Métodos directos de resolución de sistemas de ecuaciones lineales: Factorización LU

por Delgado Marcos, A; Lobato Pérez, J; Martínez Campo, A y Rodríguez Robles, P

Resumen

El documento prentende presentar el contexto de la factorización LU en los métodos directos de resolución de sistemas de ecuaciones lineales, así como la descripción de este procedimiento y su ejemplificación mediante problemas realizados con la ayuda del leguaje de programación Python.

1 Sistemas de ecuaciones lineales

En cualquier disciplina de la ingeniería solemos encontrarnos con la resolución de sistemas de ecuaciones lineales de la forma:

$$\mathrm{Ax} = b$$

Donde A es la matriz cuadrada de dimensión $n \times n$ cuyos elementos a_{ij} pueden ser números reales o complejos, mientras que x y b son un par de vectores de dimensión n en los que x representa el conjunto de la solución desconocida y b es un vector dado. Podemos escribir un sistema de ecuaciones como:

$$\begin{aligned} a_{11}x_{11} + a_{12}x_{12} + ... + a_{1n}x_{1n} &= b_1 \\ a_{21}x_{21} + a_{22}x_{22} + ... + a_{2n}x_{2n} &= b_2 \\ &\vdots \\ a_{n1}x_{n1} + a_{n2}x_{n2} + ... + a_{\mathrm{nn}}x_{\mathrm{nn}} &= b_n \end{aligned}$$

También es posible escribir los sistemas usando la notación matricial:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

La resolución de estos sistemas es muy importante ya que se emplea en el análisis de todo tipo de sistemas físicos que se presentan como modelos matemáticos lineales. Así son las estructuras, los sólidos elásticos, el transporte de calor, la mecánica de fluidos, los campos electromagnéticos, circuitos, ect.

En la física de medios continuos (mecánica de fluidos, por ejemplo) el comportamiento del sistema es descrito mediante ecuaciones diferenciales. Como el análisis numérico solo conoce variables discretas (finitas), se utilizan distintos métodos de aproximación que transforman estas expresiones diferenciales en sistemas de ecuaciones algebraicas que los computadores sí pueden resolver.

En esencia, cuando resumimos los sistemas de ecuaciones lineales en la forma $\mathrm{Ax} = b$ encontramos que para el sistema físico descrito, A son las características de éste, b resulta ser la interacción supuesta y x la respuesta del sistema físico a ésta. De este modo, queda justificada la búsqueda de métodos que lleven a la resolución de sistemas simultáneos (misma A) para el análisis repetido del mismo sistema físico en distintas condiciones iniciales con un coste computacional mínimo, como veremos que lo es la factorización LU.

Disponemos de dos tipos de resolución para los sistemas de ecuaciones lineales. Por una parte, los métodos directos son aquellos que utilizan un número finito de pasos para transformar el sistema en otro equivalente que sea más fácil de resolver. Estos pasos se conocen como operaciones elementales y no cambian la solución del sistema, aunque sí pueden cambiar el determinante de la matriz de coeficientes ($|A|$). Distinguimos tres operaciones elementales:

- Intercambio de dos ecuaciones (cambia el signo de $|A|$). $(E_i) \rightarrow (E_j)$
- Multiplicar una ecuación por una constante distinta de cero (multiplica a $|A|$ por la misma constante). $(\lambda E_i) \rightarrow (E_i)$
- Multiplicar una ecuación por una constante distinta de cero y sumarla a otra ecuación (no altera $|A|$). $(E_i + \lambda E_j) \rightarrow (E_i)$

Las operaciones elementales también pueden ser concebidas como el producto de una matriz que llamamos elemental. Es decir, el producto matricial por la matriz identidad a la que se le han aplicado las transformaciones buscadas. Si multiplicamos por la izquierda, los cambios se aplican por filas (nuestras ecuaciones), en el caso de hacerlo por la derecha obtendríamos los mismos cambios pero por columnas.

Por otro lado, los métodos iterados para la resolución de sistemas de ecuaciones lineales comienzan con una aproximación inicial de la solución X_0, que se va refinando a través de cada iteración hasta que se alcanza un criterio de convergencia elegido. Estos métodos son menos eficientes que los directos debido al gran número de iteraciones que necesitamos, aunque resultan muy interesantes cuando se trabaja con matrices con muchos ceros (algo bastante común en los sistemas físicos).

Matrices elementales

Tras tratar las operaciones elementales que podemos aplicar a las matrices de nuestros sistemas, veremos cómo expresar estas transformaciones por medio del producto de matrices elementales. Estas matrices elementales se obtienen de la matriz identidad al aplicar una operación elemental.

Computacionalmente encontramos que las matrices son un operador, es decir, un objeto que realiza una operación buscada sobre un vector. Si lo hace sobre un vector, una matriz también puede actuar sobre otra matriz actuando simultáneamente en todos los vectores:

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\mathrm{Id}_3} \xrightarrow{E_2 \rightarrow E_2 + 6E_1} \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 6 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\text{Matriz elemental}}$$

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathrm{Id}_4} \xrightarrow{E_3 \rightarrow E_3 - 5E_1} \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -5 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{Matriz elemental}}$$

El producto de esta matriz elemental por una matriz equivale a realizar las operaciones elementales sobre ella. De hacerlo por la izquierda el cambio será en las filas. En el caso contrario modificaremos las columnas:

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\text{Matriz elemental}}\begin{pmatrix} 6 & -1 & 1 \\ 7 & 2 & 3 \\ 0 & -3 & -2 \end{pmatrix}=\begin{pmatrix} 6 & -1 & 2 \\ -11 & 5 & -3 \\ 0 & -3 & -2 \end{pmatrix}$$

Esto equivale a la operación $E_2 \to E_2 - 3E_1$

Debemos considerar también el procedimiento para el cálculo de la inversa de una de estas matrices elementales, es decir, cambiar el signo de todos los elementos que no pertenecen a la diagonal principal:

$$E=\begin{pmatrix} 1 & -2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad E^{-1}=\begin{pmatrix} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Es fácil comprobar que el producto $E \cdot E^{-1} = \mathrm{Id}$

1.1 Métodos directos

Podemos resumir los principales métodos empleando la siguiente tabla:

Metodo	Forma inicial	Forma final
Eliminacion gaussiana	$Ax=b$	$Ux=c$
Descomposicion LU	$Ax=b$	$LUx=b$
Eliminacion Gauss − Jordan	$Ax=b$	$Ix=c$

1.1.1 Método de eliminación gaussiana

La explicación de este método tiene un alto contenido didáctico y es con el que todo el mundo está familiarizado. Distiguimos dos partes:

- **Fase de eliminación**: consiste en eliminar las entradas bajo la diagonal de A haciendo uso de las operaciones elementales hasta conseguir $\mathrm{Ux} = c$

  Es de importancia para posteriores técnicas de pivoteo la introducción del término λ que llamaremos multiplicador:

$$\lambda = \frac{a_{ij}}{a_{ii}}$$

La representación simbólica de cada operación es la :

$$E_i \to E_i - \lambda \cdot E_j$$

Llegando a la forma final buscada:

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

Ejemplo 1. Ejemplo de eliminación gaussiana:

$$\begin{pmatrix} 4 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 11 \\ -16 \\ 17 \end{pmatrix} \xrightarrow{E_2 \to E_2 - \frac{-2}{4} E_1}$$

$$\begin{pmatrix} 4 & -2 & 1 \\ 0 & 3 & -1.5 \\ 1 & -2 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 11 \\ -10.5 \\ 17 \end{pmatrix} \xrightarrow{E_3 \to E_3 - \frac{1}{4} E_1}$$

$$\begin{pmatrix} 4 & -2 & 1 \\ 0 & 3 & -1.5 \\ 0 & -1.5 & 3.75 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 11 \\ -10.5 \\ 14.25 \end{pmatrix} \xrightarrow{E_3 \to E_3 - \frac{-1.5}{3} E_2}$$

$$\begin{pmatrix} 4 & -2 & 1 \\ 0 & 3 & -1.5 \\ 0 & 0 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 11 \\ -10.5 \\ 9 \end{pmatrix}$$

La rutina pretende ilustrar la implementación en Python del método de eliminación gaussiana.

Algoritmo 1

```
def eliminacion_gaussiana(A, b):
    """
    * Toma:
    A --> matriz de coeficientes
    b --> matriz de términos independientes

    * Devuelve:
    A y b tras aplicar la eliminación
    """


    A = A.astype(float) # Los coeficientes de las matrices pasan a expresarse
                        como float
    b = b.astype(float) # Los coeficientes de las matrices pasan a expresarse
                        como float

    n = A.shape[0] # n es el número de filas de A (número de ecuaciones)


    # __ELIMINACIÓN GAUSIANA__    Ax=b --> Ux=c
    # Para las entradas que esten:
    # * En todas las columnas menos la última
    for j in range(0,n-1):
        # * En cada fila con índice superior al de la columna
        for i in range(j+1,n):
            # Si la entrada no es nula
            if A[i,j] != 0.0:
                # Multiplicador, A[j,j] es el pivote, A[i, j] el término
                mult = A[i,j]/A[j,j]
                # A toda la fila de la entrada le restamos la fila superior
                por el multiplicador
              A[i,:] = A[i,:] - mult*A[j,:]
                b[i] = b[i] - mult*b[j]

    # Devuelve
    return A, b
```

- **Sustitución regresiva**: Comenzamos a sustituir las incógnitas empezando por la última fila. Ésta es la forma en la que el computador resuelve los sistemas una vez que están escalonados:

$$x_n = \frac{1}{u_{\mathrm{nn}}} c_n,$$

$$x_i = \frac{1}{u_{\mathrm{ii}}} \left(c_i - \sum_{j=i+1}^{n} u_{\mathrm{ij}} x_j \right), i = n-1, ..., 1$$

Ejemplo 2. Ejemplo de sustitución regresiva:

$$\left.\begin{array}{rrcr} 4x_1 & -2x_2 & +x_3 & = & 11 \\ & 3x_2 & -1.5x_3 & = & -10.5 \\ & & 3x_3 & = & 9 \end{array}\right\} \; x_3 = \frac{1}{3}\cdot 9 = 3$$

$$\left.\begin{array}{rrcr} 4x_1 & -2x_2 & +x_3 & = & 11 \\ & 3x_2 & -1.5x_3 & = & -10.5 \\ & & x_3 & = & 3 \end{array}\right\} \; x_2 = \frac{1}{3}\cdot(-10.5 + 1.5\cdot 3) = -2$$

$$\left.\begin{array}{rrcr} 4x_1 & -2x_2 & +x_3 & = & 11 \\ & & x_2 & = & -2 \\ & & x_3 & = & 3 \end{array}\right\} \; x_1 = \frac{1}{4}\cdot(11 - 1\cdot 3 + 2\cdot(-2)) = 1$$

$$\left\{\begin{array}{rcr} x_1 & = & 1 \\ x_2 & = & -2 \\ x_3 & = & 3 \end{array}\right\}$$

La rutina pretende ilustrar la implementación en Python del método de sustitución regresiva.

Algoritmo 2

```
def sustitucion_regresiva(A, b):
    """
    * Toma:
    A --> matriz de coeficientes escalonada
    b --> matriz de términos independientes escalonada

    * Devuelve:
    b --> vector X de las soluciones (Ax=b)
    """

    A = A.astype(float) # Los coeficientes de las matrices pasan a expresarse
                        como float
    b = b.astype(float) # Los coeficientes de las matrices pasan a expresarse
                        como float

    n = A.shape[0] # n es el número de filas de A (número de ecuaciones)


    # SUSTUCIÓN REGRESIVA
    # Desde la última fila hasta la primera
    for j in range(n-1,-1,-1):
        b[j] = (b[j]-np.dot(A[j, j+1:], b[j+1:]))/A[j,j]

    return b
```

1.1.2 Método de Gauss-Jordan

La eliminacion de Gauss Jordan es una modificación de la eliminación gaussiana, aunque también estamos muy familiarizados con ella. El método de Gauss transforma A en una matriz triangular superior. El método de Gauss-Jordan continúa el proceso hasta obtener una matriz diagonal, simplificable en una matriz identidad:

$$\left(\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} l_{11} & l_{12} & l_{13} & b'_1 \\ 0 & l_{22} & l_{23} & b'_2 \\ 0 & 0 & l_{33} & b'_3 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} d_{11} & 0 & 0 & b''_1 \\ 0 & d_{22} & 0 & b''_2 \\ 0 & 0 & d_{33} & b''_3 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 1 & 0 & 0 & c_1 \\ 0 & 1 & 0 & c_2 \\ 0 & 0 & 1 & c_3 \end{array}\right)$$

Ir obteniendo *unos* en la diagonal es otra manera de desarrollar el método (tal y como se verá en el ejemplo).

Este método es muy usado sobretodo en el cálculo de matrices inversas, buscando transformaciones que conviertan la matriz A en la matriz identidad:

$$\left(\begin{array}{c|ccc} A & 1 & 0 & 0 \\ & 0 & 1 & 0 \\ & 0 & 0 & 1 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 1 & 0 & 0 & \\ 0 & 1 & 0 & A^{-1} \\ 0 & 0 & 1 & \end{array}\right)$$

Ejemplo 3. Vamos a mostrar un ejemplo de una eliminación de Gauss-Jordan, ya que el procedimiento es muy similar al de Gauss y los algoritmos se pueden deducir:

$$\left\{\begin{array}{l} x+y+z=5 \\ 2x+3y+5z=8 \\ 4x+5z=2 \end{array}\right\} \xrightarrow{\text{forma matricial}} \left(\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 2 & 3 & 5 & 8 \\ 4 & 0 & 5 & 2 \end{array}\right)$$

Primero obtenemos la matriz triangular superior:

$$\left(\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 2 & 3 & 5 & 8 \\ 4 & 0 & 5 & 2 \end{array}\right) \xrightarrow{E'_2=E_2-2E_1} \left(\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 4 & 0 & 5 & 2 \end{array}\right) \xrightarrow{E'_3=E_3-4E_1} \left(\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & -4 & 1 & -18 \end{array}\right) \xrightarrow{E'_3=E_3+4E_2} \left(\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 13 & -26 \end{array}\right)$$

A continuación vamos buscando *ceros* en la diagonal principal y sustituyendo hacia atrás:

$$\left(\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 13 & -26 \end{array}\right) \xrightarrow{E'_3=\frac{E_3}{13}} \left(\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 1 & -2 \end{array}\right) \xrightarrow[E'_1=E_1-3E_3]{E'_2=E_2-3E_3} \left(\begin{array}{ccc|c} 1 & 1 & 0 & 7 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & -2 \end{array}\right) \xrightarrow{E'_1=E_1-E_2} \left(\begin{array}{ccc|c} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & -2 \end{array}\right)$$

De esta manera obtenemos la solución del sistema:

$$\left\{\begin{array}{l} x=3 \\ y=4 \\ z=-2 \end{array}\right\}$$

1.1.3 Sustitución progresiva

Es un buen momento para introducir el algoritmo de la sustitución progresiva. El procedimiento el similar al de la sustitución regresiva y su uso será vital el los posteriores métodos.

$$x_1 = \frac{1}{u_{11}} b_{1,}$$

$$x_i = \frac{1}{u_{\mathrm{ii}}} \left(b_i - \sum_{j=1}^{n} u_{\mathrm{ij}} x_j \right), i = 2, 3, ..., n$$

Ejemplo 4. Ejemplo de sustitución progresiva:

$$\left.\begin{array}{rrrcr} 3x_1 & & & = & 9 \\ -1.5x_1 & 3x_2 & & = & -10.5 \\ x_1 & -2x_2 & +4x_3 & = & 11 \end{array}\right\} \qquad x_1 = \frac{1}{3} \cdot 9 = 3$$

$$\left.\begin{array}{rrrcr} x_1 & & & = & 3 \\ -1.5x_1 & 3x_2 & & = & -10.5 \\ x_1 & -2x_2 & +4x_3 & = & 11 \end{array}\right\} \quad x_2 = \frac{1}{3} \cdot (-10.5 + 1.5 \cdot 3) = -2$$

$$\left.\begin{array}{rrrcr} x_1 & & & = & 3 \\ & x_2 & & = & -2 \\ x_1 & -2x_2 & +4x_3 & = & 11 \end{array}\right\} \quad x_3 = \frac{1}{4} \cdot (11 - 1 \cdot 3 + 2 \cdot (-2)) = 1$$

$$\left\{\begin{array}{rcr} x_1 & = & 3 \\ x_2 & = & -2 \\ x_3 & = & 1 \end{array}\right\}$$

La rutina pretende ilustrar la implementación en Python del método de sustitución progresiva.

Algoritmo 3

```
def sustitucion_progresiva(A, b):
    """
    * Toma:
    A --> matriz de coeficientes escalonada
    b --> matriz de términos independientes escalonada

    * Devuelve:
    x --> vector X de las soluciones (Ax=b)
    """

    A = A.astype(float) # Los coeficientes de las matrices pasan a expresarse
                        como float
    b = b.astype(float) # Los coeficientes de las matrices pasan a expresarse
                        como float

    n = A.shape[0] # n es el número de filas de A (ecuaciones)

    x = np.eye(n,1).astype(float) # Vector para almacenar la solución
    # SUSTUCIÓN PROGRESIVA
    # Desde la última fila hasta la primera

    x[0] = b[0]
    for j in range(1,n):
        x[j] = b[j]-np.dot(A[j,0:j], x[0:j])

    return x
```

1.2 Pivoteo

El pivoteo dentro de la factorización LU es un punto muy importante. Parte de dicha factorización reside en ir haciendo una eliminacion Gaussiana con sustitución hacia atrás.

Para una matriz $A \in \mathbb{R}^{nxn}$, la factorización de Gauss existe y es única si y solo si las principales submatrices A_i de A (de orden $i = 1,2,...,n-1$; obtenidas restringiendo A a su primera fila y columna i) son invertibles.

Si esto no se cumpliera, es decir, si $a_{ii} = 0$ (cualquier elemento de la diagonal principal que se use para hacer el pivoteo), entonces no se puede seguir con la clásica eliminación gaussiana. Por ello surgen los distintos tipos de pivoteo, siendo dificil encontrar un buen pivote.

Ejemplo 5. Veamos un ejemplo en el que por la elección de un mal pivote (y el consecuente redondeo) se obtienen resultados muy diferentes de los reales:

Primero vamos a resolver el sistema por Gauus tomando como pivote a_{11}=0.0030000 y redondeando a 4 cifras:

$$\left\{ \begin{array}{l} 0.003x_1 + 59.14x_2 = 59.17 \\ 5.291x_1 - 6.130x_2 = 46.78 \end{array} \right\} \xrightarrow[m_{21}=\frac{a_{jk}}{a_{kk}}=\frac{a_{21}}{a_{11}}=\frac{5.291}{0.003}=1764]{E_2'=E_2-m_{21}E_1} \left\{ \begin{array}{l} 0.003x_1 + 59.14x_2 \approx 59.17 \\ -104300x_2 \approx -104400 \end{array} \right\} \rightarrow \left\{ \begin{array}{l} x_1 \approx -10 \\ x_2 \approx 1.001 \end{array} \right\}$$

Ahora vamos a resolver el sistema sin redondeo (lo cual hace más tedioso el proceso):

$$\left\{\begin{array}{l} 0.003x_1+59.14x_2=59.17 \\ 5.291x_1-6.130x_2=46.78 \end{array}\right\} \xrightarrow[m_{21}=\frac{a_{21}}{a_{11}}=\frac{5.291}{0.003}=1763.66]{E_2'=E_2-m_{21}E_1} \left\{\begin{array}{l} 0.003x_1+59.14x_2=59.17 \\ -104309.376x_2=-104309.376 \end{array}\right\} \rightarrow \left\{\begin{array}{l} x_1=10 \\ x_2=1 \end{array}\right\}$$

Debido a que el a_{11} sea un valor muy pequeño y sea elegido como pivote, hace que haya una diferencia entre x_1 aproximado y real (de 20 unidades, lo cual es un gran error). Por lo tanto se emplean distintos tipos de pivoteo para solucionar este problema.

1.2.1 Pivoteo parcial

El pivoteo parcial cosiste en analizar los elementos de la columna que estén por debajo del a_{ij} (siendo $i=j$, es decir, por debajo del elemento de la diagonal principal) y seleccciona el elemento a_{pj} con un mayor valor absoluto que el de a_{ij}, que será el pivote:

$$a_{pj}=\max_{i=j,\ldots,n}\{|a_{ij}|\} \rightarrow \text{intercambio filas } E_p \leftrightharpoons E_i$$

Ejemplo 6. Resolvamos de nuevo el sistema aplicando pivoteo parcial y redondeo a 4 cifras:

$$a_{pj}=\max\{|a_{11}|,|a_{21}|\}=\max\{|0.003|,|5.291|\}=|5.291|=|a_{21}|$$

Por lo tanto vamos a intercambiar $E_1 \rightleftharpoons E_2$

$$\left\{\begin{array}{l} 5.291x_1-6.130x_2=46.78 \\ 0.003x_1+59.14x_2=59.17 \end{array}\right\} \xrightarrow[m_{21}=\frac{a_{jk}}{a_{kk}}=\frac{a_{21}}{a_{11}}=\frac{0.003}{5.291}=0.000567]{E_2'=E_2-m_{21}E_1} \left\{\begin{array}{l} 5.291x_1-6.130x_2=46.78 \\ 59.14x_2=59.14 \end{array}\right\} \rightarrow \left\{\begin{array}{l} x_1=10 \\ x_2=1 \end{array}\right\}$$

Para hacer un cambio de filas, se emplea una matriz P_n (matriz identidad a la que se le han aplicado cambios) para que al multiplicarla por la izquierda los cambios afecten a las filas:

$$\left.\begin{array}{l} Id_2=\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ S=\begin{pmatrix} 0.003 & 59.14 \\ 5.291 & -6.130 \end{pmatrix} \end{array}\right\} \xrightarrow{E_1^{(P)} \leftrightharpoons E_2^{(P)}} P=\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \rightarrow PS=\begin{pmatrix} 5.291 & -6.130 \\ 0.003 & 59.14 \end{pmatrix}$$

Una buena matriz P cumple $P^{-1}=P^t$

Vamos a tomar de nuevo el sistema de ecuaciones anterior (habiendo multiplicado la primera fila por 10^4) y resolverlo:

$$\left\{\begin{array}{l} 30.00x_1+591400x_2=591700 \\ 5.291x_1-6.130x_2=46.78 \end{array}\right\} \xrightarrow[m_{21}=\frac{a_{21}}{a_{11}}=\frac{5.291}{30.00}=0.1764]{E_2'=E_2-m_{21}E_1} \left\{\begin{array}{l} 30.00x_1+591400x_2=591700 \\ -104300x_2=-104400 \end{array}\right\} \rightarrow \left\{\begin{array}{l} x_1=-10 \\ x_2=1.001 \end{array}\right\}$$

Como el valor a_{11} es el mayor de su columna y es distinto de cero, no se aplica el pivoteo parcial, pero los valores obtenidos vuelven a ser incorrectos. Para solucionar esto se aplica el pivoteo parcial escalado.

1.2.2 Pivoteo parcial escalado

El pivoteo parcial escalado es similar al pivoteo parcial, solo que se escoge como pivote el elemento con mayor valor absoluto respecto a todos los elementos de la fila. Primero se escoge el elemento de la fila con mayor tamaño relativo de cada fila de la matriz:

$$S_i=\max_{j=1,\ldots,n}\{|a_{ij}|\}$$

Suponiendo que no hay ningún $S_i = 0$, aplicamos la siguiente fórmula:

$$\frac{|a_{11}|}{S_1}, \frac{|a_{21}|}{S_2}, ..., \frac{|a_{i1}|}{S_i} \text{ (siendo } i = 1, 2, ..., n) \xrightarrow{p=\text{fila pivote}} \frac{|a_{p1}|}{S_p} = \max_{k=1,...,n} \frac{|a_{k1}|}{S_k} \rightarrow \text{intercambio } E_p \leftrightharpoons E_k$$

Ejemplo 7. Resolvamos el sistema multiplicado por 10^4 que anteriormente dio resultados in correctos:

$$\left\{ \begin{array}{l} 30.00x_1 + 591400x_2 = 591700 \\ 5.291x_1 - 6.130x_2 = 46.78 \end{array} \right\} \xrightarrow[\frac{|a_{21}|}{S_2} = \frac{5.291}{\max\{|5.291|,|6.130|\}} = \frac{5.291}{6.130} = 0.8631]{\frac{|a_{11}|}{S_1} = \frac{30.00}{\max\{|30.00|,|591400|\}} = \frac{30.00}{591400} = 0.507310^{-4}} \left\{ \begin{array}{l} 5.291x_1 - 6.130x_2 = 46.78 \\ 30.00x_1 + 591400x_2 = 591700 \end{array} \right\}$$

Como $\frac{|a_{11}|}{S_1} < \frac{|a_{21}|}{S_2}$, se lleva a cabo el cambio $E_1 \rightleftharpoons E_2$ (para que así a_{21} sea el pivote). Despejando se obtienen las soluciones correctas:

$$\left\{ \begin{array}{l} 5.291x_1 - 6.130x_2 = 46.78 \\ 30.00x_1 + 591400x_2 = 591700 \end{array} \right\} \xrightarrow[m_{21} = \frac{a_{jk}}{a_{kk}} = \frac{a_{21}}{a_{11}} = \frac{30.00}{5.291} = 5.670]{E_2' = E_2 - m_{21}E_1} \left\{ \begin{array}{l} 5.291x_1 - 6.130x_2 = 46.78 \\ 591434.76x_2 = 591434.76 \end{array} \right\} \rightarrow \left\{ \begin{array}{l} x_1 = 10 \\ x_2 = 1 \end{array} \right\}$$

Existe otro tipo de pivoteo, denominado pivoteo completo, que analiza filas y columnas. Su uso, debido al gran número de comparaciones que hay que llevar a cabo, se reserva solamente para algunos sistemas en los que la precisión sea esencial.

2 Factorización LU

Para la factorización LU supondremos que la matriz A de $\mathrm{Ax} = b$ se puede factorizar como el producto de una matriz triangular inferior L y una matriz triangular superior U

$$A = \mathrm{LU}$$

Siendo esto posible, podemos representar nuestro sistema como:

$$\mathrm{LUx} = b$$

Denominando y a la matriz columna de n filas resultado del producto de las matrices Ux, tenemos que la ecuación anterior se puede escribir del siguiente modo:

$$\mathrm{Ly} = b$$

Trabajaremos con $\mathrm{LUx} = b$ y $\mathrm{Ly} = b$, para resolverlas seguiremos estos pasos:

- Primero obtendremos y utilizando sustitución progresiva en la ecuación $\mathrm{Ly} = b$
- Posteriormenrte obtendremos los valores de x aplicando el algoritmo de sustitución regresiva a la ecuación $\mathrm{Ux} = y$

Encontramos que es posible obtener las matrices L y U mediante la eliminación gaussiana.

Hemos visto que cualquier matriz A que podamos factorizar en la forma $A = \mathrm{LU}$ es válida para aplicar este método. Pero la ecuación anterior no determina la forma única de L y de U:

$$\begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \cdot \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Haciendo uso de la multiplicación de matrices obtenemos las siguientes ecuaciones no lineales:

$$\begin{aligned}
l_{11}\cdot u_{11} &= a_{11}\\
l_{11}\cdot u_{12} &= a_{12}\\
l_{11}\cdot u_{13} &= a_{13}\\
l_{21}\cdot u_{11} &= a_{21}\\
l_{21}\cdot u_{12}+l_{22}\cdot u_{22} &= a_{22}\\
l_{21}\cdot u_{13}+l_{22}\cdot u_{23} &= a_{23}\\
l_{31}\cdot u_{11} &= a_{31}\\
l_{31}\cdot u_{12}+l_{32}\cdot u_{22} &= a_{32}\\
l_{31}\cdot u_{13}+l_{32}\cdot u_{23}+l_{33}\cdot u_{33} &= a_{33}
\end{aligned}$$

Este sistema es indeterminado, contamos con más incógnitas que ecuaciones. De este modo la búsqueda de $A = \mathrm{LU}$ arroja infinitas combinaciones encerradas en el anterior sistema (que es no lineal). Así, para obtener la factorización debemos imponer a las matrices L y U alguna restricción arbitraria que simplifique las ecuaciones y lleve a una solución única. De aquí nacen los distintos métodos para la descomposición LU

Nombre	Restriccion
Descomposicion de Doolittle	$L_{\mathrm{ii}} = 1 \rightarrow i = 1, 2..., n$
Descomposicion de Crout	$U_{\mathrm{ii}} = 1 \rightarrow i = 1, 2..., n$
Descomposicion de Choleski	$L = U^T$

2.1 Factorización LU en términos de operaciones elementales. Un primer algoritmo

La búsqueda de la descomposición $A = \mathrm{LU}$ nos lleva al primero de los algoritmos. Para ello utilizaremos el concepto de matriz elemental que antes hemos repasado.

Vamos a descomponer la matriz:

$$A = \begin{pmatrix} -4 & -3 & 1 \\ 8 & 11 & -1 \\ 4 & 18 & 5 \end{pmatrix}$$

En cada uno de nuestros pasos escribiremos A como el producto LU. Tomaremos L como triangular inferior y U será convertida en triangular superior paso a paso.

El primer paso es partir de $L = \mathrm{Id}_3$, $U = A$:

$$A = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} -4 & -3 & 1 \\ 8 & 11 & -1 \\ 4 & 18 & 5 \end{pmatrix}}_{U}$$

Para lograr la matriz triangular superior en U comenzaremos por eliminar la entrada $U_{2,1}$ aplicando la operación elemental $E_2 \to E_2 + 2E_1$. Aplicaremos esta operación usando la matriz elemental:

$$E = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Debemos multiplicar E por el lado izquierdo de la matriz A. Pero para seguir cumpliendo la igualdad $A = \mathrm{LU}$ utilizaremos el producto por su inversa $A = \mathrm{LE}^{-1}\mathrm{EU}$:

$$A = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{E^{-1}} \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{E} \underbrace{\begin{pmatrix} -4 & -3 & 1 \\ 8 & 11 & -1 \\ 4 & 18 & 5 \end{pmatrix}}_{E}$$

Simplificando la expresión, U sufrirá la transformación buscada ($E_2 \to E_2 + 2E_1$):

$$A = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} -4 & -3 & 1 \\ 0 & 5 & 1 \\ 4 & 18 & 5 \end{pmatrix}}_{U}$$

Tras el primer paso, A sigue siendo igual a LU, L es triangular inferior y seguimos transformando U en triangular superior. Eliminaremos $U_{3,1}$ con la operación elemental $E_3 \to E_3 + E_1$ que se expresa mediante la matriz elemental:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

Volveremos a usar la inversa de ésta para no romper la igualdad:

$$A = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{L} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \underbrace{\begin{pmatrix} -4 & -3 & 1 \\ 0 & 5 & 1 \\ 4 & 18 & 5 \end{pmatrix}}_{U}$$

Esto lleva a la operación $E_3 \to E_3 + E_1$ en las filas de U y $C_3 \to C_3 + C_1$ en las columnas de L:

$$A = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} -4 & -3 & 1 \\ 0 & 5 & 1 \\ 0 & 15 & 6 \end{pmatrix}}_{U}$$

Los resultados son similares a los del paso anterior. Pasamos a eliminar $U_{3,2}$ aplicando la operación $E_3 \to E_3 - 3E_2$ mediante la matriz elemental:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{pmatrix}$$

Continuamos con el procedimiento

$$A = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}}_{L} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 3 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{pmatrix} \underbrace{\begin{pmatrix} -4 & -3 & 1 \\ 0 & 5 & 1 \\ 0 & 15 & 6 \end{pmatrix}}_{U}$$

Finalmemte, obtenemos la operación $E_3 \to E_3 - 3E_2$ en las filas de U y $C_3 \to C_3 - 3C_2$ en las columnas de L

$$A = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 3 & 1 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} -4 & -3 & 1 \\ 0 & 5 & 1 \\ 0 & 0 & 3 \end{pmatrix}}_{U}$$

Así hemos llegado a la solución que buscábamos.

2.2 Permutación

Aunque este método resulta muy cómodo, en ocasiones tenemos algunos problemas con la situación de algún 0 en la matriz, que nos impide desarrollar dicho proceso adecuadamente. Por ello, debemos hacer algunos ajustes y recurrir a la llamada matriz de permutación P. Para ello utilizamos la siguiente expresión:

$$PA = LU$$

donde la matriz P no es más que una variación de la matriz Id_n a la que se realizan los cambios de filas deseados y se introduce a la izquierda de la matriz A, colocando sus respectivas filas de forma adecuada. Además, cabe destacar que para que P sea una buena matriz de permutación, debe tener inversa y se debe cumplir que $P^{-1} = P^t$.

Así, de forma parecida a como procedemos con el método LU planteamos una serie de expresiones para acabar consiguiendo de nuevo dos sistemas de ecuaciones que nos permitan resolver el ejercicio:

$$A \cdot X = b \to P \cdot A \cdot X = P \cdot b \xrightarrow{A = L \cdot U} L \cdot U \cdot X = P \cdot b \to \left\{ \begin{array}{l} L \cdot Y = P \cdot b \\ U \cdot X = Y \end{array} \right\}$$

En ambos métodos se tienen i-1 sumas; i-1 productos y 1 división. Por tanto el número de operaciones realizadas es:

$$\sum_{i=1}^{n} 1 + 2\sum_{i=1}^{n} (i-1) = 2\sum_{i=1}^{n} (i-n) = n^2 \text{ operaciones}$$

Ejemplo 8. Ejemplo de permutación:

$$A=\begin{pmatrix} 0 & 5 & 2 \\ 1 & 3 & -6 \\ -2 & 4 & 9 \end{pmatrix}, b=\begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}, X=\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$U_{\text{inicial}}=\begin{pmatrix} 0 & 5 & 2 \\ 1 & 3 & -6 \\ -2 & 4 & 9 \end{pmatrix} \xrightarrow{F_1 \leftrightarrows F_3} \begin{pmatrix} -2 & 4 & 9 \\ 1 & 3 & -6 \\ 0 & 5 & 2 \end{pmatrix} \xrightarrow{F_2 \to F_1+2F_2} \begin{pmatrix} -2 & 4 & 9 \\ 0 & 10 & -3 \\ 0 & 5 & 2 \end{pmatrix} \xrightarrow{F_3 \to F_2-2F_3}$$

$$\xrightarrow{F_3 \to F_2-2F_3} \begin{pmatrix} -2 & 4 & 9 \\ 0 & 10 & -3 \\ 0 & 0 & -7 \end{pmatrix}$$

$$L_{\text{inicial}}=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \xrightarrow{\text{Sin cambios}} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \xrightarrow{\text{Multiplicador 1}} \begin{pmatrix} 0 & 0 & 0 \\ -2 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \xrightarrow{\text{Multiplicador 2}}$$

$$\xrightarrow{\text{Multiplicador 2}} \begin{pmatrix} 0 & 0 & 0 \\ -2 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix} \xrightarrow{+\text{Identidad}} \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 2 & 1 \end{pmatrix}$$

$$P_{\text{inicial}}=\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \xrightarrow{F_1 \leftrightarrows F_3} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \xrightarrow{\text{Sin cambios}} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \xrightarrow{\text{Sin cambios}} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$U=\begin{pmatrix} -2 & 4 & 9 \\ 0 & 10 & -3 \\ 0 & 0 & -7 \end{pmatrix}, L=\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 2 & 1 \end{pmatrix}, P=\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, y=\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

$$\left.\begin{array}{l} Ax=b \\ PAx=Pb \\ PA=LU \\ LUx=Pb \end{array}\right\} \to \left\{\begin{array}{l} Lx=Pb \\ Ux=y \end{array}\right\}$$

De este modo se plantean los dos sistemas de ecuaciones, se resuelven y se obtienen las soluciones del problema.

2.3 Descomposición de Crout

En el método de Crout, la matriz L es una matriz triangular inferior, mientras que la matriz U es una matriz trianguilar superior con elementos unitarios en la diagonal principal. Presentan la siguiente forma:

$$L = \begin{pmatrix} L_{11} & 0 & 0 & ... & 0 \\ L_{21} & L_{22} & 0 & ... & 0 \\ L_{31} & L_{32} & L_{33} & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ L_{n1} & L_{n2} & L_{n3} & ... & L_{\mathrm{nn}} \end{pmatrix} \qquad U = \begin{pmatrix} 1 & U_{12} & U_{13} & ... & U_{1n} \\ 0 & 1 & U_{23} & ... & U_{2n} \\ 0 & 0 & 1 & ... & U_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & ... & U_{\mathrm{nn}} \end{pmatrix}$$

Una vez planteados estos esquemas, se utilizará la condición existente de que el producto de ambas matrices triangulares debe ser igual a la matriz de coeficientes A. De este modo se plantean las ecuaciones, que nos permiten determinar el valor de los términos de cada matriz. Además, es aconsejable que esta serie de cálculos se hagan en un determinado orden para poder obtener directamente los valores sin que aparezcan varias incógnitas en las expresiones, de forma que se alternen el cálculo de columnas de L con las propias de U:

$$\begin{pmatrix} L_{11} & 0 & 0 & 0 \\ L_{21} & L_{22} & 0 & 0 \\ L_{31} & L_{32} & L_{33} & 0 \\ L_{41} & L_{42} & L_{43} & L_{44} \end{pmatrix} \cdot \begin{pmatrix} 1 & U_{12} & U_{13} & U_{14} \\ 0 & 1 & U_{23} & U_{24} \\ 0 & 0 & 1 & U_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

- Se calcula la primera columna de L:

$$\begin{aligned} L_{11} &= a_{11} \\ L_{21} &= a_{21} \\ L_{31} &= a_{31} \\ L_{41} &= a_{41} \end{aligned}$$

- A continuación se procede con la primera fila de U:

$$\begin{aligned} L_{11} \cdot U_{12} = a_{12} \rightarrow U_{12} &= \frac{a_{12}}{L_{11}} \\ L_{11} \cdot U_{13} = a_{13} \rightarrow U_{13} &= \frac{a_{13}}{L_{11}} \\ L_{11} \cdot U_{14} = a_{14} \rightarrow U_{14} &= \frac{a_{14}}{L_{11}} \end{aligned}$$

- Después se van planteando el resto de ecuaciones (cuyo número dependerá de la dimensión de la matriz de coeficientes) y despejando términos hasta obtener todos los valores de las matrices L y U.
- Una vez que se tienen ambas matrices, se puede realizar la factorización LU de forma normal como se explicó anteriormente.

Ejemplo 9. Descomposición de Crout:

$$A = \begin{pmatrix} 1 & 5 & 3 \\ -1 & 0 & 1 \\ 3 & 2 & 4 \end{pmatrix}$$

Se plantea la ecuación $\mathrm{LU} = A$:

$$\begin{pmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \cdot \begin{pmatrix} 1 & U_{12} & U_{13} \\ 0 & 1 & U_{23} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 5 & 2 \\ -1 & 0 & 1 \\ 3 & 2 & 4 \end{pmatrix}$$

A continuación se van resolviendo las ecuaciones y se obtienen los valores de cada término de ambas matrices triangulares:

$$\left\{ \begin{array}{c} L_{11} = 1 \\ L_{21} = -1 \\ L_{31} = 3 \end{array} \right\}$$

$$\left\{ \begin{array}{l} U_{12} = \dfrac{5}{L_{11}} \\ U_{13} = \dfrac{2}{L_{11}} \end{array} \right\} \rightarrow \left\{ \begin{array}{l} U_{12} = 5 \\ U_{13} = 2 \end{array} \right\}$$

$$\{\ L_{21} \cdot U_{12} + L_{22} = 0\ \} \rightarrow \{\ L_{22} = 5\ \}$$

$$\{\ L_{21} \cdot U_{13} + L_{22} \cdot U_{23} = 1\ \} \rightarrow \left\{\ U_{23} = \frac{3}{5}\ \right\}$$

$$\left\{ \begin{array}{l} L_{31} \cdot U_{12} + L_{32} = 2 \\ L_{31} \cdot U_{13} + L_{32} \cdot U_{23} + L_{33} = 4 \end{array} \right\} \rightarrow \left\{ \begin{array}{l} L_{32} = -13 \\ L_{33} = 29/5 \end{array} \right\}$$

Una vez que se obtienen todos los valores tenemos las matrices L y U:

$$\begin{pmatrix} 1 & 5 & 2 \\ -1 & 0 & 1 \\ 3 & 2 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 5 & 0 \\ 3 & -13 & \dfrac{29}{5} \end{pmatrix} \cdot \begin{pmatrix} 1 & 5 & 2 \\ 0 & 1 & \dfrac{3}{5} \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{array}{c} A\,x = b \\ L\,U\,x = b \end{array} \rightarrow \left\{ \begin{array}{l} L\,y = b \\ U\,x = y \end{array} \right\}$$

Se plantean los dos sistemas de ecuaciones y se resuelven obteniendo las soluciones del problema.

2.4 Descomposición de Doolittle

En este método, la diagonal principal de la matriz L está compuesta de *unos*:

$$L=\begin{pmatrix} 1 & 0 & 0 & ... & 0 \\ L_{21} & 1 & 0 & ... & 0 \\ L_{31} & L_{32} & 1 & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ L_{n1} & L_{n2} & L_{n3} & ... & 1_{nn} \end{pmatrix} \qquad U=\begin{pmatrix} U_{11} & U_{12} & U_{13} & ... & U_{1n} \\ 0 & U_{22} & U_{23} & ... & U_{2n} \\ 0 & 0 & U_{33} & ... & U_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & ... & U_{\mathrm{nn}} \end{pmatrix}$$

- Para este tipo de descomposición, tomando una matriz 3x3 como ejemplo, las matrices L y U son de la forma:

$$L=\begin{pmatrix} 1 & 0 & 0 \\ L_{21} & 1 & 0 \\ L_{31} & L_{32} & 1 \end{pmatrix}; U=\begin{pmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{pmatrix}$$

- Aplicando la igualdad genérica A=LU:

$$A=\begin{pmatrix} U_{11} & U_{12} & U_{13} \\ U_{11}L_{21} & U_{12}L_{21}+U_{22} & U_{13}L_{21}+U_{23} \\ U_{11}L_{31} & U_{12}L_{31}+U_{22}L_{32} & U_{13}L_{31}+U_{23}L_{32}+U_{33} \end{pmatrix}$$

- Aplicando eliminación gaussiana:

$$\begin{pmatrix} U_{11} & U_{12} & U_{13} \\ U_{11}L_{21} & U_{12}L_{21}+U_{22} & U_{13}L_{21}+U_{23} \\ U_{11}L_{31} & U_{12}L_{31}+U_{22}L_{32} & U_{13}L_{31}+U_{23}L_{32}+U_{33} \end{pmatrix} \xrightarrow[F_3\to F_3-L_{31}F_1]{F_2\to F_2-L_{21}F_1} \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & U_{22}L_{32} & U_{23}L_{32}+U_{33} \end{pmatrix} \xrightarrow{F_3\to F_3-L_{32}F_2} \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{pmatrix}$$

- Es interesante al realizar la eliminación gaussiana, en vez de escribir los ceros, escribir los coeficientes que hemos aplicado para anular cada uno de los elementos. Así, finalmente nos quedaría una matriz de la siguiente forma:

$$(\ L|U\ )=\begin{pmatrix} U_{11} & U_{12} & U_{13} \\ L_{21} & U_{22} & U_{23} \\ L_{31} & L_{32} & U_{33} \end{pmatrix}$$

- Podemos hacer tres observaciones a raíz de esto:
  - → La matriz U buscada es la que resulta al aplicar la eliminación gaussiana a la matriz A
  - → L_{ij} es el coeficiente que elimina el elemento A_{ij}
  - → Es imprescindible que los elementos de L en la matriz L/U sean tomados con el signo que posean excluyendo el signo de la resta, es decir:
    - Si la operación es de resta, tomaremos L_{ij}
    - Si la operación es de suma, tomaremos $-L_{ij}$

Ejemplo 10. Ejemplo descomposición Doolittle.

Resolver el sistema $A\,x = b$:

$$A = \begin{pmatrix} 1 & 5 & 2 \\ -1 & 0 & 1 \\ 3 & 2 & 4 \end{pmatrix} B = \begin{pmatrix} 5 \\ 8 \\ -7 \end{pmatrix}, x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$\begin{gathered} A\,x = b \\ A = L\,U \\ L\,U\,x = b \\ \downarrow \\ \left\{ \begin{array}{c} L\,y = b \\ U\,x = y \end{array} \right\} \end{gathered}$$

$$U_{\text{inicial}} = \begin{pmatrix} 1 & 5 & 2 \\ -1 & 0 & 1 \\ 3 & 5 & 4 \end{pmatrix} \xrightarrow{F_2 \to F_1 + F_2} \begin{pmatrix} 1 & 5 & 2 \\ 0 & 5 & 3 \\ 3 & 5 & 4 \end{pmatrix} \xrightarrow{F_3 \to F_3 - 3F_1} \begin{pmatrix} 1 & 5 & 2 \\ 0 & 5 & 3 \\ 0 & -10 & -2 \end{pmatrix} \xrightarrow{F_3 \to F_3 + 2F_2} \begin{pmatrix} 1 & 5 & 2 \\ 0 & 5 & 3 \\ 0 & 0 & 4 \end{pmatrix}$$

$$L_{\text{inicial}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \xrightarrow{\text{Factor1}} \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \xrightarrow{\text{Factor2}} \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix} \xrightarrow{\text{Factor3}} \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 3 & -2 & 1 \end{pmatrix}$$

$$U = \begin{pmatrix} 1 & 5 & 2 \\ 0 & 5 & 3 \\ 0 & 0 & 4 \end{pmatrix}, L = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 3 & -2 & 1 \end{pmatrix}, y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 3 & -2 & 1 \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 8 \\ 15 \end{pmatrix}$$

$$\begin{pmatrix} y_1 \\ -y_1 + y_2 \\ 3y_1 - 2y_2 + y_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 8 \\ 15 \end{pmatrix}$$

Haciendo uso de la sustitución progresiva:

$$y = \begin{pmatrix} 5 \\ 13 \\ 4 \end{pmatrix}$$

Continuamos resolviendo $\mathrm{U}x = y$:

$$\begin{pmatrix} 1 & 5 & 2 \\ 0 & 5 & 3 \\ 0 & 0 & 4 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 13 \\ 4 \end{pmatrix}$$

$$\begin{pmatrix} x_1 + 5x_2 + 2x_3 \\ 5x_2 + 3x_3 \\ 4x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 13 \\ 4 \end{pmatrix}$$

Con el uso de la sustitución regresiva llegamos a la solución final del problema:

$$x = \begin{pmatrix} -7 \\ 2 \\ 1 \end{pmatrix}$$

El siguiente algoritmo ilustra la implementación de la descomposición de Doolittle en el lenguaje Python.

Algoritmo 4

```
def doolittle(A, b):
    """
    Implementación de la Factorización de Doolittle.

    * Toma:
    A --> matriz de coeficientes
    b --> matriz de términos independientes

    * Devuelve:
    L --> triangular inferior de la factorización
    U --> triangular superior de la factorización
    b --> términos independientes (no los modifica)
    x --> solución final LUx=b
    y --> solución parcial Ly=b, Ux=y
    """

    A = A.astype(float)
    b = b.astype(float)

    n = A.shape[0] # n es el número de filas de A (ecuaciones)

    # El algoritmo para Doolittle es el mismo que en la
    # eliminación gaussiana, solo que almacenamos los multiplicadores
    # en la parte inferior de la diagonal

    # Crea una matriz identidad nxn donde almacenaremos los multiplicadores
    L = np.eye(n,n)
```

```
    # ELIMINACIÓN GAUSIANA Ax=b --> Ux=c
    # Para las entradas que esten:
    # * En todas las columnas menos la última
    for j in range(0,n-1):
        # * En cada fila con índice superior al de la columna
        for i in range(j+1,n):
            # Si la entrada no es nula
            if A[i,j] != 0.0:
                # Multiplicador, A[j,j] es el pivote, A[i, j] el término
                mult = A[i,j]/A[j,j]
                # A toda la fila de la entrada le restamos la fila superior
                por el multiplicador
                A[i,:] = A[i,:] - mult*A[j,:]
                # _NO_ modificamos b
                # Almacenamos cada multiplicador en la entrada que elimina
                pero en L
                L[i,j] = mult


    # U es A después de escalonar
    U = A

    # Ly=b
    y = sustitucion_progresiva(L, b)

    # Ux=y
    x = sustitucion_regresiva(U, y)

    return L, U, b, x, y
```

2.5 Descomposición de Choleski

Las formas genéricas de las matrices L y U son:

$$L = \begin{pmatrix} L_{11} & 0 & 0 & \dots & 0 \\ L_{21} & L_{22} & 0 & \dots & 0 \\ L_{31} & L_{32} & L_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ L_{n1} & L_{n2} & L_{n3} & \dots & L_{nn} \end{pmatrix} \qquad U = L^T = \begin{pmatrix} L_{11} & L_{21} & L_{31} & \dots & L_{n1} \\ 0 & L_{22} & L_{32} & \dots & L_{n2} \\ 0 & 0 & L_{33} & \dots & L_{n3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & L_{\mathrm{nn}} \end{pmatrix}$$

En este caso, tomando una matriz 3×3 como ejemplo, las matrices L y U son de la forma:

$$L = \begin{pmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{pmatrix}; U = L^T = \begin{pmatrix} L_{11} & L_{21} & L_{31} \\ 0 & L_{22} & L_{32} \\ 0 & 0 & L_{33} \end{pmatrix}$$

Y aplicando $A = LU$:

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} = \begin{pmatrix} L_{11}^2 & L_{11}L_{21} & L_{11}L_{31} \\ L_{11}L_{21} & L_{21}^2 + L_{22}^2 & L_{21}L_{31} + L_{22}L_{32} \\ L_{11}L_{31} & L_{21}L_{31} + L_{22}L_{32} & L_{31}^2 + L_{32}^2 + L_{33}^2 \end{pmatrix}$$

Igualando elemento a elemento ambas matrices obtendremos un sistema de seis ecuaciones con seis incógnitas, ya que tomaremos los elementos de la matriz triangular inferior de cada una de ellas (es indiferente tomar la inferior o la superior) y con ello bastará porque se trata de matrices simétricas:

$$\begin{aligned} A_{11} &= L_{11}^2 & L_{11} &= \sqrt{A_{11}} \\ A_{21} &= L_{11}L_{21} & L_{21} &= A_{21}/L_{11} \\ A_{31} &= L_{11}L_{31} & L_{31} &= A_{31}/L_{11} \\ A_{22} &= L_{21}^2 + L_{22}^2 & L_{22} &= \sqrt{A_{22} - L_{21}^2} \\ A_{32} &= L_{21}L_{31} + L_{22}L_{32} & L_{32} &= (A_{32} - L_{21}L_{31})/L_{22} \\ A_{33} &= L_{31}^2 + L_{32}^2 + L_{33}^2 & L_{33} &= \sqrt{A_{33} - L_{31}^2 - L_{32}^2} \end{aligned}$$

Extrapolando a una matriz cuadrada de orden n(para obtener un elemento cualquiera de la matriz triangular inferior):

$$(\mathrm{LL}^T)_{ij} = A_{ij} = L_{i1}L_{j1} + L_{i2}L_{j2} + ... + L_{ij}L_{jj} = \sum_{k=1}^{j} L_{ik}L_{jk}; i \geqslant j; j = 1, 2, ..., n; i = j, j+1, ..., n$$

Para los términos de la primera columna:

$$\begin{aligned} L_{11} &= \sqrt{A_{11}} \\ L_{i1} &= A_{i1}/L_{11} \quad i = 2, 3, ..., n \end{aligned}$$

Para términos de otra columna, tomaremos la fórmula general y escribiremos en ella su último término, excluyéndolo del sumatorio:

$$A_{ij} = \sum_{k=1}^{j-1} L_{ik}L_{jk} + L_{ij}L_{jj}$$

Si buscamos un término de la diagonal, $i = j$, y por consiguiente:

$$L_{jj} = \sqrt{A_{jj} - \sum_{k=1}^{j-1} L_{jk}^2}; \quad j = 2, 3, ..., n$$

Si el término no pertenece a la diagonal:

$$\begin{aligned} L_{ij} = \left(A_{ij} - \sum_{k=1}^{j-1} L_{ik}L_{jk} \right) / L_{jj}; \quad & i = j+1, j+2, ..., n \\ & j = 2, 3, ..., n-1 \end{aligned}$$

Este $n-1$ se debe a que en la última columna el único elemento pertenece a la diagonal.

A pesar de ahorrarnos la fase de eliminación gaussiana con respecto a otros métodos de factorización, Choleski es un método que no se usa habitualmente debido a sus dos limitaciones:

1. Se puede aplicar solo y exclusivamente a matrices simétricas, ya que LL^Tes simétrica siempre.
2. El método conlleva recurrir a raíces cuadradas y se evita utilizar Choleski cuando hay que hacer raíces de números negativos. Para que no haya que recurrir a las raíces, A debe ser definida positiva.

Ejemplo 11. Descomponer la matriz A por Choleski:

$$A=\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{pmatrix}$$

Tomamos la matriz vista anteriormente:

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{pmatrix}=\begin{pmatrix} L_{11}^2 & L_{11}L_{21} & L_{11}L_{31} \\ L_{11}L_{21} & L_{21}^2+L_{22}^2 & L_{21}L_{31}+L_{22}L_{32} \\ L_{11}L_{31} & L_{21}L_{31}+L_{22}L_{32} & L_{31}^2+L_{32}^2+L_{33}^2 \end{pmatrix}$$

De aquí, igualando término a término sacamos las ecuaciones:

$$\begin{array}{ll} 1=L_{11}^2 & L_{11}=\sqrt{1}=1 \\ 1=L_{11}L_{21} & L_{21}=1/L_{11}=1/1=1 \\ 1=L_{11}L_{31} & L_{31}=1/L_{11}=1 \\ 2=L_{21}^2+L_{22}^2 & L_{22}=\sqrt{2-L_{21}^2}=\sqrt{2-1}=1 \\ 2=L_{21}L_{31}+L_{22}L_{32} & L_{32}=(2-L_{21}L_{31})/L_{22}=(2-1)/1=1 \\ 3=L_{31}^2+L_{32}^2+L_{33}^2 & L_{33}=\sqrt{3-L_{31}^2-L_{32}^2}=\sqrt{3-1-1}=1 \end{array}$$

Por tanto, tenemos ya la matriz L y L^T en las que se descompone A según $A=L\cdot L^T$:

$$L=\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}; L^T=\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

El siguiente algoritmo ilustra la implementación de la descomposición de Choleski en el lenguaje de programación Python.

Algoritmo 5

```
def choleski(A):
    """
    La matriz debe ser:
        * Simétrica
        * Definida positiva (para evitar números complejos)
    * Toma:
    A --> matriz de coeficientes

    * Devuelve:
    L --> triangular inferior de la factorización
    """


    A = A.astype(float)

    n = A.shape[0] # n es el número de filas de A (ecuaciones)

   # Aprovechamos A en el algoritmo en vez de crear L ya que los terminos no
se vuelven a       emplear

    # Para cada columna
    for j in range(n):
        # Si la raiz es un número complejo, lanzamos un error
        try:
            # En los elementos de la diagonal
            A[j,j] = np.sqrt(A[j,j] - np.dot(A[j,0:j],A[j,0:j]))
        except ValueError:
            error.err("La matriz no es definida positiva")
        # En los elementos bajo la diagonal
        for i in range(j+1,n):
            A[i,j] = (A[i,j] - np.dot(A[i,0:j],A[j,0:j]))/A[j,j]
    # En los términos sobre la diagonal
    for j in range(1,n):
        A[0:j,j] = 0.0


    return A
```

2.6 Matrices dispersas

Ciertas matrices simplifican los cálculos que tenemos que realizar para resolver sus sistemas asociados. Éstas reciben el nombre de matrices dispersas, en ellas el mayor número de sus elementos son ceros. El gran número de ceros permite reducir el espacio necesario para almacenar el sistema en el computador y de esta manera minimizar la memoria utilizada y el tiempo de proceso. Cuando el ancho de banda es razonablemente pequeño podemos hablar de matrices en banda:

$$\begin{pmatrix} X & X & X & \circ & \circ & \circ & \circ \\ X & X & \circ & X & X & \circ & \circ \\ X & \circ & X & \circ & X & \circ & \circ \\ \circ & X & \circ & X & \circ & X & \circ \\ \circ & X & X & \circ & X & X & X \\ \circ & \circ & \circ & X & X & X & \circ \\ \circ & \circ & \circ & \circ & X & \circ & X \end{pmatrix}$$

Existen diversos casos, entre los que distinguimos:

- **Diagonales**: con elementos no nulos en las diagonales principales de la matriz (tridiagonales, pentadiagonales...):

$$\begin{pmatrix} X & X & \circ & \circ & \circ & \circ & \circ \\ X & X & X & \circ & \circ & \circ & \circ \\ \circ & X & X & X & \circ & \circ & \circ \\ \circ & \circ & X & X & X & \circ & \circ \\ \circ & \circ & \circ & X & X & X & \circ \\ \circ & \circ & \circ & \circ & X & X & X \\ \circ & \circ & \circ & \circ & \circ & X & X \end{pmatrix}$$

- **Simétricas**: el cómputo de éstas es más sencillo, pues los elementos son iguales a un lado y al otro de la diagonal.

Para evitar almacenar todos estos ceros y aumentar la eficiencia del computo, existen métodos específicos para cada uno de estos casos que simplifican las operaciones. Lo explicaremos con el caso de las matrices tridiagonales. Estas son, como hemos dicho, matrices dispersas muy comunes en la discretización de ecuaciones diferenciales.

Si la factorización LU correspondiente existe, los factores L y U deben de ser matrices bidiagonales de la sigueinte forma:

$$\begin{pmatrix} X & X & \circ & \circ & \circ & \circ & \circ \\ X & X & X & \circ & \circ & \circ & \circ \\ \circ & X & X & X & \circ & \circ & \circ \\ \circ & \circ & X & X & X & \circ & \circ \\ \circ & \circ & \circ & X & X & X & \circ \\ \circ & \circ & \circ & \circ & X & X & X \\ \circ & \circ & \circ & \circ & \circ & X & X \end{pmatrix} = \begin{pmatrix} 1 & \circ & \circ & \circ & \circ & \circ & \circ \\ X & 1 & \circ & \circ & \circ & \circ & \circ \\ \circ & X & 1 & \circ & \circ & \circ & \circ \\ \circ & \circ & X & 1 & \circ & \circ & \circ \\ \circ & \circ & \circ & X & 1 & \circ & \circ \\ \circ & \circ & \circ & \circ & X & 1 & \circ \\ \circ & \circ & \circ & \circ & \circ & X & 1 \end{pmatrix} \cdot \begin{pmatrix} X & X & \circ & \circ & \circ & \circ & \circ \\ \circ & X & X & \circ & \circ & \circ & \circ \\ \circ & \circ & X & X & \circ & \circ & \circ \\ \circ & \circ & \circ & X & X & \circ & \circ \\ \circ & \circ & \circ & \circ & X & X & \circ \\ \circ & \circ & \circ & \circ & \circ & X & X \\ \circ & \circ & \circ & \circ & \circ & \circ & X \end{pmatrix}$$

$$A = L \cdot U$$

Basádonos en la descomposición de Doolittle encontramos que para una matriz A podemos encontrar dos matrices de esa forma:

$$A = \begin{pmatrix} d_1 & e_1 & \circ & \circ & \cdots & \circ \\ c_1 & d_2 & e_2 & \circ & \cdots & \circ \\ \circ & c_2 & d_3 & e_3 & \cdots & \circ \\ \circ & \circ & c_3 & d_4 & \cdots & \circ \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \circ & \circ & \circ & \circ & c_{n-1} & d_n \end{pmatrix}$$

$$A = \begin{pmatrix} 1 & 4 & 0 & 0 \\ 3 & 4 & 1 & 0 \\ 0 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

Almacenamos los coeficientes en tan solo 3 vectores:

$$c=\begin{pmatrix} 0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}, d=\begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix}, e=\begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_{n-1} \\ 0 \end{pmatrix}$$

$$c=\begin{pmatrix} 0 \\ 3 \\ 2 \\ 1 \end{pmatrix}, d=\begin{pmatrix} 1 \\ 4 \\ 3 \\ 3 \end{pmatrix}, e=\begin{pmatrix} 4 \\ 1 \\ 4 \\ 0 \end{pmatrix}$$

De este modo, para un sistema 100×100, que contiene 10.000 elementos, encontramos que podemos almacenarlo en $99+100+99=298$ entradas.

El siguiente paso es eliminar las diagonales c y d aplicando las siguientes operaciones elementales en cada fila k:

$$F_k' \to F_k - \left(\frac{c_{k-1}}{d_{k-1}}\right) \cdot F_{k-1}\,; \quad k=2,3,...,n$$

$$\begin{pmatrix} 1 & 4 & 0 & 0 \\ 3 & 4 & 1 & 0 \\ 0 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 \end{pmatrix} \xrightarrow{F_2 \to F_2 - \left(\frac{c_1}{d_1}\right)\cdot F_1} \begin{pmatrix} 1 & 4 & 0 & 0 \\ 0 & -8 & 1 & 0 \\ 0 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 \end{pmatrix} \xrightarrow{F_3 \to F_3 - \left(\frac{c_2}{d_2}\right)\cdot F_2}$$

$$\begin{pmatrix} 1 & 4 & 0 & 0 \\ 0 & -8 & 1 & 0 \\ 0 & 0 & 3.25 & 4 \\ 0 & 0 & 1 & 3 \end{pmatrix} \xrightarrow{F_4 \to F_4 - \left(\frac{c_3}{d_3}\right)\cdot F_3} \begin{pmatrix} 1 & 4 & 0 & 0 \\ 0 & -8 & 1 & 0 \\ 0 & 0 & 3.25 & 4 \\ 0 & 0 & 0 & \frac{23}{13} \end{pmatrix}$$

$$U=\begin{pmatrix} 1 & 4 & 0 & 0 \\ 0 & -8 & 1 & 0 \\ 0 & 0 & 3.25 & 4 \\ 0 & 0 & 0 & \frac{23}{13} \end{pmatrix}$$

De esta manera los coeficientes e_k no se ven afectados por la transformación. Para terminar almacenamos en una matriz identidad los multiplicadores $\lambda = \frac{c_{k-1}}{d_{k-1}}$ en la entrada antes ocupada por c_{k-1} :

$$L=\begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{3}{1} & 1 & 0 & 0 \\ 0 & \frac{2}{-8} & 1 & 0 \\ 0 & 0 & \frac{1}{3.25} & 1 \end{pmatrix}$$

Así, acabamos de obtener la factorización que buscábamos:

$$\begin{pmatrix} 1 & 4 & 0 & 0 \\ 3 & 4 & 1 & 0 \\ 0 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{3}{1} & 1 & 0 & 0 \\ 0 & \frac{2}{-8} & 1 & 0 \\ 0 & 0 & \frac{1}{3.25} & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 4 & 0 & 0 \\ 0 & -8 & 1 & 0 \\ 0 & 0 & 3.25 & 4 \\ 0 & 0 & 0 & \frac{23}{13} \end{pmatrix}$$

En la fase de solución resolvemos primeramente el sistema $\mathrm{L}y = b$ que expresamos de la siguiente manera con su matriz aumentada:

$$[L|b] = \left(\begin{array}{cccccc|c} 1 & 0 & 0 & 0 & \cdots & 0 & b_1 \\ c_1 & 1 & 0 & 0 & \cdots & 0 & b_2 \\ 0 & c_2 & 1 & 0 & \cdots & 0 & b_3 \\ 0 & 0 & c_3 & 1 & \cdots & 0 & b_4 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & b_5 \\ 0 & 0 & 0 & 0 & c_{n-1} & 1 & b_6 \end{array}\right)$$

$$[L|b] = \left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 & 2 \\ 0 & -\frac{1}{4} & 1 & 0 & 3 \\ 0 & 0 & \frac{1}{3.25} & 1 & 4 \end{array}\right)$$

Debemos tener en cuenta que c ya no contiene los coeficientes originales, sino los multiplicadores usados en la descomposición. Resolvemos y mediante sustitución progresiva:

$$\begin{aligned} y_1 &= b_1 \\ y_2 &= b_2 - y_1 c_1 \\ &\vdots \\ y_n &= b_n - y_{n-1} c_{n-1} \end{aligned}$$

$$\begin{aligned} y_1 &= 1 \\ y_2 &= 2 - 3 \cdot 1 = -1 \\ y_3 &= 3 + \frac{1}{4} \cdot (-1) = \frac{11}{4} \\ y_4 &= 4 - \frac{1}{3.25} \cdot \frac{11}{4} = \frac{41}{13} \end{aligned}$$

Una vez que tenemos y continuamos resolviendo el sistema Ux $= y$ en este caso mediante sustitución regresiva:

$$[U\,|\,y] = \left(\begin{array}{ccccc|c} d_1 & e_1 & 0 & \cdots & 0 & y_1 \\ 0 & d_2 & e_2 & \cdots & 0 & y_2 \\ 0 & 0 & d_3 & \cdots & 0 & y_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & ... & d_n & y_n \end{array}\right)$$

$$[U\,|\,y] = \left(\begin{array}{cccc|c} 1 & 4 & 0 & 0 & 1 \\ 0 & -8 & 1 & 0 & -1 \\ 0 & 0 & 3.25 & 4 & \frac{11}{4} \\ 0 & 0 & 0 & \frac{23}{13} & \frac{41}{13} \end{array}\right)$$

$$\begin{aligned} x_n &= y_n \\ x_{n-1} &= \frac{y_{n-1} - e_{n-1} y_n}{d_{n-1}} \\ &\vdots \\ x_1 &= \frac{y_1 - e_1 y_2}{d_2} \end{aligned}$$

$$\begin{aligned} x_4 &= \frac{41}{13} \cdot \frac{13}{23} = \frac{41}{23} \\ x_3 &= \left(\frac{11}{4} - 4 \cdot \frac{41}{23}\right) \cdot \frac{1}{3.25} = -\frac{31}{23} \\ x_2 &= \left(-1 + 1 \cdot \frac{31}{23}\right) \cdot \frac{-1}{8} = -\frac{1}{23} \\ x_1 &= \left(1 + 4 \cdot \frac{1}{23}\right) \cdot \frac{1}{1} = \frac{27}{23} \end{aligned}$$

En los casos de matrices tridiagonales la complejidad computacional para las sustituciones regresivas y progresivas es de $O(n)$ mientras que en las matrices usuales lo es de $O(n^2)$. Pero, ¿qué quiere decir esto? Como podemos observar, las sustituciones en el ejemplo anterior son lineales, su complejidad no aumenta por mayor que sea el número de incógnitas, algo que sí ocurrió en el primer ejemplo con la eliminación gaussiana.

El siguiente algoritmo muestra la implementación en Python de una rutina del algoritmo de Thomas para la descomposición de matrices tridiagonales.

Algoritmo 6

```
def tridiagonal(c, d, e, b):
    """
    * Toma:
    c,d, e --> vector con la subdiagonal, diagonal y superdiagonal
    b --> términos independientes

    * Devuelve:
    L --> triangular inferior de la factorización
    U --> triangular superior de la factorización
    b --> términos independientes (no los modifica)
    x --> solución final LUx=b
    y --> solución parcial Ly=b, Ux=y
    """

    # Los coeficientes de los vectores pasan a expresarse como float
    c, d, e = c.astype(float), d.astype(float), e.astype(float
    b = b.astype(float)

    n = d.shape[0] # n es el número de filas de A (número de ecuaciones)

    # FACTORIZACIÓN
    # En esta fase construimos U, que contiene d y e
    # Así como L, compuesta por la identidad más c

    # En cada fila desde la segunda
    for i in range(1,n):
        mult = c[i-1]/d[i-1]
        d[i] = d[i] - mult*e[i-1]
        # Almacenamos los multiplicadores en c
        c[i-1] = mult

    # RESOLUCIÓN
    # Por comodidad construiremos L y U de manera completa para la resolución
    # Esto no es un gran inconveniente puesto que el grueso de las operaciones
    # ya las hemos realizado usando solo las diagonales

    # L

    L = np.eye(n,n)

    # Para los elementos de la diagonal inferior (i, j) = (1,0), (2,1), (3,2)...
    for i in range(1, n):
            # Copia el elemento i de c: c[1], c[2]...  nunca el 0 del principio
            L[i,i-1] = c[i]

    # U

    U = np.eye(n,n)

    for i in range(n-1):
        # Para los elementos de la diagoanl copia d (todos)
        U[i,i] = d[i]
        # Por encima de la diagonal copia todos los de e menos el último (0)
        if (i!=n):
            U[i,i+1] = e[i]

    # Ly=b

    y = sustitucion_progresiva(L, b)

    # Ux=y

    x = sustitucion_regresiva(U, y)


    return L, U, b, x, y
```

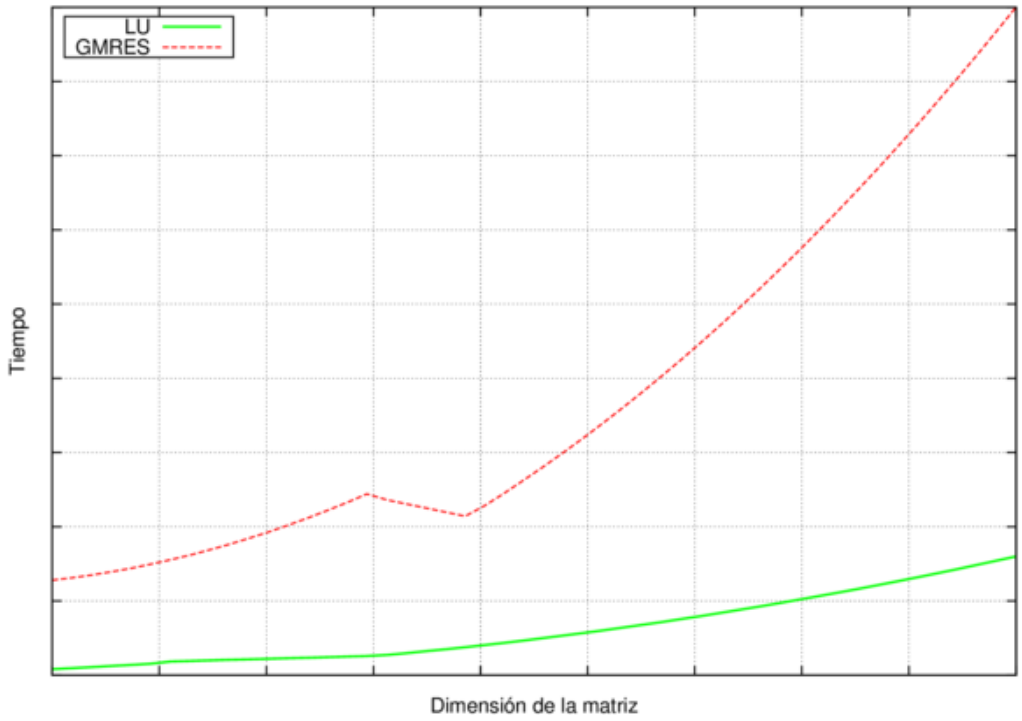
3 ¿Directo o iterado?

Ahora que conocemos gran variedad de métodos (tanto iterativos como directos) para resolver sistemas de ecuaciones lineales, se nos plantea la siguiente duda: ¿qué método escoger? Para matrices pequeñas, no hay grandes diferencias entre métodos (hablando de tiempo de procesamiento y volumen de operaciones). Pero al trabajar con sistemas grandes hay notables diferencias entre usar un método u otro. En estos casos, la elección de un método determinado dependerá de las propiedades de la matriz (simetría, número de condición...), además de los recursos computacionales disponibles.

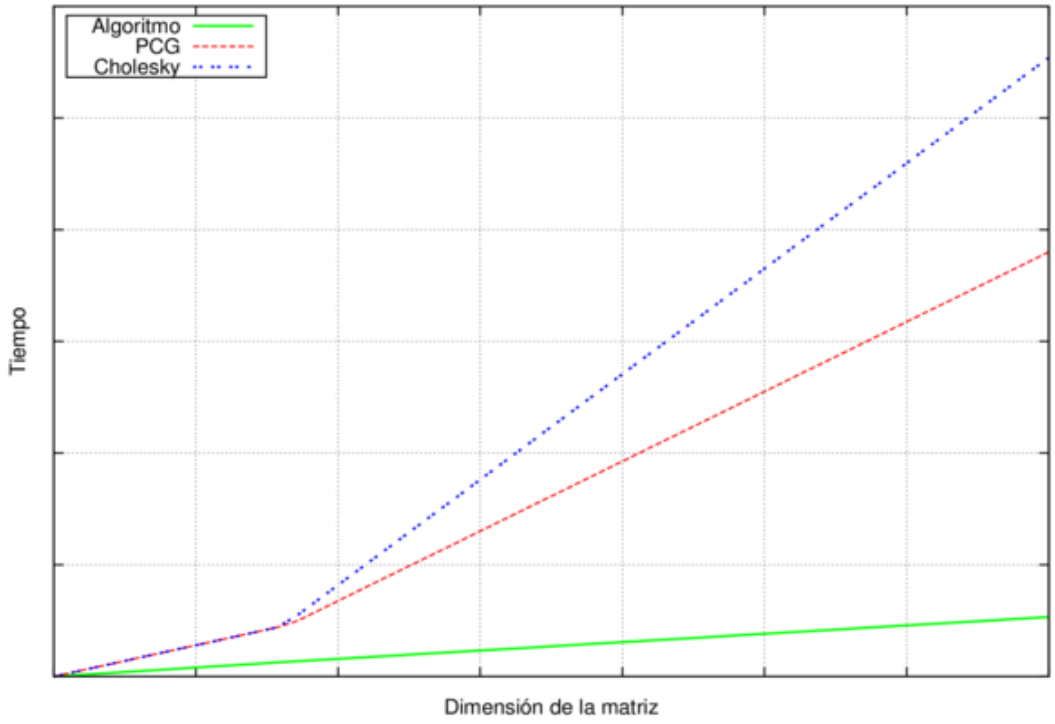
Muchos programas informáticos (como es el conocido Matlab con su o Octave con el comando) analizan la matriz A y, dependiendo de las características antes mencionadas, escogen el método idóneo para resolver ese sistema.

Vamos a analizar los costes computacionales de métodos iterados y métodos directos al resolver distintos tipos de sistemas lineales:

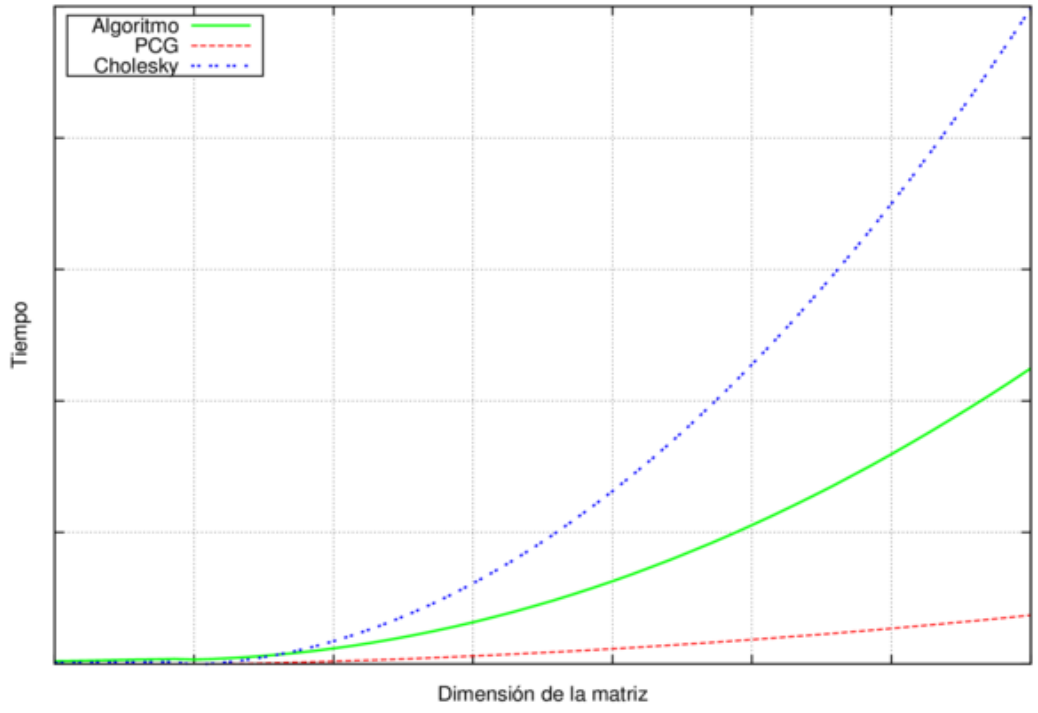
- **Matrices llenas**: Trabajando con un sistema no simétrico y cuya matriz A está llena, el mejor método para resolverlo es aplicar una de las descomposiciones descritas anteriormente, ya que si lo comparamos con el método iterativo GMRes (Residuos Mínimos Generalizados) sin preacondicionamiento, el tiempo de cáculo del método directo es menor. Por ello, antes de aplicar con éxito un método iterativo hay que llevar a cabo un preacondicionamiento.



- **Matriz de pequeña anchura de banda**: Si A es una matriz de pequeña anchura de banda, los mejores métodos a aplicar son los algoritmos de resolución de matrices n-diagonales (tridiagonales o pentadiagonales por ejemplo) implementados en los programas que resultan de combinar métodos iterados y directos. Pero distinguiendo entre estos, es más eficaz el método iterativo tal y como se puede observar en las gráficas (sólo si se aplica un preacondicionamiento al sistema). Como método directo hemos elegido Cholesky y como método iterativo, el método del gradiente conjugado (PCG) con preacondicionamiento ya que está especialmente diseñado para matrices simétricas.



- **Matriz de banda ancha**: Siendo A una matriz de banda ancha, los resultados son muy parecidos, a pesar de que dicha matriz tendrá muchos más elementos no nulos (a pesar de que sigue siendo parcialmente hueca). El mejor método para resolver el sistema es en estos casos el método iterativo. De nuevo hemos elegido los mismos métodos para factorizar la matriz: el algoritmo especial para matrices simétricas (combinación de métodos directos e iterativos), el método de Cholesky y el método del gradiente conjugado (utilizando como preacondicionador una factorización incompleta de Cholesky).



4 BIBLIOGRAFIA

BURDEN, R; FAIRES, J; 2010. Numerical Analysis; Cengage Learning

MOIN, P; 2010. Fundamentals of engineering - Numerical Analysis; Cambridge

QUARTERONI, A; SALERI, F; 2006. Calculo matematico con MATLAB y Octave; Springer

FANGOHR, H; 2014. Python for computational science and engineering; Southampton