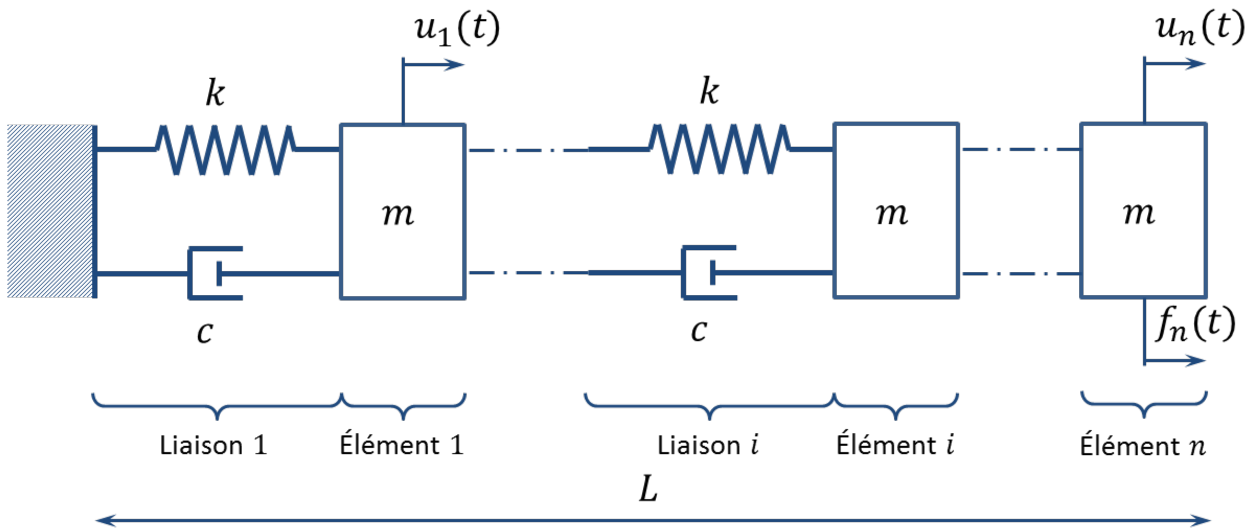
Concours Blanc : Informatique

Autour de données météorologiques

1 Mise en situation

La structure peut être modélisée par n éléments de masse m_i (i variant de 1 à n) reliés par des liaisons visco-élastiques eux-mêmes modélisés par des un ressort de raideur k en parallèle d'un élément d'amortissement c. La structure est supposée unidimensionnelle de longueur L. Le nombre d'éléments peut être de l'ordre de plusieurs milliers.

Le déplacement au cours du temps de l'élément i autour de sa position d'équlibre 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 théorème de la résultante dynamique appliqué à un élément i (i variant de 2 à $n-1$ inclus) s'écrit sous la forme :

$$m\frac{\mathrm{d}^2 u_i(t)}{\mathrm{d}t^2} = -2k u_i(t) + k u_{i-1}(t) + k u_{i+1}(t) - 2c\frac{\mathrm{d}u_i(t)}{\mathrm{d}t} + c\frac{\mathrm{d}u_{i-1}(t)}{\mathrm{d}t} + c\frac{\mathrm{d}u_{i+1}(t)}{\mathrm{d}t} \tag{1}$$

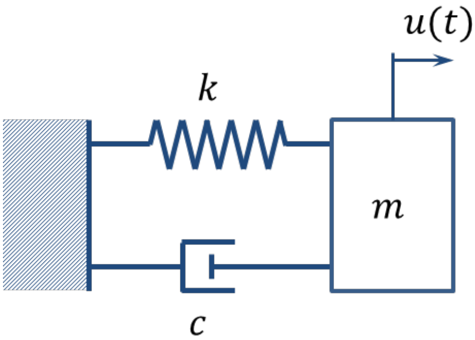
Le théorème de la résultante dynamique appliqué aux éléments 1 et n donne les équations suivantes :

$$m\frac{\mathrm{d}^2 u_1(t)}{\mathrm{d}t^2} = -2k u_1(t) + k u_2(t) - 2c\frac{\mathrm{d}u_1(t)}{\mathrm{d}t} + c\frac{\mathrm{d}u_2(t)}{\mathrm{d}t} \tag{2}$$

$$m\frac{\mathrm{d}^2 u_n(t)}{\mathrm{d}t^2} = -k\left(u_n(t) - u_{n-1}(t)\right) - c\left(\frac{\mathrm{d}u_n(t)}{\mathrm{d}t} - \frac{\mathrm{d}u_{n-1}(t)}{\mathrm{d}t}\right) + f_n(t) \tag{3}$$

Dans toute la suite, on imposera $f_n(t) = f_{max}\sin(\omega t)$.

2 Résolution d'une équation différentielle



On s'intéresse tout d'abord à une seule cellule masse ressort amortisseur. L'application du théorème de la résultante dynamique en projection sur l'axe de déplacement s'écrit donc de la manière suivante :

$$m \cdot \ddot{u}(t) + c \cdot \dot{u}(t) + k \cdot u(t) = f(t)$$

On précise que pour $t \leq 0$, $u(t) = 0$, $\dot{u}(t) = 0$ et $\ddot{u}(t) = 0$.

On pose $\left\{ \begin{array}{l} x(t) = u(t) \\ v(t) = \dot{x}(t) \end{array} \right.$.

Question 1 *Réécrire l'équation différentielle sous forme d'un système d'équation en fonction de $x(t)$, $v(t)$, $\dot{x}(t)$ et $\dot{v}(t)$*

Corrigé

On a donc :

$$\left\{ \begin{array}{l} v(t) = \dot{x}(t) \\ m \cdot \dot{v}(t) + c \cdot v(t) + k \cdot x(t) = f(t) \end{array} \right.$$

Question 2 *En utilisant un schéma d'Euler implicite et l'équation $v(t) = \dot{x}(t)$ exprimer la suite x_n en fonction de x_{n-1}, v_n et du pas de calcul noté noté h.*

Corrigé

On a $\dfrac{dx(t)}{dt} \simeq \dfrac{x_n - x_{n-1}}{h}$. On a donc $v_n = \dfrac{x_n - x_{n-1}}{h} \iff x_n = h \cdot v_n + x_{n-1}$.

Question 3 *En utilisant un schéma d'Euler implicite exprimer v_n en fonction de v_{n-1}, x_n, x_{n-1}, f_n, h, k et c.*

Corrigé

On a $\dfrac{dv(t)}{dt} \simeq \dfrac{v_n - v_{n-1}}{h}$. On a donc $\dot{v}_n = \dfrac{v_n - v_{n-1}}{h} \iff v_n = h \cdot \dot{v}_n + v_{n-1}$.

$$m \cdot \frac{v_n - v_{n-1}}{h} + c \cdot \frac{x_n - x_{n-1}}{h} + k \cdot x_n = f_n \iff m \cdot (v_n - v_{n-1}) + c \cdot (x_n - x_{n-1}) + kh \cdot x_n = hf_n$$

On a donc :

$$mv_n = hf_n - (kh + c) \cdot x_n + mv_{n-1} + cx_{n-1}$$

Question 4 *En déduire que la suite x_n peut se mettre sous la forme suivante :*

$$x_n \left(m + ch + kh^2\right) - x_{n-1}(2m + ch) + mx_{n-2} = h^2 f_n$$

Corrigé

D'après la question 2 on a :

$$v_n = \frac{x_n - x_{n-1}}{h} \quad \text{et} \quad v_{n-1} = \frac{x_{n-1} - x_{n-2}}{h}$$

En utilisant le résultat de la question 3, on a :

$$m\frac{x_n - x_{n-1}}{h} = h f_n - (kh+c)\cdot x_n + m\frac{x_{n-1} - x_{n-2}}{h} + c\,x_{n-1}$$

$$\Longleftrightarrow m(x_n - x_{n-1}) = h^2 f_n - h(kh+c)\cdot x_n + m(x_{n-1} - x_{n-2}) + c\,h\,x_{n-1}$$

$$\Longleftrightarrow m(x_n - x_{n-1}) + h(kh+c)\cdot x_n - m(x_{n-1} - x_{n-2}) - c\,h\,x_{n-1} = h^2 f_n$$

$$\Longleftrightarrow x_n\left(m + kh^2 + ch\right) - x_{n-1}(2m + ch) + m\,x_{n-2} = h^2 f_n$$

$$\Longleftrightarrow x_n = \frac{1}{m + kh^2 + ch}\left(h^2 f_n + x_{n-1}(2m + ch) - m\,x_{n-2}\right)$$

Question 5 *On note Tsimu le temps de simulation et h le pas de temps. Implémenter en Python le fonction f_omega permettant de créer une liste contenant l'ensemble des valeurs prises par la fonction f_n. On utilisera une boucle while. Les spécifications de la fonction sont les suivantes :*

```python
def f_omega(Tsimu,h,fmax,fsign):
    """
    Entrées :
        * Tsimu ( flt ) : temps de la simulation en seconde
        * h ( flt ) : pas de temps de a simulation
        * fmax ( flt ) : amplitude du signal (en Newton)
        * fsign ( flt ) : fréquence du signal (en Hertz)
    Sortie :
        * F ( list ) : liste des valeurs de la fonction
          f_n (t)= fmax sin (omega *t)
    """
```

Corrigé

```python
def f_omega(Tsimu,h,fmax,fsign):
    """
    Entrées :
        * Tsimu (flt) : temps de la simulation en seconde
        * h (flt) : pas de temps de a simulation
        * fmax (flt) : amplitude du signal (en Newton)
        * fsign (flt) : fréquence du signal (en Hertz)
    Sortie :
        * F (list) : liste des valeurs de la fonction
          f_n (t)= fmax sin (omega *t)
    """
    omega = 2*math.pi*fsign
    t=0
    F = []
    while t<Tsimu :
        F.append(fmax*math.sin(omega*t))
        t=t+h
    return F
```

Il est possible de mettre la suite déterminée à la question 4 sous la forme $x_n = \alpha f_n + \beta x_{n-1} + \gamma x_{n-2}$.

Question 6 *En utilisant une boucle while, générer, en Python, les listes T et X contenant respectivement le temps de simulation et le déplacement de la masse mobile.*

Corrigé

```python

T=[0,h]
X=[0,0]
t=2*h
i=2
while t<Tsimu:
    T.append(t)
    X.append(alpha*F[i]+beta*X[i-1] +gamma*X(i-2))
    i=i+1
    t = t+h
```

3 Résolution du problème général

Le système d'équations différentielles défini dans la première partie peut s'écrire sous forme matricielle :

$$M\ddot{X}(t)+C\dot{X}(t)+KX(t)=F(t) \quad \text{avec} \quad X(t)=\begin{bmatrix} u_1(t) \\ \dots \\ u_n(t) \end{bmatrix}$$

M, C et K des matrices carrées de taille $n \times n$.

Question 7 *En reprenant les équations (1), (2) et (3) déterminer les matrices M, C, K et F.*

Corrigé

$$M=\begin{pmatrix} m & 0 & 0\\ 0 & \ddots & 0\\ 0 & 0 & m\end{pmatrix}\quad C=\begin{pmatrix} 2c & -c & 0 & \cdots & \cdots & \cdots & 0\\ -c & 2c & -c & \ddots & & 0 & \vdots\\ 0 & -c & 2c & \ddots & \ddots & & \vdots\\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots\\ \vdots & & \ddots & \ddots & 2c & -c & 0\\ \vdots & 0 & & \ddots & -c & 2c & -c\\ 0 & \cdots & \cdots & \cdots & 0 & -c & 2c\end{pmatrix}\quad K=\begin{pmatrix} 2k & -k & 0 & \cdots & \cdots & \cdots & 0\\ -k & 2k & -k & \ddots & & 0 & \vdots\\ 0 & -k & 2k & \ddots & \ddots & & \vdots\\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots\\ \vdots & & \ddots & \ddots & 2k & -k & 0\\ \vdots & 0 & & \ddots & -k & 2k & -k\\ 0 & \cdots & \cdots & \cdots & 0 & -k & 2k\end{pmatrix}\quad F=\begin{pmatrix}0\\ \vdots\\ 0\\ f_n(t)\end{pmatrix}$$

En appliquant le schéma d'Euler explicite à l'équation différentielle matricielle, la solution reste identique à celle déterminée dans la partie précédente :

$$\left(M+h^2K+hX\right)X_n=\left(h^2F_n+X_{n-1}(2M+hC)-MX_{n-2}\right)$$

on montre qu'à l'instant k, le système peut se mettre sous la forme $H\cdot X_k=G_k$, la matrice H étant de la forme suivante (dite tridiagonale) :

$$H=\begin{pmatrix} H_{00} & H_{01} & 0 & 0 & 0\\ H_{10} & H_{11} & \ddots & 0 & 0\\ 0 & \ddots & \ddots & \ddots & 0\\ 0 & 0 & \ddots & H_{n-1,n-1} & H_{n-1,n}\\ 0 & 0 & 0 & H_{n-1,n} & H_{n,n}\end{pmatrix}$$

Remarque

Dans ces conditions on a donc $G_k=\left(\dfrac{2M}{h^2}+\dfrac{C}{h}\right)X_{k-1}-\dfrac{M}{h^2}X_{k-2}+F_k$ et $H=\dfrac{M}{h^2}+\dfrac{C}{h}+K$.

Question 8 *L'équation $H\cdot X_k=G_k$ peut être résolue grâce à la méthode du pivot de Gauss. Donner les étapes de cette méthode ainsi que l'objectif ce chacune d'entre elles. Quelle est la complexité algorithmique de la résolution d'une équation en utilisant le pivot de Gauss ?*

Corrigé

Plutôt que d'utiliser la méthode du pivot de Gauss classique, un programmateur utilise la méthode suivante pour résoudre le système d'équation :

```python
def resoud(H,X,G):
    """
    Fonction permettant de résoudre H.X = G
    Entrées :
        * H ( list ) : matrice de tridiagonale de taille nxn
```

```python
        * X ( list ) : vecteur de n lignes , 1 colonne dont toutes les valeurs sont nulles
        * G ( list ) : second membre du système d'équation de taille nx1
    Sortie :
        * X : solution du système
    """
    H2=copy(H) # Copie de la matrice H
    G2=copy(G) # Copie du vecteur G
    n=len(G2)
    for i in range(1,n):
        a=H2[i-1,i-1]
        b=H2[i,i-1]
        for k in range(n) :
            H2[i][k]=H2[i][k]*a-H2[i-1][k]*b
            G2[i][k]=G2[i][k]*a-G2[i-1][k]*b
    for k in range(n) :
        X[n-1][k]=G2[n-1][k]/H2[n-1][n-1]
    for i in range(n-2,-1,-1):
        for k in range(n) :
            X[i][k]=(G2[i][k]-H2[i][i+1]*X[i+1][k])/H2[i][i]
    return X
```

Question 9 *Expliquer en quoi cet algorithme permet de résoudre le système $H \cdot X_n = G_n$? Quelle est la complexité de cet algorithme ?*

Corrigé

4 Détermination de l'énergie dissipée

La puissance dissipée par la poutre est donnée par la relation suivante :

$$P_{\text{diss}}(t) = \sum_{i=2}^{n} c\,(u_i(t) - \dot{u}_i(t))^2$$

L'énergie imposée est donnée par la relation :

$$E_{\text{diss}}(t) = \int_0^t P_{\text{diss}}(\tau)\, d\tau$$