

Computación Científica I

Certamen N^o 2 — Sa.26.05.07, 08:00¹

Algoritmo 1 Algoritmo de Gram-Schmidt Modificado

```

for  $j = 1$  to  $n$  do
     $v_j = a_j \equiv A_{1:m,j}$ 
end for
for  $i = 1$  to  $n$  do
     $r_{ii} = \|v_i\|$ ;  $q_i = v_i/r_{ii}$ ;
    for  $j = i + 1$  to  $n$  do
         $r_{ij} = q_i^* v_j$ ;  $v_j = v_j - r_{ij} q_i$ ;
    end for
end for

```

Algoritmo 2 Algoritmo QR de Householder

```

for  $k = 1$  to  $n$  do
     $x = A_{k:m,k}$ ;
     $v_k = \text{sign}(x_1) \|x\|_2 e_1 + x$ ;
     $v_k = v_k / \|v_k\|_2$ ;
     $A_{k:m,k:n} = A_{k:m,k:n} - 2v_k (v_k^* A_{k:m,k:n})$ ;
end for

```

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 1 & 2 & -1 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & -2 & 1 \end{bmatrix} \quad (1)$$

$$Q = \begin{bmatrix} 1/\sqrt{3} & 0 & 1/\sqrt{6} \\ 0 & 1 & 0 \\ -1/\sqrt{3} & 0 & -1/\sqrt{6} \\ 1/\sqrt{3} & 0 & -\sqrt{2/3} \end{bmatrix}, \quad R = \begin{bmatrix} \sqrt{3} & -\sqrt{3} & 0 & 0 & -1/\sqrt{3} & -2/\sqrt{3} & 1/\sqrt{3} \\ 0 & 0 & 1 & -1 & 1 & 2 & -1 \\ 0 & 0 & 0 & 0 & \sqrt{2/3} & 2\sqrt{2/3} & -\sqrt{2/3} \end{bmatrix} \quad (2)$$

FACT: Dada la matriz A que aparece en (1), un célebre programa arroja una factorización QR reducida QR de la matriz A , donde las matrices Q y R vienen dadas en (2).

1. GRAM-SCHMIDT. (a) La matriz A tiene, evidentemente, columnas linealmente dependientes. Modifique adecuadamente el Algoritmo 1 para manejar esta situación.
- (b) Suponga que su algoritmo modificado le ha permitido obtener hasta el vector unitario $Q_{1:4,2}$, i.e., hasta la segunda columna de Q . Aplicando su algoritmo, obtenga $Q_{1:4,3}$.
- (c) Partiendo de la factorización QR reducida, y aplicando su algoritmo, obtenga la factorización QR completa de A .
- (d) Determine la complejidad de su algoritmo.

Desarrollo. (a) El algoritmo de la Figura 1 responde la pregunta. Este algoritmo desarrolla la siguiente simple idea: después de que se ha hallado la columna q_i de la matriz ortonormal Q y se ha actualizado todas las columnas a_j (restando de a_j su proyección $(q_i^* a_j) q_i$ sobre q_i) con $j = (i+1) : n$, se pasa a la iteración $i + 1$ y se busca la que llamaremos *vector-columna pivote*, que es el primero de los vectores-columnas $j = (i + 1) : n$ actuales cuya $(i + 1)$ -ésima componente es no nula. Ese vector-columna pivote hallado se utiliza en seguida para generar el vector-columna q_{i+1} de la matriz Q . El procedimiento se repite hasta recorrer toda la matriz A , que se actualiza en cada iteración. Es necesario incluir algunas cláusulas de verificación para impedir que se desborde el rango de las matrices consideradas.

- (b) En este punto se dispone de los vectores-columnas:

$$q_1 = Q_{1:4,1} = \begin{bmatrix} 1/\sqrt{3} \\ 0 \\ -1/\sqrt{3} \\ 1/\sqrt{3} \end{bmatrix}, \quad q_2 = Q_{1:4,2} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}. \quad (3)$$

¹© Luis Salinas Carrasco, Valparaíso, 18 de junio de 2007. De antemano se agradece toda corrección, crítica o comentario que el amable lector tenga a bien hacer llegar a luis.salinas@usm.cl.

Algoritmo 3 Algoritmo de Gram-Schmidt Modificado

```

V = A;                                ▷ Se define la matriz auxiliar V con el valor inicial A.
Q = I_m;                                ▷ Se define la matriz Q con el valor inicial I_m.
R = Z_{m,n};                            ▷ Se define la matriz R inicial como la matriz nula Z_{m,n} ∈ M(m × n, C).
L = 0;                                  ▷ Variable auxiliar para definir los pivotes de R.
for i = 1 to m do
  L = L + 1;
  R_{i,L} = √{V_{1:m,L}^* V_{1:m,L}};
  while R_{i,L} ≤ 0 and L < n do      ▷ Bucle de búsqueda del vector-columna pivote.
    L = L + 1;
    R_{i,L} = √{V_{1:m,L}^* V_{1:m,L}};
  end while
  if R_{i,L} > 0 then                  ▷ Precaución, pues el bucle while puede terminar con L = n y R_{i,L} = 0.
    Q_{1:m,i} = V_{1:m,L} / R_{i,L};
    for j = L + 1 to n do
      R_{i,j} = Q_{1:m,i}^* V_{1:m,j};
      V_{1:m,j} = V_{1:m,j} - R_{i,j} Q_{1:m,i};
    end for
  else [R_{i,L} = 0]
  end if
end for

```

FIGURA 1. Algoritmo de Gram-Schmidt modificado recargado.

Se requiere ahora determinar la quinta columna de la matriz A inmediatamente después de concluida su actualización mediante q_1 y q_2 . Por la forma de la matriz R que aparece en (2), se observa que dicha actualización ya ha permitido obtener la forma final de sus primeras cuatro columnas. Desde luego que la quinta columna buscada *no es* la quinta columna de la matriz R en (2) sino que:

$$\begin{aligned}
v_5 &:= A_{1:4,5} - (q_1^* \cdot A_{1:4,5}) q_1 - (q_2^* \cdot A_{1:4,5}) q_2 \\
&= \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \end{bmatrix} - \left(\begin{bmatrix} 1/\sqrt{3} \\ 0 \\ -1/\sqrt{3} \\ 1/\sqrt{3} \end{bmatrix}^* \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \end{bmatrix} \right) \begin{bmatrix} 1/\sqrt{3} \\ 0 \\ -1/\sqrt{3} \\ 1/\sqrt{3} \end{bmatrix} - \left(\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}^* \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \end{bmatrix} \right) \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1/3 \\ 0 \\ -1/3 \\ -2/3 \end{bmatrix}.
\end{aligned}$$

Por su definición, este v_5 ya es ortogonal a q_1 y q_2 . Para obtener q_3 basta normalizar v_5 :

$$q_3 := \frac{v_5}{\sqrt{v_5^* \cdot v_5}} = \begin{bmatrix} 1/\sqrt{6} \\ 0 \\ -1/\sqrt{6} \\ -\sqrt{2/3} \end{bmatrix}, \quad (4)$$

que coincide con lo planteado en la expresión para Q en (2)

(c) Confiando (sic!) en el “célebre programa”, podemos partir de la base que efectivamente se tiene $A = QR$, donde A , Q y R vienen dado por (1) y (2), lo que se puede verificar en un instante (please, do it!).

Observamos que Q tiene 4 filas y 3 columnas: q_1, q_2, q_3 . Luego, a Q le falta una columna q_4 ortonormal a las precedentes para constituirse en el factor \hat{Q} de la factorización QR *completa* buscada. Por su parte, R tiene 3 filas y 7 columnas. Luego, para obtener el factor \hat{R} de la factorización QR *completa* bastará agregar a R una cuarta fila de ceros.

Para obtener q_4 basta aplicar el algoritmo de Gram-Schmidt estándar a uno de los vectores canónicos e_1, e_2, e_3, e_4 de \mathbb{C}^4 , partiendo de los vectores q_1, q_2, q_3 . Se constata que ya e_1 permite calcular q_4 . Brevemente, el cálculo es el siguiente:

$$\begin{aligned}\tilde{q}_4 &:= e_1 - (q_1^* \cdot e_1) q_1 - (q_2^* \cdot e_1) q_2 - (q_3^* \cdot e_1) q_3 \\ &= \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} - \left(\begin{bmatrix} 1/\sqrt{3} \\ 0 \\ -1/\sqrt{3} \\ 1/\sqrt{3} \end{bmatrix}^* \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right) \begin{bmatrix} 1/\sqrt{3} \\ 0 \\ -1/\sqrt{3} \\ 1/\sqrt{3} \end{bmatrix} - \left(\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}^* \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right) \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} - \left(\begin{bmatrix} 1/\sqrt{6} \\ 0 \\ -1/\sqrt{6} \\ -\sqrt{2/3} \end{bmatrix}^* \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right) \begin{bmatrix} 1/\sqrt{6} \\ 0 \\ -1/\sqrt{6} \\ -\sqrt{2/3} \end{bmatrix}.\end{aligned}$$

Efectuando las operaciones indicadas resulta $\tilde{q}_4 = [1/2, 0, 1/2, 0]^T$ y normalizando el vector \tilde{q}_4 se obtiene $q_4 = [1/\sqrt{2}, 0, 1/\sqrt{2}, 0]^T$. De este modo los factores Q y R de la factorización QR completa de A quedan:

$$Q = \begin{bmatrix} 1/\sqrt{3} & 0 & 1/\sqrt{6} & 1/\sqrt{2} \\ 0 & 1 & 0 & 0 \\ -1/\sqrt{3} & 0 & -1/\sqrt{6} & 1/\sqrt{2} \\ 1/\sqrt{3} & 0 & -\sqrt{2/3} & 0 \end{bmatrix}, \quad R = \begin{bmatrix} \sqrt{3} & -\sqrt{3} & 0 & 0 & -1/\sqrt{3} & -2/\sqrt{3} & 1/\sqrt{3} \\ 0 & 0 & 1 & -1 & 1 & 2 & -1 \\ 0 & 0 & 0 & 0 & \sqrt{2/3} & 2\sqrt{2/3} & -\sqrt{2/3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (5)$$

(d) El Algoritmo 3 es esencialmente idéntico al Algoritmo 1, excepto por el bucle “**while**” de búsqueda del vector-columna pivote de cada fila.

Recuérdese que aquí sólo consideramos el caso $m \leq n$.

Fijemos un $i, i = 1 : m$, y estimemos las complejidades de los bucles “**while**” y j internos al bucle i . Para cada $L, 1 \leq i \leq L \leq n$, el bucle “**while**” requiere el cálculo del producto interno $V_{1:m,L}^* V_{1:m,L}$, que implica m multiplicaciones y $m - 1$ sumas, esto es, esencialmente $2m$ operaciones. Para cada L hay que calcular, todavía, la raíz cuadrada del producto interno precedente. La complejidad del bucle “**while**” es, entonces, a lo más:

$$\sum_{\nu=i}^n (1 + 2m) = \sum_{\nu=i}^n 1 + 2m \sum_{\nu=i}^n 1 = (n - i + 1) + 2m(n - i + 1) = (2m + 1)(n - i + 1). \quad (6)$$

El bucle j del Algoritmo 3 es idéntico al mismo bucle del Algoritmo 1. Como se sabe, la complejidad del Algoritmo 1 queda dominada por la complejidad de este bucle más interno. Para cada $j, i \leq L < j \leq n$, la primera línea del bucle j requiere m multiplicaciones y $m - 1$ adiciones. La segunda línea requiere m multiplicaciones y m subtracciones. Luego, el cálculo de cada iteración j requiere aproximadamente $4m$ “*flops*” por cada vector-columna. Por consiguiente, el número de “*flops*” requeridos por el bucle j es asintótico a:

$$\sum_{j=i+1}^n 4m = 4m \sum_{j=i+1}^n 1 = 4m(n - i). \quad (7)$$

La complejidad total se obtiene entonces sumando (6) y (7) para $i = 1 : m$:

$$\begin{aligned}\sum_{i=1}^m ((2m + 1)(n - i + 1) + 4m(n - i)) &= \frac{1}{2} m(2m + 1)(2n - m + 1) + 2m^2(2n - m - 1) \\ &\approx m^2(2n - m) + 2m^2(2n - m) = 3m^2(2n - m)\end{aligned} \quad (8)$$

Por consiguiente, la complejidad del Algoritmo 3 es del orden de $3m^2(2n - m)$. ■

2. HOUSEHOLDER. (a) Modifique adecuadamente el Algoritmo 2 de modo que le permita obtener la factorización QR completa de A .

(b) Suponga que su algoritmo modificado le ha permitido obtener hasta la cuarta columna de la matriz R . Obtenga la quinta columna de la matriz R aplicando su algoritmo.

(c) Determine la complejidad de su algoritmo.

Desarrollo. (a) El Algoritmo 2 producirá un mensaje de error (división por cero) en la k -ésima iteración si $x = A_{k:m,L} = 0$, $L = L(k)$. Conviene modificar ligeramente la notación escribiendo $x = A_{k:m,L} \in \mathbb{C}^r$, donde $r := m - k + 1$. En efecto, L no necesariamente es igual a k sino que, en general, se tiene $1 \leq k \leq L \leq n$. Después de definir el vector $x \in \mathbb{C}^r$, el Algoritmo 2 define:

$$v = \text{sign}(x_1) \|x\|_2 \mathbf{e}_1 + x \in \mathbb{C}^r, \quad r := m - k + 1.$$

Entonces se observa que:

$$\begin{aligned} v = 0 &\Leftrightarrow \begin{bmatrix} \text{sign}(x_1) \|x\|_2 + x_1 \\ x_2 \\ \vdots \\ x_r \end{bmatrix} = 0 \\ &\Leftrightarrow \text{sign}(x_1) \|x\|_2 + x_1 = x_2 = \cdots = x_r = 0 \\ &\Leftrightarrow x_1 = -\text{sign}(x_1) |x_1| = -x_1 \quad y \quad x_2 = \cdots = x_r = 0 \\ &\Leftrightarrow x_1 = x_2 = \cdots = x_r = 0 \\ &\Leftrightarrow x = 0. \end{aligned}$$

Luego, cuando $x = 0$, se tiene $v = 0$, de modo que el estado “ $v = v/\|v\|_2$ ” del Algoritmo 2 conduciría a una división por cero. La solución de este problema, sin embargo, es muy simple: En efecto, en este caso, no hay nada que calcular y se puede pasar a la columna siguiente para considerar $x = A_{k:m,L+1}$. Nótese que en este caso el rango del primer subíndice permanece constante, a saber, $k : m$.

Para modificar el Algoritmo 2 conviene introducir una nueva variable, L , para controlar el rango del segundo índice en $A_{k,\ell}$. De este modo el algoritmo modificado queda así:

Algoritmo 4 Algoritmo QR de Householder

```

R = A;
L = 1;
for k = 1 to m do
    x = R_{k:m,L};
    while \|x\|_2 ≤ 0 and L < n do
        L = L + 1;
        x = R_{k:m,L};
    end while
    if \|x\|_2 > 0 then
        v = sign(x_1) \|x\|_2 e_1 + x;
        v = v/\|v\|_2;
        R_{k:m,L:n} = R_{k:m,L:n} - 2v(v* R_{k:m,L:n});
    else[\|x\|_2 = 0]
    end if
end for

```

Nótese que, cuando $\|x\|_2 = 0$, el índice L se incrementa, lo que explica las desigualdades $1 \leq k \leq L \leq n$ anotadas más arriba.

La cláusula “**If** $\|x\|_2 > 0$ **then** ... **else**” se ha agregado para prevenir el caso en que el bucle “**while**” termina con $L = n$ y $\|x\|_2 = 0$.

En la práctica, habría que reemplazar la condición “ $\|x\|_2 > 0$ ” por “ $\|x\|_2 > \varepsilon$ ” para un $\varepsilon > 0$ que dependerá del sistema computacional.

(b) Se observa que las dos primeras iteraciones del bucle k del Algoritmo 4 ya producen la forma final de las 4 primeras columnas de la matriz R final:

$$k = 1, \quad L = 1, \quad A_1 = \begin{bmatrix} -\sqrt{3} & \sqrt{3} & 0 & 0 & \frac{1}{\sqrt{3}} & \frac{2}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \\ 0 & 0 & 1 & -1 & 1 & 2 & -1 \\ 0 & 0 & 0 & 0 & \frac{-3+\sqrt{3}}{6} & \frac{-3+\sqrt{3}}{3} & \frac{3-\sqrt{3}}{6} \\ 0 & 0 & 0 & 0 & \frac{-3-\sqrt{3}}{6} & \frac{-3-\sqrt{3}}{3} & \frac{3+\sqrt{3}}{6} \end{bmatrix},$$

$$k = 2, \quad L = 3, \quad A_2 = \begin{bmatrix} -\sqrt{3} & \sqrt{3} & 0 & 0 & \frac{1}{\sqrt{3}} & \frac{2}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \\ 0 & 0 & 1 & -1 & 1 & 2 & -1 \\ 0 & 0 & 0 & 0 & \frac{-3+\sqrt{3}}{6} & \frac{-3+\sqrt{3}}{3} & \frac{3-\sqrt{3}}{6} \\ 0 & 0 & 0 & 0 & \frac{-3-\sqrt{3}}{6} & \frac{-3-\sqrt{3}}{3} & \frac{3+\sqrt{3}}{6} \end{bmatrix}.$$

Aplicando una vez más el Algoritmo 4 se obtiene:

$$k = 3, \quad L = 5, \quad A_3 = \begin{bmatrix} -\sqrt{3} & \sqrt{3} & 0 & 0 & \mathbf{1}/\sqrt{3} & 2/\sqrt{3} & -1/\sqrt{3} \\ 0 & 0 & -1 & 1 & -\mathbf{1} & -2 & 1 \\ 0 & 0 & 0 & 0 & \sqrt{2/3} & 2\sqrt{2/3} & -\sqrt{2/3} \\ 0 & 0 & 0 & 0 & \mathbf{0} & 0 & 0 \end{bmatrix}. \quad (9)$$

La quinta columna (!) de A_3 es la columna pedida.

Estos cálculos se podrían abreviar mucho si se repara que, para cada k , sólo se requiere obtener la quinta columna de R . En efecto, en lugar de calcular la expresión $R_{k:m,L:n} = R_{k:m,L:n} - 2v(v^*R_{k:m,L:n})$ completa, basta computar $R_{k:m,L:5} = R_{k:m,L:5} - 2v(v^*R_{k:m,L:5})$.

(c) El Algoritmo 4 es, excepto por el bucle “**while**” y la cláusula “**if** $\|x\|_2 > 0$ **then** ... **else**”, idéntico al Algoritmo 2 de Householder, cuya complejidad, como se sabe (cf. [1, p. 75])

Trefethen, p. 75), es del orden de $2mn^2 - \frac{2}{3}n^3$.

Para cada iteración del bucle k , tanto en el bucle “**while**” como en la cláusula “**if** $\|x\|_2 > 0$ **then** ... **else**”, se requiere el cálculo de $\|x\|_2$. La dimensión del vector x es $m-k+1$, de modo que el cálculo de $\|x\|_2$ requiere $(m-k+1)$ multiplicaciones y $m-k$ adiciones, esto es, aproximadamente $2(m-k+1)$ “*flops*”. Luego, por este concepto se requiere, en total, aproximadamente $2\sum_{k=1}^m(m-k+1) = 2\sum_{k=1}^m k = m(m+1) \approx m^2$ “*flops*”.

Luego, la complejidad del Algoritmo 4 será del orden de:

$$2mn^2 - \frac{2}{3}n^3 + m^2 \quad \text{“flops”}. \quad (10)$$

■

3. SVD. Sea $A \in M(m \times n, \mathbb{C})$ tal que $\text{rank}(A) = r$ con $r < n < m$. Sea $A = U\Sigma V^*$ la descomposición de valor singular (SVD) de A .

(a) Demuestre o refute: $\text{Range}(A) = \langle u_1, \dots, u_r \rangle$, donde u_j es la j -ésima columna de U y $\langle u_1, \dots, u_r \rangle$ denota el espacio vectorial generado por las r primeras columnas de U .

(b) Demuestre o refute: $\text{Null}(A) = \langle u_{r+1}, \dots, u_n \rangle$.

(c) Exprese el proyector ortogonal P de \mathbb{C}^m sobre $\text{Range}(A)$ mediante la SVD de A .

Desarrollo. Primeramente observamos que:

Teorema 1. $\text{rank}(A) = \text{rank}(\Sigma) = r =$ cantidad de valores singulares no nulos de A .

Demostración. En la SVD $A = U\Sigma V^*$ de la matriz A , las matrices U y V tienen rango completo, de modo que preservan rango. Luego, $\text{rank}(A) = \text{rank}(\Sigma) = r$. ■

(a) es exactamente el contenido del siguiente:

Teorema 2. $\text{Range}(A) = \langle u_1, \dots, u_r \rangle \subseteq \mathbb{C}^m$.

Demostración. De la SVD $A = U\Sigma V^*$ de la matriz A se tiene:

$$y \in \text{Range}(A) \subseteq \mathbb{C}^m \Leftrightarrow y = A.x, \quad \text{para algún } x \in \mathbb{C}^n$$

$$\Leftrightarrow y = A.V.z = U.\Sigma.z = U. \begin{bmatrix} \sigma_1 z_1 \\ \vdots \\ \sigma_r z_r \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \text{para algún } z \in \mathbb{C}^n$$

$$\Leftrightarrow y = \sigma_1 z_1 u_1 + \dots + \sigma_r z_r u_r, \quad \text{para algún } z \in \mathbb{C}^n$$

$$\Leftrightarrow y \in \langle u_1, \dots, u_r \rangle \subseteq \mathbb{C}^m.$$

■

(b) En vista de que $\text{Null}(A) \subseteq \mathbb{C}^n$ y $\langle u_{r+1}, \dots, u_m \rangle \subseteq \mathbb{C}^m$, donde $n < m$, es claro que la proposición enunciada no puede ser cierta. No obstante se tiene:

Teorema 3. $\text{Null}(A) = \langle v_{r+1}, \dots, v_n \rangle \subseteq \mathbb{C}^n$.

Demostración. De la SVD $A = U.\Sigma.V^*$ de la matriz A se tiene:

$$x \in \text{Null}(A) \subseteq \mathbb{C}^n \Leftrightarrow A.x = U.\Sigma.V^*.x = 0 \Leftrightarrow \Sigma.V^*.x = \begin{bmatrix} \sigma_1 v_1^*.x \\ \vdots \\ \sigma_r v_r^*.x \\ 0 \\ \vdots \\ 0 \end{bmatrix} = 0$$

$$\Leftrightarrow x \perp v_1 \wedge \dots \wedge x \perp v_r \Leftrightarrow x \in \langle v_{r+1}, \dots, v_n \rangle \subseteq \mathbb{C}^n \subseteq \mathbb{C}^n.$$

■

(c) Sea $\widehat{A} = [u_1 | \dots | u_r] \in M(m \times r, \mathbb{C})$, donde $r = \text{rank}(A) < n < m$. Como se sabe, las columnas $u_1, \dots, u_r \in \mathbb{C}^m$ de U son ortonormales. Por otro lado, de (a) se sabe que $\text{Range}(A) = \langle u_1, \dots, u_r \rangle$. Luego, el proyector ortogonal P de \mathbb{C}^m sobre $\text{Range}(A)$ viene dado por $P = \widehat{U}.\widehat{U}^*$ (cf. [1, p. 45])

Cuando A es una matriz de rango completo, i.e. $r = \text{rank}(A) = n$, el proyector ortogonal P de \mathbb{C}^m sobre $\text{Range}(A)$ viene dado por $P = A.(A^*.A)^{-1}.A^*$. Como en el contexto de este ejercicio A no es de rango completo, la fórmula $P = A.(A^*.A)^{-1}.A^*$ actúa como un distractor para la resolución.

No obstante, si A es de rango completo, se recupera el resultado general $P = \widehat{U}.\widehat{U}^*$. En efecto, en este caso, reemplazando A por su SVD $A = U.\Sigma.V^*$ se obtiene:

$$\begin{aligned} P &= A.(A^*.A)^{-1}.A^* \\ &= U.\Sigma.V^*(V.\Sigma^*.U^*.U.\Sigma.V^*)^{-1}.V.\Sigma^*.U^* \\ &= U.\Sigma.V^*.V.(\Sigma^*.\Sigma)^{-1}.V^*.V.\Sigma^*.U^* \\ &= U.(\Sigma.(\Sigma^*.\Sigma)^{-1}.\Sigma^*).U^* \\ &= U. \left(\underbrace{\text{diag}(\sigma_1, \dots, \sigma_n)}_{\in M(m \times n, \mathbb{R})} \cdot \underbrace{\text{diag}(\sigma_1^{-2}, \dots, \sigma_n^{-2})}_{\in M(n \times n, \mathbb{R})} \cdot \underbrace{\text{diag}(\sigma_1, \dots, \sigma_n)}_{\in M(n \times m, \mathbb{R})} \right) .U^* \\ &= U.I_m.U^* \\ &= U.U^*, \end{aligned}$$

que coincide con el resultado general anotado más arriba.

■

REFERENCIAS BIBLIOGRÁFICAS

- [1] L.N. Trefethen and D. Bau III. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1997. A very good book indeed! More advanced than Strang's book.

Notas “sine qua non”: *Duración del examen: 90 minutos. El certamen debe ser resuelto individualmente usando un bolígrafo de tinta indeleble. ¡Buena suerte!*