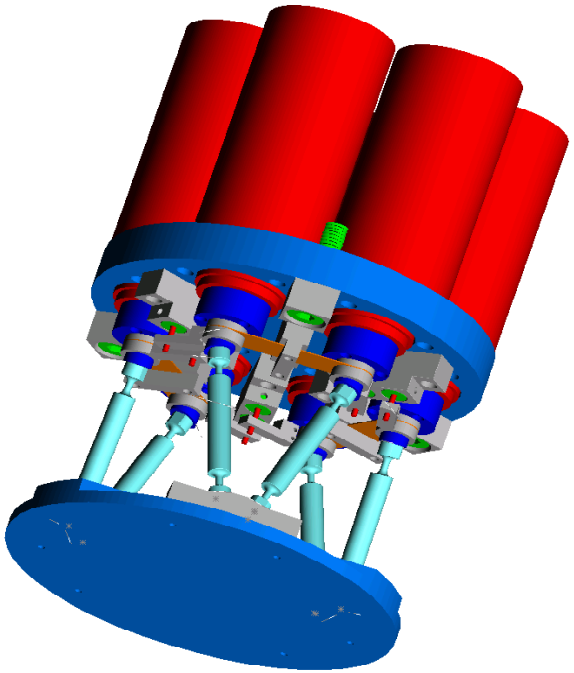
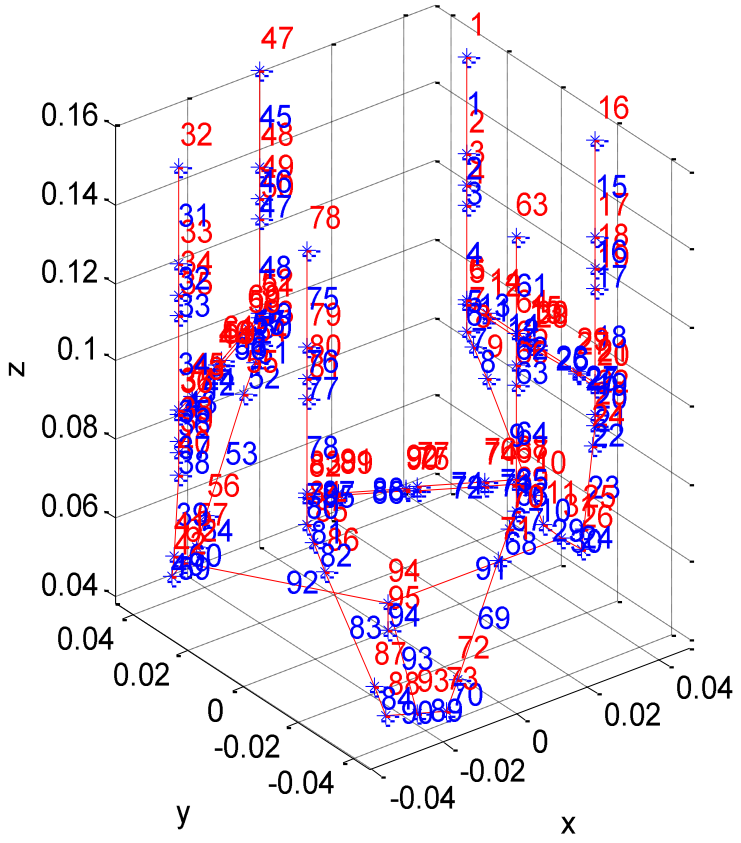
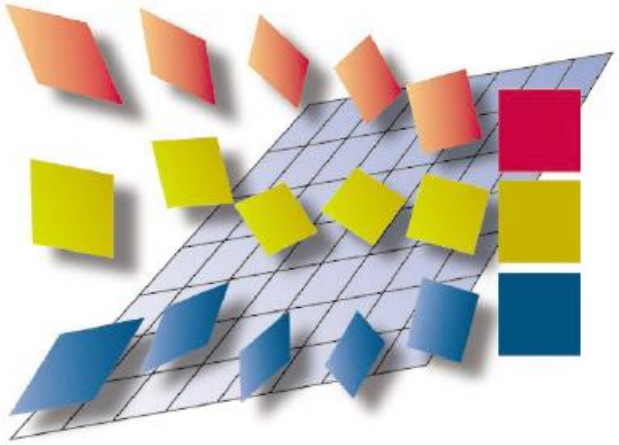
Eléments finis et réponse dynamique des structures avec Matlab





Il est possible de modéliser et calculer un système mécanique complexe de manière très fiable déjà par des modèles à éléments finis relativement simples: ici un hexapode pour un positionnement de précision en 6 axes modélisé entièrement par des éléments de type poutre-3D.

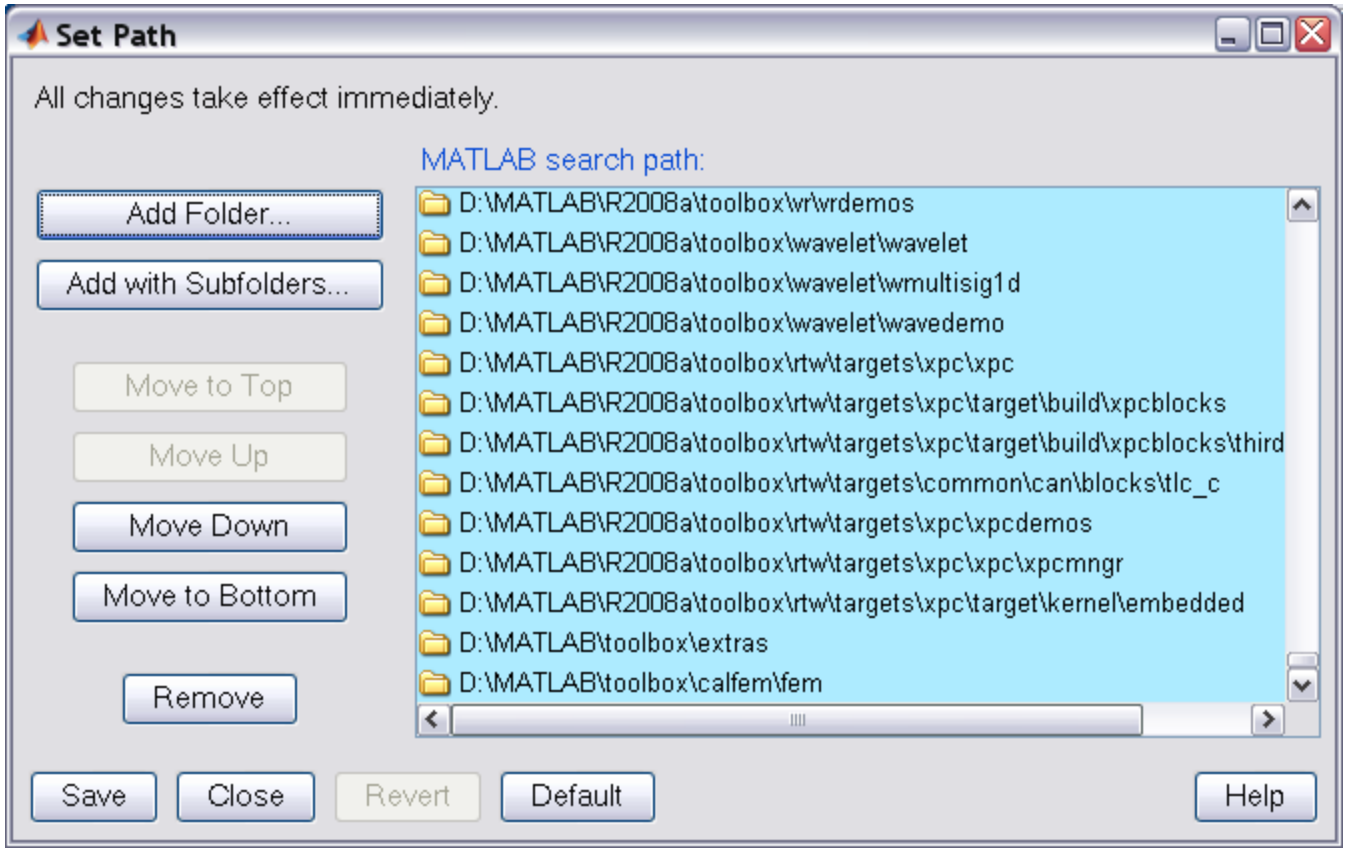
Toolbox à télécharger



Extras: toolbox Matlab «maison» avec plusieurs fonctions utiles et extensions pour CalFem et la simulation de la réponse dynamique.

Insérer les répertoires des toolbox dans le *path*

Avec la commande *addpath* ou (encore plus facile) l'utilitaire *pathtool*:



Organisation du programme

Il est important de bien organiser les diverses étapes du calcul, que l'on retrouvera dans chaque modèle, quelque soit la complexité:

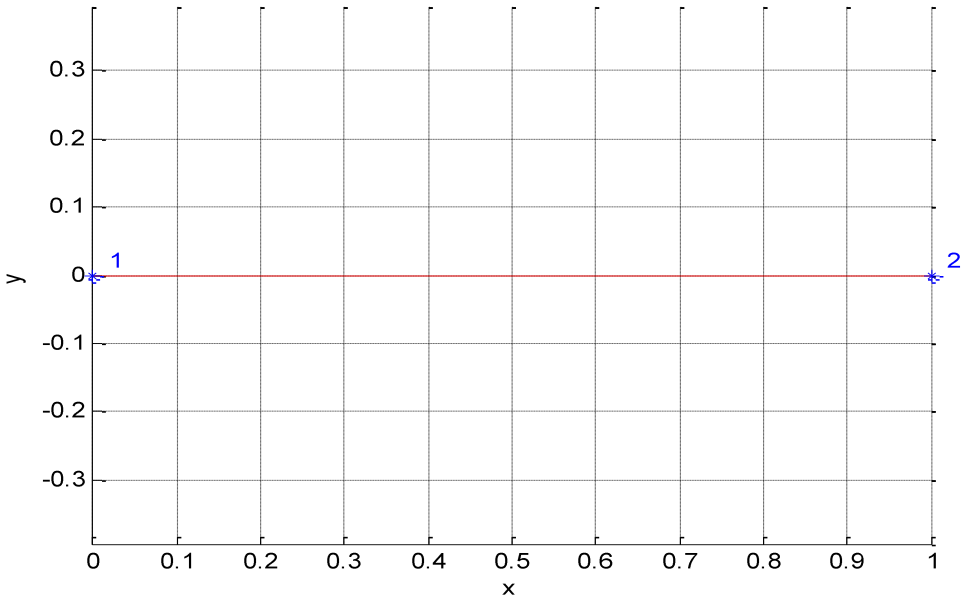
1. Dimensions, matériaux, tout autre paramètres du projet
2. Géométrie
3. Formulation de la topologie du modèle à éléments finis
4. Calcul des matrices [K] et [M]
5. Calculs statiques et des modes propres
6. Formulation du système dynamique (en général espace d'états)
7. Simulation dynamique (→ Simulink)

Selon les exigences on pourra avoir aussi d'autres types de calculs dans la séquence: paramètres électromagnétiques, thermiques, cinématique, etc.

Commençons par un premier modèle hyper-simple: Une poutre à 1 élément

On veut

1. calculer les modes d'une poutre encastrée,
2. formuler la fonction de transfert à l'extrémité libre,
3. réguler par un actionneur la position de cette extrémité



Définitions et données du modèle

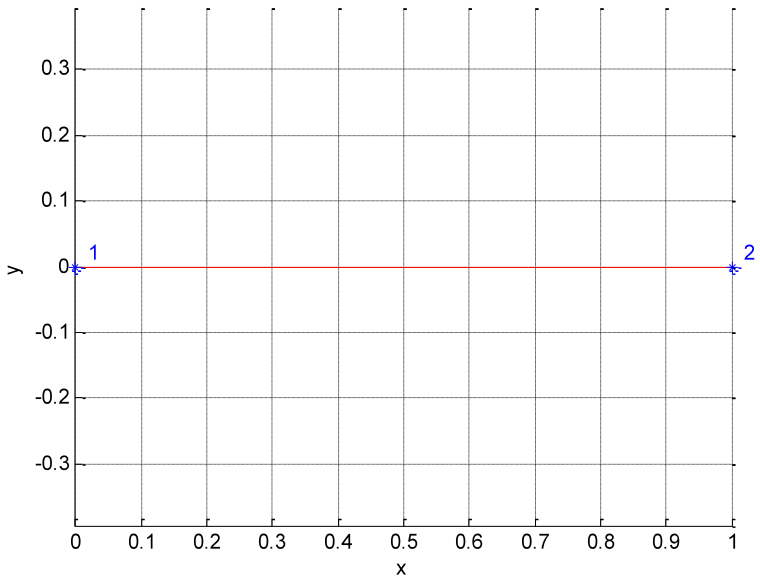
```
clear
close all
% Materiaux: acier
  pois = 0.3;         rho_st = 7800.;
  E_st = 210000e6;    G_st = E_st / (2*(1+pois));

% Géometrie
  L = 1;         % longueur de la poutre
  % Mu = 1;     % masse au bout de la poutre

% Proprietés des éléments
  A = 7.58e-4;
  Iy = 6.29e-8;  Iz = 77.8e-8;  J = Iy+Iz;
  Ep_poutre = [E_st  G_st  A Iy Iz J A*rho_st];

% geometrie du modèle
  Coord (1,:) = [0  0  0];
  Coord (2,:) = [ L 0 0 ];
  Elem(1,:) = [ 1 2 ];
  ePoutre = 1;
  n_fix = 1;  n_free = 2;
```

Topologie du modèle

```
% Topologie du modèle
  n_nodes = 2;
  n_dof = 6;
  n_elem = 1;
  Dof = [ 1     2     3     4     5     6;
          7     8     9    10    11    12 ];
  Edof = ...
  [1     1     2     3     4     5     6     7     8     9    10    11    12];
  Ex = [0 1];
  Ey = [0 0];
  Ez = [0 0];

  % [n_nodes,n_dof,n_elem,n_nel,Dof,Edof] = topol (Coord,Elem);
  % [Ex,Ey,Ez] = coordxtr(Edof,Coord,Dof,n_nel);
```

Graphique du modèle

```
% Plot modèle
  Coord_xy(:,1) = Coord(:,1);
  Coord_xy(:,2) = Coord(:,2);
  figure; femdraw2 (Coord_xy,Ex,Ey);
  ylabel('y'); grid;
```



Formation des matrices K et M

```
%  Matrices K et M
  nd = n_nodes*n_dof;
  K = zeros(nd); M = zeros(nd); C = zeros(nd);

  ie = 1;
  eo(ie,:) = [0 0 1];
  [ke,me] = beam3d (Ex(ie,:),Ey(ie,:),Ez(ie,:),eo(ie,:),Ep_poutre);

  K = assem(Edof(ie,:),K,ke);
  M = assem(Edof(ie,:),M,me);
```

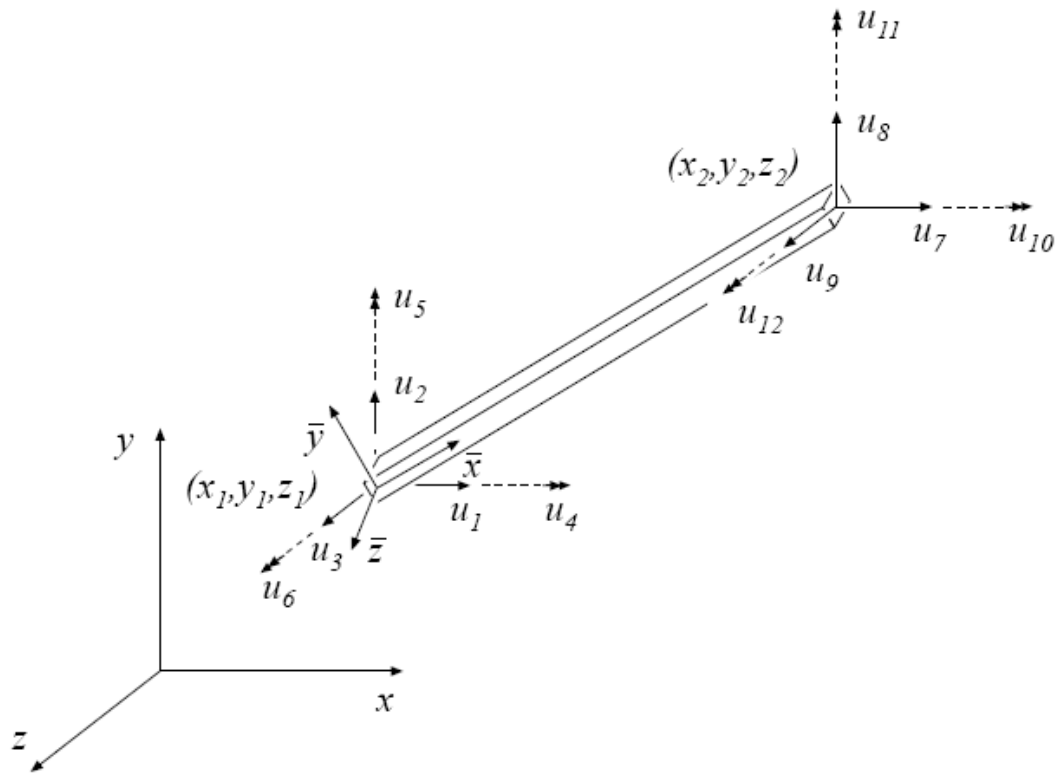
beam3d(): fonction Extras
(identique à beam3e de CelFem plus matrice M – voir doc).

assem(): fonction CalFem –voir doc.

beam3d () – même paramètres comme beam3e

Purpose:

Compute element stiffness matrix for a three dimensional beam element.



Syntax:

Ke=beam3e(ex,ey,ez,eo,ep)

beam3d ()

Description:

beam3e provides the global element stiffness matrix Ke for a three dimensional beam element.

The input variables

$$\mathsf{ex} = [\ x_1 \quad x_2\ ]$$
$$\mathsf{ey} = [\ y_1 \quad y_2\ ] \qquad \mathsf{eo} = [\ x_{\bar{z}} \quad y_{\bar{z}} \quad z_{\bar{z}}\ ]$$
$$\mathsf{ez} = [\ z_1 \quad z_2\ ]$$

supply the element nodal coordinates x_1, y_1, etc. as well as the direction of the local beam coordinate system $(\bar{x}, \bar{y}, \bar{z})$. By giving a global vector $(x_{\bar{z}}, y_{\bar{z}}, z_{\bar{z}})$ parallel with the positive local $\bar{z}$ axis of the beam, the local beam coordinate system is defined. The variable

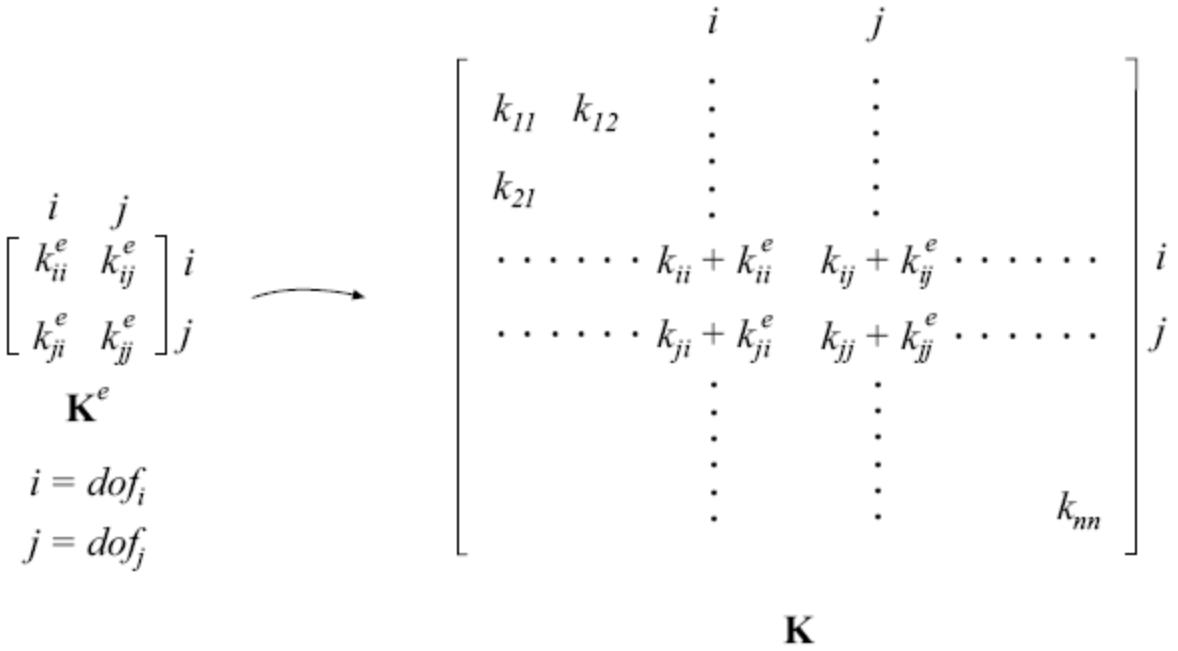
$$\mathsf{ep} = [\ E \quad G \quad A \quad I_{\bar{y}} \quad I_{\bar{z}} \quad K_v\ ]$$

supplies the modulus of elasticity E, the shear modulus G, the cross section area A, the moment of inertia with respect to the $\bar{y}$ axis I_y, the moment of inertia with respect to the $\bar{z}$ axis I_z, and St Venant torsional stiffness K_v.

assem ()

Purpose:

Assemble element matrices.



Syntax:

K=assem(edof,K,Ke)

assem ()

Description:

assem adds the element stiffness matrix $\mathbf{K}^e$, stored in Ke, to the structure stiffness matrix $\mathbf{K}$, stored in K, according to the topology matrix edof.

The element topology matrix edof is defined as

$$\mathsf{edof} = [el \quad \underbrace{dof_1 \quad dof_2 \quad \ldots \quad dof_{ned}}_{\text{global dof.}}]$$

where the first column contains the element number, and the columns 2 to $(ned + 1)$ contain the corresponding global degrees of freedom (ned = number of element degrees of freedom).

In the case where the matrix $\mathbf{K}^e$ is identical for several elements, assembling of these can be carried out simultaneously. Each row in Edof then represents one element, i.e. nel is the total number of considered elements.

$$\mathsf{Edof} = \left.\begin{bmatrix} el_1 & dof_1 & dof_2 & \ldots & dof_{ned} \\ el_2 & dof_1 & dof_2 & \ldots & dof_{ned} \\ \vdots & \vdots & \vdots & & \vdots \\ el_{nel} & dof_1 & dof_2 & \ldots & dof_{ned} \end{bmatrix}\right\} one\ row\ for\ each\ element$$

Modes et vecteurs propres

```
% Modes et vecteurs propres
% on obtient:  n_modes = nombre de modes calculés
%              freq = fréquences propres (Hz)
%              Egv = vecteurs propres (eigenvectors)

  b = [1:6];   % on fixe le noeud 1: Dof 1 à 6
  [L,Egv] = eigen (K,M,b);
  freq = sqrt(L)/(2*pi)  % fréquence propre en Hz
  n_modes = length (freq)

  for i = 1:length(freq)
    forme_t = reshape (Egv(:,i),n_dof,n_nodes); % extraction de chaque mode
    forme = forme_t';
    forme_free(i,:) = forme(n_free,:);
  end
  forme_free    % afficher les formes modales
```

eigen(): fonction CalFem, calcule les modes propres avec des conditions limites – voir doc.

Paramètres pour la fonction de transfert

Une fonction de transfert est une représentation mathématique de la relation entre les entrée (forces et moments) et les sorties (déplacements) d'un système mécatronique linéaire invariant.

Ici la fonction de transfert sera déterminée par:

- Les fréquences propres
- Les forces (entrées) exprimées dans l'espace modale
- Les déplacements (sorties) exprimées dans l'espace modale
- L'amortissement structurel, ici supposé constant pour tous les modes.

```
% Un input (actionneur) ----------------------------------
  in = zeros(n_nodes*n_dof, 1);
  in(8) = 1;   % DoF dir Y de n_free
  inm = Egv'*in;                              % forces modales

% Un output ----------------------------------------------
  out = zeros(1, n_nodes*n_dof);
  out(8) = 1;   % DoF dir Y de n_free
  outm  = [ out*Egv   zeros(1,n_modes) ];  % déplacements modaux

  freqvec = logspace(0,3,100)'; % de 0 à 1000 Hz selon une échelle logarithmique
  w=2*pi*freqvec;                % vecteur de pulsations en rad/s
  om = 2*pi*freq;           % nos fréquences propres en rad/s
  sda = 0.002;              % amortissement structurel = 1/2*Q
```

Réponse en fréquence et fonction de transfert

La réponse en fréquence est la mesure de la réponse du système à une excitation (ici une force) de fréquence variable (mais d'amplitude constante).

```
xf = nor2xf (om,sda,inm,outm,w);
figure; loglog (freqvec,abs(xf)); grid
title ('Réponse en fréquence pour F = 1 N'); xlabel('Hz')

[a,b,c,d] = nor2ss (om,sda,inm,outm);  % modèle state space
sys = ss(a,b,c,d);
size_of_sys = size(sys)
my_plot_bode (w, sys(1,1),'b','Réponse à l''extrêmité');
```

nor2xf(), nor2ss(), my_plot_bode(): fonction Extras

nor2ss ()

Transformation from normal mode form to the state-space form with ability to truncate modes and introduce static correction modes to account for the low frequency components of the truncated modes.

Synopsis:

```
[a,b,c,d]=nor2ss(OM, GA, PHIB, CPHI)
```

nor2ss creates the state space model associated to the normal mode model composed of:

- OM the modal stiffness which can be replaced by a column vector of modal frequencies FREQ (in rad/s)
- GA the modal damping matrix can be replaced by a column vector of modal damping ratio DAMP or a single damping ratio to be applied to all modes
- PHIB and CPHI the normal mode input and output shape matrices

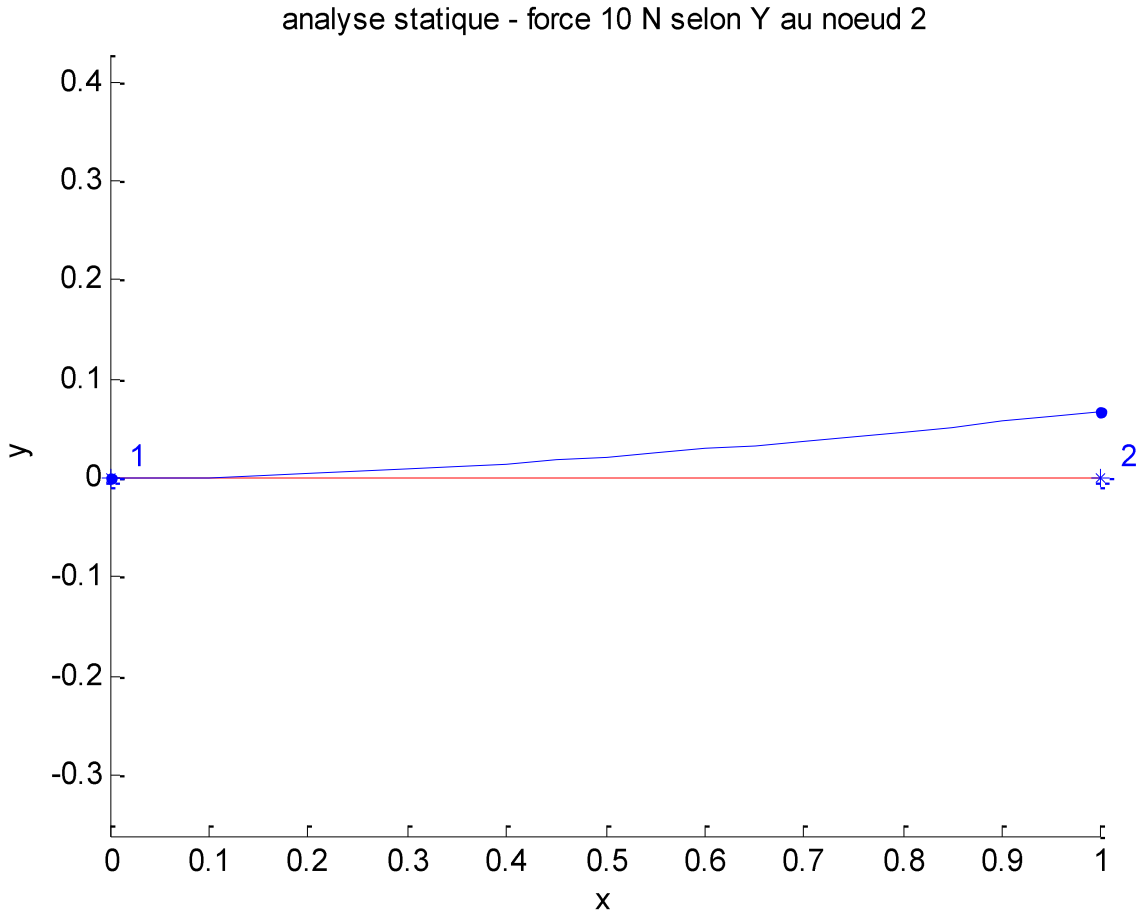
To obtain a velocity output, specify displacement CPHID and velocity CPHIV modal outputs and use *nor2ss* (OM,GA,PHIB,[CPHID CPHIV])

Calcul statique

Une force de 10 N à l'extrémité

```
% Conditions aux limites
bc = [];  [b,bc,nb] = fix_point (bc, n_fix, Dof);

%  On définit les forces et moments
p = zeros(size(K,1),1);
i = n_free;  dof_exc = (i-1)*n_dof+2;
p(dof_exc) = 10;   % N

[X, R, xyzF] = fe_stat (K,p,b,n_dof,n_nodes);

Edb = extract (Edof,X);
figure; femdraw2 ([Coord(:,1) Coord(:,2)],Ex,Ey);
Edbxy = [Edb(:,1) Edb(:,2) Edb(:,6) Edb(:,7) Edb(:,8) Edb(:,12)];
femdisp2 (Ex,Ey,Edbxy);  ylabel('y');
title('Calcul statique - force 1 N selon Y au noeud 2');

disp ('Déplacements des noeuds (mm)');
disp (xyzF*1000)
```

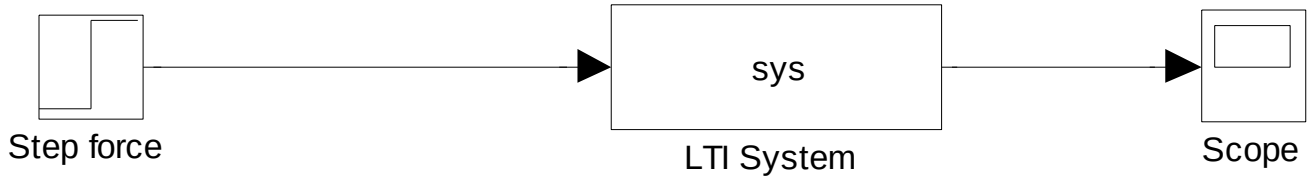
fe_stat(): fonction Extras
femdraw2(), femdisp2() (): fonctions Extras – modifications des fonctions CalFem de même nom.
solveq() – appelée par fe_stat , extract(): fonctions CalFem

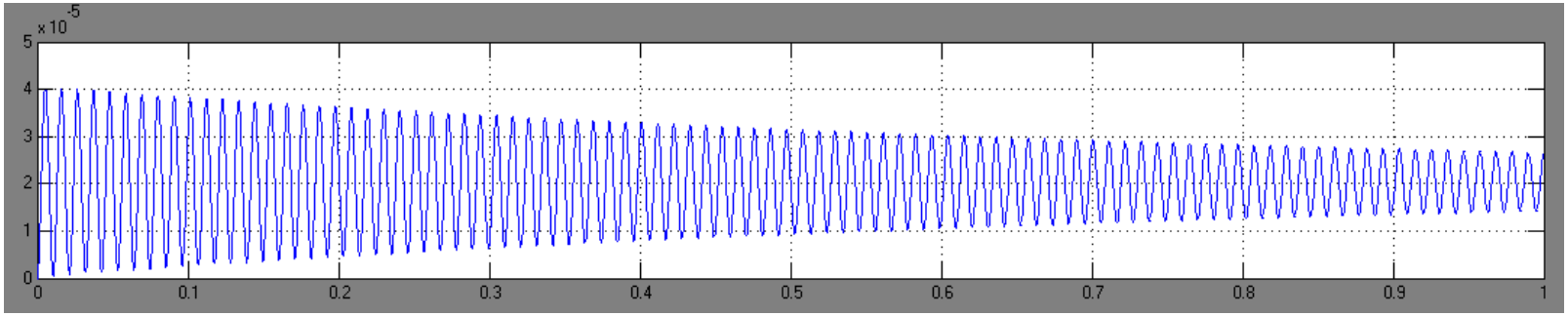


Déplacements des noeuds (mm)

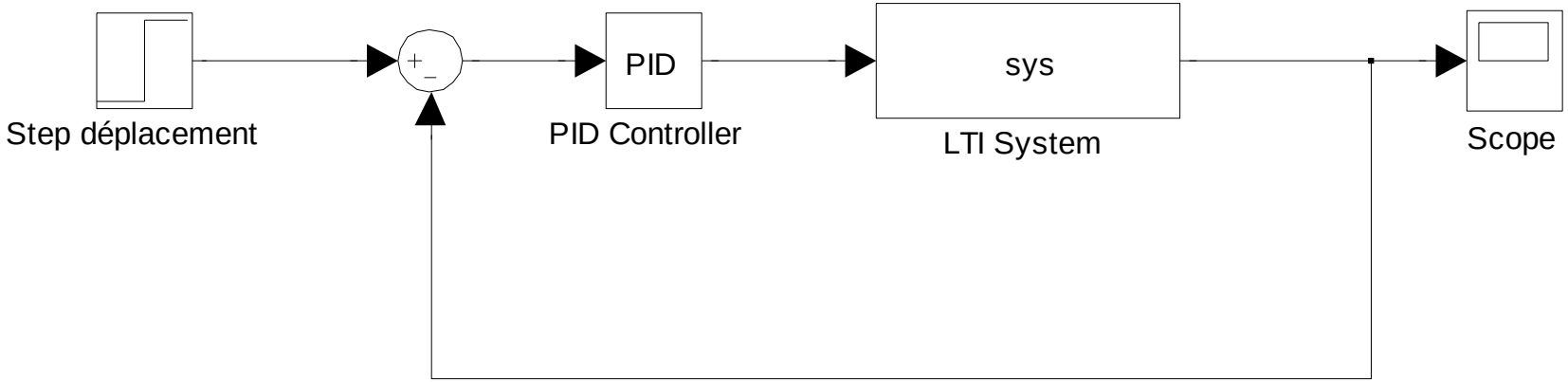
0	0	0	0	0	0
0	0.0204	0	0	0	0.0306

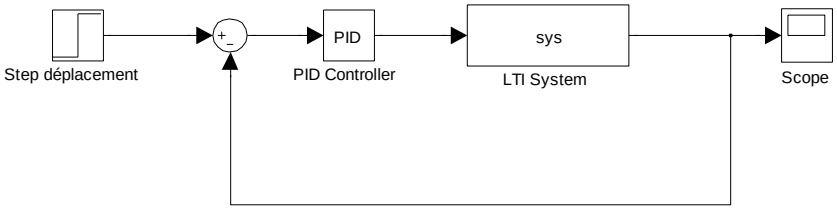
Passage dans Simulink

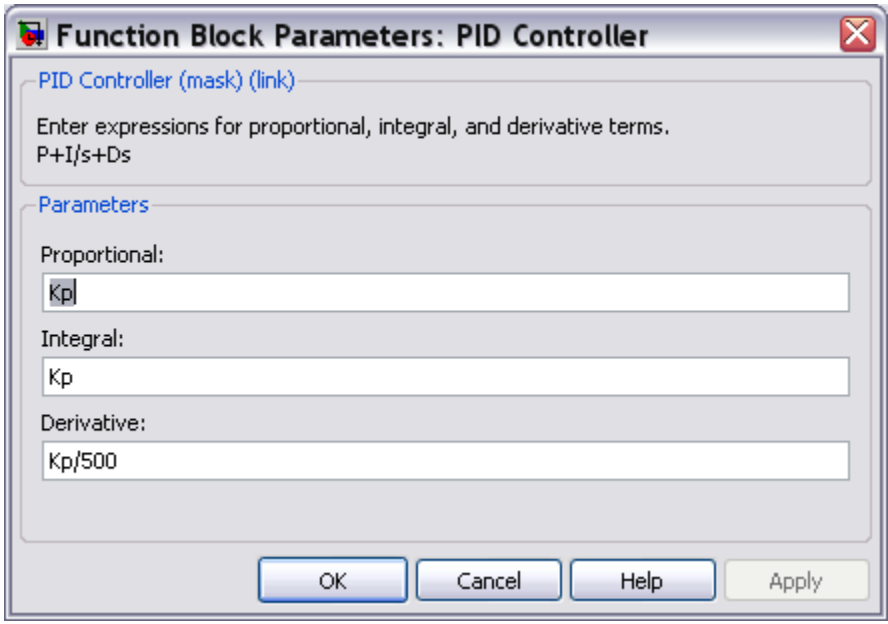




- et avec un réglage actif …

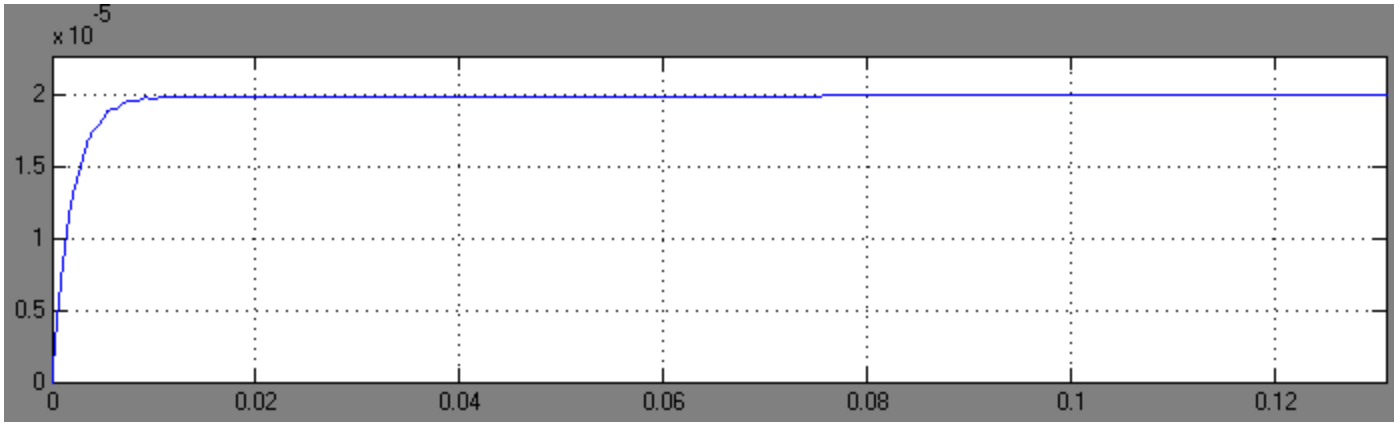






- Par exemple …

  Kp = 20e6;



Variante de calcul

Ajouter une masse et une inertie à l'extrémité de la poutre:

```
for i = 1:3
    ndof = (n_free-1)*n_dof+i;
    M(ndof,ndof) = M(ndof,ndof) + Mu;
end
for i = 4:6
    ndof = (n_free-1)*n_dof+i;
    M(ndof,ndof) = M(ndof,ndof) + Iu;
end
```

Variante de calcul

Modéliser la poutre avec plus d'éléments (ici `ne`):

```
% geometrie du modèle
  ne = 5;
  Coord (1,:) = [0  0  0];
  for i = 1:ne
    Coord (1+i,:) = [ Coord(i,1)+L/ne  0  0 ];
    Elem(i,:) = [ i  i+1 ];
  end
  ePoutre = [1:ne];
  n_fix = 1;  n_free = ne+1;

% Topologie du modèle
  [n_nodes,n_dof,n_elem,n_nel,Dof,Edof] = topol (Coord,Elem);
  [Ex,Ey,Ez] = coordxtr(Edof,Coord,Dof,n_nel); (1,:) = [0  0  0];

                                          ...

for ie = 1:ne;
    eo(ie,:) = [0 0 1];
    [ke,me] = beam3d (Ex(ie,:),Ey(ie,:),Ez(ie,:),eo(ie,:),Ep_poutre);
    K = assem(Edof(ie,:),K,ke);
    M = assem(Edof(ie,:),M,me);
  end
```