Compte Rendu TP6 - Commande Optimale

Introduction

Ce TP a pour but la mise en place de différentes méthodes de stabilisation par retour d'état d'un système instable. Ce système est le robot lego EV3. Similaire à un Segway, nous cherchons à le maintenir à la verticale sur ses deux roues. Pour y arriver, nous avons pour but de réaliser 4 commandes. Un commande LQ par retour d'état"classique" puis une version avec intégrateur, une commande par placement de pôle puis enfin avec intégrateur.
En raison de divers problèmes software et hardware, beaucoup de temps a été perdu sur ce TP et tester en réel les différentes parties fut impossible.

Commande LQ

Notre vecteur d'état est de la forme suivante.

$$\mathbf{x_1} = \begin{bmatrix} \theta, & \psi, & \dot{\theta}, & \dot{\psi} \end{bmatrix}^T, \quad \mathbf{x_2} = \begin{bmatrix} \phi, & \dot{\phi} \end{bmatrix}^T, \quad \mathbf{u} = \begin{bmatrix} v_l, & v_r \end{bmatrix}^T$$

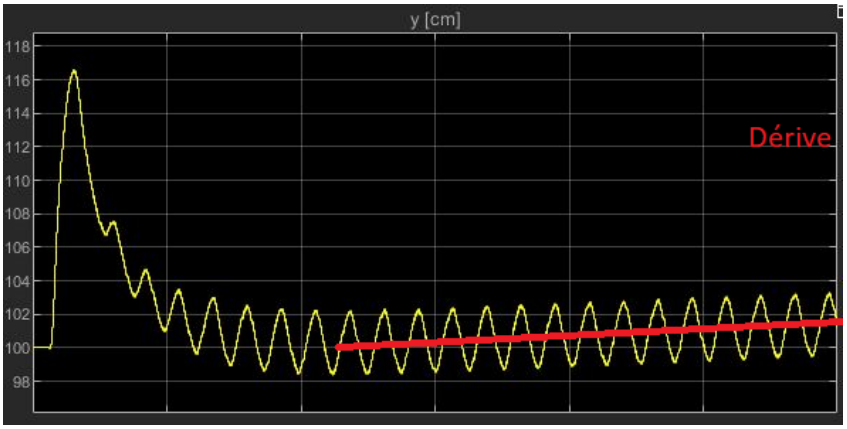
Vecteur d'état

Puisque le modèle répartit le même couple sur les deux moteurs de roues, on peut simplifier u=[v]. Aussi, on cherche R scalaire et Q matrice 4*4. Afin de tester ce modèle, on fixe R à 1 et Q à ρ*Id. En modifiant R, on obtient des résultats équivalents à si l'on modifiait tous les coefficients de Q. Fixer R et jouer indépendamment sur les coefficients de Q nous permet un ajustement plus fin.
On fait varier ρ afin de déterminer la combinaison optimale qui garantie des variations d'angle et de position acceptables aux vues de l'application en stationnaire.

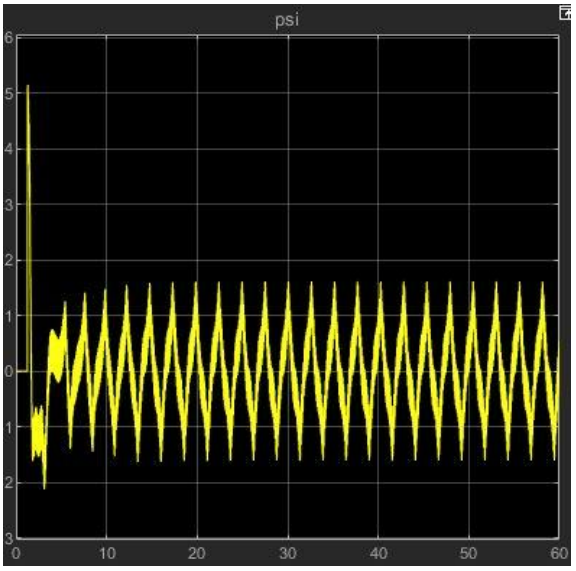
Q1	Q2	Q3	Q4	Dépassement	Angle max	Période angle	Dérive	Oscillation pos (crête à crête)	Commentaire
1	1	1	1	17	1,7	3/10sec	lente	4cm	
0,1	1	1	1	30	0,7	300/10 sec	lente	1cm	
10	1	1	1	10	3	4/10sec	très lente	8 cm	
1	0,1	1	1	17	1,7	3/10sec	lente	6cm	
1	10	1	1	17	1,7	3/10sec	lente	6cm	
1	1	0,1	1	11	0	nulle	lente	inexistante	temps trop important pour revenir à la pos initiale (> 10 sec)
1	1	10	1	65	9	400/ 10 sec	lente	95 cm	pics d'angles à 9° periodiquement
1	1	1	0,1	16	1,7	4/10 sec	lente	4cm	
1	1	1	10	23	3	2,5/10sec	lente	15 cm	

Tableau comparatif des résultats

Afin de choisir les résultats, nous nous sommes basés sur le dépassement de la valeur recherché, sur l'angle maximum pris par le robot, ses oscillations, sa période angulaire et la dérive(chaque résultat comportant une erreur statique).
Nous avons choisi les coefficients(1, 1, 1, 0.1). Cette configuration ayant lavantage de peu dépasser la valeur cible, tout en ayant une légère dérive et des résultats satisfaisant concernant les autres critères.



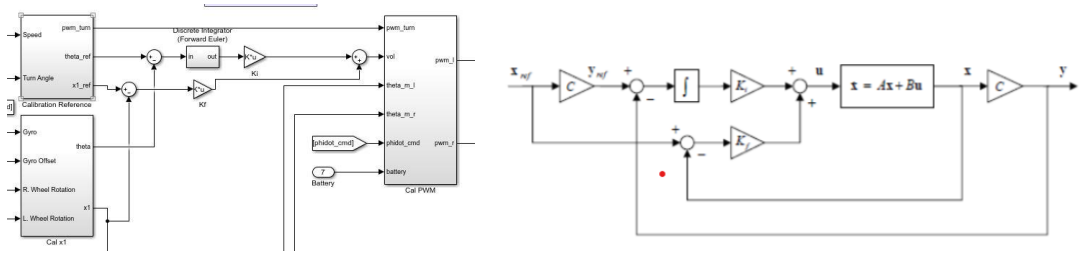
Simulation du déplacement selon y



Simulation de la commande angulaire

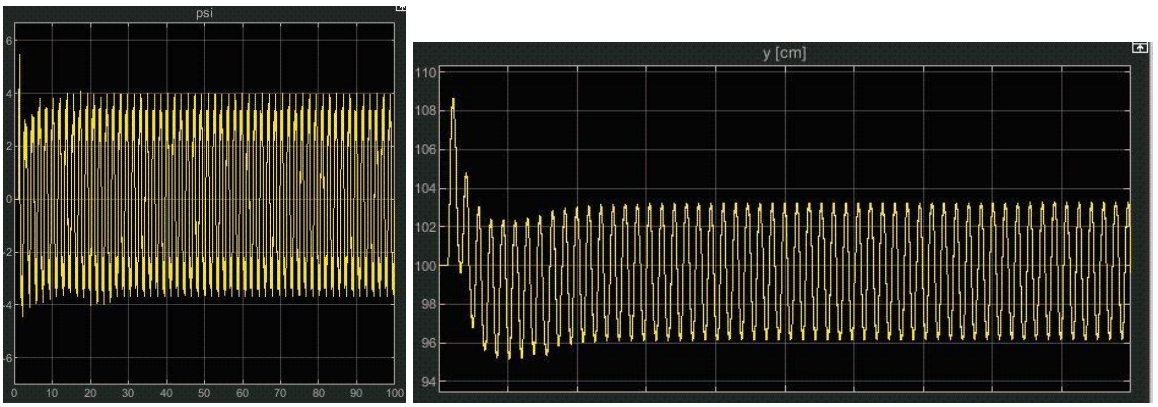
Action Intégrale

Dans la partie précédente, une dérive créée par une erreur statique a été observée. Dans des systèmes de stabilisation comme ce dernier, prendre en compte la dérive est essentiel. Une retour par intégration a ainsi été mise en œuvre pour supprimer l'erreur statique.



Schémas Simulink/bloc avec action intégrale

Pour déterminer Ki, facteur permettant d'agir sur l'intégration, nous avons augmenté notre système puis appliqué la méthode LQR(CF Annexe 2). Les meilleurs résultats ont été observés pour Ki=0.1.



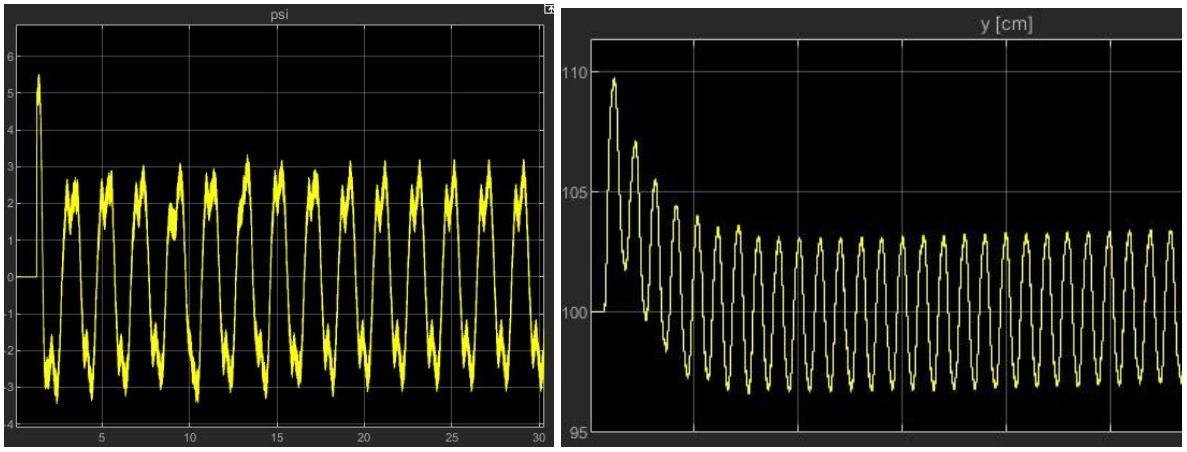
Courbes de résultat avec nos valeurs retenues

En comparaison avec la méthode LQR classique, on constate l'absence de dérive sur la position du robot. L'action intégrale remplit donc correctement son rôle.

Commande par placement de pôles

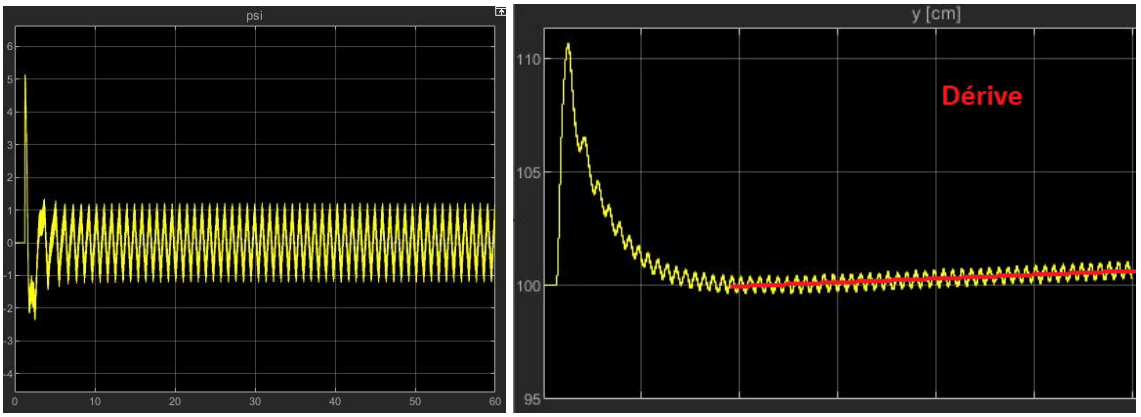
La méthode LQ présente le désavantage de ne pas permettre de choisir les valeurs propres. Afin de mieux maîtriser la dynamique du système, un travail autour du choix de ces dernières a été fait(Cf Annexe 3).

Dans un premier temps, nous avons trouvé les valeurs propres imposées par la commande LQ puis les avons trouver les valeurs de Kf pour imposer ces valeurs propres grâce à la fonction Kf.



Courbes de résultat avec les valeurs propres imposées par la méthode LQ

Les résultats sont comparables à ceux obtenus précédemment. En faisant varier les valeurs propres imposées au robot, on a réussi à améliorer son comportement.

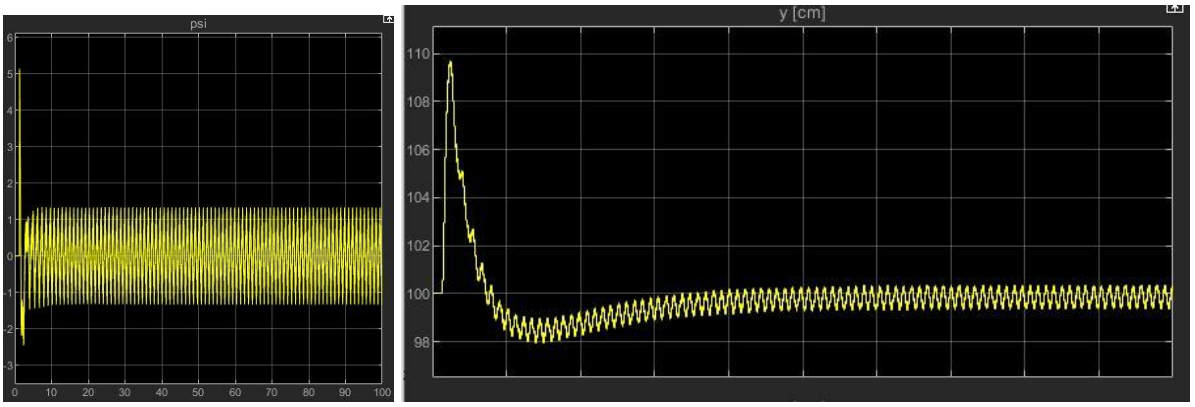


Courbes de résultat avec les valeurs propres optimales

Comparé à la méthode LQ, le placement de pôle offre de bien meilleurs résultats que ce soit en terme de dépassement qu'en termes d'oscillation de la commande angulaire. Cependant une dérive est de nouveau présente.

Placement de pôles par action intégrale

Dans le but de supprimer la dérive observée dans la partie précédente, une nouvelle action intégrale a été mise en œuvre (Cf Annexe 4).



Simulation avec les valeurs propres optimales et l'action intégrale

L'absence de dérive en régime permanent est remarquable, signe de la bonne mise en œuvre de l'action intégrale. De plus, il a été possible de garder des résultats bien meilleurs à ceux obtenus avec la méthode LQR

Conclusion

Ce TP nous a permis de mettre en œuvre deux approches assez différentes afin de commander ce système et comment les optimiser grâce à une action intégrale. La méthode LQ offre une grande facilité de mise en œuvre mais possède un résultat perfectible. La méthode par placement de pôle quant à elle est plus complexe car nécessite une connaissance du système étudié, mais est plus performante.

Nous avons donc à travers ce TP utilisé la méthode LQ pour aborder le robot, puis celle par placement de pôles pour affiner nos résultats. Effectuer la méthode LQ en premier puis de se rendre compte de l'influence des pôles ainsi générés à été très formateur, malgré le fait qu'il ait été terni l'impossibilité de tester en réel ces méthodes.

Annexe 1

```
% Controller Parameters

% Copyright 2011 The MathWorks, Inc.

% Servo Gain Calculation using Optimal Regulator
QQ = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 .1];
 RR = 1  ;

KK = lqr(A1,B1,QQ,RR)  ;
vp = eig(A1-B1*KK);
KK = KK(1,:);
```

Code de la commande LQR

Annexe 2

```
% PARTIE 2
%on recrée des matrices "augmentées" càd on ajoute un paramètre q5 pour
%jouer sur l'intégration

C=[1 0 0 0];

Z_col=zeros(4,1);

A1i = [A1 Z_col
        C 0];

Z_lig=zeros(1,2);

B1i = [B1
    Z_lig];


q5 =0.1;

QQi = [q1 0 0 0 0; 0 q2 0 0 0 ; 0 0 q3 0 0; 0 0 0 q4 0 ;0 0 0 0 q5];

RRi = [1 0
      0 1];

KKi=lqr(A1i, B1i, QQi, RRi);
```

Code de la commande LQR avec action intégrale

Annexe 3

```
% PARTIE 3

Kf_poles = place(A1, B1, vp);

vp_test = [vp(1)+30 vp(2)+1 vp(3) vp(4) ]
KF_test = place(A1, B1, vp_test);
```

Code de la commande par placement de pôle

Annexe 4

```
% PARTIE 4

Ai_BF = A1i - B1i*KKi;
pole_Ai_BF=eig(Ai_BF);
%choix des pôles optimisant la réponse
poles_int = [-220.98235 + 0.000000000000000i;-7.14 + 1.25i;-7.14 - 1.25i;-1.60;-0.1];
Ki_poles=place(A1i,B1i,poles_int);
```

Code de la commande par placement de pôle avec action intégrale