MÉTODOS NUMÉRICOS Y PROGRAMACIÓN 2023/2024

Almacenamiento y manipulación de matrices (PRÁCTICA 3)

1.— Se desea calcular el producto matricial $\boldsymbol{K} = \boldsymbol{L}\boldsymbol{U}$ siendo $\boldsymbol{L}$ una matriz triangular inferior y $\boldsymbol{U}$ una matriz triangular superior, ambas de orden n. Se pide:

a) ¿Cuál es la forma de la matriz $\boldsymbol{K}$?

b) Diseñar los esquemas de almacenamiento mínimos para las tres matrices.

c) Escribir un algoritmo de multiplicación adaptado a los esquemas de almacenamiento anteriores.

d) Describir como crece el coste computacional (medido tanto en términos de la cantidad de memoria como del tiempo de cálculo requerido) en función del orden de las matrices. Compararlo con el que se derivaría de almacenar las matrices completas y utilizar un algoritmo de multiplicación para matrices llenas.

Sol. 1.

$$\boldsymbol{K} = \boldsymbol{L}\,\boldsymbol{U}, \qquad \boldsymbol{L} = \begin{bmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix}, \qquad \boldsymbol{U} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{bmatrix}$$

a) El producto de matrices se puede plantear como:

$$\boldsymbol{K} = [k_{ij}], \qquad k_{ij} = \sum_{m=1}^{n} l_{im} u_{mj}$$

pero hay que tener en cuenta que:

$$\begin{aligned} l_{im} &= 0 \quad \text{si } m > i \\ u_{mj} &= 0 \quad \text{si } m > j \end{aligned}$$

De este modo el producto se puede obtener como:

$$k_{ij} = \sum_{\substack{m\,=\,1,\,n \\ m \,\le\, i,\ m \,\le\, j}} l_{im}\ u_{mj} \ = \sum_{m=1}^{\text{mín}\{i,j\}} l_{im}\ u_{mj}$$

Se observa que en general $k_{ij} \neq 0$, pues siempre hay algún producto que aporta algo al valor de k_{ij}

Luego $\boldsymbol{K}$ es una matriz llena.

b) Almacenamos:

$\boldsymbol{L} \longrightarrow$ triangular inferior por filas
$\boldsymbol{U} \longrightarrow$ triangular inferior por columnas
$\boldsymbol{K} \longrightarrow$ llena por columnas

De modo que:

$$l_{ij} \rightsquigarrow vl(lpl) \quad \text{con} \quad lpl = \frac{i(i-1)}{2} + j; \qquad j \leq i$$

$$u_{ij} \rightsquigarrow vu(lpu) \quad \text{con} \quad lpu = \frac{j(j-1)}{2} + i; \qquad i \leq j$$

$$k_{ij} \rightsquigarrow vk(lpk) \quad \text{con} \quad lpk = (j-1)n + i$$

c)

```
do j=1,n
  lpk0=(j-1)*n
  lpu0=(j*(j-1))/2
  do i=1,n
    lpl0=(i*(i-1))/2
    lpk=lpk0+i
    vk(lpk)=0.
    do m=1,min(i,j)
      lpl=lpl0+m
      lpu=lpu0+m
      vk(lpk)=vk(lpk)+vl(lpl)*vu(lpu)
    enddo
  enddo
enddo
```

d) El algoritmo anterior requiere:

1) Almacenamiento $= \dfrac{n(n+1)}{2}$ componentes para $\boldsymbol{L}$

$\dfrac{n(n+1)}{2}$ componentes para $\boldsymbol{U}$

n^2 componentes para $\boldsymbol{K}$

Total $= 2n^2 + n \quad \Rightarrow \quad \mathcal{A}(2n^2+n) \approx \mathcal{A}(2n^2)$

2) Tiempo de computación $=$

$(2(n-1)+1)$ veces [1 producto y 1 suma] $\rightsquigarrow$

$$\begin{bmatrix} \bullet & \bullet & \cdots & \bullet & \bullet \\ \bullet & & & & \\ \vdots & & & & \\ \bullet & & & & \\ \bullet & & & & \end{bmatrix}$$

$(2(n-2)+1)$ veces [2 productos y 2 sumas] $\rightsquigarrow$

$$\begin{bmatrix} * & * & \cdots & * & * \\ * & \bullet & \cdots & \bullet & \bullet \\ \vdots & \vdots & & & \\ * & \bullet & & & \\ * & \bullet & & & \end{bmatrix}$$

$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$

$(2(n-n)+1)$ veces [(n) productos y (n) sumas] $\rightsquigarrow$

$$\begin{bmatrix} * & * & \cdots & * & * \\ * & * & \cdots & * & * \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ * & * & \cdots & * & * \\ * & * & \cdots & * & \bullet \end{bmatrix}$$

$$\begin{aligned} \text{Total} &= \sum_{d=1}^{n} (2(n-d)+1)\,[\ (d)\ \text{“productos”}\ +(d)\ \text{“sumas”}\ ] \\ &= \sum_{d=1}^{n} (2(n-d)+1)\ (2d) \qquad \text{OCF} \quad \text{(Operaciones en coma flotante)} \\ &= \left(\sum_{d=1}^{n} 2d(2n+1) - \sum_{d=1}^{n} (2d)^2\right)\ \text{OCF} = \left((2n+1)(n+1)n - \frac{2}{3}(2n+1)(n+1)n\right)\ \text{OCF} \\ &= \left(\frac{2n^3}{3} + \frac{3n^2}{3} + \frac{n}{3}\right)\ \text{OCF}\ \Rightarrow T\left(\frac{2n^3}{3} + \frac{3n^2}{3} + \frac{n}{3}\right) \approx T\left(\frac{2n^3}{3}\right) \end{aligned}$$

Si se hubiese trabajado con matrices llenas habrían hecho falta:

$3n^2$ componentes para $\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{K} \Rightarrow\ \mathcal{A}(3n^2)$
$2n^3$ OCF para calcular $\boldsymbol{K}\ \Rightarrow\ T(2n^3)$

Luego se ahorra aproximadamente $\begin{cases} 1/3 \text{ de la memoria} \\ 2/3 \text{ del tiempo de computación} \end{cases}$

2.— Repetir el problema anterior cuando las matrices $\boldsymbol{L}$ y $\boldsymbol{U}$ tienen semianchos de banda l y u respectivamente, siendo $l << n$, y $u << n$.

Sol. 2.

$$\boldsymbol{L} = \begin{bmatrix} l_{11} & & & & & & & \\ l_{21} & l_{22} & & & & & & \\ \vdots & \vdots & \ddots & & & & & \\ l_{l+1,1} & l_{l+1,2} & \cdots & l_{l+1,l+1} & & & & \\ & \ddots & \ddots & & \ddots & & & \\ & & l_{i,i-l} & l_{i,i-l+1} & & l_{i,i} & & \\ & & & \ddots & \ddots & & \ddots & \\ & & & & l_{n,n-l} & l_{n,n-l+1} & \cdots & l_{n,n} \end{bmatrix}$$

$$\boldsymbol{U} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1,u+1} & & & & \\ & u_{22} & \cdots & u_{2,u+1} & \ddots & & & \\ & & \ddots & \vdots & \ddots & u_{j-u,j} & & \\ & & & u_{u+1,u+1} & u_{j-u+1,j} & \ddots & & \\ & & & & \ddots & \vdots & \ddots & u_{n-u,n} \\ & & & & & u_{j,j} & & u_{n-u+1,n} \\ & & & & & & \ddots & \vdots \\ & & & & & & & u_{n,n} \end{bmatrix}$$

a) $\boldsymbol{K} = [k_{ij}]; \quad k_{ij} = \sum_{m=1}^{n} l_{im} u_{mj}$

pero

$$\begin{array}{rcccc} l_{im} = 0 & \text{si} & m > i & \text{ó} & m < i - l \\ u_{mj} = 0 & \text{si} & m > j & \text{ó} & m < j - u \end{array}$$

De modo que:

$$K_{ij} = \sum_{\substack{m\,=\,1,\,n \\ i-l \,\leq\, m \,\leq\, i \\ j-u \,\leq\, m \,\leq\, j}} l_{im} u_{mj} \qquad = \sum_{m \,=\, \text{máx}\{i-l,\, j-u,\, 1\}}^{\text{mín}\{i,\ j\}} l_{im} u_{mj}$$

Luego cuando mín$\{i,\ j\}$ < máx$\{i-l, j-u, 1\}$ no habrá ningún producto que aporte algo al valor de k_{ij}. Por tanto, sólo serán no nulos (en principio) los coeficientes k_{ij} tales que mín$\{i,\ j\} \geq$ máx$\{i-l,\ j-u,\ 1\}$, es decir tales que $i \geq i-l$, $i \geq j-u$, $j \geq i-l$, $j \geq j-u$.

Como $i \geq i-l$ y $j \geq j-u$ se cumplen siempre entonces:

$$k_{ij} \text{ puede ser no nulo } \Longleftrightarrow \begin{cases} i \geq j-u \;\Rightarrow\; \text{semiancho de banda superior (u)} \\ j \geq i-l \;\Rightarrow\; \text{semiancho de banda inferior (l)} \end{cases}$$

Luego $\boldsymbol{K}$ es una matriz en banda cuyo semiancho de banda inferior es el de $\boldsymbol{L}$ y cuyo semiancho de banda superior es el $\boldsymbol{U}$.

$$\boldsymbol{K} = \begin{bmatrix} \ddots & * & \cdots & * & & & & & & & \\ * & \ddots & * & \cdots & * & & & & & & \\ \vdots & * & \ddots & * & \cdots & * & & & & & \\ \vdots & \vdots & * & \ddots & * & \cdots & * & & & & \\ * & \vdots & \vdots & * & \ddots & * & \cdots & * & & & \\ & * & \vdots & \vdots & * & \ddots & * & \cdots & * & & \\ & & * & \vdots & \vdots & * & \ddots & * & \cdots & * & \\ & & & * & \vdots & \vdots & * & \ddots & * & \cdots & * \\ & & & & * & \vdots & \vdots & * & \ddots & * & \vdots \\ & & & & & * & \vdots & \vdots & * & \ddots & * \\ & & & & & & * & \cdots & \cdots & * & \ddots \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ l \\ \\ \\ \\ \\ u \\ \\ \end{matrix}$$

b) Almacenamos las matrices $\boldsymbol{L}$, $\boldsymbol{U}$, $\boldsymbol{K}$ en banda (por diagonales)

$$l_{ij} \rightsquigarrow vl(lpl) \quad \text{con} \quad lpl = (j-i+l)n+i; \quad i-l \leq j \leq i$$

$$u_{ij} \rightsquigarrow vu(lpu) \quad \text{con} \quad lpu = (j-i)n+i; \quad j-u \leq i \leq j$$

$$k_{ij} \rightsquigarrow vk(lpk) \quad \text{con} \quad lpk = (j-i+l)n+i; \quad \begin{cases} i-l \leq j \leq i \\ j-u \leq i \leq j \end{cases}$$

c)

```
do j=1,n
  do i=max(1,j-u), min(n,j+l)
    lpk=(j-i+l)*n+i
    vk(lpk)=0.d+00
    do k=max(i-l,j-u,1), min(i,j)
      lpl=(k-i+l)*n+i
      lpu=(j-k)*n+k
      vk(lpk)=vk(lpk)+vl(lpl)*vu(lpu)
    enddo
  enddo
enddo
```

d) El algoritmo requiere:

1) Almacenamiento = $n(l+1)$ componentes para $\boldsymbol{L}$

$n(u+1)$ componentes para $\boldsymbol{U}$

$n(l+1+u)$ componentes para $\boldsymbol{K}$

Total = $2n(l+1+u)+n \quad \Rightarrow \quad \mathcal{A}(2n(l+1+u)+n) \;=\; \mathcal{A}(2n(l+u+1{,}5))$

2) Tiempo de computación

El cálculo exacto es complicado, pero sabemos que $l << n$ y $u << n$

Calculamos lo que sucede en las filas y columnas centrales, y suponemos que todas se comportan igual (lo que es aceptable, pues las únicas anómalas serían las $(l+1)$ primeras filas y las $(1+u)$ últimas columnas)

Para calcular las $(l+1+u)$ componentes de la fila i es preciso realizar:

$$\begin{cases} (1) \text{ producto} + (1) \text{ suma} \\ (2) \text{ productos} + (2) \text{ sumas} \\ \quad \vdots \\ (\text{mín}(l,\, u){+}1) \text{ productos} + (\text{mín}(l,\, u){+}1) \text{ sumas} \\ \quad \vdots \\ (\text{mín}(l,\, u){+}1) \text{ productos} + (\text{mín}(l,\, u){+}1) \text{ sumas} \\ \quad \vdots \\ (2) \text{ productos} + (2) \text{ sumas} \\ (1) \text{ producto} + (1) \text{ suma} \end{cases}$$

Luego el número de operaciones por fila es:

$$(l+1+u-\text{mín}(l,u))\; 2(\text{mín}(l,u)+1) \quad \text{OCF}$$

$$2(\text{máx}(l,u)+1)\; (\text{mín}(l,u)+1) \quad \text{OCF}$$

$$2(l+1)(u+1) \quad \text{OCF}$$

Luego en total el tiempo de computación será aproximadamente proporcional a $T(2n(l{+}1)(u{+}1))$

Recordamos que para matrices llenas habrían hecho falta:

$$\begin{cases} 3n^2 \text{ componentes} & \text{para } \boldsymbol{L},\, \boldsymbol{U},\, \boldsymbol{K} \quad \Longrightarrow \quad \mathcal{A}(3n^2) \\ 2n^3 \text{ OCF} & \text{para calcular } \boldsymbol{K} \quad \Longrightarrow \quad T(2n^3) \end{cases}$$

Luego cuando $l << n$ y $u << n$ se ahorra mucho espacio de almacenamiento y mucho tiempo de computación (sobre todo tiempo).

3.— De la matriz $\boldsymbol{L}$ se sabe que es triangular inferior y que todos sus elementos son nulos, excepto los situados en la diagonal principal (que son siempre distintos de cero) y los situados en la fila α (que en general serán distintos de cero, aunque no necesariamente), esto es:

$$\boldsymbol{L} = \begin{bmatrix} l_{11} & & & & & & & \\ & l_{22} & & & & & & \\ & & \ddots & & & & & \\ l_{\alpha 1} & l_{\alpha 2} & \dots & l_{\alpha\alpha} & & & & \\ & & & & \ddots & & & \\ & & & & & l_{\beta\beta} & & \\ & & & & & & \ddots & \\ & & & & & & & l_{nn} \end{bmatrix}.$$

Se desea calcular la matriz producto $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$.

Se pide:

a) ¿Podemos asegurar que $\boldsymbol{A}$ es simétrica? ¿Y que es definida positiva? ¿Por qué?

b) ¿Cuál es la forma general de la matriz $\boldsymbol{A}$?

c) Desarrollar los esquemas de almacenamiento que se consideren más adecuados para las dos matrices. A medida que se va calculando la matriz $\boldsymbol{A}$, ¿es posible almacenar sus coeficientes en el lugar ocupado por los correspondientes de la matriz $\boldsymbol{L}$ para ahorrar espacio en memoria? ¿Por qué?

d) Escribir un algoritmo adaptado para realizar el producto matricial con los esquemas de almacenamiento descritos en el apartado anterior.

e) ¿Como crece el tiempo de computación necesario para calcular $\boldsymbol{A}$ a medida que aumenta el orden de la matriz?

Sol. 3.

a) $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$ con $det(\boldsymbol{L}) = \prod_{i=1}^{n} l_{ii} \neq 0$ es

$$\begin{cases} \text{1) Simétrica, pues } \boldsymbol{A}^T = (\boldsymbol{L}\boldsymbol{L}^T)^T = (\boldsymbol{L}^T)^T\,\boldsymbol{L}^T = \boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{A} \\ \text{2) Semidefinida positiva, pues } \boldsymbol{v}^T\boldsymbol{A}\boldsymbol{v} = \boldsymbol{v}^T(\boldsymbol{L}\boldsymbol{L}^T)\boldsymbol{v} = \\ \qquad (\boldsymbol{v}^T\boldsymbol{L})(\boldsymbol{L}^T\boldsymbol{v}) = \underbrace{(\boldsymbol{L}^T\boldsymbol{v})^T}_{\boldsymbol{w}^T}\underbrace{(\boldsymbol{L}^T\boldsymbol{v})}_{\boldsymbol{w}} = \boldsymbol{w}^T\boldsymbol{w} \geq 0 \\ \text{3) Definida positiva, pues } \boldsymbol{v}^T\boldsymbol{A}\boldsymbol{v} = 0 \Longleftrightarrow \boldsymbol{w} = \boldsymbol{L}^T\boldsymbol{v} = \boldsymbol{0} \overset{det(\boldsymbol{L})\neq 0}{\Longrightarrow} \boldsymbol{v} = \boldsymbol{0} \end{cases}$$

b)

$$\boldsymbol{L} = \begin{bmatrix} l_{11} & & & & & \\ & l_{22} & & & & \\ & & \ddots & & & \\ l_{\alpha 1} & l_{\alpha 2} & \cdots & l_{\alpha\alpha} & & \\ & & & & \ddots & \\ & & & & & l_{nn} \end{bmatrix} \qquad \boldsymbol{L}^T = \begin{bmatrix} l_{11} & & & l_{\alpha 1} & & \\ & l_{22} & & l_{\alpha 2} & & \\ & & \ddots & \vdots & & \\ & & & l_{\alpha\alpha} & & \\ & & & & \ddots & \\ & & & & & l_{nn} \end{bmatrix}$$

Luego

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T = \begin{bmatrix} l_{11}^2 & & & & l_{11}l_{\alpha 1} & & & \\ & l_{22}^2 & & & l_{22}l_{\alpha 2} & & & \\ & & \ddots & & \vdots & & & \\ & & & l_{\alpha-1,\alpha-1}^2 & l_{\alpha-1,\alpha-1}l_{\alpha,\alpha-1} & & & \\ & & & & \sum_{i=1}^{\alpha} l_{\alpha i}^2 & & & \\ & \text{SIM.} & & & & l_{\alpha+1,\alpha+1}^2 & & \\ & & & & & & \ddots & \\ & & & & & & & l_{nn}^2 \end{bmatrix}$$

Por tanto todos los elementos de $\boldsymbol{A}$ son nulos excepto la diagonal principal, la fila α (parte inferior) y la columna α (parte superior)

c) Como $\boldsymbol{A}$ es simétrica basta con almacenar la parte inferior. Para almacenar la matriz $\boldsymbol{L}$ y la parte inferior de $\boldsymbol{A}$ bastan sendos vectores de $n + (\alpha - 1)$ componentes:

$$\boldsymbol{vl} \equiv (l_{11},\, l_{22},\, \cdots,\, l_{\alpha-1,\alpha-1},\, l_{\alpha 1},\, l_{\alpha 2},\, \cdots\, l_{\alpha\alpha},\, l_{\alpha+1,\alpha+1},\, \cdots,\, l_{nn})$$

$$\boldsymbol{va} \equiv (a_{11},\, a_{22},\, \cdots,\, a_{\alpha-1,\alpha-1},\, a_{\alpha 1},\, a_{\alpha 2},\, \cdots\, a_{\alpha\alpha},\, a_{\alpha+1,\alpha+1},\, \cdots,\, a_{nn})$$

Sí es posible almacenar $\boldsymbol{A}$ sobre $\boldsymbol{L}$, aunque las componentes de $\boldsymbol{L}$ (hasta el $l_{\alpha\alpha}$ son necesarios para calcular varios coeficientes de $\boldsymbol{A}$ (por ejemplo, l_{11} se utiliza para calcular a_{11} y $a_{\alpha 1}$) por lo que el orden de las operaciones es importante. No se pueden programar las operaciones de cualquier forma. Hay que seguir un orden.

d)

do $i = 1, \alpha - 1$		`do i=1,ia-1`
$a_{ii} = l_{ii}^2$	$\rightsquigarrow$	`va(i) = vl(i) * vl(i)`
$a_{\alpha i} = l_{ii}\, l_{\alpha i}$	$\rightsquigarrow$	`va(ia-1+i) = vl(i) * vl(ia-1+i)`
enddo		`enddo`
		`va(2*ia-1) = 0.d+00`
		`do k=1,ia`
$a_{\alpha\alpha} = \sum_{k=1}^{\alpha} l_{\alpha k}^2$	$\rightsquigarrow$	`va(2*ia-1) = va(2*ia-1) + vl(ia-1+k) * vl(ia-1+k)`
		`enddo`
do $i = \alpha + 1, n$		`do i=ia+1,n`
$a_{ii} = l_{ii}^2$	$\rightsquigarrow$	`va(ia-1+i) = vl(ia-1+i) * vl(ia-1+i)`
enddo		`enddo`

e) El número de operaciones en coma flotante que hay que realizar para obtener $\boldsymbol{A}$ es:

$$\begin{array}{rcl} & & (n-1) \text{ productos} \\ & & (\alpha - 1) \text{ productos} \\ & & (\alpha) \text{ productos} + (\alpha) \text{ sumas} \\ \text{Total} & = & (n + 2\alpha - 2) \text{ productos} + (\alpha) \text{ sumas} \\ & = & (n + 3\alpha - 2) \text{ OCF} \implies \mathrm{T}(n + (3\alpha - 2)) \end{array}$$

Luego el tiempo de computación crece linealmente con el orden de la matriz

---

4.— De las matrices **L** y **U** se sabe que son triangular inferior y triangular superior respectivamente, y que todos sus elementos son nulos excepto los situados en la diagonal principal, en la fila α de **L** y en la columna β de **U**. Esto es:

$$\boldsymbol{L} = \begin{bmatrix} l_{11} & & & & & & & \\ & l_{22} & & & & & & \\ & & \ddots & & & & & \\ l_{\alpha 1} & l_{\alpha 2} & \dots & l_{\alpha\alpha} & & & & \\ & & & & \ddots & & & \\ & & & & & l_{\beta\beta} & & \\ & & & & & & \ddots & \\ & & & & & & & l_{nn} \end{bmatrix}, \boldsymbol{U} = \begin{bmatrix} u_{11} & & & & & u_{1\beta} & & \\ & u_{22} & & & & u_{2\beta} & & \\ & & \ddots & & & \vdots & & \\ & & & u_{\alpha\alpha} & & \vdots & & \\ & & & & \ddots & \vdots & & \\ & & & & & u_{\beta\beta} & & \\ & & & & & & \ddots & \\ & & & & & & & u_{nn} \end{bmatrix}$$

Se pide:

a) ¿Cuál es la forma general de la matriz producto $\boldsymbol{A} = \boldsymbol{LU}$?

b) Desarrollar los esquemas de almacenamiento que se consideren más adecuados para las tres matrices.

c) Escribir un algoritmo adaptado a los esquemas de almacenamiento descritos en el apartado anterior para realizar el producto matricial.

Sol. 4.

a) Se trata de un caso particular de producto de matrices en perfil.

Sabemos que $\boldsymbol{A} = \boldsymbol{LU}$ tiene el mismo perfil inferior que $\boldsymbol{L}$ y el mismo perfil superior que $\boldsymbol{U}$, es decir que los únicos elementos de $\boldsymbol{A}$ distintos de cero serán:

$$\begin{array}{ll} a_{ii}; & i = 1, \dots, n \\ a_{\alpha j}; & j = 1, \dots, \alpha - 1 \\ a_{i\beta}; & i = 1, \dots, \beta - 1 \end{array}$$

Luego la matriz producto tiene el siguiente aspecto:

$$\begin{cases} \text{Caso } \alpha < \beta \quad \rightsquigarrow \quad \boldsymbol{A} = \begin{bmatrix} \ddots & & \vdots & \\ \cdots & \ddots & \vdots & \\ & & \ddots & \\ & & & \ddots \end{bmatrix} \\ \text{Caso } \alpha = \beta \quad \rightsquigarrow \quad \boldsymbol{A} = \begin{bmatrix} \ddots & & \vdots & \\ & \ddots & \vdots & \\ \cdots & \cdots & \ddots & \\ & & & \ddots \end{bmatrix} \\ \text{Caso } \alpha > \beta \quad \rightsquigarrow \quad \boldsymbol{A} = \begin{bmatrix} \ddots & \vdots & & \\ & \ddots & & \\ \cdots & \cdots & \ddots & \\ & & & \ddots \end{bmatrix} \end{cases}$$

b) Para almacenar las matrices $\boldsymbol{L}$ y $\boldsymbol{U}$ necesitamos 2 vectores de $(n + (\alpha - 1))$ y $(n + (\beta - 1))$ componentes respectivamente:

$$\boldsymbol{vl} = (l_{11},\, l_{22},\, \ldots,\, l_{\alpha-1,\alpha-1},\, l_{\alpha 1},\, l_{\alpha 2},\, \ldots,\, l_{\alpha,\alpha},\, l_{\alpha+1,\alpha+1},\, \ldots,\, l_{nn})$$
$$\boldsymbol{vu} = (u_{11},\, u_{22},\, \ldots,\, u_{\beta-1,\beta-1},\, u_{1\beta},\, u_{2\beta},\, \ldots,\, u_{\beta\beta},\, u_{\beta+1,\beta+1},\, \ldots,\, u_{nn})$$

Para almacenar la matriz $\boldsymbol{A}$ hace falta un vector de $(n + (\alpha - 1) + (\beta - 1))$ componentes.

Podemos ordenar los datos, por ejemplo, en la forma:

$$\boldsymbol{va} = (a_{11},\, a_{22},\, \ldots,\, a_{nn},\, a_{\alpha 1},\, \ldots,\, a_{\alpha,\alpha-1},\, a_{1\beta},\, \ldots,\, a_{\beta-1,\beta})$$

c) Este problema se resuelve igual que el problema anterior.

Tenemos que multiplicar $\boldsymbol{LU}$ e identificar los valores de los coeficientes de $\boldsymbol{A}$.

Hay que tener en cuenta los tres casos posibles ($\alpha < \beta$, $\alpha = \beta$, $\alpha > \beta$).

Finalmente se sustituyen los coeficientes l_{ik}, u_{kj} y a_{ij} por sus referencias de acuerdo con el esquema planteado en el ejercicio anterior.