1 Equation aux valeurs propres associé à un oscillateur linéaire

Le but de cette étude est de calculer les fréquences propres et les modes propres d'un oscillateur linéaire formé de quatre sphères couplées deux à deux par des ressorts.

Pour simplifier, on suppose que les sphères sont de masse identique m et que les ressorts ont tous la même constante de raideur k.

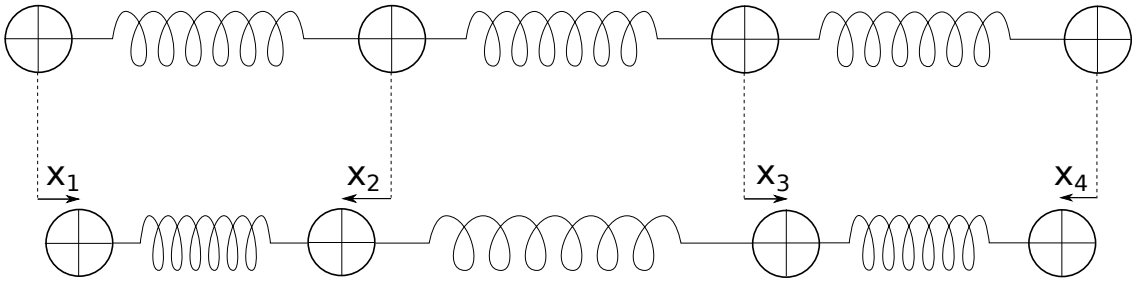

FIGURE 1 – ressorts couplés

Le principe fondamental de la dynamique appliqué successivement à chaque masse mène à un système de quatre équations différentielles couplées :

$$\left\{\begin{array}{lcl} m\ddot{x}_1 &=& -k\,(x_1-x_2) \\ m\ddot{x}_2 &=& -k\,(x_2-x_1)-k\,(x_2-x_3) \\ m\ddot{x}_3 &=& -k\,(x_3-x_2)-k\,(x_3-x_4) \\ m\ddot{x}_4 &=& -k\,(x_4-x_3) \end{array}\right. \iff \begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \ddot{x}_3 \\ \ddot{x}_4 \end{pmatrix} = -\frac{k}{m}\begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

En se plaçant dans le régime harmonique et en utilisant la notation complexe, on écrit : $x_i(t) = a_i\, exp(i\omega t)$ où $i \in [1,4]$ et $a_i \in \mathbb{C}$. Noter que les déplacements physiques coïncident avec les parties réelles de ces quantités.

On obtient, en notant $\omega_0^2 = k/m$, le système linéaire suivant :

$$\omega_0^2\begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \omega^2\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix}$$

Le problème revient à diagonaliser une matrice de couplage que l'on note K.

$$K = \omega_0^2\begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

où chaque valeur propre λ correspond au carré de la pulsation propre ω.

1.1 Mise en application

Développer un script Python permettant de déterminer les pulsations propres et les modes associés.

- La matrice K est une matrice pleine stockée dans un tableau numpy.ndarray. Les valeurs propres et vecteurs propres sont calculées avec la fonction `linalg.eig` de la librairie `numpy`.
- La matrice K est une matrice creuse. On utilisera la fonction `sparse.csr_matrix` pour stocker les termes non nuls de la librairie `scipy`. Les valeurs propres et vecteurs propres sont calculées avec la fonction `sparse.linalg.eigsh` de la librairie `scipy`.

2 Application de conditions de déplacement nul en certains noeuds

On présente ici une méthode générale pour calculer les modes propres en s'appuyant sur l'étude précédente lorsque certaines masses du systèmes sont fixes. Ces conditions s'apparentent aux conditions de dirichlet du type $x_i = 0$.

En notant N_d le nombre de masses où la condition de dirichlet est imposée, le nombre de degrés de liberté se réduit alors à $N_{dof} = N - N_d$ et correspond ici au nombre de masses pouvant osciller.

2.1 Algorithme

L'algorithme consiste à former une liste l où le numéro de chaque masse pouvant osciller, apparaît de façon unique. Sa taille est $N_{dof} = N - N_d$ et s'identifie, ici, au nombre de degrés de liberté.

On forme ensuite une matrice de projection P permettant de passer de l'espace des solutions à N degrés de liberté (étude précédente) à celui restreint à N_{dof} degrés de liberté. Les dimensions de P sont $N_{dof} \times N$ (nombre de lignes $\times$ nombre de colonnes). On initialise d'abord la matrice P à zéro ; puis en parcourant séquentiellement la liste l, on impose $P[i, l[i]] = 1$ avec $i \in [1, N_{dof}]$. On remarque que $PP^t = \texttt{Id}(N_{dof})$.

Une autre façon de construire la matrice P est de l'initialiser avec la matrice identité $\texttt{Id}(N)$ puis de supprimer toutes les lignes correspondant aux noeuds de dirichlet. Cette technique est bien adaptée à Numpy/Scipy en masquant les numéros de lignes correspondant aux noeuds de dirichlet.

```
1 def delete_rows(mat, indices):
2     indices = list(indices) # transformation en liste
3     mask = np.ones(mat.shape[0], dtype=bool) # ligne de True
4     mask[indices] = False # True->False pour les noeuds de dirichlet
5     return mat[mask]
```

$$P = \begin{pmatrix} \cancel{1} & \cancel{0} & \cancel{0} & \cancel{0} \\ 0 & 1 & 0 & 0 \\ \cancel{0} & \cancel{0} & \cancel{1} & \cancel{0} \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

FIGURE 2 – exemple de matrice P avec $ld = [0, 2]$

On construit ensuite K_p qui est la matrice projetée de K dans l'espace solution à N_{dof} degrés de liberté : $K_p = PKP^t$. C'est cette matrice qui est finalement diagonalisée. On obtient alors les valeurs propres λ_p associées aux vecteurs propres a_p. Autrement dit, cela revient à résoudre : $PKP^t a_p = \lambda_p a_p$.

On reconstruit la solution dans l'espace à N corps en appliquant $a = P^t a_p$. Elle vérifie automatiquement, $a[i] = 0$ pour tout noeud i de dirichlet.

2.2 Mise en application

- Utiliser sous Numpy, la fonction `ld=numpy.unique(ld)` pour éliminer tout doublon dans la liste `ld`.
- Calculer numériquement les modes propres du système lorsque les masses 0 et 2 sont fixes.
- Vérifier vos résultats numériques avec une approche analytique.