# TP: METHODOLOGIE DE LA COMMANDE

HOUSSEYNE NADOUR
JULIEN OIKNINE

Ecole Centrale de Nantes

ECOLE CENTRALE NANTES  [Company address]

# 1 Linéarisation

En posant : $X = \begin{pmatrix} x1 \\ x2 \\ x3 \\ x4 \end{pmatrix} = \begin{pmatrix} \theta \\ \dot{\theta} \\ p \\ \dot{p} \end{pmatrix}$, et $Y = \begin{pmatrix} \theta \\ p \end{pmatrix}$, on obtient d'après les équations linéarisées: $\dot{X} = A.X + B.U$ et $Y = C.X + D.U$ tel que : $C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$ et D = $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

C'est un système de 4 états, une entrée et deux sorties.

Les pôles du système sont : 0, 0, 3.2833, -3.2833
On remarque que le système a trois pôles instables, d'où l'objective est de stabiliser le système.

### Simulation en boucle ouverte :

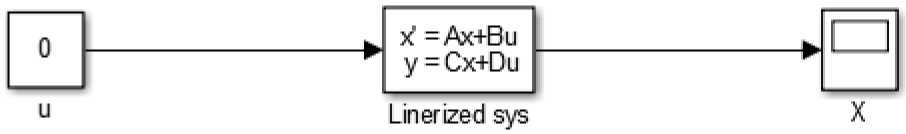



*Figure 1: Schéma Simulink de la boucle ouverte*

Pour une position initiale non nulle $X_0 = \begin{pmatrix} 0.1 \\ 0 \\ 0.1 \\ 0 \end{pmatrix}$ le système va diverger

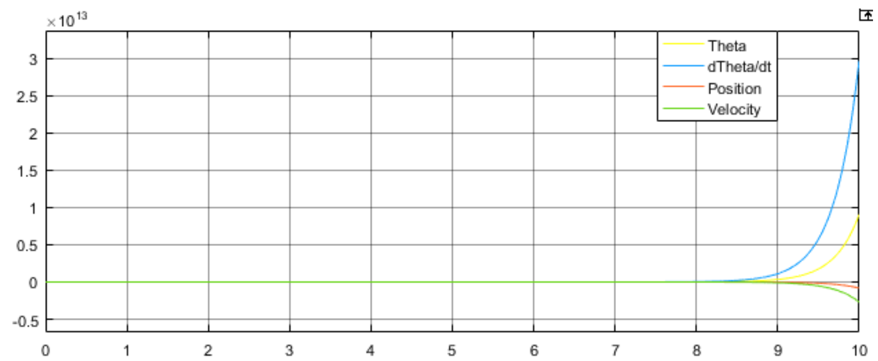



*Figure 2: Simulation en boucle ouverte*

# 2. Commande par retour d'état (sans observateur)

## 2.1 La commandabilité

Pour savoir si on peut stabiliser la barre autour d'une position d'équilibre θ=0, il faut savoir si le système comporte des parties non commandables instables ou non. Pour cela on calcule la Matrice de commandabilité $C = [B \quad A.B \quad A^2.B \quad A^3.B]$ et son rang.

Le rang de C est 4=n, le système est donc complètement commandable (il ne comporte pas de partie non commandable), on peut donc stabiliser la barre autour de la position d'équilibre θ=0 grâce à un retour d'état.

## 2.2 Calcule de la matrice de retour d'état :

Pour calculer la matrice de retour d'état F, on utilise soit la fonction 'acker' (le système a une seule entrée) soit la fonction 'place',

Pour des pôles désirés $P = \begin{bmatrix} -1+i \\ -1-i \\ -2-2i \\ -2+i \end{bmatrix}$ On trouve : $F = -[152.0633 \;\; 42.2449 \;\; 8.1633 \;\; 12.2449\,]$

### Simulation :

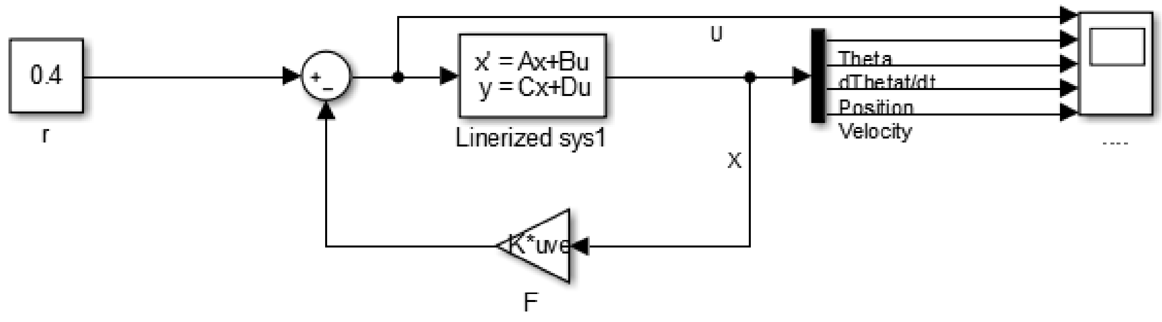



*Figure 3: Schéma Simulink du retour d'état*

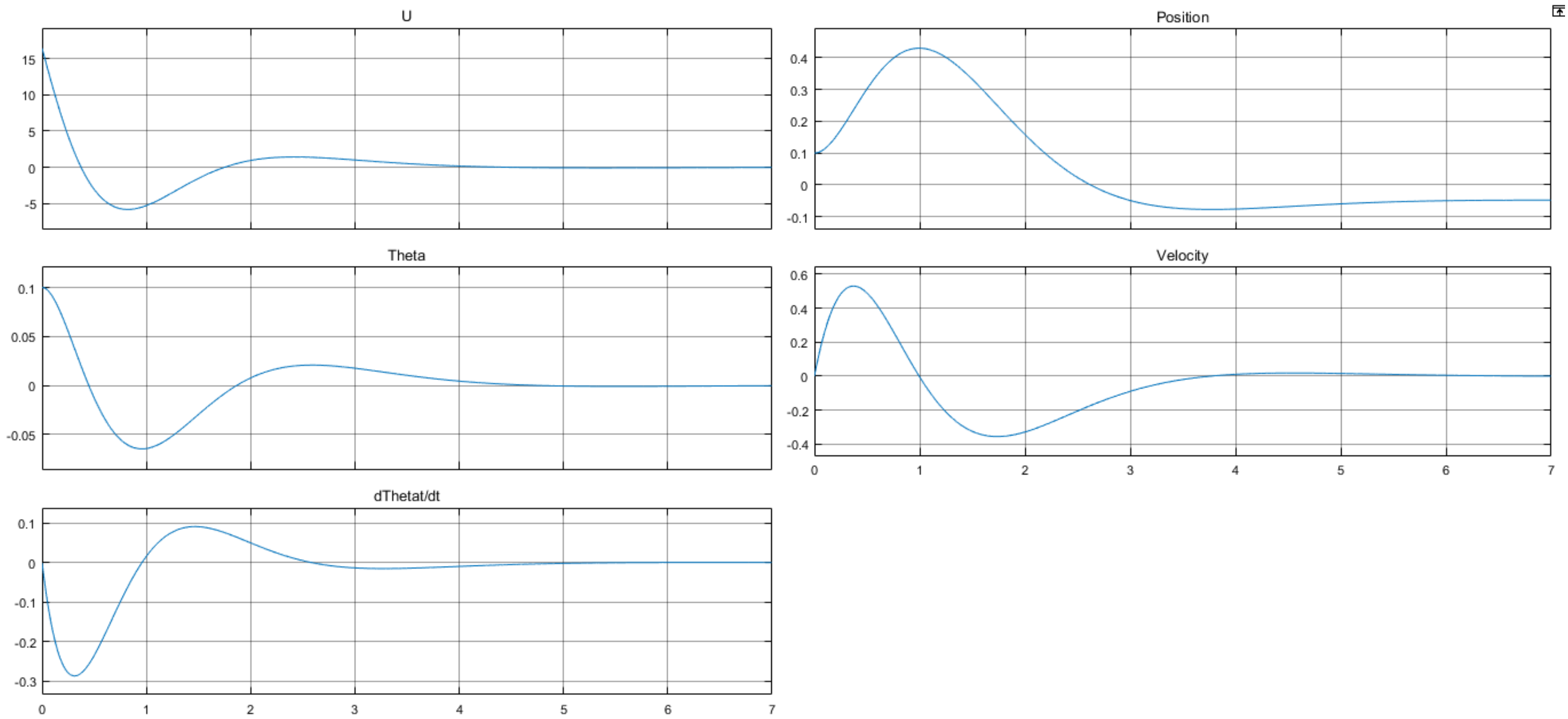



*Figure 4: Simulation du Retour d'état*

Remarques :

Le retour d'état a assuré la stabilité, par contre on voie que la poursuite n'est pas vérifiée.

# 2. Commande par retour d'état reconstruit :

## 2.1) Calcule de la matrice d'observation K :

### A) Observabilité :

Le système est observable si et seulement si la matrice d'observabilité

$\vartheta = \begin{bmatrix} C \\ A.C \\ A.C^2 \\ A.C^3 \end{bmatrix}$ est de rang n=4, a l'aide de la fonction 'rank' on trouve que

$rang(\vartheta) = 4 = n.$

### B) Calcule de la matrice d'observation K :

A l'aide de la fonction place (K = (acker(A ',C',Poles)') on trouve :

**B.a) l'observateur est plus rapide que la commande :**

$\mathrm{K} = 10^6 \begin{bmatrix} 0.0201 + 0.0002i & 0.0007 + 0.0005i \\ 1.9959 + 0.0041i & 0.0679 + 0.0521i \\ 0.0008 - 0.0005i & 0.0003 - 0.0002i \\ 0.0745 - 0.0646i & 0.0244 - 0.0201i \end{bmatrix}$ pour les pôles suivants : Po = $\begin{bmatrix} -100 + i \\ -100 - i \\ -200 + 2i \\ -200 - 2i \end{bmatrix}$

Ce résultat n'est pas réalisable parce que la matrice K contient des parties imaginaires. Mais pour assurer que l'observateur soit plus rapide que la commande, alors il suffit de prendre

$\mathrm{Po} = 100 * (\text{Pôles de la commande}) = \begin{bmatrix} -100 + 100i \\ -100 - 100i \\ -200 + 200i \\ -200 - 200i \end{bmatrix}$ C'est un résultat réalisable.

**B.b) l'observateur est aussi rapide que la commande :**

$K = \begin{bmatrix} 2.9998 & -0.9985 \\ 14.7801 & 0.0043 \\ 1.0015 & 3.0002 \\ -0.9757 & 3.9999 \end{bmatrix}$ Pour les pôles suivants $Po = P = \begin{bmatrix} -1 + i \\ -1 - i \\ -2 - 2i \\ -2 + i \end{bmatrix}$

**2) Simulation :**

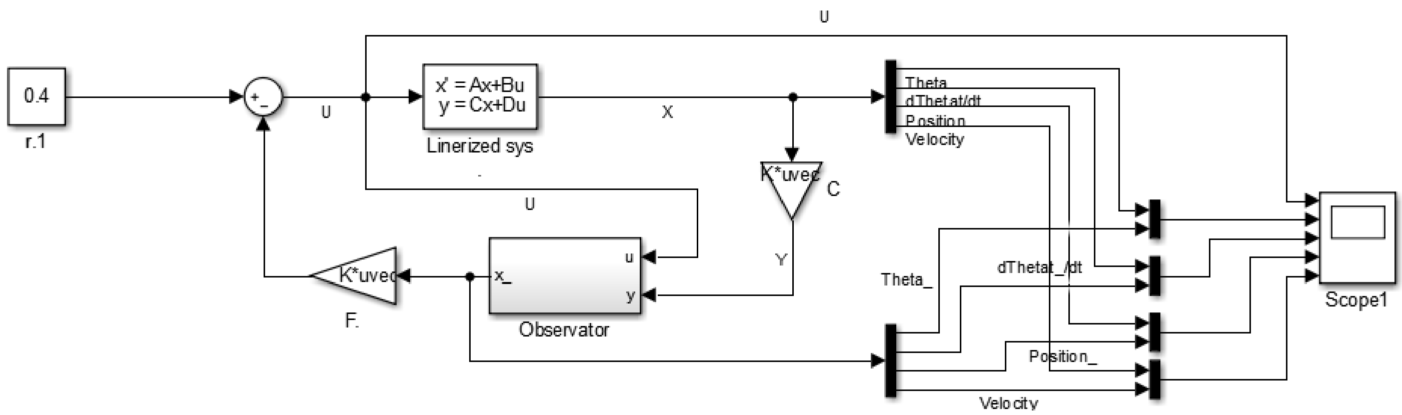



*Figure 5:Shema Simulink du retour d'état reconstruit*

La simulation pour des conditions initiales nulles donne le même résultat dans les deux cas de l'observateur (pôles plus rapides que ceux de la commande, et pôles aussi rapide que ceux de la commande).

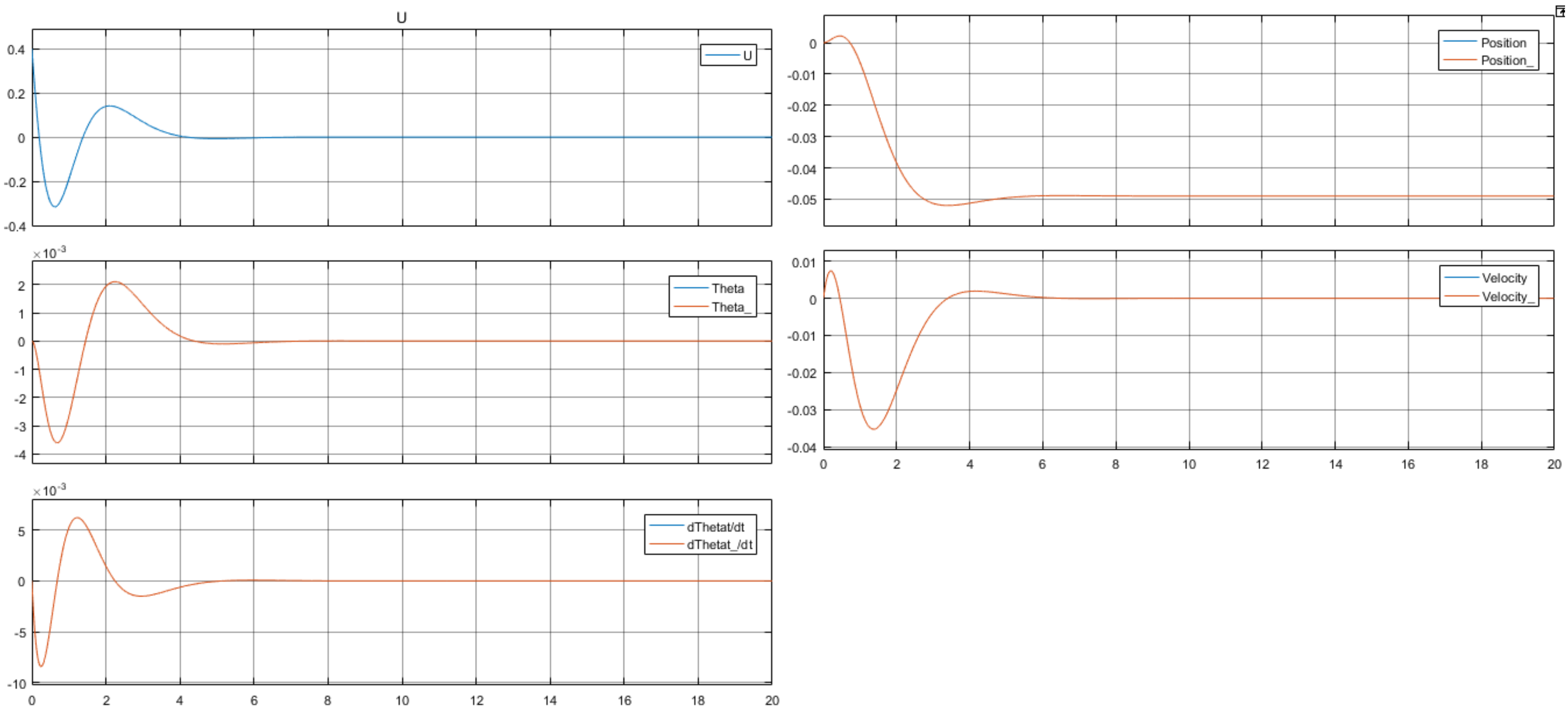



*Figure 6:Simulation du retour d'état reconstruit*

Les signaux reconstruits sont bien identiques à ceux du système, qui montre l'efficacité de l'observation.

Pour montrer la différence entre les deux cas on prend des conditions initiales non nulles $X_0 = \begin{pmatrix} 0.1 \\ 0 \\ 0.1 \\ 0 \end{pmatrix}$ :

**A) Pour des pôles de l'observateur plus rapides que ceux de la commande :**

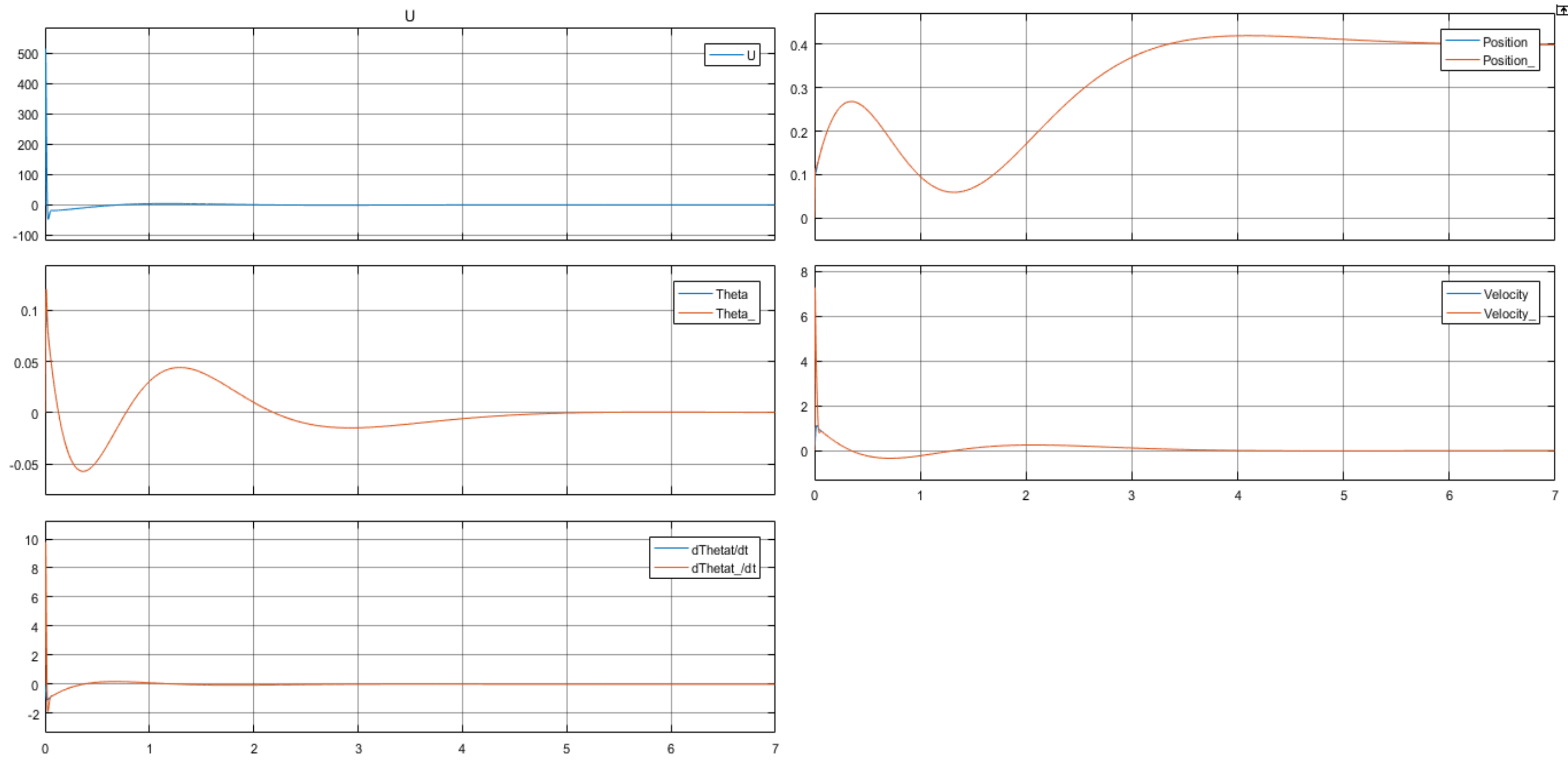



*Figure 7: Simulation, retour d'état reconstruit, avec des pôles d'observation 100 fois plus grand que ceux de la commande*

## B) Pour des pôles de l'observateur aussi rapides que ceux de la commande

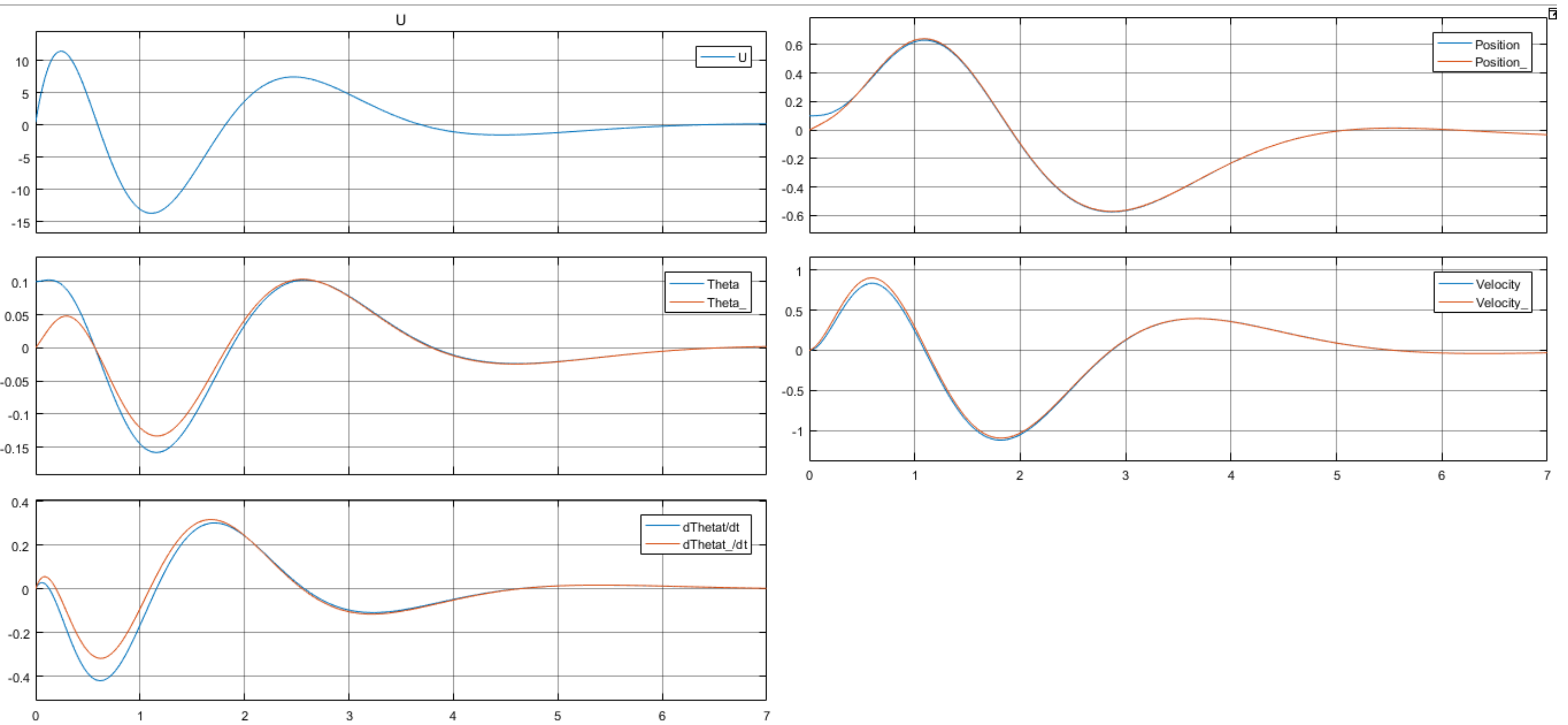



*Figure 8: Simulation, retour d'état reconstruit, avec des pôles d'observation sont les mêmes que ceux de la commande*

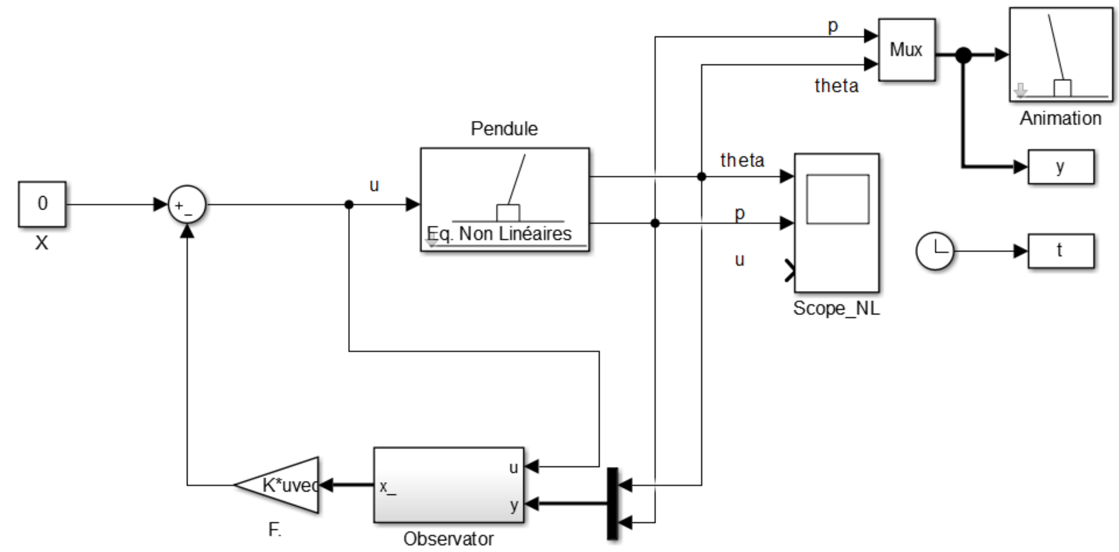
C'est clair que la barre dans le premier cas se stabilise plus rapide que le deuxième cas, parce que l'observation est presque instantanée par rapport à la commande, contrairement au deuxième cas, le retard de l'observation est important par rapport à la commande d'où la lenteur du système dans le deuxième cas.

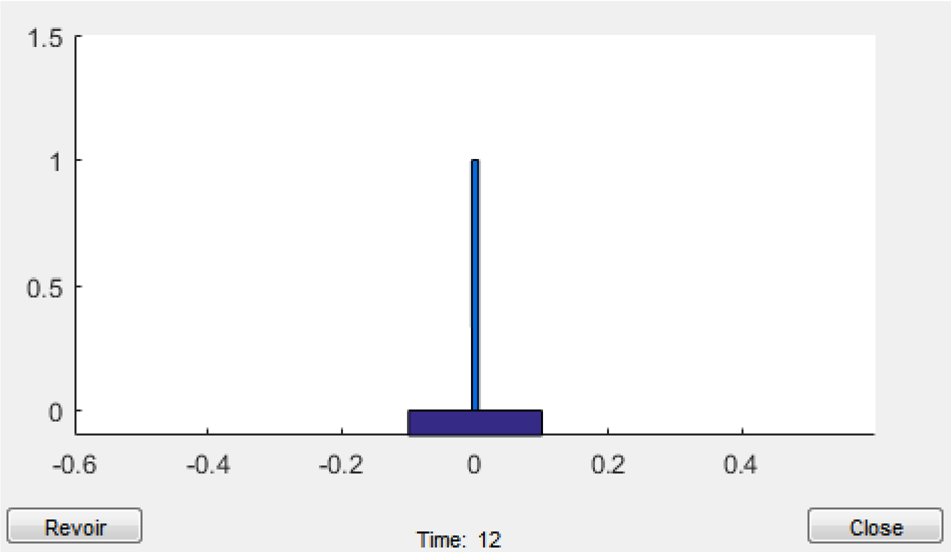
Le système peut diverger si l'observation est plus lente que la commande.

**Animation :**

⇨ **Note :** La loi de la commande par retour d'états reconstruit, appliqué sur le système linéarisé autour d'un angle nul, s'applique correctement sur le système non linéaire (réel), tant que le système est toujours autour du point d'équilibre pour laquelle on a fait la linéarisation (angle nul), sinon le comportement du système va être complètement dissimilaire au celui du système linéarisé, en effet si on prend une condition initiale $X_0 = \begin{pmatrix} 0.4\,rad = 22.8° \\ 0 \\ 0 \\ 0 \end{pmatrix}$, alors le système diverge. Masi si on prend $X_0 = \begin{pmatrix} 0.35\,rad = 20° \\ 0 \\ 0 \\ 0 \end{pmatrix}$ alors le system converge, que signifie qu'on peut prend $\theta = \mp 20°$ comme la plus large région d'attraction assurée pour cette loi de commande.

**Pour l'animation on prend une condition initiale $\boldsymbol{\theta = 0.1\,rad}$ :**






## 5) La poursuite et la robustesse :

On veut dans cette partie asservir la position du chariot.
pour cela on propose deux méthodes :

1) **En introduisant un précompensateur :**

   le précompensateur sert à rendre le gain statique du système égale a 1, alors aura besoin de calculer les fonctions de transfert entre la consigne et la position :

$$G(s) = C.\left(s.I - A_{bf}\right)^{-1}.B \ tel\ que\ A_{bf} = A - B.F$$

   On trouve à l'aide du Matlab que :

$$G(s) = \begin{pmatrix} \dfrac{-0.2\,s\text{^}6\ -\ 1.2\,s\text{^}5\ -\ 3.6\,s\text{^}4\ -\ 4.8\,s\text{^}3\ -\ 3.2\,s\text{^}2\ +\ 2.132e-14\,s\ +\ 7.105e-15}{s\text{^}8\ +\ 12\,s\text{^}7\ +\ 72\,s\text{^}6\ +\ 264\,s\text{^}5\ +\ 644\,s\text{^}4\ +\ 1056\,s\text{^}3\ +\ 1152\,s\text{^}2\ +\ 768\,s\ +\ 256} \\ \\ \dfrac{0.2\,s\text{^}6\ +\ 1.2\,s\text{^}5\ +\ 1.64\,s\text{^}4\ -\ 6.96\,s\text{^}3\ -\ 32.08\,s\text{^}2\ -\ 47.04\,s\ -\ 31.36}{s\text{^}8\ +\ 12\,s\text{^}7\ +\ 72\,s\text{^}6\ +\ 264\,s\text{^}5\ +\ 644\,s\text{^}4\ +\ 1056\,s\text{^}3\ +\ 1152\,s\text{^}2\ +\ 768\,s\ +\ 256} \end{pmatrix}$$

On s'intéresse au deuxième ligne qui est la fonction de transfert entre la consigne et la position, le gain statique de cette fonction est G(0) = -31.36/256, on multiplie la consigne par K = 1/G(0).

Simulation :

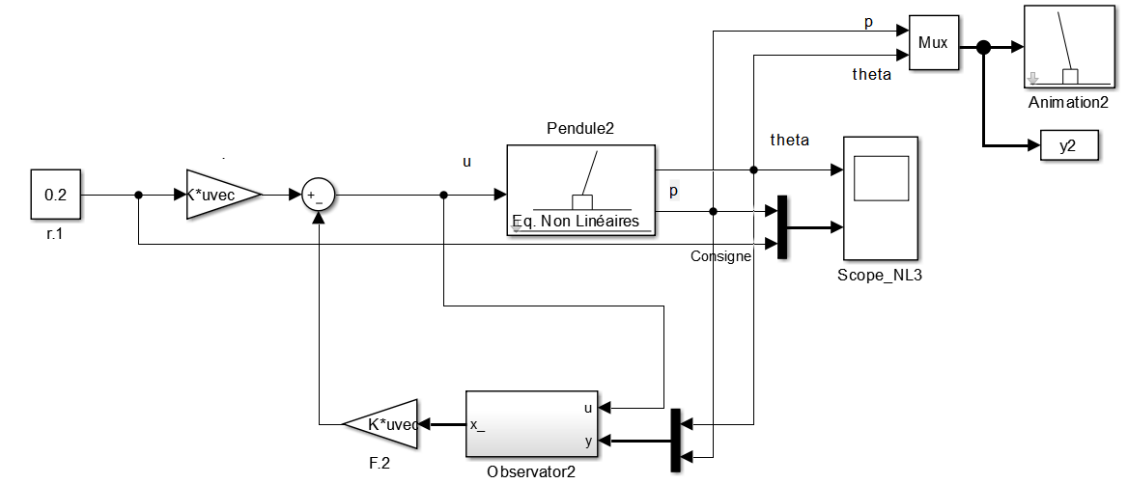



*Figure 9:Shema Simulink pour retour d'état reconstruit, avec précompensateur statique.*

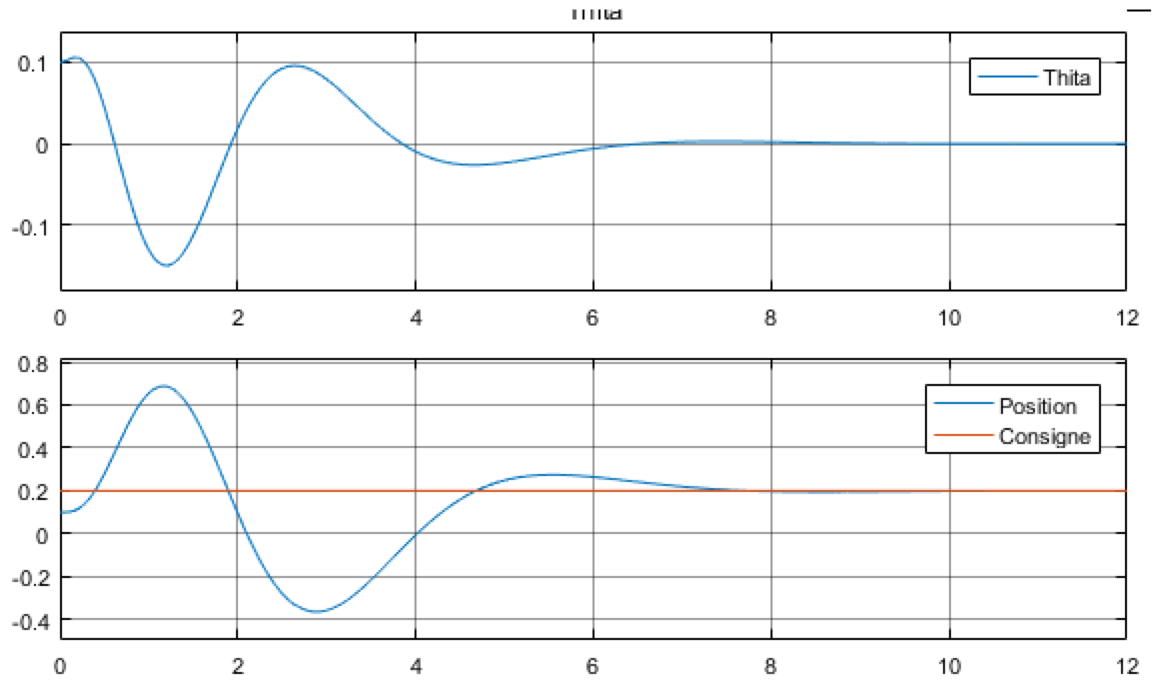



*Figure 10:Simulation pour retour d'état reconstruit, avec précompensateur statique.*

La poursuite statique est assurée, mais cette méthode n'est pas pratique, en effet, la poursuite dynamique n'est pas assurée, en plus cette commande n'a pas de performances en régulation (ni statique ni dynamique), alors on switch a une autre méthode de commande.

## 2) Changement de variable (ou bien changement de repere :

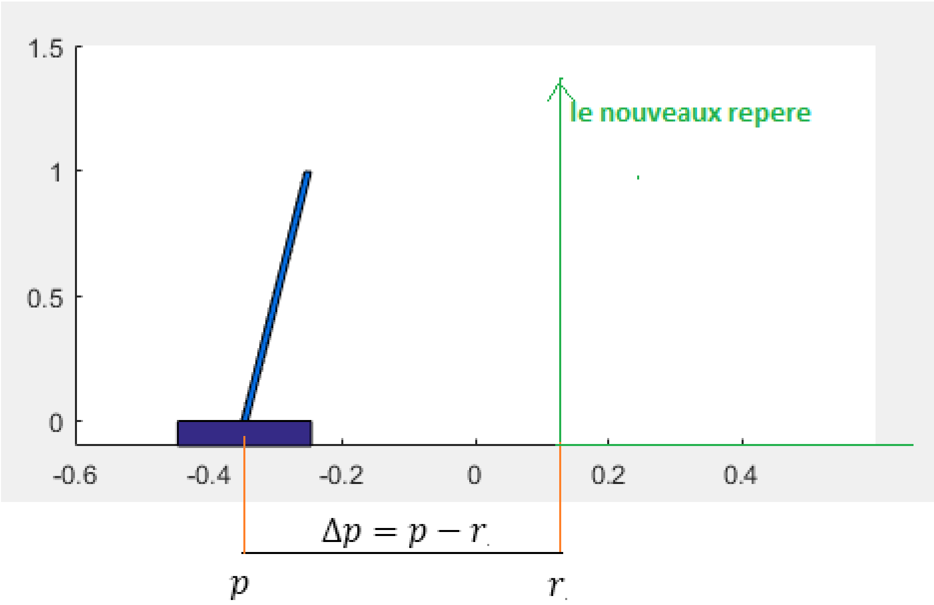
La régulation retour d'état $U = -F.X + v$, assure la convergence, en effet :

$X(t) = X_0.e^{-(A-B.F)t} + e^{-(A-B.F)t} \int e^{-(A-B.F)t}.v(t)dt$

Si : $v(t) = 0$, alors x est obligé de converger vers zéro.

L'idée est de changer les variable d'état, alors au lieux de penser à commander X, on pense à commander $\Delta X = X - X_d$

tel que $X_d = \begin{pmatrix} 0 \\ 0 \\ r \\ 0 \end{pmatrix}, alors\colon \Delta X = \begin{pmatrix} \theta \\ \dot{\theta} \\ \Delta p \\ \dot{p} \end{pmatrix}$ et $\Delta Y = \begin{pmatrix} \theta \\ \Delta p \end{pmatrix}$ tel que $\Delta p = p - r,$ et r représente la position désirée (consigne statique, on va perdre l'asservissement dynamique pour avoir un calcule simple) , et ça tout en gardant $v = 0$.




Puisque la linéarisation a été faite par rapport $\theta$ seulement, alors les matrices de la linéarisation pour $\Delta X$ ne sera pas changer :

$$\Delta \dot{X} = A.\Delta X + B.\Delta U \text{ et } \Delta Y = C.\Delta X$$

On va garder la même forme de la loi de commande par retour d'état reconstruit :

$\Delta U = -F.\Delta X$ , qui va assurer que $\lim_{t=\infty} \Delta X = 0$

On substitue la valeur de $\Delta X$ et de U dans l'équation, on obtient :

$$\dot{X} - \dot{X}_d = A.X - AX_d - B.F.\Delta X$$

Puisque r est statique alors $\dot{X}_d = 0$, en plus $AX_d = 0$, on obtient :

$$\dot{X} = A.X - B.F.\Delta X$$

Donc il suffit de prendre $u = -F.\Delta X$ au lieu de $u = -F.X + v$ pour assurer que X tend vers Xd.

**Simulation :**

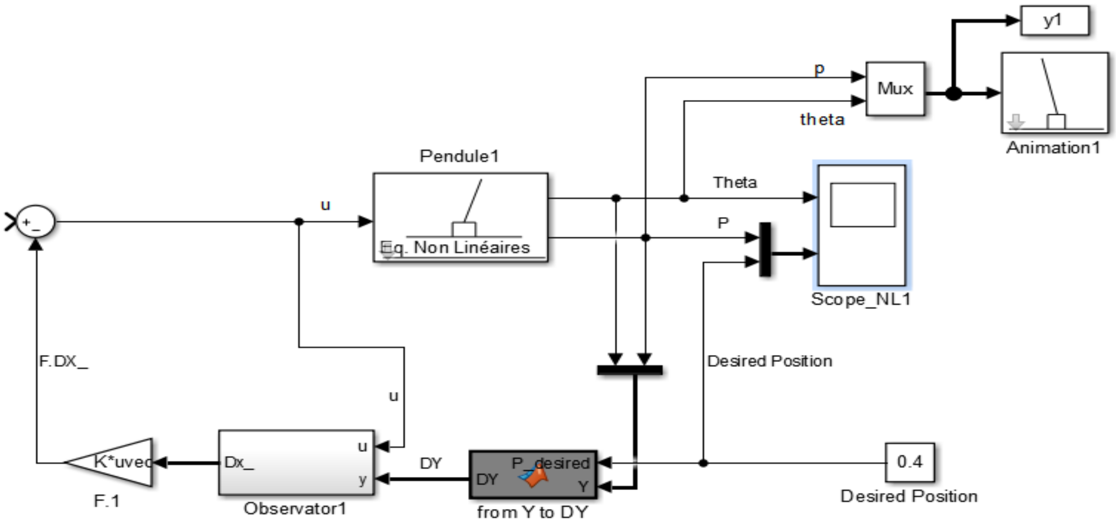



*Figure 11:Shema Simulink pour un retour d'état reconstruit, avec changement de variable*

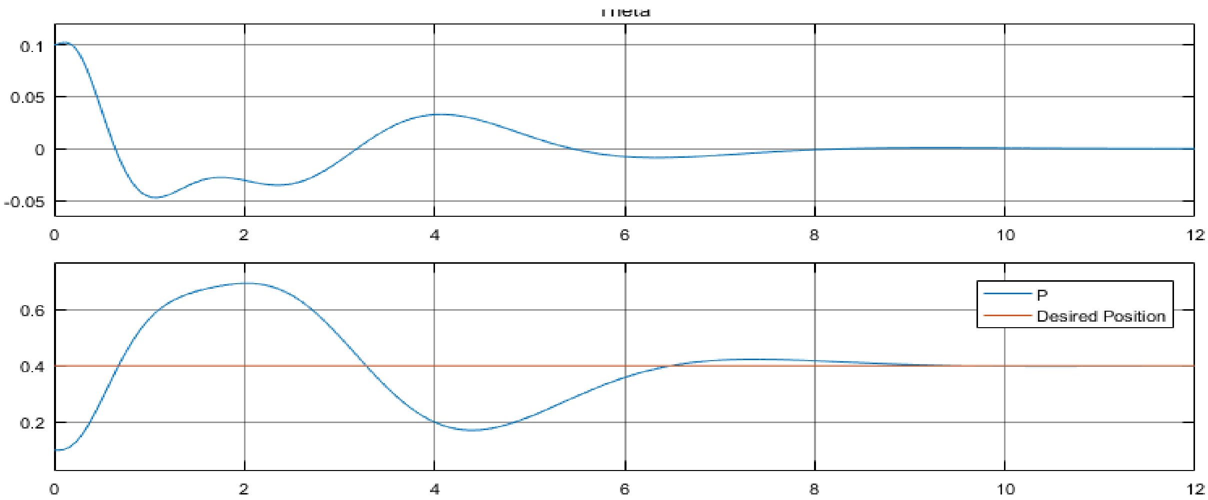



*Figure 12:Simulation pour un retour d'état reconstruit, avec changement de variable*

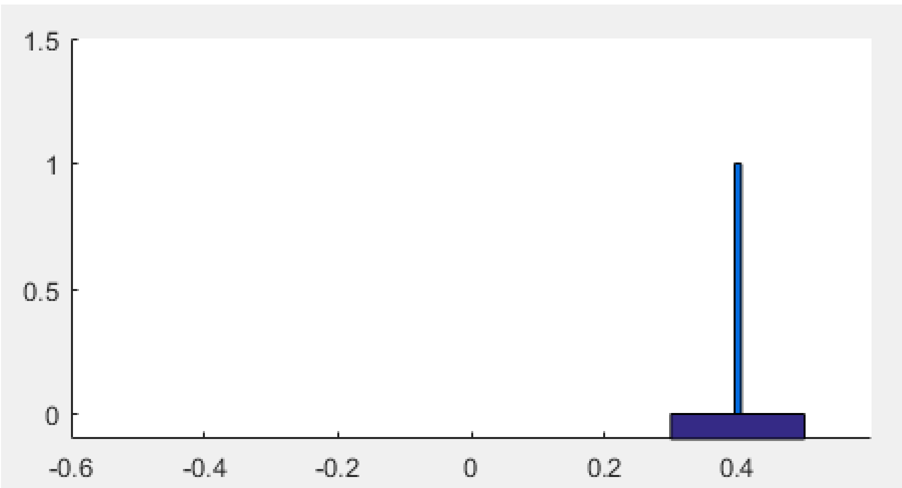



*Figure 13: Animation qui affiche que la barre converge vers la consigne*

## Test de la robustesse :

On va prendre une condition initiale $\theta = 0.3\, rad$ pour assurer qu'il y a des erreurs de modélisation, et on injecte une perturbation de mesure à haute fréquence 0.2.sin(100t),sur la position et sur l'angle. On injecte aussi une perturbation de commande a base fréquence 0.1sin(0.1t). Tout ça pour tester la robustesse contre les erreurs de modélisation, et les différentes de perturbation

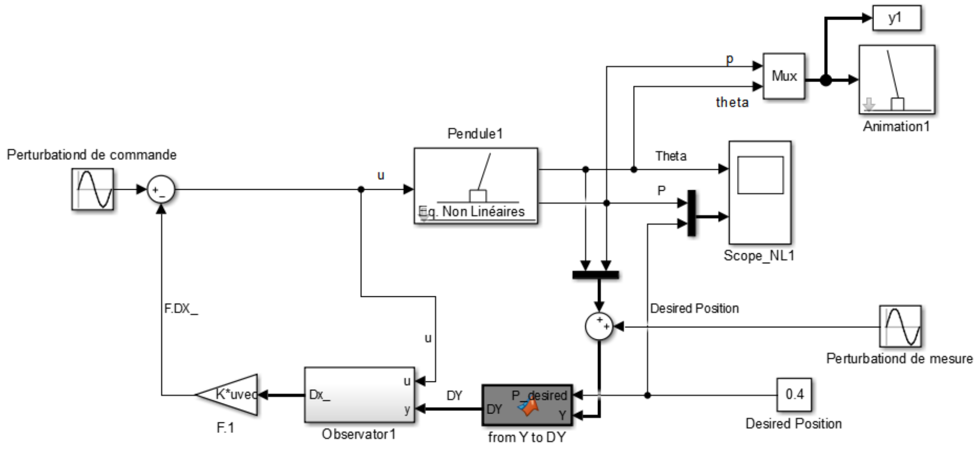
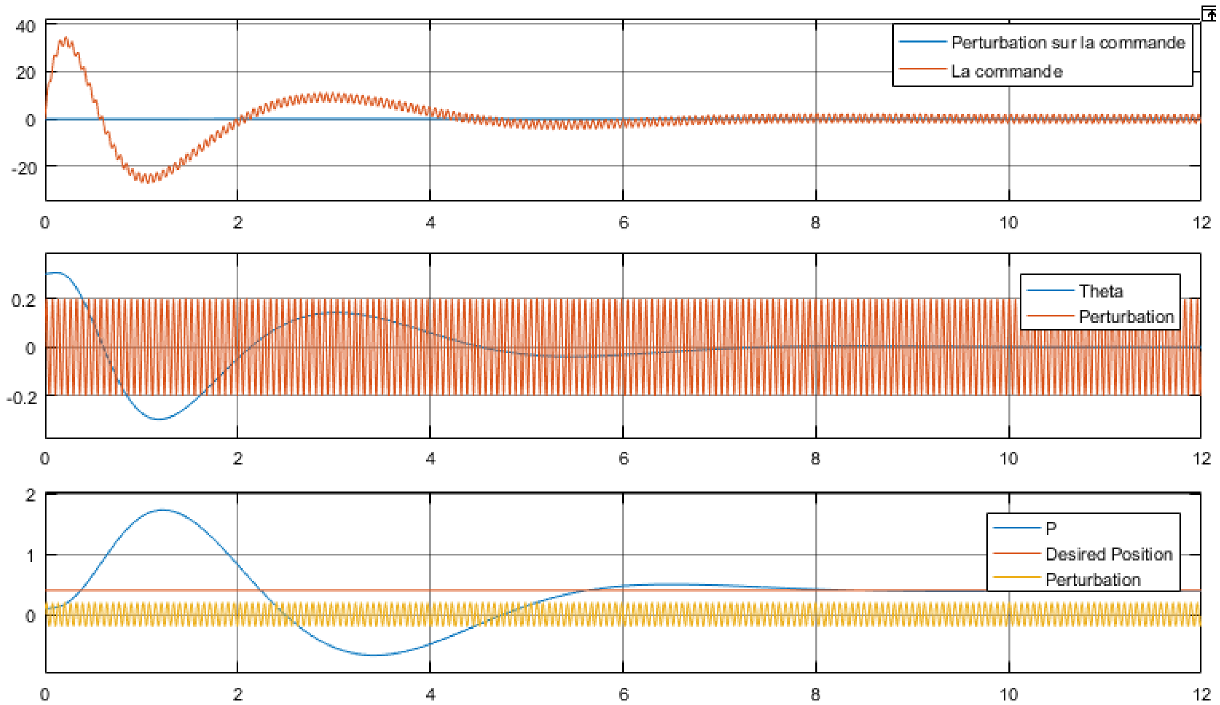



*Figure 14: Schéma Simulink pour tester la robustesse de la commande*




On voie bien que le system a bien rejeté les erreurs de la modélisation, et les effets de perturbations, il a resté stable sans perdre la performance en asservissement. Et d'où la robustesse de la commande.