La simulation numérique
Résolution des systèmes linéaires sous Python

Implémentation de la méthode du pivot de Gauss en python

```python
import numpy as np
```

1. Ecrire une fonction python sans paramètres, appelée **taille**, qui permet de saisir et retourner un entier ≥ 2.

```python
def taille():
    while True:
        try:
            n=int(input("nombre d'équations = "))
            if n>=2:
                return n
            else:
                continue
        except:
            continue
```

2. Ecrire une fonction python, nommée **saisie_Vect** qui permet la saisie des n coefficients réels d'un vecteur. Cette fonction doit retourner une matrice B unicolonne d'ordre $(n, 1)$.

```python
def saisie_vect(n):
    L=[]
    for i in range(n):
        while True:
            try:
                x=float(input("coefficient d'indice {} ?".format(i)))
                break
            except:
                continue
        L.append(x)
    return np.array(L).reshape(n,1)
```

3. Ecrire une fonction python, nommée **saisie_Mat** qui permet la saisie des coefficients réels d'une matrice carrée d'ordre n. Cette fonction doit retourner une matrice A carrée d'ordre (n, n).

```
def saisie_mat(n):
    LL=[]
    for i in range(n):
        L=[]
        for j in range(n):
            while True:
                try:
                    x=float(input("coefficient d'indice {},{} ?".format(i,j)))
                    break
                except:
                    continue
            L.append(x)
        LL.append(L)
    return np.array(LL)
```

4. Ecrire une fonction python, nommée **triangulaire**, qui prend en paramètres une matrice **M** d'ordre $(n, n+1)$ et qui rend triangulaire supérieure la matrice carréé $M[1:n,1:n]$.

```
def triangulaire(M):
    n = M.shape[0]
    for k in range(n-1):
        if M[k,k] == 0: #Rechercher un pivot non nul
            l = k+1 #à partir de la ligne suivante
            while True:
                if l > n-1:
                    break
                if M[l,k] !=0:
                    break
                else:
                    l+=1
            if l > n-1: #pivot non nul n'existe pas dans cette colonne k
                print("pivot nul")
                break
            else: #Permuter les lignes l et k
                MT = np.copy(M) #Matrice temporaire
                M[k,:] = MT[l,:]
                M[l,:] = MT[k,:]
        if M[k,k] != 0:
            for i in range(k+1,n):
                #terme à terme entre ligne i et ligne k
                M[i,:]  = M[i,:] - (M[i,k]/M[k,k]) * M[k,:]
    return (M)
```

5. Ecrire une fonction python, nommée **remontee**, qui prend en paramètres une matrice **M** d'ordre $(n, n+1)$ triangulaire supérieure et retourne un vecteur **X** solution du système triangulaire.

```python
def remontee(M):
    X = np.zeros((n,1))
    X[n-1] = M[n-1,n]/M[n-1,n-1]
    for i in range(n-2,-1,-1):
        somme = M[i,n]
        for j in range(i+1,n):
            somme -= M[i,j]*X[j]
        X[i] = somme / M[i,i]
    return (X)
```

6. Ecrire une fonction python, nommée **pivot_Gauss**, qui prend en paramètres une matrice **A** et un vecteur **B** et retourne le vecteur **X**, solution du système.

```python
def pivot_Gauss(A,B):
    M = np.concatenate((A,B),axis=1) # M : Matrice augmentée
    M1 = triangulaire(M) #M1 : Matrice augmentée triangulaire supérieure
    X = remontee(M1) #X :vecteur solution
    return X
```

7. Déterminer la matrice des inconnus X :

$$A = \begin{pmatrix} 2 & -5 & 1 & 3 \\ 4 & 7 & 8 & 2 \\ 3 & 1 & 1 & 6 \\ 4 & 1 & 7 & 9 \end{pmatrix} \qquad X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \qquad B = \begin{pmatrix} 5 \\ 10 \\ 2 \\ 6 \end{pmatrix}$$

```python
n = taille()
#n=4

A = saisie_mat(n)
#A = np.array([[2,-5,1,3],[4,7,8,2],[3,1,1,6],[4,1,7,9]],float)

B = saisie_vect(n)
#B = np.array([5,10,2,6]).reshape(4,1)

print(pivot_Gauss(A,B))
X=[[ 2.29699248]
   [-0.45363409]
   [ 0.71303258]
   [-0.85839599]]


#Vérifier avec numpy.linalg.solve
x = numpy.linalg.solve(A,B); x
array([[ 2.29699248],
       [-0.45363409],
       [ 0.71303258],
       [-0.85839599]])
```