Compte Rendu - TP6
Commande optimale : le pendule inversé

Février 2022

1 Introduction

L'objectif de ce TP est d'implémenter différentes commandes sur un robot EV3 afin que ce dernier se tienne debout (à la verticale). Le robot étant un système instable sur deux roues, il nous faut implémenter une régulation pour atteindre l'objectif. Pour cela, nous sommes passés par quatre commandes différentes : une commande LQ, une commande avec action intégrale et enfin une commande par placement de pôles sans puis avec action intégrale.

2 Commande LQ

Dans cette première approche nous allons commander le robot par retour d'état, le but étant d'obtenir un gain le plus optimal possible. Pour cela, dans le fichier MATLAB (Annexe 1), nous complétons les matrices QQ et RR afin d'extraire KK par la commande $'lqr'$. On définit les coefficients q_i et r_i pour pouvoir changer rapidement les valeurs des diagonales. On fait varier ces coefficients par puissances de 10. Le fichier Simulink est complété par la suite pour réaliser la commande LQ (Annexe 2).

2.1 Tableau comparatif en simulation pour différentes valeurs des coefficients q_i

Coefficient modifié : q1	Ψ : premier dépassement (°)	Ψ : en régime permanent « peak to peak » (°)
1	5,1	2,71
10	5,15	6,15
100	5,8	19,6

Coefficient modifié : q2	Ψ : premier dépassement (°)	Ψ : en régime permanent « peak to peak » (°)
1	5	2,71
10	5	2,67
100	5	2,88
1000	5	2,76
10 000	5	2,3
100 000	5,3	0,77

Figure 1: Tableau comparatif pour différentes valeurs de q_1 et q_2

2.2 Commande optimale obtenue en manipulation

Après les différents essais récapitulés dans le tableau comparatif précédent, nous avons choisi de garder les valeurs de coefficients des matrices QQ et RR suivantes :

$$q_1 = q_3 = q_4 = 1; \quad q_2 = 1000$$

$$r_1 = r_2 = 1$$

En effet, c'est pour ces valeurs que la simulation avait le meilleur comportement tout en étant conforme à la réalité lorsque nous l'avons testée sur le robot. q_2 est le coefficient agissant sur ψ, l'angle réalisé par le corps du robot par rapport à la verticale. Les tests ont montré que lorsqu'on augmente ce poids, le robot réussit mieux à garder sa position 'debout'. Pour $q_2 = 10e5$, la commande paraissait optimale mais en réalité le robot a un comportement très saccadé pour garder sa position verticale. Un bon compromis était donc de choisir $q_2 = 10e3$.

Concernant q_1, en augmentant sa valeur, la valeur prises par ψ deviennent de plus en plus grandes (physiquement cela signifie que le robot a tendance à se pencher beaucoup plus). En modifiant r_1 et r_2 ou encore q_4 nous nous attendions à adoucir la commande, pourtant ce n'est pas l'effet que nous avons observé; l'effet a plutôt été négatif pour la posture du robot donc nous avons gardé 1 pour chacune de ces valeurs.

L'évolution de l'angle ψ se trouve en Annexe 3. On remarque que la variation 'peak to peak' de l'angle est de $2,76°$ ce qui est tout à fait correct.

Finalement, nous avons observé le déplacement du robot selon l'axe y. On remarque une légère déviation en simulation (Annexe 4) qui correspond bien à la réalité puisque le robot avance et recule. Cela va être réglé par l'impémentation d'une action intégrale dans la prochaine simulation.

3 Action intégrale

Afin de régler le soucis rencontré précédemment avec la trajectoire du robot, on ajoute un intégrateur dans la chaîne directe de la commande. Ceci est illustré sur le modèle Simulink en Annexe 5.

On procède ensuite à l'augmentation de la matrice QQ avec la paramètre q_5 qui jouera sur l'intégration. Le code MATLAB se trouve en Annexe 6.

3.1 Tableau comparatif en simulation pour différentes valeurs du coefficient q_5

Coefficient modifié : q5	Ψ : premier dépassement (°)	Ψ : en régime permanent « peak to peak » (°)
1	5,08	5,46
10	5	8,49
100	5,48	12,61

Figure 2: Tableau comparatif pour différentes valeurs de q_5

3.2 Commande optimale obtenue en manipulation

Après les différents tests nous avons choisi de garder la valeur de $q_5 = 1$ car c'est celle qui permettait de stabiliser au mieux le robot. Nous n'avons pas modifié les autres paramètres de la matrice QQ.

Le gain optimal résultant est donc (on n'écrit qu'une seule ligne car les deux sont les mêmes) :

$$KKi = -1.5370 - 51.5222 - 1.8313 - 5.0096 - 0.7071$$

En testant cette commande en réel sur le robot, nous avons remarqué qu'il y avait beaucoup moins de variations en y. Le robot a tendance à rester sur place tout en effectuant de très légères rotations sur lui-même (quasiment imperceptibles). La photo en Annexe 7 souligne la correction de la déviation en y grâce à l'action intégrale. On s'approche donc de plus en plus du but recherché : la stature "debout" du robot. De plus, il réussit à retrouver une posture verticale stable lorsqu'on le perturbe d'une tape modérée dans le dos.

En testant la commande avec $q_5 = 1000$, la simulation concorde bien avec la réalité car le robot, trop instable, se met à tourner trop et tombe rapidement (encore plus lorsqu'on le perturbe en le poussant en peu).

4 Commande par placement de pôles

Afin de pouvoir choisir un peu plus la dynamique du système par rapport à une commande LQ qui impose tout toute seule, on propose d'implémenter une commande par placement de pôles. Il suffit pour cela d'affiner les paramètres de la commande LQ précédente. Dans un premier temps nous avons trouvé les valeurs propres qui étaient imposées par la commande LQ optimale dans la première partie.

```
%extraction des pôles

A_BF=A1-B1*KK;

pole_A_BF=eig(A_BF);
```

```
pole_A_BF =

 -257.8225
  -13.0806
   -0.8185
   -4.0453
```

Le gain K_f qui permettait d'imposer ces pôles est :

$$K_f = -0.7071 - 47.8766 - 1.6099 - 4.5084$$

On décide désormais d'imposer nos propres pôles pour affiner la commande et obtenir une réponse plus stable. Le tableau suivant résume les valeurs testées pour les deux derniers pôles et en Annexe 8 à 10 se trouvent les réponses du système pour chacune des commandes implémentées en simulation.

	Simu 1	Simu 2	Simu 3
3ème pôle	-1	-1	-1
4ème pôle	-1.1	-5	-10

Figure 3: Couples de valeurs pour les deux derniers pôles testés en simulation

Au vu des résultats, en choisissant les deux derniers pôles avec les valeurs -1 et -1.1, nous obtenons une réponse tout à fait satisfaisante (les autres ayant une variation de ψ bien trop importante).

En testant en réel sur le robot avec ces valeurs, on remarque que l'angle du corps est quasiment nul (voire imperceptible); par contre le robot avance et recule d'une dizaine de cm. Il ne tombe jamais. En appliquant une perturbation, le robot est extrêmement réactif et se stabilise très vite. Seule une grosse perturbation (d'un angle de plus de 30° environ) fait tomber le robot. Tous ces résultats en réel concordent avec la simulation.

```
%choix des pôles
poles=[ -257.8225, -13.0806, -1, -1.1];

%gain pour imposer ces pôles
K_poles=place(A1,B1,poles);
```

Figure 4: Code MATLAB - choix de la valeur des deux derniers pôles pour réponse optimale

On obtient le gain optimal suivant :

$$K_{poles} = -0.2349 - 39.8100 - 0.9818 - 3.0789$$

Nous avons testé en réel avec les valeurs des derniers pôles à -1 et -5. Le robot a moins de variations selon l'axe y par contre il oscille beaucoup plus et a du mal à rester droit. En appliquant une perturbation, le robot reste relativement stable (moins que pour la commande précédente toutefois).

Par contre, quelle que soit la commande, on observe à nouveau une déviation en y (Annexe 11). Nous allons donc passer à une commande par placement de pôles avec action intégrale afin de remédier à cet écart.

5 Placement de pôles avec action intégrale

Afin de corriger la déviation du robot dans l'espace, on décide d'insérer une action intégrale à la commande par placement de pôles.

```
%extraction pôles intégration + placement de pôles

Ai_BF = A1i - B1i*KKi;

pole_Ai_BF=eig(Ai_BF);

%choix des pôles (inchangés pour cette partie)
poles_int=[-257.82,-13.08,-40.5,-0.76 + 0.5i,-0.76 - 0.5i];

Ki_poles=place(A1i,B1i,poles_int);
```

Figure 5: Code MATLAB - Action intégrale et placement de pôles sans modification de ces derniers

La simulation de cette commande est jointe en Annexe 12. La réponse est acceptable. Commes ces pôles sont déjà très proches de 0 et par manque de temps, nous avons gardé ces valeurs de pôles car en réel ils donnent aussi un excellent résultat. En effet, le robot est très stable, l'angle du corps est nul et la déviation en y (Annexe 13 pour la simulation) est très faible (de l'ordre de 2 à 3 cm).

Le gain nécessaire à l'implémentation de cette commande est donné ci-dessous :

$$Ki_{poles} = -36.4 + 462.3i, -222.3 + 1177.3i, -14.4 + 50.6i, -33.4 + 113.6i, -221.2 + 325.4i$$

De plus, lorsqu'on applique une perturbation, le robot se stabilise très rapidement. L'objectif de stabilisation de la posture verticale est donc atteint.

6 Conclusion

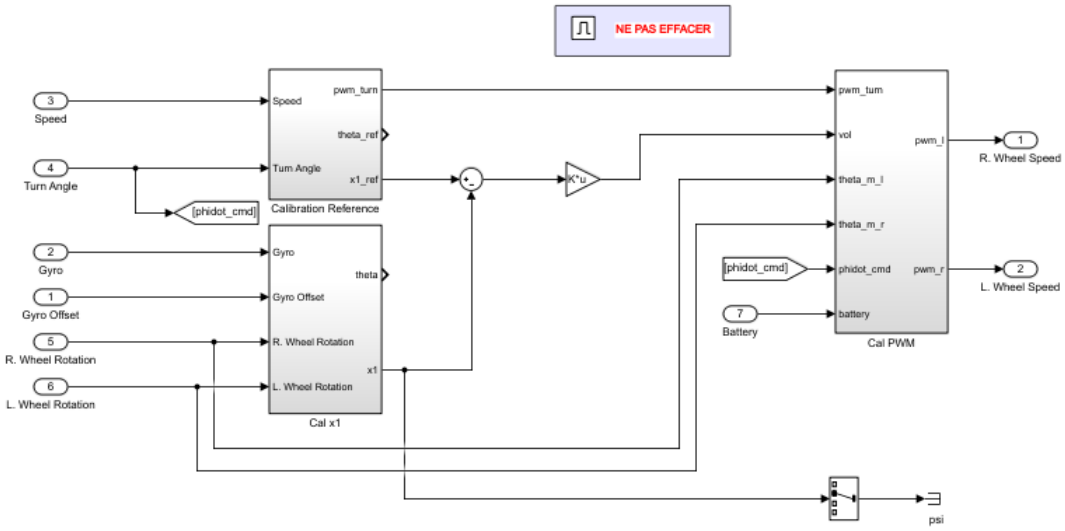
Ce TP nous a permis d'étudier l'influence de différentes commandes pour la stabilisation d'un système instable : un robot sur deux roues. Au fur et à mesure des manipulations, nous avons réussi à améliorer la réponse du système et à satisfaire l'objectif fixé au départ à savoir maintenir le robot à la verticale. Nous avons pu mettre en évidence l'impact d'une action intégrale sur le système mais également celui du choix des pôles pour avoir une réponse plus "douce". Finalement, la combinaison des deux a permis d'effectuer une double action sur la réponse : la réduction des oscillations du corps du robot ainsi que l'annulation du déplacement en avant/arrière de ce dernier. Une étude plus approfondie aurait certainement pu permettre de dégager de meilleures valeurs pour les différents coefficients, plus "optimales" pour la stabilisation du système. Cela dit, les résultats restent très convenables et les paramètres choisis conduisent tout à fait à la satisfaction du besoin initial.

Annexes

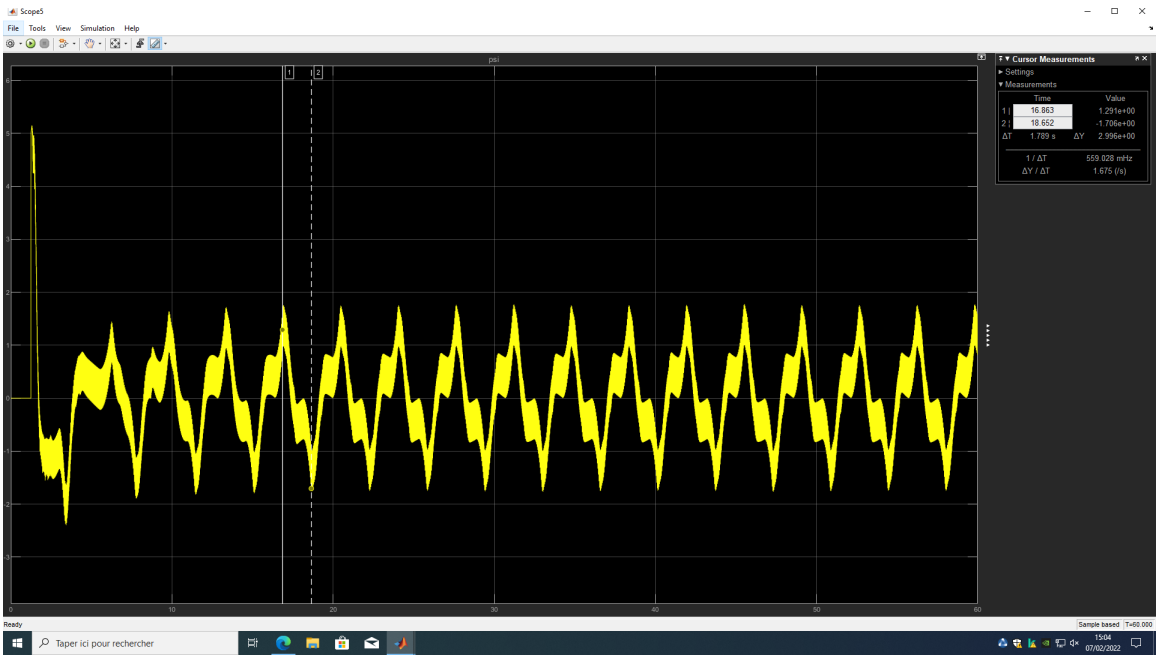
Commande LQ

```
%on part des coefficients à 1 (Identité) et on
%augmente par puissance de 10

q1 = 10e0;
q2 = 10e3;
q3 = 10e0;
q4 = 10e0;

r1 = 10e0;
r2 = 10e0;

QQ = [q1 0 0 0
      0 q2 0 0
      0 0 q3 0
      0 0 0 q4];

RR = [r1 0
       0 r2];

KK = lqr(A1,B1,QQ,RR);
```

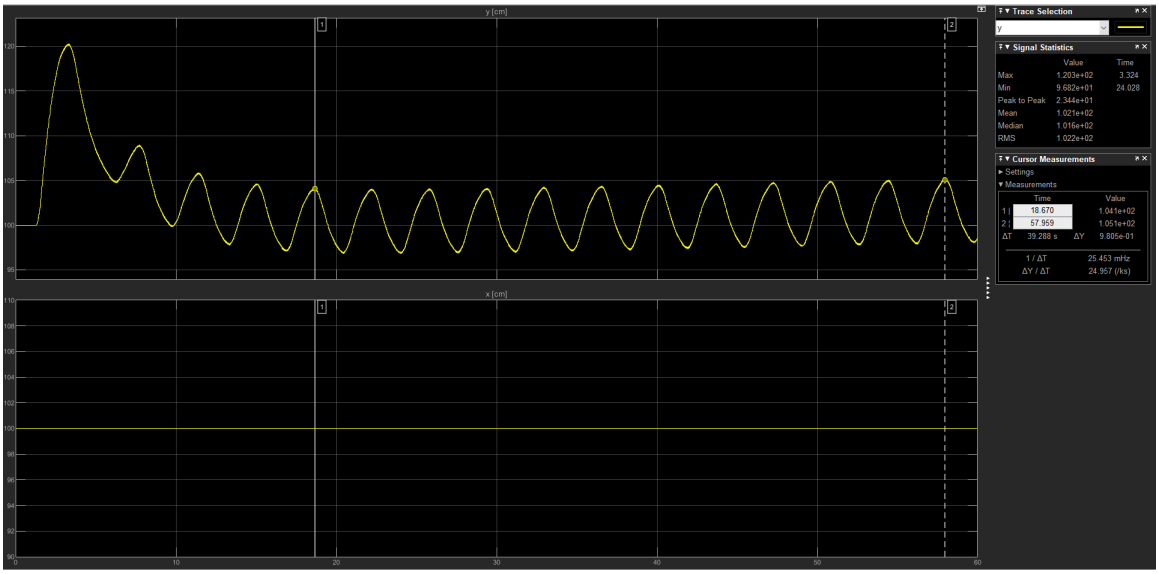
ANNEXE 1. Code MATLAB pour la commande LQ



ANNEXE 2. Modèle Simulink pour la commande LQ

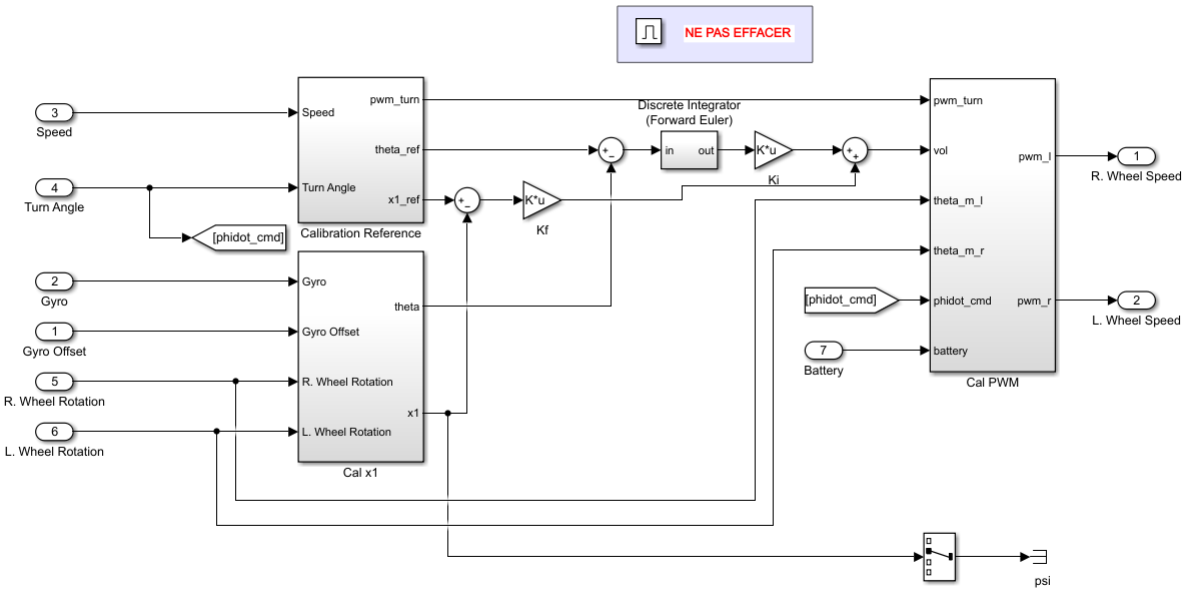


ANNEXE 3. Simulation - variation de ψ pour une commande LQ avec $q_2 = 10^3$



ANNEXE 4. Déviation selon l'axe y avec la commande LQ

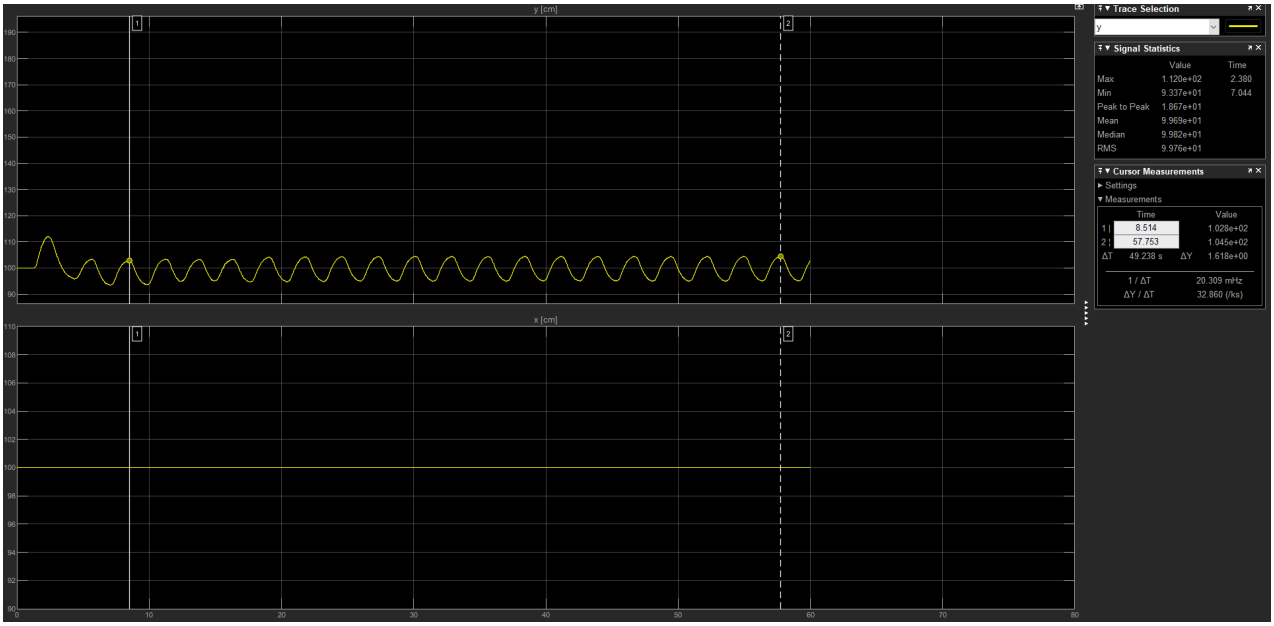
Action intégrale



ANNEXE 5. Modèle Simulink de la commande avec action intégrale

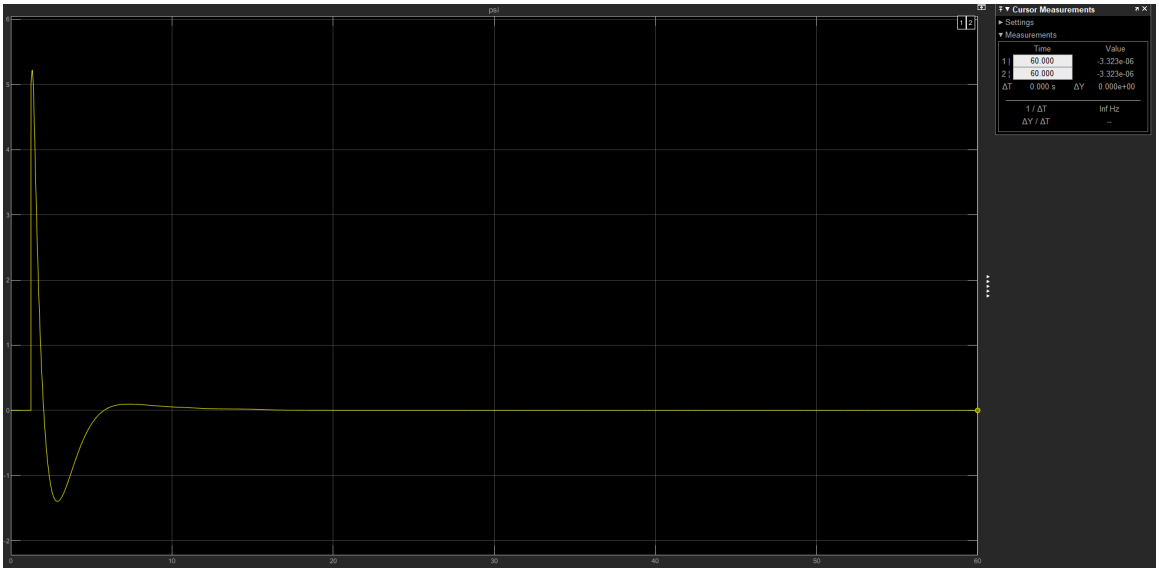
```
%on recrée des matrices "augmentées" càd on ajoute un paramètre q5 pour
%jouer sur l'intégration

q5=10e0;
C=[1 0 0 0];
Z_col=zeros(4,1);
Z_lig=zeros(1,2);

A1i = [A1 Z_col
       C 0];

B1i = [B1
    Z_lig];

 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 = [r1 0
    0 r2];

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

ANNEXE 6. Code MATLAB pour l'augmentation de la matrice QQ

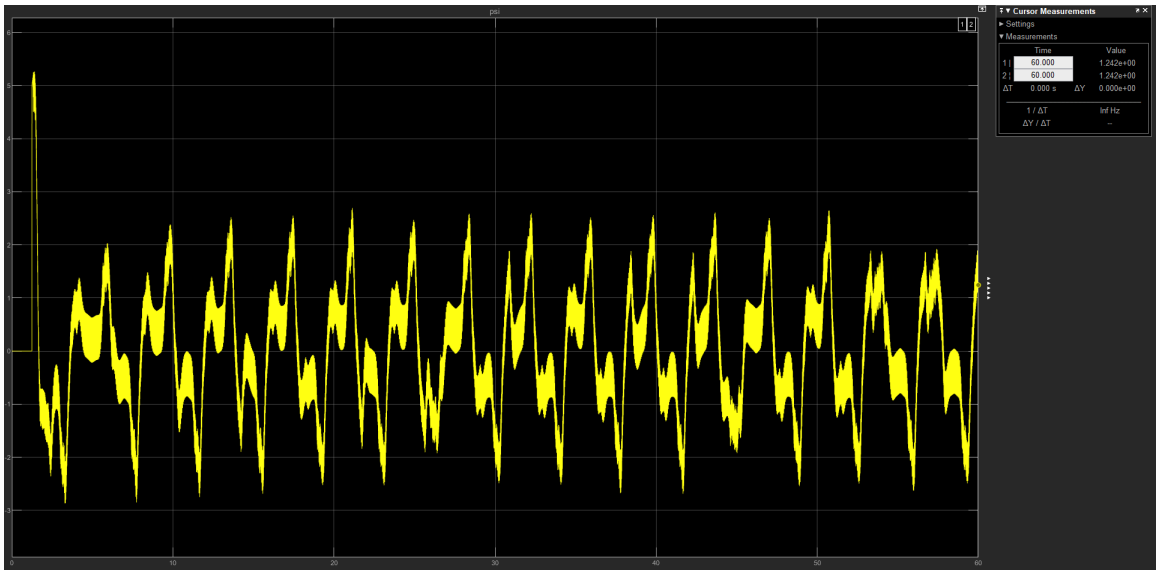


ANNEXE 7. Variations de ψ pour une action intégrale avec $q_5 = 1$

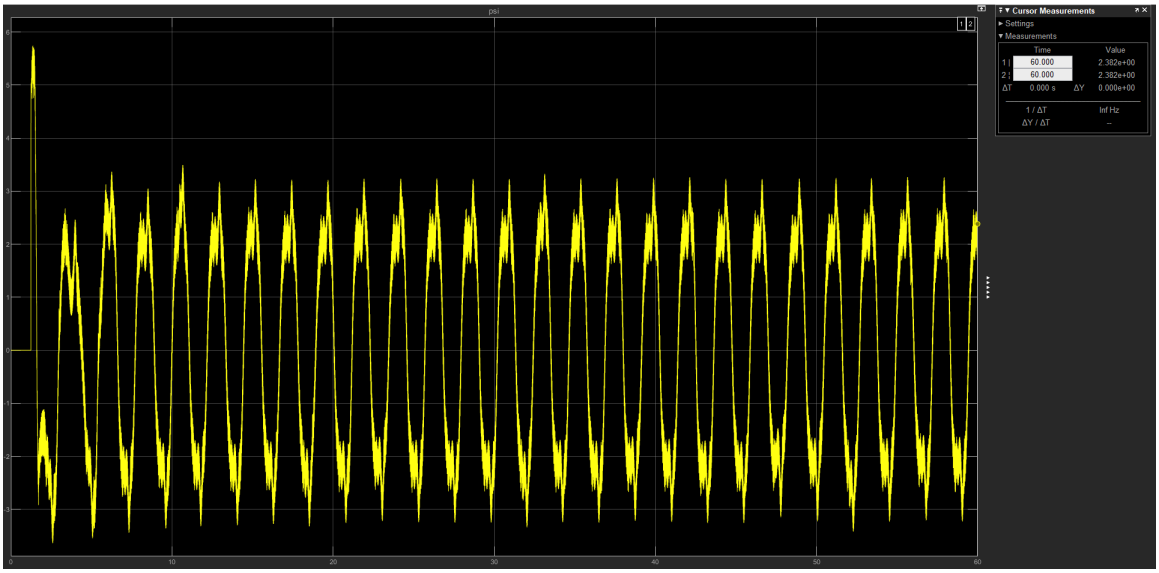
Placement de pôles



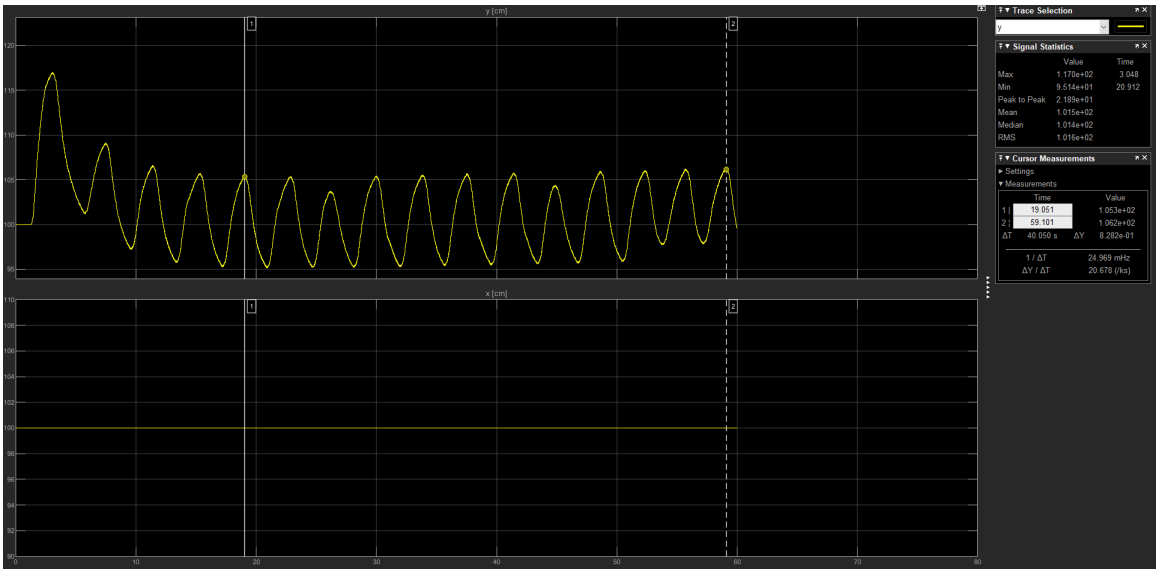
ANNEXE 8. Placement des pôles - Simu 1



ANNEXE 9. Placement des pôles - Simu 2

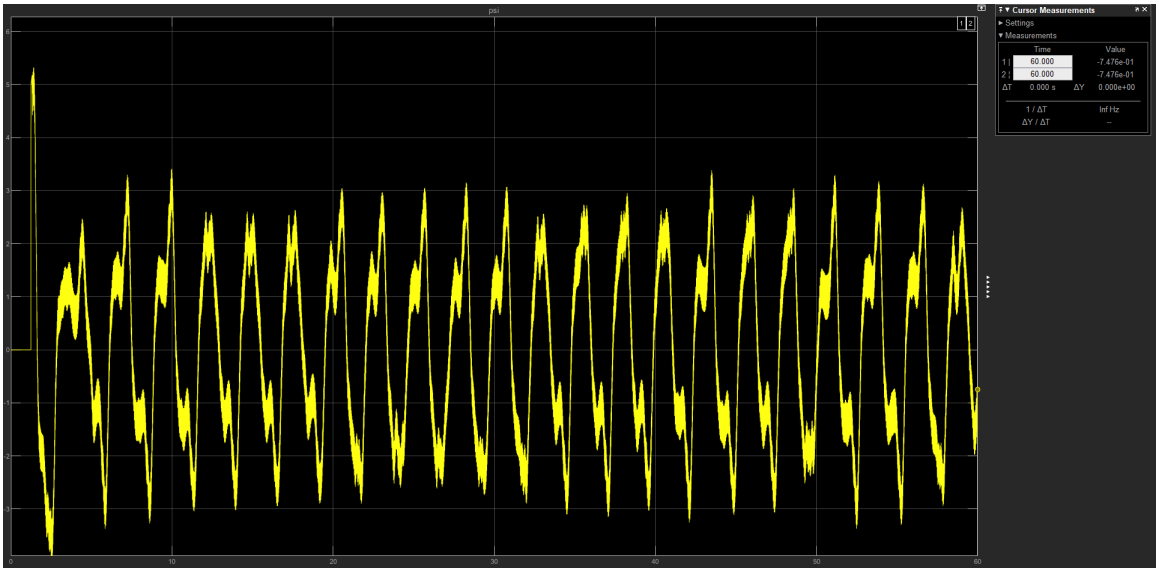


ANNEXE 10. Placement des pôles - Simu 3

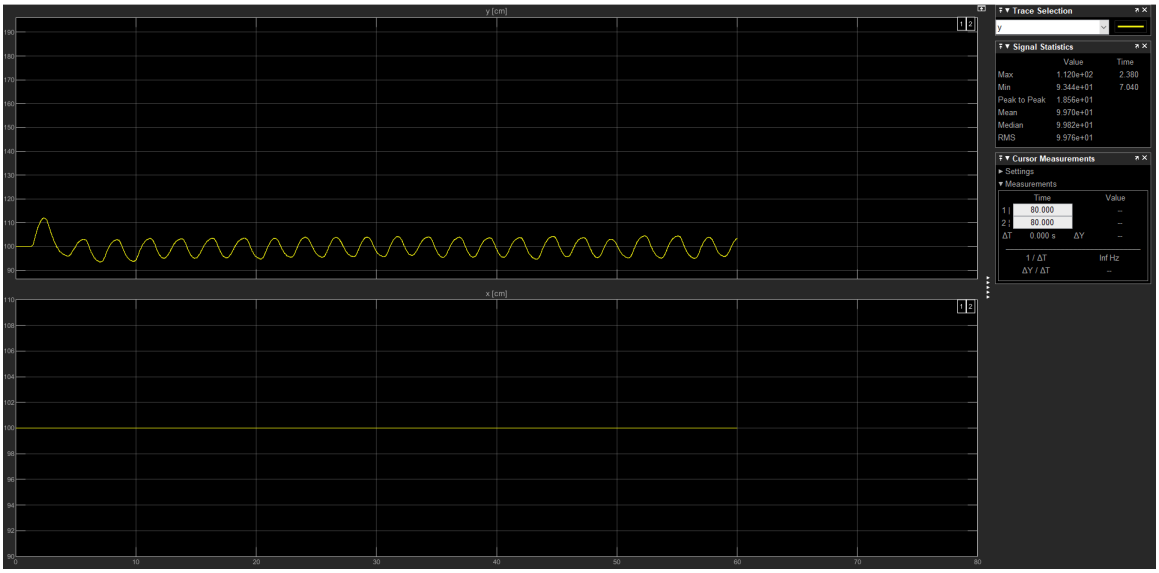


ANNEXE 11. Déviation en y pour une commande à placement de pôles (avec valeurs considérées optimales)

Placement de pôles et action intégrale



ANNEXE 12. Simulation de commande avec placement de pôles (inchangés) et action intégrale



ANNEXE 13. Déviation en y pour une commande à placement de pôles et action intégrale