



Master Ingénierie des Systèmes Complexes

TP1

Asservissement d'un portique de transport

Auteurs :

M. Otmane ATTOU

M. Avotra Ny Aina

Mampionontsoa

RAKOTONDRAVONY

Encadrants :

M. Jean François BALMAT

Version 0.1 du
18 décembre 2019

Table des matières

Introduction	1
1 Modélisation et Etude du portique de transport	3
1.1 linéarisation des équations différentielles :	3
1.2 Le choix du vecteur d'état :	3
1.3 Le système dans l'espace d'état :	4
2 Analyse du système en boucle ouverte	7
2.1 L'étude du système	7
2.1.1 La stabilité :	7
2.1.2 La commandabilité :	8
2.1.3 L'observabilité :	8
2.2 Vérification par simulation numérique	9
2.2.1 Réponse du système	9
2.2.2 La simulation sur Simulink	10
3 Commande par retour d'état	15
3.1 Calcul de la contre réaction L	15
3.2 Résultat et vérification avec la fonction place	17
3.3 Réponse du système à $v(t) = 1$ et valeur finale de $d(t)$	18
3.4 Gain statique	21
3.5 Évolution de $d(t)$ et $\theta(t)$	21
4 Observateur d'état d'ordre plein	23

4.1	Choix de pôles de l'observateur	23
4.2	Détermination de la matrice de gain de l'observateur	23
4.3	Simulation sous Simulink	23
4.4	Observateur de Luenberger et Kalman discret	24
Conclusion		27

Table des figures

1	Portique de transport	1
2.1	Paramètres fixes du problème	7
2.2	La fonction permettant d'obtenir les valeurs propres de A	8
2.3	les valeurs propres de A	8
2.4	La commande permettant de vérifier si le système est commandable	8
2.5	La commande permettant de vérifier si le système est observable	9
2.6	La commandabilité et l'observabilité avec les fonction ctrb et obsv	9
2.7	La commande pour obtenir la réponse du système	9
2.8	La réponse du système	10
2.9	La représentation de $f(t)$	11
2.10	Le schéma-bloc avec la commande state space sur Simulink	11
2.11	Les caractéristiques de la commande state space	12
2.12	La réponse du système	12
2.13	Le schéma bloc en câblant le système	13
3.1	Code permettant d'avoir la matrice de contre réaction	17
3.2	La matrice de la contre réaction obtenues par les deux méthodes	18
3.3	Réponse du système à $v=1$	18
3.4	L'évolution de d et θ avec le temps	19
3.5	Schéma-bloc par retour d'état	19
3.6	La réponse par retour d'état	20
3.7	La réponse par retour d'état en utilisant l'outil data cursor	21
3.8	Le schéma-bloc avec la boucle de contre réaction	22

3.9	d (en jaune) converge bien vers la valeur 1	22
4.1	Code permettant d'avoir l'observateur de Luenberger sur Matlab	23
4.2	Schéma bloc avec retour d'état et l'observateur d'ordre plein	24
4.3	L'évolution de l'état estimé et l'état réel par rapport au temps	24
4.4	Observateur discret et l'initialisation de l'état estimé	25
4.5	Observateur de Luenberger et l'observateur de Kalman	25
4.6	Comparaison entre l'évolution des états estimés et des états réels en fonction du temps	26

Introduction

Un portique de transport est un appareil de levage muni d'une poutre en porte-à-faux et de montants, se déplaçant sur rails ou sur roues et conçu pour manipuler des conteneurs.

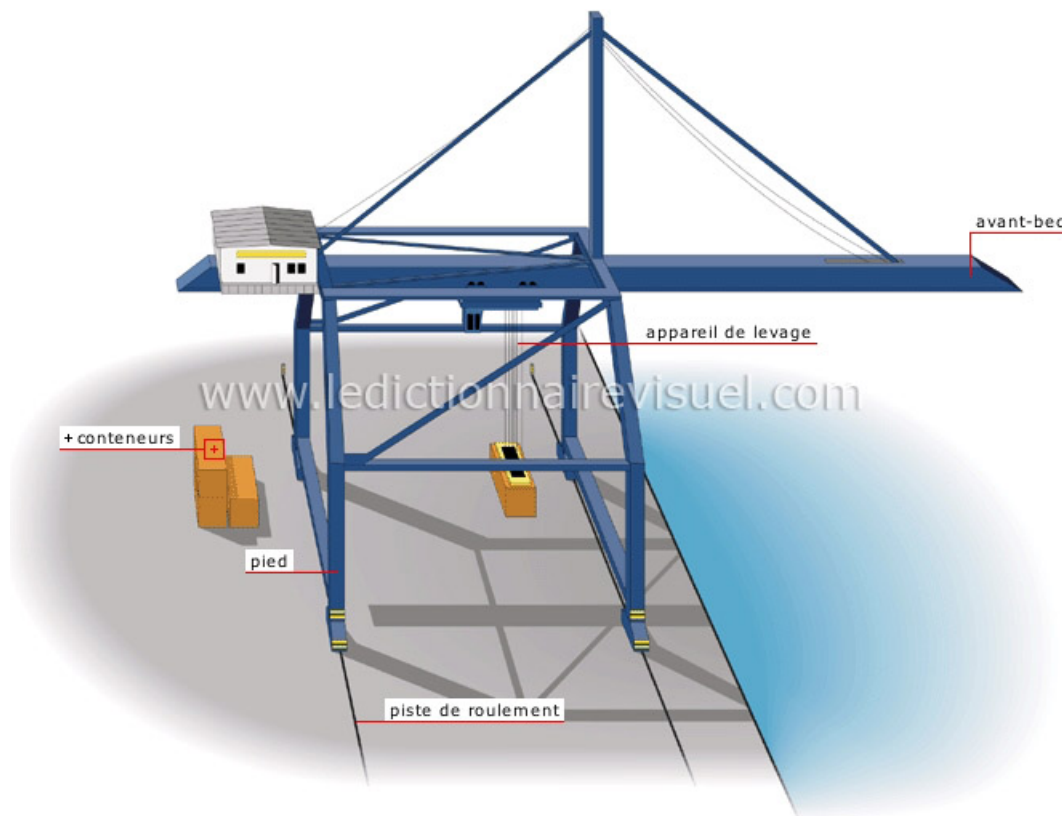


FIGURE 1 – Portique de transport

Chapitre 1

Modélisation et Etude du portique de transport

1.1 linéarisation des équations différentielles :

Les équations différentielles du processus ont été obtenues à partir du bilan des énergies cinétique et potentielle :

$$(M + m).d''(t) + m.l.\theta''.\cos\theta(t) - m.l.\theta'^2.\sin(\theta) + \alpha.d'(t) = f(t) \quad (1.1)$$

$$l.\theta''(t) + d''(t).\cos\theta(t) = -g.\sin\theta(t) \quad (1.2)$$

Comme ce système n'est pas linéaire, nous faisons l'hypothèse de petites variations des grandeurs autour du point d'équilibre $\theta_0 = 0$.

De surcroît, $\theta' = 0, \cos(\theta) = 1$ et $\sin(\theta) = \theta$.

c'est par cette façon que nous trouvons ces équations linéaires :

$$(M + m).d''(t) + m.l.\theta'' + \alpha.d'(t) = f(t) \quad (1.3)$$

$$l.\theta''(t) + d''(t) = -g.\theta(t) \quad (1.4)$$

1.2 Le choix du vecteur d'état :

Les raisons du choix de vecteur d'état ont été choisis en fonction :

1. de paramètres physiques de notre problème.

2. du nombre d'équations différentielles (2), et de l'ordre de ces deux dernières (2).

Donc le vecteur d'état doit contenir quatre éléments ($2 * 2$).

Le vecteur d'état X est le suivant :

$$X = \begin{pmatrix} d(t) \\ d'(t) \\ \theta(t) \\ \theta'(t) \end{pmatrix}$$

et :

$$Y = \begin{pmatrix} d(t) \\ \theta(t) \end{pmatrix}$$

et :

$$U(t) = f(t)$$

1.3 Le système dans l'espace d'état :

En calculant $d''(t)$ en fonction de $\theta''(t)$, $d'(t)$ et $f(t)$ en utilisant la formule (1.3). Puis, nous remplaçons $\theta''(t)$ dans cette dernière équation. Nous trouvons l'expression de $d''(t)$ telle que :

$$d''(t) = \alpha/M.d'(t) + m.g/M\theta''(t) + f(t)/M \quad (1.5)$$

Nous injectons l'expression de $d''(t)$ obtenue dans l'équation (1.5) dans l'expression (1.4) afin d'obtenir l'expression θ'' . Nous obtenons :

$$\theta''(t) = \alpha/(l.M).d'(t) - (m + M).g/(l.M)\theta'(t) - f(t)/(M.l) \quad (1.6)$$

La matrice d'état A et la matrice d'entrée B sont les suivantes :

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & -\alpha/M & g.m/M & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \alpha/(l.M) & -g(m + M)/(l.M) & 0 \end{pmatrix}$$

$$B = \begin{pmatrix} 0 \\ 1/M \\ 0 \\ -1/(M.l) \end{pmatrix}$$

Comme $Y = C.X + D.U$, donc la matrice de sortie C et la matrice de couplage D sont les suivantes :

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$D = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Chapitre 2

Analyse du système en boucle ouverte

2.1 L'étude du système

Nous travaillons sur l'interface de Matlab afin d'analyser les différentes caractéristiques du système étudié.

Dans un premier temps, nous définissons les différentes paramètres fixes du problème. .

```
clc
clear all
close all

%%définir les parametres du systeme

M=10000; %en kg
m=1000; %en kg
alpha=400; %en N/(m/s)
g=10;
l=35;
```

FIGURE 2.1 – Paramètres fixes du problème

2.1.1 La stabilité :

Nous remarquons que la partie réelle de la première valeur propre est nulle, et les valeurs réelles des autres valeurs propres tendent vers 0. On en déduit que le système est dans sa limite mathématique de stabilité. Si nous utilisons le critère de stabilité de Lyapunov. Le résultat ne sera pas le même parce que nous sommes dans la limite de stabilité.

```
%les valeurs propres de A

VA_P=eig(A)
```

FIGURE 2.2 – La fonction permettant d’obtenir les valeurs propres de A

```
VA_P =

    0.0000 + 0.0000i
   -0.0364 + 0.0000i
   -0.0018 + 0.5605i
   -0.0018 - 0.5605i
```

FIGURE 2.3 – les valeurs propres de A

2.1.2 La commandabilité :

Nous calculons la matrice de commandabilité Co, et nous trouvons que cette dernière est de rang plein. Par conséquent, le système est commandable.

```
%commandabilit?|
Co=[B A*B A^2*B A^3*B];
rang_Co=rank(Co); %Nous trouvons que le rang de Obs vaut 4, donc le syst?me est commandable
```

FIGURE 2.4 – La commande permettant de vérifier si le système est commandable

2.1.3 L’observabilité :

Nous calculons la matrice d’observabilité Ob, et nous trouvons que cette dernière est de rang plein. Par conséquent, le système est observable.

```
%observabilit?

Obs=[C; C*A; C*A^2; C*A^3];
rang_Obs=rank(Obs); %Nous trouvons que le rang de Obs vaut 4, donc le syst?me est observable
```

FIGURE 2.5 – La commande permettant de vérifier si le système est observable

Nous pouvons calculer la matrice de commandabilité en utilisant la fonction **ctrb(sys)**. Cette fonction prend comme paramètre en entrée une variable de type **ss** (système) qu'il faut créer avec la fonction **ss(A,B,C,D)** qui quand à elle, prend nos matrices A, B, C et D en entrée. De même, la fonction **obsv** permet d'obtenir la matrice d'observabilité.

```
28      %% observabilité
29 -    sys = ss(A,B,C,D);
30 -    OB = obsv(sys);
31 -    Ro = rank(OB);
32      %% commandabilité
33 -    CO = ctrb(sys);
34 -    Rc = rank(CO);
```

FIGURE 2.6 – La commandabilité et l'observabilité avec les fonctions ctrb et obsv

2.2 Vérification par simulation numérique

2.2.1 Réponse du système

Pour tracer la réponse de notre système aux états initiaux imposés, il faut créer un vecteur x_0 qui va contenir toutes les valeurs de nos états $x_i(d, d', \theta, \theta')$ et l'envoyer avec la variable système que l'on a créée en entrée de la fonction *initial(sys, x₀)* dans matlab.

```
35      %% réponse du system
36 -    x0=[0,0,0.1,0];
37 -    initial(sys,x0)
```

FIGURE 2.7 – La commande pour obtenir la réponse du système

Nous pouvons constater qu'une valeur de 0.1 rad de l'angle que fait le câble avec la verticale à l'instant initial entraîne l'oscillation du câble et ainsi fait balancer la charge, et aussi une oscillation de d qui provoque un mouvement de va et vient du chariot. Au bout de 3 secondes, ces oscillations sont atténuées et le système redevient immobile.

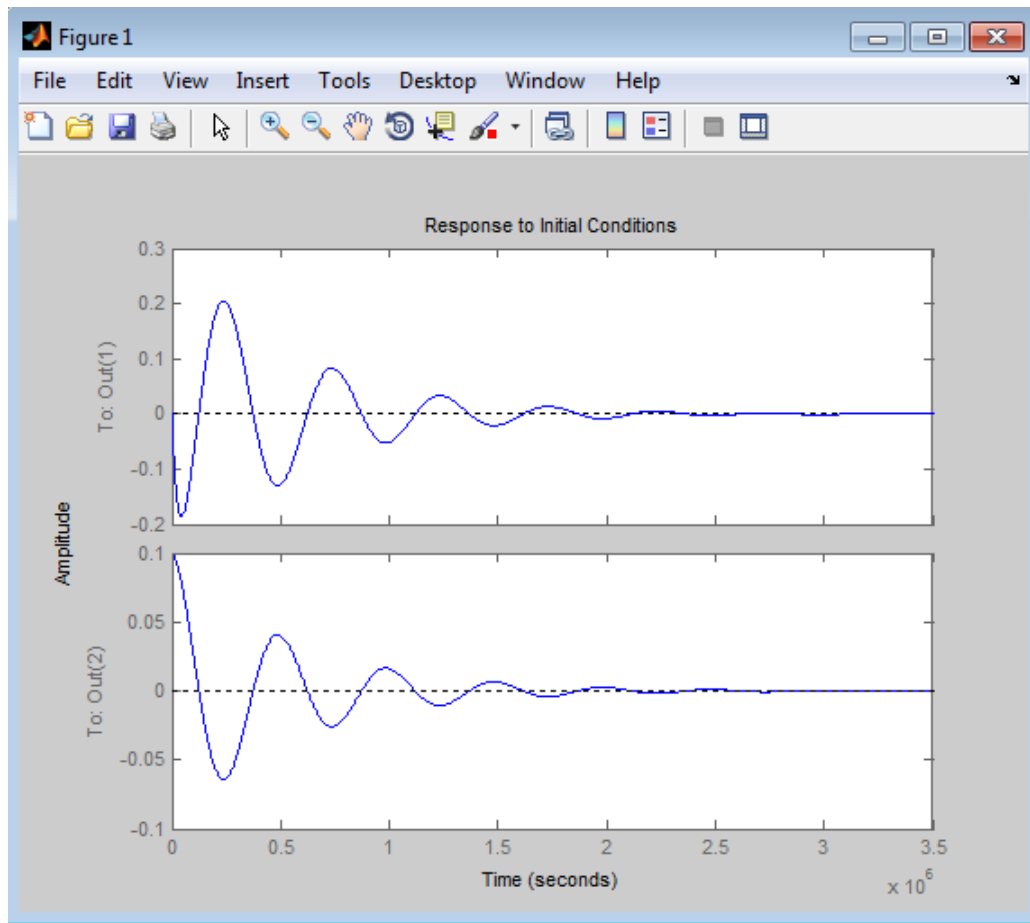


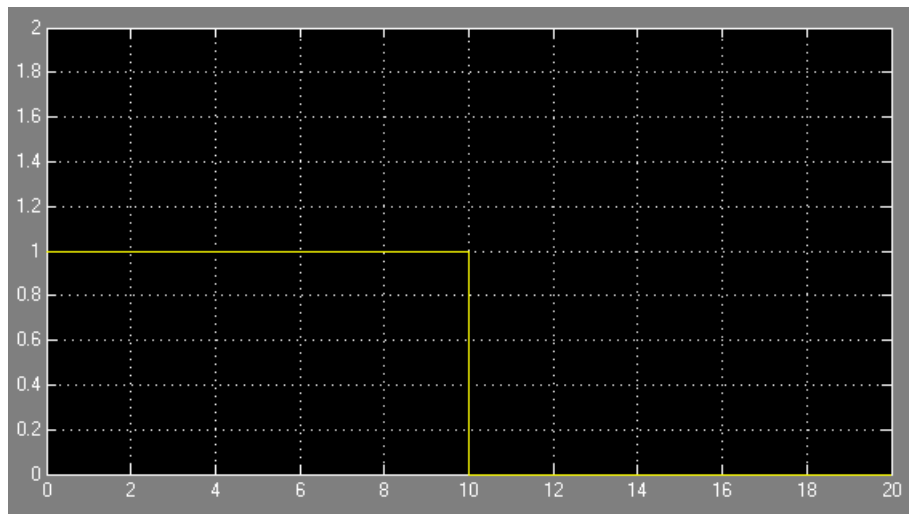
FIGURE 2.8 – La réponse du système

2.2.2 La simulation sur Simulink

Nous commandons ce système par la fonction state-space en prenant une entrée nulle et des conditions initiales définies par le vecteur X_0 suivant :

$$X_0 = \begin{pmatrix} 0 \\ 0 \\ 0.1 \\ 0 \end{pmatrix}$$

Et nous prenons l'entrée $f(t)$ comme un échelon. La figure () montre sa représentation, qui sera utilisé également dans le paragraphe 2.2.3.

FIGURE 2.9 – La représentation de $f(t)$

Avec la fonction state-space

Les figures 2.6 et 2.7 montrent comment nous avons obtenues la réponse en utilisant la fonction state-space. La figure 2.8 montre la réponse.

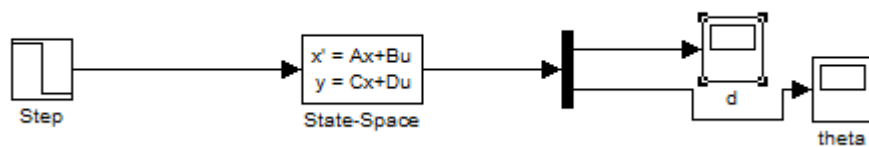


FIGURE 2.10 – Le schéma-bloc avec la commande state-space sur Simulink

Dans ce bloc, il est possible d'entrer directement les matrices du modèle d'état du système.

State-space model:
 $\dot{x}/dt = Ax + Bu$
 $y = Cx + Du$

Parameters

A:
 $[0 \ 1 \ 0 \ 0; 0 \ -\alpha/M \ g*m/M \ 0; 0 \ 0 \ 0 \ 1; 0 \ \alpha/l/M \ -g*(M+m)/(l*M) \ 0]$

B:
 $[0 \ l/M \ 0 \ -1/(M*l)]'$

C:
 $[1 \ 0 \ 0 \ 0; 0 \ 0 \ 1 \ 0]$

D:
 $[0 \ ; 0 \]$

Initial conditions:
 0

Absolute tolerance:
 auto

State Name: (e.g., 'position')
 "

FIGURE 2.11 – Les caractéristiques de la commande state space

On importe le bloc dans notre modèle et on applique à son entrée le signal $f(t)$ et on relie la sortie à l'entrée d'un scope pour avoir les courbes des réponses du système au signal. Les sorties mesurées par les deux scopes donnent les courbes suivantes pour d et θ :

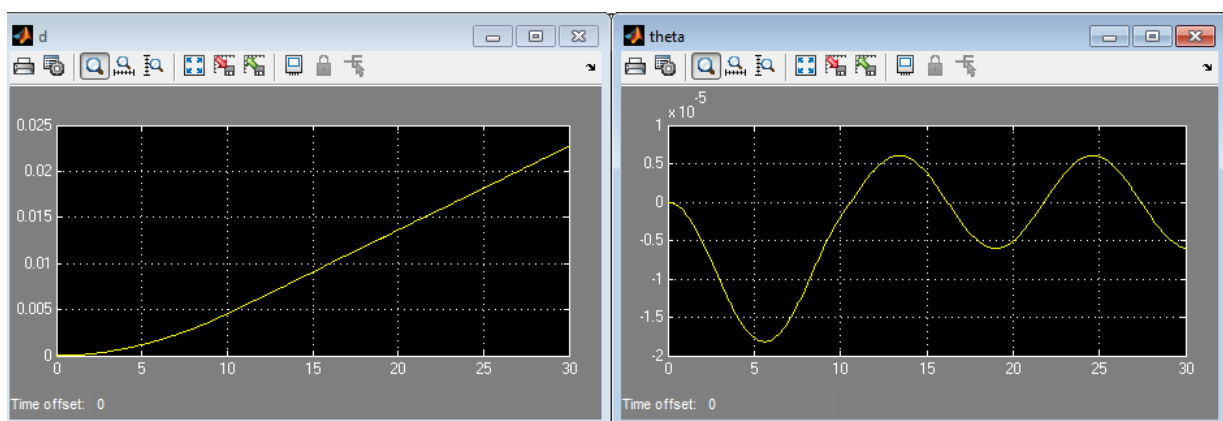


FIGURE 2.12 – La réponse du système

Le schéma-bloc en câblant le système

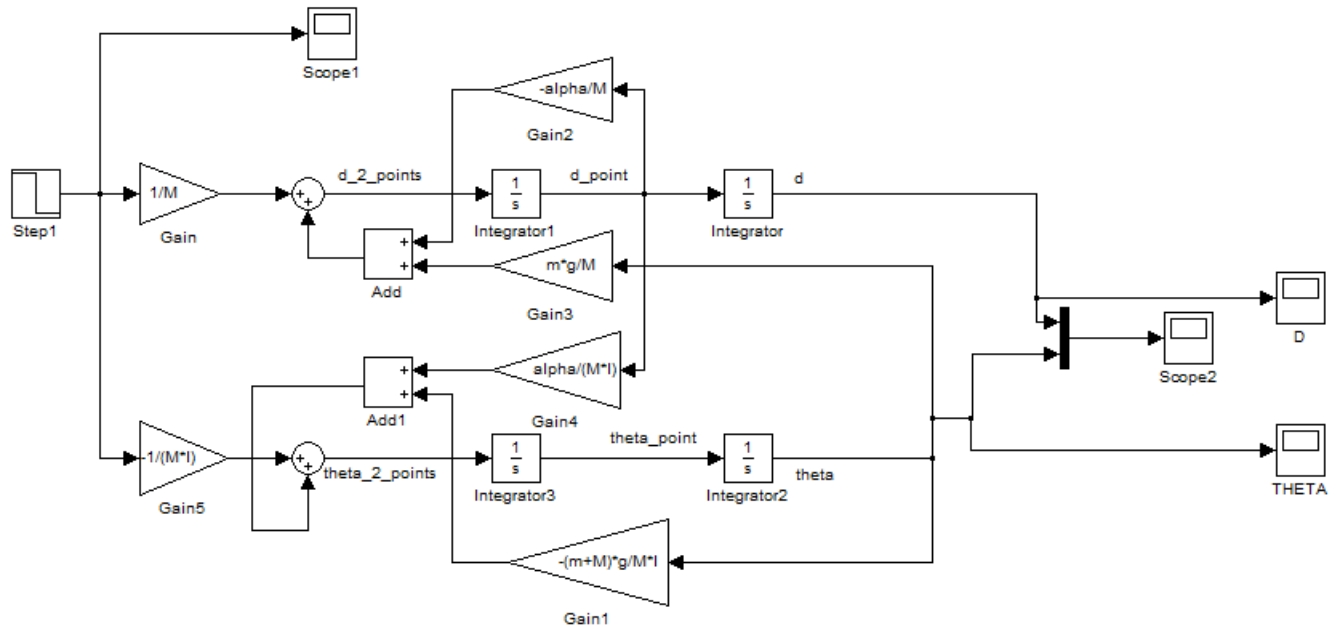


FIGURE 2.13 – Le schéma bloc en câblant le système

Chapitre 3

Commande par retour d'état

On désire réaliser une commande par retour d'état $f(t) = v(t) - Lx(t)$, pour obtenir des valeurs propres en boucle fermée : $2+2j$ $-2-2j$ -5 et -10 .

3.1 Calcul de la contre réaction L

Le calcul de L se fait en quatre étapes :

- Détermination du polynôme caractéristique en boucle ouvert.
- Détermination du polynôme caractéristique désiré.
- Mettre sous forme compagne commandable le système.
- Calculer de L_{com} dans la base de commandabilité.
- Calculer de L dans la base initiale.

1. Étape 1 : Détermination du polynôme caractéristique en BO

Le polynôme caractéristique du système en BO est :

$$|zI - A| = z^n + a_{n-1}z^{n-1} + a_{n-2}z^{n-2} + \dots + a_1z + a_0$$

Dans matlab, la fonction **poly(A)** permet d'avoir le polynôme caractéristique du système en Boucle ouverte.

- (a) Item
- (b) Item

i. Item

2. Étape 2 : Détermination du polynôme caractéristique désiré

Le polynôme caractéristique désiré en boucle fermé s'écrit :

$$P(z) = (z - z_1)(z - z_2) \dots (z - z_n) = z^n + \beta_{n-1}z^{n-1} + \beta_{n-2}z^{n-2} + \dots + \beta_1z + \beta_0$$

Avec z_1, z_2, \dots, z_n les valeurs propres en boucle fermé désirés.

Dans matlab, il faut créer un vecteur $poles = [z_1, z_2, \dots, z_n]$ contenant les valeurs propres désirées. Ensuite, on utilise la fonction **poly(poles)** pour avoir le polynôme caractéristique en boucle fermé désiré.

3. Étape 3 : Forme compagne commandable On a :

$$x(t) = Ax(t) + f(t)$$

$$f(t) = v(t) - Lx(t)$$

En remplaçant la loi de commande l'équation d'état devient :

$$x(t) = (A - BL)x(t) + Bv(t)$$

avec :

$$A_{df} = A - BL$$

$$A_{df} = \begin{pmatrix} -a_{n-1} - l_{n-1} & -a_{n-2} - l_{n-2} & -a_{n-3} - l_{n-3} & \dots & -a_0 - l_0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix}$$

A_{df} est sous forme compagne commandable.

Et l'équation caractéristique du système en boucle fermée est alors :

$$|zI - A_{df}| = z^n + (a_{n-1} + l_{n-1})z^{n-1} + (a_{n-2} + l_{n-2})z^{n-2} + \dots + (a_1 + l_1)z + (a_0 + l_0)$$

4. Étape 4 : Forme compagne commandable

En faisant l'identification : $P(z) = |zI - A_{df}|$ on peut calculer la contre réaction L_{com} dans forme compagne de commandabilité tel que :

$$\begin{cases} a_{n-1} + l_{n-1} = \beta_{n-1} \\ a_{n-2} + l_{n-2} = \beta_{n-2} \\ \dots \\ a_0 + l_0 = \beta_0 \end{cases} \text{ ce qui nous donne : } L_{com} = (l_{n-1} \quad l_{n-2} \quad \dots \quad l_0)$$

Étape 5 : Calcul de L dans la base initiale

Pour calculer L dans la base initiale,

$$L = L_{com}M^{-1}$$

Avec M, la matrice de transition pour passer de la base initiale à la forme compagne de commandabilité telle que :

$$x(k) = Mx_{com}(k)$$

avec :

$$M = \begin{pmatrix} m_n & l_{n-1} & \dots & m_1 \end{pmatrix}$$

tel que pour $j=1$ à $n-1$: $\{ m_n = Bm_{n-j} = Am_{n-j+1} + an - jB_d$

3.2 Résultat et vérification avec la fonction place

Avec matlab, il est possible d'implémenter cet algorithme pour calculer L. Et il est aussi possible de le calculer directement avec la fonction **place (A, B, pôles)** qui prend en paramètre les matrices A et B et aussi le vecteur pôles qui contient les pôles désirés.

```

%% calcule de L
poles = [-2+2i -2-2i -5 -10];


---


%% algo de calcule de L
polynome = poly(A);
polynome_cible = poly(poles); %
Lcom = zeros(1,4);
for i=2:5
    Lcom(i-1)=polynome_cible(i)-polynome(i);
end

M_tran = zeros(4,4);

M_tran(:,1)=B;
for j=2:4
    M_tran(:,j)=A*M_tran(:,j-1)+polynome(j)*B;
end
L1 = Lcom*inv(M_tran);


---


%% calcule de L avec place
L = place (A,B,poles);

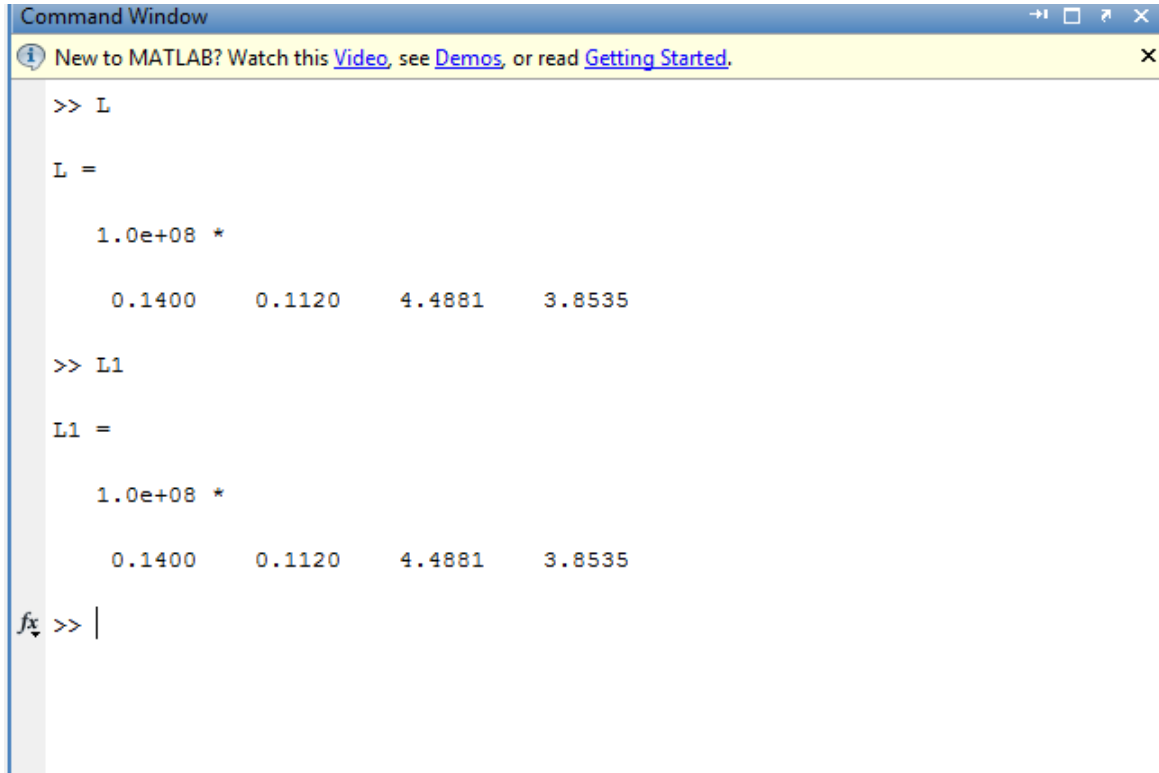

---



```

FIGURE 3.1 – La fonction permettant d'obtenir les valeurs propres de A

Nous pouvons constater que les deux méthodes donnent bien le même résultat.



```

Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> L

L =

    1.0e+08 *
    0.1400    0.1120    4.4881    3.8535

>> L1

L1 =

    1.0e+08 *
    0.1400    0.1120    4.4881    3.8535

```

FIGURE 3.2 – La matrice de la contre réaction obtenues par les deux méthodes

3.3 Réponse du système à $v(t) = 1$ et valeur finale de $d(t)$

Pour avoir les évolutions de θ et $d(t)$, nous avons 2 possibilités :

- Créant un système avec $sys = ss(A_{df}, B, C, D)$ avec $A_{df} = A - BL$.

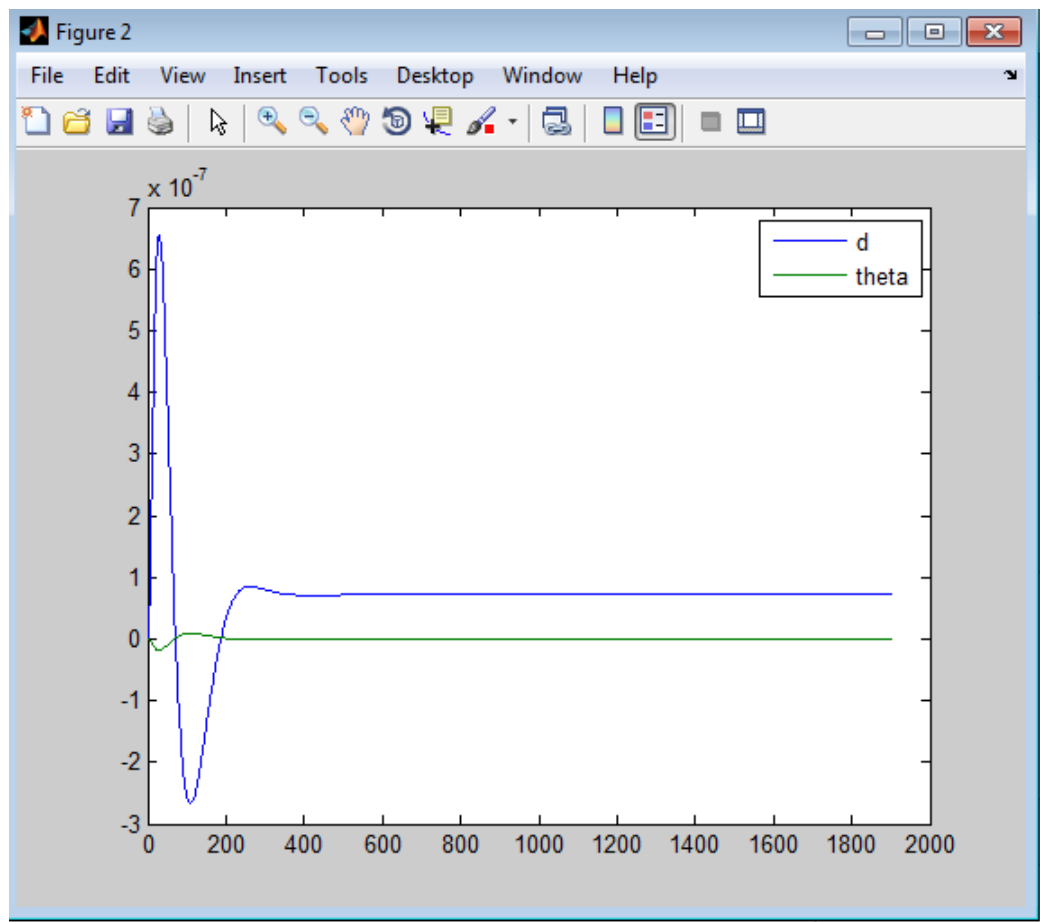
Nous utilisant la fonction **step**() qui calcule la réponse d'un système à un échelon unité et qui prend en paramètre la variable sys.

```

58      %% reponse du systeme à v=1
59 -    Ac = (A - B*L);
60 -    sys1 = ss(Ac,B,C,D);
61 -    y = step(sys1,20);
62 -    figure(2)
63 -    plot (y);

```

FIGURE 3.3 – Réponse du système à v=1

FIGURE 3.4 – L'évolution de d et θ avec le temps

— En câblant le schéma-bloc du système en BF sous simulink.

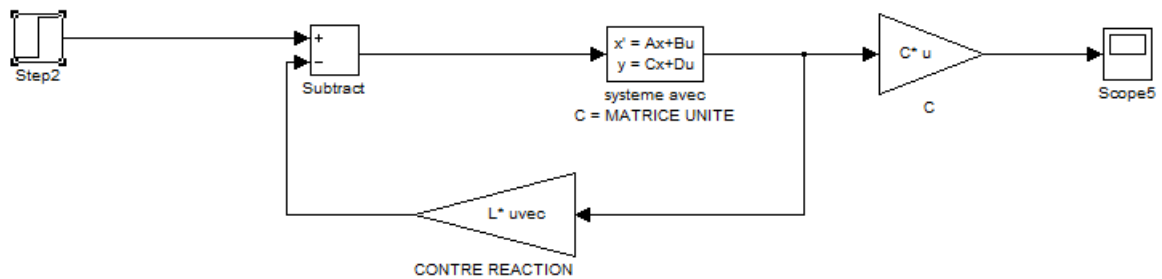


FIGURE 3.5 – Schéma-bloc par retour d'état

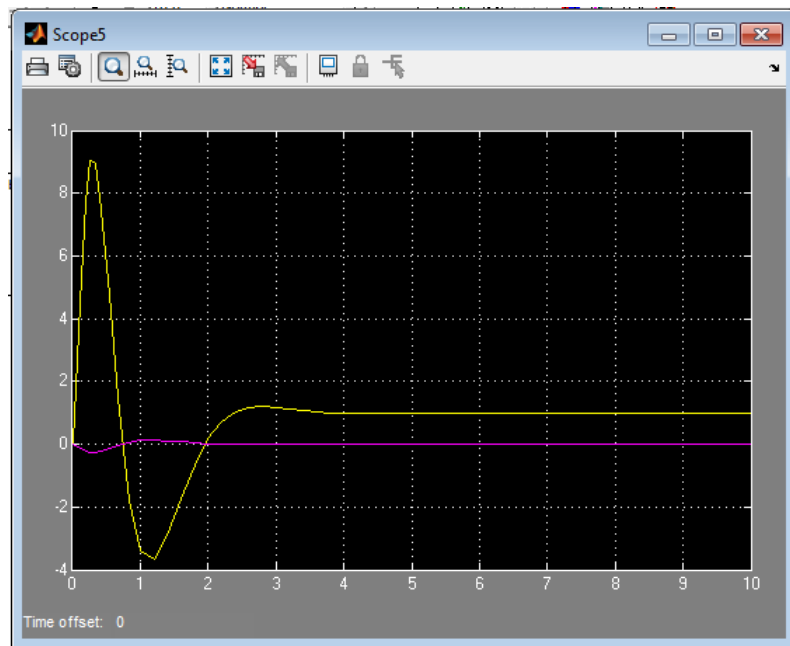


FIGURE 3.6 – La réponse par retour d'état

On peut voir que sur les deux méthodes, on obtient les mêmes résultats pour l'évolution de d et θ . On peut aussi remarquer que d converge vers une valeur constante. On peut avoir cette valeur finale de d avec l'outil **data cursor** qu'on peut utiliser sur les figures dans matlab.

La valeur finale de d est alors : $d_{finale} = 7,14 \cdot 10^{-8}$.

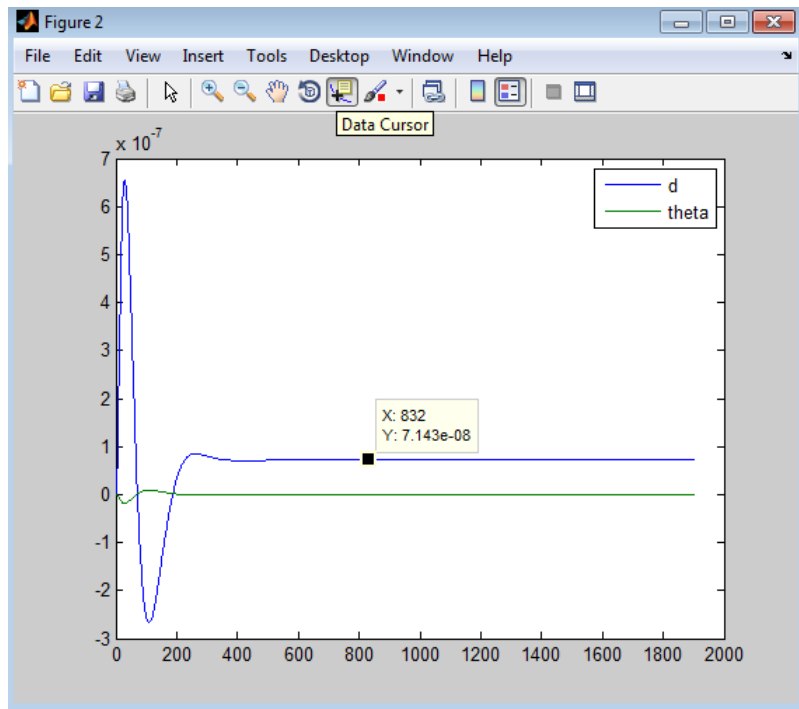


FIGURE 3.7 – La réponse par retour d'état en utilisant l'outil data cursor

3.4 Gain statique

On utilise la loi de commande $f(t) = K.v(t) - L.x(t)$. Pour avoir une erreur statique nulle, on considère que le gain statique du modèle en boucle fermé soit unitaire. Donc : $Kv(t) = d(t)$ or la valeur statique de d est $7,14.10^{-8}$ et $v(t) = 1$. On alors un gain statique :

$$k = \frac{1}{7,14.10^{-8}}$$

3.5 Évolution de $d(t)$ et $\theta(t)$

Sous Simulink, on ajoute au schéma-bloc pour avoir l'évolution de $d(t)$ et $\theta(t)$:

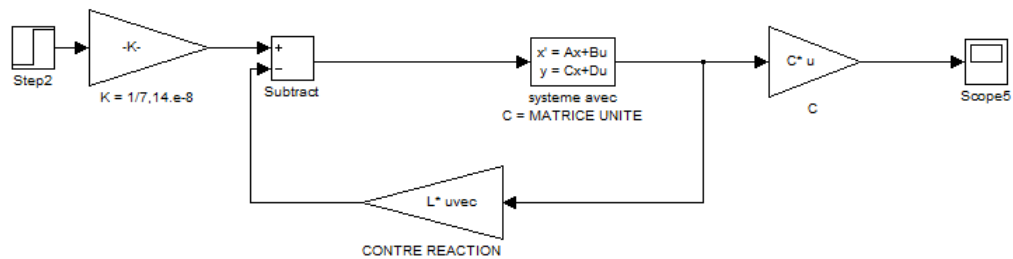
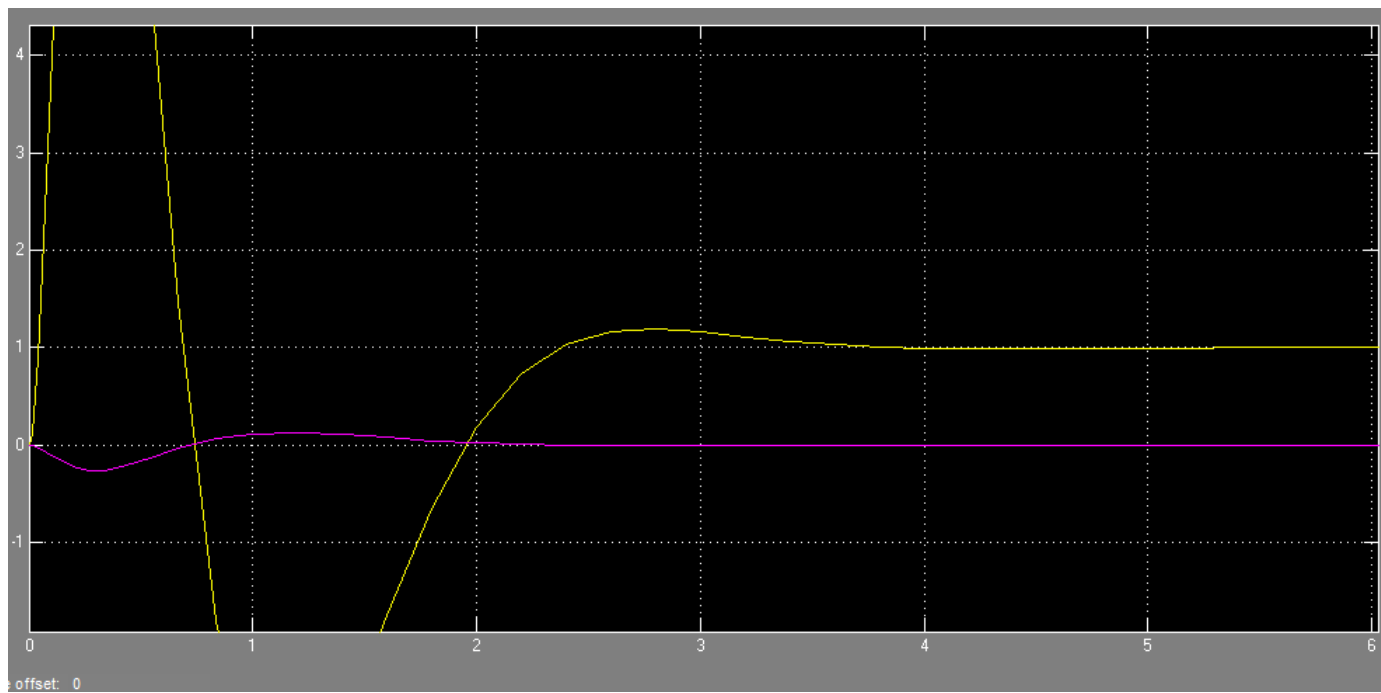


FIGURE 3.8 – Le schéma-bloc avec la boucle de contre réaction

FIGURE 3.9 – d (en jaune) converge bien vers la valeur 1

Chapitre 4

Observateur d'état d'ordre plein

4.1 Choix de pôles de l'observateur

On choisi pour pôles de notre observateur :

$$[-10, -20, -30, -40]$$

4.2 Détermination de la matrice de gain de l'observateur

Pour calculer la matrice de gain, on se réfère à l'annexe 1. Pour cela, on choisit $r = [1, 10]$, ensuite on calcule $g = [0, \dots, 0, 1].Q_o^{-1}.\phi_{bf}(A^T)$. $\phi_{bf}(A^T)$ se calcule avec la fonction **polyvalm(P,A)** de Matlab. Enfin, pour avoir K_{obs} , il faut multiplier g^T par r .

```
64 %% Observateur luenberger
65 - r = [1 10];
66 - poles1 = [-10 -20 -30 -40];
67 - P = poly(poles1);
68 - phi = polyvalm(P,A');
69 - G = [0 0 0 1]*inv(CO)*phi;
70 - Kobs = G'*r;
```

FIGURE 4.1 – Code permettant d'avoir l'observateur de Luenberger sur Matlab

4.3 Simulation sous Simulink

On peut voir que les états estimés avec notre observateur convergent bien vers les mêmes valeurs que les états réels.

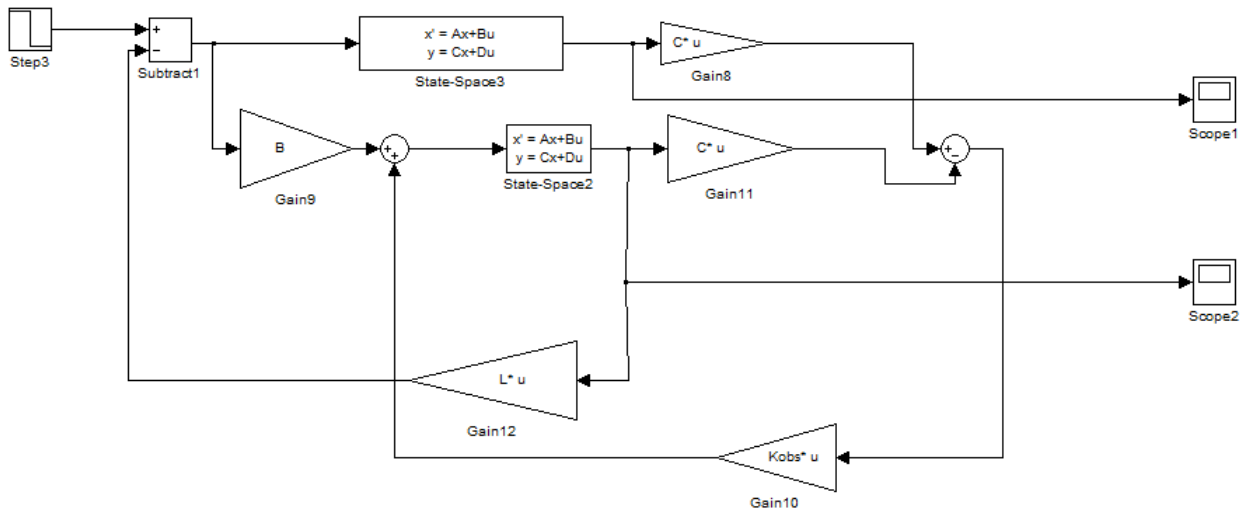


FIGURE 4.2 – Schéma bloc avec retour d'état et l'observateur d'ordre plein

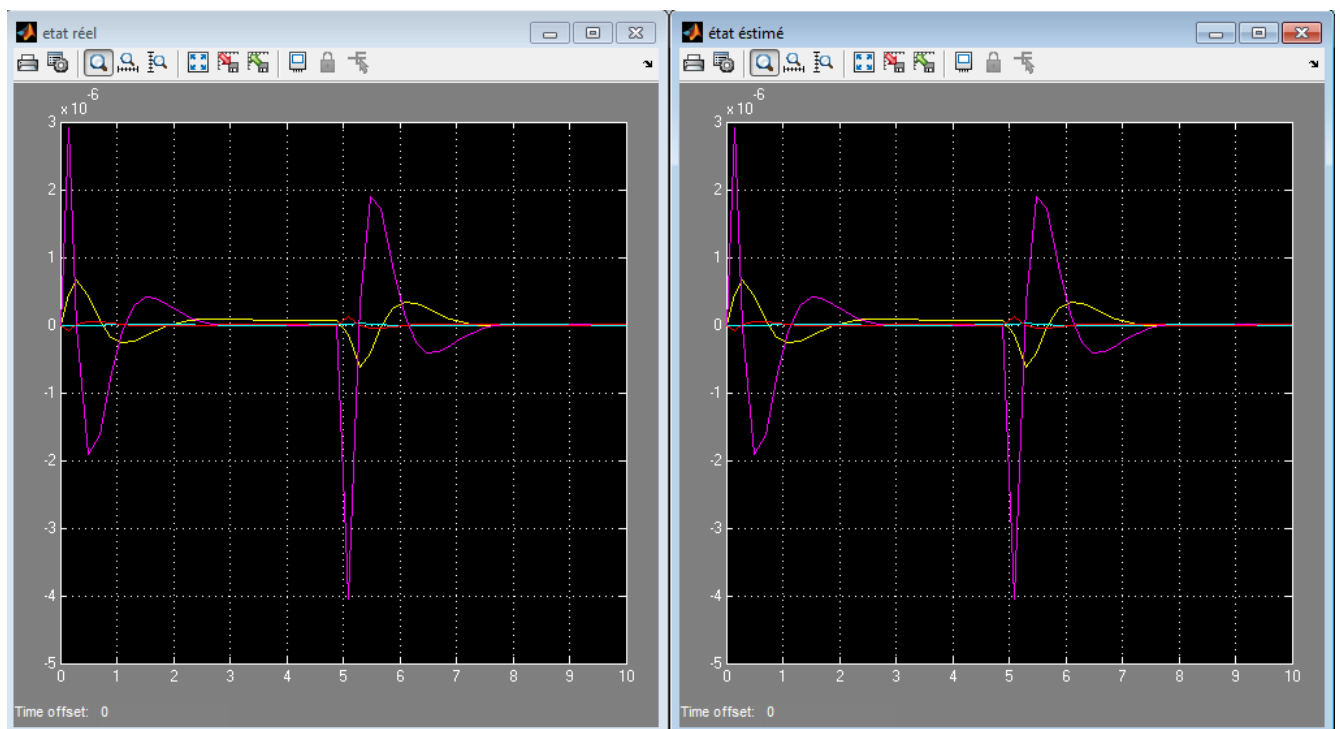


FIGURE 4.3 – L'évolution de l'état estimé et l'état réel par rapport au temps

4.4 Observateur de Luenberger et Kalman discret

Les états estimés avec l'observateur de Luenberger (en rouge) convergent plus vite que les états estimés avec l'observateur de Kalman (en noir). Par contre, l'observa-

teur de Luenberger est plus sensible aux bruit par rapport à Kalman.

```

73      %% observateur discret
74 -    Te = 0.01;
75 -    sys_d = c2d(sys,Te);
76 -    [Ad,Bd,Cd,Dd] = ssdata(sys_d);
77 -    poles_d =[0.15 0.2 0.25 0.3];
78
79 -    Kobs_d = place(Ad',Cd',poles_d); %gain de l'observateur de Luenberg discret
80      %% initialisation
81 -    dure =2000;
82 -    x = zeros(4,dure);
83 -    x_est = zeros(4,dure);
84 -    x_estK = zeros(4,dure);
85 -    x(:,1)= [0 0 0 0]';
86 -    x_est(:,1) =[0.02 0.05 0 0.07]';%initialisation etat estimé Luenberg
87 -    x_estK(:,1)=[0.032 0.04 0 0.03]';%initialisation etat estimé Kalman
88 -    Y_reel = Cd*x;
89 -    Y_est = Cd*x_est;
90
91 -    Q = diag([1 2 3 4]);
92 -    R= diag([1 2]);
93 -    P0= diag(0.01^2*ones(1,4));
94

```

FIGURE 4.4 – Observateur discret et l'initialisation de l'état estimé

```

95 -   for i=1:dure
96 -       if i<10
97 -           u=1;
98 -       else u=0;
99 -       end
100 -    neta = 1e-4*randn(4,1); % Bruit d'état
101 -    nu = 1e-5*randn(2,1); % Bruit de mesure
102      %% Luenberg
103 -    x(:,i+1)=Ad*x(:,i)+Bd*u +neta; % etats réel
104 -    Y_est = Cd*x_est(:,i); % sortie estimé avec luenberg
105 -    x_est(:,i+1) = Ad*x_est(:,i)+Bd*u+Kobs_d'*(Y_reel(:,i)-Y_est); % état estimé avec luenberg
106 -    Y_reel(:,i+1)= Cd*x(:,i+1)+nu; % sortie réel
107      %% Kalman
108 -    Kf=Ad*P0*Cd'*inv(Cd*P0*Cd'+R); %gain de l'observateur de Kalman
109 -    x_estK(:,i+1) = Ad*x_estK(:,i)+Bd*u+Kf*(Y_reel(:,i)-Cd*x_estK(:,i)); % etat estimé avec Kalman
110 -    P0=Ad*(P0-P0*Cd'*inv(Cd*P0*Cd'+R)*Cd*P0)*Ad'+Q; % Mise à jour de PO
111 -   end

```

FIGURE 4.5 – Observateur de Luenberger et l'observateur de Kalman

```

95 - for i=1:dure
96 -     if i<10
97 -         u=1;
98 -         else u=0;
99 -         end
100 -     neta = 1e-4*randn(4,1); % Bruit d'état
101 -     nu = 1e-5*randn(2,1); % Bruit de mesure
102 -     %% Luenberg
103 -     x(:,i+1)=Ad*x(:,i)+Bd*u +neta; % etats réel
104 -     Y_est = Cd*x_est(:,i); % sortie estimé avec luenberg
105 -     x_est(:,i+1) = Ad*x_est(:,i)+Bd*u+Kobs_d'*(Y_reel(:,i)-Y_est); % état estimé avec luenberg
106 -     Y_reel(:,i+1)= Cd*x(:,i+1)+nu; % sortie réel
107 -     %% Kalman
108 -     Kf=Ad*P0*Cd'*inv(Cd*P0*Cd'+R); %gain de l'observateur de Kalman
109 -     x_estK(:,i+1) = Ad*x_estK(:,i)+Bd*u+Kf*(Y_reel(:,i)-Cd*x_estK(:,i)); % etat estimé avec Kalman
110 -     P0=Ad*(P0-P0*Cd'*inv(Cd*P0*Cd'+R)*Cd*P0)*Ad'+Q; % Mise à jour de PO
111 - end

```

FIGURE 4.6 – Comparaison entre l'évolution des états estimés et des états réels en fonction du temps

Conclusion et perspectives

En guise de conclusion, ce TP nous a permis de mettre en pratique les notions théoriques abordées en cours par un système que nous trouvons dans plusieurs applications et domaines. Dans un premier temps, nous avons modélisé et étudié le portique du transport dans l'espace d'état, en s'appuyant sur le bilan des énergies cinétique et potentielle. Ensuite, nous avons analysé le système en boucle ouverte pour conclure sur sa stabilité, sa commandabilité et son observabilité. Ce qui nous a permis d'optimiser sa commande en utilisant la commande par retour d'état. Ainsi, nous avons conçu l'observateur de Luenberger et de Kalman à l'ordre plein. In fine, nous avons présenté la comparaison entre ses deux observateurs.