



- 2.— Escribir un algoritmo de resolución para sistemas lineales de la forma  $\mathbf{K} \mathbf{u} = \mathbf{f}$ , siendo  $\mathbf{K}$  una matriz simétrica y definida positiva, de semiancho de banda  $s$ , almacenada en un vector. Utilícese el esquema de almacenamiento y el método de solución más adecuados.

**Solución 2.**

En este caso y dado que la matriz del sistema es simétrica y definida positiva, el método más adecuado es utilizar una factorización de Cholesky y resolver el sistema  $\mathbf{A} \mathbf{x} = \mathbf{b}$  de la siguiente forma:

$$\mathbf{A} = \mathbf{L} \mathbf{D} \mathbf{L}^t \rightarrow \mathbf{L} \underbrace{(\mathbf{D} (\mathbf{L}^t \mathbf{x}))}_{\mathbf{z}} = \mathbf{b} \rightarrow \begin{cases} \mathbf{L} \mathbf{z} = \mathbf{b} \\ \mathbf{D} \mathbf{y} = \mathbf{z} \\ \mathbf{L}^t \mathbf{x} = \mathbf{y} \end{cases}$$

La factorización en su planteamiento general de una matriz  $\mathbf{A}$  se plantea de la siguiente forma. Sean:

$$\mathbf{A}_k = \begin{bmatrix} a_{11} & \dots & a_{1k} \\ \vdots & & \vdots \\ a_{k1} & \dots & a_{kk} \end{bmatrix} ; \quad \mathbf{A}_{k+1} = \left[ \begin{array}{c|c} \mathbf{A}_k & \mathbf{f}_{k+1} \\ \hline \mathbf{f}_{k+1}^t & a_{k+1,k+1} \end{array} \right] ; \quad \text{con } \mathbf{f}_{k+1}^t = [a_{k+1,1}, \dots, a_{k+1,k}]$$

$$\mathbf{L}_k = \begin{bmatrix} l_{11} & \dots & l_{1k} \\ \vdots & & \vdots \\ l_{k1} & \dots & l_{kk} \end{bmatrix} ; \quad \mathbf{L}_{k+1} = \left[ \begin{array}{c|c} \mathbf{L}_k & \mathbf{0} \\ \hline \mathbf{l}_{k+1}^t & l_{k+1,k+1} \end{array} \right] ; \quad \text{con } \mathbf{l}_{k+1}^t = [l_{k+1,1}, \dots, l_{k+1,k}]$$

$$\mathbf{D}_k = \begin{bmatrix} d_{11} & & \\ & \ddots & \\ & & d_{kk} \end{bmatrix} ; \quad \mathbf{D}_{k+1} = \left[ \begin{array}{c|c} \mathbf{D}_k & \mathbf{0} \\ \hline \mathbf{0} & d_{k+1,k+1} \end{array} \right]$$

Si suponemos conocidos  $\mathbf{L}_k$  y  $\mathbf{D}_k$  tales que  $\mathbf{A}_k = \mathbf{L}_k \mathbf{D}_k \mathbf{L}_k^t$ , podemos imponer que se cumpla  $\mathbf{A}_{k+1} = \mathbf{L}_{k+1} \mathbf{D}_{k+1} \mathbf{L}_{k+1}^t$ , es decir:

$$\left[ \begin{array}{c|c} \mathbf{A}_k & \mathbf{f}_{k+1} \\ \hline \mathbf{f}_{k+1}^t & a_{k+1,k+1} \end{array} \right] = \left[ \begin{array}{c|c} \mathbf{L}_k \mathbf{D}_k \mathbf{L}_k^t & \mathbf{L}_k \mathbf{D}_k \mathbf{l}_{k+1} \\ \hline \mathbf{l}_{k+1}^t \mathbf{D}_k \mathbf{L}_k^t & \mathbf{l}_{k+1}^t \mathbf{D}_k \mathbf{l}_{k+1} + l_{k+1,k+1}^2 d_{k+1,k+1} \end{array} \right]$$

Por tanto, se tiene que cumplir:

$$\begin{cases} \mathbf{L}_k \mathbf{D}_k \mathbf{l}_{k+1} = \mathbf{f}_{k+1} \\ a_{k+1,k+1} = \mathbf{l}_{k+1}^t \mathbf{D}_k \mathbf{l}_{k+1} + l_{k+1,k+1}^2 d_{k+1,k+1} \end{cases}$$

luego:

- $\mathbf{l}_{k+1}$  se obtiene resolviendo el sistema  $(\mathbf{L}_k \mathbf{D}_k) \mathbf{l}_{k+1} = \mathbf{f}_{k+1}$
- $l_{k+1,k+1} = 1$  de forma arbitraria.
- $d_{k+1,k+1} = a_{k+1,k+1} - \mathbf{l}_{k+1}^t \mathbf{D}_k \mathbf{l}_{k+1}$

Para  $k = 1$  se obtiene directamente que  $d_{11} = a_{11}$  y  $l_{11} = 1$ . Por tanto el algoritmo general de factorización para una matriz simétrica será:

$$l_{11} = 1 \quad ; \quad d_{11} = a_{11}$$

do  $k = 1, n - 1$

$$l_{k+1,i} = a_{k+1,i} - \sum_{j=1}^{i-1} l_{ij} l_{k+1,j}; \quad i = 1, \dots, k$$

$$l_{k+1,i} = l_{k+1,i} / d_{ii} \quad i = 1, \dots, k$$

$$l_{k+1,k+1} = 1$$

$$d_{k+1,k+1} = a_{k+1,k+1} - \sum_{j=1}^k l_{k+1,j}^2 d_{jj}$$

enddo

Y la resolución del sistema:

$$z_i = b_i - \sum_{j=1}^{i-1} l_{ij} z_j \quad ; \quad i = 1, \dots, n$$

$$y_i = z_i / d_{ii} \quad ; \quad i = 1, \dots, n$$

$$x_i = y_i - \sum_{j=1+1}^n l_{ji} x_j \quad ; \quad i = n, \dots, 1 \quad (\text{Hacia atrás})$$

Dado que la diagonal de  $\mathbf{L}$  y los elementos de  $\mathbf{A}$  no se vuelven a utilizar, es posible almacenar la información de las matrices  $\mathbf{L}$  y  $\mathbf{D}$  sobre  $\mathbf{A}$ . Lo mismo ocurre con los vectores  $\mathbf{z}$ ,  $\mathbf{y}$  y  $\mathbf{x}$  y el término independiente del sistema  $\mathbf{b}$ .

Teniendo en cuenta lo anterior, la factorización y la resolución del sistema resultarían:

do  $k = 1, \dots, n - 1$

$$a_{k+1,i} = a_{k+1,i} - \sum_{j=1}^{i-1} a_{ij} a_{k+1,j}; \quad i = 2, \dots, k$$

$$a_{k+1,i} = a_{k+1,i} / a_{ii} \quad i = 1, \dots, k$$

$$a_{k+1,k+1} = a_{k+1,k+1} - \sum_{j=1}^k a_{k+1,j}^2 a_{jj}$$

enddo

$$b_i = b_i - \sum_{j=1}^{i-1} a_{ij} b_j \quad ; \quad i = 2, \dots, n$$

$$b_i = b_i / a_{ii} \quad ; \quad i = 1, n$$

$$b_i = b_i - \sum_{j=1+1}^n a_{ji} b_j \quad ; \quad i = n-1, \dots, 1 \quad (\text{Hacia atrás})$$

Se puede comprobar además que el semiancho de banda de la matriz  $\mathbf{L}$  es el mismo que el de  $\mathbf{A}$ . Por tanto, tanto antes como después de la factorización, los elementos  $a_{ij}$  fuera de la banda son nulos. Modificando los bucles para no operar con esos elementos obtenemos:

do  $k = 1, n-1$

$$a_{k+1,i} = a_{k+1,i} - \sum_{j=\max\{i-s, k+1-s, 1\}}^{i-1} a_{ij} a_{k+1,j} \quad ; \quad i = \max\{k+1-s+1, 2\}, \dots, k$$

$$a_{k+1,i} = a_{k+1,i} / a_{ii} \quad ; \quad i = \max\{k+1-s, 1\}, \dots, k$$

$$a_{k+1,k+1} = a_{k+1,k+1} - \sum_{j=\max\{k+1-s, 1\}}^k a_{k+1,j}^2 a_{jj}$$

enddo

$$b_i = b_i - \sum_{j=\max\{i-s, 1\}}^{i-1} a_{ij} b_j \quad ; \quad i = 2, \dots, n$$

$$b_i = b_i / a_{ii} \quad ; \quad i = 1, \dots, n$$

$$b_i = b_i - \sum_{j=1+1}^{\min\{i+s, n\}} a_{ji} b_j \quad ; \quad i = n-1, \dots, 1 \quad (\text{Hacia atrás})$$

Además, podemos almacenar a su vez la matriz en banda  $\mathbf{A}$  en un vector de forma que los elementos  $a_{\alpha,\beta}$  se almacenen en un vector  $v_\gamma$  con  $\gamma = (\alpha-1)(1+s) + (\beta-\alpha+1+s) = \beta + \alpha s$ . De esta forma:

do  $k = 1, n-1$

$$v_{(k+1)s+i} = v_{(k+1)s+i} - \sum_{j=\max\{i-s, k+1-s, 1\}}^{i-1} v_{is+j} v_{(k+1)s+j} \quad ; \quad i = \max\{k+1-s+1, 2\}, \dots, k$$

$$v_{(k+1)s+i} = v_{(k+1)s+i} / v_{is+i} \quad ; \quad i = \max\{k+1-s, 1\}, \dots, k$$

$$v_{(k+1)s+(k+1)} = v_{(k+1)s+(k+1)} - \sum_{j=\max\{k+1-s, 1\}}^k v_{(k+1)s+j}^2 v_{js+j}$$

enddo

$$\begin{aligned}
 b_i &= b_i - \sum_{j=\max\{i-s,1\}}^{i-1} v_{is+j} b_j & ; & \quad i = 2, \dots, n \\
 b_i &= b_i / v_{is+i} & ; & \quad i = 1, \dots, n \\
 b_i &= b_i - \sum_{j=1+1}^{\min\{i+s,n\}} v_{js+i} b_j & ; & \quad i = n-1, \dots, 1 \quad (\text{Hacia atrás})
 \end{aligned}$$


---

3.— Generalizar el algoritmo anterior para la resolución de  $m$  sistemas de ecuaciones con la misma matriz y diferentes términos independientes  $\{\mathbf{f}^1, \mathbf{f}^2, \dots, \mathbf{f}^m\}$ , hallándose éstos almacenados en un vector.

---

**Solución 3.**

Nos piden resolver  $m$  sistemas de ecuaciones de la forma:

$$\mathbf{A} \mathbf{x}^p = \mathbf{b}^p \quad p = 1, \dots, m$$

Dado que la matriz es la misma, la factorización es igual a la del problema anterior y solamente se realiza una vez. Luego solo es necesario repetir  $m$  veces la solución de sistemas.

Si almacenamos los  $m$  términos independientes en un único vector  $\mathbf{g}$ , las componentes  $f_\alpha^p$  se almacenarán en  $g_\gamma$  con  $\gamma = (p-1)n + \alpha$ . El algoritmo de resolución sería por tanto:

do  $p = 1, \dots, m$

$$\begin{aligned}
 b_{(p-1)n+i} &= b_{(p-1)n+i} - \sum_{j=\max\{i-s,1\}}^{i-1} v_{is+j} b_{(p-1)n+j} & ; & \quad i = 2, \dots, n \\
 b_{(p-1)n+i} &= b_{(p-1)n+i} / v_{is+i} & ; & \quad i = 1, \dots, n \\
 b_{(p-1)n+i} &= b_{(p-1)n+i} - \sum_{j=1+1}^{\min\{i+s,n\}} v_{js+i} b_{(p-1)n+j} & ; & \quad i = n-1, \dots, 1 \quad (\text{Hacia atrás})
 \end{aligned}$$

enddo

---

4.— Para analizar un cierto problema de ingeniería es preciso resolver sistemas de ecuaciones lineales  $\mathbf{Ax} = \mathbf{b}$ , donde la matriz  $\mathbf{A}$  es simétrica, definida positiva y tiene la forma:

$$\mathbf{A} = \begin{bmatrix}
 a_{11} & & & & & & & \\
 a_{21} & a_{22} & & & & & & \\
 a_{31} & 0 & a_{33} & & & & & \\
 a_{41} & 0 & 0 & a_{44} & & & & \\
 a_{51} & 0 & 0 & 0 & a_{55} & & & \\
 \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & & \\
 a_{n1} & 0 & \dots & \dots & \dots & 0 & a_{nn} & 
 \end{bmatrix} \quad \begin{aligned}
 & 1000 \leq n \leq 5000 \\
 & |a_{ii}| \gg |a_{ij}| \quad \forall j \neq i
 \end{aligned}$$



De esta forma el sistema de ecuaciones sigue siendo el mismo pero la matriz del sistema para a ser simétrica en perfil, es decir, únicamente hay que almacenar  $2n - 1$  componentes y el tiempo de computación pasa a ser  $T(n)$ .

5.— Deducir la condición suficiente para que el algoritmo iterativo

$$\mathbf{B}^{k+1} = \mathbf{B}^k [2\mathbf{I} - \mathbf{A}\mathbf{B}^k]; \quad k = 0, \dots$$

converja a la inversa de la matriz  $\mathbf{A}$ , siendo  $\mathbf{B}^0$  una aproximación inicial a  $\mathbf{A}^{-1}$  tal que:

$$\mathbf{A}\mathbf{B} = \mathbf{I} + \mathbf{E}; \quad (\mathbf{E} = \text{matriz de error})$$

### Solución 5.

Definimos el error que se comete en la iteración  $k$  como:

$$\mathbf{E}_k = \mathbf{A}\mathbf{B}_k - \mathbf{I}$$

por tanto:

$$\begin{aligned} \mathbf{E}_{k+1} &= \mathbf{A}\mathbf{B}_{k+1} - \mathbf{I} = \mathbf{A}\mathbf{B}_k [2\mathbf{I} - \mathbf{A}\mathbf{B}_k] - \mathbf{I} \\ &= (\mathbf{E}_k + \mathbf{I}) [2\mathbf{I} - (\mathbf{E}_k + \mathbf{I})] - \mathbf{I} \\ &= (\mathbf{E}_k + \mathbf{I}) [\mathbf{E}_k + \mathbf{I}] - \mathbf{I} \\ &= -(\mathbf{E}_k)^2 + \mathbf{E}_k - \mathbf{E}_k + \mathbf{I} - \mathbf{I} \\ &= -(\mathbf{E}_k)^2 \end{aligned}$$

así, el error:

$$\begin{aligned} \mathbf{E}_0 &= \mathbf{E} \\ \mathbf{E}_1 &= -(\mathbf{E}_0)^2 = -\mathbf{E}^2 \\ \mathbf{E}_2 &= -(\mathbf{E}_1)^2 = -\mathbf{E}^4 \\ \mathbf{E}_3 &= -(\mathbf{E}_2)^2 = -\mathbf{E}^8 \\ &\vdots \\ \mathbf{E}_k &= -(\mathbf{E}_{k-1})^2 = -\mathbf{E}^{(2^k)} \end{aligned}$$

La condición de convergencia será:

$$\lim_{k \rightarrow \infty} \mathbf{B}_k = \mathbf{A}^{-1} \Leftrightarrow \lim_{k \rightarrow \infty} \mathbf{E}_k = \mathbf{0} \Leftrightarrow \rho(\mathbf{E}) < 1$$

siendo  $\rho(\mathbf{E})$  el radio espectral de la matriz  $\mathbf{E}$

Podemos comprobar además el orden de convergencia:

$$\begin{aligned}(\mathbf{B}_{k+1} - \mathbf{A}^{-1}) &= \mathbf{A}^{-1}(\mathbf{A} \mathbf{B}_{k+1} - \mathbf{I}) = \mathbf{A}^{-1} \mathbf{E}_{k+1} = -\mathbf{A}^{-1}(\mathbf{E}_k)^2 \\ &= -\mathbf{A}^{-1}(\mathbf{A} \mathbf{B}_k - \mathbf{I})^2 = -\mathbf{A}^{-1}(\mathbf{A}(\mathbf{B}_k - \mathbf{A}^{-1}))^2 \\ &= -\mathbf{A}^{-1} \mathbf{A}(\mathbf{B}_k - \mathbf{A}^{-1})\mathbf{A}(\mathbf{B}_k - \mathbf{A}^{-1}) = -(\mathbf{B}_k - \mathbf{A}^{-1})\mathbf{A}(\mathbf{B}_k - \mathbf{A}^{-1}) \\ \Rightarrow \|\mathbf{B}_{k+1} - \mathbf{A}^{-1}\| &\leq \|\mathbf{A}\| \|\mathbf{B}_k - \mathbf{A}^{-1}\|^2 \longrightarrow \text{Convergencia Cuadrática}\end{aligned}$$


---

6.— Para resolver el sistema de ecuaciones:

$$\begin{pmatrix} 4 & 1 \\ 1 & 4 \end{pmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} 6 \\ 9 \end{Bmatrix}$$

Se desea utilizar el método de Gauss-Seidel sobrerrelajado. Se pide:

- Estudiar la convergencia del algoritmo en función del coeficiente de sobrerrelajación  $\alpha$  usado ( $\alpha =$  constante en todas las iteraciones).
  - ¿Existe algún valor óptimo del coeficiente de sobrerrelajación ( constante en todas las iteraciones) para que la convergencia sea más rápida? En caso afirmativo, hallarlo. (Sugerencia: representar gráficamente el radio espectral de la matriz correspondiente en función del coeficiente de sobrerrelajación).
  - Realizar las cinco primeras iteraciones para  $\alpha = 1$  y para  $\alpha = \frac{32}{31}$ , partiendo de la aproximación inicial  $x_1^0 = 0$ ,  $x_2^0 = 0$ .
  - En la práctica, ¿podría realizarse un análisis como el anterior para un sistema de varios miles de ecuaciones? ¿Por qué?
- 

### Solución 6.a

El esquema del algoritmo de Gauss-Seidel sobrerrelajado será:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha \mathbf{C}^{-1}(\mathbf{b} - \mathbf{A} \mathbf{x}^k) \quad ; \quad \text{con} \quad \mathbf{C} = \begin{bmatrix} 4 & 0 \\ 1 & 4 \end{bmatrix}$$

Definimos el error en la iteración  $k$  como :

$$\mathbf{e}^k = \mathbf{x} - \mathbf{x}^k \quad \longrightarrow \quad \mathbf{e}^{k+1} = [\mathbf{I} - \alpha \mathbf{C}^{-1} \mathbf{A}] \mathbf{e}^k$$

por lo que la condición de convergencia es  $\rho(\mathbf{I} - \alpha \mathbf{C}^{-1} \mathbf{A}) < 1$ , y cuanto menor sea, mayor será la velocidad de convergencia.

Para estudiar la convergencia, por tanto, estudiamos los autovalores de  $(\mathbf{I} - \alpha \mathbf{C}^{-1} \mathbf{A})$  en función de  $\alpha$

$$(\mathbf{I} - \alpha \mathbf{C}^{-1} \mathbf{A})\mathbf{u} = \lambda \mathbf{u} \quad \Leftrightarrow \quad \det(\mathbf{I} - \alpha \mathbf{C}^{-1} \mathbf{A} - \lambda \mathbf{I}) = 0 \Leftrightarrow$$

$$\Leftrightarrow \det((1-\lambda)\mathbf{C} - \alpha\mathbf{A}) = 0 \Leftrightarrow \det(\alpha\mathbf{A} + (\lambda-1)\mathbf{C}) = 0 \Leftrightarrow$$

$$\Leftrightarrow \begin{vmatrix} 4\alpha + (\lambda-1)4 & 1\alpha + (\lambda-1)0 \\ 1\alpha + (\lambda-1)1 & 4\alpha + (\lambda-1)4 \end{vmatrix} = \begin{vmatrix} 4(\alpha + \lambda - 1) & \alpha \\ (\alpha + \lambda - 1) & 4(\alpha + \lambda - 1) \end{vmatrix} = 0 \Leftrightarrow$$

$$\Leftrightarrow 16(\alpha + \lambda - 1)^2 - \alpha(\alpha + \lambda - 1) = 0 \Leftrightarrow \begin{cases} (\alpha + \lambda - 1) = 0 \rightarrow \lambda = 1 - \alpha \\ \text{ó} \\ 16(\alpha + 1 - 1) - \alpha = 0 \rightarrow \lambda = 1 - \frac{15\alpha}{16} \end{cases}$$

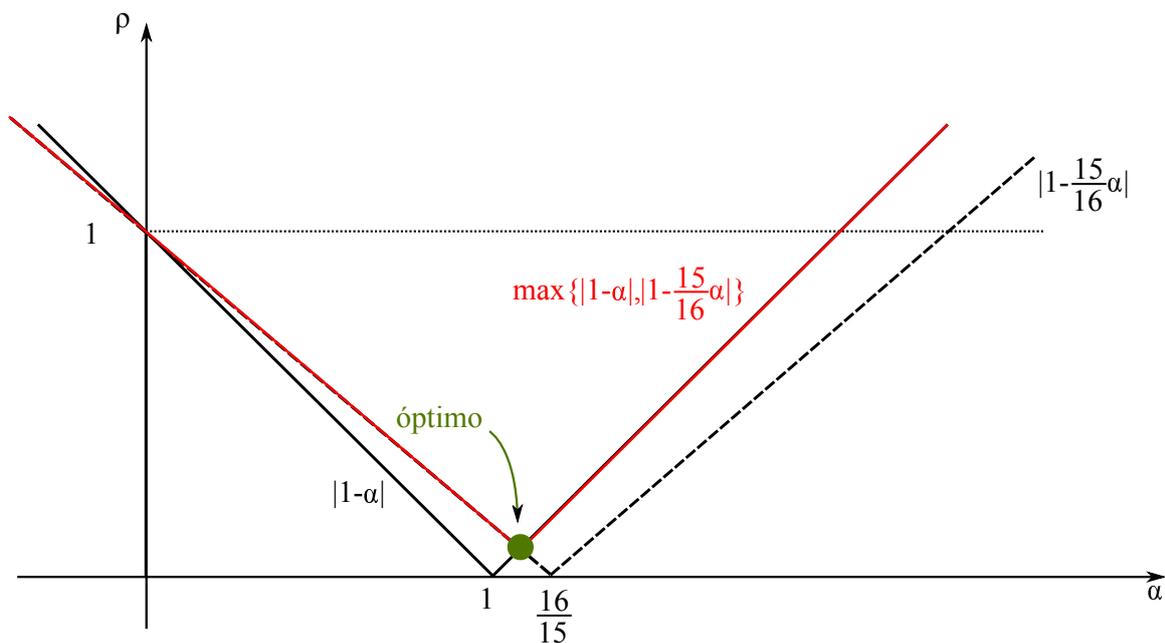
$$\rho(\mathbf{I} - \alpha\mathbf{C}^{-1}\mathbf{A}) = \max \left\{ |1 - \alpha|, \left| 1 - \frac{15\alpha}{16} \right| \right\}$$

Por lo que, para que el algoritmo converja.

$$\left\{ \begin{array}{l} |1 - \alpha| < 1 \Leftrightarrow -1 < 1 - \alpha < 1 \Leftrightarrow -2 < -\alpha < 0 \\ y \\ \left| 1 - \frac{15\alpha}{16} \right| < 1 \Leftrightarrow -1 < 1 - \frac{15\alpha}{16} < 1 \Leftrightarrow -2 < 1 - \frac{15\alpha}{16} < 0 \end{array} \right\} \Leftrightarrow \left\{ \begin{array}{l} \alpha > 0 \\ \alpha < 2 \\ \alpha < \frac{32}{15} \end{array} \right\}$$

Luego el algoritmo es convergente si  $0 < \alpha < 2$ .

**Solución 6.b**



El valor óptimo de  $\alpha$  es aquel que minimiza el el radio espectral  $\rho(\mathbf{I} - \alpha\mathbf{C}^{-1}\mathbf{A})$ , es decir:

$$\alpha - 1 = 1 - \frac{15\alpha}{16} \Leftrightarrow \alpha \left( 1 + \frac{15}{16} \right) = 2 \Leftrightarrow \alpha = \frac{32}{31} = \alpha_{opt}$$

**Solución 6.c**

La forma más práctica de realizar los cálculos es la siguiente:

$$\left\{ \begin{array}{l} \mathbf{x}^{k+1} = \mathbf{x}^k + \alpha \mathbf{s}^k \\ \mathbf{C} \mathbf{s}^k = \mathbf{b} - \mathbf{A} \mathbf{x}^k \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} \begin{Bmatrix} x_1^{k+1} \\ x_2^{k+1} \end{Bmatrix} = \begin{Bmatrix} x_1^k \\ x_2^k \end{Bmatrix} + \alpha \begin{Bmatrix} s_1^k \\ s_2^k \end{Bmatrix} \\ \begin{bmatrix} 4 & 0 \\ 1 & 4 \end{bmatrix} \begin{Bmatrix} s_1^k \\ s_2^k \end{Bmatrix} = \begin{Bmatrix} 6 \\ 9 \end{Bmatrix} - \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix} \begin{Bmatrix} x_1^k \\ x_2^k \end{Bmatrix} \end{array} \right.$$

$$\alpha = 1 \Rightarrow \begin{cases} x_1^{k+1} = -x_2^k/4 + 3/2 \\ x_2^{k+1} = x_2^k/16 + 15/8 \end{cases}$$

$$\alpha = 32/31 \Rightarrow \begin{cases} x_1^{k+1} = 48/31 - 1/31 x_1^k - 8/31 x_2^k \\ x_2^{k+1} = 60/31 + 2/31 x_2^k \end{cases}$$

Realizando las primeras 5 iteraciones partiendo de la solución inicial proporcionada:

k	$x_1^k (\alpha = 1)$	$x_2^k (\alpha = 1)$	$x_1^k (\alpha = 32/31)$	$x_2^k (\alpha = 32/31)$
0	0.000000000	0.000000000	0.000000000	0.000000000
1	1.500000000	1.875000000	1.548387097	1.935483871
2	1.031250000	1.992187500	0.998959417	1.997918835
3	1.001953125	1.999511719	1.000570642	1.999932866
4	1.000122070	1.999969482	0.999998917	1.999997834
5	1.000007629	1.999998093	1.000000594	1.999999930

Observamos que los resultados son muy similares, y en ambos casos el método converge a la solución  $\mathbf{x} = \{1, 2\}$ .

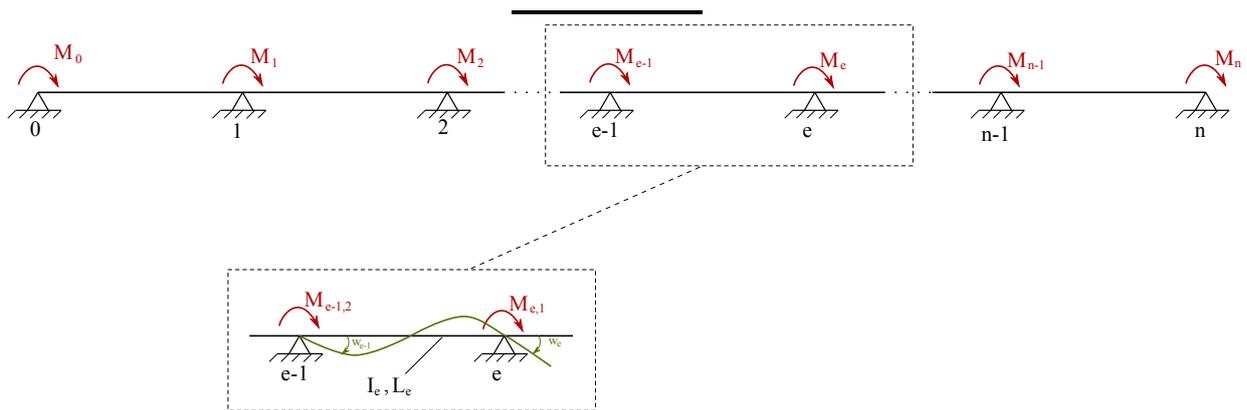
Se puede observar también que para el  $\alpha$  óptimo la convergencia es casi cuadrática aunque en el caso de  $\alpha = 1$  la velocidad es también rápida.

### Solución 6.d

Para grandes sistemas no es posible realizar un estudio analítico como el anterior porque no es posible obtener analíticamente el radio espectral de la matriz. Pero es posible realizar algunas iteraciones para distintos valores de  $\alpha$  para un sistema con la misma matriz y solución conocida (por ejemplo  $\mathbf{Ax} = \mathbf{0}$ ) con el fin de obtener experimentalmente un valor de  $\alpha$  que produzca una velocidad de convergencia razonablemente alta.

7.— Sea una viga continua formada por  $n$  vanos. Se numeran los vanos y los apoyos correlativamente, de forma que el vano  $e \in \{1, \dots, n\}$  se extiende desde el apoyo  $e - 1$  al apoyo  $e$ . Sea  $E$  el módulo de elasticidad del material, y sean  $I_e$  y  $L_e$  el momento de inercia de la sección y la longitud correspondientes al vano  $e$ -ésimo, respectivamente. Dados los momentos  $\{M_i\}_{i=0, \dots, n}$  que actúan sobre los apoyos se desea calcular los correspondientes giros  $\{\omega_i\}_{i=0, \dots, n}$  que se producen. Se pide:

- Plantear el sistema de ecuaciones lineales que es preciso resolver para calcular los giros.
- Verificar que la matriz de coeficientes del sistema anterior es simétrica y definida positiva.
- Proponer y desarrollar completamente el método directo que se considere más adecuado para resolver el sistema.
- Implementar el método seleccionado en un programa FORTRAN que permita resolver este tipo de problemas.
- Plantear la solución del sistema mediante el método iterativo de Gauss-Seidel. ¿Se puede asegurar que este método convergerá? Interpretar el funcionamiento del método desde el punto de vista estructural.



**Solución 7.a**

$$\begin{cases} \omega_{e-1} = M_{e-1,2} \frac{L_e}{3EI_e} - M_{e,1} \frac{L_e}{6EI_e} \\ \omega_e = M_{e,1} \frac{L_e}{3EI_e} - M_{e-1,2} \frac{L_e}{6EI_e} \end{cases} \Leftrightarrow \Leftrightarrow \begin{cases} \omega_{e-1} \\ \omega_e \end{cases} = \frac{L_e}{6EI_e} \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{cases} M_{e-1,2} \\ M_{e,1} \end{cases}$$

Luego:

$$\Leftrightarrow \begin{cases} M_{e-1,2} \\ M_{e,1} \end{cases} = \frac{EI_e}{L_e} \begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix} \begin{cases} \omega_{e-1} \\ \omega_e \end{cases}$$

Por equilibrio de momentos:

$$\begin{cases} M_0 = M_{1,2} \\ M_i = M_{i,1} + M_{i,2} \quad ; \quad i = 1, \dots, n-1 \\ M_n = M_{n,1} \end{cases}$$

Por tanto:





```

    IMPLICIT REAL * 4 (A - H, O - Z)
    PARAMETER(MXN = 100, MXKP = 4 * MXN + 2)
    INTEGER * 4 N
    DIMENSIONWORK(MXKP)

1  WRITE(6, 2(A, $))'NUMERO DE VANOS :/'
    READ(5, *)N
    IF(N.GT.MXN)CALL ERRORFATAL('NUMERO EXCESIVO DE VANOS(N > MXN)')

    KPLIBRE = 1
    CALL DIMENSIONAMIENTODINAMICO(MXKP, KPLIBRE, N, KPK)
    CALL DIMENSIONAMIENTODINAMICO(MXKP, KPLIBRE, N, KPL)
    CALL DIMENSIONAMIENTODINAMICO(MXKP, KPLIBRE, N + 1, KPD)
    CALL DIMENSIONAMIENTODINAMICO(MXKP, KPLIBRE, N + 1, KPW)

    CALL LEERDATOS(N, WORK(KPK), WORK(KPW))
    CALL RESOLVERSISTEMA(N, WORK(KPK), WORK(KPL), WORK(KPD), WORK(KPW))
    CALL ESCRIBIRRESULTADOS(N, WORK(KPW))

    END
-----
    SUBROUTINE LEERDATOS(N, RK, W)
    IMPLICITREAL * 8(A - H, O - Z)
    DIMENSION RK(N), W(0 : N)

    DO I = 1, N
        WRITE(6, 100)I
100  FORMAT('VANO', I5, '----- > RIGIDEZ EI/L =', $)
        READ(5, *)RK(I)
    ENDDO

    DO I = 1, N
        WRITE(6, 110)I
110  FORMAT('NODO', I5, '----- > MOMENTO =', $)
        READ(5, *)W(I)
    ENDDO

    RETURN
    END
-----

```

```

SUBROUTINE RESOLVERSISTEMA(N, RK, RL, RD, W)
  IMPLICIT REAL * 8(A - H, O - Z)
  DIMENSION RK(N), RL(N), RD(0 : N), W(0 : N)

  I = 0
  RD(I) = 4,0D + 00 * (RK(I + 1))
  DO I = 1, N - 1
    RL(I) = 2,0D + 00 * RK(I) / RD(I - 1)
    RD(I) = 4,0D + 00 * (RK(I) + RK(I + 1)) - RL(I) * RD(I - 1) * RL(I)
  ENDDO
  I = N
  RL(I) = 2,0D + 00 * RK(I) / RD(I - 1)
  RD(I) = 4,0D + 00 * (RK(I) - RL(I) * RD(I - 1) * RL(I))

  DO I = 1, N
    W(I) = W(I) - RL(I) * W(I - 1)
  ENDDO
  DO I = 0, N
    W(I) = W(I) / RD(I)
  ENDDO
  DO I = N - 1, 0, -1
    W(I) = W(I) - RL(I + 1) * W(I + 1)
  ENDDO

  RETURN
END
-----
SUBROUTINE ESCRIBIRRESULTADOS(N, W)
  IMPLICIT REAL * 8(A - H, O - Z)
  DIMENSION W(0 : N)

  WRITE(6, 200)
200  FORMAT('GIROSENLOS NODOS', /
.      '=====', /
.      'NODO GIRO(RADIANES)', /
.      '=====', /

  DO I = 0, N
    WRITE(6, 210) I, W(I)
210  FORMAT(1X, I10, D15, 6)
  ENDDO

  RETURN
END
-----

```

```

SUBROUTINE DIMENSIONAMIENTODINAMICO(MXKP, KPLIBRE, NCOMPONENTES, KP

KP = KPLIBRE
KPLIBRE = KPLIBRE + NCOMPONENTES
IF(KPLIBRE.GT.MXKP + 1)CALL ERRORFATAL('MEMORIA INSUFICIENTE')

RETURN
END
-----
SUBROUTINE ERRORFATAL(MENSAJE)
CHARACTER * (*)MENSAJE

WRITE(6, 100)MENSAJE
100 FORMAT('ERROR :', A)

STOP
END

```

### Solución 7.e

La solución del sistema por el método iterativo de Gauss-Seidel sería la siguiente:

$$\begin{aligned} \omega_0^{k+1} &= (u_0 - 2 k_1 \omega_1^k) / (4 k_1) \\ \omega_1^{k+1} &= (u_1 - 2 k_2 \omega_0^{k+1} - 2 k_2 \omega_2^k) / (4(k_1 + k_2)) \\ &\vdots \\ \omega_i^{k+1} &= (u_i - 2 k_i \omega_{i-1}^{k+1} - 2 k_{i+1} \omega_{i+1}^k) / (4(k_i + k_{i+1})) \quad i = 1, \dots, n-1 \\ &\vdots \\ \omega_{n-1}^{k+1} &= (u_{n-1} - 2 k_{n-1} \omega_{n-2}^{k+1} - 2 k_n \omega_n^k) / (4(k_{n-1} + k_n)) \\ \omega_n^k &= (u_n - 2 k_n \omega_{n-1}^{k+1}) / (4 k_n) \end{aligned}$$

El método funcionará con total seguridad, pues la matriz  $\mathbf{K}$  es diagonalmente dominante:

$$\begin{cases} |4 k_1| > |2 k_1| \\ |4(k_e + k_{e+1})| > |2 k_e| + |2 k_{e+1}| \\ |4 k_n| > |2 k_n| \end{cases}$$

Desde un punto de vista estructural, el método puede interpretarse de la siguiente forma:

- Se suponen unos giros inicialmente.
- Se repite hasta convergencia:
  - Se empotran todos los apoyos con sus giros actuales.
  - Se libera el nodo 0 y se recalcula su giro.

- Se vuelven a empotrar todos los apoyos.
- Se libera el nodo 1 y se recalcula su giro.
- Se procede igualmente para todos los nodos restantes.

Esta es la base de los antiguos métodos de cálculo estructural previos al análisis matricial, como los métodos de Cross, Kani, etc.

8.— Sea  $\mathbf{A}$  una matriz simétrica y regular de orden  $n$ . Se desea encontrar la solución del sistema lineal de ecuaciones:

$$\mathbf{A} \mathbf{x} = \mathbf{b}$$

dado  $\mathbf{b}$ , se pide:

- Escribir el algoritmo de Gauss sin pivotamiento **evitando las operaciones inútiles**. Calcular el número de operaciones necesarias para resolver el problema.
- ¿Puede emplearse el algoritmo de Gauss con pivotamiento conservando la simetría? ¿Por qué?

**Sugerencia:** Recuérdese que en cada paso del proceso de eliminación, cuando se anulan los términos de la columna  $k$ -ésima por debajo del pivote,  $a_{kk}$ , sólo se recalcula la submatriz:

$$\begin{pmatrix} a_{k+1,k+1} & \cdots & a_{k+1,n} \\ \vdots & \ddots & \vdots \\ a_{n,k+1} & \cdots & a_{n,n} \end{pmatrix}$$

### Solución 8.a

Partiendo del sistema de ecuaciones:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix}$$

tras finalizar el primero paso de normalización de la primera ecuación y eliminación en las restantes, el sistema tiene la forma siguiente:

$$\begin{bmatrix} a_{11} & a_{12}/a_{11} & a_{13}/a_{11} & \cdots & a_{1n}/a_{11} \\ 0 & a_{22} - a_{21} \frac{a_{12}}{a_{11}} & a_{23} - a_{21} \frac{a_{13}}{a_{11}} & \cdots & a_{2n} - a_{21} \frac{a_{1n}}{a_{11}} \\ 0 & a_{32} - a_{31} \frac{a_{12}}{a_{11}} & a_{33} - a_{31} \frac{a_{13}}{a_{11}} & \cdots & a_{3n} - a_{31} \frac{a_{1n}}{a_{11}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2} - a_{n1} \frac{a_{12}}{a_{11}} & a_{n3} - a_{n1} \frac{a_{13}}{a_{11}} & \cdots & a_{nn} - a_{n1} \frac{a_{1n}}{a_{11}} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1/a_{11} \\ b_2 - a_{21} \frac{b_1}{a_{11}} \\ b_3 - a_{31} \frac{b_1}{a_{11}} \\ \vdots \\ b_n - a_{n1} \frac{b_1}{a_{11}} \end{pmatrix}$$

Se puede observar fácilmente que si la matriz era inicialmente simétrica, la submatriz con la que se trabajara en la fase siguiente también lo es. Las submatrices con las que se opera en cada paso conservan la simetría a lo largo de todo el proceso.

Por tanto podemos modificar el método de Gauss para evitar almacenar elementos fuera de la diagonal dos veces. Si lo planteamos almacenando la parte triangular superior de la matriz:

$$\begin{array}{l}
 \text{Triangularización / Eliminación} \\
 \text{Elim. Térm. Indep.} \\
 \text{Resolución}
 \end{array}
 \left\{ \begin{array}{l}
 \text{do } i = 1, n - 1 \\
 \quad \text{do } k = i + 1, n \\
 \quad \quad c = a_{ik}/a_{ii} \\
 \quad \quad a_{kj} = a_{kj} - c a_{ij} \quad ; \quad j = k, \dots, n \\
 \quad \text{enddo} \\
 \text{enddo} \\
 \\
 \text{do } i = 1, n - 1 \\
 \quad \text{do } k = i + 1, n \\
 \quad \quad c = a_{ik}/a_{ii} \\
 \quad \quad v_k = b_k - c b_i \quad ; \quad j = k, \dots, n \\
 \quad \text{enddo} \\
 \text{enddo} \\
 \\
 \left\{ \begin{array}{l}
 x_n = b_n/a_{nn} \\
 x_i = (b_i - \sum_{j=i+1}^n a_{ij} x_j) / a_{ii}; \quad i = n - 1, \dots, 1 \quad (\text{Hacia atrás})
 \end{array} \right.
 \end{array}$$

### Solución 8.b

No podemos plantear un algoritmo de Gauss con pivotamiento para matrices simétricas porque el pivotamiento destruye la simetría de la matriz.

El método de Gauss es conceptualmente más complicado que el método de Cholesky y no tiene ninguna ventaja por lo que normalmente no se utiliza. En general se prefiere utilizar el método de Cholesky para sistemas con matrices simétricas definidas positivas.