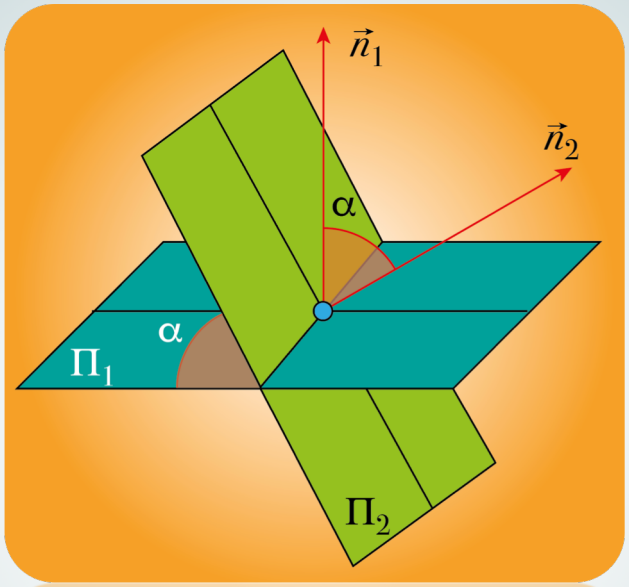
Álgebra y Geometría

Práctica 8. Espacios euclídeo (II)



Rodrigo García Manzanas
Ruth Carballo Fidalgo

Departamento de Matemática Aplicada y
Ciencias de la Computación

Grado en Ingeniería Civil

G1954: Álgebra y Geometría

Práctica 8: Espacio euclídeo (II)

Rodrigo García Manzanas (rodrigo.manzanas@unican.es)

Ruth Carballo Fidalgo (ruth.carballo@unican.es)

Objetivos

- Resolución (aproximada) de sistemas incompatibles por mínimos cuadrados
- Ajuste a una nube de puntos

Resolución (aproximada) de sistemas incompatibles por mínimos cuadrados

Imaginemos el siguiente sistema de ecuaciones lineales:

$$\left\{\begin{array}{rcr} x-y & = & 1 \\ y+z & = & 0 \\ x+y+2z & = & -1 \end{array}\right.$$

Es fácil comprobar que es incompatible (no tiene solución):

```
A = [1 -1 0; 0 1 1; 1 1 2];  % matriz de coeficientes
b = [1 0 -1]';  % vector de términos independientes
Aamp = [A b];  % matriz ampliada
[rank(A) rank(Aamp)]  % S.I.
```

```
ans = 1×2
     2     3
```

Hemos visto que en estas circunstancias siempre es posible obtener una solución aproximada por mínimos cuadrados. Se trataría de sustituir el vector de términos indpendientes $\vec{b}$, para el cual no existe solución, por otro vector $\vec{c}$ para el cual el sistema sí tenga solución/soluciones. Este vector $\vec{c}$ será la proyección de $\vec{b}$ sobre el subespacio generado por las columnas de A. Para encontrar nuestro $\vec{c}$ tendremos que hallar en primer lugar una base de este subespacio, por lo que hay que comprobar si las columnas de A son L.I.:

```
rref(A)  % la tercera columna es combinación lineal de las dos primeras, que son L.I.
```

```
ans = 3×3
     1     0     1
     0     1     1
     0     0     0
```

Ahora ya estamos en condiciones de calcular la matriz de proyección sobre el subespacio generado por las columnas de A, puesto que conocemos una base del mismo. A partir de esta base obtendremos la matriz de proyección que nos permitirá encontrar el vector $\vec{c}$ para el cual el sistema pasa a ser compatible.

```
P = A(:,1:2)*inv(A(:,1:2)'*A(:,1:2))*A(:,1:2)'  % matriz de proyección (me quedo
```

```
P = 3×3
    0.8333   -0.3333    0.1667
   -0.3333    0.3333    0.3333
    0.1667    0.3333    0.8333
```

```
% únicamente las columnas de A que son L.I.)
c = P*b  % vector para el cual el sistema pasa a ser compatible. Es
```

```
c = 3×1
    0.6667
   -0.6667
   -0.6667
```

```
% la proyección de b sobre el subespacio generado por las columnas de A
[rank(A), rank([A c])]  % S.C.I.
```

```
ans = 1×2
     2     2
```

Nuestro nuevo sistema (el que resulta de sustituir $\vec{b}$ por $\vec{c}$) es compatible indeterminado (infinitas soluciones). Ya sabemos cómo resolver este tipo de sistemas:

```
syms x y z real
X = [x y z]';  % vector de incógnitas simbólicas
sol = solve(A*X == c, [x y])  % resuelvo en las incógnitas principales (columnas
```

```
sol = struct with fields:
    x: [1×1 sym]
    y: [1×1 sym]
```

```
% pivotales de A)
sol_mc = [sol.x sol.y z]  % solución aproximada por mínimos cuadrados
```

```
sol_mc =
```

$$\begin{pmatrix} -z & -z-\frac{2}{3} & z \end{pmatrix}$$

Siempre es recomendable comprobar que la solución que hemos hallado es correcta:

```
% Compruebo que así sea para un par de valores del parámetro z
A*subs(sol_mc, z, 0)' - c  % para z=0
```

```
ans =
```

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

```
A*subs(sol_mc, z, 2.365)' - c   % para z=2.365
```

```
ans =
```

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Y en cualquier caso el error cuadrático cometido sería:

```
norm(c-b)^2  % error cuadrático
```

```
ans = 0.6667
```

Si las columnas de la matriz de coeficientes del sistema incompatible que trato de resolver son L.I., el proceso se simplifica, puesto que una vez que hallemos nuestro $\vec{c}$, nos quedará un S.C.D con una única solución, que podríamos resolver directamente por medio del operador "\". De hecho, una vez sabemos que las columnas de la matriz de coeficientes son L.I. no haría falta ni siquiera pasar por la matriz de proyección, puesto que sabemos que la solución de mínimos cuadrados puede calcularse directamente como $(A^tA)^{-1}A^tb$. Como ejemplo, vamos a resolver el siguiente sistema:

$$\begin{cases} -x+4y &= 15 \\ y &= 6 \\ -y &= 4 \\ x-2y &= -5 \end{cases}$$

```
A = [-1 4; 0 1; 0 -1; 1 -2];  % matriz de coeficientes
b = [15 6 4 -5]';  % vector de términos independientes
[rank(A) rank([A b])]  % S.I.
```

```
ans = 1×2
     2     3
```

```
% Como las columnas de A son L.I. (rg(A)=2), podemos calcular
% directamente la solución (única) de mínimos cuadrados como:
sol_mc = inv(A'*A)*A'*b
```

```
sol_mc = 2×1
   -1.0000
    3.0000
```

Acabamos de hallar que la solución de mínimos cuadrados es $x=-1, y=3$. Esto quiere decir que el vector $\vec{c}$ se puede expresar como combinación lineal de las columnas de A, siendo los coeficientes de dicha C.L. -1 y 3.

Ajuste a una nube de puntos

Hemos visto que dada una nube de puntos, se puede ajustar por un tipo de función predefinida (por ejemplo una recta, una parábola, una exponencial, etc.). Para ello habría que resolver el sistema que resulte de forzar a que nuestra función aproximadora pase por todos los puntos de la nube. Previsiblemente, este sistema será incompatible (no tendrá solución). Sin embargo, podremos buscar una solución aproximada por mínimos cuadrados. A modo de ejemplo, imaginemos que tenemos los siguientes datos experimentales relativos a la medida de la intensidad de corriente que atraviesa un hilo conductor para distintos voltajes:

x=voltaje (V)	2	5	7
y=intensidad (A)	6	7.9	8.5

Para obtener un ajuste lineal (es decir, una recta $y = ax + b$), habría que resolver el sistema

$$\begin{cases} 2a + b &= 6 \\ 5a + b &= 7.9 \\ 7a + b &= 8.5 \end{cases}$$

en el que las incógnitas serán a y b (coeficientes de la recta de ajuste). Lo resolvemos:

```
x0 = [2 5 7];  % coordenada x de los puntos de la nube
y0 = [6 7.9 8.5];  % coordenada y de los puntos de la nube
A = [x0', ones(1, length(x0))']  % matriz de coeficientes del sistema a resolver
```

```
A = 3×2
     2     1
     5     1
     7     1
```

```
b = y0'  % vector de términos independientes del sistema a resolver
```

```
b = 3×1
    6.0000
    7.9000
    8.5000
```

```
[rank(A) rank([A b])]  % S.I.
```

```
ans = 1×2
     2     3
```

Las columnas de A son L.I. ($rg(A) = 2$), por lo que podemos calcular directamente la solución de mínimos cuadrados como:

```
sol_mc = inv(A'*A)*A'*b  % solución de mínimos cuadrados
```

```
sol_mc = 2×1
    0.5105
    5.0842
```

El primer (segundo) elemento de esta solución será el coeficiente a (b) de nuestro ajuste lineal. Por tanto, el polinomio aproximador buscado es $y = 0.5105x + 5.0842$.

Si en lugar de un polinomio aproximador de grado 1 (una recta), quisiera uno de grado 2 (una parábola $y = ax^2 + bx + c$), tendría que resolver el siguiente sistema:

$$\begin{cases} 2^2a + 2b + c &= 6 \\ 5^2a + 5b + c &= 7.9 \\ 7^2a + 7b + c &= 8.5 \end{cases}$$

Donde mis incógnitas serán a, b y c (coeficientes de la curva de ajuste). Lo resolvemos:

```
A = [x0.^2', x0', ones(1, length(x0))']  % matriz de coeficientes (el
```

```
A = 3×3
     4     2     1
    25     5     1
    49     7     1
```

```
% simbolo ".^" se utiliza para realizar la potenciación elemento a elemento)
b = y0'  % vector de términos independientes
```

```
b = 3×1
    6.0000
    7.9000
    8.5000
```

```
[rank(A) rank([A b])]  % S.C.D.
```

```
ans = 1×2
     3     3
```

Vemos que en este caso, el sistema es compatible determinado (existe solución exacta, y es única). Esto ocurre porque tenemos únicamente tres puntos en nuestra nube, por lo que, el número de ecuaciones y el de incógnitas se igualan. *En general, si tenemos una nube con n puntos, siempre podremos encontrar un polinomio interpolador de grado $n - 1$ que pase por todos ellos.* En nuestro caso, la parábola que buscamos vendrá dada por la solución (única) del sistema que hemos planteado:

```
sol = A\b  % S.C.D.
```

```
sol = 3×1
   -0.0667
    1.1000
    4.0667
```

Es decir, $y = -0.0667x^2 + 1.1x + 4.0667$.

Por último, si quisiéramos un ajuste exponencial del tipo $y = ae^{bx}$, podríamos tomar logaritmos a ambos lados de la expresión, quedándonos $ln(y) = ln(a) + bx$, que se convierte en un polinomio de grado 1. Se trataría por tanto de resolver el siguiente sistema:

$$\begin{cases} 2b + ln(a) &= ln(6) \\ 5b + ln(a) &= ln(7.9) \\ 7b + ln(a) &= ln(8.5) \end{cases}$$

Donde mis incógnitas serán b y $ln(a)$ (a partir de las cuales podremos obtener los coeficientes de la curva de ajuste). Lo resolvemos:

```
A = [x0', ones(1, length(x0))']  % matriz de coeficientes
```

```
A = 3×2
     2     1
     5     1
     7     1
```

```
b = log(y0')  % vector de términos independientes (en MATLAB, el
```

```
b = 3×1
    1.7918
    2.0669
    2.1401
```

```
% logaritmo neperiano se calcula con la función "log")
[rank(A) rank([A b])]  % S.I.
```

```
ans = 1×2
     2     3
```

Las columnas de A son L.I. ($rg(A) = 2$), por lo que pudo calcular directamente la solución de mínimos cuadrados como:

```
sol_mc = inv(A'*A)*A'*b  % solución de mínimos cuadrados
```

```
sol_mc = 2×1
    0.0714
    1.6664
```

Por tanto, el coeficiente b en nuestra función de ajuste sería 0.0714, y el a sería $e^{1.664}$. La exponencial que buscábamos será por tanto $y = 5.29284e^{0.0714x}$.

Para el caso de ajustes polinómicos, MATLAB dispone de la función *polyfit,* que devuelve directamente los coeficientes del polinomio aproximador. Podemos utilizarla para comprobar que la recta y la parábola aproximadoras que hemos obtenido anteriormente son correctas:

```
% En la función polyfit, el tercer argumento de entrada hace referencia
% al grado del polinomio aproximador que queremos considerar
p1 = polyfit(x0, y0, 1)  % coeficientes de la RECTA de ajuste
```

```
p1 = 1×2
    0.5105    5.0842
```

```
% y = 0.5105x + 5.0842
p2 = polyfit(x0, y0, 2)  % coeficientes de la PARÁBOLA de ajuste
```

```
p2 = 1×3
   -0.0667    1.1000    4.0667
```

```
% y=-0.0667x^2 + 1.1x + 4.0667
```

Del mismo modo, también podemos utilizar *polyfit* para comprobar que el ajuste exponencial al que habíamos llegado es correcto:

```
p3 = polyfit(x0, log(y0), 1)   % coeficientes del ajuste EXPONENCIAL
```

```
p3 = 1×2
    0.0714    1.6664
```

```
% y = exp(1.6664)*exp(0.0714x)
```

Una vez hemos obtenido los tres ajustes que buscábamos, podemos representar todos ellos en un mismo gráfico junto a la nube de puntos inicial, para lo cual utilizaremos las funciones *polyval* y *plot. polyval* permite evaluar un polinomio (definido por sus coeficientes) en un determinado punto (o puntos). Por ejemplo, podría interesarnos saber cuál sería el valor aproximado para la intensidad cuando el voltaje es de $7V$ de acuerdo a la recta y a la parábola de ajuste que hemos obtenido:

```
polyval(p1, 7)  % valor estimado por la RECTA de ajuste para la
```

```
ans = 8.6579
```

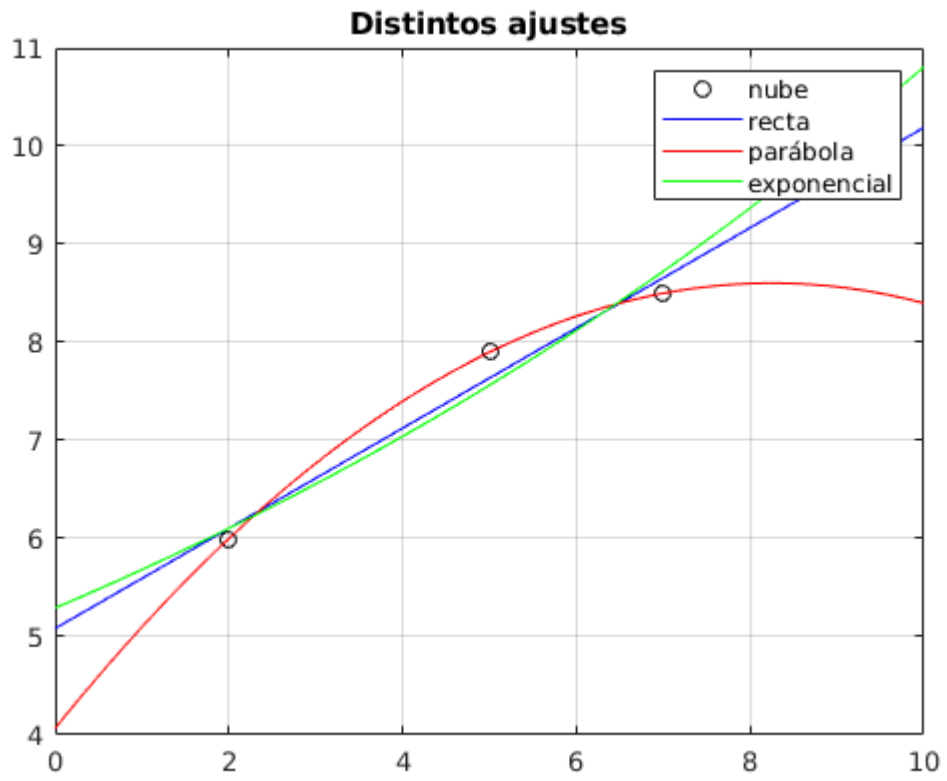
```
% intensidad cuando V=7V
polyval(p2, 7)  % valor estimado por la PARÁBOLA de ajuste para la
```

```
ans = 8.5000
```

```
% intensidad cuando V=7V
```

Por su parte, la función *plot* se usa para representar pares de puntos *(x,y).* Por tanto, construiríamos el gráfico pedido del siguiente modo:

```
figure  % abro una nueva ventana gráfica
plot(x0, y0, 'ok')  % nube de puntos. El argumento opcional 'ob' indica
% que quiero representar círculos (o) negros (k: blacK)

x = 0:0.01:10;  % doy valores a 'x' en el intervalo que quiero dibujar: [0, 10]

hold  on  % para seguir dibujando en la misma ventana gráfica
plot(x, polyval(p1, x), 'b')  % RECTA de ajuste, en azul (b: Blue)

hold  on
plot(x, polyval(p2, x), 'r'),  % PARÁBOLA de ajuste, en rojo (r: Red)

hold  on
plot(x, exp(p3(2))*exp(p3(1)*x), 'g')  % exponencial, en verde (g: Green)

legend('nube', 'recta', 'parábola', 'exponencial')  % añado una leyenda. Los
% textos tienen que ir en el mismo orden que las representaciones
% que he ido haciendo
title('Distintos ajustes')  % añado un título
grid on  % dibuja una rejilla en al fondo (facilita la visualización)
```



Para finalizar, podríamos calcular el error cuadrático cometido en cada una de las tres aproximaciones:

```
norm(y0 - polyval(p1, x0))^2   % error cuadrático cometido con la RECTA
```

```
ans = 0.1053
```

```
norm(y0 - polyval(p2, x0))^2  % error cuadrático cometido con la PARÁBOLA
```

```
ans = 9.4663e-30
```

```
norm(y0 - exp(p3(2))*exp(p3(1)*x0))^2  % error cuadrático cometido con la función
```

```
ans = 0.1747
```

```
% EXPONENCIAL
```

Ejercicios propuestos

Ejercicio 1

Comprueba que los siguientes sistemas son incompatibles y busca una solución de mínimos cuadrados. Calcula el error cuadrático cometido.

a) $\begin{cases} x-2y+4z &= -1 \\ x-y+z &= -4 \\ x &= -3 \\ x+y+z &= -2 \\ x+2y+4z &= 0 \end{cases}$ *b)* $\begin{cases} 3x+y+8z &= 4 \\ x-y+4z &= 3 \\ x+3y &= 7 \end{cases}$

Ejercicio 2

Dado el siguiente conjunto de puntos en el plano $\mathbb{R}^2$: $\{(-3,10),(-2,8),(-1,7),(0,6),(1,4),(2,5),(3,6)\}$.

- *a)* Utilizando *polyfit*, obtén los polinomios de orden 2, 4 y 6 que mejor se ajusten a la nube
- *b)* Representa en un mismo gráfico la nube y los tres polinomios aproximadores
- *c)* Calcula el error cuadrático cometido con cada uno de los tres polinimos aproximadores
- *d)* Calcula el error (en valor absoluto) que se comete con cada uno de los tres polinomios aproximadores en el punto $x=1$

Ejercicio 3

Utiliza la función *polyfit* para encontrar un ajuste de tipo potencial ($y=ax^b$) a la siguientes datos: $\{(10,1.06),(20,1.33),(30,1.52),(40,1.68),(50,1.81),(60,1.91),(70,2.01),(80,2.11)\}$.

- *a)* Representa la nube y la curva de ajuste hallada
- *b)* Calcula el error cuadrático cometido por la curva de ajuste
- *c)* Calcula el eror (en valor absoluto) cometido por la curva de ajuste en el punto $x=40$