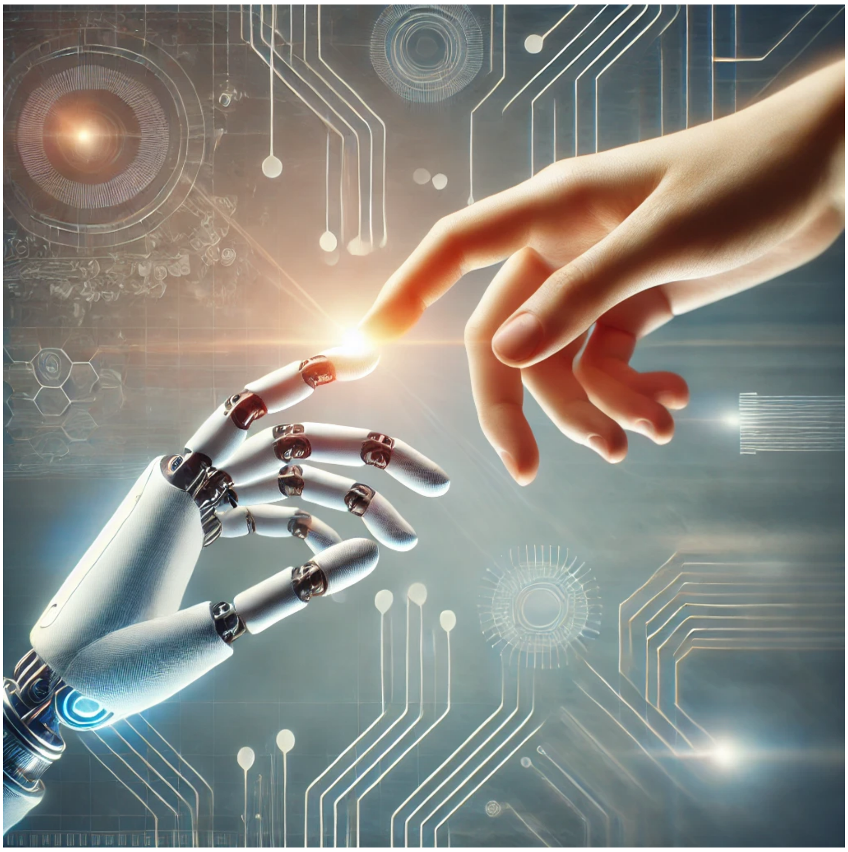
UQAC

Interaction Humain-Robot : Devoir 2



Constance ALOYAU		ALOC25530200
Erwan MAWART	16 novembre 2024	MAWE14050200
Benjamin PELLIEUX		PELB28120100

Table des matières

Table des figures

Liste des tableaux

Introduction

Les vibrations dans un mécanisme robotique manipulé par un opérateur humain peuvent significativement compromettre la précision et la qualité de la tâche effectuée. Lorsque l'opérateur applique une force via une poignée équipée d'un capteur, des vibrations non désirées peuvent survenir, surtout lorsque l'opérateur ajuste la rigidité de son bras. Ces vibrations perturbent non seulement la performance mais réduisent aussi le confort de l'utilisateur.

Dans le cadre de ce projet, nous cherchons à modéliser et analyser l'impact perceptuel de ces vibrations en interaction humain-robot. Notre approche implique le développement d'un observateur de vibrations capable de détecter et segmenter les parties du signal problématiques, permettant ainsi un ajustement dynamique d'un contrôleur proportionnel en fonction de l'indice de vibration mesuré. L'objectif est de minimiser ces vibrations, tout en tenant compte de l'expérience ressentie par l'opérateur.

Ce rapport aborde plusieurs aspects de la conception et de la simulation du modèle du mécanisme, incluant la formulation des équations d'état du système, l'implémentation d'un critère de stabilité basé sur le critère de Routh-Hurwitz, ainsi que la configuration de la boucle de contrôle pour limiter les vibrations. Nous explorons également la détection autonome des caractéristiques vibratoires à travers une simulation pour tester l'efficacité de notre modèle dans des scénarios variés.

Énoncé 1 : Conception du modèle du mécanisme

Question 1.1

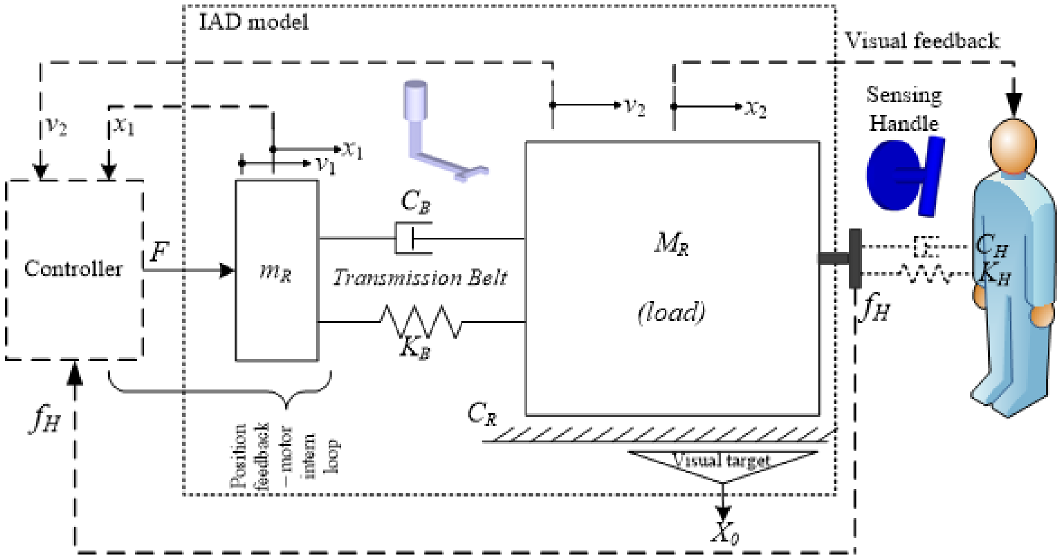


FIGURE 1 – Modèle du mécanisme robotique et de l'humain

La figure 1 présente le mécanisme robotique que nous allons utiliser, où :

- M_R : masse de la charge *[kg]*.
- m_R : masse du système *[kg]*.
- K_B : constante de raideur des courroies *[N/m]*.
- C_B : coefficient d'amortissement des courroies *[$N \cdot s/m$]*.
- K_H : coefficient de raideur de l'humain*[N/m]*.
- C_H : coefficient d'amortissement de l'humain *[$N \cdot s/m$]*.
- C_R : coefficient de frottement *[$N \cdot s/m$]*.
- F : Force commandée par le système *[N]*.
- f_H : Force appliquée par l'humain *[N]*.
- x_1 : position des moteurs *[m]*.
- x_2 : position de la charge *[m]*.
- v_1 : vitesse des moteurs *[m/s]*, la dérivée de la position ($v_1 = \dot{x_1}$).
- v_2 : vitesse de la charge *[m/s]*, la dérivée de la position ($v_2 = \dot{x_2}$).

Ainsi, on déduit les variables d'états ci-dessous.

$$X = \begin{bmatrix} x_1 \\ x_2 \\ v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \dot{x_1} \\ \dot{x_2} \end{bmatrix}$$

En effectuant la somme des forces appliquées sur chaque masse, on obtient le système suivant.

$$\begin{cases} \sum F_{ext/m_R} = m_R \ddot{x_1} \\ \sum F_{ext/M_R} = M_R \ddot{x_2} \end{cases} \Leftrightarrow \begin{cases} m_R \ddot{x_1} = F - K_B x_1 + K_B x_2 - C_B \dot{x_1} + C_B \dot{x_2} \\ M_R \ddot{x_2} = K_B x_1 - K_B x_2 + C_B \dot{x_1} - C_B \dot{x_2} - C_R \dot{x_2} \end{cases}$$

$$\Leftrightarrow \begin{cases} \ddot{x}_1 = \frac{F - K_B x_1 + K_B x_2 - C_B \dot{x}_1 + C_B \dot{x}_2}{m_R} \\ \ddot{x}_2 = \frac{K_B x_1 - K_B x_2 + C_B \dot{x}_1 - C_B \dot{x}_2 - C_R \dot{x}_2}{M_R} \end{cases} \tag{1}$$

Grâce à celle-ci, on construit les deux équations de fonctionnement du système :

$$\dot{X} = AX + BU \Leftrightarrow \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{-K_B}{m_R} & \frac{K_B}{m_R} & \frac{-C_B}{m_R} & \frac{C_B}{m_R} \\ \frac{K_B}{M_R} & \frac{-K_B}{M_R} & \frac{C_B}{M_R} & \frac{-(C_B+C_R)}{M_R} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{m_R} \\ 0 \end{bmatrix} \cdot F \tag{2}$$

$$Y = CX + DU \Leftrightarrow Y = CX \Leftrightarrow \begin{bmatrix} x_2 \\ \dot{x}_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} \tag{3}$$

L'équation 3 présente la sortie du système. Dans notre cas, on utilise x_2 (la position de la charge) et v_1 (la vitesse des moteurs) afin d'asservir notre système. Où x_2 permet à l'humain de corriger la trajectoire de la charge et v_1 d'asservir les moteurs.

Question 1.2

Lors d'une étude précédente, nous nous penchions sur la détectection des vibrations générées par le système lorsque l'humain se régédifie. Pour cela, nous analysions la vitesse des moteurs pour en extraire des caractéristiques temporelles et/ou fréquentielles qui nous permettent de déterminer la présence des vibrations.

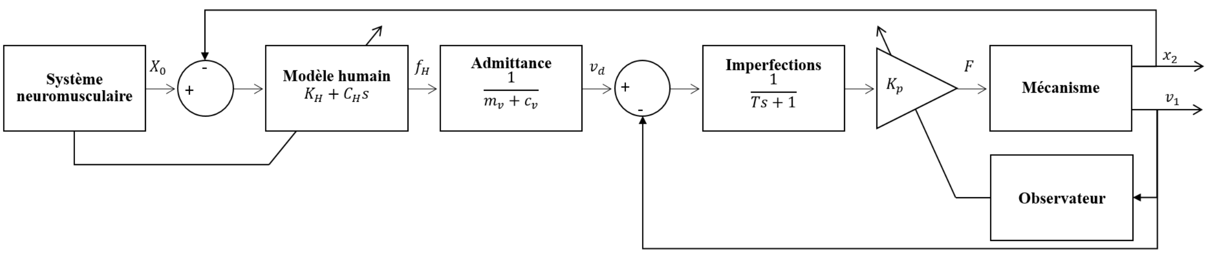


FIGURE 2 – Commande des mécanisme robotique et humain avec observateur

Afin de rendre le système autonome, il est donc nécessaire de rendre automatique cette recherche de caractéristiques. C'est pourquoi, comme le montre la figure 2, on ajoute un observateur qui va influer sur la valeur du gain K_p, et donc, limiter les vibrations générées par le système.

Dans Simulink, on représentera ce bloc avec l'architecture et la fonction contenues dans l'annexe 7.

Question 1.3

Le critère de Routh-Hurwitz est une méthode mathématique permettant déterminer rapidement la stabilité d'un système. Elle consiste à générer un tableau à partir des coefficients du dénominateur de la fonction de transfert tel que présenté au tableau 1. Une fois construit, on observe sa première colonne. Le nombre de changement de signe détermine la quantité de racines étant dans la partie astable du plan-s (c'est-à-dire dans la partie réel positive).

s^n	a_n	a_{n-2}	a_{n-4}	$\dots$
s^{n-1}	a_{n-1}	a_{n-3}	a_{n-5}	$\dots$
s^{n-2}	b_{n-1}	b_{n-3}	b_{n-5}	$\dots$
s^{n-3}	c_{n-1}	c_{n-3}	c_{n-5}	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
s^0	h_{n-1}	h_{n-3}	h_{n-5}	$\dots$

TABLE 1 – Tableau théorique du critère de Routh-Hurwitz

Dans le tableau 1, on retrouve :

- a_{n-i}, les coefficients du dénominateur.
- b_{n-i}, donnés par : $b_{n-i} = \frac{-1}{a_{n-1}} \begin{vmatrix} a_n & a_{n-i-1} \\ a_{n-1} & a_{n-i-2} \end{vmatrix} = \frac{a_{n-1}a_{n-i-1} - a_n a_{n-i-2}}{a_{n-1}}$
- c_{n-i}, donnés par : $c_{n-i} = \frac{-1}{b_{n-1}} \begin{vmatrix} a_n & a_{n-i-2} \\ b_{n-1} & b_{n-i-2} \end{vmatrix} = \frac{b_{n-1}a_{n-i-2} - a_n b_{n-i-2}}{b_{n-1}}$
- etc.

On utilise cette méthode afin de déterminer la plage de valeurs des gain K_p. Afin de construire cette table plus simplement, on utilise cette fonction dans Matlab :

Listing 1 – Fonction Matlab génération du critère de Routh-Hurwitz

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parametres :                                             %
%    - TF : la fonction de transfert a analyser            %
%    - Oldvars : les variables a remplacer                 %
%    - Newvars : les valeurs a implementer                 %
%    - s : la variable de Laplace                          %
% Sortie :                                                 %
%    - S : le tableau construit                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function S = calcTabRH(TF, Oldvars,  Newvars, s)
    % Extraction des coefficients du denominateur =====
    [~, Den] = numden(TF);
    [an, terms] = coeffs(Den, s);
    l_terms = length(terms);
    l = round(l_terms/2);

    % Creation du tableau ==============================
    S = sym(zeros(l_terms, l));
    % Ajout des coefficients du denominateur
    for n=1:l
        i = n*2;
        S(l_terms, n) = an(i-1);
        if i < l_terms
```

```
            S(l_terms-1, n) = an(i);
        end
    end
    % Calcul des autres elements du tableau
    for k=l_terms-2:-1:1
        for n=1:l-1
            S(k, n) = (S(k+1, 1)*S(k+2, n+1)-S(k+2,
               1)*S(k+1, n+1))/S(k+1, 1);
        end
    end

    % Remplacement des valeurs connues ================
    S = simplify(subs(S, Oldvars, Newvars));

    % Affichage du tableau ============================
    disp("Tableau du critere de Routh-Hurwitz :")
    disp(S)
end
```

Les gains K_p et K_H étant dépendants l'un de l'autre, leur plage de valeurs varie en fonction de l'autre. Par conséquent, il nous est obligatoire de fixer l'un des deux. Ici, fixons $K_H = 50N/m$, s'assurant ainsi la stabilité du système.

En utilisant les valeurs données en annexe 1 dans le tableau du critère de Routh-Hurwitz, on obtient donc cinq coefficients contenant K_p. Parmis ceux-ci, le critère S^2 devient négatif à $K_p = 90$, comme présenté à la figure 3. Ainsi, on déduit que $K_p \in \langle 0, 90 \rangle$.

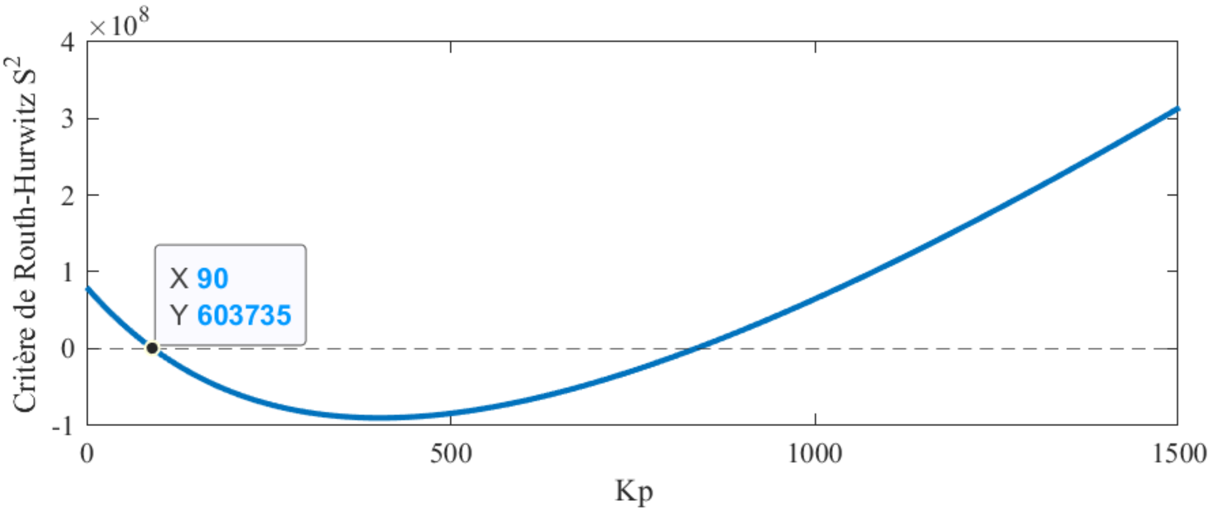


FIGURE 3 – Critère de Routh-Hurwitz S^2 en fonction de K_p

Afin de confirmer cette valeur, nous pouvons utiliser le lieu des racines. Cette méthode consiste à faire varier un gain en entrée du système afin de déterminer le moment auquel ce-dernier devient astable.

Dans notre cas, le gain variable est K_H et on fixe $K_p = 90$, condition limite déterminée à la figure 3. En traçant le lieu des racines, comme montré à la figure 4, on remarque que le système est stable jusqu'à $K_H \approx 50$ ($K_H = 46.9532$ au point de mesure noté sur la figure 4).

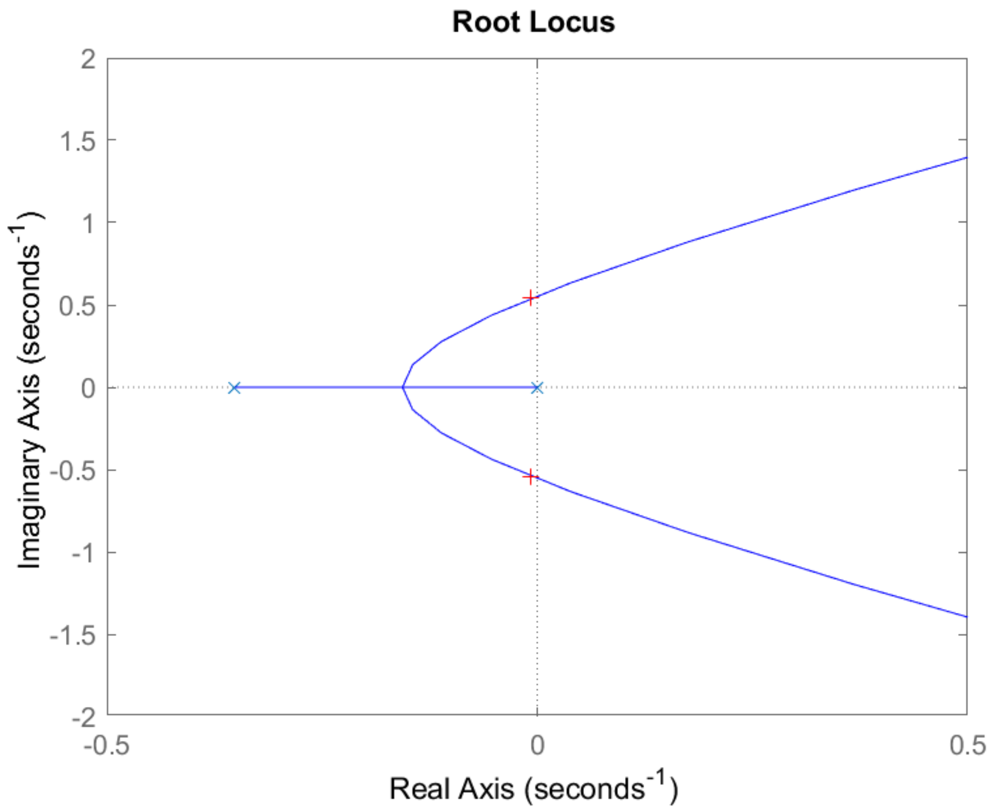


FIGURE 4 – Lieu des racines du système

Énoncé 2 : Simulation

Question 2.1

Dans le but de confirmer les plages de gains admissibles, nous allons faire varier progressivement la valeur de K_H selon une fonction rampe. Cette approche permettra d'observer l'effet de l'augmentation graduelle de la rigidité sur le système, et de déterminer précisément le seuil auquel une instabilité se manifeste.

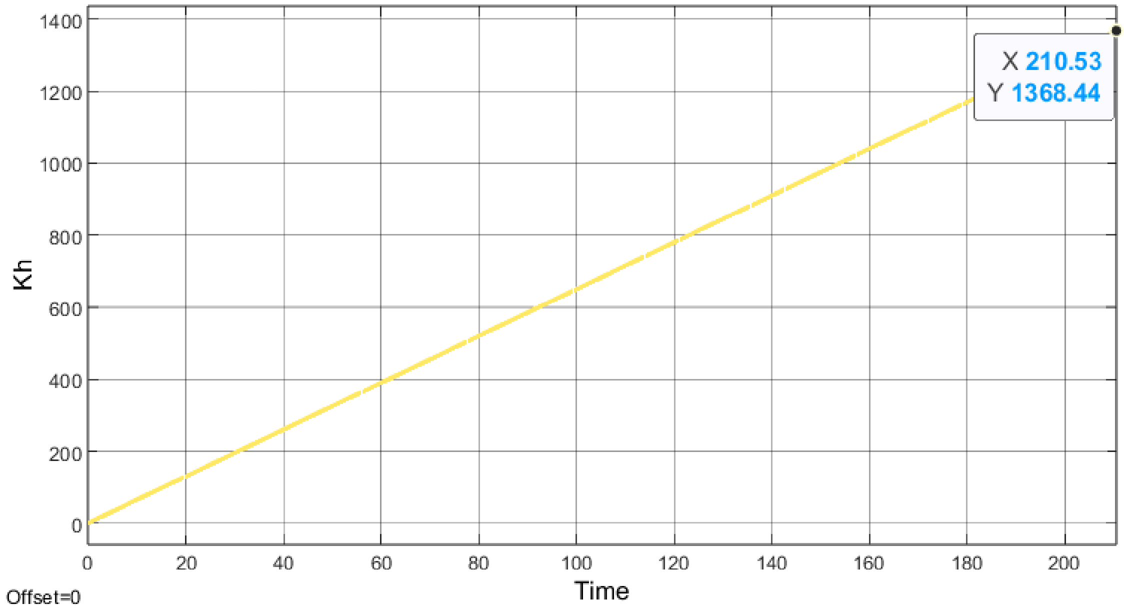


FIGURE 5 – Évolution de K_H selon une rampe

La figure 6 présente la réponse du système pour cette simulation, tandis que la figure 5 illustre l'évolution de la rigidité K_H au fil du temps. On observe que la réponse du système tend vers l'infini lorsque $K_H \approx 1368$, ce qui indique une instabilité. Cette observation permet de conclure que la plage de valeurs de K_H admettant un comportement stable est $K_H \in \langle 0, 1350 \rangle$.

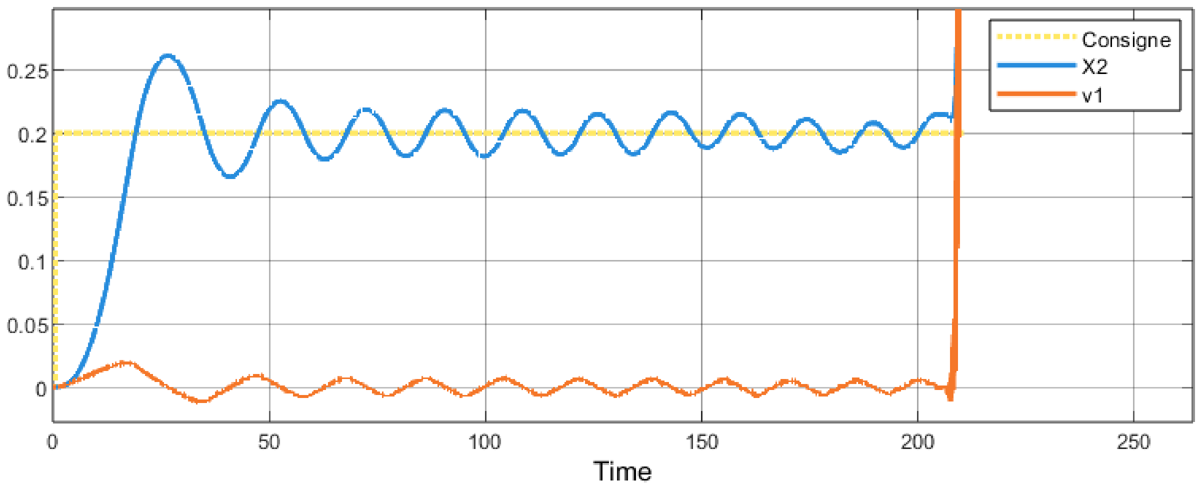


FIGURE 6 – Évolution de la réponse avec K_H selon une rampe

Question 2.2

Afin de confirmer la stabilité du système, on retire le modèle humain, comme le montre la figure 7. Cela nous permettra d'observer le fonctionnement du système sans perturbations en entrée.

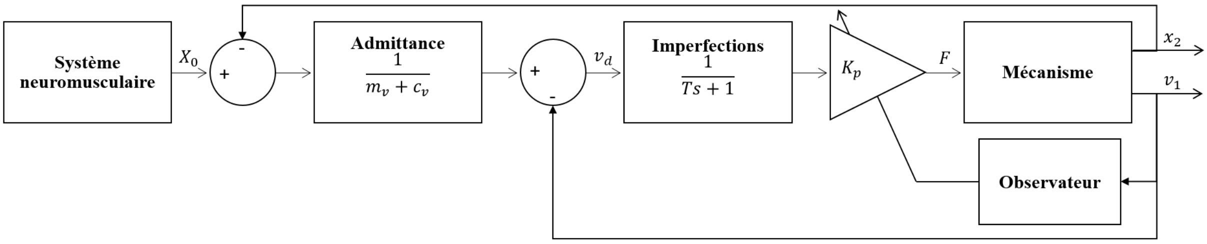


FIGURE 7 – Évolution de la réponse sans humain

La réponse obtenue présente un gain (cf. figure 8) sans oscillation, ce qui témoigne d'une stabilité satisfaisante du système. En effet, l'absence d'oscillations suggère que le système atteint son état d'équilibre sans dépasser ou fluctuer autour de la consigne, caractéristique d'un comportement stable et bien contrôlé.

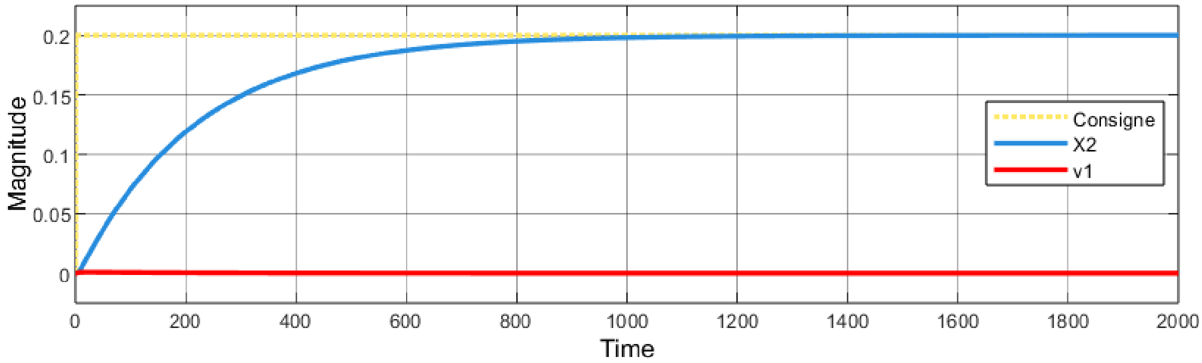


FIGURE 8 – Évolution de la réponse sans humain

Cette observation est renforcée par l'analyse du déphasage, illustrée dans la figure 9. Au cours de la simulation, aucun déphasage significatif n'est observé, ce qui confirme que la phase de la sortie suit de manière quasi-linéaire l'entrée, sans délai ou retard perceptible.

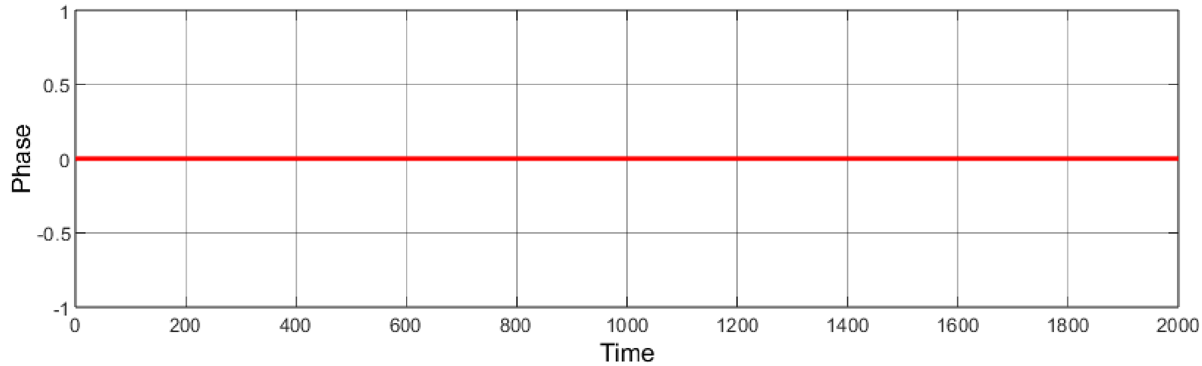


FIGURE 9 – Évolution de la phase de la réponse sans humain

Question 2.3

Notre dernière simulation consistera à appliquer différentes rigiditées humaine afin d'observer la réponse de notre système. Pour cela, on fait varier K_H selon trois valeurs définies par trois fonctions "step" configurées tel que présenté dans l'annexe 5.

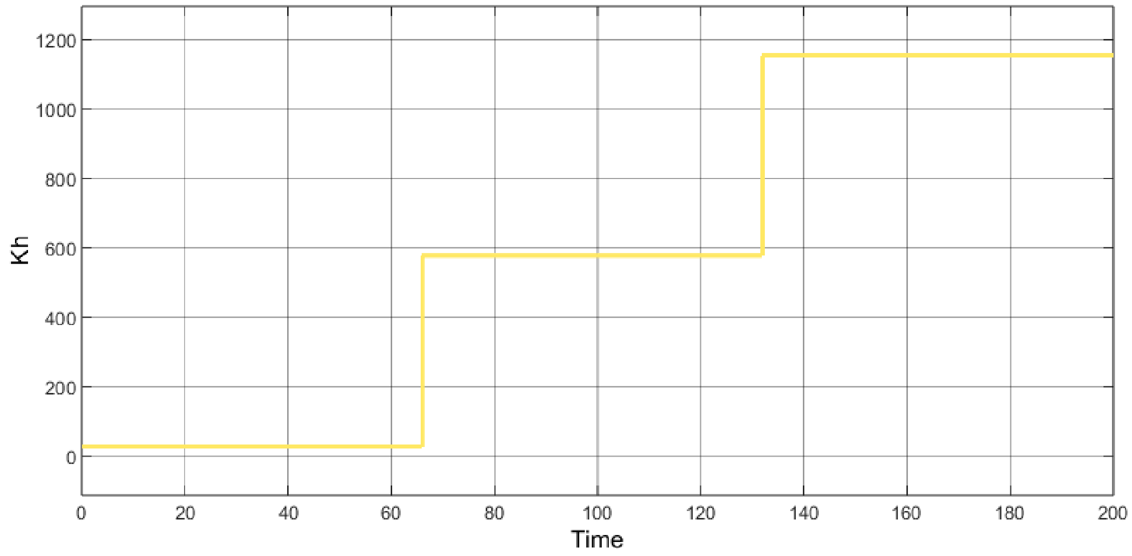


FIGURE 10 – Évolution de K_H selon trois fonctions step

Sur la réponse présentée dans la figure 11, on observe que l'amortissement du système est satisfaisant tant que la valeur de K_H reste proche de zéro. Cependant, lors de la première augmentation de la rigidité, on note l'apparition progressive d'ondulations. Dès que K_H augmente pour sa valeur finale, ces ondulations deviennent plus prononcées. De plus, des vibrations commencent à se manifester sur le signal v_1, qui représente la vitesse des moteurs.

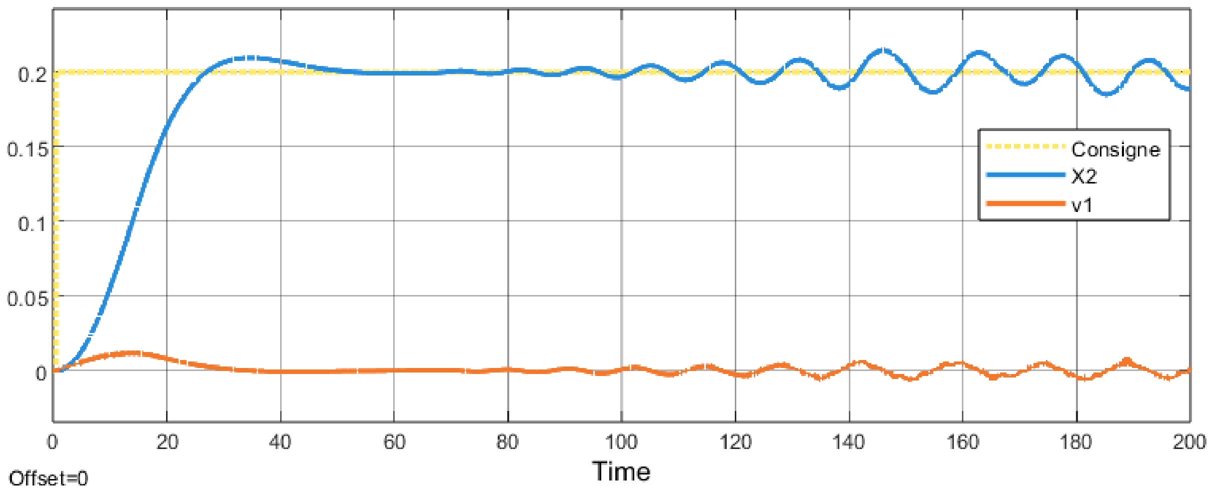


FIGURE 11 – Évolution de la réponse avec K_H selon trois fonctions step

Discussions

Conclusion

Annexes

Annexe 1 : Liste des valeurs fixes

M_R	$= 500\ kg$	*(masse de la charge)*
m_R	$= 50\ kg$	*(masse du système)*
m_v	$= 20\ kg$	*(masse ressentie)*
K_B	$= 40000\ N/m$	*(constante de raideur des courroies)*
C_B	$= 40\ N \cdot s/m$	*(coefficient d'amortissement des courroies)*
C_H	$= 23.45\ N \cdot s/m$	*(coefficient d'amortissement de l'humain)*
C_R	$= 100\ N \cdot s/m$	*(coefficient de frottement)*
c_v	$= 20\ N \cdot s/m$	*(coefficient d'amortissement vituel)*
T	$= 0.1\ s$	*(temps d'échantillonnage)*

Annexe 2 : Fichier de calculs des plages de K_p et K_H

Listing 2 – Fonction simulink de calcul du compensateur

```
% Initialisation =======================================
close all
system_config
clear Kp Kh
FontName = 'Times';

% Calcul des boucles ===================================
[Boucle_ouverte, Boucle] = Calc_Sys();

% Creation de la table de Routh-Hurwitz ================
syms Kp Kh s
S_boucle = calcTabRH(Boucle, [sym('MR') sym('mR')
   sym('Kb') sym('Cb') sym('CR') sym('T') sym('mv')
   sym('cv')], [MR mR Kb Cb CR T mv cv], s);

% Recherche de la plage de valeurs Kp =================
Kp = 0:1500;
Kh = 50;
figure;
hold on
for i = 1:length(S_boucle(:,1))
    if ~isempty(find(symvar(S_boucle(i,1)) == sym('Kp'),
       1))
        S_boucle(i,1) = subs(S_boucle(i,1), sym('Kh'),
           Kh);
        S_calc = eval(subs(S_boucle(i,1), sym('Kp'), Kp));

        subplot(3, 2, i)
        plot(Kp, S_calc, 'LineWidth', 2)
        yline(0, '--')
        xlabel('Kp')
```

```
        ylabel("Critere de Routh-Hurwitz S^" + i)
        fontname(FontName)
    end
end
hold off

% Recherche de la plage de valeurs Kh =================
TF = subs( ...
    Boucle_ouverte, ...
    [sym('MR') sym('mR') sym('Kb') sym('Cb') sym('CR')
       sym('T') sym('mv') sym('cv') sym('Kp')], ...
    [MR mR Kb Cb CR T mv cv 90] ...
);
clear s
TFFun = matlabFunction(TF);
TFFun = str2func(regexprep(func2str(TFFun),
   '\.([/^\\*])', '$1'));
figure;
sys = tf(TFFun(tf('s')));
rlocus(sys)
axis([-0.5, 0.5, -2, 2])
[K, poles] = rlocfind(sys);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parametres :                                         %
%    (Aucun)                                           %
% Sortie :                                             %
%    - Boucle_NoKh : boucle ouverte                    %
%    - Boucle1 : boucle fermee                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Boucle_NoKh, Boucle1] = Calc_Sys()
    syms MR mR Kb Cb CR Kh Kp T mv cv s

    % Calcul de la boucle de vitesse ==================
    BoucleVitesse0 = Kp*(MR*s^2+Kb+Cb*s+CR*s)*s*1 / (s *
       (mR*s^3*MR + mR*s*Kb + mR*s^2*Cb + mR*s^2*CR +
       Kb*MR*s + Kb*CR + Cb*s^2*MR + Cb*s*CR) * (T*s+1));
    BoucleVitesse1 =
       collect(simplify(BoucleVitesse0/(1+BoucleVitesse0)),
       s);

    Humain = Kh;
    Admittance = 1/(mv*s+cv);
    % Calcul de la boucle complete ====================
    Boucle0 = Humain * Admittance * BoucleVitesse1 *
       ((MR*s^2+Kb+Cb*s+CR*s)*s)^(-1) * (Kb+Cb*s);
```

```
    Boucle_NoKh = Admittance * BoucleVitesse1 *
       ((MR*s^2+Kb+Cb*s+CR*s)*s)^(-1) * (Kb+Cb*s);
    Boucle1 = collect(simplify(Boucle0*(1+Boucle0)^(-1)),
       s);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parametres :                                        %
%    - TF : la fonction de transfert a analyser        %
%    - Oldvars : les variables a remplacer             %
%    - Newvars : les valeurs a implementer             %
%    - s : la variable de Laplace                      %
% Sortie :                                            %
%    - S : le tableau construit                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function S = calcTabRH(TF, Oldvars,  Newvars, s)
    % Extraction des coefficients du denominateur =====
    [~, Den] = numden(TF);
    [an, terms] = coeffs(Den, s);
    l_terms = length(terms);
    l = round(l_terms/2);

    % Creation du tableau =============================
    S = sym(zeros(l_terms, l));
    % Ajout des coefficients du denominateur
    for n=1:l
        i = n*2;
        S(l_terms, n) = an(i-1);
        if i < l_terms
            S(l_terms-1, n) = an(i);
        end
    end
    % Calcul des autres elements du tableau
    for k=l_terms-2:-1:1
        for n=1:l-1
            S(k, n) = (S(k+1, 1)*S(k+2, n+1)-S(k+2,
               1)*S(k+1, n+1))/S(k+1, 1);
        end
    end

    % Remplacement des valeurs connues ================
    S = simplify(subs(S, Oldvars, Newvars));

    % Affichage du tableau ============================
    disp("Tableau du critere de Routh-Hurwitz :")
    disp(S)
end
```

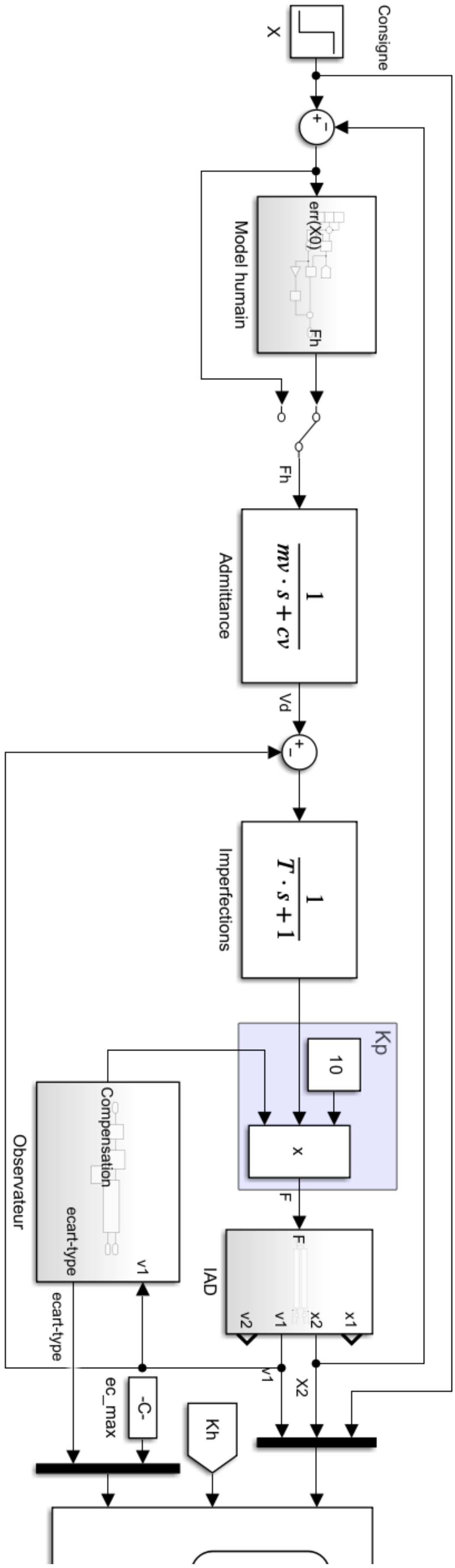
Annexe 3 : Fichier de configuration de la simulation

Listing 3 – Fonction simulink de calcul du compensateur

```
clear;
clc;

%Temps de simulation
sim_time = 200;      % s
% Taille de la memoire tampon de l'observateur
buffer_size = 128;

% Bloc "Modele humain" =================================
Kh = 550;            % N/m
Ch = 23.45;          % N*s/m

% Admittance ==========================================
cv = 20;             % N*s/m
mv = 20;             % kg

% Imperfections =======================================
T = .1;              % s

% Mecanisme ===========================================
mR = 50;             % kg
MR = 500;            % kg
CR = 100;            % N*s/m
Cb = 40;             % N*s/m
Kb = 40000;          % N/m
[A, B, C, D] = calcIAD(Kb, Cb, CR, mR, MR);

% Observateur =========================================
% Ecart-type limite avant vibrations
ec_max = 1e-3;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parametres :                                        %
%   - K : ici Kb                                      %
%   - C1 : ici Cb                                     %
%   - C2 : ici CR                                     %
%   - m : ici mR                                      %
%   - M : ici MR                                      %
% Sortie :                                            %
%   - A : la matrice d'etat                           %
%   - B : la matrice de commande                      %
%   - C : la matrice d'observation                    %
%   - D : la matrice d'action directe                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [A, B, C, D] = calcIAD(K, C1, C2, m, M)
    mK = K / m;
    mC = C1 / m;
    MK = K / M;

    A = [
         0       0       1       0;
         0       0       0       1;
        -mK      mK      -mC     mC;
         MK      -MK     C1/M    -(C1+C2)/M
        ];
    B = [
         0;
         0;
         1/m;
         0
        ];
    C = [
         0       1       0       0;
         0       0       1       0;
        ];
    D = zeros(2, 1);
end
```

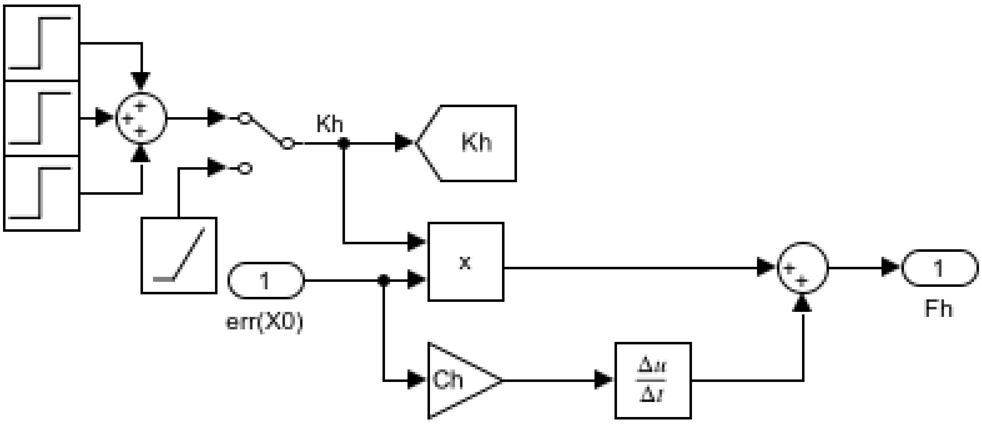
Annexe 4 : Modèle complet



Où le bloc "step" (**X**) est paramétré avec :

- "step time" = 0.5
- "initial value" = 0
- "final value" = 0.2

Annexe 5 : Modèle humain



Où :

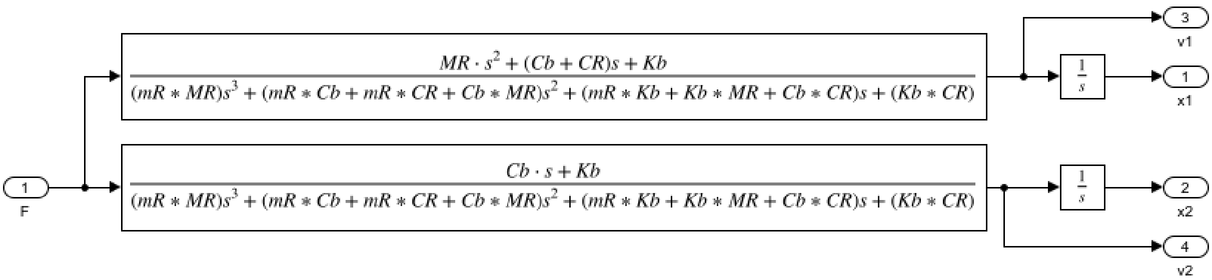
Le bloc "rate transition" est paramétré avec :

- "slope" = 6.5
- "start time" = 0
- "initial output" = 0

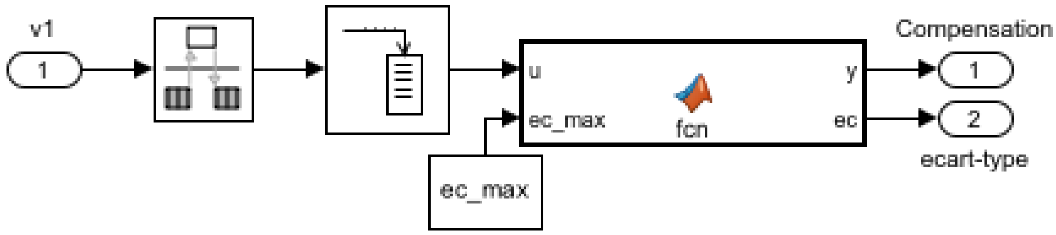
Et les blocs "step" sont paramétrés avec :

Step haut	Step milieu	Step bas
• "step time" = 0	• "step time" = 66	• "step time" = 132
• "initial value" = 0	• "initial value" = 0	• "initial value" = 0
• "final value" = 27.5	• "final value" = 550	• "final value" = 577.5

Annexe 6 : Modèle IAD



Annexe 7 : Modèle observateur



Où :

Le bloc "rate transition" est paramétré avec :

- "initial conditions" = 0
- "ouput port sample time" = 0.01

Le bloc "buffer" est paramétré avec :

- "ouput buffer size (per channel)" = 128
- "buffer overlap" = 64
- "initial conditions" = 0

Et le bloc "MATLAB Function" contient :

Listing 4 – Fonction simulink de calcul du compensateur

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parametres :                                            %
%    - u : segment de signal                              %
%    - ec_max : ecart-type limite avant vibration         %
% Sortie :                                                %
%    - y : le coefficient de compensation                 %
%    - ec : l'ecart-type du segment                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [y, ec] = fcn(u, ec_max)
    % Calcul de l'ecart-type ==========================
    ec = std(u);

    % Calcul de la compensation =======================
    y = 1 - ec * (1/ec_max);
end
```

Lien vers le Dépôt GitHub

https://github.com/BlueWan14/Cours_IHR/tree/main/Devoir_2