In [ ]:
```
import numpy as np
import time as t

#verif = True

#while verif:

choix = input('Si vous voulez donner ta propre matrice tapez << oui >> sinon tapez << non >> :')

if choix == 'oui':
    n = int(input("donner l'ordre de la matrice :"))
    A = []
    for i in range(n):
        #ajout a la fin de la liste
        A.append([0]*n)
        for j in range(n):
            A[i][j] = int(input("entrer les elements de la matrice A : "))

    b = []
    for i in range(0,n):
        elt = int(input("entrer les elements du vecteur b : "))
        b.append(elt)



else:
    A = np.random.randint(100,size=(5,5))
    b = np.random.randint(100,size=5)
    #if np.linalg.det(A) == 0 :
        #verif = True
    #else:
        #verif = False

w = float(input('donner le facteur de relaxation: '))

print (b);
print (A);
```

In [ ]:

In [ ]:
```
def relaxation(A,b,w,eps):

    start = t.time()
    # Matrices de décomposition D,E et F
    D = np.diag(np.diag(A)) # np.diag(A) retourne un vecteur donc il faut que j'ajoute np.diag puisque D est une matrice
    E = - np.tril(A,-1) # Matrice triangulaire inferieure (-1 car des zeros au dessus du diagonale + diagonale)
    F = - np.triu(A,1) # Matrice triangulaire supérieure (1 car des zeros au dessous du diagonale + diagonale)

    x = b
    nb_iter = 0

    # eps donne la precision sur Ax-b

    while np.linalg.norm(np.dot(A,x) - b) > eps:

        x = np.dot( np.linalg.inv(D - (w * E)) , np.dot((1-w) * D , x) + np.dot(w * F,x) ) + np.dot( w * (np.linalg.inv(D - w * E)) , b )
        nb_iter += 1

    c = np.linalg.cond(A,np.inf) #np.inf = norme de l'infinie
    end = t.time()
    return [c , x, start, end, D, E, F, nb_iter]


#nIter_Max = 100
eps = 0
#appel de la methode
[c , x,start, end, D, E, F, nb_iter] = relaxation(A,b,w,eps)

#Affichage
print('La matrice A: \n',A,'\n')
print('b=',b,'\n')
print('Cond(A)=',c,'\n')
print('La solution X: \nX= \n',x,'\n')
print('Le temps de calcul est: ',end - start,'\n')
print('Les matrices de décomposition: \nD = \n',D,' \nE = \n',E,' \nF = \n',F,'\n')
print("Le nombre d'iteration : ",nb_iter)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: