Rapid integrated modeling of an active structure

What is

Integrated modélisation (also named *end-to-end modeling)* ?

Purpose:

- We wish to make an integrated model which is able to evaluate in a **single computation cycle** all the relevant aspects of the design being studied:
  - Elastic and modal caracteristics
  - Boundary conditions
  - Moteurs and/or other active systems
  - Feedback and control
  - Any other interesting of relevant aspect: cinematic, thermal, optical, etc.

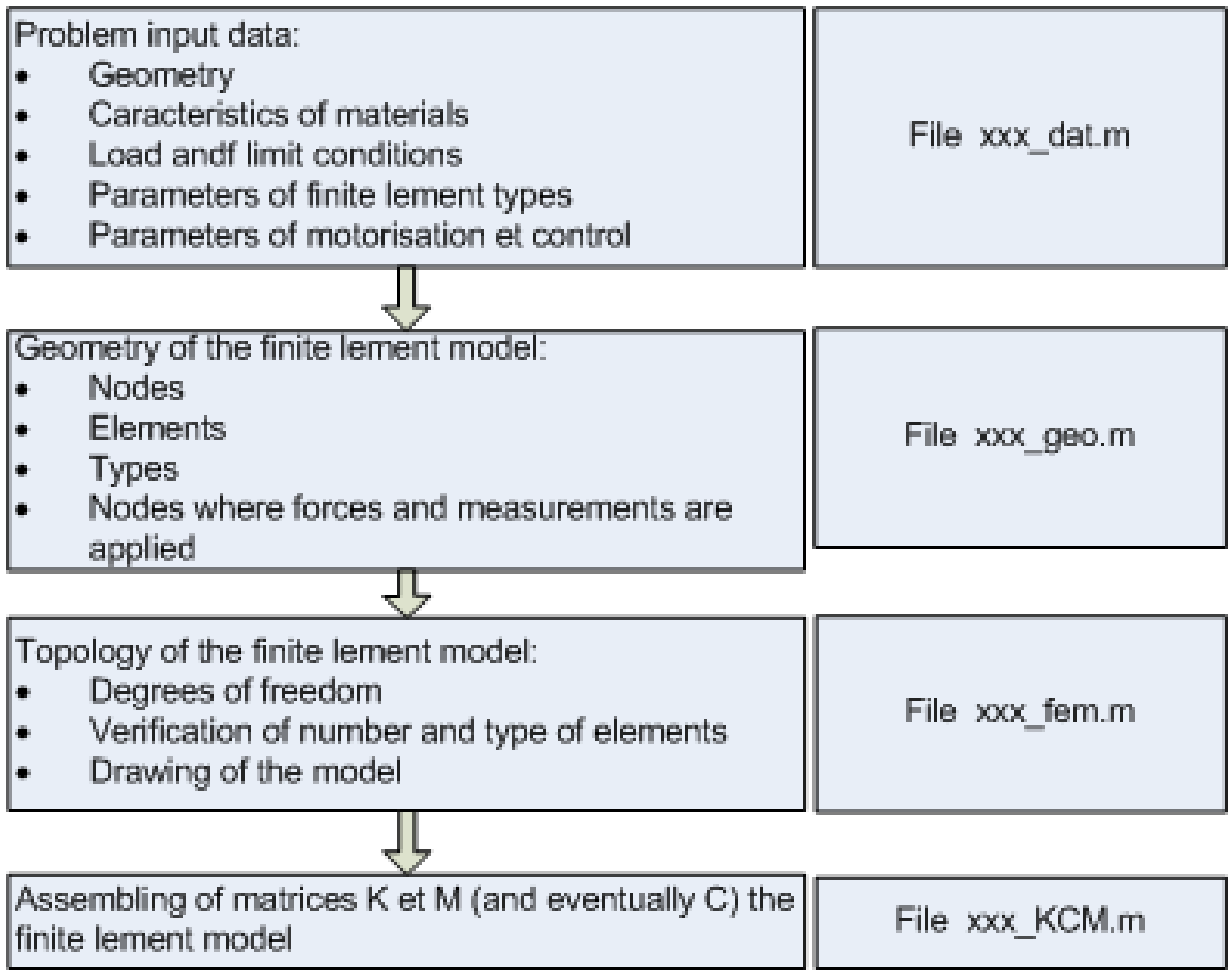
- We wish this model to be **parametric**:
  - When we change one or more input parameters we wish to immdiately compute the results and impacts on all performances:
    - Static deformations, material stresses
    - Eigenmodes and frequencies
    - Response to actuators, environment,
    - etc.

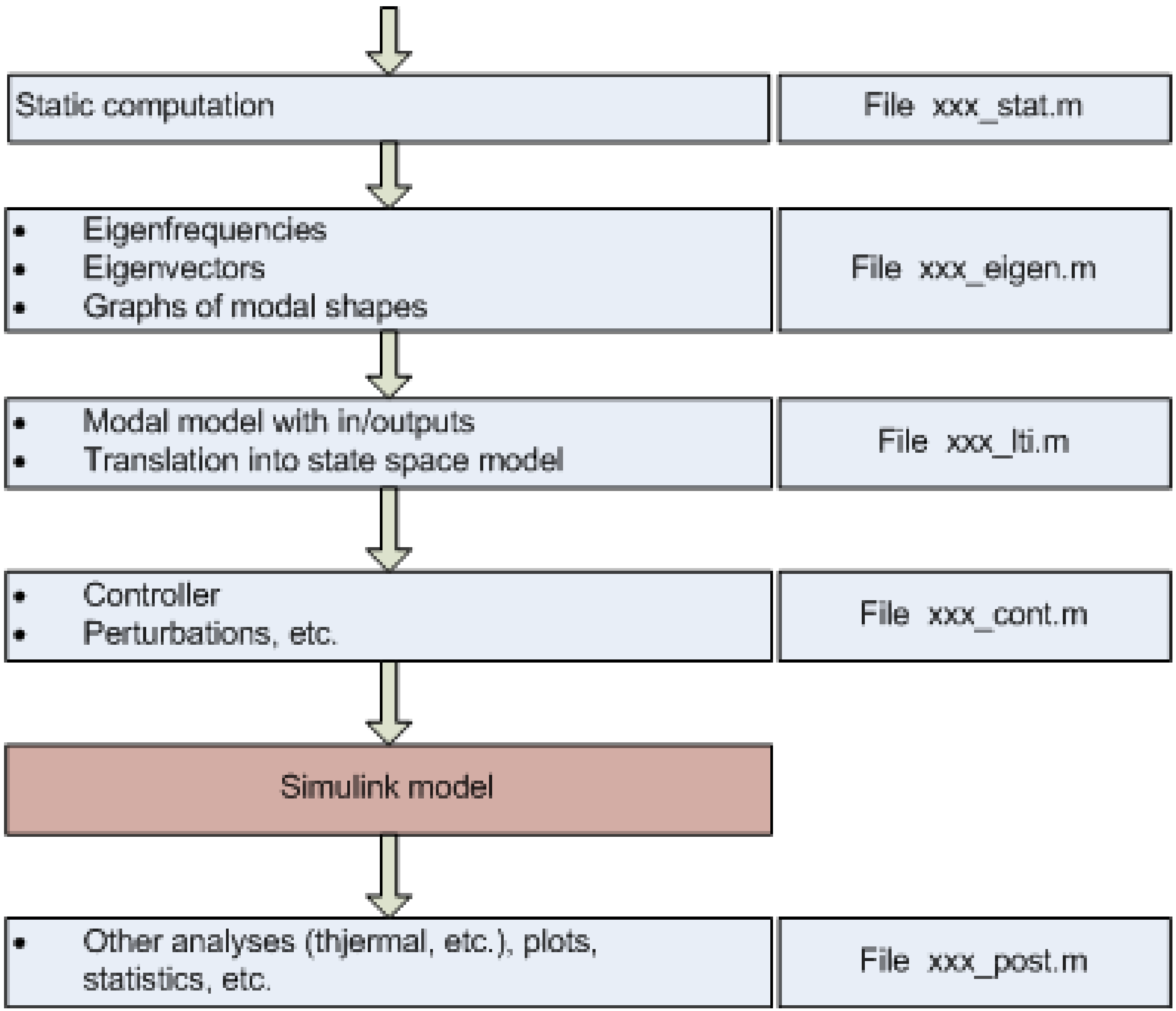
- We wish this model to be made **rapidly**.
  This is very important, for instance
  - to be able to check the feasibility of controversial designs
  - to make a reliable proposal to a customer
  - …

Integrated modeling based on Matlab

- A MAIN script *xxx_main.m* calls the modules dealing with the various aspects of the system:
  - Inputs and parameters
  - Structures
  - Static and dynamic response
  - Mcatronic aspects (actuators, etc.)
  - Feedback loops
  - Other relevant: thermal, optical, …

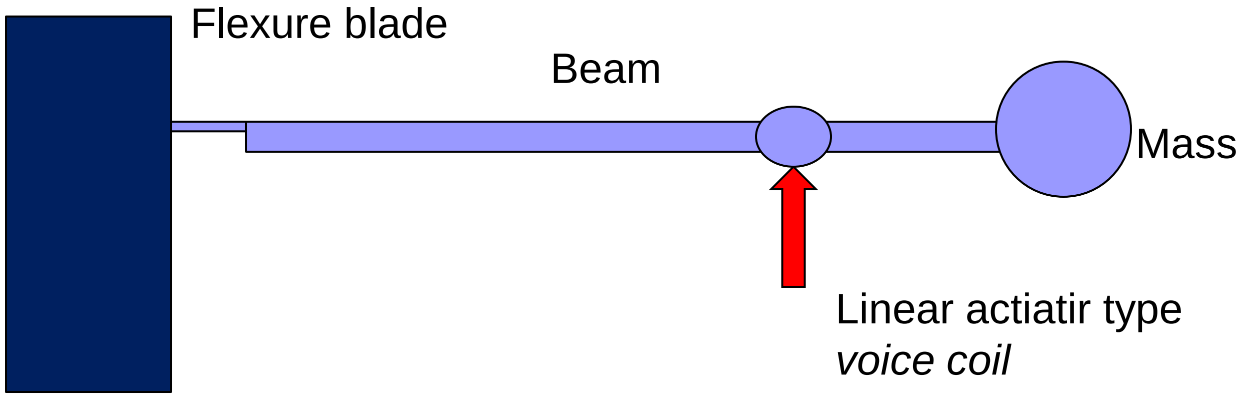
Proposed structure of the model




Static computation
File xxx_stat.m
Eigenfrequencies
Eigenvectors
Graphs of modal shapes
File xxx_eigen.m
Modal model with in/outputs
Translation into state space model
File xxx_lti.m
Controller
Perturbations, etc.
File xxx_cont.m
Simulink model
Other analyses (thjermal, etc.), plots, statistics, etc.
File xxx_post.m

Example:

Motion control of a mass guide by a flexure beam

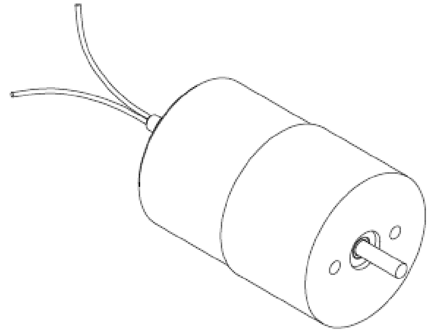


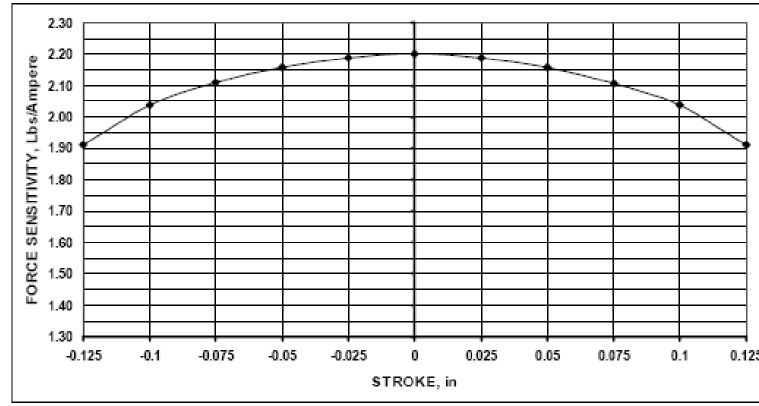
LTR	ECO NO.	DESCRIPTION	DRN	APP'D	DATE
X1	090305	INITIAL RELEASE	RG	MG	01/14/10

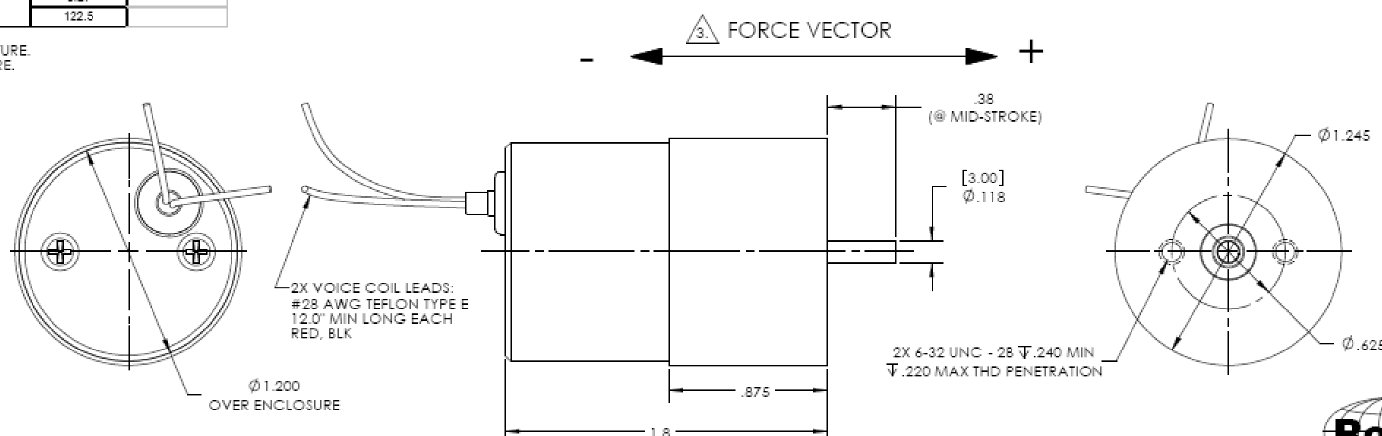
Winding Constants *	Units	Tol	Symbol	Wdg	Z
DC Resistance	Ohms	± 12.5%	R	17.1	
Voltage @ F_P	Volts	Nominal	V_P	27.2	
Current @ F_P	Amps	Nominal	I_P	1.59	
Force Sensitivity	LB/Amp	± 10%	K_F	2.2	
	N/Amp	± 10%		9.79	
Back EMF Constant	V/(ft/sec)	± 10%	K_B	2.98	
	V/(m/sec)	± 10%		9.79	
Inductance ****	milli-Henry	± 15%	L	2.8	

Linear Actuator Parameters *	Units	Symbol	Value
Peak Force **	LB	F_P	3.5
	N		15.57
Continuos Stall Force ***	LB	F_{CS}	1.14
	N		5.07
Actuator Constant	$LB/\sqrt{Watt}$	K_A	0.53
	$N/\sqrt{Watt}$		2.36
Electrical Time Constant	micro-sec	τ_E	164
Mechanical Time Constant	milli-sec	τ_M	2.84
Theoretical Acceleration	ft/sec^2	α_T	3212.6
	m/sec^2		979.2
Max Theoretical Frequency @ Full Stroke and Sinusoidal / Triangular Motion	Hz	f_{max}	88.4/98.2
Power I^2R @ F_P	Watts	P_P	43.3
Stroke:	± in		0.125
	± mm		3.18
Clearance on Each side of Coil	in		0.015
	mm		0.38
Thermal Resistance of Coil in still air	°C/Watt	Θ_{TH}	18.7
Maximum Allowable Coil Winding Temp	°C	Temp	155
Weight of Coil Assembly	LB	WT_C	0.035
	g		15.9
Total Weight	LB	WT_T	0.27
	g		122.5

* AT MID-STROKE POSITION AND @ 25°C AMBIENT TEMPERATURE.
** 10 SECONDS @ 25°C AMBIENT & 155°C COIL TEMPERATURE.
*** @25°C AMBIENT & 155°C COIL TEMPERATURE.
**** MEASURED AT 1000 Hz.







4. MECHINAL OVERTRAVEL IN THE POSITIVE (+) DIRECTION IS .025 MINIMUM.

3. A POSITIVE (+) VOLTAGE APPLIED TO THE RED LEAD WILL PRODUCE A FORCE ON THE COIL ASSEMBLY (SHAFT) IN THE POSITIVE (+) DIRECTION.

2. INTERPRET DRAWING IAW Y14.100.

1. INTERPRET DIMENSIONING AND TOLERANCING IAW ASME Y14.5M-1994.

NOTES: UNLESS OTHERWISE SPECIFIED

THIRD ANGLE PROJECTION

UNLESS OTHERWISE SPECIFIED:
- ALL DIMENSIONS ARE IN INCHES
- BREAK SHARP EDGES .015 MAX
- SURFACE ROUGHNESS 63 ✓
- DIMENSIONS APPLY AFTER FINISH
- MAX FILLET R.010
- DIAMETERS SHALL NOT EXCEED A RUNOUT OF .005 FIM

TOLERANCES:

DECIMALS	ANGULAR
.X ±.03	±0°30'
.XX ±.01	
.XXX ±.005	

DO NOT SCALE DRAWING

BEI KIMCO MAGNETICS DIVISION
VISTA, CA 92081

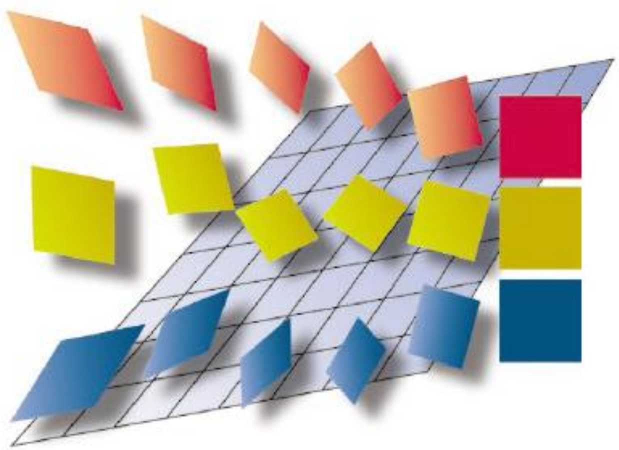
	NAME	DATE
DRAWN	GUERRERO	12/01/09
CHECK	McGHEE	01/14/10
APPD	GODKIN	01/14/10
FILE NO.	L\TOP L\LAH\	

TITLE: LINEAR ACTUATOR HOUSED

SIZE	FSCM NO.	DWG NO.	REV
C	55789	LAH13-18-000A	X1

SCALE: NONE SHEET: 1 OF 1

Some FEM toolboxes for Matlab



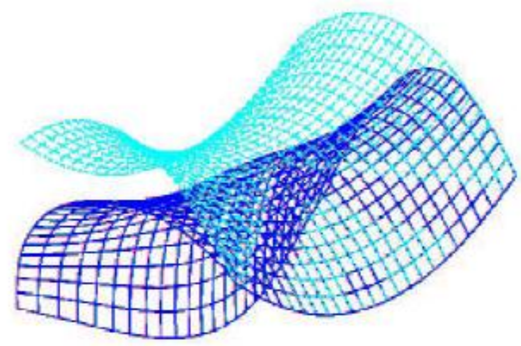
CALFEM

A FINITE ELEMENT TOOLBOX

Version 3.4

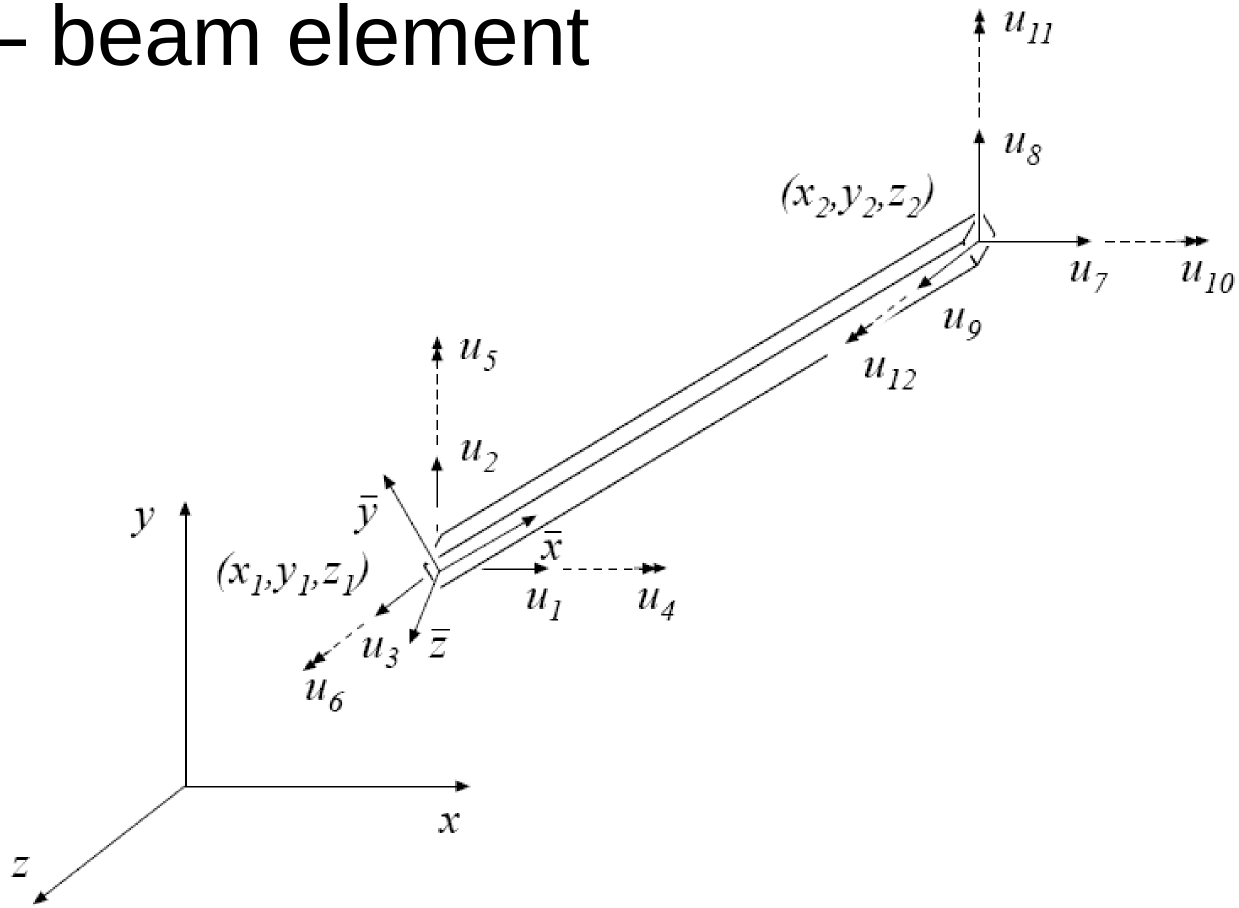
OpenFEM

A finite element toolbox for Matlab and Scilab



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

beam3d – beam element



- Syntax:

```
[ke, me] = beam3d (ex,ey,ez,eo,ep);
```

- Element based on **beam3e** of CalFem: we added the computation of the mass matrix **[me]**.

The input variables

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

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

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

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.

Main script   ex1_main.m

```
clear
close all

ex1_dat
ex1_geo
ex1_fem
ex1_KCM
ex1_stat
ex1_eigen
ex1_lti
…
```

Simulink model

ex1_sim_xx.m

ex1_dat.m

All input parameters

```
% Materiaux
  pois = 0.3;
% Acier
  rho_st = 7800.;
  E_st = 210000e6;   G_st = E_st / (2*(1+pois));

% Géometrie
  Lp1 = 0.2;   Lp2 = 0.1;   % longueur des segments de poutre
  np1 = 10;    np2 = 5;     % nombre d'éléments des segments de poutre

  hp = 0.006;  bp = 0.006;  % section de la poutre

  Lf = 0.005;      % longueur de la lame
  hf = 0.0003;   bf = 0.006; % section de la lame

  Mu = 1;    % masse au bout de la poutre
  Ma = 0.1;  % masse au point d'actionnement

  Kf   = 9.79;    % constante de l'actionneur [N/A]

% Proprietés des éléments
  [A, Iy, Iz, J] = rect_pro (bf,hf);
  Ep_flex = [E_st  G_st  A Iy Iz J A*rho_st];

  [A, Iy, Iz, J] = rect_pro (bp,hp);
  Ep_poutre = [E_st  G_st  A Iy Iz J A*rho_st];
```

ex1_geo.m

Model geometry:

- Nodes
- Elements and type
- Nodes for boundary conditions and loads

```
% geometrie du modèle
Coord (1,:) = [0  0  0];
n_fix = 1;

% element flex
Coord (2,:) = [ Lf 0 0 ];
Elem(1,:) = [ 1 2 ];
eFlex = 1;

% elements poutre
n = size(Coord,1);  ne = size(Elem,1);
for i = 1:np1
    Coord (n+i,:) = [ Lf+Lp1*i/np1     0    0 ];
    Elem(ne+i,:) = [ n+i-1  n+i ];
end
n_act = n+np1;

n = size(Coord,1);  ne = size(Elem,1);
for i = 1:np2
    Coord (n+i,:) = [ Lf+Lp1+Lp2*i/np2     0    0 ];
    Elem(ne+i,:) = [ n+i-1  n+i ];
end
n_mass = n+np2;

n = size(Coord,1);  ne = size(Elem,1);
ePoutre = [ 2:ne ];
```

ex1_fem.m

Topology of the finite elements model:

- Dof(), Edof(), Ex(), Ey(), Ez()
- Plots

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

% Check elements
Number_of_elements = size(Elem,1)
Check = length([ eFlex ePoutre])

Coord_xy(:,1) = Coord(:,1);       Coord_xy(:,2) = Coord(:,2);
figure; femdraw2 (Coord_xy,Ex,Ey,[1 4 2 5 0.]);
ylabel('y'); grid;
```

ex1_KCM.m

```
% ------ generate element matrices, assemble in global matrices
nd = n_nodes*n_dof;
clear K M C
K = zeros(nd); M = zeros(nd); C = zeros(nd);
nr_elem = 0;

loc = 'elements flex';
for ie = eFlex
  eo(ie,:) = [0 0 1];    % défini le sense de l'axe Z de l'élément par
                         % rapport au système d'axes du modèle général
  [ke,me] = beam3d (Ex(ie,:),Ey(ie,:),Ez(ie,:),[0 0 1],Ep_flex);
  K = assem(Edof(ie,:),K,ke);
  M = assem(Edof(ie,:),M,me);
  nr_elem = nr_elem+1;
end

loc = 'elements poutre';
for ie = ePoutre
  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);
  nr_elem = nr_elem+1;
end

Check = nr_elem
```

ex1_stat.m

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

%  ------ Load - Z moment ----------------------------------
p = zeros(size(K,1),1);
i = n_act
   dof_exc = (i-1)*n_dof+2;
   p(dof_exc) = 0.1;   % 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,[2 4 2 4 0.]);
Edbxy = [Edb(:,1) Edb(:,2) Edb(:,6) Edb(:,7) Edb(:,8) Edb(:,12)];
femdisp2 (Ex,Ey,Edbxy,[1 5 0 5],1); ylabel('y');
title('analyse static - force 0.1 N au noeud 3');

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

ex1_eigen.m

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

ex1_mass;  % first we add the node masses

b = (n_fix-1)+[1:6];   % fixing node 1
[L,Egv] = eigen (K,M,bc);
freq = sqrt(L)/(2*pi);
disp ('fréq (Hz) =')
disp (freq(1:4))
n_modes = length (freq)

for i = [1:3]
  plt_mode (Coord,Ex,Ey,Ez,Edof,real(Egv),real(freq),i,0.01);
end
```

ex1_KCM.m

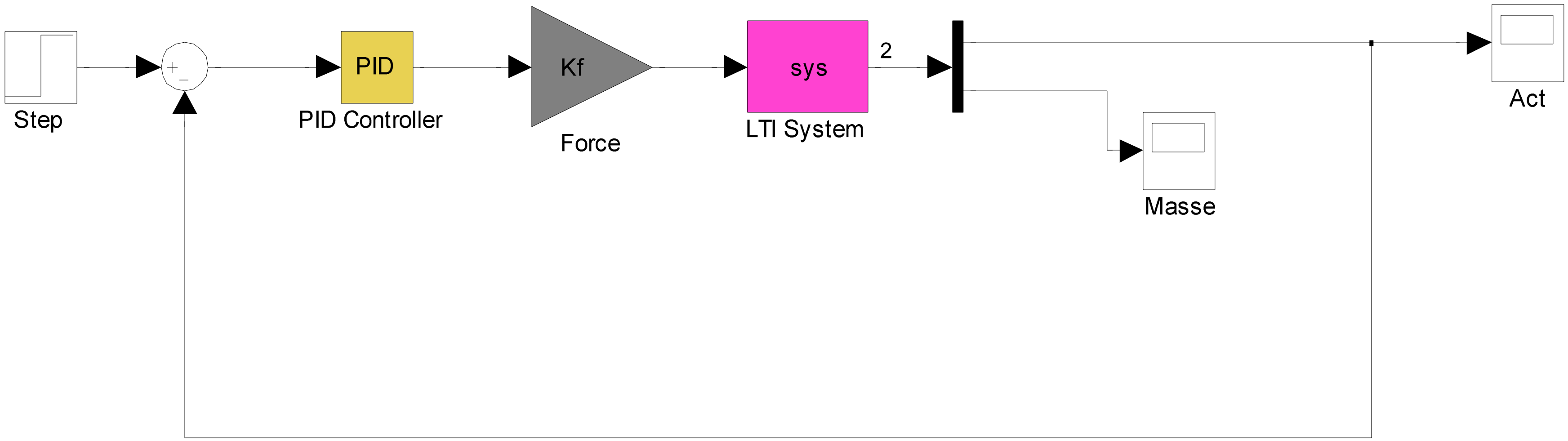
```
% ------ generate element matrices, assemble in global matrices
nd = n_nodes*n_dof;
clear K M C
K = zeros(nd); M = zeros(nd); C = zeros(nd);
nr_elem = 0;

loc = 'elements flex';
for ie = eFlex
  eo(ie,:) = [0 0 1];     % défini le sense de l'axe Z de l'élément par
                          % rapport au système d'axes du modèle général
  [ke,me] = beam3d (Ex(ie,:),Ey(ie,:),Ez(ie,:),[0 0 1],Ep_flex);
  K = assem(Edof(ie,:),K,ke);
  M = assem(Edof(ie,:),M,me);
  nr_elem = nr_elem+1;
end

loc = 'elements poutre';
for ie = ePoutre
  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);
  nr_elem = nr_elem+1;
end

Check = nr_elem
```

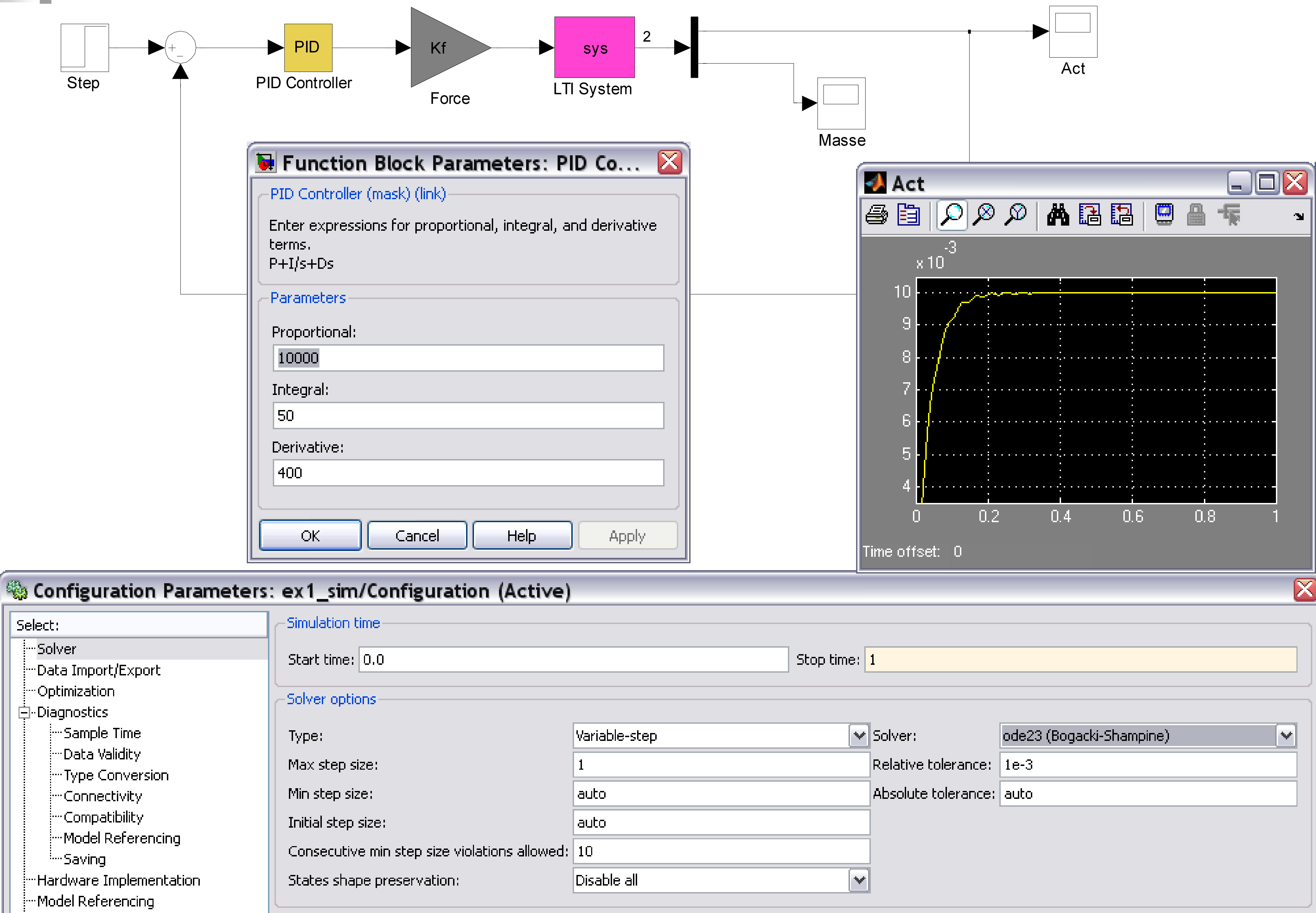
ex1_lti.m *lti = linear time invariant*

```
% Transfer function
mdof = sdt_mdof(n_nodes,n_dof);

% 1 input (actuator) -----------------------------------
b_act = fe_c (mdof,[n_act+0.02]',[1])';
pb1 = Egv'*b_act;
pb = [ pb1 ];

% 2 outputs ------------------------------------------------
c_act = fe_c(mdof,[n_act+0.02]',[1]);
cp1  = c_act*Egv;
c_mass = fe_c(mdof,[n_mass+0.02]',[1]);
cp2  = c_mass*Egv;
cp = [ cp1 zeros(1,n_modes); cp2 zeros(1,n_modes) ];

sda = 0.005;      %  structural damping = 1/2*Q

freqvec = logspace(0,3,100)'; % de 0 à 1000 Hz
w=2*pi*freqvec;                % omega
freqrad = 2*pi*freq;   % eigenfrequencies in rad/s

[a,b,c,d] = nor2ss (freqrad,sda,pb,cp);
sys = ss(a,b,c,d);
size_of_sys = size(sys)
my_plot_bode (w,sys(1,1),'b','act - act');
my_plot_bode (w,sys(2,1),'r','masse - act');
```

Simulink – step one

1: analog model

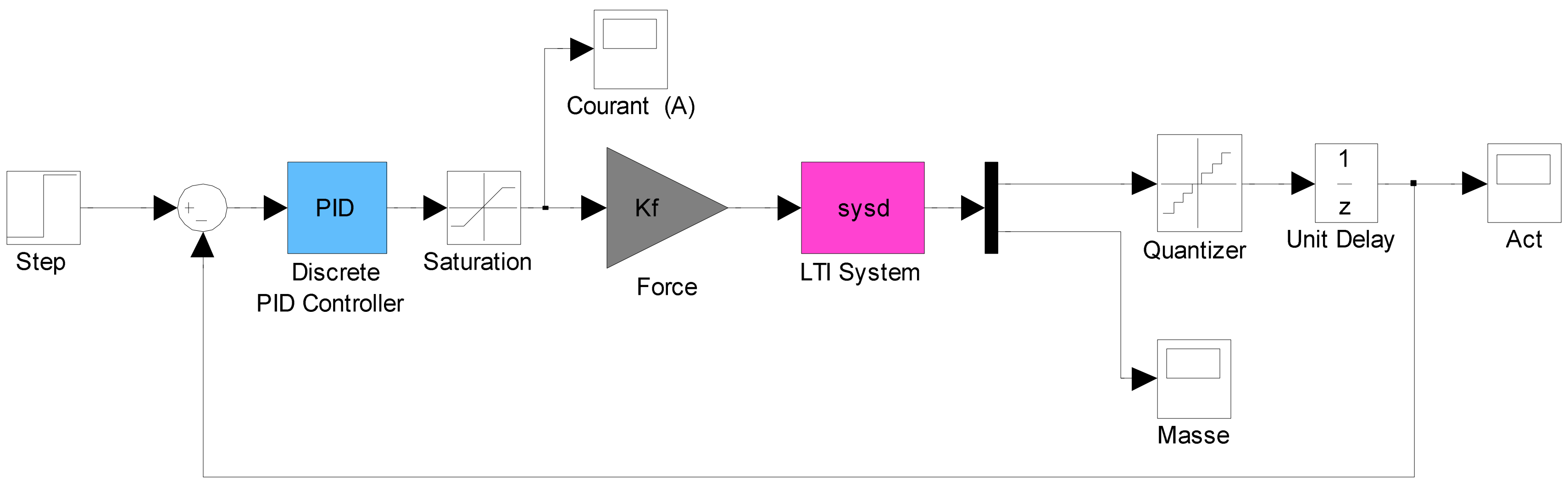


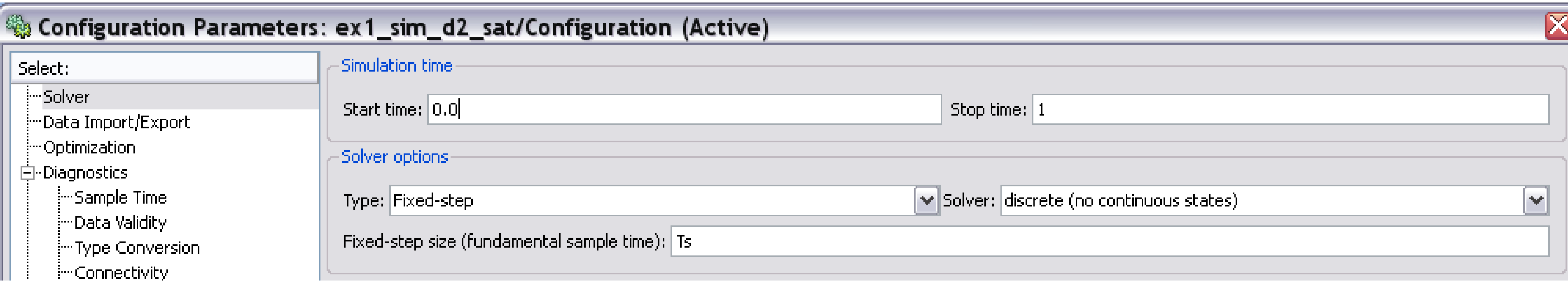

Step
PID
PID Controller
Kf
Force
sys
LTI System
2
Act
Masse
Function Block Parameters: PID Co...
PID Controller (mask) (link)
Enter expressions for proportional, integral, and derivative terms.
P+I/s+Ds
Parameters
Proportional:
10000
Integral:
50
Derivative:
400
OK
Cancel
Help
Apply
Act
x 10
-3
10
9
8
7
6
5
4
0
0.2
0.4
0.6
0.8
1
Time offset: 0
Configuration Parameters: ex1_sim/Configuration (Active)
Select:
Solver
Data Import/Export
Optimization
Diagnostics
Sample Time
Data Validity
Type Conversion
Connectivity
Compatibility
Model Referencing
Saving
Hardware Implementation
Model Referencing
Simulation time
Start time: 0.0
Stop time: 1
Solver options
Type: Variable-step
Solver: ode23 (Bogacki-Shampine)
Max step size: 1
Relative tolerance: 1e-3
Min step size: auto
Absolute tolerance: auto
Initial step size: auto
Consecutive min step size violations allowed: 10
States shape preservation: Disable all

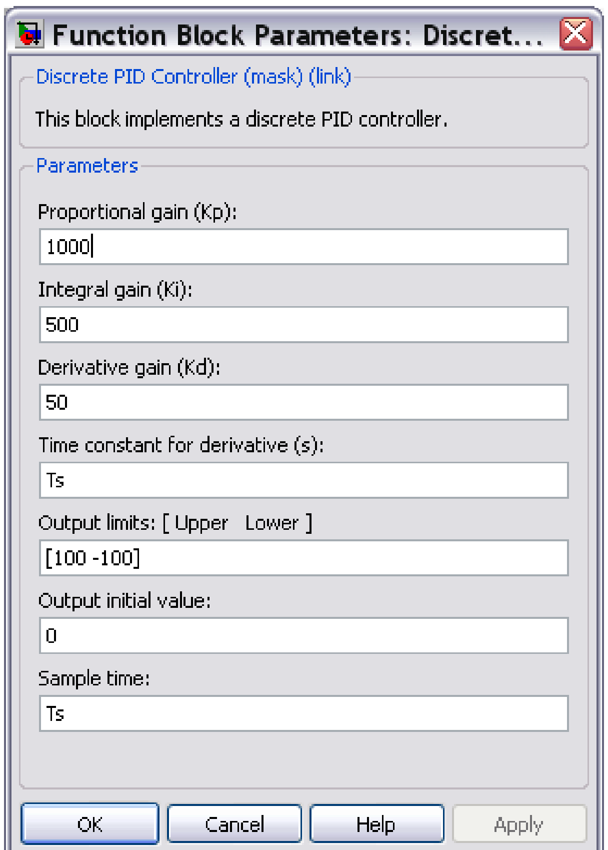
Simulink – step 2

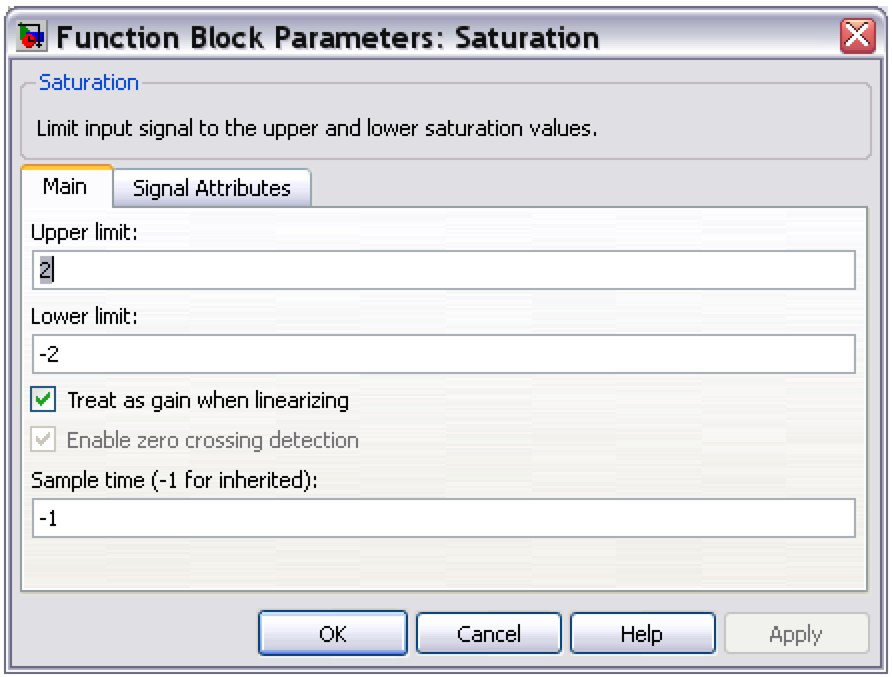
2: digital feedback model

```
Ts = 100e-6
sysd = c2d(sys,Ts);
```



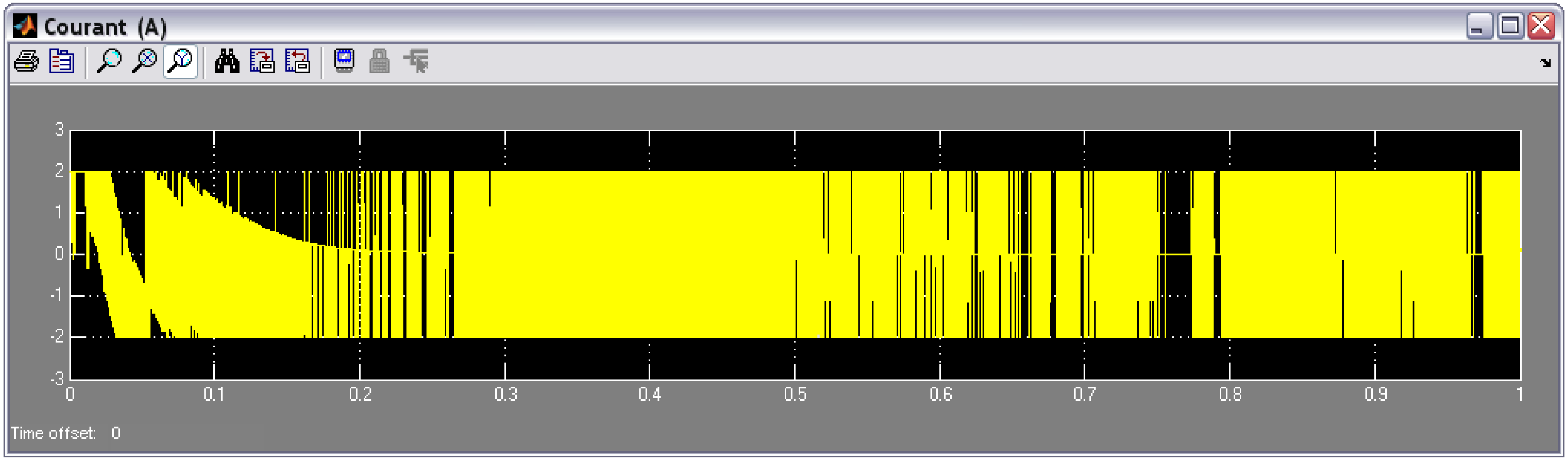

Configuration Parameters: ex1_sim_d2_sat/Configuration (Active)
Select:
Solver
Data Import/Export
Optimization
Diagnostics
Sample Time
Data Validity
Type Conversion
Connectivity
Simulation time
Start time: 0.0
Stop time: 1
Solver options
Type: Fixed-step
Solver: discrete (no continuous states)
Fixed-step size (fundamental sample time): Ts

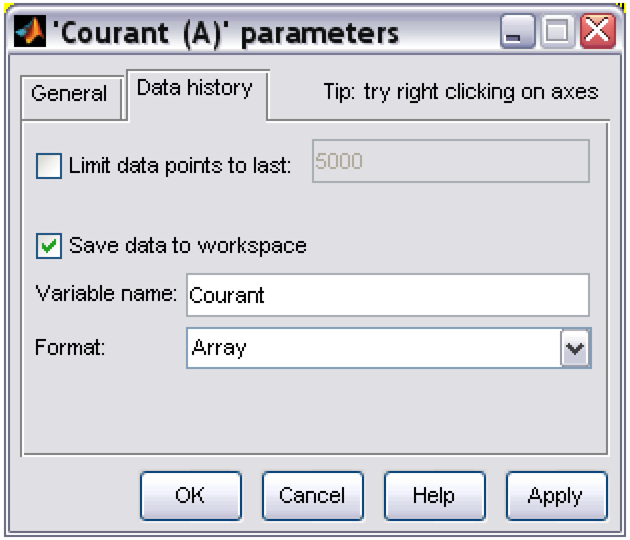

Function Block Parameters: Discret...
Discrete PID Controller (mask) (link)
This block implements a discrete PID controller.
Parameters
Proportional gain (Kp):
1000
Integral gain (Ki):
500
Derivative gain (Kd):
50
Time constant for derivative (s):
Ts
Output limits: [Upper Lower]
[100 -100]
Output initial value:
0
Sample time:
Ts
OK
Cancel
Help
Apply


Function Block Parameters: Saturation
Saturation
Limit input signal to the upper and lower saturation values.
Main
Signal Attributes
Upper limit:
2
Lower limit:
-2
Treat as gain when linearizing
Enable zero crossing detection
Sample time (-1 for inherited):
-1
OK
Cancel
Help
Apply

Other analyses

example: heat dissipation of the voice coil actuator






```
dissip = courant(:,2).^2 * 17.1;
dissip_moyen = mean(dissip)
```

…we could then program a thermal analysis to evaluate the temperature distribution in the beam, etc. ...