Série de TP2

Méthode directe : élimination de Gauss

Ecrire en python le script qui permet de résoudre le système d'équation ci-dessous, par l'algorithme d'élimination de Gauss (pivot de Gauss)

$$\begin{cases} 2X_1 + 4X_2 - 2X_3 = -6 \\ X_1 + 3X_2 + X_4 = 0 \\ 3X_1 + X_2 + X_3 + 2X_4 = 8 \\ -X_2 + 2X_3 + X_4 = 6 \end{cases}$$

Etapes :

1. Étant donnés une matrice NumPy a et deux entiers i et j, échanger les lignes i et j de a.

```
def swap_lines(A, i, j):
    tmp = A[i, :].copy()
    A[i, :] = A[j, :]
    A[j, :] = tmp
```

2. Étant donnés une matrice NumPy a, deux entiers i et j et un flotant x, appliquer la transvection *Lj <-- Lj + xLi* sur la matrice a.

```
def transvection_lines(A, i, k, factor):
    A[k, :] = A[k, :] + A[i, :] * factor
```

3. Étant donnée une matrice A et un indice i, écrire la function pivot_index(A,i) qui renvoie l'indice j ≥ i t.q. $|a_{ji}|$ est maximal

```
def pivot_index(A, i):
    n = A.shape[0]  # nombre de lignes
    j = i
    for k in range(i + 1, n):
        if abs(A[k, i]) > abs(A[j, i]):
            j = k
    return j
```

4. Étant donnés une matrice A triangulaire supérieure, et un vecteur b de même taille, implémenter la résolution de Ax = b

```
def solve_triangular(A, b):
    n = len(b)
    x = numpy.zeros([n])  # vecteur de n zeros
    for i in range(n - 1, -1, -1):
        # x_i = 1/a_ii (b_i - sum_{j=i+1}^{n-1} a_ij x_j)
        sum = numpy.dot(A[i, :], x)
        x[i] = (b[i] - sum) / A[i, i]
    return x
```

5. Algorithme pivot de Gauss

```
def gauss(A0, b0):
    A = A0.copy() # pour ne pas détruire A
    b = b0.copy()
    n = len(b)  # on pourrait aussi la taille de a
    for i in range(n - 1):
        ipiv = pivot_index(A, i)
        if ipiv != i:  # echanges
            swap_lines(A, i, ipiv)
            tmp = b[ipiv]
            b[ipiv] = b[i]
            b[i] = tmp
        for k in range(i + 1, n):  # pivotage
            factor = -A[k, i] / A[i, i]
            transvection_lines(A, i, k, factor)
            b[k] = b[k] + b[i] * factor
    return [A, b]
```

6. Fonction principale

```
# Fonction principale
def solve(A0, b0):
    Ab = gauss(A0, b0)
    return solve_triangular(Ab[0], Ab[1])
```

7. Tester la solution