DS 3 - Informatique

Exercice 1 On dit qu'une matrice est tridiagonale lorsqu'elle est de la forme :

$$\begin{pmatrix} a & c & 0 & \cdots & 0 \\ b & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & c \\ 0 & \cdots & 0 & b & a \end{pmatrix}$$

où a, b, c sont trois réels.
Dans cet exercice, on utilise le module numpy avec l'alias np

Pour construire la matrice nulle avec n lignes et p colonnes, on utilisera la commande np.zeros((n,p)).

On rappelle que si M est une matrice numpy, on obtient les dimensions de la matrice grâce à la commande n,p=np.shape(M).

1. Écrire une fonction tridiagonale qui retourne la matrice tridiagonale d'ordre n avec a sur diagonale, b au-dessous, c au-dessus.
   L'entier n et les réels a, b et c sont donnés en paramètres.
2. Écrire une fonction augmente qui prend en argument une matrice carrée A et qui retourne la matrice augmentée $(A|I)$ où I est la matrice identité de même taille que A.
3. Écrire une fonction transvection qui prend en argument une matrice M et qui remplace la ligne L_i de M par la ligne $L_i - xL_j$.
   Les entiers i et j ainsi que le réel x sont donnés en argument.
4. Expliquer la fonction trinf ci-dessous.

```
def trinf(M):
    n,p=np.shape(M)
    for i in range(0,n-1):

        j=i
        for k in range(i+1,n):
            if abs(M[k,i])>abs(M[j,i]):
                j=k

        for k in range(p):
            M[i,k],M[j,k]=M[j,k],M[i,k]

        for j in range(i+1,n):
            transvection(M,j,i,M[j,i]/M[i,i])
```

5. Expliquer la fonction diagonale ci-dessous.

```
def diagonale(M):
    n,p=np.shape(M)
    for i in range(n-1,0,-1):
        for j in range(i-1,-1,-1):
            transvection(M,j,i,M[j,i]/M[i,i])
```

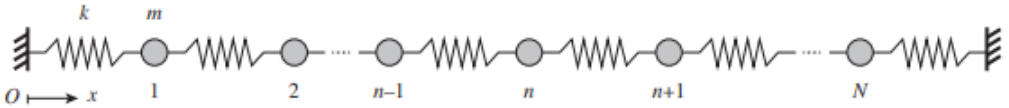
6. Compléter la fonction ci-dessous pour qu'elle retourne la matrice inverse de la matrice tridiagonale de paramètres n, a, b, c.

```
def inverse(n,a,b,c):
    M=tridiagonale(n,a,b,c)
    N= ---------------------------
       ---------------------------
       ---------------------------
    P=np.zeros((n,n))
    for i in range(n):
        for j in range(n):
            P[i,j]= -------------------
    return P
```

7. Améliorer la fonction précédente pour qu'elle retourne l'inverse si la matrice est inversible et False si elle ne l'est pas.

Exercice 2

On s'intéresse au comportement d'une chaîne de N oscillateurs montés en série entre deux bâtis fixes (figure ci-dessous). Tous les oscillateurs sont supposés identiques, de longueur L et constitués chacun d'une masselotte de masse m et d'un ressort de raideur k_0. Par simplicité, on ne prend pas en compte les frottements ni les effets de la gravité.



On suppose que le système oscille dans la direction (Ox). Le PFD appliqué à la masselotte numéro n s'écrit, pour n compris entre 1 et N :

$$\ddot{x}_n = -\omega_0^2(2x_n - x_{n-1} - x_{n+1}) \quad \text{avec} \quad \omega_0 = \sqrt{\frac{k}{m}}$$

où x_n est l'écart de la masselotte n par rapport à sa position d'équilibre (la position de la masselotte n à l'équilibre est $n \times L$).

Les conditions aux bords fixes imposent que $x_0 = x_{N+1} = 0$. Donc $\dot{x}_0 = \dot{x}_{N+1} = 0$ et $\ddot{x}_0 = \ddot{x}_{N+1} = 0$.

La liste des écarts par rapport aux positions d'équilibre $(x_0, \ldots, x_{N+1})$ est stockée dans une liste `X`

La liste des vitesses $(\dot{x}_0, \ldots, \dot{x}_{N+1})$ est stockée dans une liste `V`

La liste des accélérations $(\ddot{x}_0, \ldots, \ddot{x}_{N+1})$ est stockée dans une liste `A`

Le script commence par les lignes suivantes :

```
import numpy as np
import matplotlib.pyplot as plt
k,m,L,tmax,dt=1,1,1,10,1e-3
```

1. À l'instant 0, on écarte l'oscillateur i de ε par rapport à sa position d'équilibre ; les autres oscillateurs ne sont pas déplacés et toutes les vitesses sont nulles. Écrire une fonction `initialisation` de paramètre N, i et ε, et qui retourne la liste des écarts `X` et la liste des vitesses `V` à l'instant 0.
2. Écrire une fonction `f` de paramètre la liste des écarts `X` à un instant t et qui renvoie la liste `A` des accélérations au même instant.
3. Après avoir déplacé l'oscillateur i, on veut déterminer la position de chaque oscillateur aux instants $t_k = k \times dt$ inférieurs à $t_{\max}$. Pour cela, on applique l'algorithme d'Euler utilisant les approximations affines :
$$x(t+dt) = x(t) + \dot{x}(t)dt$$
$$\dot{x}(t+dt) = \dot{x}(t) + \ddot{x}(t)dt$$
où dt est le pas de l'approximation.
Les positions aux instants $t_0, t_1, t_2, \ldots$ de l'oscillateur k sont stockées dans la liste `Ressort[k]`.
Compléter la fonction `mouvement` qui calcule les positions de tous les oscillateurs sur l'intervalle de temps $[0, t_{\max}]$:

```
def mouvement(N,i,epsilon):
    t,T=  ----------
    X,V=  ----------

    Ressorts=[[  ----------  ] for k in range(N+1)]

    while t<tmax:
        A=  ----------
        for k in range(1,N+1):
            X[k]=  ----------
            V[k]=  ----------
        t=  ----------
        T.append(t)
        for k in range(0,N+1):
            Ressorts[k].append(  ----------  )

    return T,Ressorts
```

4. Écrire une fonction `graphique` de paramètre (N, i, ε) et qui représente les positions des oscillateurs en fonction du temps.