Sakun Withanage Perera Groupe AB
Arthur Bongiovanni

BE Automatique

Plan:

- Objectif/intro
- Analyse théorique (TD03)
- Simulation modèle simplifié (TP02)
- Simulation modèle plus complexe(TP03)
- Conclusion (ouverture TP4)

Objectif :

On souhaite étudier le comportement d'un petit robot et essayer de trouver des paramètres lui permettant de rester en équilibre. Pour ce faire, on commencera par le simplifier à un pendule inversé pour une analyse théorique, avant d'effectuer des simulations de ce modèle et de vérifier nos résultats théoriques. Enfin on simulera notre robot par un système plus complexe, avant d'implémenter un code correspondant au modèle final directement dans la machine et de vérifier si effectivement le robot parvient à rester en équilibre, même avec un point de départ incliné.

TP02:

Nous étudions la situation suivante:

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

avec

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

On est ici dans un cas non linéaire, on peut réécrire le système sous la forme:

$f: R^3 \rightarrow R^2$

(x,u) $\rightarrow \left(x_2, \frac{g}{l}sin(x_1) - \frac{cos(x_1)u}{l}\right)$

par un simple calcul (f(x,u)=0) nous pouvons obtenir les points de fonctionnements qui sont: $(x_e, u_e) = (x_{e,1}, 0, g\ tan(x_{e,1}))$ avec $x_{e,1} \in R/\left\{\frac{\Pi}{2} + K\Pi\right\}$

Comme le système n'est pas linéaire on ne pourra stabiliser le système seulement autour de ces points de fonctionnements.

Contrôle par retour d'état

on pose u(t) = $u_e + K(x(t) - x_e)$

D'après le cours un système non linéaire se stabilise si et seulement si f'(x_e) est à valeur propres strictement négatives.

$$f'(x_e) = \frac{\partial f(x_e,u_e)}{\partial x} + \frac{\partial f(x_e,u_e)}{\partial u} K$$

On a : $$f'(x_e) = \begin{pmatrix} 0 & 1 \\ \frac{\cos(x_{e,1})}{l}(g - k_1) + \frac{\sin(x_{e,1})u_e}{l} & -\frac{\cos(x_{e,1})}{l} k_2 \end{pmatrix}$$

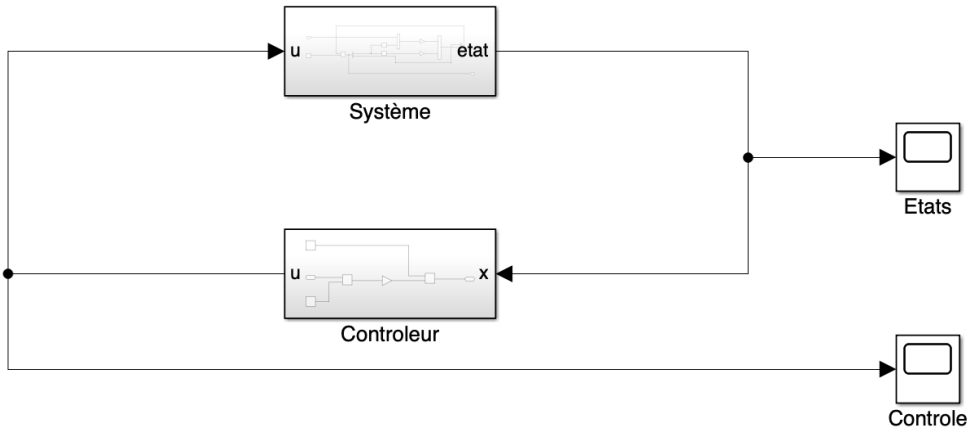
$$\begin{cases} \det(f'(x_e)) = \frac{\cos(x_{e,1})}{l}(k_1 - g) - \frac{\sin(x_{e,1})u_e}{l} > 0 \\ tr(f'(x_e)) = -\frac{\cos(x_{e,1})}{l} k_2 < 0 \end{cases}$$

On a donc k1> $g + tan(x_{e,1})\, u_e$ et k2>0

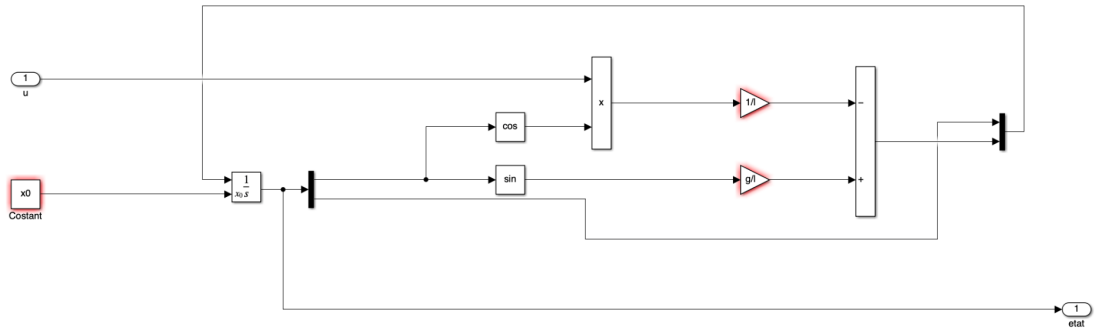
Puis avec les données du TP ((x_e , u_e) = (0,0,0)) nous avons k1>g et k2>0

Cependant nous voyons lors de ce TP que la stabilité peut différer selon certains facteurs comme X0, l'intégrateur et bien sûr K.

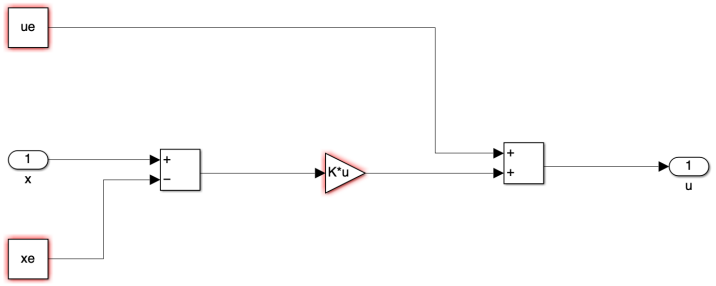
Pour cela, nous allons mettre en place le bloc suivant :



Dans le sous bloc “Système” nous modélisant l’équation du système:

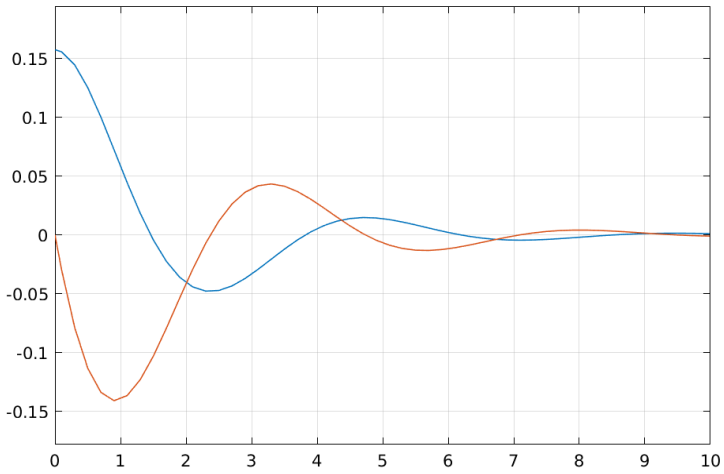


Puis dans le sous-bloc Controle l’équation correspondant au contrôleur:



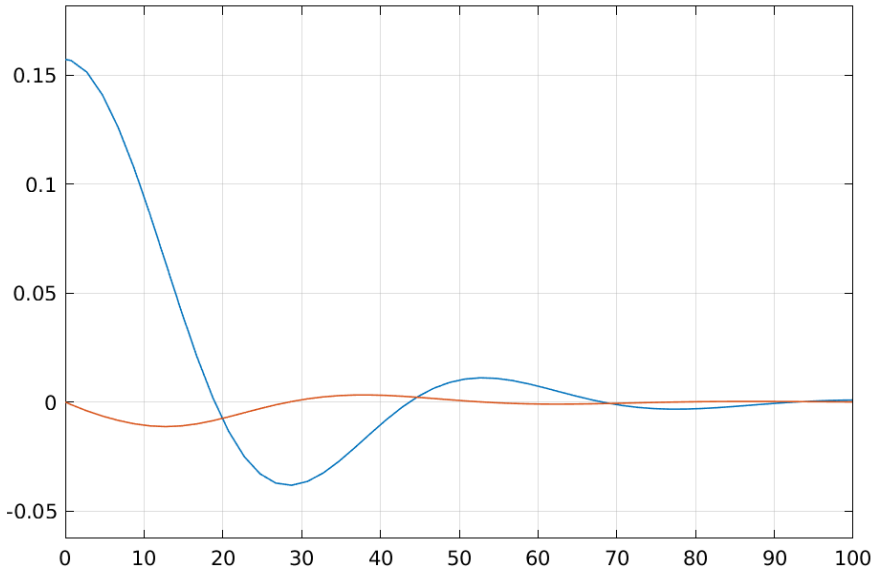
Nous allons jouer principalement avec trois paramètres lors de ce TP:

- X0 : la position initiale
- tf : le temps d’exécution totale
- K= (k1, k2) : la matrice de contrôle



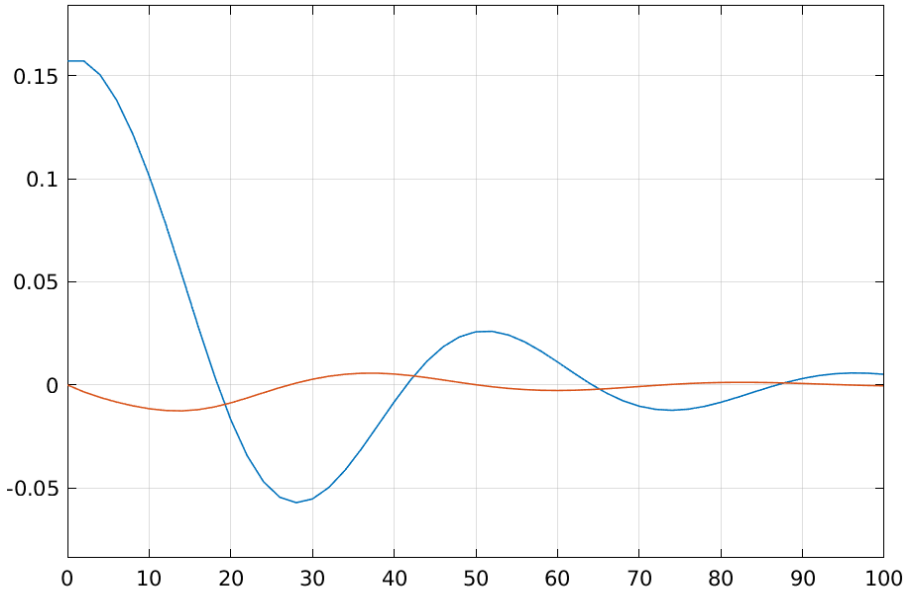
cas 1.1

Dans le cas numéro 1 avec comme paramètres X0=pi/20, tf=10, K=[30,10] et ode45 comme intégrateur. On observe que l'équilibre est atteint au bout d'environ 9 secondes.



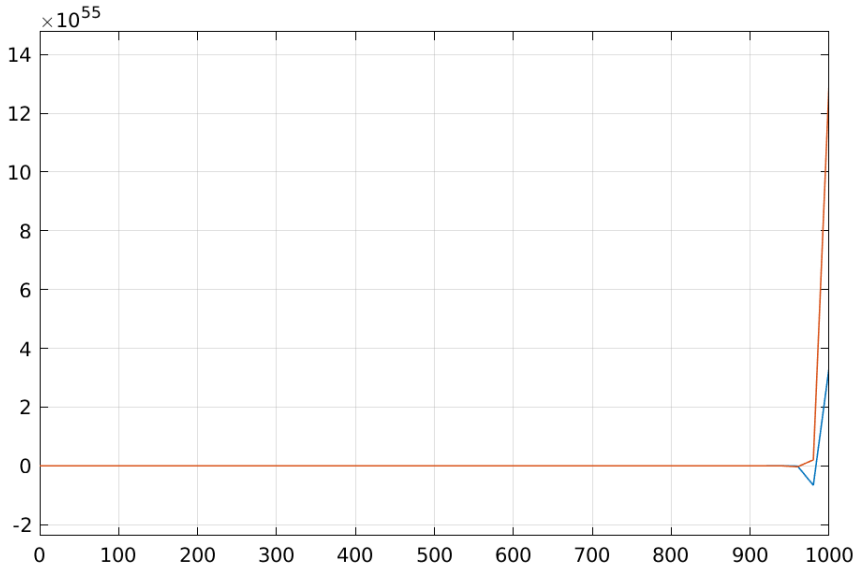
cas 1.2

Dans le cas numéro 2 avec comme paramètres X0=pi/20, tf=100, K=[10,1] et ode45 comme intégrateur. On observe que l'équilibre est atteint bien plus tard vers les 70 secondes dû à la variation de K.

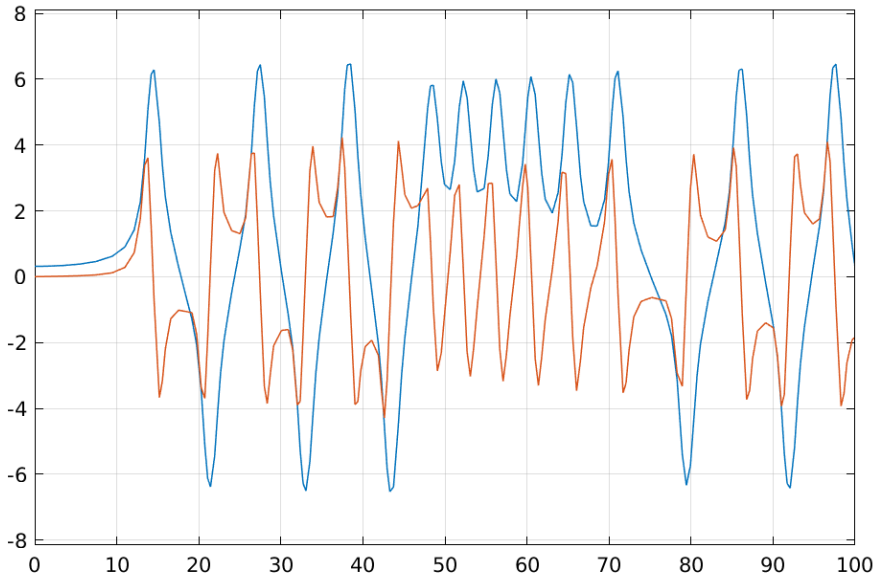


cas 1.3

Ici seul l'intégrateur change et on aperçoit que la stabilité change aussi, il semblerait tout de même que l'on tende vers une stabilité dans ces conditions. Cependant lorsque nous avons laissé la simulation plus longtemps, on remarque que les valeurs s'emballent (valeurs d'ordre 10^56). Cela est dû au fait que notre système n'est pas optimisé pour un temps de simulation (et donc par extension un 'dt') aussi grand

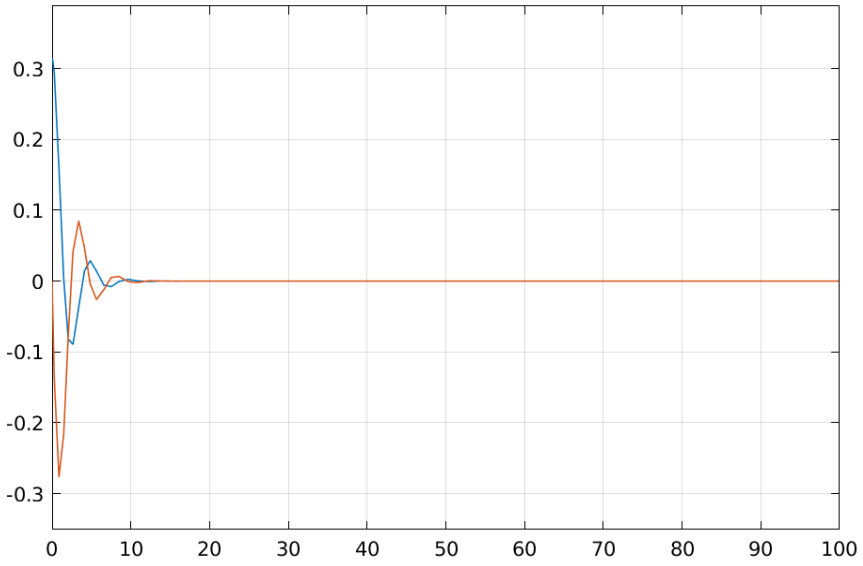


cas 1.4



cas 1.5

Dans le cas numéro 5 après avoir changé la valeur de X0 et l'intégrateur nous voyons que le système n'est pas stable



cas 1.6

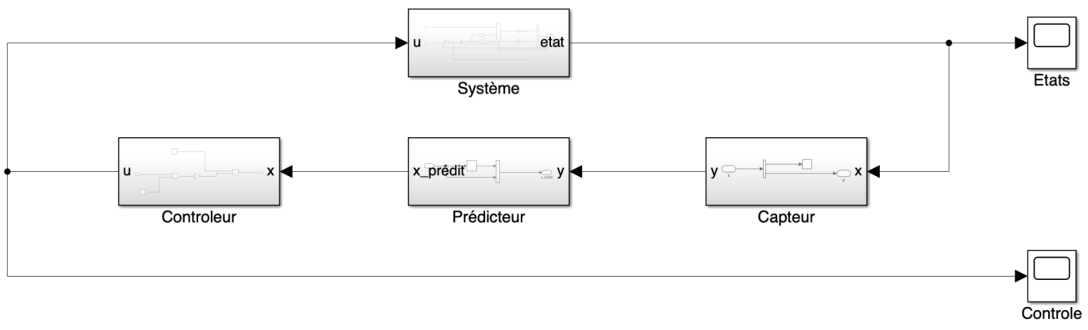
Dans le cas numéro 6, après avoir modifié la valeur de K, on retrouve l'équilibre.

Si nous comparons le cas 1 et 6, seul X0 change. Nous remarquons alors que l'équilibre est atteint plus tôt dans la cas 1 que dans le cas 6 car X0=pi/20 est plus proche de l'origine. De même, les cas 2 et 5 confirment le fait qu'avoir un X0 plus proche de l'origine permet d'obtenir une meilleure stabilisation. De plus, nous remarquons que la précision de l'intégrateur joue un rôle aussi comme montré dans le cas 2 et dans le cas 3.
On peut aussi remarquer que la meilleure configuration pour la stabilité est K=(30,10) (cas 1 et 6) et XO le plus petit possible donc ici pi/20. Ainsi le meilleur cas de stabilité est le cas 1.

Capteurs

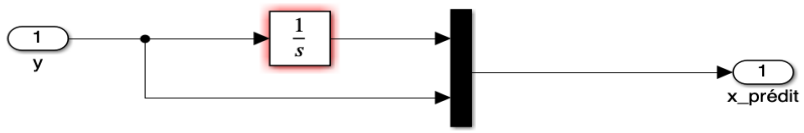
On suppose maintenant que l'on n'a accès qu'à α_point. On introduit donc dans le schéma deux sous systèmes : un capteur et un prédicteur (on utilisera un intégrateur continu pour la prédiction) pour reconstruire α

Pour cela, nous allons ajouter quelques modifications au bloc de la première partie du TP, on y ajoute un prédicateur et évidement un capteur:



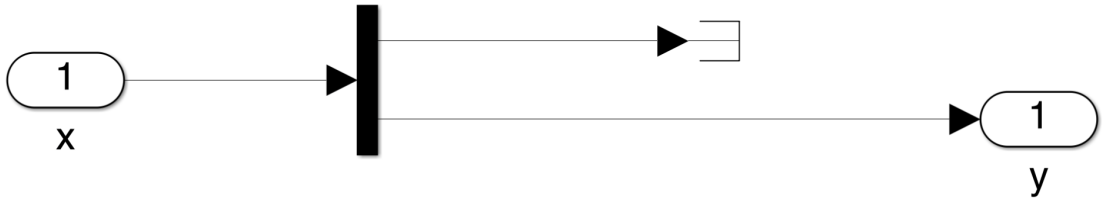
prédicateur:

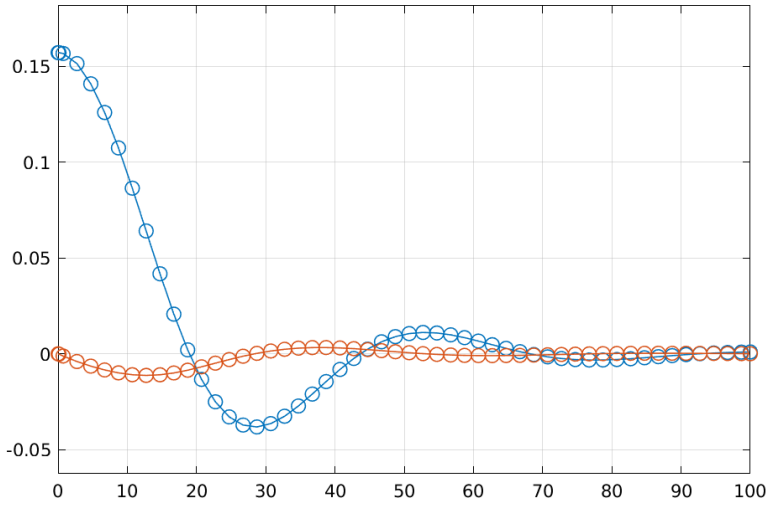
La valeur x2 est donnée prédicateur, elle est intégré pour obtenir x1. Ensuite le prédicateur le prédicteur réforme le vecteur et le stock dans la variable x_predit



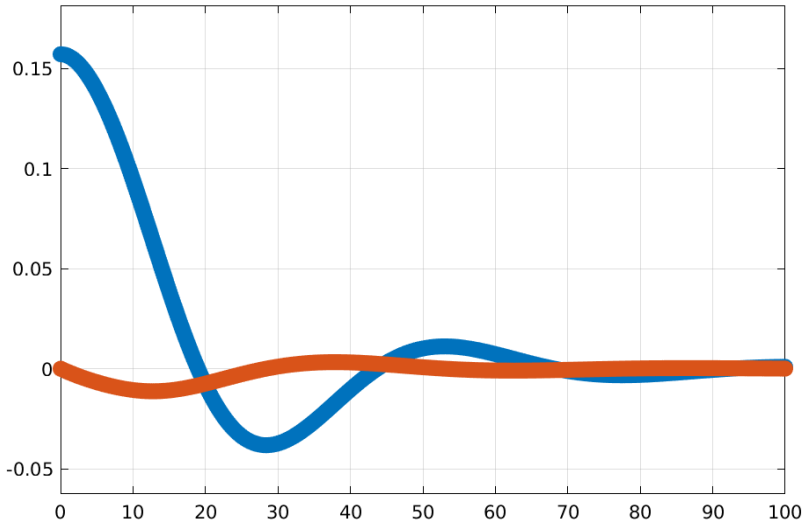
capteur:

Le capteur reçoit le vecteur x.

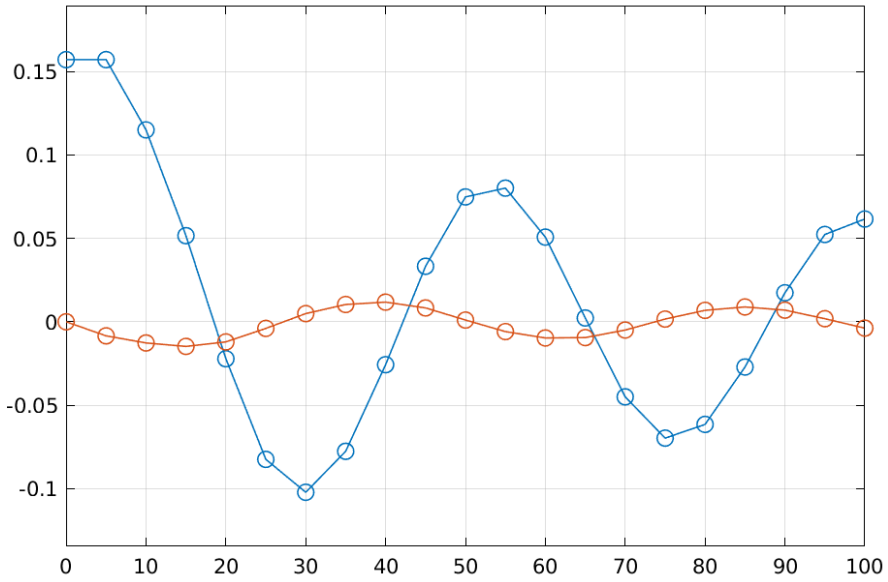




cas1: X0=pi/20, tf=100, pas: par défaut, intégrateur: ode45



cas2: X0=pi/20, tf=100, pas=0,001, intégrateur: Euler ode1

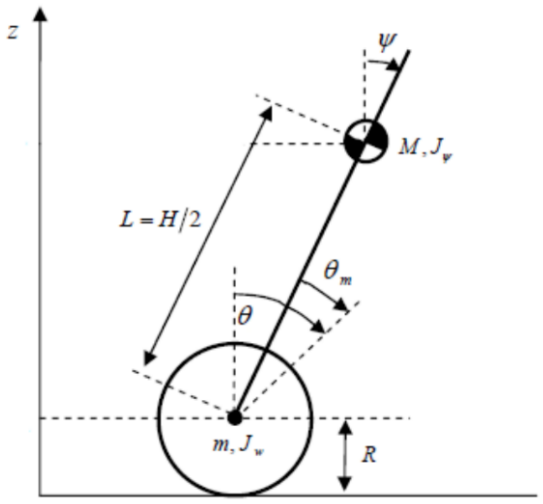


cas3: X0=pi/20, tf=100, pas=5, integrateur: Euler ode1

Nous voyons que dans les cas 1 et 2 le système est stabilisé et de manière assez similaires mais dans que cas3 ce n'est pas le cas. Le seul paramètre qui a été changé est le pas de l'intégrateur. Le fait que la stabilité n'est pas atteinte peut s'expliquer par la diminution de la précision dû à l'augmentation du pas de l'intégrateur.

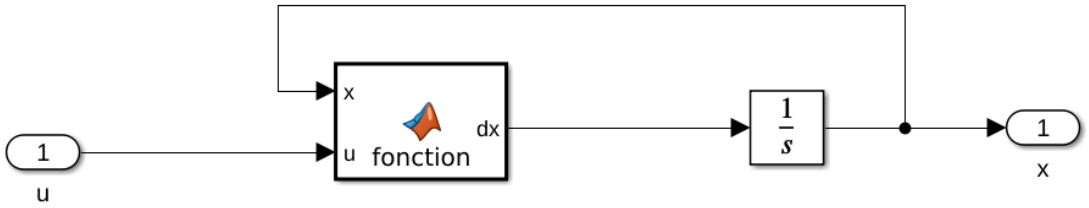
TP03:

On décide maintenant de modéliser notre robot par un système plus complexe avec 4 paramètre : θ, θ', ψ et ψ'



θ : l'angle d'inclinaison au niveau des roues
θ' : la vitesse angulaire de θ
ψ : l'angle d'inclinaison au niveau de la "tête"
ψ': la vitesse angulaire de ψ

L'évolution du système ce fera désormais via une fonction matlab directement implanté dans le bloc 'système' de notre simulation :

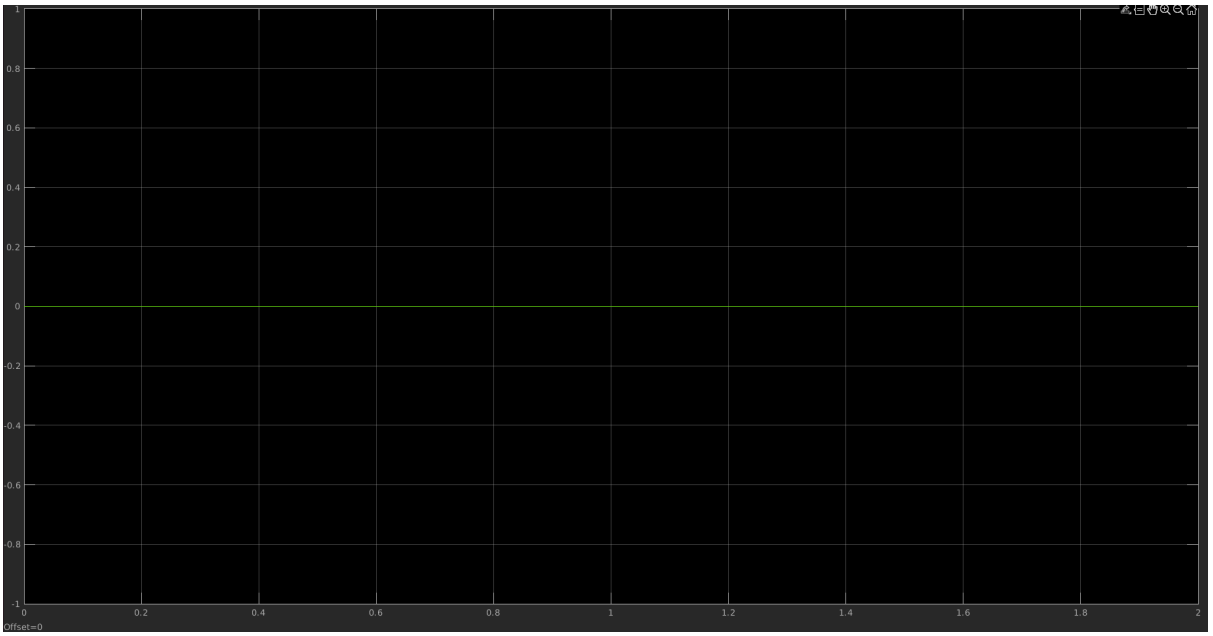


Modèle continue

On commence par calculer les coefficients K pour le contrôleur par retour d'états.
et on obtient le K suivant :
[0.670018499483966, 19.905472454679757, 1.074709928140668, 1.961419611424519]

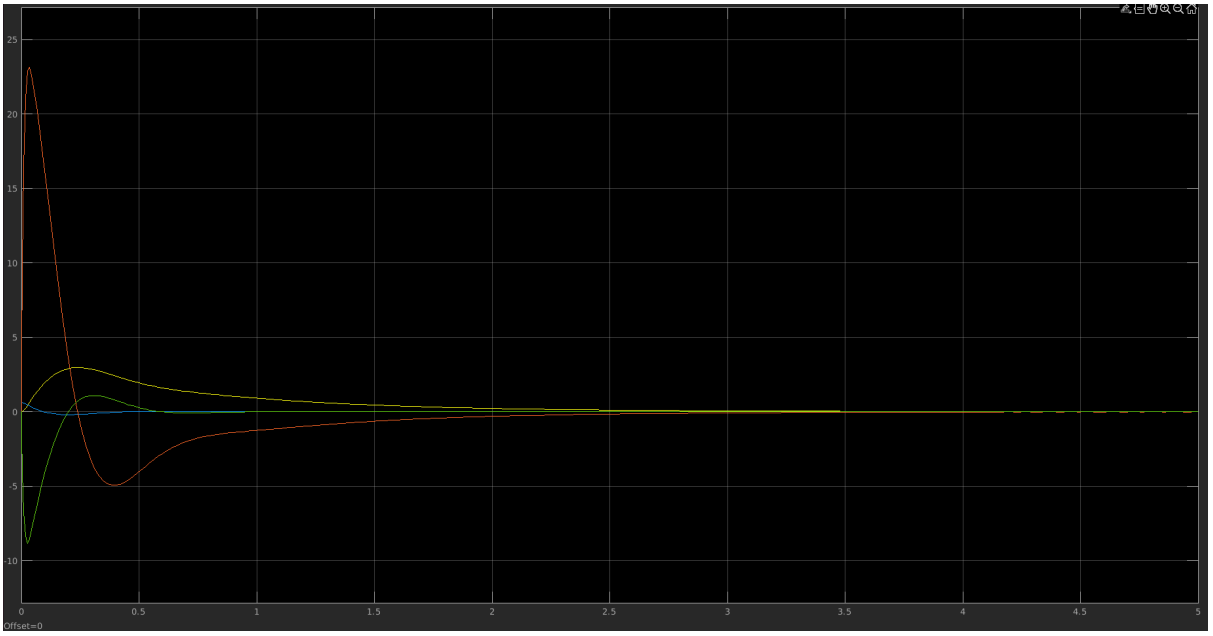
Puis on simule notre système sur un modèle continu (intégrateur ode45).

cas1 (X0 = [0,0,0,0]; tf = 2)



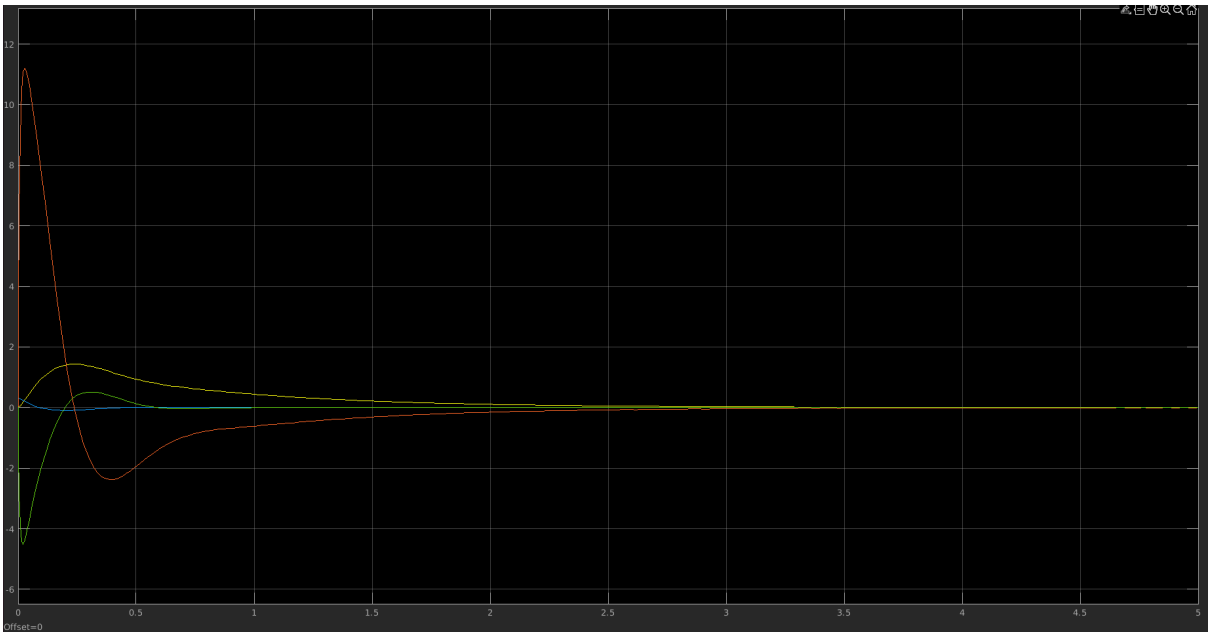
Dans ce cas, on observe qu'en partant de l'origine, un point d'équilibre stable, aucun des paramètres du modèle n'évoluent et le système reste stable.

cas2 (X0 = [0,pi/5,0,0]; tf = 5)



Ici, en partant d'un angle ψ de pi/5, on arrive à un équilibre au bout de 3 secondes environ. Cette perturbation initial de ψ correspond à une perturbation au niveau de la tête du robot entraînant son l'inclinaison. en effet, la valeur de ψ augmente au début (signifiant que la perturbation entraîne une plus grande inclinaison) avant de tendre vers 0. On observe aussi une augmentation de θ suite à la perturbation initial (qui finit aussi par tendre vers 0).

cas3 (X0 = [0,pi/10,0,0]; tf = 5)



On remarque dans ce cas qu'en divisant par 2 l'angle de départ, on à la même dynamique d'évolution des paramètres du modèles mais avec des valeurs 2 fois inférieures. on arrive toujours à l'équilibre en environ 3 secondes.

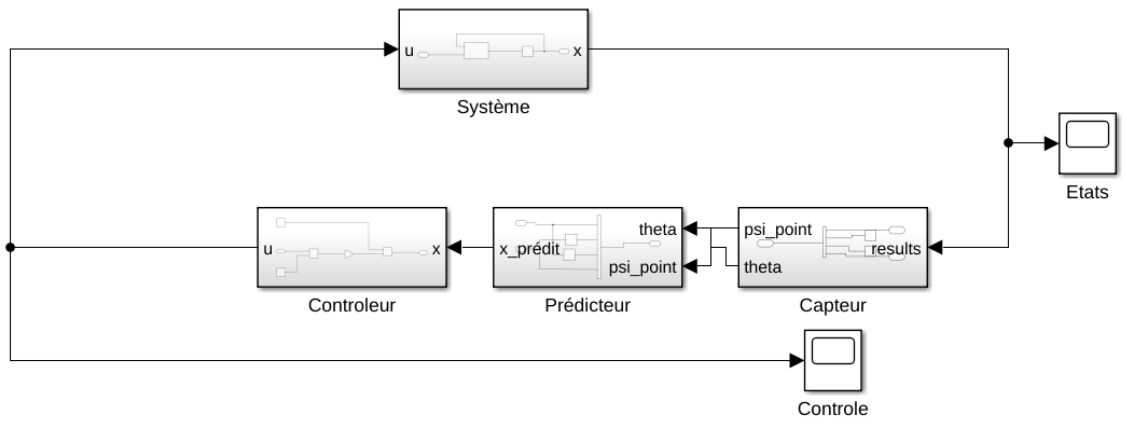
Introduction des capteurs et actionneurs

En réalité, on n'as pas accès à toutes les composantes en sortie du système, mais à seulement 2 obtenue avec :

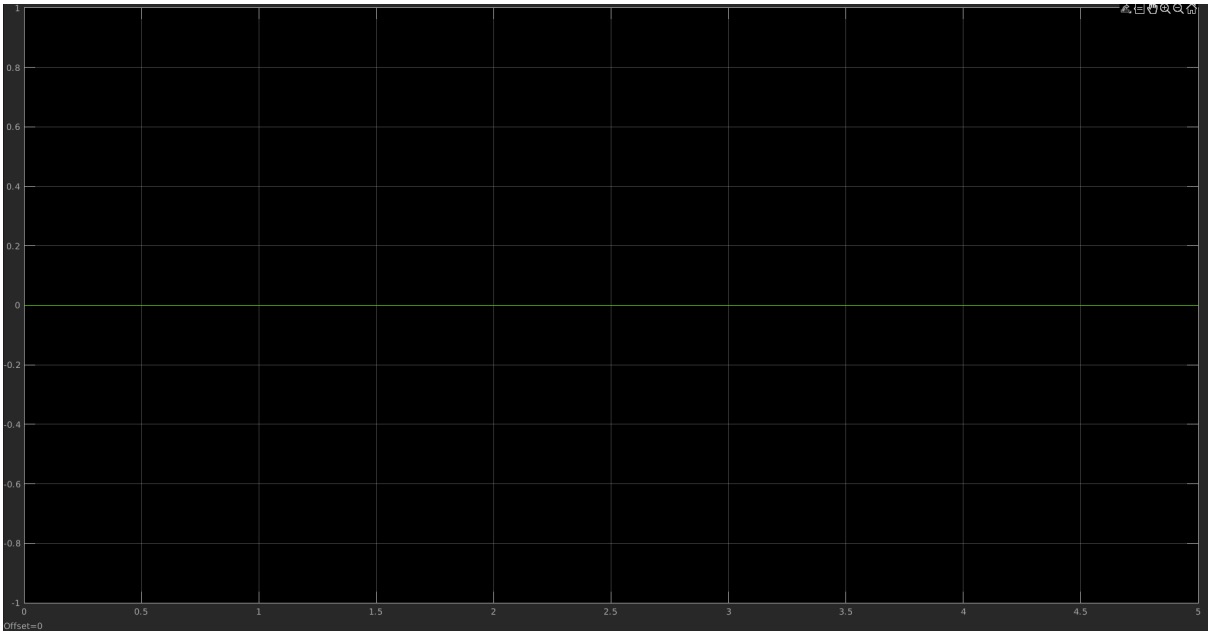
- Un gyroscope mesure la vitesse de changement d'angle du corps du robot ψ̇(t)
- Un capteur mesure l'angle θ(t)

Pour compenser le manque d'information en sortie, on introduit un sous-système prédicateur qui recalcule les autres composantes.

notre système est donc de la structure suivante :

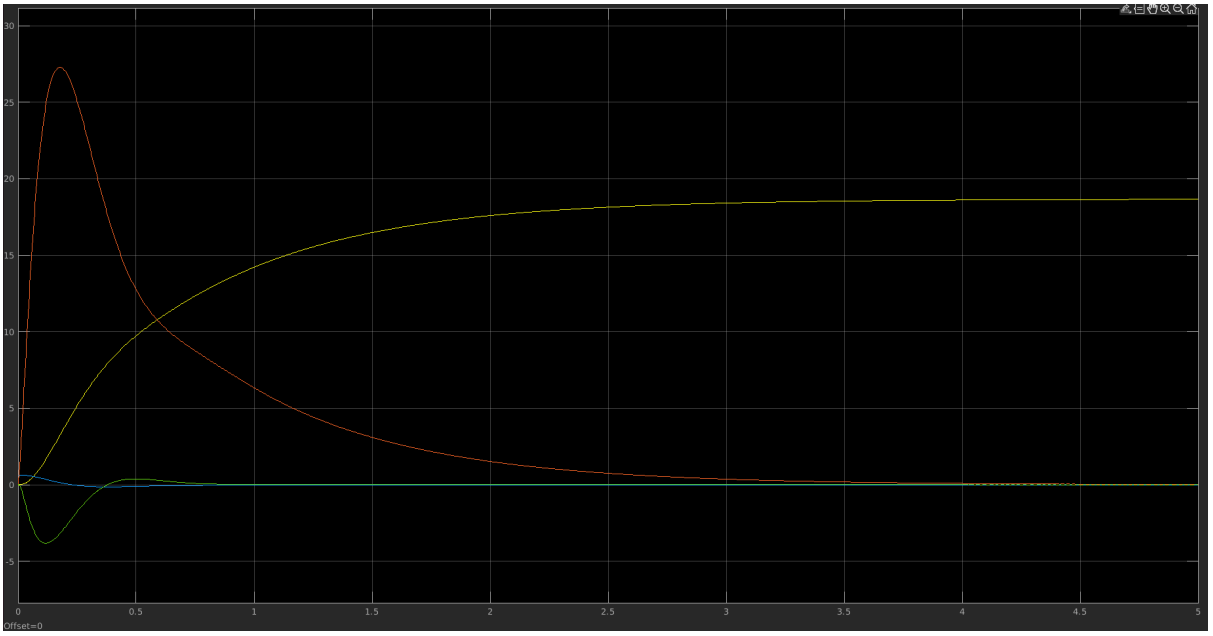


cas1 (X0 = [0,0,0,0]; tf = 2)

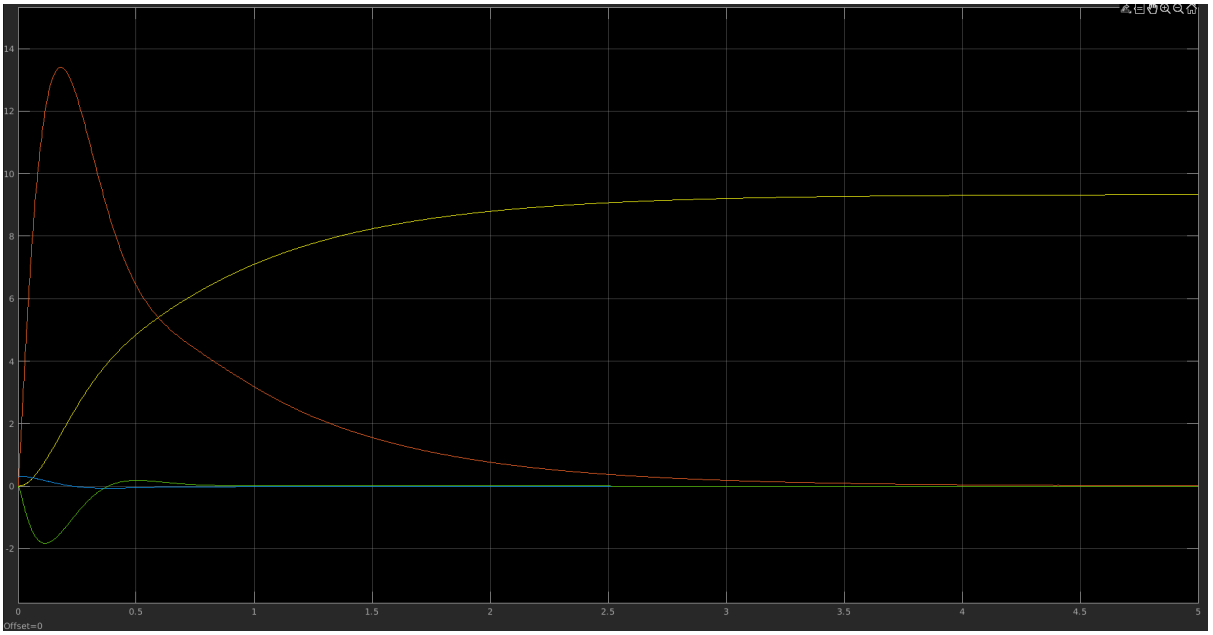


On a toujours un équilibre stable en partant de l'origine.

cas2 (X0 = [0,pi/5,0,0]; tf = 5)



cas3 (X0 = [0,pi/10,0,0]; tf = 5)

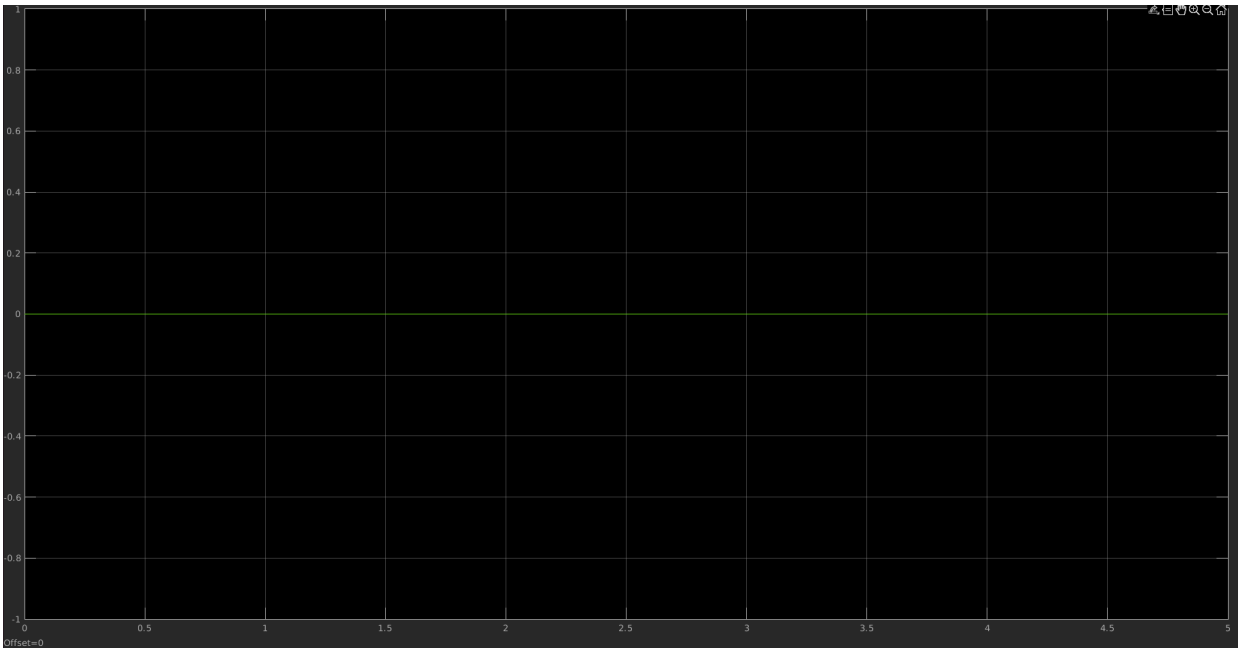


Comme précédemment, on observe exactement la même dynamique avec des valeurs 2 fois plus petites dans le cas 3. Mais, on n'atteint plus d'équilibre. Ceci est sûrement dû à une erreur dans notre prédicateur, en effet, la seule valeur ne tendant pas vers 0 atteint tout de même un plateau 'stable' au même moment que les autres paramètres.

Construction du modèle hybride

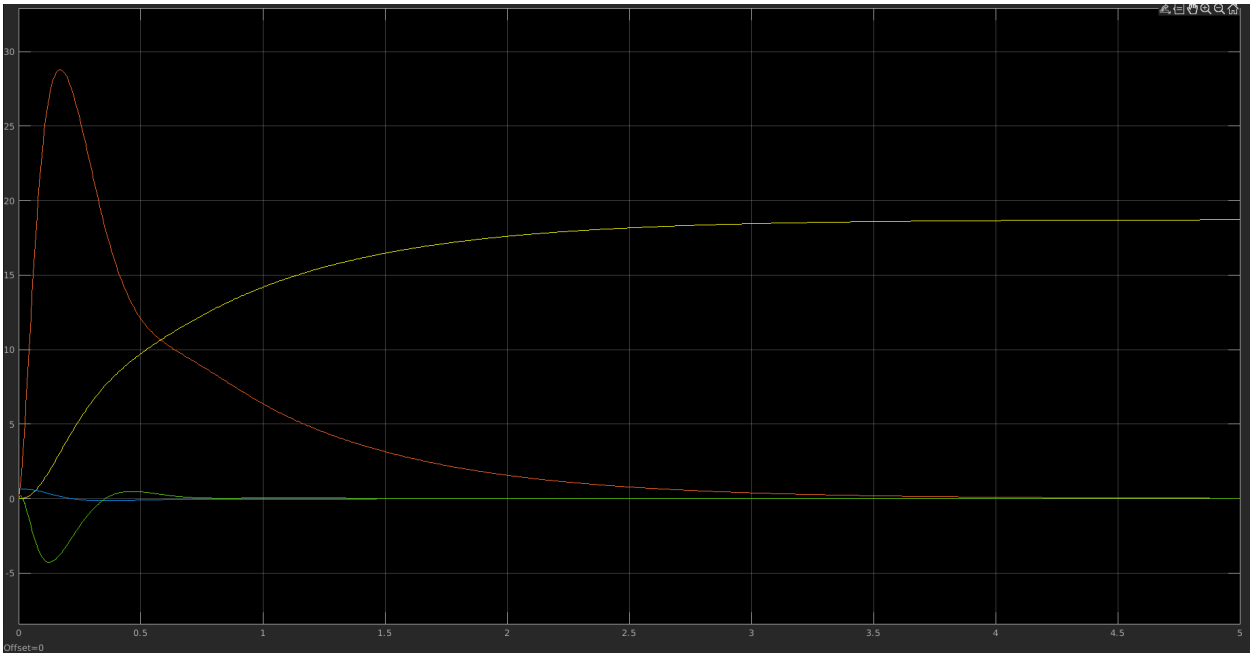
Encore une fois, en réalité nos capteurs nous renverront des valeurs discrètes et non continues.
On modifie donc notre prédicateur pour qu'il prenne en compte des valeurs discrètes et non continues.

cas1 (X0 = [0,0,0,0]; tf = 2)

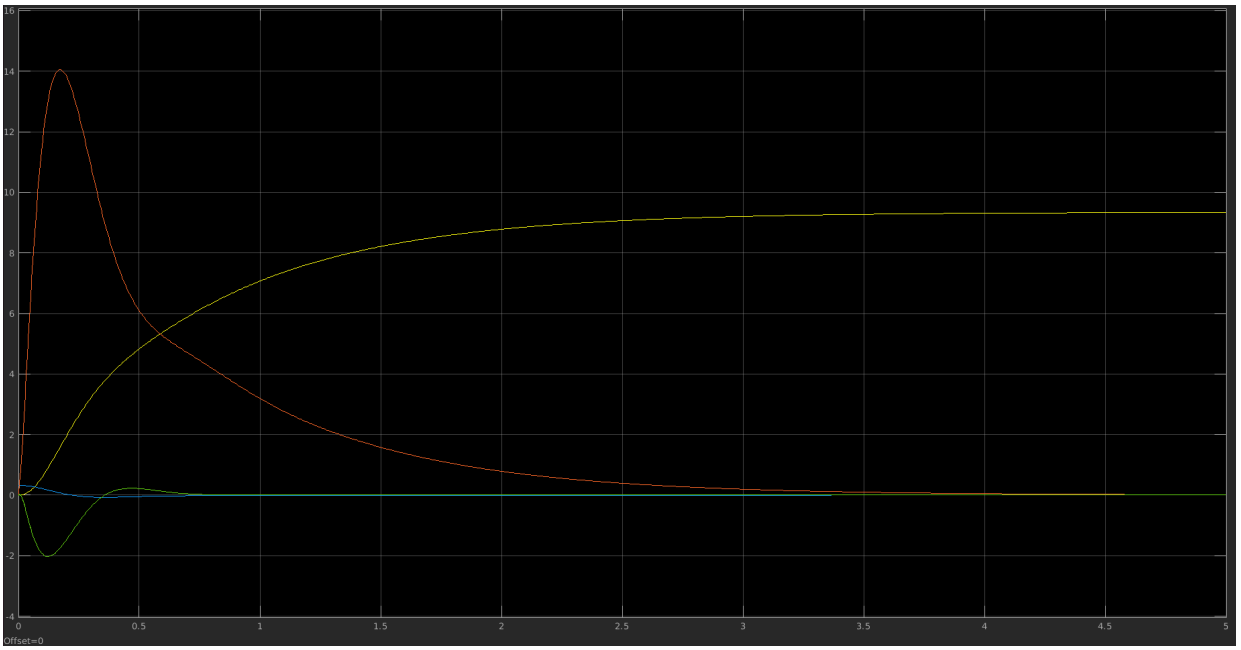


On a toujours un équilibre stable en partant de l'origine.

cas2 (X0 = [0,pi/5,0,0]; tf = 5)



cas3 (X0 = [0,pi/10,0,0]; tf = 5)



On n’observe pas de réel changement en assumant des valeurs discrètes pour nos capteurs plutôt que des valeurs continues. Donc même si nos résultats n'indiquent pas un équilibre, on assumera que l’équilibre est atteint avec ces valeurs.

Conclusion :

Nous avons au cours des ces TPs réussi à modéliser un robot capable de tenir en équilibre. Ce que nous avons pu vérifier en implémentant un code correspondant au modèle dans le petit robot qui a réussi à se stabiliser et à tenir en équilibre.