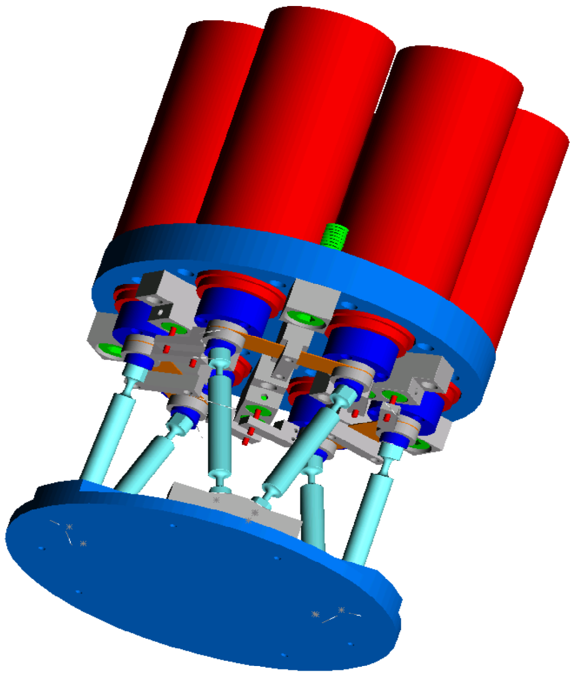
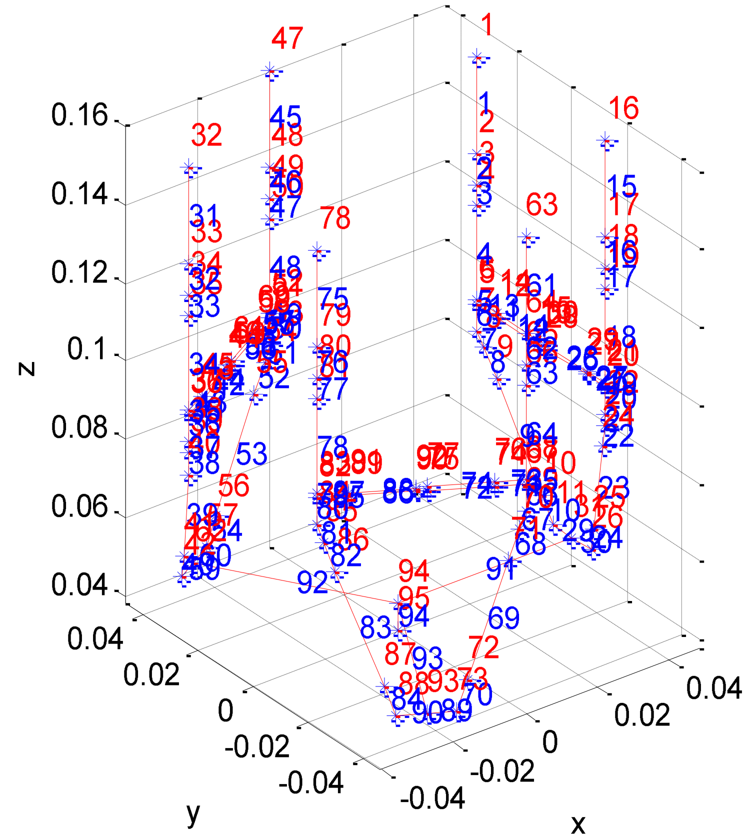
Finite elements analysis and dynamic réponse of a simple beam in Matlab





It is possible to model and compute a complex mechanical system very reliably already using a simple finite element model.

Here a hexapode designed for precision positioning in 6 axes is modeled entireley by 3D-beam elements.

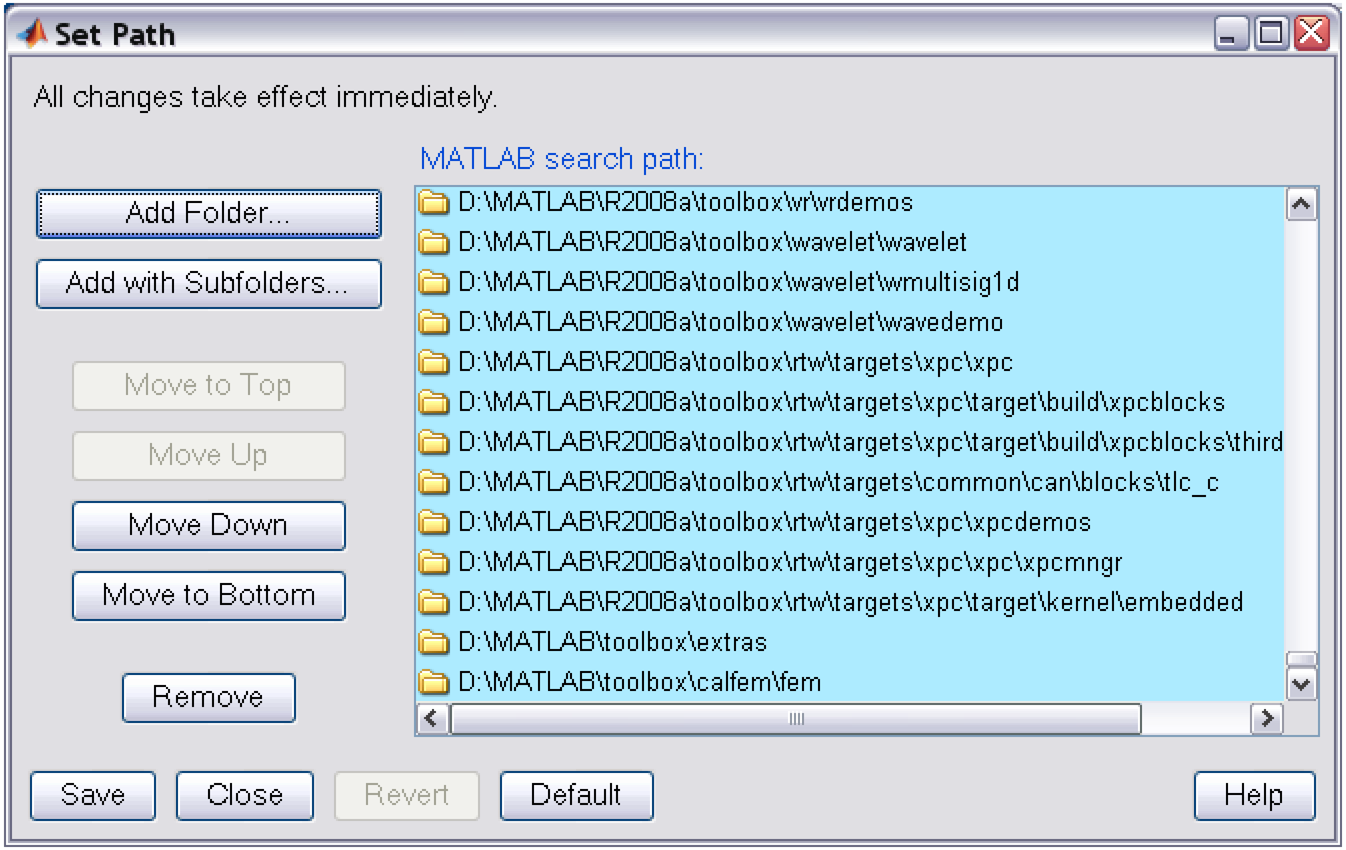
Some FEM toolboxes for Matlab

CALFEM
A FINITE ELEMENT TOOLBOX
Version 3.4

Extras: «home-made» Matlab toolbox «maison» with various useful functions and extensions for CalFem and dynamic simulation.

Adding toolboxes in the *path*

With *addpath* or (even easier) the *pathtool* utlity:



Program organization

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

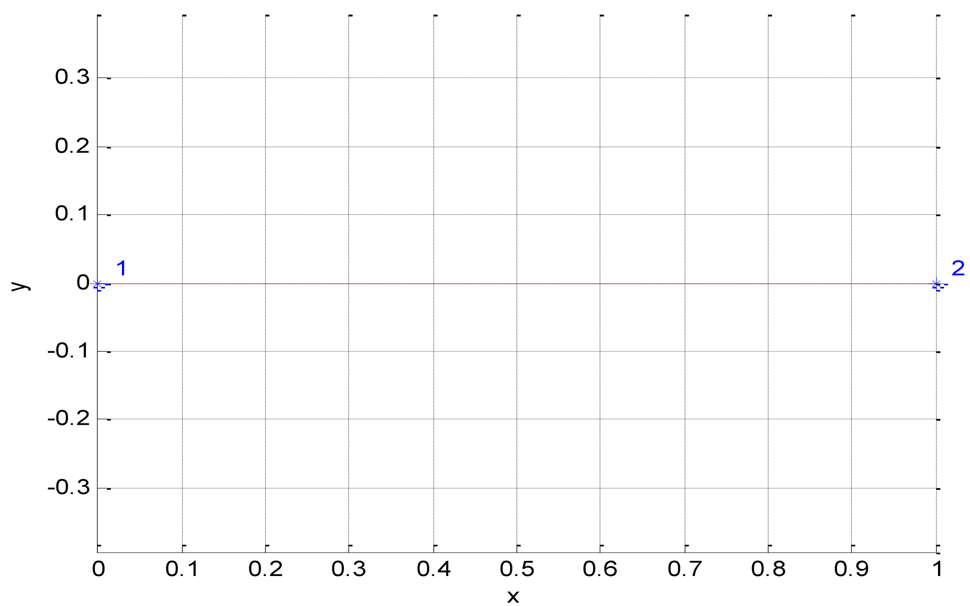
1. Dimensions, matériaux, tout autre paramètres du projet
2. Géométrie
3. Formulation de la topologie du modèle à éléments finis
4. Calcul des matrices [K] et [M]
5. Calculs statiques et des modes propres
6. Formulation du système dynamique (en général espace d'états)
7. Simulation dynamique (→ Simulink)

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

Let’s start with an ultra-simple model: A 1-element beam

We wish to

1. Compute the modes of a cantilever beam,
2. Formulate the transfer function at the free end,
3. Control actively by an actuator the position of this end.



Définitions and data of the model

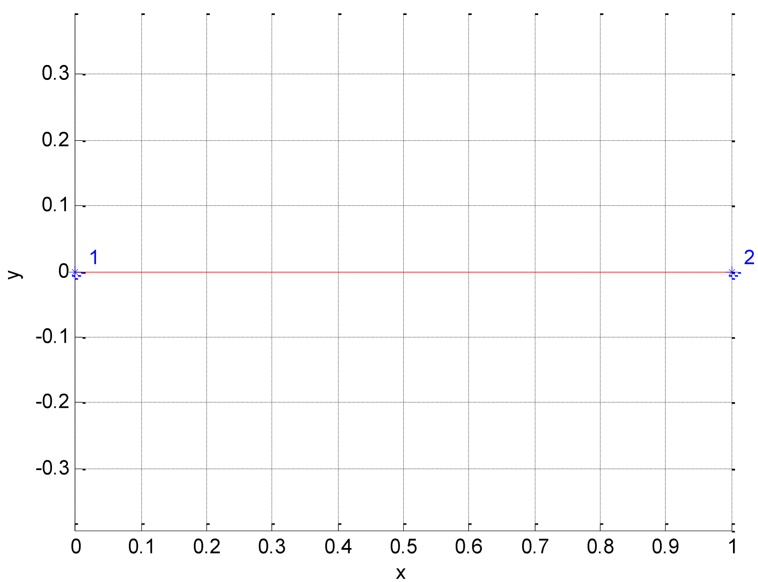
```
clear
close all
% Material: steel
  pois = 0.3;         rho_st = 7800.;
  E_st = 210000e6;    G_st = E_st / (2*(1+pois));

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

% Element properties
  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];

% geometry
  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);
```

Plot of the model

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



K and M matrices

```
% 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);
```

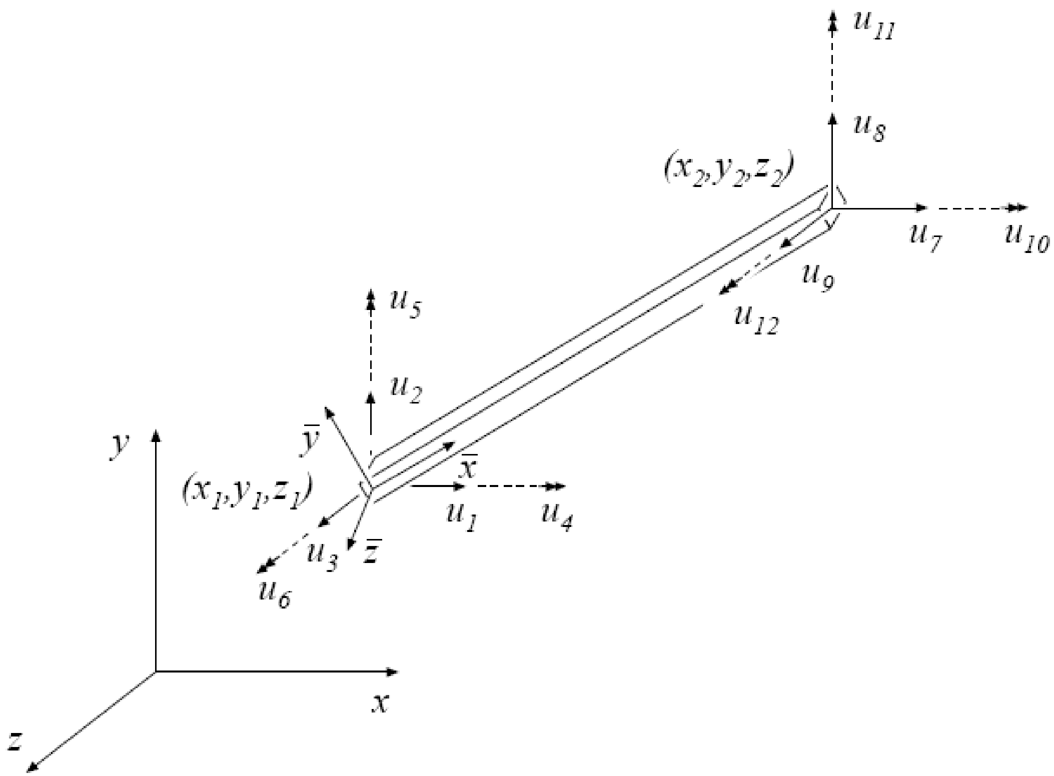
beam3d(): function in Extras
(identical to **beam3e** of CalFem plus matrix M).

assem(): function CalFem.

beam3d () – same parameters as 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

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

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

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

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.

$$
\begin{array}{c}
\begin{array}{cc} i & j \end{array} \\
\begin{array}{c}
\begin{bmatrix} k_{ii}^e & k_{ij}^e \\ k_{ji}^e & k_{jj}^e \end{bmatrix}
\end{array}
\begin{array}{c} i \\ j \end{array} \\
\mathbf{K}^e
\end{array}
\longrightarrow
\begin{array}{c}
\begin{array}{ccccccc} & & & i & j & & \end{array} \\
\begin{bmatrix}
k_{11} & k_{12} & & \vdots & \vdots & & \\
k_{21} & & & \vdots & \vdots & & \\
\cdots & \cdots & \cdots & k_{ii}+k_{ii}^e & k_{ij}+k_{ij}^e & \cdots & \cdots \\
\cdots & \cdots & \cdots & k_{ji}+k_{ji}^e & k_{jj}+k_{jj}^e & \cdots & \cdots \\
 & & & \vdots & \vdots & & \\
 & & & \vdots & \vdots & & k_{nn}
\end{bmatrix}
\begin{array}{c} \\ \\ i \\ j \\ \\ \\ \end{array} \\
\mathbf{K}
\end{array}
$$

$i = dof_i$

$j = dof_j$

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

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

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.

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

Eigenmodes et vectors

```
% Eigenmodes and eigenvectors
% we obtain:   n_modes = nomber of computed modes
%              freq = eigenfrequencies (Hz)
%              Egv = eigenvectors

  b = [1:6];   % fixing node 1: Dof 1 to 6
  [L,Egv] = eigen (K,M,b);
  freq = sqrt(L)/(2*pi)  % eigenfrequency in Hz
  n_modes = length (freq)

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

eigen(): function CalFem, computes eigenmodes with given boundary conditions.

Parameters for the transfer function

A transfer function is a mathematical representation of the relation bewteen inputs (forces and moments) and outputs (displacements) of a **linear invariant** mechatronic system.

Here the transfer function will be determined by:

- The eigenfrequencies
- The forces (inputs) expressed in the **modal space**
- The displacements (outputs) expressed in the **modal space**
- The modal structural damping, here assumed constant for all modes.

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

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

  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;           % eigenfrequencies in rad/s
  sda = 0.002;              % structural damping = 1/2*Q
```

Frequency response and transfer function

The frequency response is the qunatification of the system response to an excitation (here a force) of varying frequency (but constant amplitude).

```
xf = nor2xf (om,sda,inm,outm,w);
figure; loglog (freqvec,abs(xf)); grid
title ('Response in frequency for 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','Respose at the end of the beam');
```

nor2xf(), nor2ss(), my_plot_bode(): functions from Extras toolbox

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:

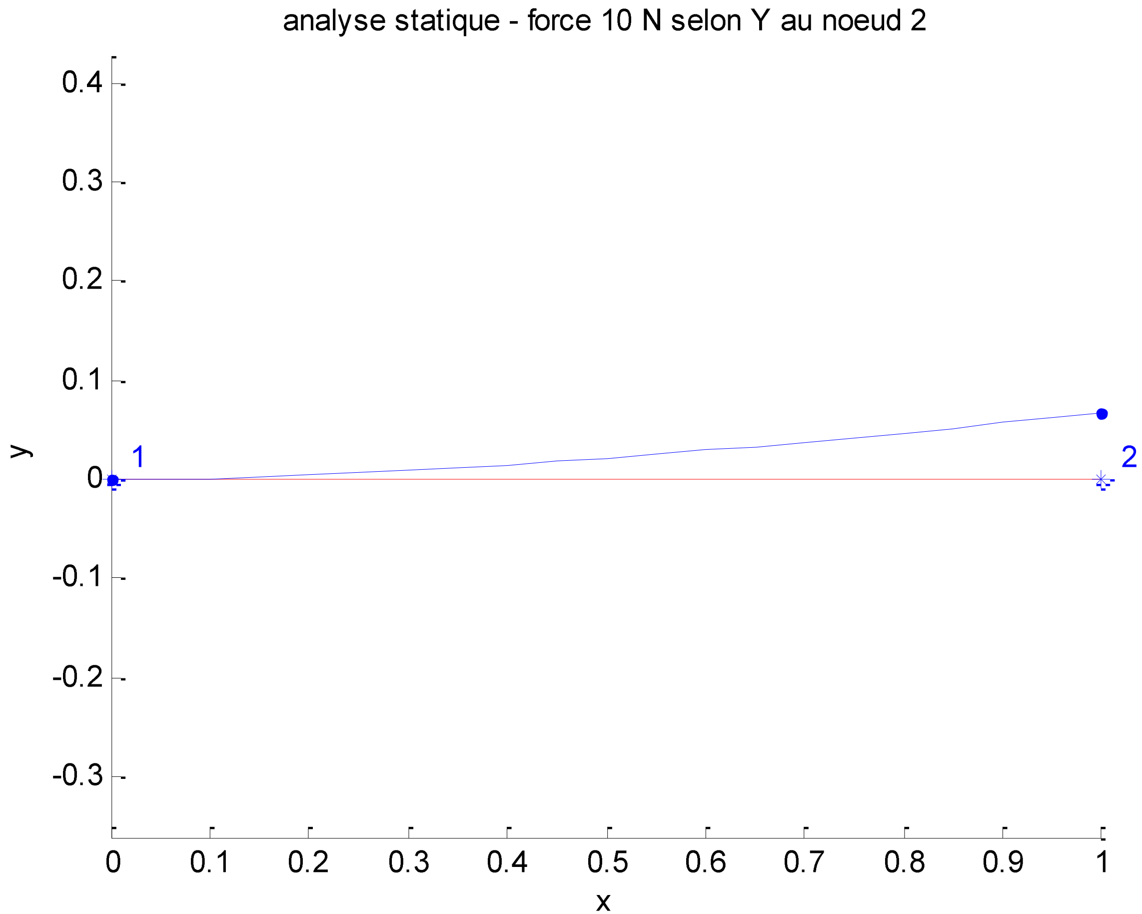
- OM the modal stiffness which can be replaced by a column vector of modal frequencies FREQ (in rad/s)
- GA the modal damping matrix can be replaced by a column vector of modal damping ratio DAMP or a single damping ratio to be applied to all modes
- PHIB and CPHI the normal mode input and output shape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

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

Static computation - a 10 N force at the end of the beam

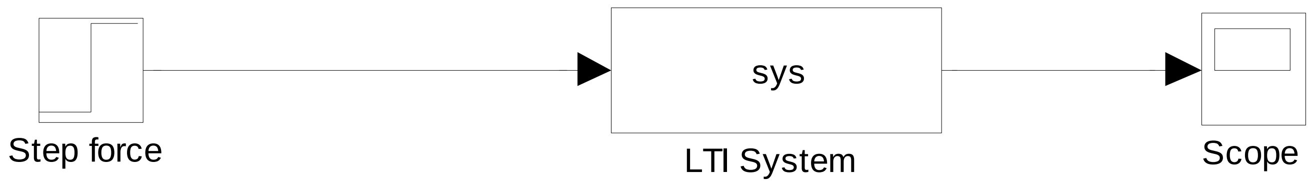
```
% Boundary conditions
bc = [];  [b,bc,nb] = fix_point (bc, n_fix, Dof);

%  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)
```

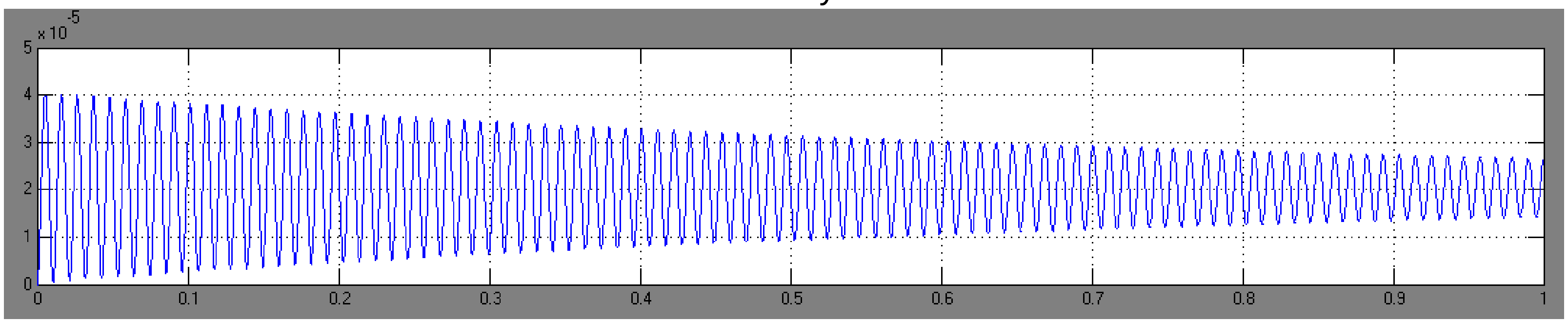
fe_stat(): function Extras
femdraw2(), femdisp2() (): functions Extras – modifications of functions CalFem of same name.
solveq() – called by fe_stat , extract(): functions CalFem



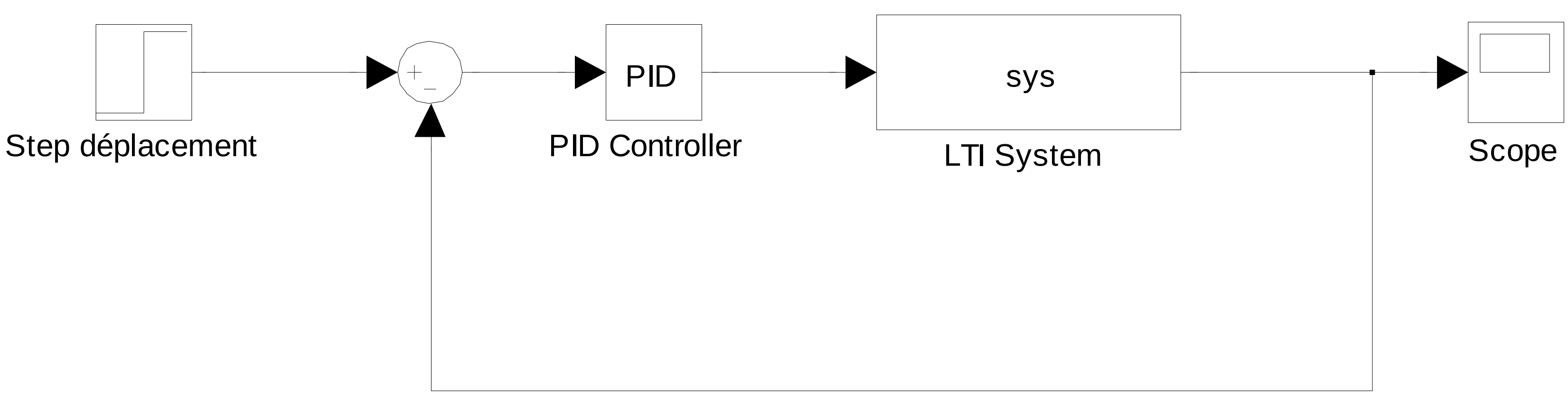

```
Node displacement(mm)
         0         0         0         0         0         0
         0    0.0204         0         0         0    0.0306
```

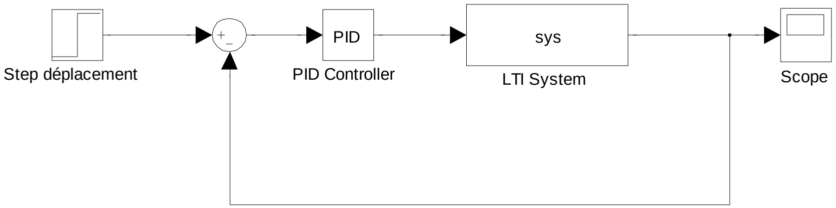
Transfer into Simulink

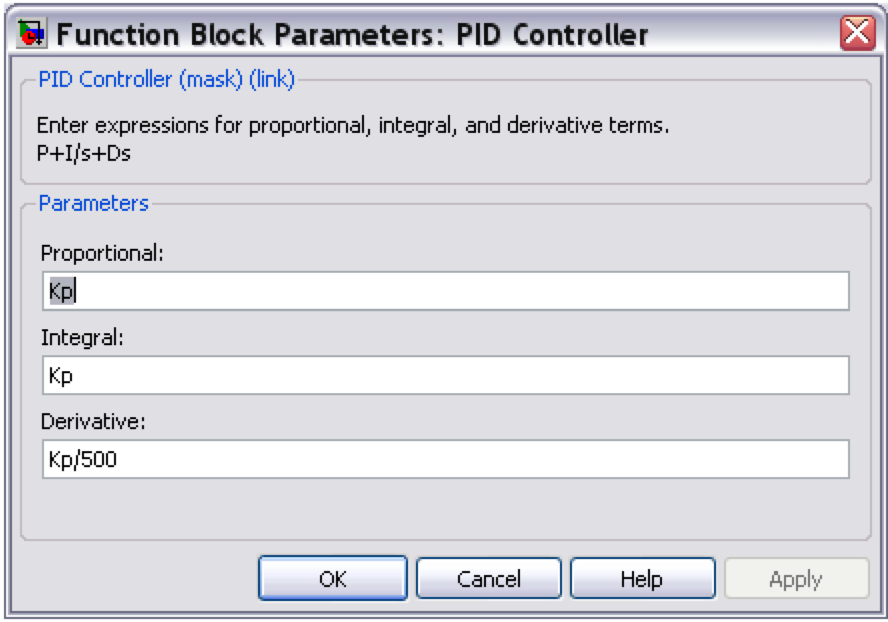




- and with active feedback control …

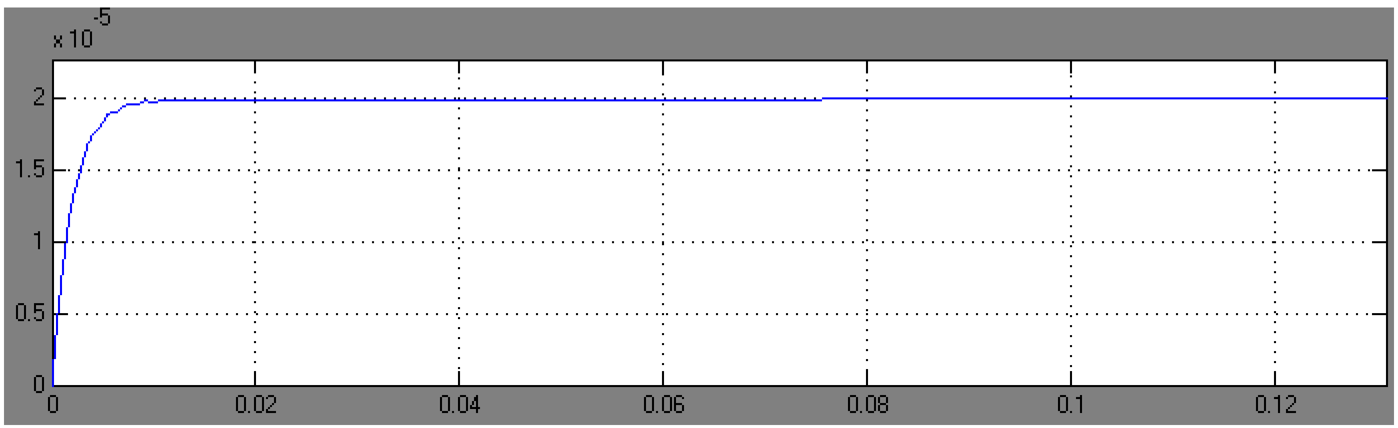






- For instance…

  Kp = 20e6;



Variant

Add a mass and an inertia at the end of the beam:

```
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
```

Variant

Model the beam with more elements (`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
```