[redacted]

GRUPA: 341

= Temă Nr 2 =

1) a) Începem cu un caz particular: $n=5$

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ l_1 & 1 & 0 & 0 & 0 \\ 0 & l_2 & 1 & 0 & 0 \\ 0 & 0 & l_3 & 1 & 0 \\ 0 & 0 & 0 & l_4 & 1 \end{pmatrix} \qquad R = \begin{pmatrix} r_1 & s_1 & 0 & 0 & 0 \\ 0 & r_2 & s_2 & 0 & 0 \\ 0 & 0 & r_3 & s_3 & 0 \\ 0 & 0 & 0 & r_4 & s_4 \\ 0 & 0 & 0 & 0 & r_5 \end{pmatrix}$$

Din enunț, "A" este de tipul:

$$A = \begin{pmatrix} a_1 & b_1 & 0 & 0 & 0 \\ c_1 & a_2 & b_2 & 0 & 0 \\ 0 & c_2 & a_3 & b_3 & 0 \\ 0 & 0 & c_3 & a_4 & b_4 \\ 0 & 0 & 0 & c_4 & a_5 \end{pmatrix}$$

(Cazul general reiese în mod similar.)

De asemenea, $A = L \cdot R \Rightarrow$

$$A = \begin{pmatrix} r_1 & s_1 & 0 & 0 & 0 \\ l_1 \cdot r_1 & l_1 \cdot s_1 + r_2 & s_2 & 0 & 0 \\ 0 & l_2 \cdot r_2 & l_2 \cdot s_2 + r_3 & s_3 & 0 \\ 0 & 0 & l_3 \cdot r_3 & l_3 \cdot s_3 + r_4 & s_4 \\ 0 & 0 & 0 & l_4 \cdot r_4 & l_4 \cdot s_4 + r_5 \end{pmatrix}$$

De aici, putem deduce:

- $b_i = s_i \quad ; \ i = \overline{1, n-1}$
- $c_i = l_i \cdot r_i \ ; \ i = \overline{1, n-1}$
- $a_1 = r_1 \ ; \ a_i = l_{i-1} \cdot s_{i-1} + r_i \ ; \ i = \overline{2, n}$

Schema numerică oferită:

$r_1 = a_1$ (dem. mai sus ✓)

for $i = 1 : (n-1)$ do

  $l_i = c_i / r_i$ ($\Leftrightarrow c_i = l_i \cdot r_i$ ✓)

  $s_i = b_i$ (dem. mai sus ✓)

  $r_{i+1} = a_{i+1} - l_i \cdot s_i$ ($\Leftrightarrow a_{i+1} = r_{i+1} + l_i \cdot s_i$ ✓)

end for

b) $\left.\begin{array}{l} Ax = b \\ A = L\cdot R \end{array}\right\} \Rightarrow L\cdot \underbrace{Rx}_{y} = b$

$L\cdot y = b$

Începem cu un caz particular: $n = 5$

$$L\cdot y = b \Leftrightarrow \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ l_1 & 1 & 0 & 0 & 0 \\ 0 & l_2 & 1 & 0 & 0 \\ 0 & 0 & l_3 & 1 & 0 \\ 0 & 0 & 0 & l_4 & 1 \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix} \Leftrightarrow \begin{pmatrix} y_1 \\ l_1\cdot y_1 + y_2 \\ l_2\cdot y_2 + y_3 \\ l_3\cdot y_3 + y_4 \\ l_4\cdot y_4 + y_5 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix} \Rightarrow$$

$$\Rightarrow \begin{cases} y_1 = b_1 \\ y_i = b_i - l_{i-1}\cdot y_{i-1}, \ i = \overline{2,n} \end{cases}$$ (pt. cazul general)

$$Rx = y \Leftrightarrow \begin{pmatrix} r_1 & s_1 & 0 & 0 & 0 \\ 0 & r_2 & s_2 & 0 & 0 \\ 0 & 0 & r_3 & s_3 & 0 \\ 0 & 0 & 0 & r_4 & s_4 \\ 0 & 0 & 0 & 0 & r_5 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} \Leftrightarrow \begin{pmatrix} r_1 x_1 + s_1 x_2 \\ r_2 x_2 + s_2 x_3 \\ r_3 x_3 + s_3 x_4 \\ r_4 x_4 + s_4 x_5 \\ r_5 x_5 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} \Rightarrow$$

$$\Rightarrow \begin{cases} x_n = y_n / r_n \\ x_i = (y_i - s_i \cdot x_{i+1}) / r_i \ ; \ i = \overline{1,n-1} \end{cases}$$ (pt. cazul general)

In [1]:
```
"""
Created on Wed Nov 17 02:05:08 2020

@author:    [redacted]
@grupa:     341
@nr.crt:    159
@varianta:  15
"""
```

Out[1]:
```
'\nCreated on Wed Nov 17 02:05:08 2020\n\n@author: [redacted] \n@grupa:     341\n@
nr.crt:    159\n@varianta:  15\n'
```

In [2]:
```
# Librării
import numpy as np
import math
```

In [3]:
```
"""     Exercițiul 2 -> Rezolvare     """

# Datele Problemei
d = 18
f = -6
c = -3
n = 20

# Reprezentăm matricea A a sistemului (tridiagonală)
A = np.zeros((n, n))
for i in range(n):
    A[i][i] = d * 1.0
for i in range(1, n):
    A[i][i - 1] = c * 1.0
    A[i - 1][i] = f * 1.0

# Vectorul b - termenii liberi
b = [1.] * n
b[0] = b[n - 1] = 2.
```

In [4]:
```
"""
Conform Exercițiului 1, putem reprezenta matricea A = L * R, deci calculăm
elementele matricilor L si R conform algoritmului prezentat în subpunctul a
al Exercițiului 1.
"""

# Inițial, avem toate elementele 0
L = np.zeros((n, n))
R = np.zeros((n, n))

# Diagonala principală a lui L are numai valori = 1
for i in range(n):
    L[i][i] = 1.0

# Algoritmul de la Ex1.a)
R[0][0] = A[0][0]
for i in range(n - 1):
    L[i + 1][i] = A[i + 1][i] / R[i][i]
    R[i][i + 1] = A[i][i + 1]
    R[i + 1][i + 1] = A[i + 1][i + 1] - L[i + 1][i] * R[i][i + 1]


# Verificare A = L * R
print(f"A: \n{A}")
print(f"L * R: \n{L@R}")
```

```
A:
[[18. -6.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [-3. 18. -6.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0. -3. 18. -6.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0. -3. 18. -6.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0. -3. 18. -6.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0. -3. 18. -6.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0. -3. 18. -6.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0. -3. 18. -6.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0. -3. 18. -6.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0. -3. 18. -6.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0. -3. 18. -6.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -3. 18. -6.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -3. 18. -6.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -3. 18. -6.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -3. 18. -6.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -3. 18. -6.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -3. 18. -6.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -3. 18.
  -6.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -3.
  18. -6.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
  -3. 18.]]
L * R:
[[18. -6.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [-3. 18. -6.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0. -3. 18. -6.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0. -3. 18. -6.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0. -3. 18. -6.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0. -3. 18. -6.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0. -3. 18. -6.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0. -3. 18. -6.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0. -3. 18. -6.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0. -3. 18. -6.  0.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0. -3. 18. -6.  0.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -3. 18. -6.  0.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -3. 18. -6.  0.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -3. 18. -6.  0.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -3. 18. -6.  0.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -3. 18. -6.  0.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -3. 18. -6.
   0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -3. 18.
  -6.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -3.
  18. -6.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
  -3. 18.]]
```

In [5]:
```
"""
Aplic metoda de factorizare LR, după cum am prezentat-o în Ex1.b)
"""

# Calculez y
y = np.zeros(n)
y[0] = b[0]
for i in range(1, n):
    y[i] = b[i] - L[i][i - 1] * y[i - 1]


# Verificare: L * y = b
print(f"b: \n{b}\n")
print(f"L * y: \n{L@y}")
```

```
b:
[2.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 2.0]

L * y:
[2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2.]
```

In [6]:
```
# Calculez x (-> Soluție pt A * x = b)
x = np.zeros(n)
x[n - 1] = y[n - 1] / R[n - 1][n - 1]
for i in range(n - 2, -1, -1):
    x[i] = (y[i] - R[i][i + 1] * x[i + 1]) / R[i][i]

# Verificare: R * x = y
print(f"y: \n{y}\n")
print(f"R * x: \n{R@x}")
```

```
y:
[2.         1.33333333 1.23529412 1.21875    1.21586716 1.21535948
 1.21526974 1.21525385 1.21525104 1.21525054 1.21525046 1.21525044
 1.21525044 1.21525044 1.21525044 1.21525044 1.21525044 1.21525044
 1.21525044 2.21525044]

R * x:
[2.         1.33333333 1.23529412 1.21875    1.21586716 1.21535948
 1.21526974 1.21525385 1.21525104 1.21525054 1.21525046 1.21525044
 1.21525044 1.21525044 1.21525044 1.21525044 1.21525044 1.21525044
 1.21525044 2.21525044]
```

In [7]:
```
# Verificare: A * x = b
print(f"b: \n{b}\n")
print(f"A * x: \n{A@x}")
```

```
b:
[2.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 2.0]

A * x:
[2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2.]
```

In [8]:
```
print(f"Soluția Sistemului Tridiagonal A * x = b este x: \n{x}")
```

```
Soluția Sistemului Tridiagonal A * x = b este x:
[0.15047208 0.1180829  0.11234598 0.11132984 0.11114986 0.11111798
 0.11111235 0.1111114  0.11111137 0.11111173 0.11111284 0.11111599
 0.11112489 0.11115001 0.1112209  0.11142104 0.11198602 0.11358086
 0.1180829  0.13079159]
```

In [9]:
```
"""     Exercițiul 3 -> Rezolvare     """

# Datele Problemei
A = np.array([[12, 9, 17],
              [4, 2, 5],
              [20, 22, 38]])
b = np.array([[31], [12], [50]])
```

In [10]:
```
# Metoda Substituției Descendente (Funcție Auxiliară pt implementarea Metodei Gauss)
def metSubDesc(A, b, tol):
    """

    Parameters
    ----------
    A : matrice pătratică, superior triunghiulară, cu toate elementele de pe diagonala principală nenul
e.
    b : vectorul termenilor liberi.
    tol : toleranță = valoare numerică foarte mică în raport cu care vom compara numerele apropiate de
 0.

    Returns
    -------
    x = Soluția Sistemului.

    """

    # Verificăm dacă matricea este pătratică
    m, n = np.shape(A)
    if m!= n:
        print("Matricea nu este pătratică. Introduceți altă matrice.")
        x = None
        return x

    # Verificăm dacă matricea este superior triunghiulară
    for i in range(m):
       for j in range(i):
           if abs(A[i][j] > tol):
               print("Matricea nu este superior triunghiulară.")
               x = None
               return x

    # Verificam dacă toate elementele de pe diagonala principală sunt nenule => Si. este compatibil det
erminat
    # (adică am soluție unică)
    for i in range(n):
        if A[i][i] == 0:
            print("Sistemul nu este compatibil determinat.")
            x = None
            return x

    x = np.zeros((n, 1))
    x[n - 1] = b[n - 1] / A[n - 1][n - 1]

    k = n - 2
    while k >= 0:
        s = 0
        for j in range(k + 1, n):
            s += x[j] * A[k][j]

        x[k] = (1 / A[k][k]) * (b[k] - s)
        k -= 1

    return x
```

In [11]:
```
# Definim Procedura InvDet(A)
def InvDet(A):
    """

    Parameters
    ----------
    A : matricea.

    Returns
    -------
    det = determinantul lui A;
    A_invers = inversul lui A

    """

    # Nu mai are sens să verific dacă matricea este pătratică, pt ca lucrăm pe un caz particular
    # Iau dimensiunea matricei A
    n = len(A)

    # Matricea Identitate
    I = np.zeros((n, n))
    for i in range(n):
        I[i][i] = 1

    # Matrice Extinsă (Concatenez la A matricea I)
    A_extins = np.concatenate((A, I), axis = 1)


    # pt soluții
    index = np.array(range(n))


    pivot = np.zeros(n) # unde rețin pivoții pentru calcul determinant
    s = 0               # unde rețin nr permutări tot pt determinant


    for k in range(n - 1):
        p = k
        m = k

        for i in range(k, n):
            for j in range(k, n):
                if abs(A_extins[p][m]) < abs(A_extins[i][j]):
                    p = i
                    m = j

        if A_extins[p][m] == 0:
            print("Sistem Incompatibil sau Compatibil Nedeterminat!")
            break

        pivot[k] = A_extins[p][m]

        if p != k:
            A_extins[[p, k]] = A_extins[[k, p]] # interschimb linia p cu linia k
            s += 1

        if m != k:
            A_extins[:, [m, k]] = A_extins[:, [k, m]] # interschimb coloana m cu coloana k
            s += 1
            index[[m, k]] = index[[k, m]]

        for l in range(k + 1, n):
            m = A_extins[l][k] / A_extins[k][k]
            A_extins[l] = A_extins[l] - m * A_extins[k]

        if A_extins[n - 1][n - 1] == 0:
            print("Sistem Incompatibil sau Compatibil Nedeterminat!")
            break

    pivot[n - 1] = A_extins[n - 1][n - 1]



    # calcul determinant = (-1) ** s * pivot1 * pivot2 * ... * pivotn
    det = (-1) ** s
    for i in range(n):
        det *= pivot[i]


    # calcul inversă
    A_invers = np.zeros((n, n))
    coloana = 0 # ca să concatenez x-urile la A_invers

    for i in range(n):
        y = metSubDesc(A_extins[:, 0:n], A_extins[:, [n + i]], 10 ** (-10))
        x = np.zeros(len(y))
        for i in range(n):
            x[index[i]] = y[i]

        for i in range(n):
            A_invers[i][coloana] = x[i]
        coloana += 1



    return det, A_invers
```

In [12]:
```
det, A_invers = InvDet(A)
print("Determinant: {:.2f}".format(det))
print(f"\nMatricea Inversă A_invers: \n{A_invers}")
```

```
Determinant: -60.00

Matricea Inversă A_invers:
[[ 0.56666667 -0.53333333 -0.18333333]
 [ 0.86666667 -1.93333333 -0.13333333]
 [-0.8         1.4         0.2       ]]
```

In [13]:
```
x = A_invers@b
print(f"Soluția Sistemului -> x: \n{x}")
```

```
Soluția Sistemului -> x:
[[ 2.]
 [-3.]
 [ 2.]]
```

In [14]:
```
"""     Exercițiul 4 -> Rezolvare     """

# Datele Problemei -> Subpunctul 1
n = 5

b = np.zeros(n)
for i in range(1, n + 1):
    b[i - 1] = i ** 4
print(f"b: {b}\n")

# Definim Vectorul a
a = np.zeros(n)
for i in range(n, -1, -1):
    a[i % n] = 2 ** (n - (i % n))
print(f"Vectorul a: {a}\n")

# Definim Matricea Simetrică A
A = np.zeros((n, n))
for i in range(n):
    for j in range(i, n):
        A[i][j] = a[j - i]
        A[j][i] = a[j - i]
print(f"Matricea Simetrică A: \n{A}")
```

```
b: [  1.  16.  81. 256. 625.]

Vectorul a: [32. 16.  8.  4.  2.]

Matricea Simetrică A:
[[32. 16.  8.  4.  2.]
 [16. 32. 16.  8.  4.]
 [ 8. 16. 32. 16.  8.]
 [ 4.  8. 16. 32. 16.]
 [ 2.  4.  8. 16. 32.]]
```

In [15]:
```
# Subpunctul 2 -> Criteriul Sylvester
def Sylvester(A):
    ok = 0
    for i in range(1, n + 1):
        aux = A[0:i, 0:i]
        if np.linalg.det(aux) <= 0:
            print("Matricea NU este Pozitiv Definită!")
            ok = 1

    return ok


if Sylvester(A) == 0:
    print("Matricea este Pozitiv Definită!")
```

```
Matricea este Pozitiv Definită!
```

In [16]:
```
# Subpunctul 3 -> Factorizarea Cholesky



# Funcție Auxiliară pt FactCholesky
def metSubAsc(A, b, tol):
    """

    Parameters
    ----------
    A : matrice inferior triunghiulară.
    b : vectorul termenilor liberi.
    tol : toleranța.

    Returns
    -------
    soluția.

    """

    # Verificăm dacă matricea este pătratică
    m, n = np.shape(A)
    if m != n:
        print("Matricea nu este pătratică. Introduceți altă matrice.")
        x = None
        return x

    # Verificăm dacă matricea este superior triunghiulară
    for i in range(m):
       for j in range(i):
           if abs(A[j][i]) > tol:
               print("Matricea nu este inferior triunghiulară.")
               x = None
               return x

    # Verificam dacă toate elementele de pe diagonala principală sunt nenule =>
    # Si. este compatibil ddeterminat (adică am soluție unică)
    for i in range(n):
        if abs(A[i][i]) <= tol:
            print("Sistemul nu este compatibil determinat.")
            x = None
            return x

    x = np.zeros((m, 1))
    x[0] = b[0] / A[0][0]

    for k in range(1, n):

        sum = 0
        for j in range(k):
            sum += A[k][j] * x[j]

        x[k] = (1 / A[k][k]) * (b[k] - sum)

    return x



# Factorizarea Cholesky
def FactCholesky(A, b):
    """

    Parameters
    ----------
    A : matrice simetrică, pozitiv definită.
    b : vectorul termenilor liberi.

    Returns
    -------
    L => A = L * L_transpus
    x => soluția sistemului Ax = b

    """

    # Pasul 1
    A_transpus = np.transpose(A)
    if not np.array_equal(A, A_transpus):
        print("Matricea NU este Simetrică!")
        return None, None

    if Sylvester(A) == 1:
        return None, None

    L = np.zeros((n, n))
    L[0][0] = math.sqrt(A[0][0])

    for i in range(1, n):
        L[i][0] = A[i][0] / L[0][0]

    # Pasul 2
    for k in range(1, n):
        sum = 0
        for s in range(k):
            sum += L[k][s] ** 2

        sum = A[k][k] - sum
        if sum <= 0:
            print("Matricea NU este Pozitiv Definită!")
            return None, None

        L[k][k] = math.sqrt(sum)

        for i in range(k + 1, n):
            sum = 0
            for s in range(k):
                sum += L[i][s] * L[k][s]

            sum = A[i][k] - sum
            L[i][k] = sum / L[k][k]

    # Pasul 3
    y = metSubAsc(L, b, 10 ** (-10))

    # Pasul 4
    x = metSubDesc(np.transpose(L), y, 10 ** (-10))


    return L, x
```

In [17]:
```
# Subpunctul 4, 5, 6

L, x = FactCholesky(A, b)
L_transpus = np.transpose(L)
print("A = L * L_transpus\n")
print(f"L = \n{L}\n")
print(f"L_transpus = \n{L_transpus}\n")
print(f"L * L_transpus = \n {L@L_transpus}")
```

```
A = L * L_transpus

L =
[[5.65685425 0.         0.         0.         0.        ]
 [2.82842712 4.89897949 0.         0.         0.        ]
 [1.41421356 2.44948974 4.89897949 0.         0.        ]
 [0.70710678 1.22474487 2.44948974 4.89897949 0.        ]
 [0.35355339 0.61237244 1.22474487 2.44948974 4.89897949]]

L_transpus =
[[5.65685425 2.82842712 1.41421356 0.70710678 0.35355339]
 [0.         4.89897949 2.44948974 1.22474487 0.61237244]
 [0.         0.         4.89897949 2.44948974 1.22474487]
 [0.         0.         0.         4.89897949 2.44948974]
 [0.         0.         0.         0.         4.89897949]]

L * L_transpus =
 [[32. 16.  8.  4.  2.]
 [16. 32. 16.  8.  4.]
 [ 8. 16. 32. 16.  8.]
 [ 4.  8. 16. 32. 16.]
 [ 2.  4.  8. 16. 32.]]
```

In [18]:
```
print(f"Soluția Sistemului -> x: \n{x}")
```

```
Soluția Sistemului -> x:
[[-0.29166667]
 [-0.875     ]
 [-1.44791667]
 [-1.375     ]
 [20.70833333]]
```

In [19]:
```
# Verificare
print(f"b: {b}\n")
print(f"A * x: \n{A@x}")
```

```
b: [  1.  16.  81. 256. 625.]

A * x:
[[  1.]
 [ 16.]
 [ 81.]
 [256.]
 [625.]]
```