ *mediante el método de Gauss con pivote parcial y ...*
4. *de paso escribe la descomposición LU de la matriz* N.
5. *Utiliza esta descomposición para calcular el* $\det(N)$.
6. *Calcula* $\|N\|_\infty$ *y* $\mathrm{cond}_\infty(N)$.
7. *A la vista de los resultados, escribe el valor de* g *con tantas cifras significativas correctas como puedas.*

Ejercicio 4.9 *Tomemos la señal*

$$s(t) = \operatorname{sen}(2 \cdot 2\pi t)$$

a)¿Porqué no es suficiente con tomar $n = 4$ como en el Ejercicio 4.6?

La muestreamos hasta $T = 1$ con una frecuencia de muestreo $Fs = n = 8$.

b) Escribe los puntos de muestreo t_k, $k = 0 : 7$ y los valores de la señal discreta y_k.

c) Calcula la transformada de Fourier discreta del vector $y = (y_0, y_1, \cdots, y_7)^T$ usando al menos una vez el algoritmo de transformada rápida de Fourier y la matriz F^4.

d) Identifica la frecuencia de Nyquist y representa el diagrama frecuencia–potencia hasta esa frecuencia.

e) Recupera las aproximaciones de los coeficientes c_j, a_j y b_j de la teoría clásica de Fourier a partir de los Y_j.

Ejercicio 4.10 *Repite los apartados b)-e) del ejercicio 4.9 para*

$$s(t) = \operatorname{sen}(1 \cdot 2\pi t)$$

tomando $Fs = 4$ y $n = 8$ (calcula primero T). Cuidado al hacer el diagrama frecuencia-potencia. Ten en cuenta que las frecuencias se calculan con la fórmula (3) *(también te puede ser útil la fórmula* (4)*)*

Ejercicio 4.11 *a) Repite los apartados b)-e) del ejercicio 4.9 para*

$$s(t) = \operatorname{sen}(6 \cdot 2\pi t)$$

¿Qué observas? Este efecto se llama aliasing. Busca información sobre él.

b) Repite los apartados b)-e) del ejercicio 4.9 para

$$s(t) = \operatorname{sen}\left(\frac{3}{2} \cdot 2\pi t\right)$$

¿Qué observas? Este efecto se llama picket-fence. Busca información sobre él.