École de Technologie Supérieure

Devoir 1– Février 2015

Simulation numérique des systèmes

SYS 810

Milan Assuied, Adam Burney

https://www.sharelatex.com/project/

1 Problème 1

Circuit électrique non linéaire

1. Écrire la représentation d'état non linéaire de ce circuit.
   Choisir comme variables d'état $x_1 = e_c$ et $x_2 = i_l$.

   Les équations du circuit s'écrivent:

$$\begin{aligned} i_2 + i_c &= i_r + i_l && (1.1)\\ e_i &= e_c + e_l && (1.2)\\ &= e_c + e_r && (1.3)\\ e_r &= e_l \end{aligned}$$

   De (1.1) il vient:

$$\frac{1}{8}e_c^3 + C\frac{de_c}{dt} = i_l + i_r$$

   or de (1.3):

$$i_r = \frac{1}{R}e_i - \frac{1}{R}e_c$$

   Il en résulte:

$$\boxed{\frac{de_c}{dt} = -\frac{1}{8C}e_c^3 - \frac{1}{RC}e_c + \frac{1}{C}i_l + \frac{1}{RC}e_i} \quad (1.4)$$

   De (1.2) il vient:

$$\begin{aligned} e_i &= e_c + e_l\\ e_i &= e_c + L\frac{di_l}{dt}\\ \boxed{\frac{di_l}{dt} &= -\frac{1}{L}e_c + \frac{1}{L}e_i} \end{aligned} \quad (1.5)$$

   De (1.4) et (1.5) on déduit la représentation d'état:

$$\boxed{\begin{cases} \dfrac{dx_1}{dt} = -\dfrac{1}{8C}x_1^3 - \dfrac{1}{RC}x_1 + \dfrac{1}{C}x_2 + \dfrac{1}{RC}\cdot e_i, & \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} e_c \\ i_l \end{bmatrix} \\ \dfrac{dx_2}{dt} = -\dfrac{1}{L}x_1 + \dfrac{1}{L}\cdot e_i \end{cases}} \quad (1.6)$$

2. Déterminer le point d'équilibre du système.

L'équilibre est donné par

$$\begin{cases} -\frac{1}{8C}{e_c^*}^3 - \frac{1}{RC}e_c^* + \frac{1}{C}i_l^* + \frac{1}{RC}\langle e_i\rangle = 0 \\ -\frac{1}{L}e_c^* + \frac{1}{L}\langle e_i\rangle = 0 \end{cases}$$

Soit:

$$\begin{cases} -\frac{1}{8C}\langle e_i\rangle^3 - \frac{1}{RC}\langle e_i\rangle + \frac{1}{C}i_l^* + \frac{1}{RC}\langle e_i\rangle = 0 \\ e_c^* = \langle e_i\rangle \end{cases}$$

Finalement:

$$\boxed{\begin{cases} i_l^* = \frac{1}{8}\langle e_i\rangle^3 \\ e_c^* = \langle e_i\rangle \end{cases}} \tag{1.7}$$

On linéarise autour du point d'équilibre:

$$\dot{\delta}x = \left.\frac{\partial F}{\partial X}\right|_{(X_*,U_*)} \cdot \delta x + \left.\frac{\partial F}{\partial U}\right|_{(X_*,U_*)\cdot\delta u}$$

Soit:

$$\dot{\delta}x = \begin{bmatrix} \left.\frac{\partial\left(-\frac{1}{8C}x_1^3 - \frac{1}{RC}x_1 + \frac{1}{C}x_2\right)}{\partial x1}\right|_{x1_*} & \left.\frac{\partial\left(-\frac{1}{8C}x_1^3 - \frac{1}{RC}x_1 + \frac{1}{C}x_2\right)}{\partial x2}\right|_{x2_*} \\ \left.\frac{\partial\left(-\frac{1}{L}x_1\right)}{\partial x1}\right|_{x1_*} & \left.\frac{\partial\left(-\frac{1}{L}x_1\right)}{\partial x2}\right|_{x2_*} \end{bmatrix} \cdot \delta x + \begin{bmatrix} \left.\frac{\partial\left(\frac{1}{RC}\cdot e_i\right)}{\partial e_i}\right|_{\langle e_i\rangle} \\ \left.\frac{\partial\left(\frac{1}{L}\cdot e_i\right)}{\partial e_i}\right|_{\langle e_i\rangle} \end{bmatrix} \cdot \delta u$$

Par suite:

$$\boxed{\dot{\delta}x = \begin{bmatrix} \frac{-3}{8C}\langle e_i\rangle^2 - \frac{1}{RC} & \frac{1}{C} \\ \frac{-1}{L} & 0 \end{bmatrix} \cdot \delta x + \begin{bmatrix} \frac{1}{RC} \\ \frac{1}{L} \end{bmatrix} \cdot \delta u} \tag{1.8}$$

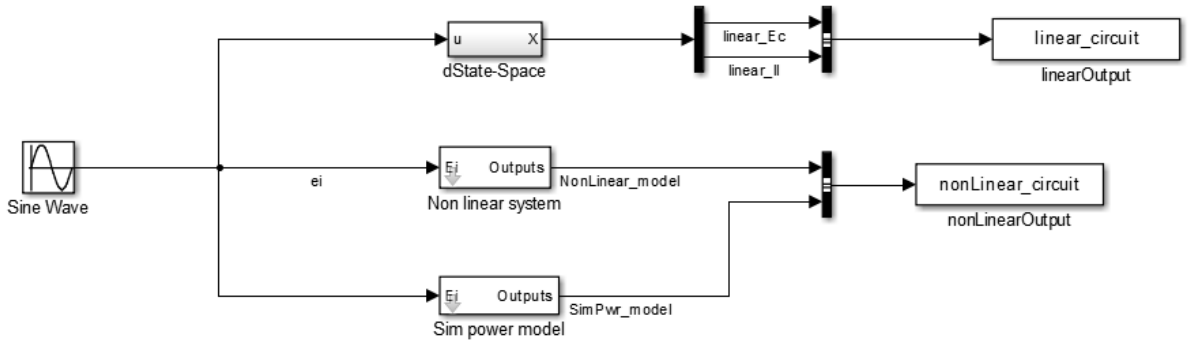

Figure 1: Simulink model used for electrical simulation and comparison

3. Exécuter la simulation.

   Pour effectuer la simulation, nous utilisons une représentation du modèle exact supporté par une modèle réalisé avec SimPowers.

   Ce type de montage permet de vérifier qu'en tout temps la sortie du modèle représenté par le système d'équations non linéaires est égale à la sortie générée par le modèle de type SimPower. Le modèle ici étant très simple, il n'est pas nécessaire d'ajouter une tolérance sur l'assertion.

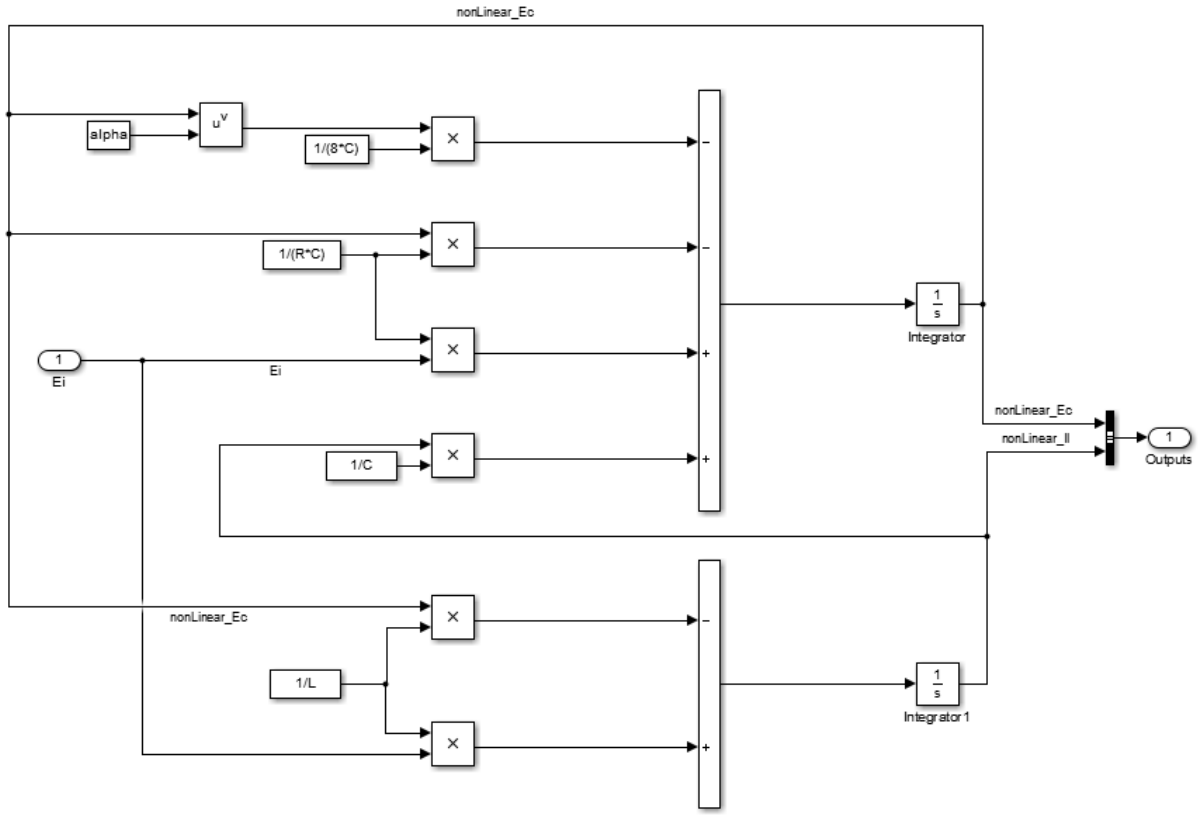

Figure 2: Simulink exact model

Dans un soucis de clareté, des sous-systèmes sont utilisés:

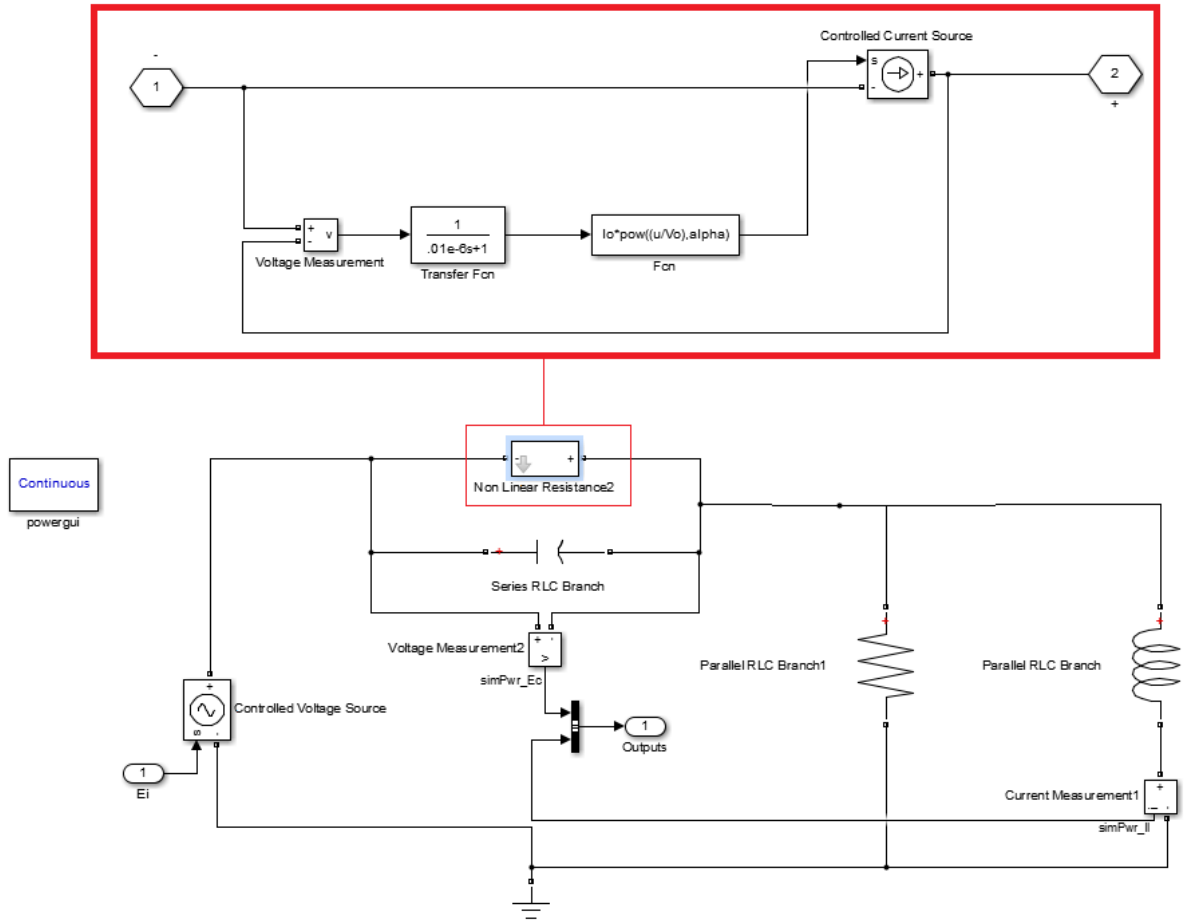

Figure 3: Simpower model

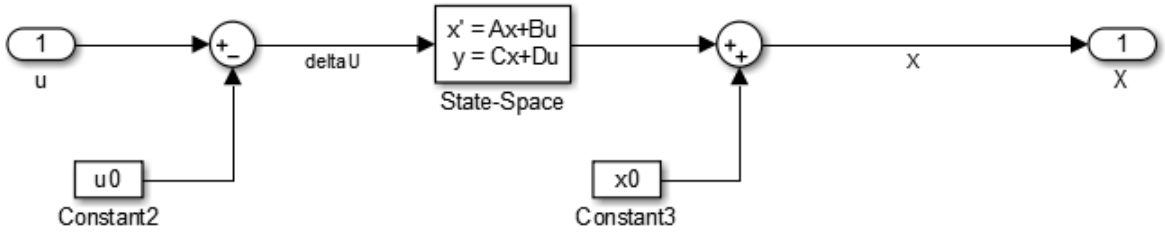

Figure 4: Simpower linearized state space model

4. Résultats

   On trace et on superpose, pour trois valeurs différentes d'amplitude de la source de tension, l'intensité aux bornes de la bobine. Avec une amplitude très faible, la bobine est presque équivalente à un court circuit tandis que le condensateur équivaut à une borne ouverte. Le circuit est alors équivalent à la seule résistance non linéaire en série avec la source de tension, la tension a ses bornes est équivalente à la tension aux bornes de la source, en conséquence l'intensitée attendue aux bornes de l'inducteur est $-\frac{1}{8C}e_c^3 = 1A$. L'annulation des états dérivés du circuit doit donc se matérialiser par un circuit pratiquement linéaire. Ceci est confirmé sur les courbes suivantes:

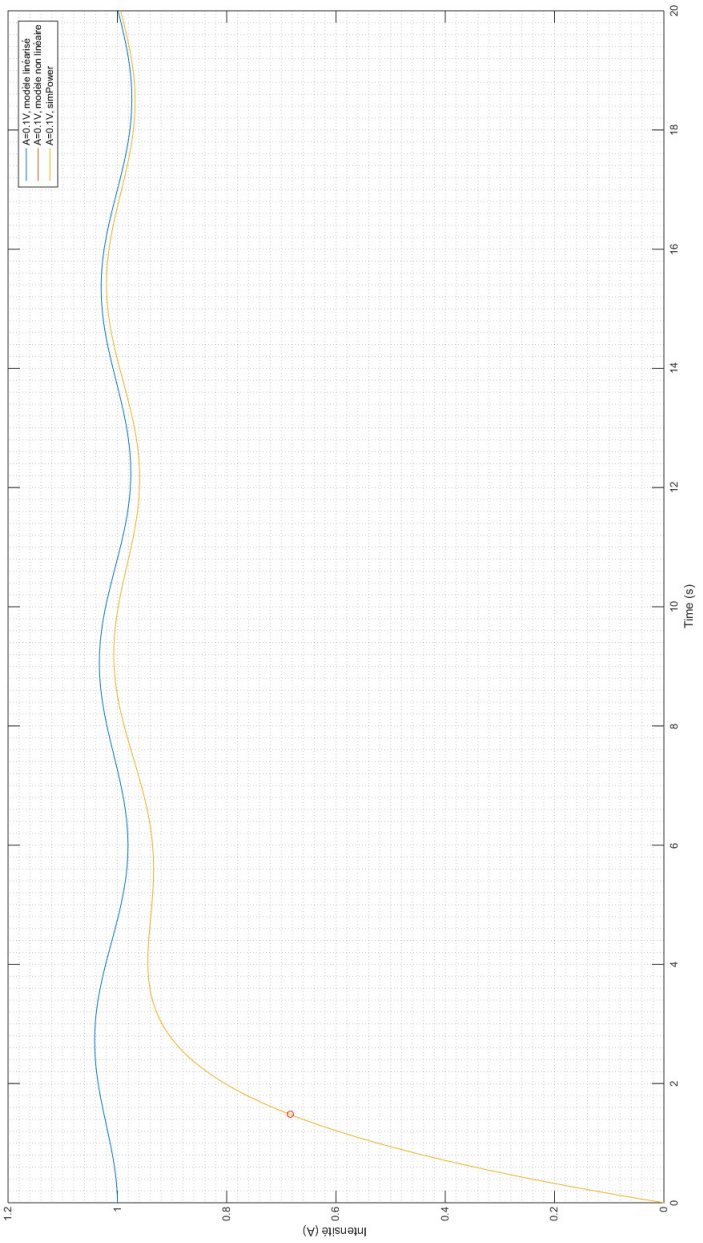

Figure 5: Courbe d'intensité au cours du temps avec A=0.1V

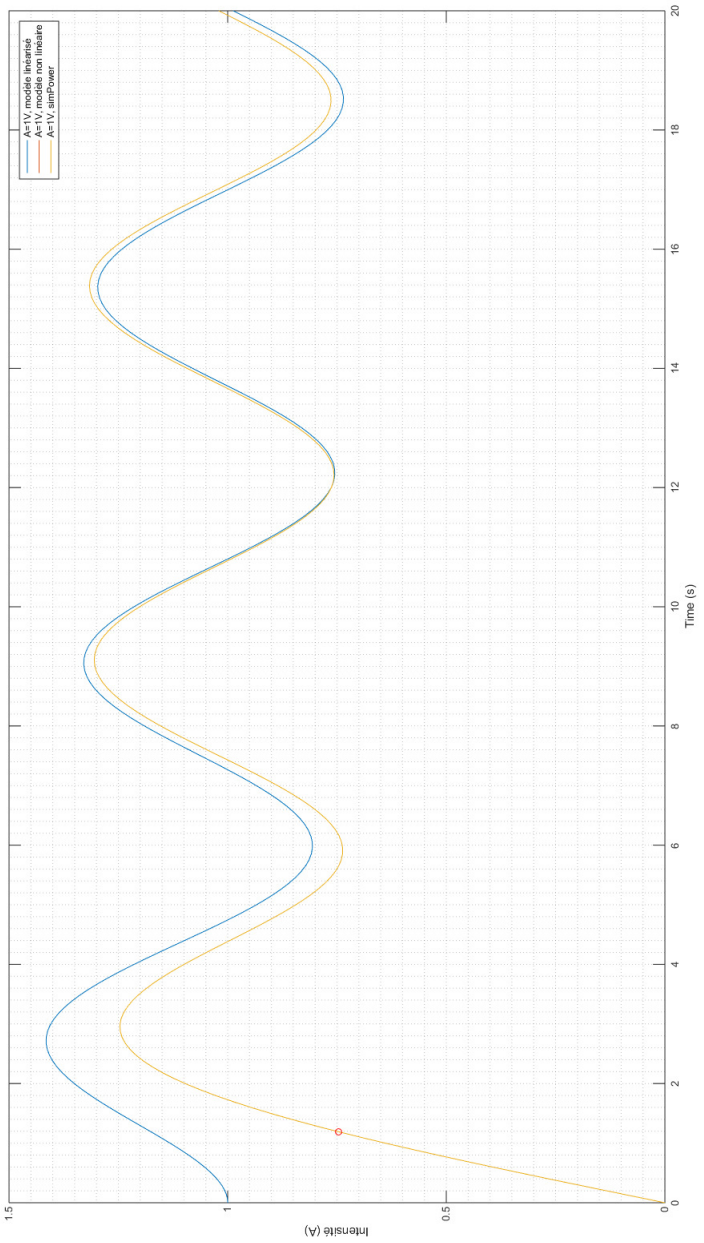

Figure 6: Courbe d'intensité au cours du temps avec A=1V

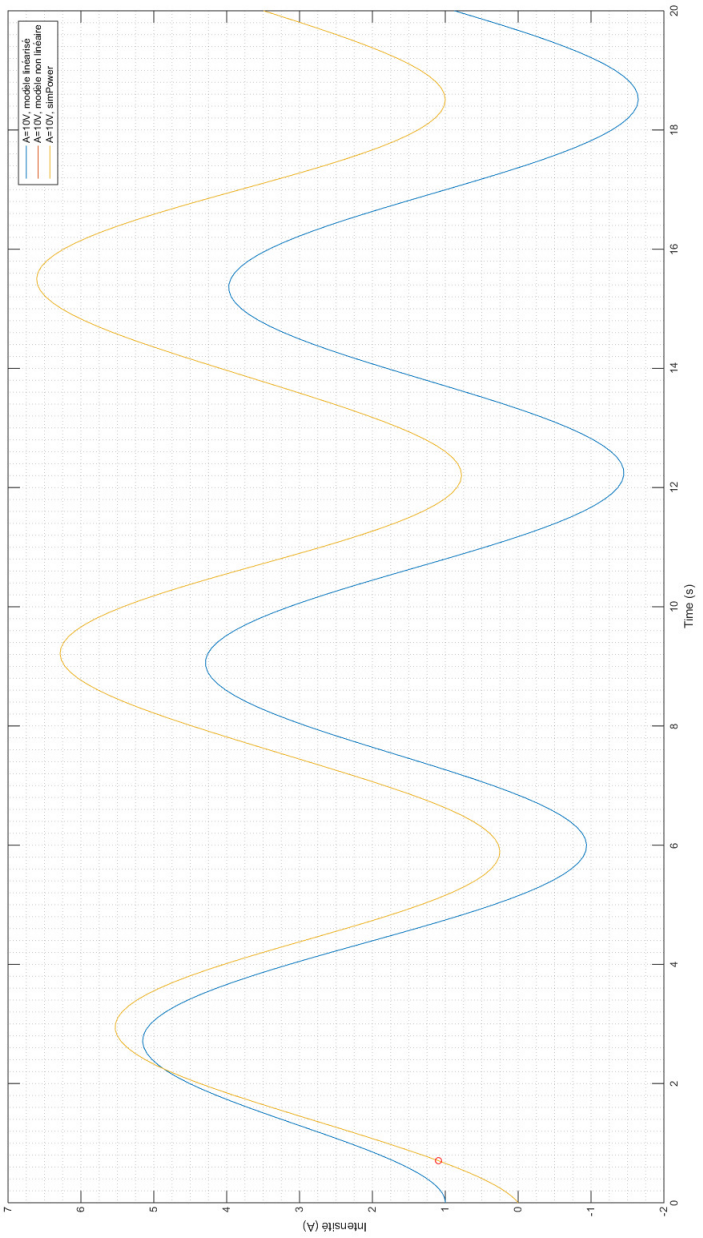

Figure 7: Courbe d'intensité au cours du temps avec A=10V

Ces courbes permettent de tirer les deux conclusions suivantes:

- Le modèle linéarisé ne permet pas de représenter les modes transitoires.
  Ceci est normal puisque le modèle linéaire n'est représentatif qu'autour de l'équilibre.
  Les valeurs initiales sont donc faussées.
- Le modèle linéarisé offre une plage de résultats relativement acceptables jusqu'à $A = 1A$.
  Au dela les valeurs n'ont aucune représentativité au regard de la part alternative de la source de tension.

2 Problème 2

Système masse ressort non linéaire

1. Écrire les équations différentielles qui modélisent ce système.

   Il est admis que la masse est située à x_0 à l'état d'équilibre. Par suite, l'équation d'équilibre s'écrit:

   $$-M.g + k_1.(x_l - x_0) = 0$$

   Il en résulte l'élongation naturelle du ressort:

   $$\boxed{x_l = \frac{M.g}{k_1} + x_0} \tag{2.1}$$

   En ignorant le plateau amortisseur, on établit l'équation libre du mouvement:

   $$\begin{aligned} M.\frac{d^2x}{dt^2} &= -M.g - k_1(x - x_l) \\ \frac{d^2x}{dt^2} &= -g - \frac{k_1}{M}(x - x_l) \\ \frac{d^2x}{dt^2} &= -g - \frac{k_1}{M}(x - (\frac{M.g}{k_1} + x_0)) \end{aligned} \tag{2.2}$$

   Lorsque le plateau amortisseur est compressé, l'équation devient:

   $$\frac{d^2x}{dt^2} = -g - \frac{k_1}{M}(x - x_l) - \frac{k_2}{M}(x - x_1) - \frac{b}{M}\frac{dx}{dt} \tag{2.3}$$

   Cependant, il convient de noter qu'en l'absence de viscosité le plateau ne peut exercer qu'une action positive sur la masse aussi l'hypothèse suivante est effectuée:

   La masse remontant plus vite que le plateau en raison de l'action du ressort k_1, l'action du plateau n'est pas constante lorsque $\frac{dx}{dt} > 0$. Or la masse du plateau amortisseur étant considérée comme nulle on obtient:

   $$\lim_{Mp \to 0} -\frac{k_2}{Mp}(x - x_1) - \frac{b}{Mp}\frac{dx}{dt} = \frac{d^2Xp}{dt^2} = \infty$$

   On considère donc le plateau en contact constant durant la phase de remontée.

   Il en résulte l'équation finale:

   $$\frac{d^2x}{dt^2} = -g - \frac{k_1}{M}(x - (\frac{M.g}{k_1} + x_0)) + f(x)$$

$$\begin{cases} f(x) = 0, \ x >= x_1 \\ f(x) = \min[-\frac{k_2}{M}(x - x_1) - \frac{b}{M}\frac{dx}{dt}, 0], \ x < x_1 \end{cases}$$

Pour la suite de l'exercice on note:

$$\boxed{\frac{d^2x}{dt^2} = F_M(x) + F_s(x)} \tag{2.4}$$

$$\boxed{\begin{cases} F_M(x) = -g - \frac{k_1}{M}(x - (\frac{M.g}{k_1} + x_0)) & , \forall x \\ F_s(x) = 0 & , x >= x_1 \\ F_s(x) = \min[-\frac{k_2}{M}(x - x_1) - \frac{b}{M}\frac{dx}{dt}, 0] & , x < x_1 \end{cases}} \tag{2.5}$$

2. Simuler le système.

   Pour modéliser la non linéarité, nous utilisons dans un cas un bloc switch, dans un autre cas un bloc matlab function représentant le bloc switch. Le modèle est par soucis de clarté décomposé en deux sous-systèmes, l'un représentant $F_M(x)$ et l'autre $F_s(x)$. Note: afin de valider le modèle, un frottement visqueux est rajouté à $F_M(x)$. Compte tenu des vitesses mises en jeu, un frottement de type $f(v)$ au lieu de $f(v^2)$ est acceptable. Cet amortissement est nul durant les simulations présentées dans ce rapport.

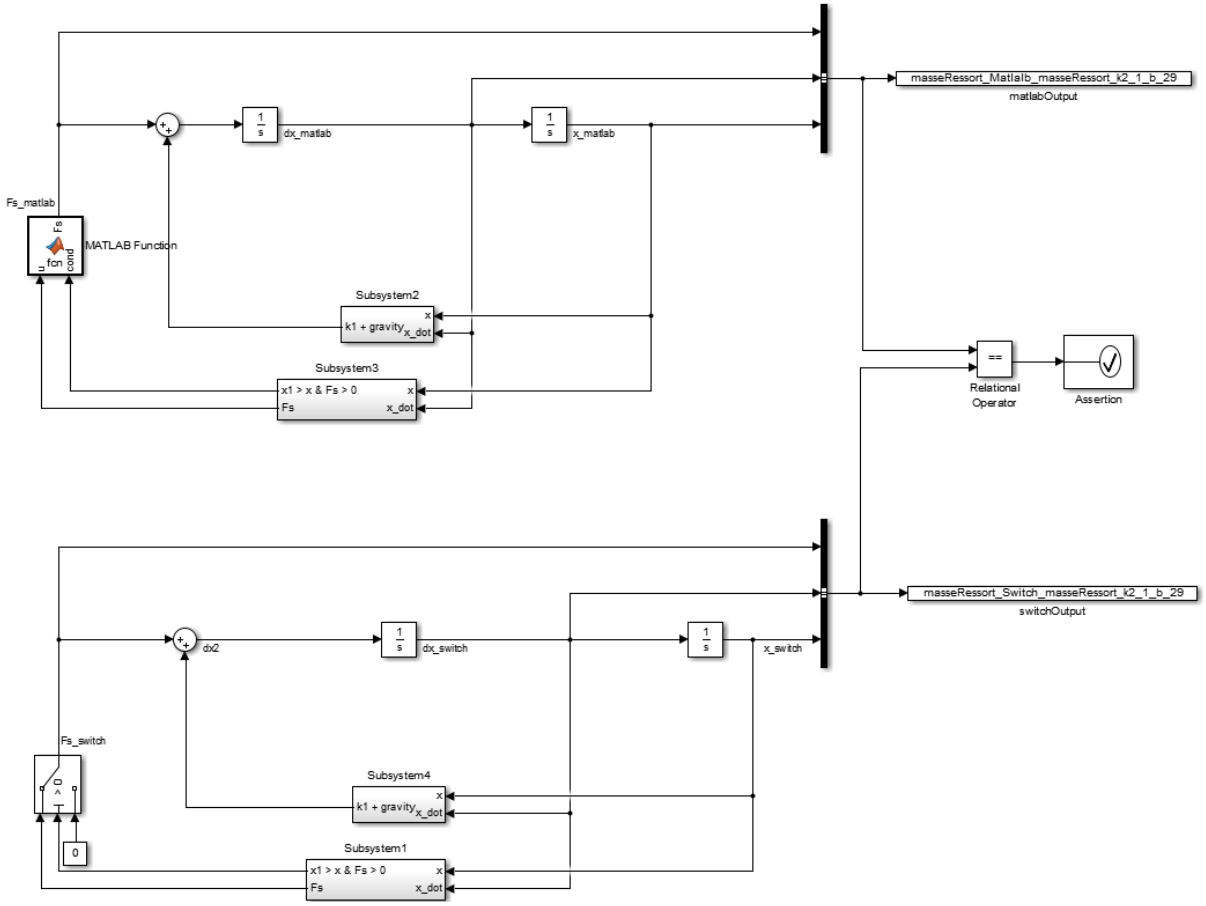

Figure 8: Simulink overview

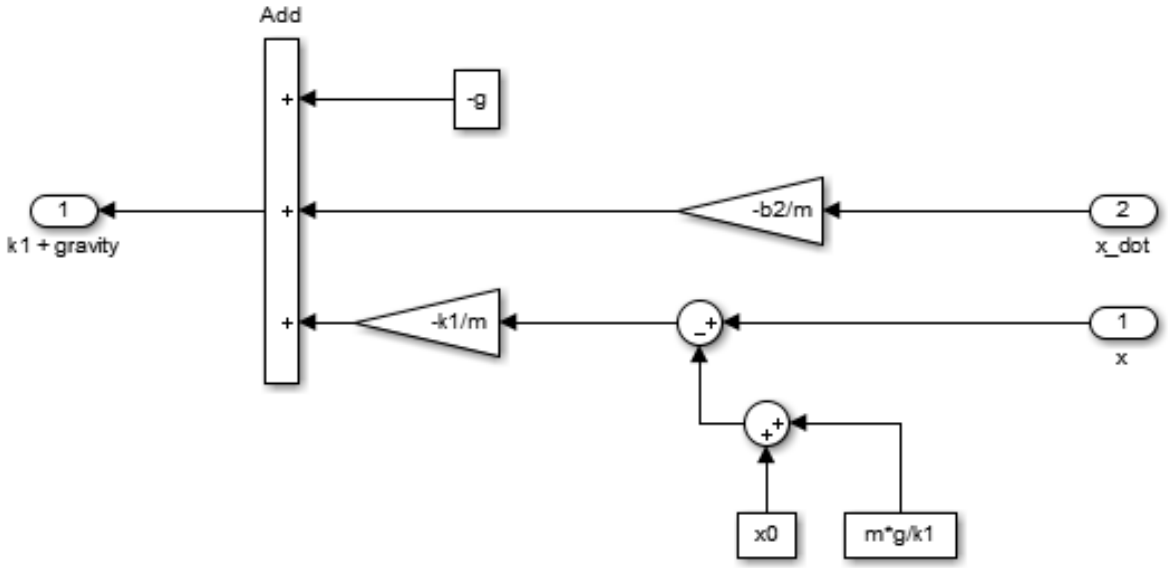

Figure 9: Fm

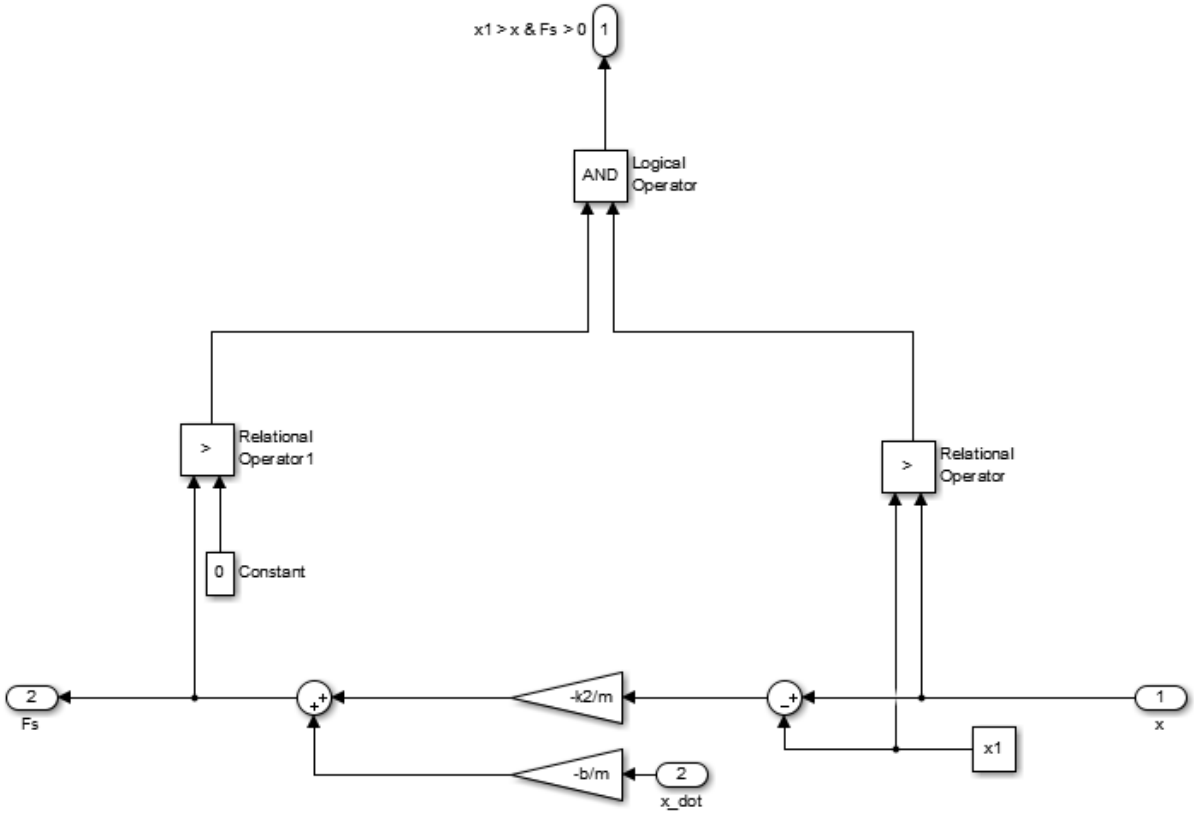
x1 > x & Fs > 0
1
AND
Logical Operator
>
Relational Operator1
0
Constant
>
Relational Operator
2
Fs
+
+
-k2/m
-
+
1
x
x1
-b/m
2
x_dot

Figure 10: Fs

3. Présentation des résultats.

   On trace les courbes associées à $x, F_s(x)$ et $\frac{dx}{dt}$ pour $K_2 = 1, b = 1$:

   La figure 11 permet de voir que la force F_s ne s'applique que pour $x < 1$ et est strictement positive.

   On trace ensuite le déplacement maximal de la masse en fonction de K_2 et b. Pour une meilleure résolution le pas de valeur est de 1 entre 10 et 50 pour chacune des variables.

   Il apparait clairement que le déplacement minimal est atteint aux valeurs maximales de K_2 et b. Cependant pour des raisons de coût et d'optimisation, il est pertinent de s'intéresser au gain procuré par les plus faibles valeurs de K_2 et b. La figure 12 montre clairement que le gain est très fort lorsque K_2 et b augmentent faiblement alors que leurs valeurs sont elles même faibles. Mais ce gain diminue rapidement.

   On s'intéresse donc au lieu des gradients tels que $\vec{\nabla}(X_{Min}(k2, b)) < 0.01$. Cette valeur est arbitraire et on la considère comme étant le seuil à partir duquel le gain est nul.

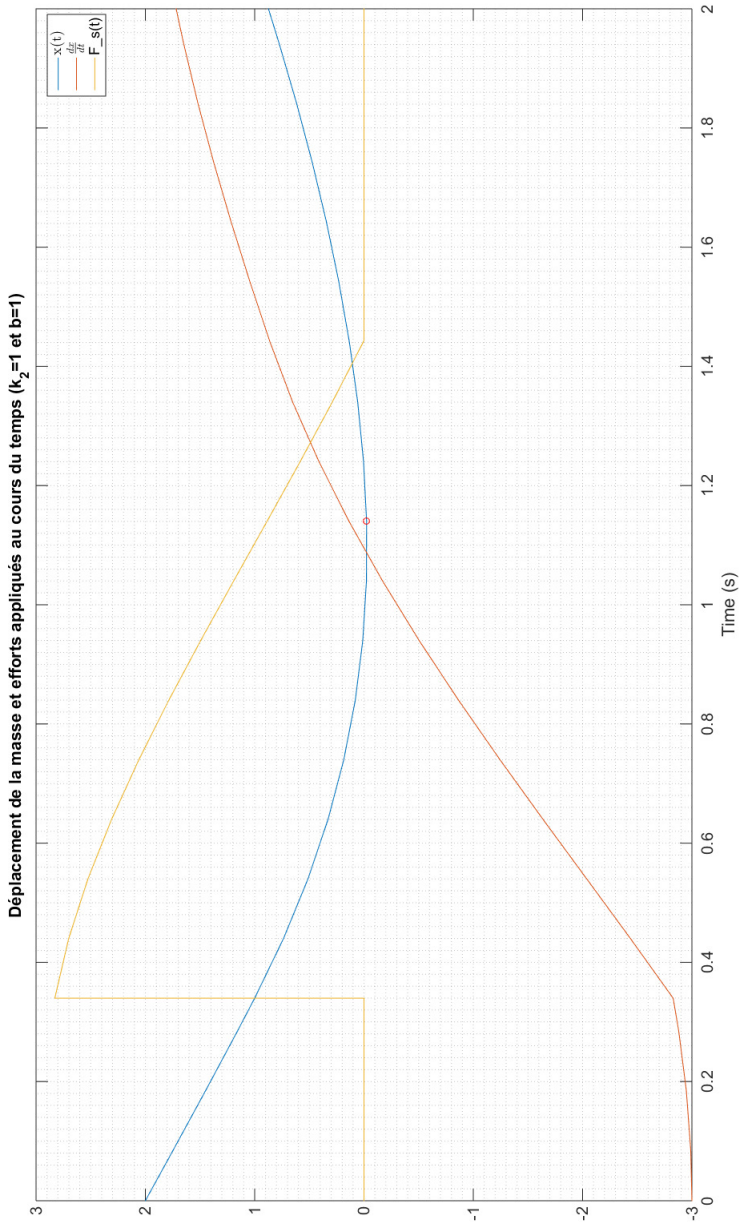

Figure 11: Évolutions de $x, F_s(x)$ et $\frac{dx}{dt}$ pour $K_2 = 1, b = 1$

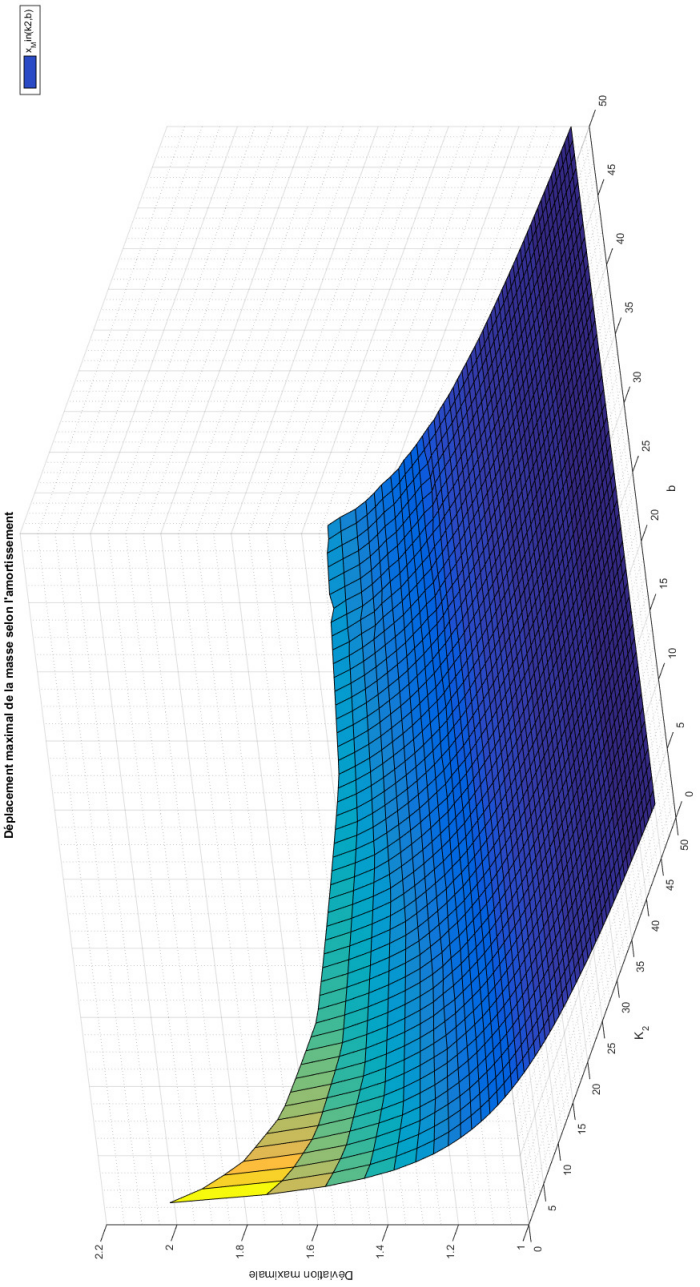

Figure 12: Déplacement maximal de la masse depuis sa position d'origine.

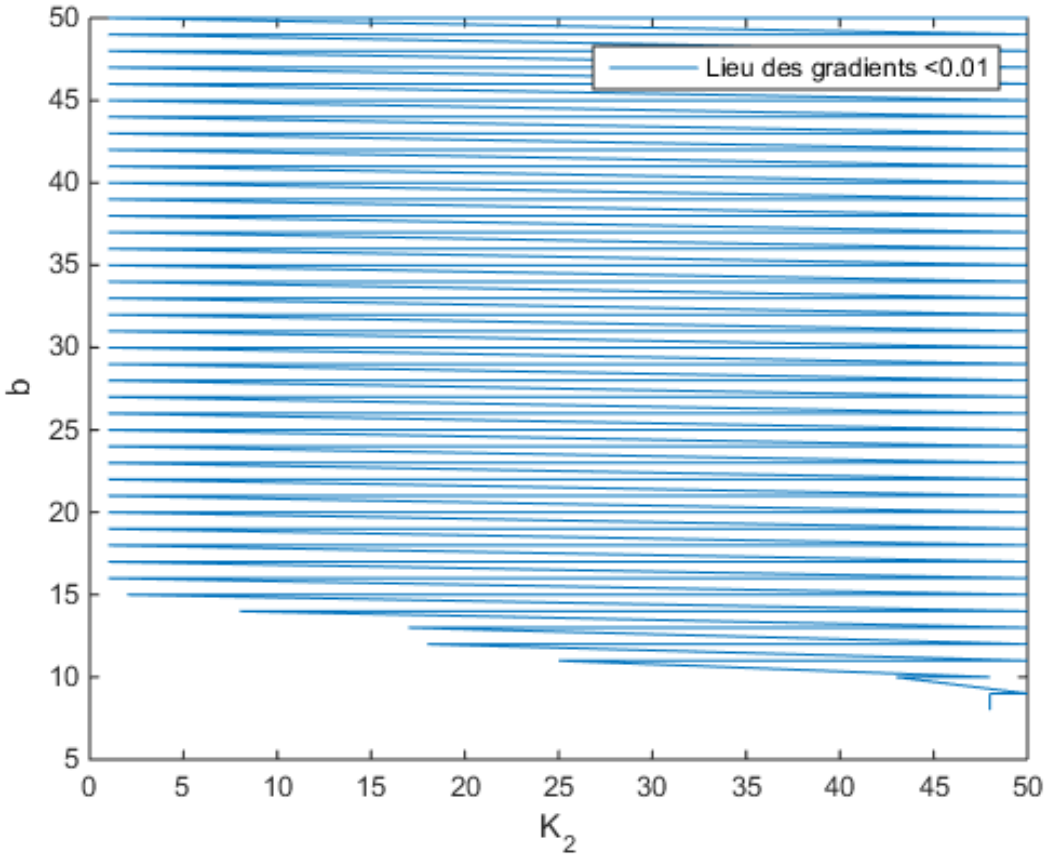

Figure 13: Lieux des gradients

La figure 13 montre qu'il est possible de choisir le couple (K_2, b) dans des plages de valeurs acceptables telles que (25,20). En comparaison, le gain avec le couple (50,50) n'est que de 6%. Cette façon de faire laisse également plus de choix pour répondre aux contraintes de confort, ou d'effort applicable à la masse, tout en gardant un amortissement équivalent.

ANNEXES: Code Matlab

1. Exercice1 : Code d'exécution

```
addpath(genpath('..\..'));

clear all; %#ok<CLSCR>
%Input tension amplitude
Ais=[0.1,1,10];
Ei=2;

%Varistor parameters
Io=1/8;
Vo=1;
alpha=3;

%Circuit constants
Ca=2;
L=3;
R=1;

%Point d'equilibre
Ei_mean = Ei;
u0 = Ei_mean;
x0 = [Ei_mean;1/8*Ei_mean^3];

%Representation d'etat linearisee autour du point d'equilibre:
A=[-3/(8*Ca)*Ei_mean^2-1/(R*Ca), 1/Ca;-1/L,0];
B=[1/(R*Ca);1/L];
C=eye(2);
D=[0;0];

%Parametres de simulation
t0=0;
tf=20;
max_step=0.1;
min_step=0.001;
tol=0.001;

%Jacobian computation
syms x1 x2 ei x1_dot x2_dot Rj Cj Lj eij
x1_dot = -1/(8*Cj)*x1^3 - 1/(Rj*Cj)*x1 + 1/Cj*x2 + 1/(Rj*Cj)*eij;
x2_dot = -1/Lj*x1 + 1/Lj*eij;
Aj = jacobian([x1_dot;x2_dot],[x1;x2])
Bj = jacobian([x1_dot;x2_dot],eij)

As = subs(Aj,Cj,Ca);
As = subs(As,Rj,R);
As = subs(As,Lj,L);
As = subs(As,x1,Ei_mean);
As = double(As);

Bs = subs(Bj,Cj,Ca);
Bs = subs(Bs,Rj,R);
Bs = subs(Bs,Lj,L);
```

```
Bs = double(Bs);

%Parametrage simulation
model='circuitElectrique';
load_system(model)
tic
for i=1:length(Ais)
    Ai = Ais(i);

    wSaveFileName        = strcat('circuit_Ai_',..
        strrep(num2str(Ai),'.','dec'));
    wLinearOutputName    = strcat('linear_',wSaveFileName);
    wNonLinearOutputName = strcat('nonLinear_',wSaveFileName);

    set_param(model,'StopFcn',..
        'save(wSaveFileName,wLinearOutputName,wNonLinearOutputName)');
    set_param(strcat(model,'/linearOutput'),..
        'VariableName',wLinearOutputName);
    set_param(strcat(model,'/nonLinearOutput'),..
        'VariableName',wNonLinearOutputName);

    myopts=simset('SrcWorkspace','current','DstWorkspace','current');

    sim(model,tf,myopts);
    while (strcmp(get_param(model,'SimulationStatus'),'stopped')==0);
    end
end
t_sim = toc;
close_system(model,false)
fprintf('\nTemps de simulation => %3.3g s\n',t_sim)
```

2. Exercice1 : Code de post-process

```
wAi = cellstr(['Ai_0dec1';'Ai_1    ';'Ai_10   ']);
wAin = cellstr(['0.1';'1  ';'10 ']);

for i=1:size(wAi,1)

    wCurrentAi  = wAi(i);
    wCurrentAin = wAin(i);

    wCurrentAi  = wCurrentAi{1};
    wCurrentAin = wCurrentAin{1};

    figure();
    wFileName = strcat('circuit_',wCurrentAi,'.mat');
    load(wFileName);
    wLinearModelIl    =  eval(strcat('linear_circuit_',..
        wCurrentAi,'.linear_Il'));
    wNonlinearModelIl =  eval(strcat('nonLinear_circuit_',..
        wCurrentAi,'.NonLinear_model.nonLinear_Il'));
    wSimPowerIl       =  eval(strcat('nonLinear_circuit_',..
        wCurrentAi,'.SimPwr_model.simPwr_Il'));

    wTau = 0.67*mean(wLinearModelIl.Data);
    wIndexTau = find(wNonlinearModelIl.Data>wTau,1,'first');

    plot(wLinearModelIl.Time,wLinearModelIl.Data);
    hold('on');

    plot(wNonlinearModelIl.Time,wNonlinearModelIl.Data);
    plot(wSimPowerIl.Time,wSimPowerIl.Data);
    plot(wNonlinearModelIl.Time(wIndexTau),..
        wNonlinearModelIl.Data(wIndexTau),'ro');

    grid minor;
    legend(strcat('A=',wCurrentAin,'V, modele linearise'),..
        strcat('A=',wCurrentAin,'V, modele non lineaire'),..
        strcat('A=',wCurrentAin,'V, simPower'));
    xlabel('Time (s)');
    ylabel('Intensite (A)');

end
```

3. Exercice2 : Code d'exécution

```
addpath(genpath('..\..'));

clear all; %#ok<CLSCR>

%Parametres
g=9.8; %#ok<*NASGU>
m=1;
k1=1;
k2s=1:1:50;
bs=1:1:50;
b2=0;
v0=-3;
x1=1;
x0=2;

%Parametres de simulation
t0=0;
tf=20;
max_step=0.1;
min_step=0.0001;
tol=0.0001;

%Parametrage simulation
model='masseRessort';
load_system(model)
tic
for i=1:length(k2s)
    k2 = k2s(i);

    for j=1:length(bs)
        b=bs(j);

        wSaveFileName     = strcat('masseRessort_k2_',..
            strrep(strcat(num2str(k2),'_b_',num2str(b)),'.','dec'));
        wSwitchOutputName = strcat('masseRessort_Switch_',wSaveFileName);
        wMatlabOutputName = strcat('masseRessort_Matlalb_',wSaveFileName);

        set_param(model,'StopFcn',..
        'save(wSaveFileName,wSwitchOutputName,wMatlabOutputName)');
        set_param(strcat(model,'/switchOutput'),..
        'VariableName',wSwitchOutputName);
        set_param(strcat(model,'/matlabOutput'),..
        'VariableName',wMatlabOutputName);

        myopts=simset('SrcWorkspace','current',..
            'DstWorkspace','current');

        sim(model,tf,myopts);
        while (strcmp(get_param(model,'SimulationStatus'),'stopped')==0);
        end
    end
```

```
end
t_sim = toc;
close_system(model,false)
fprintf('\nTemps de simulation => %3.3g s\n',t_sim)
```

4. Exercice2 : Code de post-process

```
k2s=[1:1:50];
bs=[1:1:50];

%Specific 3-D post treatment

K2=zeros(length(k2s),1);
B=zeros(length(bs),1);
Xm=zeros(length(k2s),length(bs));

for i=1:length(k2s)
    k2 = k2s(i);
    K2(i)=k2;

    for j=1:length(bs)
        b=bs(j);
        B(j)=b;

        wSaveFileName     = strcat('masseRessort_k2_',..
            strrep(strcat(num2str(k2),'_b_',num2str(b)),'.','dec'));
        wSwitchOutputName = strcat('masseRessort_Switch_',wSaveFileName);
        load(wSaveFileName);
        wStruct = eval(wSwitchOutputName);

        Xm(i,j)=x0-min(wStruct.x_switch.Data);

         if (k2==1 && b==1)

             wIndexMin = find(wStruct.x_switch.Data==min(wStruct.x_switch.Data));

             figure();
             plot(wStruct.x_switch.Time,wStruct.x_switch.Data);
             hold('on');

             plot(wStruct.dx_switch.Time,wStruct.dx_switch.Data);
             plot(wStruct.Fs_switch.Time,wStruct.Fs_switch.Data);
             plot(wStruct.x_switch.Time(wIndexMin),..
                wStruct.x_switch.Data(wIndexMin),'ro');
             grid minor;

             title(strcat('Deplacement de la masse et efforts',..
                'appliques au cours du temps');
                (k_2=1 et b=1)');
             h = legend('x(t)','$\frac{dx}{dt}$','F_s(t)');
             set(h,'interpreter','latex');
             set(h,'position',[0.8 0.8 0.15 0.15]);
             set(gca,'fontsize',15)
             xlabel('Time (s)');

         end

    end
end
```

```
figure();
surf(K2,B,Xm);
grid minor;
legend('x_{Min}(k2,b)');
title('Deplacement maximal de la masse selon l''amortissement');
xlabel('K_2');
ylabel('b');
zlabel('Deviation maximale');

%Calcul du gradient
Gxm = gradient(Xm,1,1);

figure();
surf(K2,B,Xm);
grid minor;
h = legend('$\vec\nabla(X_{Min}(k2,b))$');
set(h,'interpreter','latex');
title('Gradient du deplacement maximal de la masse selon l''amortissement');
xlabel('K_2');
ylabel('b');
zlabel('Gradient');

wTreshold = 0.01
[r,c]=find(abs(Gxm)<wTreshold);
A=[r,c];

figure();
plot(r,c);
h = legend(strcat('Lieu des gradients < ',num2str(wTreshold)));
xlabel('K_2');
ylabel('b');
```

5. Exercice2 : Code du bloc switch

```
function Fs = fcn(u,cond)
%#codegen

if (cond > 0)
    Fs = u;
else
    Fs = 0;
end
```

——— *Fin du devoir* ———