Rouot Emile
Frenois Etan
Groupe A
1A - *SN*

ENSEEIHT
Décembre 2023

Bureau d'études Systèmes Cyber Physique

1. Introduction

Dans la continuité du cours d'automatique, nous avons réalisé trois TPs dans le but de mettre en œuvre nos connaissances théoriques à travers un cas concret dont le but était de réussir à stabiliser un robot Lego NXT que l'on peut assimiler à un pendule inversé.
L'objectif du premier TP était de simuler le pendule inversé contrôlé par retour d'état en cherchant à stabiliser le système autour de sa position d'équilibre.
Ensuite, l'objectif du deuxième TP était d'étudier puis modéliser et simuler le robot Lego pendule inversé.
En dernier lieu, le troisième TP, consistait à implanter le système du pendule inversé dans le système physique, le robot Lego.

2. TP2 - Simulation du pendule inversé contrôlé par retour d'état et de sortie

2.1. Problème

Le système s'écrit de la manière suivante :

$$(S)\begin{cases} \dot{x}_1(t) = x_2(t) \\ \dot{x}_2(t) = \frac{g}{l}\sin(x_1(t)) - \frac{\cos(x_1(t))u(t)}{l} \\ x_1(0) = x_{0,1} = \alpha_0 \\ x_2(0) = x_{0,2} = \dot{\alpha}_0, \end{cases}$$

- $g = 9.81$;
- $l = 10$;
- $t_0 = 0$;
- $x_e = (0, 0)$;
- $u_e = 0$;
- $u(t) = u_e + K(x(t) - x_e)$
- $K = (k_1, k_2)$.

On peut écrire ce système sous la forme :

$$\begin{cases} \dot{x}(t) = f(x(t), u(t)) \\ x(0) = x_0, \end{cases} \quad \text{avec} \quad \begin{matrix} f : R^2 \times R \to R^2 \\ (x, u) \longmapsto \end{matrix} \begin{pmatrix} x_2 \\ \frac{g}{l} sin(x_1) - \frac{cos(x_1) \times u}{l} \end{pmatrix}$$

On calcule la Jacobienne de f pour $u = 0$:

$$Jf(x, 0) = \begin{pmatrix} 0 & 1 \\ \frac{g}{l} cos(x_1) & 0 \end{pmatrix}$$

Puis,

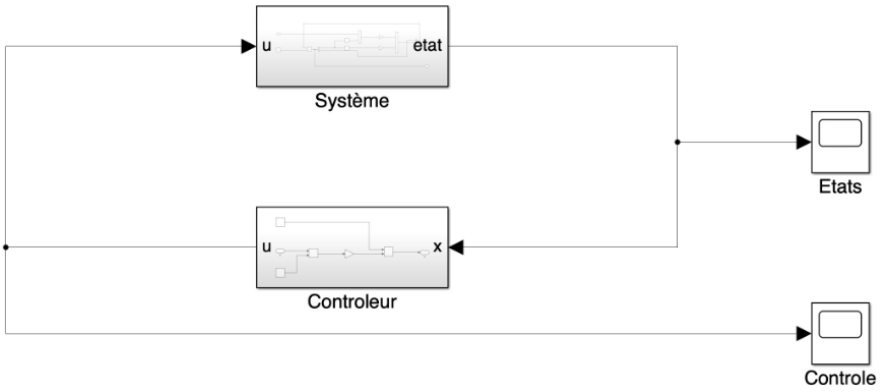
$$Jf(0, 0) = \begin{pmatrix} 0 & 1 \\ \frac{g}{l} & 0 \end{pmatrix}$$

On trouve, pour le système non contrôlé, que les valeurs propres de $J_f(0, 0)$ sont $+\sqrt{\frac{g}{l}}$ et $-\sqrt{\frac{g}{l}}$.

Nous en déduisons que le système n'est pas stable car une valeur propre est positive. C'est pour cela que dans la suite nous souhaitons ajouter un contrôle par retour d'état pour essayer de stabiliser asymptotiquement l'état du système.

2.2. Contrôle par retour d'état

En ajoutant le contrôle par retour d'état $u(t) = u_e + K(x(t) - x_e)$ puis en calculant $J_f(x,u)$, nous trouvons des conditions sur K telles que $k_1 > g$ et $k_2 > 0$. Ces conditions sont suffisantes pour contrôler asymptotiquement le système,
si $(\alpha_0, \dot{\alpha}_0)$ est suffisamment proche de x_e.
En modélisant ensuite ce système contrôlé par retour d'état à l'aide de *Simulink,* nous avons réalisé plusieurs simulations (*voir annexes 3.1.1*) avec différentes données pour le contrôle par retour d'état.



Cas	x_0	t_f	K	**Intégrateur**
Cas 1.1	$(\pi/20, 0)$	10	$(30, 10)$	`ode45`
Cas 1.2	$(\pi/20, 0)$	100	$(10, 1)$	`ode45`
Cas 1.3	$(\pi/20, 0)$	100	$(10, 1)$	Euler, `ode1`
Cas 1.4	$(\pi/20, 0)$	1000	$(10, 1)$	Euler, `ode1`
Cas 1.5	$(\pi/10, 0)$	100	$(10, 1)$	`ode45`
Cas 1.6	$(\pi/10, 0)$	100	$(30, 10)$	`ode45`

Pour le *cas 1.1*, on observe des oscillations amorties ce qui montre que le système revient vers sa position d'équilibre. Le système est donc stable asymptotiquement.

Pour les *cas 1.2 et 1.3*, on a diminué la valeur de K et augmenté le temps de simulation. On remarque que dans les deux cas, le système revient vers sa position d'équilibre mais au bout d'un temps plus long. Le système est toujours stable asymptotiquement pour ces cas aussi. Néanmoins on remarque que le système se stabilise moins rapidement que pour le *cas 1.1*. Cela s'explique par la modification du paramètre K.

Pour le *cas 1.4*, on remarque que pour un temps de simulation long ($t_f = 1000$), l'état du système diverge et ne revient pas vers sa position d'équilibre. Cela peut être dû aux valeurs des paramètres $x_0 \, et \, K$.

Ensuite, pour le *cas 1.5*, la position initiale du système est plus éloignée de sa position d'équilibre. On observe que le système ne revient pas vers sa position d'équilibre et ne stabilise pas à cause de sa position initiale qui est trop éloignée de sa position d'équilibre. De plus, le temps de la simulation est assez court.

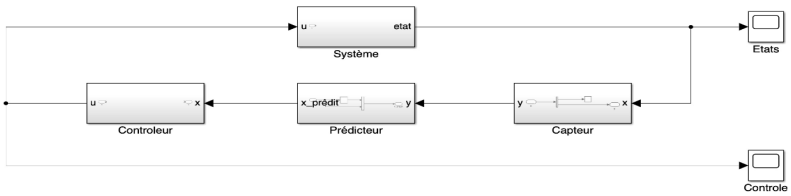
Enfin, pour le *cas 1.6*, on a gardé la même position initiale que le cas *1.5* mais on a augmenté la valeur de K. Le système parvient quand même à revenir vers sa position d'équilibre et le système se stabilise asymptotiquement.

En définitive, la stabilité d'un système va dépendre de plusieurs paramètres comme sa position initiale et la valeur de K.
Prenons l'exemple du robot NXT Lego que l'on peut assimiler au système du pendule inversé. Si la position initiale du robot est très éloignée de sa position d'équilibre et qu'on lui donne un coup, il n'arrivera pas à revenir vers sa position d'équilibre et se stabiliser. Par ailleurs, si sa position initiale est proche de sa position d'équilibre et qu'on lui donne un coup, le robot sera capable de revenir vers sa position d'équilibre et donc de se stabiliser. De même si la position initiale du robot est éloignée de sa position d'équilibre mais qu' aucun facteur extérieur agit sur le système, il sera également capable de revenir vers sa position d'équilibre et de se stabiliser. La longueur du robot (= la longueur du pendule inversé) peut également avoir un impact quant à la stabilité du système et les différentes contraintes que l'on exerce sur celui-ci.

2.3. Contrôle par retour de sortie avec prédiction de l'état

Dans cette partie, on suppose que l'on a accès qu'à $\dot{\alpha}_0$. On décide donc d'introduire deux sous systèmes: un capteur et un prédicteur pour reconstruire α.

En modélisant ensuite ce système contrôlé par retour de sortie avec prédiction de l'état à l'aide de *Simulink,* nous avons réalisé trois simulations (*voir annexes 3.1.2*) avec les mêmes données pour le contrôle mais avec des pas et des intégrateurs différents selon les cas.



Cas	x_0	t_f	K	pas	Intégrateur
Cas 1	$(\pi/20, 0)$	100	$(10, 1)$	par défaut	`ode45`
Cas 2	$(\pi/20, 0)$	100	$(10, 1)$	0.001	Euler, `ode1`
Cas 3	$(\pi/20, 0)$	100	$(10, 1)$	5	Euler, `ode1`

Pour le *cas 1*, on utilise l'intégrateur *ode45* et on observe des oscillations amorties ce qui montre que le système revient vers sa position d'équilibre. Le système est donc stable asymptotiquement.

En revanche, pour les *cas 2 et 3*, on utilise un intégrateur *Euler.*
Pour le *cas 2*, on définit un pas de 0.001 et on observe également des oscillations amorties. Le système retourne donc vers sa position d'équilibre et il est stable asymptotiquement.
Néanmoins, pour le *cas 3*, on définit un pas de 5 et on observe un état de sortie discret contrairement au deux cas précédents où l'on observait des états de sorties continues.
Dans ce *cas 3*, le système ne parvient pas à revenir vers sa position d'équilibre et n'est pas stable asymptotiquement. Cependant, on peut supposer que pour un intervalle de temps plus élevé, le système parviendra à revenir vers sa position d'équilibre car on observe un début d'oscillations amorties.

On peut en déduire que, pour notre système, la définition du pas pour l'intégrateur *Euler*, impacte directement la stabilité du système. En effet, en utilisant un pas important, l'intégrateur *Euler* est moins précis que lorsqu'on utilise un pas faible.

3. TP3 - Simulation du robot Lego

L'objectif de ce TP était d'étudier le modèle du robot Lego pendule inversé. Nous avons construit un modèle *Simulink* continu comportant un système et un contrôleur par retour d'état.

3.1. Modèle continue

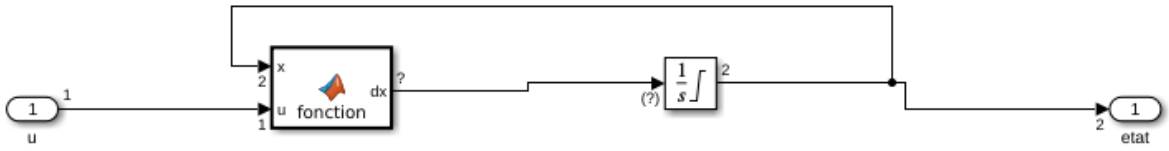
3.1.1. Utilisation de Matlab pour représenter le modèle du robot Lego pendule inversé

Pour modéliser le robot Lego, on a repris le modèle du pendule inversé du TP2 avec le système du robot représenté par les équations dynamiques suivantes :

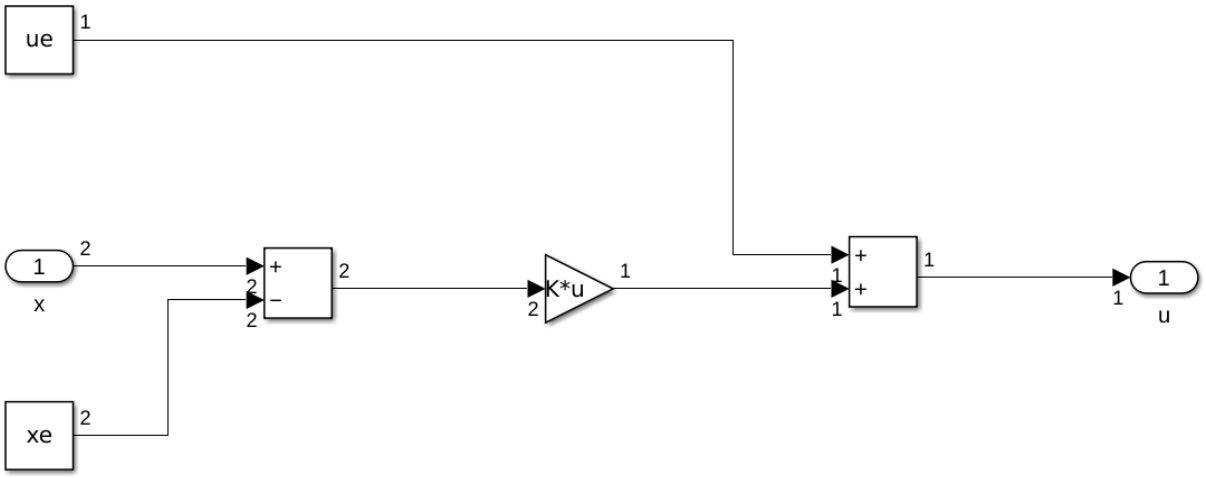
Ces équations du bloc *fonction* nous donne la dérivé de l'angle que l'on souhaite visualiser, donc on a ajouté un bloc $\frac{1}{s}$ pour intégrer :

```
% Equations of motion
c1 = (2*m+M)*R^2 + 2*Jw + 2*n^2*Jm;
c2 = M*L*R*cos(psi) - 2*n^2*Jm;
c3 = -M*L*R*dpsi^2*sin(psi);
c4 = M*L^2 + Jpsi + 2*n^2*Jm;
c5 = -M*g*L*sin(psi);

alpha = Kt/Rm;
beta = Kt*Kb/Rm + fm;

Ftheta = alpha*(2*u) - 2*(beta+fw)*dtheta + 2*beta*dpsi;
Fpsi = -alpha*(2*u) + 2*beta*dtheta - 2*beta*dpsi;

ddtheta = (c4*c3 - c2*c5 + c4*Ftheta - c2*Fpsi)/(c4*c1 - c2^2);
ddpsi = (Fpsi- c2*ddtheta - c5)/c4;
dx = [dtheta dpsi ddtheta ddpsi]';
```

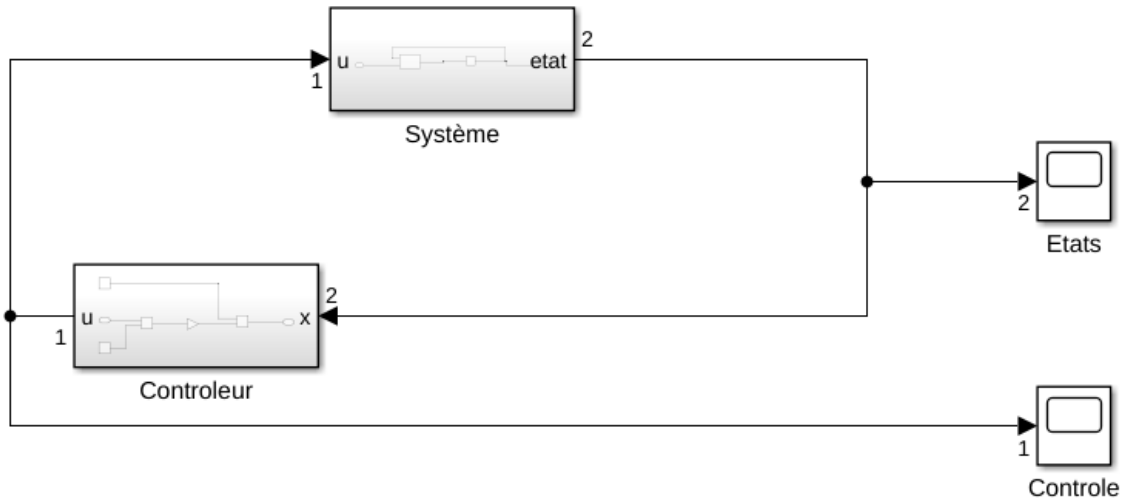
```
% State
psi = x(2);
dtheta = x(3);
dpsi = x(4);

% Parameters

g = 9.81;
m = 0.03;
R = 0.04;
Jw = m*R^2/2;
M = 0.6;
H = 0.144;
L = H/2;
Jpsi = M*L^2/3;
Jm = 1e-5;
Rm = 6.69;
Kb = 0.468;
Kt = 0.317;
n = 1;
fm = 0.0022;
fw = 0;
```



Pour le contrôleur, il est de la forme $u(t) = u_e + K(x(t) - x_e)$ d'où le modèle *Simulink* :



La construction du modèle *Simulink* continu comportant un système et un contrôleur par retour d'état est donc :



La simulation de ce modèle donne la *figure 0 (voir annexe 4.2 -TP3)* où K n'a pas été initialisé donc on ne peut pas conclure pour le moment.

3.1.2. Synthèse du contrôleur pour le modèle du robot Lego pendule inversé

Après avoir linéarisé le système contrôlé, sous le retour d'état $u = -Kx$, la dynamique en boucle fermée est donnée par $x = (A - BK)x$. Les matrices A et B nous ont été données et on peut calculer la matrice K pour les valeurs propres V (-136.5905, -2.6555, -3.5026, -5.9946) avec la fonction place. Elle place les pôles en fonction des valeurs de A et B.

On simule alors le modèle contrôlé pour différentes valeurs de positions angulaires initiales.

Cas	x_0	t_f	K	**Intégrateur**
Cas 1.1	$(0, 0, 0, 0)$	2	K	ode45
Cas 1.2	$(0, \pi/5, 0, 0)$	5	K	ode45
Cas 1.3	$(0, \pi/10, 0, 0)$	5	K	ode45

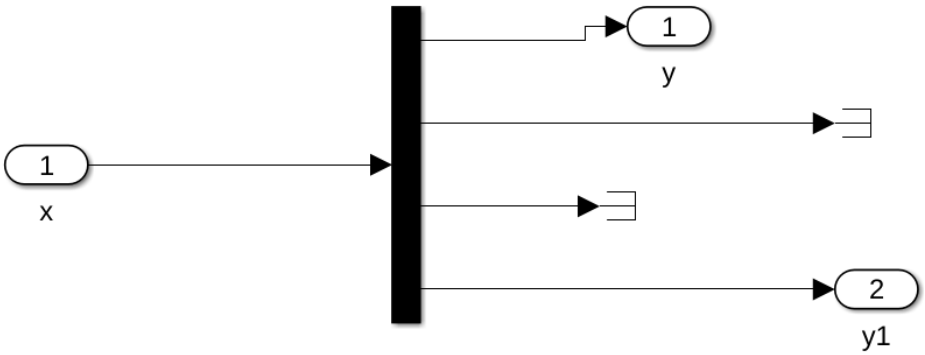
Pour le *cas 1.1* où $x_{0cas1} = (0, 0, 0, 0)$, le système commence à son point d'équilibre et il y reste. On en déduit donc que $x_e = (0, 0, 0, 0)$.

On teste alors avec des valeurs de x_0 au voisinage de $x_e = (0, 0, 0, 0)$ pour tester la stabilité de ce point d'équilibre.

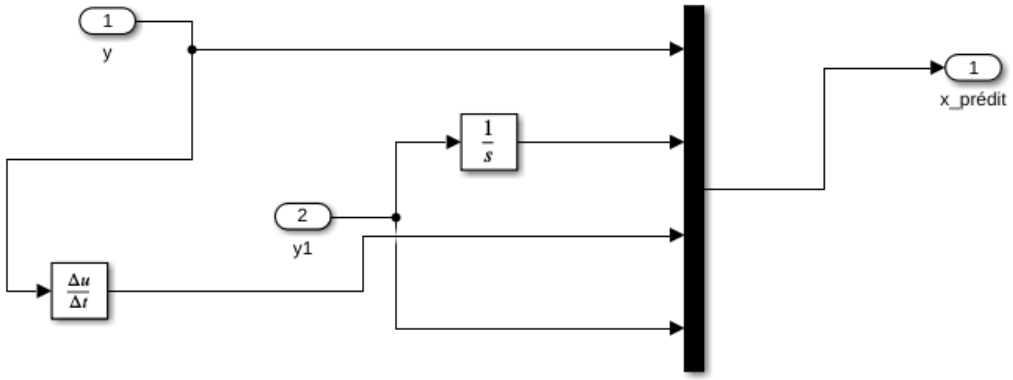
Pour les *cas 1.2 et 1.3*, le système débute avec un décalage angulaire de $\frac{\pi}{5}$ pour le *1.2* et de $\frac{\pi}{10}$ pour le *1.3*. On remarque pour les deux cas que le système revient à son état d'équilibre x_e donc on peut conclure qu'il existe un voisinage V autour de x_e tel que tel que, pour tout $x_0 \in V$, $lim_{t\to+\infty} x(t, x_0) = x_e = (0, 0, 0, 0)$ donc x_e est un équilibre asymptotiquement stable pour ce contrôle. De plus, on remarque que la réponse du système est la même pour les deux cas, à une échelle différente. En effet, la courbe bleu de position a une amplitude deux fois plus élevée lorsque $x_{0cas2} = (0, \frac{\pi}{5}, 0, 0)$ que quand $x_{0cas3} = (0, \frac{\pi}{10}, 0, 0)$, car $x_{0cas2} = 2x_{0cas3}$. Cela correspond à une légère déviation angulaire par rapport à la verticale du robot, comme s' il avait subi un léger choc et qu'il s'est remis verticalement et les roues se sont remises dans leur position initiale.

3.2. Introduction des capteurs et actionneurs

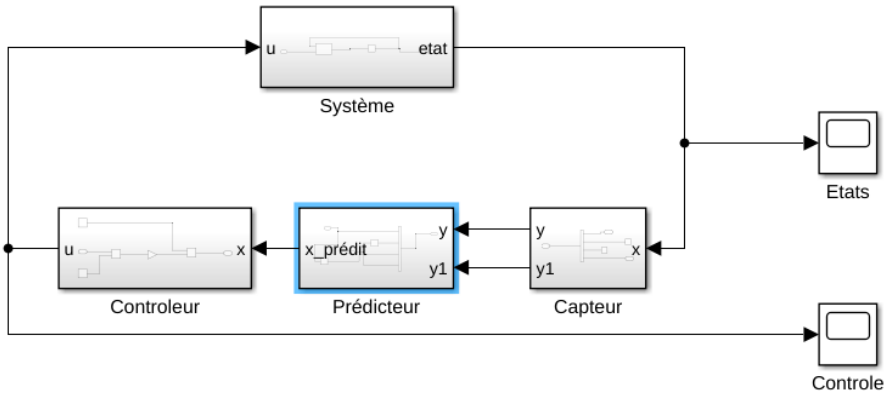
On ajoute maintenant au robot un gyroscope et un capteur qui mesure la vitesse de l'angle $\psi(t)$ et la position. Le modèle *Simulink* du capteur doit donc garder la première et la dernière composante de $x(t)$ en sortie:



Ce bloc est suivi d'un bloc prédicteur qui doit restaurer les valeurs que le capteur ne nous a pas fourni, c'est à dire le dérivé de $\theta(t)$ et $\psi(t)$. Pour cela on intègre et dérive respectivement les deux données renvoyées par le capteur :



Enfin, la construction du modèle *Simulink* continu comportant un système, un contrôleur par retour d'état et un capteur est :



On simule alors le modèle contrôlé avec capteur pour différentes valeurs de positions angulaires initiales.

Cas	x_0	t_f	K	Intégrateur
Cas 1.1	$(0, 0, 0, 0)$	2	K	ode45
Cas 1.2	$(0, \pi/5, 0, 0)$	5	K	ode45
Cas 1.3	$(0, \pi/10, 0, 0)$	5	K	ode45

Pour le *cas 1.1* où $x_{0cas1} = (0, 0, 0, 0)$, le système commence à son point d'équilibre et il y reste. On en déduit donc que $x_e = (0, 0, 0, 0)$ est un point d'équilibre.

On teste alors avec des valeurs de x_0 au voisinage de $x_e = (0, 0, 0, 0)$ pour tester la stabilité de ce point d'équilibre.

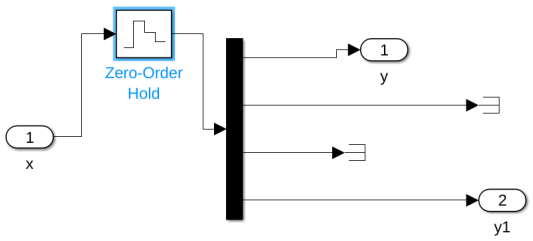
Pour les *cas 1.2 et 1.3*, le système débute avec un décalage angulaire de $\frac{\pi}{5}$ pour le *1.2* et de $\frac{\pi}{10}$ pour le *1.3*. On remarque que dans les deux cas la position angulaire des roues ne converge pas vers 0 mais vers une valeur. On remarque de plus que lorsque $x_0$est divisé par deux, la réponse de $x(t, x_0)$ est aussi divisé par deux. On en conclut que pour tout $\epsilon > 0$, il existe $\delta > 0$, tel que

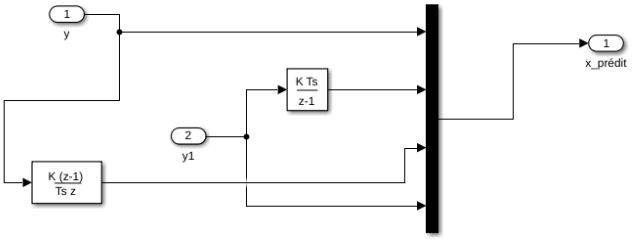
$\|x_0 - x_e\| < \delta \, et \, t > 0 \Rightarrow \|x(t, x_0) - x_e\| < \varepsilon$,car il suffit de prendre un δ assez petit pour que $||x(t, x_0)|| < \varepsilon$. Donc l'origine est stable mais pas asymptotiquement.

Cela correspond à une légère déviation angulaire par rapport à la verticale du robot, comme s' il avait subi un léger choc et qu'il s'est stabilisé verticalement mais qu'il a dû avancer un peu.

3.3. Construction du modèle hybride

Dans la suite, le contrôleur et le prédicteur seront implantés en logiciel dans le robot Lego, donc sous la forme d'un modèle discret or, le système représentant la physique reste un modèle continu. C'est pour cela qu'on introduit dans le capteur un bloc *Zero-Order Hold* dans le but de reconstruire en sortie du capteur un état discret. On a ensuite modifié le prédicteur pour utiliser des opérateurs discrets (*voir ci-dessous)*. Enfin nous avons simulé le modèle contrôlé dans le cas *1.3* (*voir annexe 4.2.2 - Cas modèle hybride).*





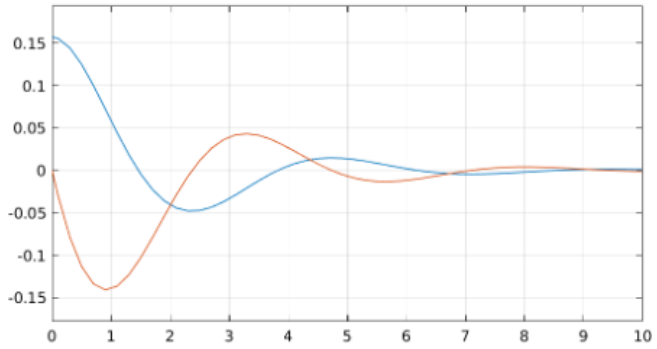
On remarque alors que pour le modèle hybride, on obtient quasiment les mêmes résultats car les courbes prennent presque les mêmes valeurs, la différence est due à la discrétisation du capteur. On en déduit que ce modèle est cohérent.

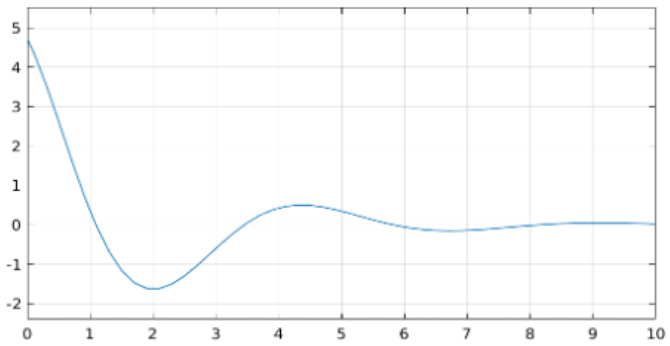
4. Annexes

4.1. TP2

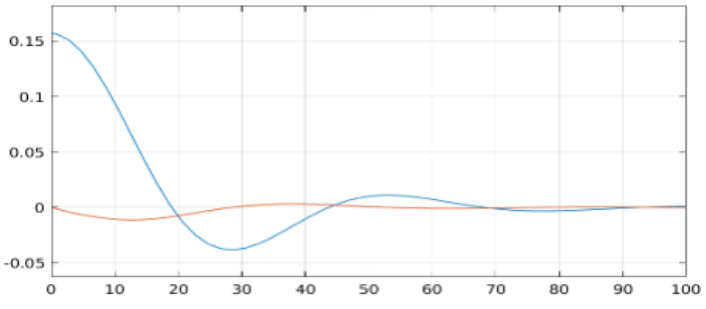
4.1.1. Table 1

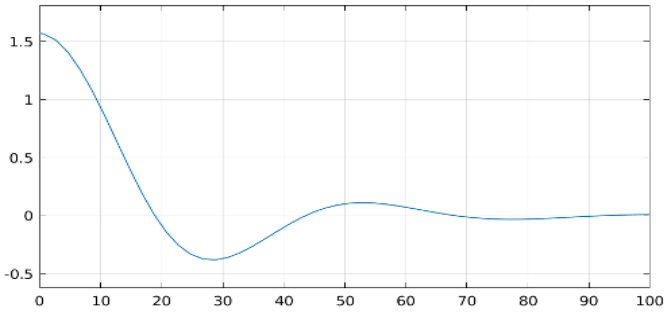
Cas 1.1



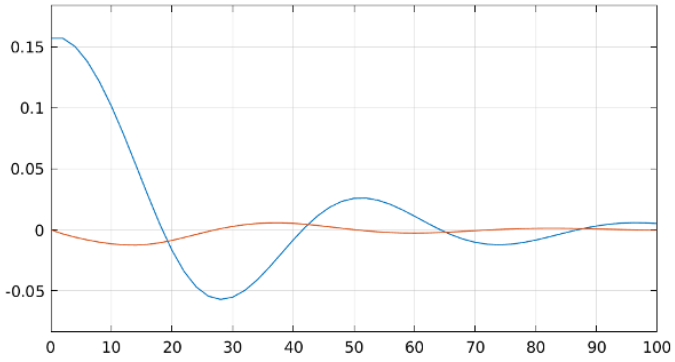


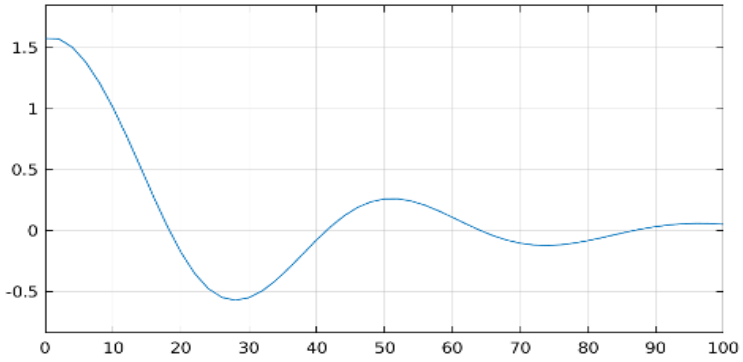
Cas 1.2



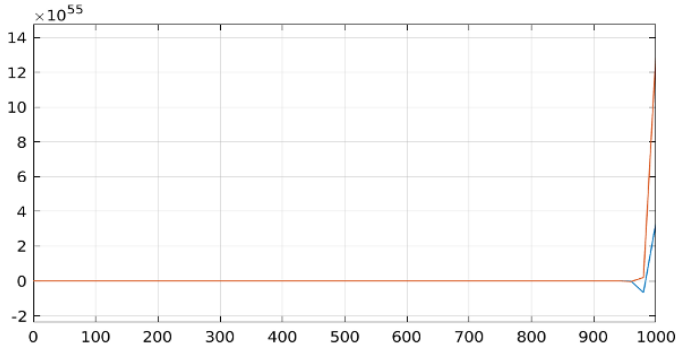


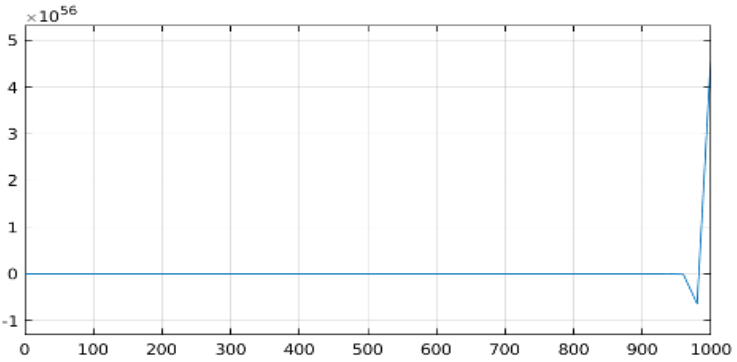
Cas 1.3



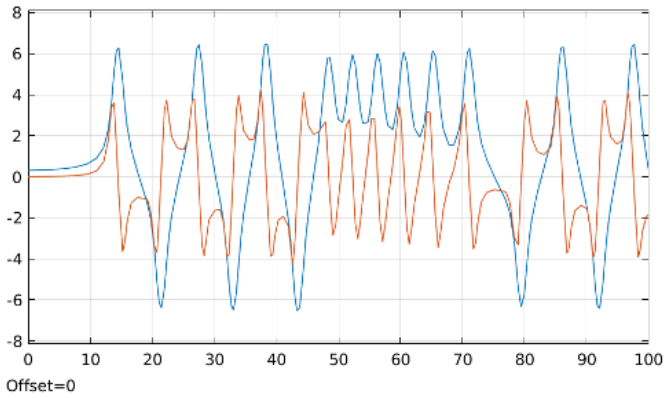


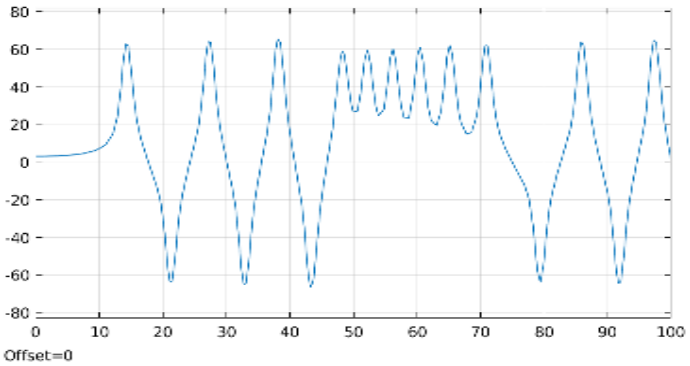
Cas 1.4



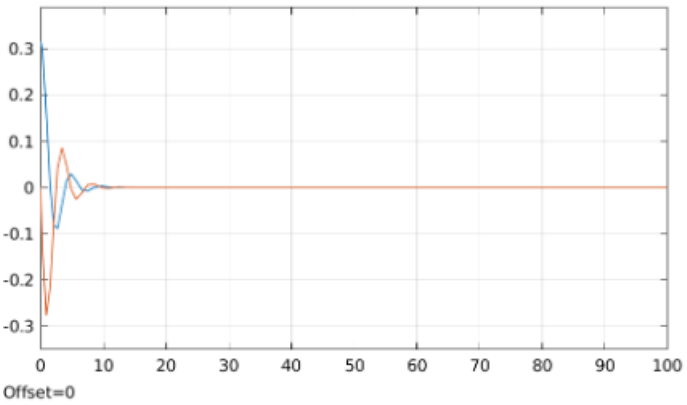


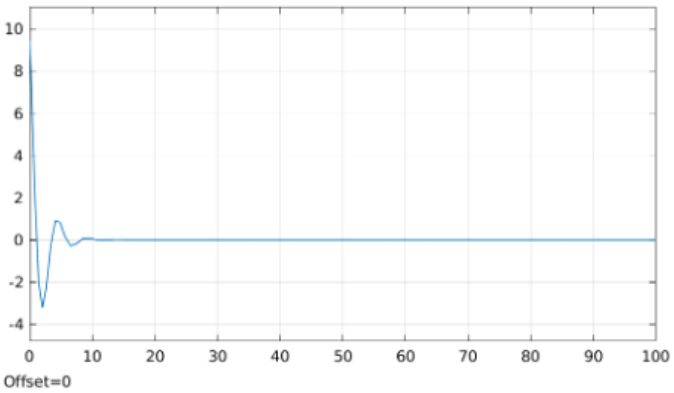
Cas 1.5





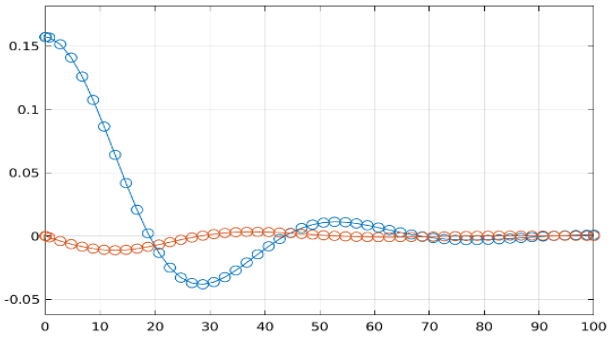
Cas 1.6

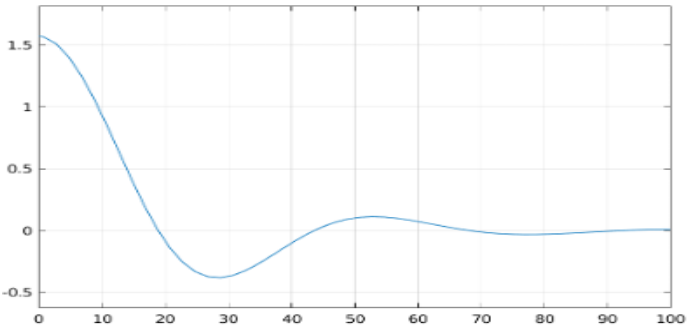




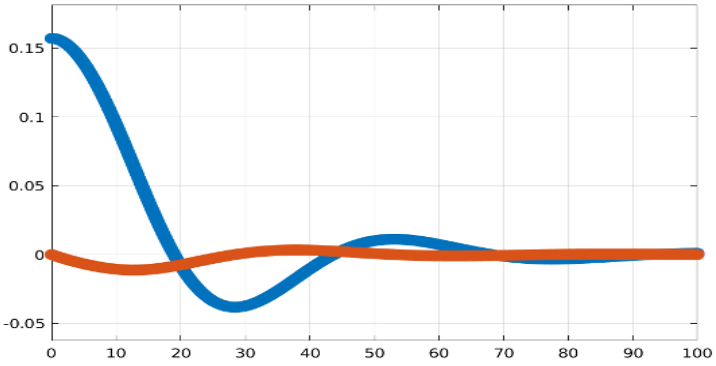
4.1.2. Table 2

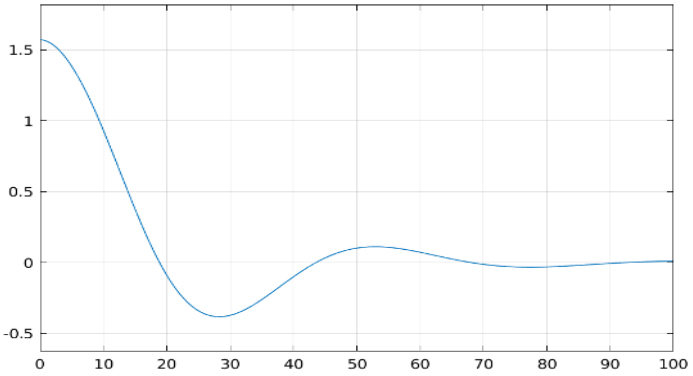
Cas 1



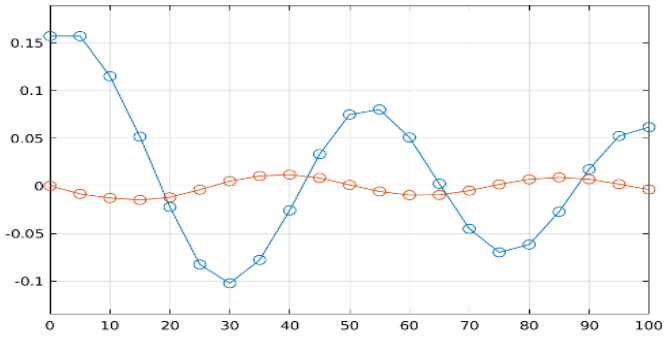


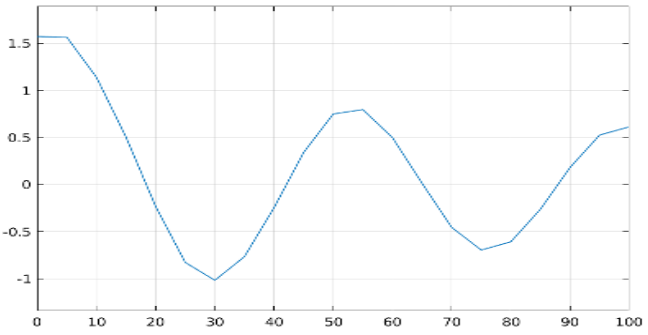
Cas 2





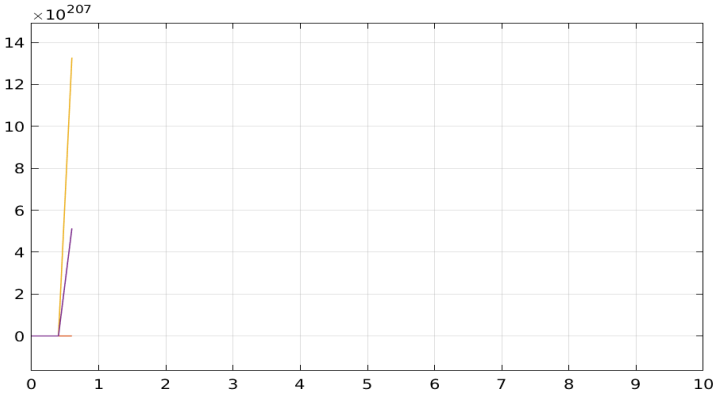
Cas 3





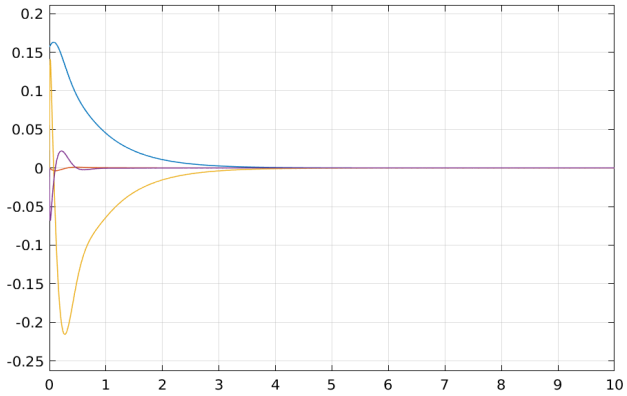
4.2. TP3

Figure 0

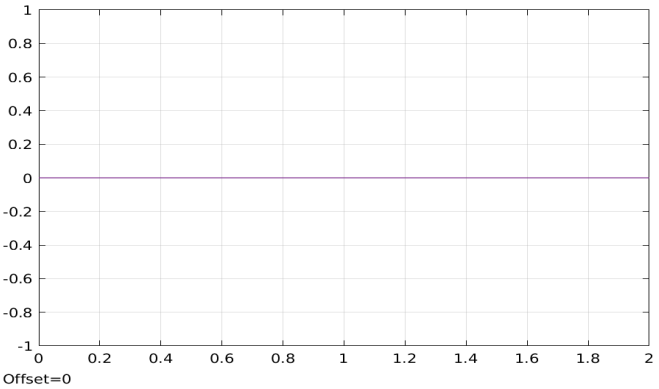


4.2.1. Table 1

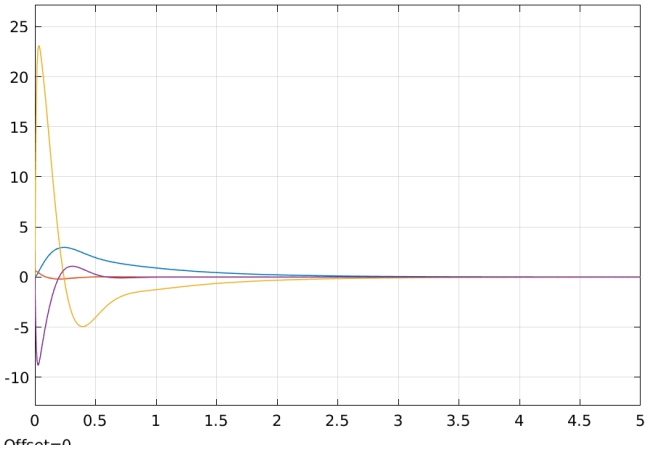
Cas 0



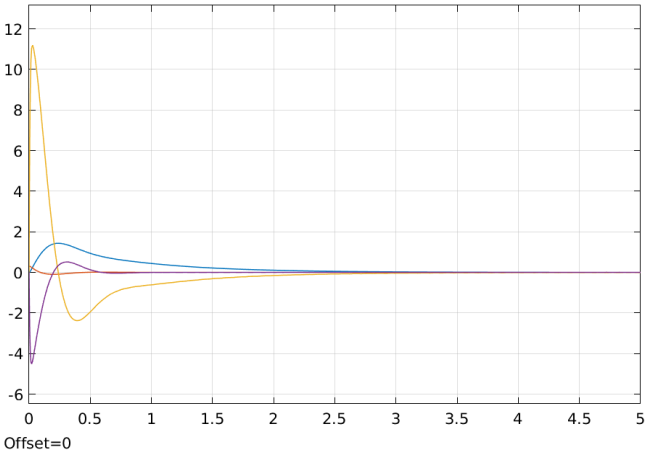
Cas 1.1



Cas 1.2

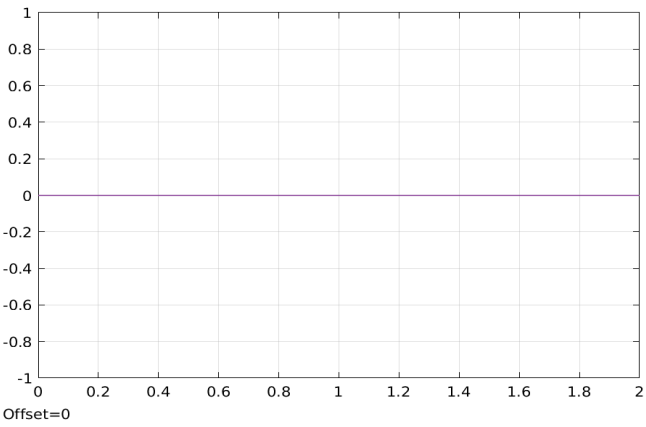


Cas 1.3

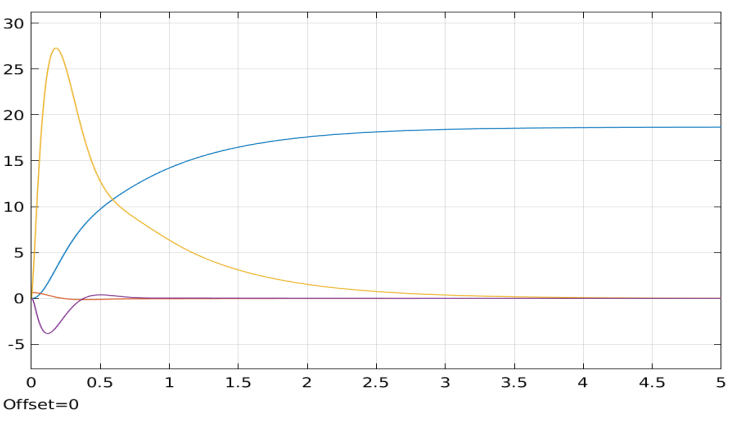


4.2.2. Table 2

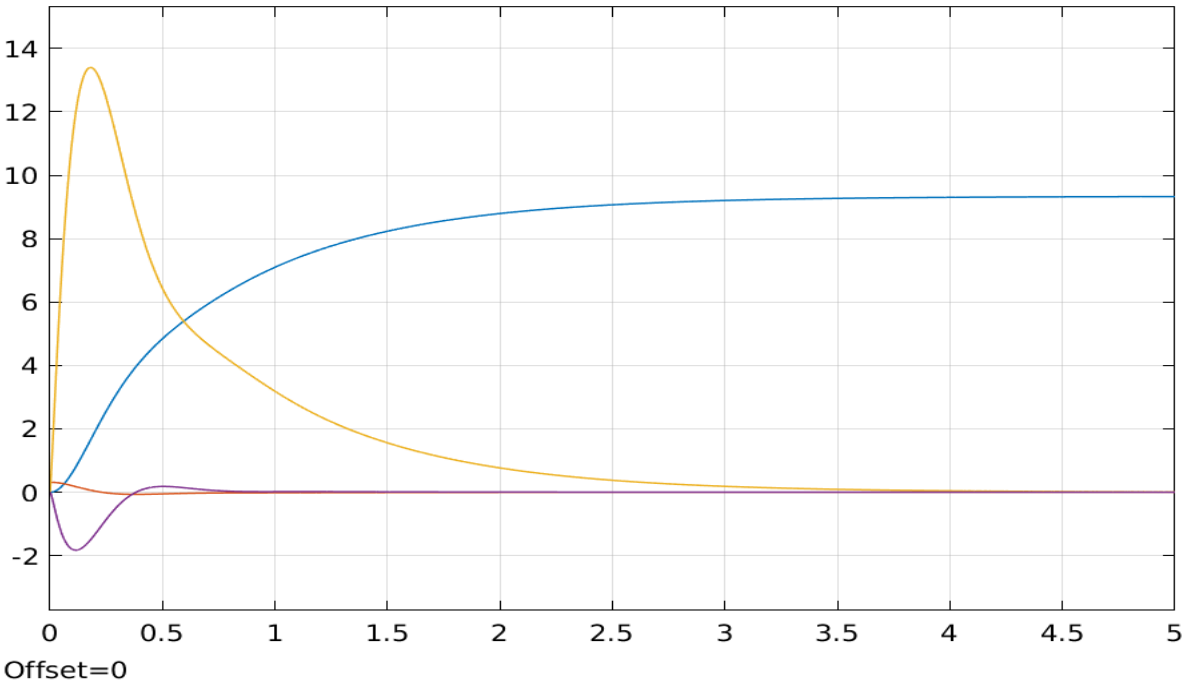
Cas 1.1



Cas 1.2



Cas 1.3



Cas modèle hybride