Travaux Pratiques 1
Groupe 2- Equipe B01

Commande des Systèmes Dynamiques

Ana Luísa Fogaça Nogueira
Pedro Sacramento Xavier Barreto Rosa
Victor Freitas Teodoro

Metz
CentraleSupélec

CentraleSupélec
Octobre 14, 2022

Table de Matières

1 Objectifs

Le but de ce travail pratique est de mètre en ouvre des asservissements pour commander un dispositif réel (figure 1) à différents cahiers des charges. Les calculs et modélisations ont été effectués dans le cadre des concepts vus aux cours de Commande des Systèmes Dynamiques, et les simulations ont été obtenues à travers de l'extension SIMULINK de MATLAB.

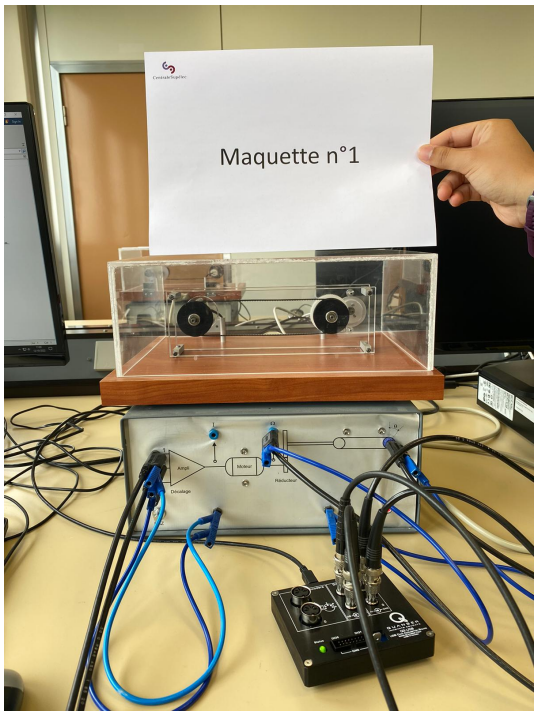

Figure 1: maquette n°1.

2 Démarches et résultats

2.1 Identification

Le modèle choisit pour étudier le système réel est régi fondamentalement par 5 paramètres, inconnus dans un premier moment. Ils sont : Kv (constante de vitesse), τ_m (constante de temps mécanique),τ_e (constante de temps électrique), Kt (rapport entre $V_\theta(t)$ et $\theta(t)$ et g (rapport entre $V_\omega(t)$ et $\omega(t)$). Sans connaître les valeurs de ces paramètres, on ne peut ni faire des simulations ni créer les asservissements nécessaires. On utilise alors des valeurs standards, fournis dans le sujet, et exécute une comparaison entre la simulation du modèle et le système réel. Un algorithme qui minimise la différence entre les deux réponses en variant les valeurs des paramètres va fournir des approximations pour ces dernièrs (figure 2). Les valeurs standards et les valeurs fournies par l'algorithme sont trouvées dans le Tableau 1.

	Valeur Standard	Valeur Estimée
$Kv(rd/s/V)$	35	45,6
$\tau_m(15ms)$	15	15,6
$\tau_e(3ms)$	3	2,8
$Kt(V/rd)$	0,5	0,5
$g(V/rd/s)$	0,001	0,009

Table 1: Paramètres du système

.

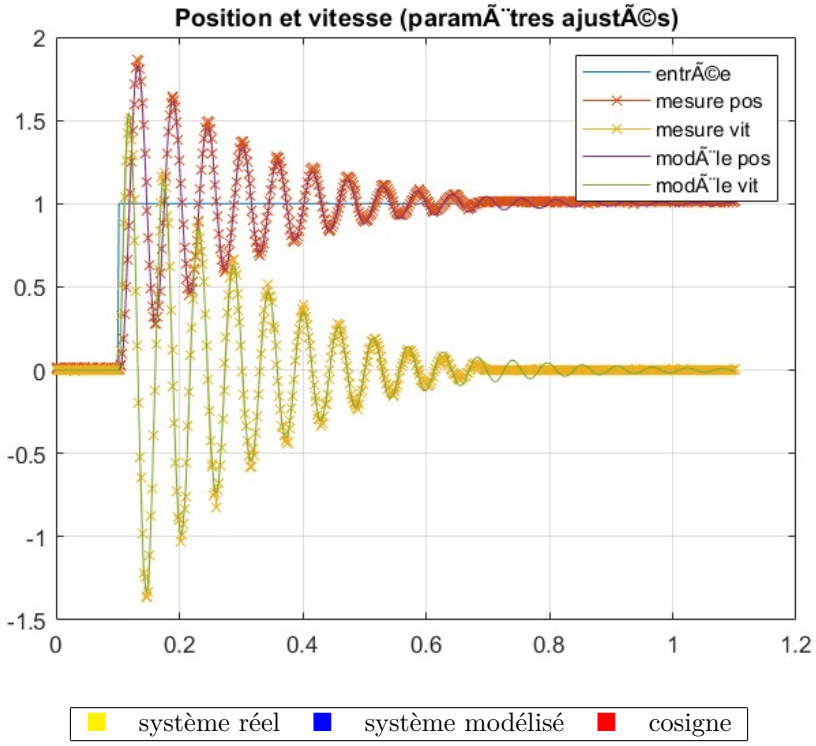

Figure 2: Consigne et Mesure de Position sans décalage.

2.2 Correction série

Pour le système en boucle ouverte, on vérifie qu'il y a une réponse indésirable du système lors d'un changement de la consigne, soit pour un déplacement trop important, soit pour un temps de montée trop grand. Pour cette raison, l'ajout d'un correcteur continu au système se fait nécessaire pour respecter le cahier des charges imposé par l'énoncé. Pour mettre en place un tel correcteur, la pulsation propre ω_c du système a été calculé à partir de l'équation 1 extraite de l'analyse de l'abaque illustré dans la figure 3.

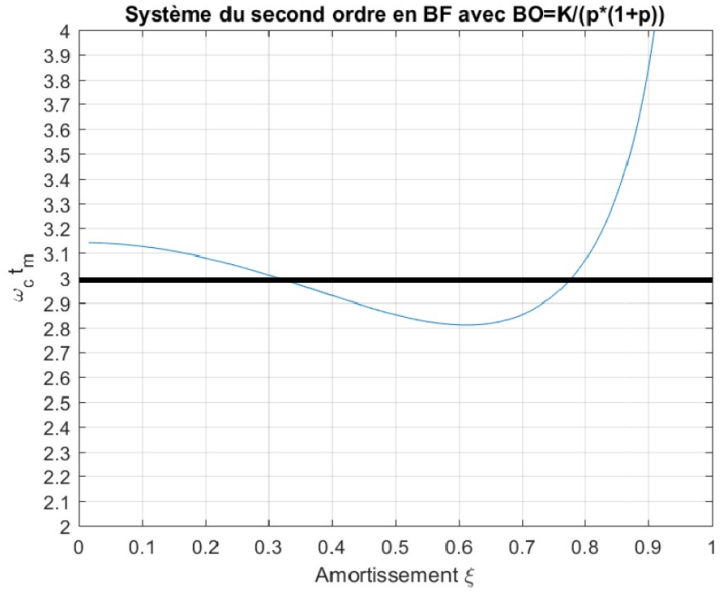

Figure 3: Abaque avec la relation du Amortissement et $\omega_c t_m$.

$$\omega_c t_m = 3 \tag{1}$$

Comme, d'après le cahier des charges, $t_m = 40ms$, on obtient $\omega_c = 75rad/s$. En plus, avec la valeur de déplacement désiré de 15%, on calcule la marge de phase de $\phi = 55$ à partir de l'abaque contenue dans la figure 4 .

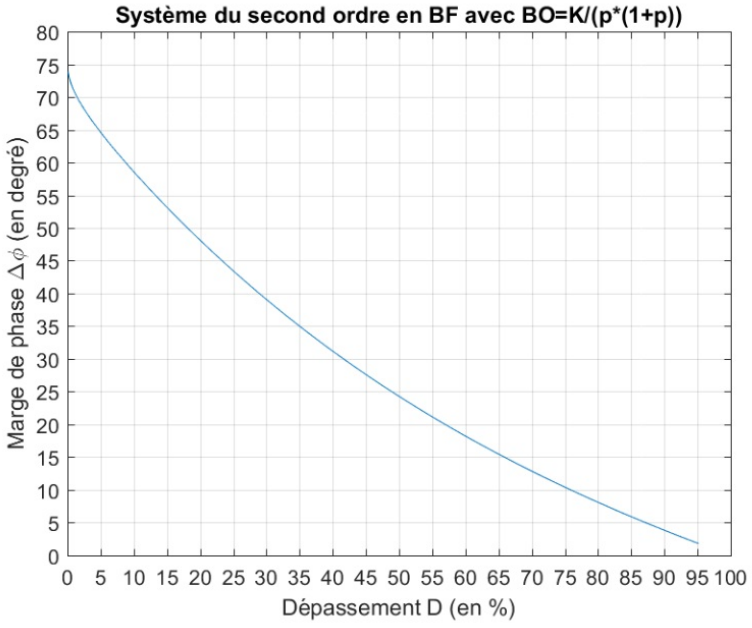

Figure 4: Abaque avec la relation du Dépassement D et la Marge de phase $\Delta\phi$.

Ensuite, la démarche pour créer un correcteur à avance de phase, comme indiqué dans

l'annexe A-3 de la polycopie du cours, a été faite sur MATLAB. Le gain $G = 0.1929$ et le $\phi = -151.6315$ ont été trouvés à partir de la fonction "bode" (indiqué sur l'item 6.6.4 du sujet). La forme analytique a aussi été déterminé à partir des données trouvées sur MATLAB.

Pour pouvoir étudier sur SIMULINK l'action du correcteur, on discrétise ce dernier à travers de la formule `c2d`, indiquée dans la section 6 du sujet. Ça permet la détermination les matrices numC e denC :

$$numC = \begin{bmatrix} 7.9673 & -7.2754 \end{bmatrix} \tag{2}$$

$$denC = \begin{bmatrix} 1 & -0.7795 \end{bmatrix} \tag{3}$$

La figure 5 montre la simulation du système sous influence du correcteur continu modélisé. On observe que le dépassement est bien autour de 15%, le temps de monté est d'environ 40ms et que les système converge vers la consigne.

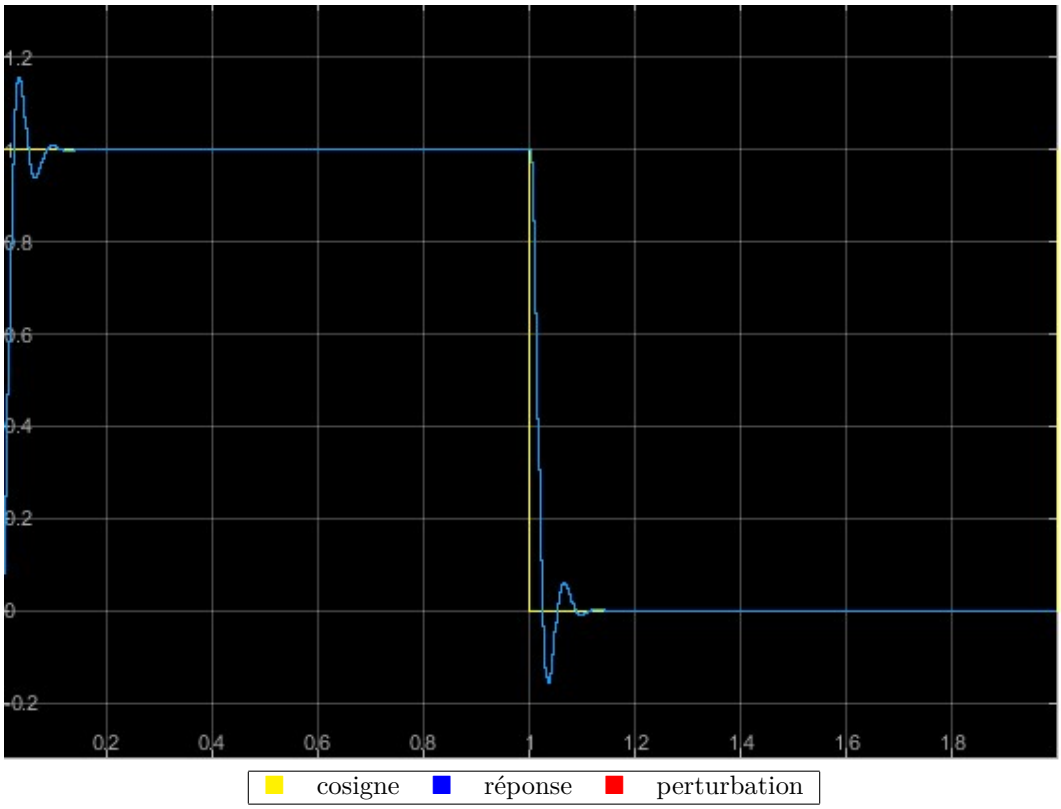

Figure 5: Réponse du système avec le correcteur continu.

On valide la simulation en comparant la réponse du système modélisé avec celle du système réel. La figure 6 montre une qu'il il a une similarité entre les formats des deux courbes, mais avec un certain écart. Cette différence est due au décalage du système réel, qui n'a pas été modélisé.

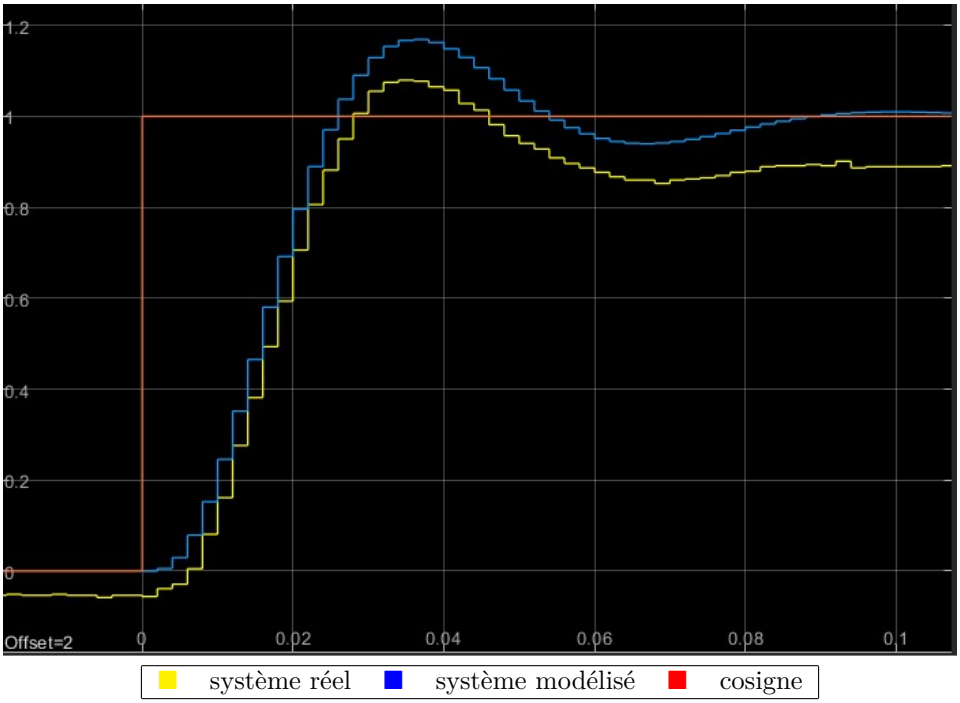

Figure 6: Consigne et Mesure de Position avec décalage.

Après réaliser des ajustements sur le système réel pour supprimer l'effet du décalage, on voit que l'écart n'existe plus et les courbes sont très similaires, comme montré dans la figure 7. Dans le prochaines comparaisons entre le modèle et le système réel effectués dans cette étude, l'effet du décalage n'apparaîtra plus, due aux ajustements réalisés sur la maquette au préalable.

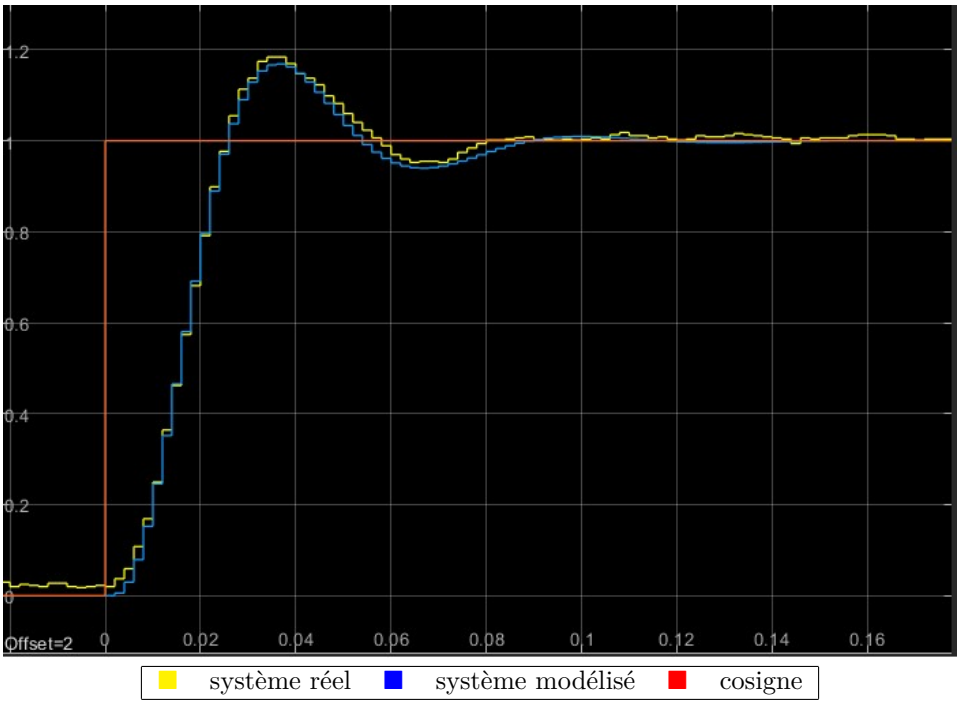

Figure 7: Consigne et Mesure de Position sans décalage.

2.3 Commande modale

On commence par les fonctions de transfert :

$$\frac{\Omega(p)}{U(p)} = \frac{K_V}{(1+\tau_m p)} \iff \tag{4}$$

$$K_v U(p) = \Omega(p) + \tau_m \Omega(p) p \tag{5}$$

Et grâce à l'utilisation de la transformée inverse de Laplace :

$$\dot{\omega}(t) = -\frac{\omega(t)}{\tau_m} + \frac{K_v u(t)}{\tau_m} \tag{6}$$

Comme :

$$\dot{\theta}(t) = \omega(t) \tag{7}$$

$$\theta(t) = \frac{V_\theta(t)}{K_t} \tag{8}$$

$$\omega(t) = \frac{V_\omega(t)}{g} \tag{9}$$

Il est possible d'écrire :

$$\dot{x}(t) = \begin{bmatrix} 0 & \frac{K_t}{g} \\ 0 & -\frac{1}{\tau_m} \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ g\frac{K_v}{\tau_m} \end{bmatrix} u(t) \tag{10}$$

Où :

$$x(t) = \begin{bmatrix} V_\theta(t) \\ V_\omega(t) \end{bmatrix} \tag{11}$$

On peut déterminer la commandabilité du système à travers du critère de Kalman. On voit que la matrice $\mathscr{C}$ est de rang plein, donc le système est totalement commandable.

$$\mathscr{C} = \begin{pmatrix} B & AB \end{pmatrix} = \begin{pmatrix} 0 & \frac{K_t K_v}{\tau_m^2} \\ g\frac{K_v}{\tau_m} & -g\frac{K_v}{\tau_m^2} \end{pmatrix} \tag{12}$$

Ensuite, il faut trouver les équations numériques de l'espace d'état. Pour cet objectif, on utilise la fonction c2d de MATLAB, ainsi que et les valeurs présentes dans l'énoncé. Les matrices résultantes sont trouvées en [13].

$$F = \begin{bmatrix} 1 & 0,0936 \\ 0 & 0,8752 \end{bmatrix} \quad G = \begin{bmatrix} 0,022 \\ 0,0437 \end{bmatrix} \tag{13}$$

Pour trouver l'emplacement souhaité des pôles, le bout de code suivant a été utilisé :

```
% second-order damping ratio and natural frequency
ksi = abs(log(D))/sqrt(log(D)^2 + pi^2);
w0 = pi/(tm * sqrt(1-ksi^2));

% desired poles
pk = roots([1 2*ksi*w0 w0^2]); % continuous
zk = exp(pk*T);                % discrete
```

Où `tm` est le temps de premier maximum et `D` est le dépassement. Ça donne:

$$\begin{aligned} p_1 &= -63,24 + 104,72i \quad & p_2 &= -63,24 - 104,72i \\ z_1 &= 0,86 + 0,18i & z_2 &= 0,86 - 0,18i \end{aligned} \tag{14}$$

Ensuite, il faut trouver les gains de la matrice L appropriés. Ça se fait en utilisant la fonction `place` du MATLAB. On vérifie aussi le comportement du système soumit a un échelon, figure 8.

```
% pole placement
L = place(etatD.A, etatD.B, zk)

% closed loop discrete system
cloopD = L(1) * feedback(etatD,L);
step(cloopD)
```

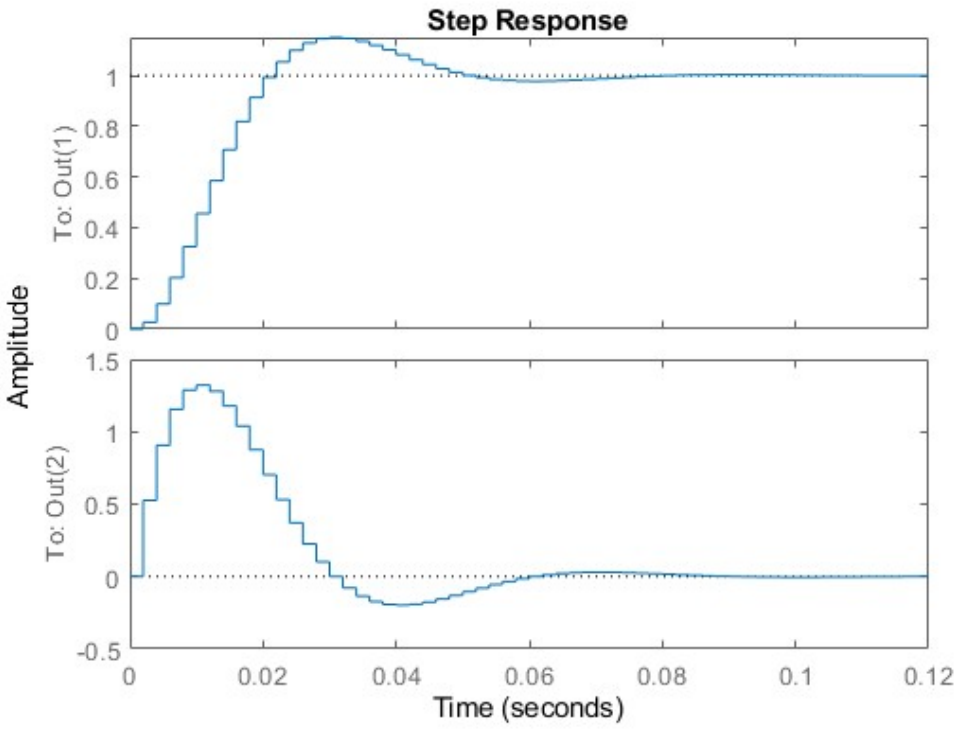

Figure 8: Réponse à échelon.

$$L = \begin{bmatrix} 12,0457 & 2,8473 \end{bmatrix} \tag{15}$$

On souligne que dans cette modélisation l'ordre des variables d'état est V_θ suivi par V_ω et pour cette raison la séquence des valeurs dans la matrice L peut changer.

la figure 9 montre la comparaison entre le comportement de la simulation et celui du système réel. Une approximation entre les deux courbes est constatée, même s'ils existent certaines perturbations qui n'ont pas été modélisées (frottements, imperfections dans les surfaces, etc) et qui originent des bruits.

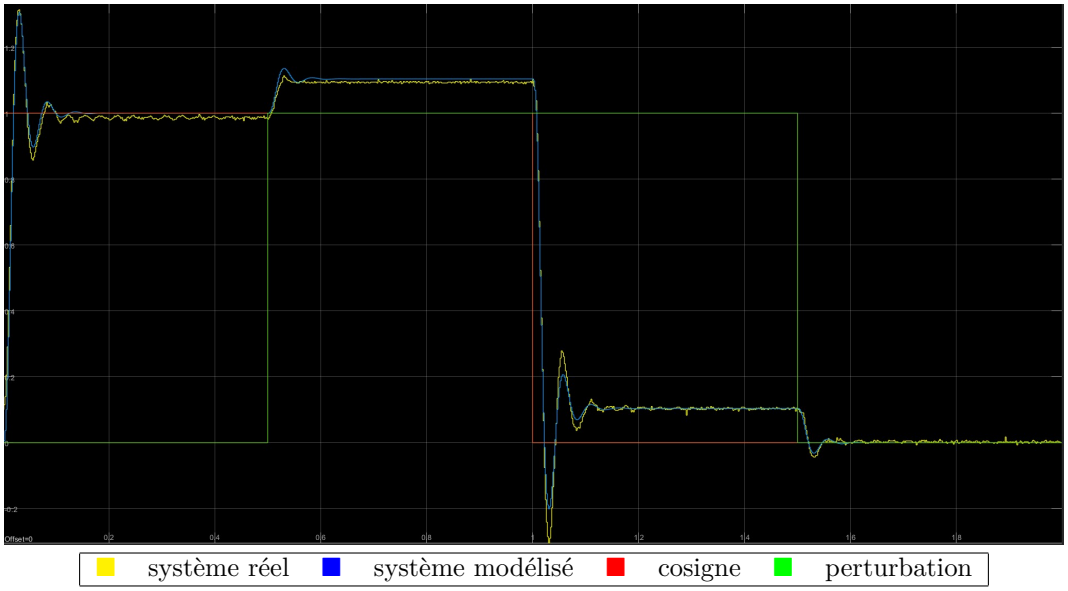

Figure 9: Consigne et Mesure de Position sans décalage.

2.4 Annulation de l'erreur statique avec action intégrale

Visant annuler l'erreur statique, on ajoute au système un intégrateur dont la sortie est donnée par $\dot{z} = \int_0^t (y_{ref} - y)dt$ et la fonction de transfert est donnée par $\frac{1}{1-z^{-1}}$. Ceci se fait en insérant la variable z au vecteur d'état, de façon à obtenir une représentation d'état étendue (16 e 17) dont le gain K, maintenant composé par 3 pôles, permettra l'élimination de l'erreur statistique.

$$\dot{x}(t) = \begin{bmatrix} 1.0000 & 0.0938 & 0 \\ 0 & 0.8750 & 0 \\ 1.0000 & 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ z(t) \end{bmatrix} + \begin{bmatrix} 0.0022 \\ 0.0438 \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} y_{ref} \tag{16}$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ z(t) \end{bmatrix} \tag{17}$$

Pour la commandabilité du système, on montre que :

$$det \begin{bmatrix} B_a & B_a A_a & B_a A_a^2 \end{bmatrix} = det \begin{bmatrix} 0.0027 & 0.0079 & 0.0124 \\ 0.0477 & 0.0420 & 0.0369 \\ 0 & 0.0027 & 0.0106 \end{bmatrix} \neq 0 \tag{18}$$

Donc, la matrice est de rang plein est le système est totalement commandable.

Avec le but trouver le nouvel gain K, on fait d'abord intervenir l'équation 19 pour trouver l'amortissement ξ, en respectant le cahier des charges fournit $t_m = 50ms$ et $D = 15\%$, ce que résulte en $\xi = 0{,}52$. Avec cette valeur, l'équation 20 est utilisé pour trouver la pulsation propre $\omega_0 = 73{,}4$.

$$\xi = \frac{|ln(D)|}{\sqrt{ln(D)^2 + \pi^2}} \tag{19}$$

$$\omega_0 = \frac{\pi}{t_m \sqrt{1 - \xi^2}} \tag{20}$$

$$p \cong \lambda^2 + 2\xi\omega_0 + \omega_0^2 \tag{21}$$

En suite, on fait une approximation du modèle par un système de deuxième ordre, dont le polynôme caractéristique est définit par l'équation 21. Les racines du polynôme sont calculés et le troisième pôle est obtenu en multipliant par un facteur 5 la plus négative partie réelle parmi les racines. Le résultat est donné par l'équation 22. L'équation 23 donne les valeurs de z_k, qui est le vecteur p_k discrétise pour un traitement numérique.

$$p_k = \begin{bmatrix} -0.3794 + 0.6283i \\ -0.3794 - 0.6283i \\ -1.8971 \end{bmatrix} \tag{22}$$

$$z_k = e^{p_k T_e} = \begin{bmatrix} 0.9196 + 0.1162i \\ 0.9196 - 0.1162i \\ 0.6843 \end{bmatrix} \tag{23}$$

Finalement, en utilisant la fonction *place* au MATLAB avec les matrices A_a, Ba et le vecteur z_k comme arguments, on trouve :

$$L = \begin{bmatrix} 12.71047.03131.2004 \end{bmatrix} \tag{24}$$

Les figures 10 et 11, illustrent la simulation asserv_etat_etendu_sim.slx obtenue sur MATLAB en utilisant le vecteur L trouvé dans le fichier asserv_etat_etendu_ini. Il est possible d'observer dans la figure 10 que l'erreur statistique n'existe plus, même avec l'intervention de la perturbation, et le modèle semble présenter le comportement désiré. Le seul problème avec l'action intégrale est la saturation de la commande u, qui atteint la limite de -10V pendant la simulation, comme constaté dans la figure 11.

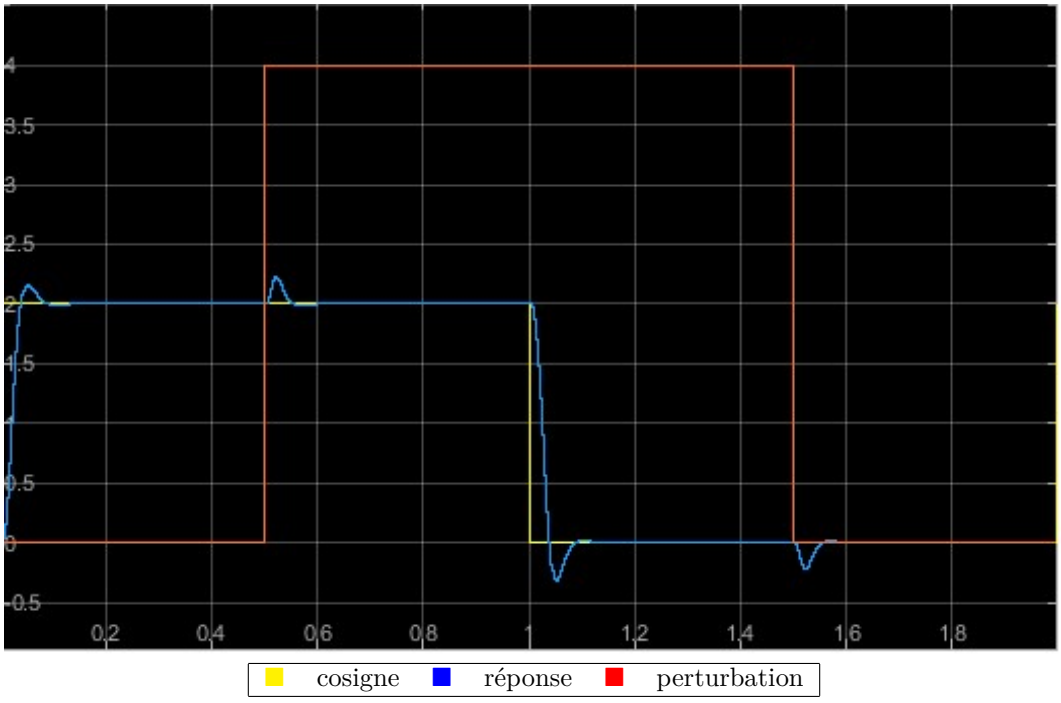

Figure 10: Réponse du système avec action intégrale.

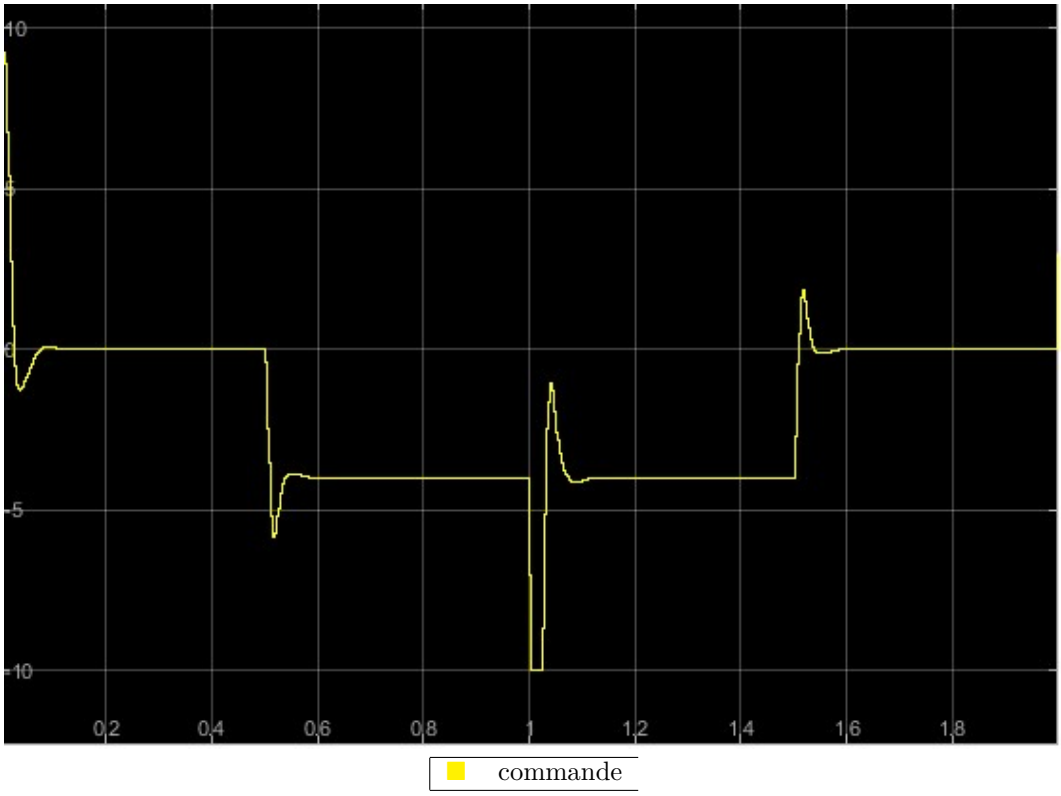

Figure 11: Intensité de la commande u avec action intégrale.

Finalement, on observe sur la figure 12 que la réponse du système simulé et celle présente pour le système réel suivent presque la même trajectoire. Même après les changements de consigne et de l'action de la perturbation, les courbes sont très similaires. L'expérience valide donc la mise en ouvre des asservissements demandés dans cette section.

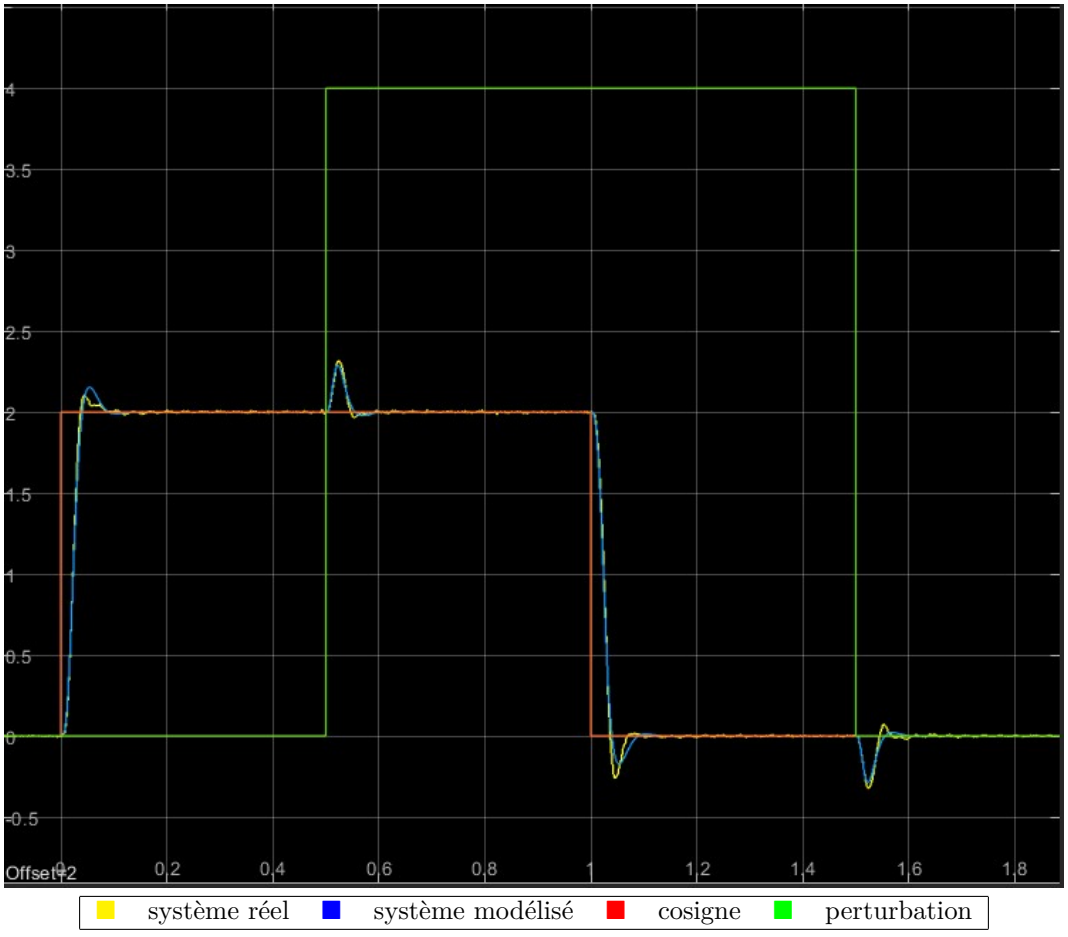

Figure 12: Validation par comparation du système modélise e le système réel.

2.5 Commande linéaire quadratique avec action intégrale

Bien que l'action intégrale construite dans la section précédente ait corrigé l'erreur statique, son implémentation a créé le problème de saturation des commandes, ce qui empêche son utilisation dans un système réel. Pour résoudre ce souci, on ajoute une commande Linéaire Quadratique à l'action intégrale, utilisant la fonction de coût suivante :

$$J = \sum_{n=0}^{\infty} q_1 V_\omega(nT_e)^2 + q_2(V_\theta(nT_e) - V_\theta(\infty))^2 + q_3(V_q(nT_e) - V_q(\infty))^2 + u(nT_e)^2$$

Les conditions d'existence d'une solution de J sont telles que (Aa, Ba) soit stabilisable, fait qui a été démontré dans la section 5.4, et que (Aa, Ha) soit détectable, où $Q_a = H_a^T H_a$. En posant $H_a = [q_1\ q_2\ q_3]$, on trouve :

$$det(Qo) = det\begin{bmatrix} H \\ HA \\ HA^2 \end{bmatrix} = q_3^2(\frac{q_2}{64} - \frac{469q_1}{4000} + \frac{469q_3}{5000})$$

Pour que Q_o soit de rang plein, c'est-à-dire $det(Qo) \neq 0$, if faut juste que $q_3 \neq 0$. Avec cette condition attendue, on montre que Q_o est observable, donc détectable, et on peut arbitrer les valeurs de q_1 et q_2 de façon libre.

Une fois les composants q_1, q_2 et q_3 du vecteur H définis, il suffit d'utiliser la fonction *dlqr* de MATLAB pour trouver le vecteur de gains L, ainsi que de s'assurer que la valeur

Itération	q_1 et q_2	q_3	L	Max |u|
1	0	2	[6.3085 16.9278 1.6798]	>10
2	0	1	[4.8168 11.0349 0.8778	=10
3	0	0,85	[4.5115 9.9812 0.7527]	9.7

Table 2: Gain L trouvé avec LQ

.

maximale absolue de u ne dépasse pas l'intervalle limite imposé [-10 V, 10 V]. Si cette valeur dépasse la limite, q_1, q_2, et q_3 sont redéfinies et le processus est refait de manière itérative. On trouve dans le tableau 2.5 les résultats des itérations réalisées pour le système modélise.

La figure 13 montre la simulation asserv_etat_etendu_sim.slx obtenue au MATLAB en utilisant L = [4.5115 9.9812 0.7527] dans le fichier asserv_etat_etendu_ini. On constate que les caractéristiques de la réponse, dues aux asservissements réalisés dans les sections précédentes, sont maintenues. La grande différence est trouvée dans la figure 14, où il est possible d'observer que les commandes du système ne saturent plus, c'est-à-dire |u| < 10. Ça signifie, donc, qu'on a abouti à un système qui respecte les contraintes physiques imposées.

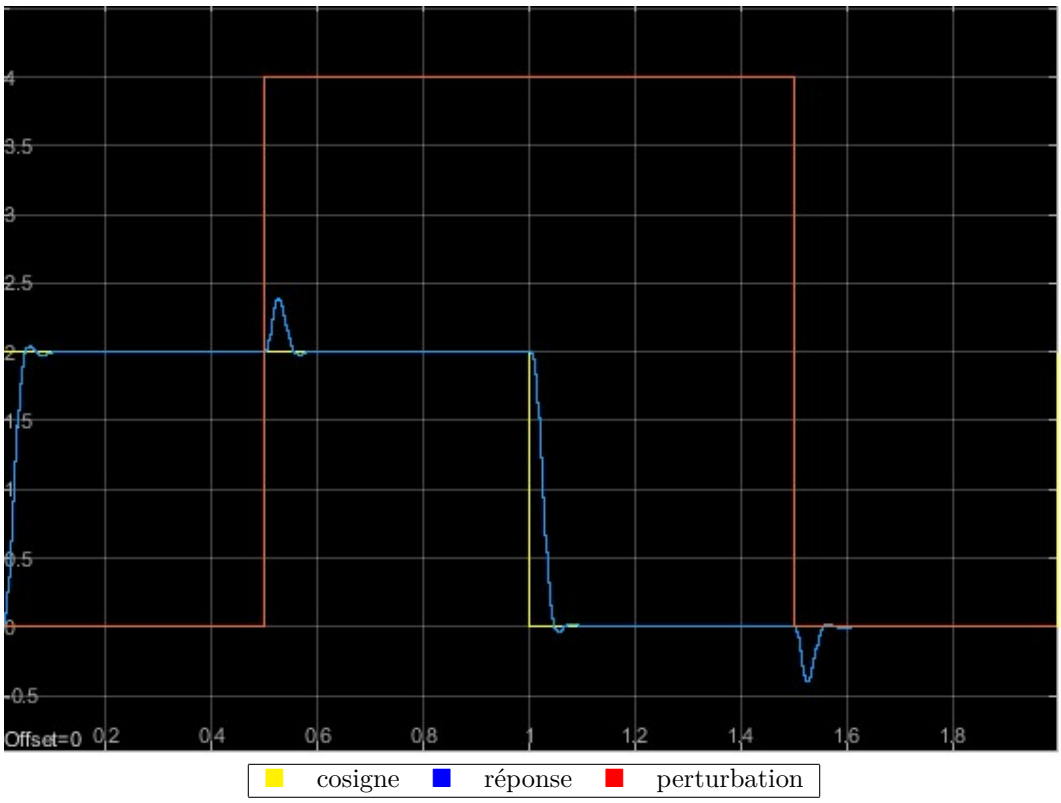

Figure 13: Réponse du système avec commande LQ.

Figure 14: Intensité de commande u avec commande LQ.

Ultimement, comme cela s'est passé lors de la validation du modèle réel de la figure 12, non seulement une trajectoire similaire a été perçue entre le comportement réel et la simulation, mais encore une réponse très proche des changements de perturbation et de consigne, comme le montre la figure 15. S'appliquant donc de manière satisfaisante à la réalité.

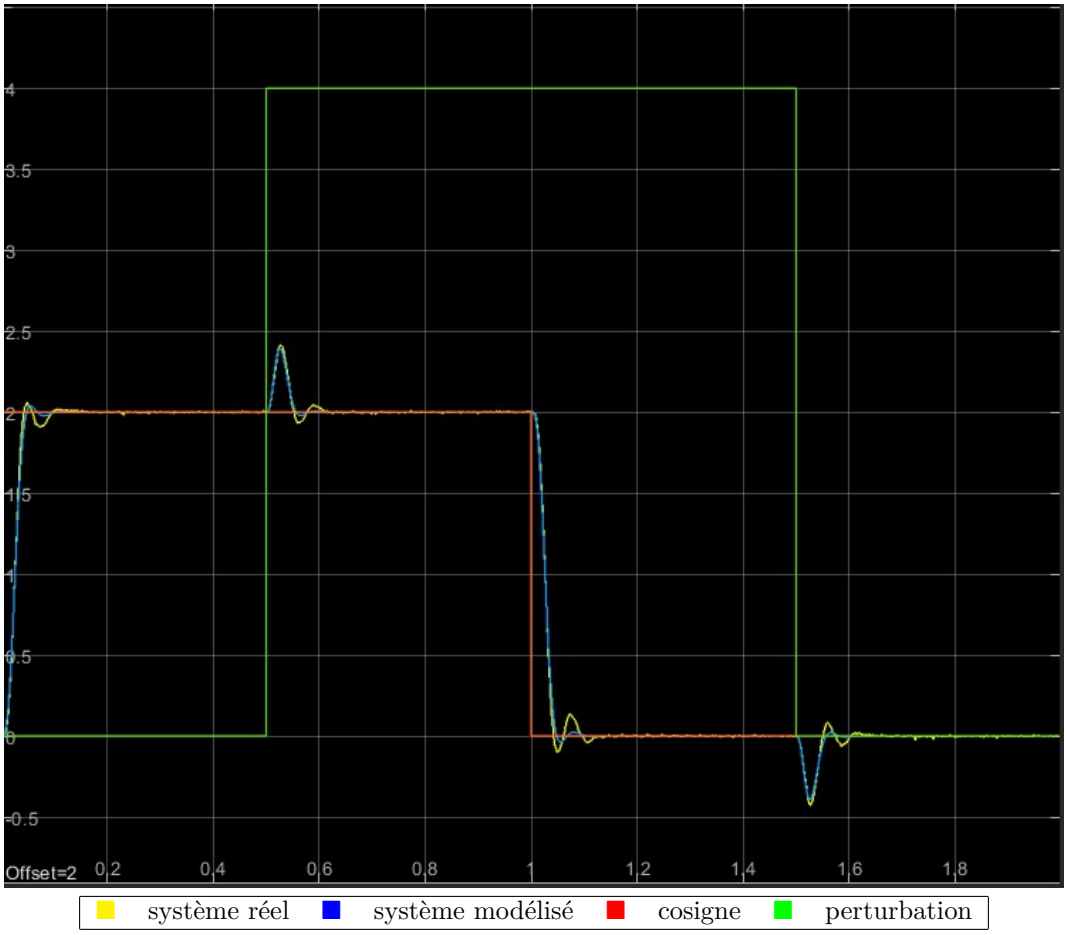

Figure 15: Validation par comparation du système modélise e le système réel.

3 Conclusion

Après l'exécution du travail, on peut voir comment les étapes et les commandes sont étroitement liées. Dès le début, lorsqu'une réponse indésirable du système a été observée et qu'un correcteur continu a été créé pour la corriger ; suivi par les équations d'état du modèle numérique ; suivi par la perception d'une erreur statique qui a été résolue, mais a entraîné une saturation des commandes qui empêcherait son utilisation dans un système réel ; suivi finalement de l'exécution d'une commande linéaire quadratique qui a conduit à un résultat satisfaisant, on peut voir comment les étapes sont essentielles pour une réponse affirmative. En outre, on constate que toutes les comparaisons entre les commandes réelles et les commandes de simulation sont satisfaisantes, ce qui montre comment notre système modélisé s'applique dans la réalité et conclut notre objectif.