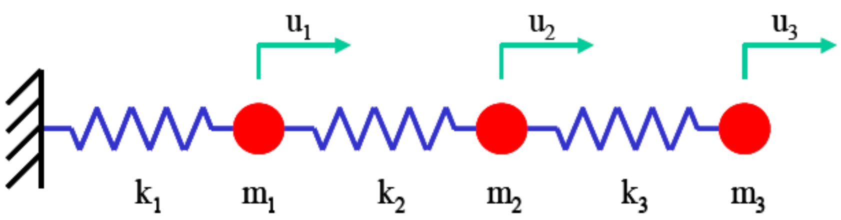
On considère le système composé de trois masses m1, m2 et m3, et de trois ressorts de traction compression k1, k2 et k3. Chaque masse est repérée par sa position le long d'un axe horizontal, les paramètres de déplacement des masses sont notés u1, u2 et u3.



$$k_1 = 10^4 \text{ N/m} \qquad k_2 = 5.10^4 \text{ N/m} \qquad k_3 = 3.10^4 \text{ N/m}$$

$$m_1 = 100 \text{ kg} \qquad m_2 = 75 \text{ kg} \qquad m_3 = 50 \text{ kg}$$

$$F_0 = 100 \text{ N} \qquad \xi = 3\ \% \qquad V_0 = 1 \text{ m/s}$$

Déterminer :

les fréquences propres et les vecteurs propres du système ; les modes seront normés en composante maximale unitaire.

Etude théorique : Un rapide calcul donne :

$$2E_C = m_1\dot{u}_1^2 + m_2\dot{u}_2^2 + m_3\dot{u}_3^2 \qquad \text{soit :} \quad \mathbb{M} = \begin{bmatrix} m_1 & & \\ & m_2 & \\ & & m_3 \end{bmatrix}$$

$$2E_D = k_1u_1^2 + k_2(u_2 - u_1)^2 + k_3(u_3 - u_2)^2 \qquad \text{soit :} \quad \mathbb{K} = \begin{bmatrix} k_1 + k_2 & -k_2 & 0 \\ -k_2 & k_2 + k_3 & -k_3 \\ 0 & -k_3 & k_3 \end{bmatrix}$$

Les pulsations propres et les vecteurs propres du système sont donc déterminés à partir de ces deux matrices.

Les vecteurs propres seront normés en composante maximale unitaire (plus grande valeur absolue égale à 1).

Mise en oeuvre sous Matlab :

```
lose all
clear all
% SAISIE DES DONNEES
% raideurs
k1 = 1e4 ;
k2 = 5e4 ;
k3 = 3e4 ;
% masses
m1 = 100 ;
m2 = 75 ;
m3 = 50 ;
% MATRICES DU PROBLEMES
% matrice de masse
matm = diag([m1 m2 m3]) ;
% matrice de raideur
matk = [k1+k2 -k2 0 ; ...
-k2 k2+k3 -k3 ; ...
0 -k3 k3 ] ;
% RECHERCHE DES MODES PROPRES
[V,D]=eig(matk,matm) ;
% fréquences propres
fqce = sqrt(diag(D))/(2*pi)
% vecteurs propres normés en composante maximale unitaire
V1 = V(:,1)/max(abs(V(:,1)))*sign(V(find(abs(V(:,1))==max(abs(V(:,1)))),1))
V2 = V(:,2)/max(abs(V(:,2)))*sign(V(find(abs(V(:,2))==max(abs(V(:,2)))),2))
V3 = V(:,3)/max(abs(V(:,3)))*sign(V(find(abs(V(:,3))==max(abs(V(:,3)))),3))
```

Résultats :

fqce = 1.0192 3.8985 6.4170	V1 = 0.8333 0.9317 1.0000	V2 = -0.6000 0.0000 1.0000	V3 = -0.4875 1.0000 -0.5850

Simutation numérique d'un système en vibration:

On doit convertir le second ordre en deux équations du premier ordre. On définit donc deux nouvelles variables *x_1* and *x_2* :

$$m\ddot{x} + c\dot{x} + kx = 0$$

$$\text{Let } \; x_1 = x, \quad x_2 = \dot{x}$$

On écrit donc deux EDO du premier ordre:

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -\frac{c}{m}x_2 - \frac{k}{m}x_1$$

Called state space

Ecriture Matricielle

En combinant:

$$\underbrace{\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} 0 & 1 \\ -\dfrac{k}{m} & -\dfrac{c}{m} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{x}$$

State vector

State matrix

La méthode d'Euler peut nous donner une solution approchée:

$$x(t_{i+1}) = x(t_i) + \Delta t A x(t_i)$$

$$x_{i+1} = \left[I + \Delta t A\right] x_i$$

Fonctions Matlab ‘ode23’ et ‘ode45’

- **Utilise Runge-Kutta (plus précis)**
- **Fonction avec fonction NL**

Create Matlab function

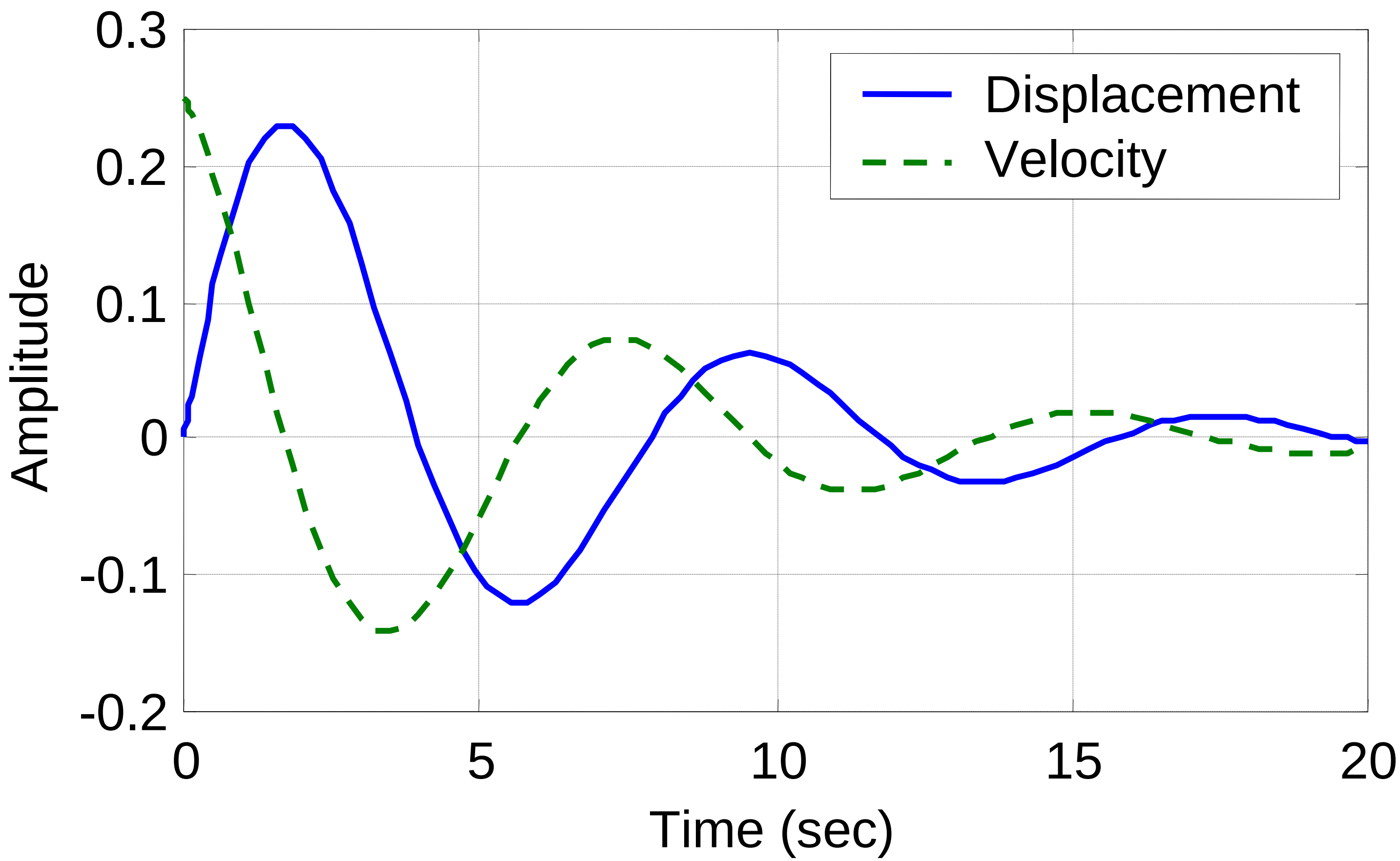
```
function xdot=sdof(t,x)
k=2;c=1;m=3;
A=[0 1;-k/m -c/m];
xdot=A*x;
```

Saved as sdof.m

In the command window

```
» t0=0;tf=20;
» x0=[0 ; 0.25];
» [t,x]=ode45('sdof',[t0 tf],x0);
» plot(t,x)
```

solution



0.3
0.2
0.1
0
-0.1
-0.2
0
5
10
15
20
Amplitude
Time (sec)
Displacement
Velocity

Simulation en excitation forcée

Réécriture de l'EDO:

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = F_0 \cos \omega t$$

$$\ddot{x}(t) + 2\zeta\omega_n \dot{x}(t) + \omega_n^2 x(t) = f_0 \cos \omega t$$

Ecriture des vecteurs d'états

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -2\zeta\omega_n x_2 - \omega_n^2 x_1 + f_0 \cos \omega t$$

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + \mathbf{f}(t), \quad \mathbf{f}(t) = \begin{bmatrix} 0 \\ f_0 \cos \omega t \end{bmatrix}$$

Résolution par intégration numérique

$$\text{Euler:}\ \ \mathbf{x}(t_{i+1}) = \mathbf{x}(t_i) + A\mathbf{x}(t_i)\Delta t + \mathbf{f}(t_i)\Delta t$$

Utiliser ODE45 function

```
>>TSPAN=[0 10];
>>Y0=[0;0];
>>[t,y] =ode45('num_for',TSPAN,Y0);
>>plot(t,y(:,1))
```

En incluant l'excitation

```
function Xdot=num_for(t,X)
m=100;k=1000;c=25;
ze=c/(2*sqrt(k*m));
wn=sqrt(k/m);
w=2.5;F=1000;f=F/m;
f=[0 ;f*cos(w*t)];
A=[0 1;-wn*wn -2*ze*wn];
Xdot=A*X+f;
```

Conditions initiales nulles