INTRODUCCIÓN A LOS MÉTODOS NUMÉRICOS 2024/2025

CÁLCULO MATRICIAL Y SEL (PRÁCTICA 4)

1. Cierto problema de ingeniería conduce a la utilización de la matriz simétrica:

$$\boldsymbol{A} = \begin{bmatrix} 2 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & \ddots & & & \\ & & \ddots & \ddots & -1 & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & \end{bmatrix} \tag{1}$$

Se pide:

a) Proponer un esquema de almacenamiento eficiente para esta matriz.

b) Desarrollar un algoritmo específico y eficiente que permita realizar el producto de matriz por vector para matrices con estructura similar a la propuesta.

c) Aplicar en el algoritmo desarrollado en el apartado anterior el esquema de almacenamiento propuesto en el apartado a).

d) Proponer un algoritmo que permita descomponer la matriz $\boldsymbol{A}$ como $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$ de forma eficiente.

e) Aplicar el esquema de almacenamiento propuesto en el apartado a) en el algoritmo de factorización anterior.

f) Proponer, a partir de la descomposición anterior, algoritmos eficientes para resolver un sistema del tipo $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.

g) Aplicar en estos algoritmos el esquema de almacenamiento propuesto.

h) Desarrollar un programa de ordenador que permita resolver el sistema de ecuaciones propuesto para un vector de términos independientes $\boldsymbol{b}$ tal que $b_i = 1,\ i = 1, \ldots, n$ con $n \leq 30$

Solución 1.a Para establecer un esquema de almacenamiento adecuado para una matriz es importante tener en cuenta las operaciones que vamos a realizar con ella. Así, el esquema de almacenamiento que propongamos para esta matriz variará si por ejemplo queremos hacer un producto de la matriz por un vector o si queremos calcular la inversa de la misma. Para el producto de matriz por vector podemos plantear un esquema de almacenamiento que sólo almacene los términos no nulos, porque son los que intervienen en la operación. Los términos nulos no aportan nada al producto. Sin embargo, si queremos calcular la matriz inversa se puede comprobar que la inversa de una matriz tridiagonal es una matriz llena. Y por tanto, no sería razonable utilizar el mismo esquema de almacenamiento que para el producto.

En el caso que nos ocupa vamos a calcular el producto de una matriz por un vector y la factorización de Cholesky. La factorización conserva el ancho de banda, por lo que un esquema de almacenamiento en banda será la forma más eficiente de almacenar esta matriz.

$$\boldsymbol{A} = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} \quad \rightsquigarrow \quad \boldsymbol{T} = \begin{bmatrix} & 2 \\ -1 & 2 \\ -1 & 2 \\ \vdots & \vdots \\ -1 & 2 \end{bmatrix} \tag{2}$$

De modo que:

$$a_{ij} \quad \rightsquigarrow \quad t_{\alpha\beta} \qquad \begin{array}{ll} \nearrow & \text{Si } i \geq j \quad \begin{cases} \alpha = i \\ \beta = 2 + j - i \end{cases} \\ \searrow & \text{Si } i \geq j \quad \begin{cases} \alpha = j \\ \beta = 2 + i - j \end{cases} \end{array} \tag{3}$$

Solución 1.b El algoritmo de producto de matriz por vector para una matriz simétrica en banda lo vamos a obtener de modo parecido a como se planteó en el ejemplo práctico en teoría. Partimos de la ecuación algebraica del producto de matriz por vector y escribimos el algoritmo en pseudocódigo.

$$\boldsymbol{p} = \boldsymbol{A}\boldsymbol{x} \tag{4}$$

Escribimos un algoritmo general de producto de matriz por vector:

$$\begin{Bmatrix} p_1 \\ \vdots \\ p_i \\ \vdots \\ p_n \end{Bmatrix} = \begin{bmatrix} a_{11} & \cdots & \cdots & \cdots & a_{1n} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{i1} & \cdots & a_{ii} & \cdots & a_{in} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & \cdots & a_{n,n} \end{bmatrix} \begin{Bmatrix} v_1 \\ \vdots \\ v_i \\ \vdots \\ v_n \end{Bmatrix} \tag{5}$$

$$\begin{array}{l} do \;\; i = 1, n \\ \quad p(i) = \sum_{j=1}^{n} a(i,j) * v(j) \\ enddo \end{array} \tag{6}$$

El algoritmo adaptado para una estructura de matriz tridiagonal quedaría como:

$$\begin{Bmatrix} p_1 \\ p_2 \\ \vdots \\ \vdots \\ p_i \\ \vdots \\ \vdots \\ p_n \end{Bmatrix} = \begin{bmatrix} a_{11} & a_{12} & & & & & & \\ a_{21} & a_{22} & a_{23} & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & a_{i,i-1} & a_{ii} & a_{i,i+1} & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & \ddots & \ddots & a_{n-1,n} \\ & & & & & & a_{n,n-1} & a_{n,n} \end{bmatrix} \begin{Bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{i-1} \\ v_i \\ v_{i+1} \\ \vdots \\ v_{n-1} \\ v_n \end{Bmatrix} \tag{7}$$

$$\begin{array}{l} do \;\; i = 1, n \\ \quad p(i) = \sum_{j=\text{máx}(1,i-1)}^{\text{mín}(n,i+1)} a(i,j) * v(j) \\ enddo \end{array} \tag{8}$$

Si ahora tenemos en cuenta que la matriz es simétrica y se adopta como criterio que se almacena sólo la parte inferior de la misma tenemos que separar los términos que están en la parte superior de la matriz de los que están en la parte inferior.

$$\begin{Bmatrix} p_1 \\ p_2 \\ \vdots \\ \vdots \\ p_i \\ \vdots \\ \vdots \\ p_n \end{Bmatrix} = \begin{bmatrix} a_{11} & a_{12} & & & & & & \\ a_{21} & a_{22} & a_{23} & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & a_{i,i-1} & a_{ii} & a_{i,i+1} & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & \ddots & \ddots & a_{n-1,n} \\ & & & & & & a_{n,n-1} & a_{n,n} \end{bmatrix} \begin{Bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{i-1} \\ v_i \\ v_{i+1} \\ \vdots \\ v_{n-1} \\ v_n \end{Bmatrix} \tag{9}$$

$$\begin{aligned} &do \;\; i = 1, n \\ &\quad p(i) = \sum_{j=\text{máx}(1,i-1)}^{i} a(i,j) * v(j) \;\; + \sum_{j=i+1}^{\text{mín}(n,i+1)} a(i,j) * v(j) \\ &enddo \end{aligned} \tag{10}$$

Solución 1.c Solución Una vez separados los términos de la parte superior (en color azul) y los de la parte inferior (en color verde) podemos aplicar el esquema de almacenamiento propuesto en (3), de modo que el algoritmo quedaría ahora como:

$$\begin{aligned} &do \;\; i = 1, n \\ &\quad p(i) = \sum_{j=\text{máx}(1,i-1)}^{i} t(i, 2+j-i) * v(j) \;\; + \sum_{j=i+1}^{\text{mín}(n,i+1)} t(j, 2+i-j) * v(j) \\ &enddo \end{aligned} \tag{11}$$

Solución 1.d El algoritmo de descomposición en este caso sería el algoritmo propuesto en teoría para matriz tridiagonal particularizado para matriz simétrica. Lo escribiríamos en lenguaje Fortran como:

```
d(1,1)=a(1,1)
do i=1,n-1
  l(i+1,i)=a(i+1,i)/d(i,i)
  d(i+1,i+1)=a(i+1,i+1)-d(i,i)*l(i+1,i)**2
  l(i+1,i+1)=1
enddo
```

Solución 1.e A continuación aplicamos el esquema de almacenamiento propuesto al algoritmo de descomposición anterior. Para ello tendremos en cuenta que la diagonal de la matriz $\boldsymbol{D}$ se almacenará en un vector $\boldsymbol{d}$ y que la matriz $\boldsymbol{L}$ se almacena también como matriz en banda $\boldsymbol{L}_b$.

$$\begin{aligned} a_{11} &\rightsquigarrow t_{12} \\ a_{i+1,i} &\rightsquigarrow t_{i+1,1} \\ a_{i+1,i+1} &\rightsquigarrow t_{i+1,2} \\ l_{i+1,i} &\rightsquigarrow lb_{i+1,1} \\ l_{i+1,i+1} &\rightsquigarrow lb_{i+1,2} \end{aligned} \tag{12}$$

Si se hace así el algoritmo quedaría como:

```
d(1)=t(1,2)
do i=1,n-1
  l(i+1,1)=t(i+1,1)/d(i)
  d(i+1)=t(i+1,2)-d(i)*lb(i+1,1)**2
  l(i+1,2)=1
enddo
```

Como esta forma de resolver el problema involucra varios esquemas de almacenamiento para las diferentes matrices y además es necesario redefinir nuevas matrices y vectores ($\boldsymbol{d}$ y $\boldsymbol{L}b$) lo más fácil y razonable sería plantear el algoritmo almacenando la factorización sobre la propia matriz de datos. El algoritmo quedaría así como:

```
do i=1,n-1
  t(i+1,1)=t(i+1,1)/t(i,2)
  t(i+1,2)=t(i+1,2)-t(i,2)*t(i+1,1)**2
enddo
```

Solución 1.e Dada la descomposición de cholesky realizada en los apartados anteriores la resolución de los sistemas se puede plantear como:

$$\boldsymbol{LDL}^T\boldsymbol{x} \;=\; \boldsymbol{b} \quad \longrightarrow \quad \begin{cases} \boldsymbol{L}\,\boldsymbol{z} = \boldsymbol{b} \\ \boldsymbol{D}\,\boldsymbol{y} = \boldsymbol{z} \\ \boldsymbol{L}^T\boldsymbol{x} = \boldsymbol{y} \end{cases} \tag{13}$$

Entonces los algoritmos de resolución quedarían como:

```
z(1)=b(1)
do i=2,n
  z(i)=b(i)-l(i,i-1)*z(i-1)
enddo

do i=1,n
  y(i)=z(i)/d(i,i)
enddo

x(n)=y(n)
do i=n-1,1,-1
  x(i)=y(i)-l(i+1,i)*x(i+1)
enddo
```

Solución 1.g Si se aplican los esquemas de almacenamiento propuestos anteriormente para $\boldsymbol{D}$ y para $\boldsymbol{L}$ el algoritmo quedaría como:

```
z(1)=b(1)
do i=2,n
  z(i)=b(i)-lb(i,1)*z(i-1)
enddo

do i=1,n
  y(i)=z(i)/d(i)
enddo
```

```
x(n)=y(n)
do i=n-1,1,-1
  x(i)=y(i)-lb(i+1,1)*x(i+1)
enddo
```

En todo caso, desde un punto de vista de implementación en un programa de ordenador lo más sencillo sería almacenar todas las matrices sobre la propia matriz de datos (de acuerdo con la factorización propuesta al final del apartado 1.e) y resolver los sistemas de ecuaciones almacenando los resultados sobre el propio vector de términos independientes. En ese caso estos algoritmos quedarían como:

```
do i=2,n
  b(i)=b(i)-t(i,1)*b(i-1)
enddo

do i=1,n
  b(i)=b(i)/t(i,2)
enddo

do i=n-1,1,-1
  b(i)=b(i)-t(i+1,1)*b(i+1)
enddo
```

La factorización propuesta al final del apartado 1.e) y el algoritmo de resolución de sistemas inmediatos que acabamos de desarrollar ahora serían la forma más sencilla de implementar en un programa de ordenador la resolución de este tipo de sistemas de ecuaciones lineales.

---

2. Dado el sistema de ecuaciones con matriz:

$$\boldsymbol{A} = \begin{bmatrix} a_{1,1} & & & \\ a_{2,1} & a_{2,2} & & \\ \vdots & & \ddots & \\ a_{n,1} & & & a_{n,n} \end{bmatrix} \tag{14}$$

Se pide:

a) Si es posible, desarrollar un algoritmo que permita resolver eficientemente el sistema de ecuaciones $\boldsymbol{Ax} = \boldsymbol{b}$ de forma directa sin necesidad de transformar previamente el problema.

b) Deducir la complejidad computacional (mediante el número de operaciones requerido) del algoritmo resultante.

Solución 2.a El sistema propuesto se trata de un sistema triangular inferior. Por tanto se puede resolver de forma inmediata con un algoritmo de sustitución hacia adelante.

$$\begin{bmatrix} a_{11} & & & & \\ a_{21} & a_{22} & & & \\ a_{31} & & a_{33} & & \\ \vdots & & & \ddots & \\ a_{n1} & & & & a_{nn} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{Bmatrix} \tag{15}$$

$$\begin{array}{lcl} a_{11}\,x_1 = b_1 & \longrightarrow & x_1 = b_1/a_{11} \\ a_{21}\,x_1 + a_{22}\,x_2 = b_2 & \longrightarrow & x_2 = (b_2 - a_{21}\,x_1)/a_{22} \\ a_{31}\,x_1 + a_{33}\,x_3 = b_3 & \longrightarrow & x_3 = (b_3 - a_{31}\,x_1)/a_{33} \\ \vdots \qquad\qquad\quad \vdots & & \vdots \qquad\quad \vdots \\ a_{n1}\,x_1 + a_{nn}\,x_n = b_n & \longrightarrow & x_n = (b_n - a_{n1}\,x_1)/a_{nn} \end{array} \tag{16}$$

El algoritmo quedaría como:

```
x(1)=b(1)/a(1,1)
do i=2,n
  x(i)=(b(i)-a(i,1)*x(1))/a(i,i)
enddo
```

Solución 2.b Las operaciones a realizar son:

$$\left.\begin{array}{l} 1 \text{ cociente} \\ \left.\begin{array}{l} 1 \text{ cociente} \\ 1 \text{ resta} \\ 1 \text{ producto} \end{array}\right\} (n-1) \text{ veces} \end{array}\right\} \longrightarrow \quad T(n) \tag{17}$$

3. Se desea resolver un sistema de ecuaciones con matriz $\boldsymbol{A}$ mediante un método de eliminación de Gauss. La matriz $\boldsymbol{A}$ se define como:

$$\boldsymbol{A} = \begin{bmatrix} 4 & 1 & & \\ 1 & 4 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & 4 \end{bmatrix} \tag{18}$$

Se pide:

a) Desarrollar completamente el algoritmo de eliminación de Gauss sin pivotamiento. ¿Es posible realizar el algoritmo anterior de forma eficiente sin incrementar el ancho de banda de la matriz?

b) Proponer y desarrollar un algoritmo eficiente que permita resolver el sistema de ecuaciones resultante tras aplicar el Método propuesto en el apartado a).

c) Desarrollar completamente el algoritmo de eliminación de Gauss-Jordan. ¿Es posible realizar el algoritmo anterior de forma eficiente sin incrementar el ancho de banda de la matriz?

Solución 3.a El algoritmo de eliminación de Gauss trata de conseguir ceros en la parte inferior de la matriz a base de realizar transformaciones por filas. Si se analiza el proceso para esta matriz tenemos que conseguir que los unos que forman la subdiagonal de la matriz se conviertan en ceros. Por ejemplo, para que el elemento a_{21} se anule hay que sumarle a la fila 2 la fila 1 multiplicada por $(-1/4)$. Y es inmediato comprobar que esta operación no modifica el ancho de banda de la matriz y logra que $a_{21} = 0$. Por tanto se puede aplicar el algoritmo sin incrementar el ancho de banda. El algoritmo sería:

```
do i=1,n
  pivote=a(i,i)
  do j=i,min(n,i+1)
    a(i,j)=a(i,j)/pivote
  enddo
  b(i)=b(i)/pivote
  do k=i+1,min(n,i+1)   ! El bucle do no se repite. Se ejecuta para k=i+1
    factor=a(k,i)
    do j=i,min(n,i+1)   ! Tambien seria correcto do j=i,i+1
      a(k,j)=a(k,j)-a(i,j)*factor
    enddo
    b(k)=b(k)-b(i)*factor
  enddo
enddo
```

Las consideraciones adicionales que se han tenido en cuenta para este algoritmo son:

- La normalización de la ecuación correspondiente al pivote en cada caso se aplica sobre los términos no nulos.
- La anulación de los elementos que están por debajo del pivote sólo hay que aplicarla sobre la ecuación siguiente a la del pivote ($k = i + 1$), ya que en el resto de ecuaciones ya tenemos ceros.
- En la fila en la que queremos eliminar coeficientes (fila $k = i + 1$) sólo hay que calcular dos coeficientes: el que se anula y el de la diagonal principal de esa fila.

Solución 3.b De acuerdo con lo indicado en el apartado anterior la matriz tras la eliminación de Gauss quedará como:

$$\boldsymbol{G\,x = b_g} \qquad \text{con} \qquad \boldsymbol{G} = \begin{bmatrix} 1 & g_{12} & & & & \\ & 1 & g_{23} & & & \\ & & \ddots & \ddots & & \\ & & & 1 & g_{n-1,n} & \\ & & & & 1 & \end{bmatrix} \tag{19}$$

Por lo tanto el algoritmo quedaría como:

```
x(n)=bg(n)
do i=n-1,1,-1
  x(i)=bg(i)-g(i,i+1)*x(i+1)
enddo
```

Solución 3.c El algoritmo de Gauss-Jordan sería:

```
do i=1,n
  pivote=a(i,i)
  do j=i,min(n,i+1)
    a(i,j)=a(i,j)/pivote
  enddo
  b(i)=b(i)/pivote
  do k=1,min(n,i+1)
    if (k.ne.i)then
      factor=a(k,i)
      do j=i,min(n,i+1)
        a(k,j)=a(k,j)-a(i,j)*factor
      enddo
      b(k)=b(k)-b(i)*factor
    endif
  enddo
enddo
```

Como se puede observar al realizar las operaciones no es posible aplicar el método de eliminación de Gauss-Jordan sin amplificar el ancho de banda, porque al conseguir ceros en la parte superior destruimos el ancho de banda. Por este motivo el bucle en k empieza en $k = 1$.

4. La resolución de un determinado sistema de ecuaciones lineales ($\boldsymbol{Ly} = \boldsymbol{b}$) nos proporciona valores puntuales de la solución de un determinado problema en un conjunto de n puntos ($x_i,\ i = 1, \ldots, n$) equiespaciados en el intervalo [0,1], donde $x_1 = 0$ y $x_n = 1$. La matriz del sistema de ecuaciones corresponde a una matriz triangular inferior con estructura en banda y semiancho de banda inferior $l = 2$:

$$\boldsymbol{L} = \begin{bmatrix} l_{1,1} & & & & & \\ l_{2,1} & l_{2,2} & & & & \\ l_{3,1} & l_{3,2} & l_{3,3} & & & \\ & l_{4,2} & l_{4,3} & l_{4,4} & & \\ & & \ddots & \ddots & \ddots & \\ & & & l_{n,n-2} & l_{n,n-1} & l_{n,n} \end{bmatrix} \tag{20}$$

Se pide:

a) Desarrollar completamente un algoritmo eficiente de resolución del sistema con matriz triangular inferior ($\boldsymbol{L}$) que nos permita obtener $\boldsymbol{y}$ a partir de $\boldsymbol{b}$.

b) Desarrollar completamente un esquema de almacenamiento eficiente para esta matriz y aplicar este esquema en el algoritmo de resolución propuesto en el apartado anterior.

Solución 4.a El problema propuesto corresponde a la solución de un sistema de ecuaciones triangular inferior en banda. Por lo tanto, el problema se resuelve mediante sustitución hacia adelante.

$$\begin{aligned} & l_{11}\, y_1 = b_1 \\ & l_{21}\, y_1 + l_{22}\, y_2 = b_2 \\ & l_{31}\, y_1 + l_{32}\, y_2 + l_{33}\, y_3 = b_3 \\ & l_{42}\, y_2 + l_{43}\, y_3 + l_{44}\, y_4 = b_4 \\ & \qquad\qquad \vdots \\ & l_{n,n-2}\, y_{n-2} + l_{n,n-1}\, y_{n-1} + l_{nn}\, y_n = b_n \end{aligned} \tag{21}$$

De modo que el algoritmo queda como:

```
y(1)=b(1)/(l(1,1)
y(2)=(b(2)-l(2,1)*y(1))/l(2,2)
do i=3,n
  y(i)=(b(i)-l(i,i-2)*y(i-2)-l(i,i-1)*y(i-1))/l(i,i)
enddo
```

Solución 4.b Tenemos una matriz triangular inferior en banda. Por lo tanto dado que lo que queremos resolver un sistema de ecuaciones lo más adecuado será utilizar un esquema de almacenamiento en banda.

$$\boldsymbol{L} \quad \rightsquigarrow \quad \boldsymbol{T} = \begin{bmatrix} & & l_{11} \\ & l_{21} & l_{22} \\ l_{31} & l_{32} & l_{33} \\ l_{42} & l_{43} & l_{44} \\ \vdots & \vdots & \vdots \\ l_{n,n-2} & l_{n,n-1} & l_{nn} \end{bmatrix} \tag{22}$$

$$l(if, ic) \rightsquigarrow t(\alpha, \beta) \qquad \text{con} \qquad \begin{cases} \alpha = if \\ \beta = 2 + 1 + ic - if \end{cases} \tag{23}$$

Si aplicamos el esquema de almacenamiento:

```
y(1)=b(1)/t(1,3)
y(2)=(b(2)-t(2,2)*y(1))/t(2,3)
do i=3,n
  y(i)=(b(i)-t(i,1)*y(i-2)-t(i,2)*y(i-1))/t(i,3)
enddo
```

5. El análisis de la flexión estructural de una viga empotrada en ambos extremos sometida a una carga transversal a su eje viene determinada por un sistema de ecuaciones con matriz:

$$\boldsymbol{A} = \begin{bmatrix} 7 & -4 & 1 & & & \\ -4 & 6 & -4 & \ddots & & \\ 1 & -4 & 6 & \ddots & 1 & \\ & \ddots & \ddots & \ddots & -4 & 1 \\ & & 1 & -4 & 6 & -4 \\ & & & 1 & -4 & 7 \end{bmatrix} \tag{24}$$

Se pide:

a) Plantear y desarrollar un algoritmo eficiente de factorización que permita descomponer la matriz $\boldsymbol{A}$ como $\boldsymbol{LDL}^T$.

b) Proponer, a partir de la factorización anterior, algoritmos eficientes para resolver un sistema del tipo $\boldsymbol{Ax} = \boldsymbol{b}$.

c) Proponer un esquema de almacenamiento eficiente que tenga en cuenta las características particulares de esta matriz.

Solución 5.a El algoritmo de factorización es un caso particular del algoritmo de factorización para matriz simétrica en banda con semiancho de banda $\ell = 2$.

$$\begin{aligned}
&l_{1,1} = 1 \\
&d_{1,1} = a_{1,1} \\
&do\ k = 1, n-1 \\
&\quad l_{k+1,i} = a_{k+1,i} - \sum_{j=\text{máx}(1,k-1)}^{i-1} l_{ij}\, l_{k+1,j} \qquad i = \text{máx}(1,k-1), \ldots, k \\
&\quad l_{k+1,i} = l_{k+1,i}/d_{ii} \qquad i = \text{máx}(1,k-1), \ldots, k \\
&\quad l_{k+1,k+1} = 1 \\
&\quad d_{k+1,k+1} = a_{k+1,k+1} - \sum_{j=\text{máx}(1,k-1)}^{k} l_{k+1,j}\, d_{jj}\, l_{k+1,j} \\
&enddo
\end{aligned} \tag{25}$$

Solución 5.b Los sistemas de ecuaciones con solución inmediata resultantes se resolverían como:

$$\begin{aligned}
&z_i = b_i - \sum_{j=\text{máx}(1,i-2)}^{i-1} l_{ij}\, z_j \qquad i = 1, \ldots, n \\
&y_i = z_i/d_{ii} \qquad i = 1, \ldots, n \\
&x_i = y_i - \sum_{j=i+1}^{\text{mín}(n,i+2)} l_{ji}\, x_j \qquad i = n, \ldots, 1, (-1) \quad \leftarrow \text{(hacia atrás)}
\end{aligned} \tag{26}$$

Solución 5.c

$$\boldsymbol{A} = \begin{bmatrix} 7 & -4 & 1 & & & \\ -4 & 6 & -4 & \ddots & & \\ 1 & -4 & 6 & \ddots & 1 & \\ & \ddots & \ddots & \ddots & -4 & 1 \\ & & 1 & -4 & 6 & -4 \\ & & & 1 & -4 & 7 \end{bmatrix} \quad \rightsquigarrow \quad \boldsymbol{T} = \begin{bmatrix} & & 7 \\ & -4 & 6 \\ 1 & -4 & 6 \\ 1 & -4 & 6 \\ \vdots & \vdots & \vdots \\ 1 & -4 & 6 \\ 1 & -4 & 7 \end{bmatrix} \tag{27}$$

$$a_{if,ic} \quad \rightsquigarrow \quad t_{\alpha\beta} \quad \rightarrow \quad \begin{cases} if \geq ic & \rightarrow \quad \begin{cases} \alpha = if \\ \beta = 3 + ic - if \end{cases} \\ if < ic & \rightarrow \quad \begin{cases} \alpha = ic \\ \beta = 3 + i - j \end{cases} \end{cases} \tag{28}$$

6. Dado el sistema de ecuaciones con matriz:

$$\boldsymbol{A} = \begin{bmatrix} a_{1,1} & & & & & \\ & \ddots & & & & \\ a_{i,1} & \cdots & a_{i,i} & & & \\ & & & \ddots & & \\ & & & & a_{n,n} \end{bmatrix} \tag{29}$$

Se pide:

a) Describir la forma general de la matriz $\boldsymbol{U}$ que se obtiene como resultado de descomponer la matriz $\boldsymbol{A}$ como $\boldsymbol{LDU}$.

b) Si es posible, desarrollar un algoritmo que permita resolver el sistema de ecuaciones $\boldsymbol{Ax} = \boldsymbol{b}$ de forma directa?

Solución 6.a Dado que la matriz $\boldsymbol{A}$ es triangular inferior la matriz $\boldsymbol{U}$ será la matriz identidad ($\boldsymbol{U} = \boldsymbol{I}$). La matriz $\boldsymbol{U}$ conserva el ancho de banda de la matriz original y tiene unos en la diagonal.

Solución 6.b El sistema de ecuaciones propuesto corresponde a un sistema con matriz triangular inferior particular y por tanto se puede resolver mediante sustitución hacia adelante teniendo en cuenta la estructura específica de esta matriz.

$$\begin{bmatrix} a_{1,1} & & & & & & \\ & \ddots & & & & & \\ & & a_{i-1,i-1} & & & & \\ a_{i,1} & \cdots & a_{i,i-1} & a_{i,i} & & & \\ & & & & a_{i+1,i+1} & & \\ & & & & & \ddots & \\ & & & & & & a_{n,n} \end{bmatrix} \begin{Bmatrix} x_1 \\ \vdots \\ x_{i-1} \\ x_i \\ x_{i+1} \\ \vdots \\ x_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ \vdots \\ b_{i-1} \\ b_i \\ b_{i+1} \\ \vdots \\ b_n \end{Bmatrix} \tag{30}$$

De donde se puede decir que:

$$\begin{aligned} & a_{11}\, x_1 = b_1 \\ & \quad \vdots \\ & a_{i-1,i-1}\, x_{i-1} = b_{i-1} \\ & a_{i1}\, x_1 + \ldots + a_{i,i-1}\, x_{i-1} + a_{i,i}\, x_i = b_i \\ & a_{i-1,i-1}\, x_{i-1} = b_{i-1} \\ & \quad \vdots \\ & a_{n,n}\, x_n = b_n \end{aligned} \tag{31}$$

De modo que el algoritmo queda como:

$$\begin{aligned} & x_j = b_j / a_{jj}; \qquad j = 1, \ldots, i-1 \\ & x_i = \Big(b_i - \sum_{j=1}^{i-1} a_{ij} x_j \Big) / a_{ii} \\ & x_j = b_j / a_{jj}; \qquad j = i+1, \ldots, n \end{aligned} \tag{32}$$

Si escribimos el algoritmo en un programa de Fortran las instrucciones serían tal que así:

```
do j=1,i-1
  x(j)=b(j)/a(j,j)
enddo

suma=0.d+00
do j=1,i-1
  suma=suma+a(i,j)*x(j)
enddo
x(i)=(b(i)-suma)/a(i,i)

do j=i+1,n
  x(j)=b(j)/a(j,j)
enddo
```

---

7. Un ingeniero desea resolver sistemas de ecuaciones en los que la matriz presenta la siguiente forma:

$$\boldsymbol{A} = \begin{bmatrix} a_{1,1} & & & & & a_{1,n} \\ a_{2,1} & a_{2,2} & & & & a_{2,n} \\ a_{3,1} & & a_{3,3} & & & a_{3,n} \\ \vdots & & & \ddots & & \vdots \\ \vdots & & & & a_{n-1,n-1} & a_{n-1,n} \\ a_{n,1} & & & & & a_{n,n} \end{bmatrix} \tag{33}$$

Se pide:

a) Si se pretende realizar el producto de esta matriz por un vector, indicar justificadamente si es posible utilizar un esquema de almacenamiento específico y singular para este caso en el que sólo se almacenen los coeficientes no nulos de la matriz.

b) Indicar la forma que tendrán, en general, las matrices que se obtendrán como resultado de descomponer la matriz como $\boldsymbol{A} = \boldsymbol{LDU}$.

c) Desarrollar completamente un algoritmo específico y eficiente de resolución del sistema de ecuaciones $\boldsymbol{Ux} = \boldsymbol{b}$, siendo $\boldsymbol{b}$ el vector de términos independientes, $\boldsymbol{x}$ la solución buscada y $\boldsymbol{U}$ la matriz triangular superior obtenida como resultado de factorizar la matriz $\boldsymbol{A}$.

Solución 7.a El algoritmo de producto de matriz por vector sólo necesita conocer los términos no nulos de la matriz ya que los términos no nulos no aportan nada al resultado y se pueden retirar del algoritmo de producto para que se calcule de forma más rápida y eficiente.

Solución 7.b Las matrices resultado de aplicar el algoritmo de factorización mantendrán el perfil por filas en el caso de la matriz triangular inferior y el perfil por columnas en el caso de la matriz triangular superior. Por tanto, las matrices resultado de la factorización tendrán con carácter general la siguiente forma:

$$\boldsymbol{L} = \begin{bmatrix} 1 & & & \\ * & \ddots & & \\ \vdots & \ddots & \ddots & \\ * & \cdots & * & 1 \end{bmatrix}; \quad \boldsymbol{D} = \begin{bmatrix} * & & & \\ & \ddots & & \\ & & \ddots & \\ & & & * \end{bmatrix}; \quad \boldsymbol{U} = \begin{bmatrix} 1 & & & * \\ & \ddots & & \vdots \\ & & \ddots & * \\ & & & 1 \end{bmatrix} \tag{34}$$

Solución 7.c Planteamos la resolución del sistema de ecuaciones triangular superior con la matriz $\boldsymbol{U}$ propuesta en el apartado anterior.

$$\begin{bmatrix} 1 & & & & u_{1n} \\ & \ddots & & & \vdots \\ & & 1 & & u_{n-2,n} \\ & & & 1 & u_{n-1,n} \\ & & & & 1 \end{bmatrix} \begin{Bmatrix} x_1 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ \vdots \\ b_{n-2} \\ b_{n-1} \\ b_n \end{Bmatrix} \tag{35}$$

$$\begin{array}{c} x_1 + u_{1n}\, x_n = b_1 \\ \vdots \\ x_{n-2} + u_{n-2,n}\, x_n = b_{n-2} \\ x_{n-1} + u_{n-1,n}\, x_n = b_{n-1} \\ x_n = b_n \end{array} \tag{36}$$

De modo que el algoritmo de resolución quedaría como:

```
x(n)=b(n)
do i=n-1,1,-1
  x(i)=b(i)-u(i,n)*x(n)
enddo
```

8. Un ingeniero necesita resolver un sistema de ecuaciones con matriz:

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & & & \\ a_{31} & & a_{33} & & \\ \vdots & & & \ddots & \\ a_{n1} & & & & a_{nn} \end{bmatrix} \tag{37}$$

Se pide:

a) Si se pretende descomponer la matriz $\boldsymbol{A}$ como $\boldsymbol{LDU}$, indicar qué forma tendrían las matrices.

b) Desarrollar completamente el algoritmo que permite resolver el sistema de ecuaciones $\boldsymbol{Ux} = \boldsymbol{b}$ siendo $\boldsymbol{U}$ la matriz resultante de factorizar la matriz $\boldsymbol{A}$.

c) Indicar de forma justificada si el método de eliminación de Gauss sin pivotamiento permitiría resolver el sistema de ecuaciones con matriz $\boldsymbol{A}$ de forma eficiente sin operar con los coeficientes nulos.

Solución 8.a La forma que tendrán las matrices triangulares tras la factorización será aquella que conserva el perfil por filas en la parte inferior y el perfil por columnas en la parte superior. Es decir una matriz triangular inferior llena y una triangular superior llena también.

$$\boldsymbol{L} = \begin{bmatrix} 1 & & & \\ * & \ddots & & \\ \vdots & \ddots & \ddots & \\ * & \cdots & * & 1 \end{bmatrix}; \quad \boldsymbol{D} = \begin{bmatrix} * & & & \\ & \ddots & & \\ & & \ddots & \\ & & & * \end{bmatrix}; \quad \boldsymbol{U} = \begin{bmatrix} 1 & * & \cdots & * \\ & \ddots & \ddots & \vdots \\ & & \ddots & * \\ & & & 1 \end{bmatrix} \tag{38}$$

Solución 8.b El algoritmo de resolución del sistema triangular superior

$$\begin{bmatrix} 1 & u_{12} & \cdots & \cdots & u_{1n} \\ & \ddots & \ddots & \vdots & \vdots \\ & & 1 & u_{n-2,n-1} & u_{n-2,n} \\ & & & 1 & u_{n-1,n} \\ & & & & 1 \end{bmatrix} \begin{Bmatrix} x_1 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ \vdots \\ b_{n-2} \\ b_{n-1} \\ b_n \end{Bmatrix} \tag{39}$$

Sería:

$$\begin{aligned} & x_n = b_n \\ & x_i = b_i - \sum_{j=i+1}^{n} u_{ij}\, x_j \qquad i = n-1, \ldots, 1, (-1) \qquad \longleftarrow \text{(hacia atrás)} \end{aligned} \tag{40}$$

Que en lenguaje Fortran quedaría como:

```
x(n)=b(x)
do i=n-1,1,-1
  suma=0.d+00
  do j=i+1,n
    suma=suma+u(i,j)*x(j)
  enddo
```

```
  x(i)=b(i)-suma
enddo
```

Solución 8.c Si se plantea la anulación de los términos de la primera columna (a excepción de la diagonal) no es posible hacerlo de forma eficiente, ya que al anular los términos de la primera columna se convierten en no nulos los términos de la parte inferior de la matriz, y también los de la parte superior. Simplemente con intentar conseguir ceros en la primera columna haremos que toda la matriz restante sea llena.

Sin embargo, existe una alternativa de eliminación más eficiente. Si se plantea la anulación de la parte superior (anular los términos de la primera fila) sí que se puede hacer de forma eficitente ya que se mantienen todos los coeficientes nulos existentes. Para conseguir que el término $a_{1,i}$ se anule hay que combinar la ecuación 1 con la ecuación i. Y este procedimiento hay que repetirlo para todos los coeficientes de la primera fila excepto el de la diagonal principal.
