```
# ========================================================== #
# ALGEBRE et ARITHMETIQUE                                    #
# ENSAM (PSI) 2015 - Planche 269 - Exercice II               #
# (3ème exercice du recueil d'exercices pour les révisions) #
# ========================================================== #

from numpy import zeros
from random import random

# Génération de matrices carrées aléatoires à coefficients entiers.
# L'argument n passé à la fonction est la dimension de la matrice.
def MatGen(n):
    M = np.zeros((n,n))
    for i in range(n):
        for j in range(n):
            M[i,j] = int(9*random())
    return(M)

# Fonction déterminant si une matrice (B) est une sous-matrice d'une matrice
# donnée (A).
def MotifIn(A,B):
    # On récupère les dimensions des matrices A et B
    (nA,pA) = A.shape
    (nB,pB) = B.shape
    # Il faut nB <= nA ET pB <= pA
    if nB > nA or pB > pA :
        return(False)
    else:
        ncB = nB * pB
        for n in range(nA-nB+1):
            for p in range(pA-pB+1):
                c = 0
                for i in range(nB):
                    for j in range(pB):
                        if A[n+i,p+j] == B[i,j]:
                            c += 1
                if c == ncB :
                    return(True)
        return(False)

# Fonction déterminant le plus petit coefficient d'une matrice (A).
def MinMat(A):
    # On récupère les dimensions de la matrice A
    (nA,pA) = A.shape
    minA = A[0,0]
    for i in range(nA):
        for j in range(pA):
            if A[i,j] < minA:
                minA = A[i,j]
    return(minA)

# Fonction renvoyant la somme des carrés des différences entre coefficients
# de deux matrices de mêmes dimensions.
def SquaresSum(A,B):
    (n,p) = A.shape
    (n2,p2) = B.shape
    if n != n2 or p != p2:
        Return("Erreur de dimension")
    else:
        S = 0
        for i in range(n):
            for j in range(p):
                S += (A[i,j] - B[i,j])**2
        return(S)

# Fonction renvoyant un motif d'une matrice (A) le plus proche d'une autre
# matrice (B) au sens des moindres carrés.
```

```
def ClosestMotif(A,B):
    # On récupère les dimensions des matrices A et B
    (nA,pA) = A.shape
    (nB,pB) = B.shape
    # Il faut nB <= nA ET pB <= pA
    if nB > nA or pB > pA :
        return("Erreur de dimension")
    else:
        nbest, pbest = 0, 0
        Smin = SquaresSum(A[:nB,:pB],B)
        for n in range(nA-nB+1):
            for p in range(pA-pB+1):
                Scurr = SquaresSum(A[n:n+nB,p:p+pB],B)
                if Scurr < Smin :
                    Smin = Scurr
                    nbest, pbest = n, p
        return(A[nbest:nbest+nB,pbest:pbest+pB])
```