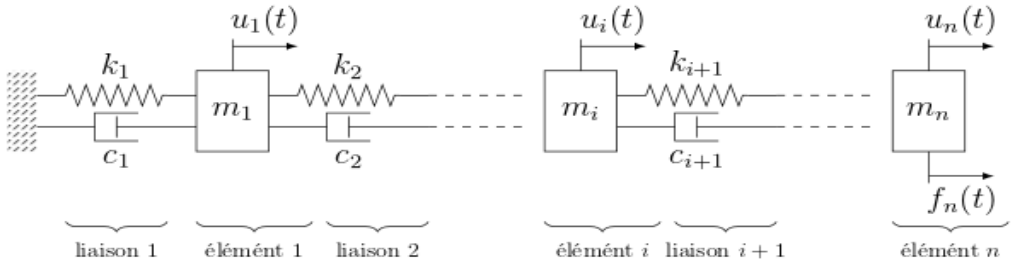
Devoir de modélisation

On considère une structure de n éléments de masse m_i pour i variant de 1 à n, reliés par des liaisons visco-élastiques de raideur k_i et d'amortissement c_i (voir la figure ci-dessous). La structure est supposée unidimensionnelle de longeur L.

Le déplacement au cours du temps de l'élément i autour de sa position d'équilibre est noté $u_i(t)$. Une force $f_n(t)$ est appliquée sur l'élément n uniquement. L'extrémité gauche de la structure est bloquée. Les effets de la pesanteur sont négligés.



Le principe fondamental de la dynamique appliqué à un élément i permet d'aboutir aux équations suivantes pour $i = 2$ jusqu'à $n - 1$:

$$m_i \frac{\mathrm{d}^2 u_i(t)}{(\mathrm{d}t)^2} = -k_i\,(u_i(t) - u_{i-1}(t)) - k_{i+1}\,(u_i(t) - u_{i+1}(t)) - c_i\left(\frac{\mathrm{d}u_i(t)}{\mathrm{d}t} - \frac{\mathrm{d}u_{i-1}(t)}{\mathrm{d}t}\right) - c_{i+1}\left(\frac{\mathrm{d}u_i(t)}{\mathrm{d}t} - \frac{\mathrm{d}u_{i+1}(t)}{\mathrm{d}t}\right)$$

Les équations des éléments situés aux extrémités sont les suivantes :

$$m_1 \frac{\mathrm{d}^2 u_1(t)}{(\mathrm{d}t)^2} = -k_1\,u_1(t) - k_2\,(u_1(t) - u_2(t)) - c_1\frac{\mathrm{d}u_1(t)}{\mathrm{d}t} - c_2\left(\frac{\mathrm{d}u_1(t)}{\mathrm{d}t} - \frac{\mathrm{d}u_2(t)}{\mathrm{d}t}\right)$$

$$m_n \frac{\mathrm{d}^2 u_n(t)}{(\mathrm{d}t)^2} = -k_n\,(u_n(t) - u_{n-1}(t)) - c_n\left(\frac{\mathrm{d}u_n(t)}{\mathrm{d}t} - \frac{\mathrm{d}u_{n-1}(t)}{\mathrm{d}t}\right) + f_n(t)$$

1. Création de matrices

Le système d'équations différentielles défini ci-dessus peut s'écrire sous forme matricielle :

$$M\,\ddot{X}(t) + C\,\dot{X}(t) + K\,X(t) = F(t)$$

où $X(t) = \begin{pmatrix} u_1(t) \\ \vdots \\ u_n(t) \end{pmatrix}$, les matrices M, C et K sont des matrices carrées de taille n, et F une matrice unicolonne de n lignes.

a. Déterminer les matrices M, K et C.

b. On suppose que les masses m_i, les coefficients k_i et les coefficients c_i sont répertoriés dans des listes.
Écrire en Python une fonction notée `creation_C` qui prend en entrée l'entier n et une liste `ListeC` (qui dans l'application de la fonction sera la liste des coefficients c_i) et qui retourne la matrice C.
On pourra utiliser la commande `zeros((n,n),float)` qui permet de créer la matrice nulle de taille $n \times n$.

2. Résolution

La résolution de l'équation différentielle matricielle ci-dessus est obtenue à l'aide d'un schéma d'Euler.

L'intervalle de temps $[0, T]$ est discrétisé en pas de temps de même durée, notés $\mathrm{d}t$. Ainsi le piquet de temps t_{q+1} est donné par $t_{q+1} = \mathrm{d}t + t_q$, soit, par récurrence, $t_q = q\,\mathrm{d}t$ avec $t_0 = 0$.
À l'instant t_q, le vecteur déplacement est noté X_q (à la place de $X(t_q)$), la vitesse $\dot{X}_q$ et l'accélération $\ddot{X}_q$.
Le schéma d'Euler consiste à écrire $\dot{X}_q = \dfrac{X_q - X_{q-1}}{\mathrm{d}t}$ et $\ddot{X}_q = \dfrac{\dot{X}_q - \dot{X}_{q-1}}{\mathrm{d}t}$.
Écrire le système matriciel à l'instant t et le mettre sous la forme $H\,X_q = G_q$ où vous exprimerez H en fonction de M, K et C et le pas de temps $\mathrm{d}t$, et G_q en fonction de M, K, C, X_{q-1}, X_{q-2}, $\mathrm{d}t$ et F_q le vecteur force calculé à l'instant t_q.
La matrice H est calculée une fois pour toute avant les itérations sur les piquets de temps. Elle à la forme suivante :

$$H = \begin{pmatrix} H_{1,1} & H_{1,2} & 0 & \dots & 0 \\ H_{2,1} & H_{2,2} & H_{2,3} & \ddots & \vdots \\ 0 & H_{3,2} & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & H_{n-1,n-1} & H_{n-1,n} \\ 0 & \dots & 0 & H_{n,n-1} & H_{n,n} \end{pmatrix}$$

Le second membre G_q est réévalué à chaque piquet de temps puis le système est résolu pour obtenir X_q. Le vecteur de force extérieur F_q est calculé dans la boucle itérative du temps afin de ne pas stocker sa valeur sur tout l'intervalle de temps.
Le résultat de la simulation sera stocké dans une matrice notée X de taille $n \times ntemps$ du type
$X = (X_0, X_1, \dots, X_q, \dots)$ avec $ntemps$ le nombre de pas de temps et n le nombre d'éléments de la structure.

a. La résolution de l'équation $H\,X_q = G_q$ peut-être faite à l'aide d'un pivot de Gauss classique. Cependant compte-tenu de la forme tridiagonale de la matrice H, la procédure peut-être simplifiée.
Écrire en Python une fonction que prend en entrée une matrice du type H et qui retourne par cette méthode simplifiée une matrice triangulaire supérieure.
On supposera que tous les "pivots" utilisés sont non nuls.

b. Déterminer la complexité de cette méthode et comparer à la complexité de la méthode de Gauss classique.

c. En pratique, les matrices M, K, C et H et le vecteur G contiennent très peu de termes et sont stockés de manière particulière. Ceci ne constitue pas un coût de stockage important; par compte le stockage de la matrice X peut s'avérer très lourd si le nombre de masses n est grand et le pas de temps $\mathrm{d}t$ petit.
Une étude typique consiste à résoudre ce système matriciel avec les caractéristiques suivantes :

$$n = 200, \quad T = 0.3s, \quad \text{pas de temps : } 2 \times 10^{-6}s$$

avec un stockage de la solution au format réel double précision (64 bits) sur toute la durée T de la simulation.
Déterminer la quantité de mémoire RAM en Go nécessaire pour stocker le résultat d'un calcul sur la durée T. On ne tiendra compte que de la matrice X.