UNIVERSITÉ TOULOUSE III PAUL SABATIER

Université de Toulouse

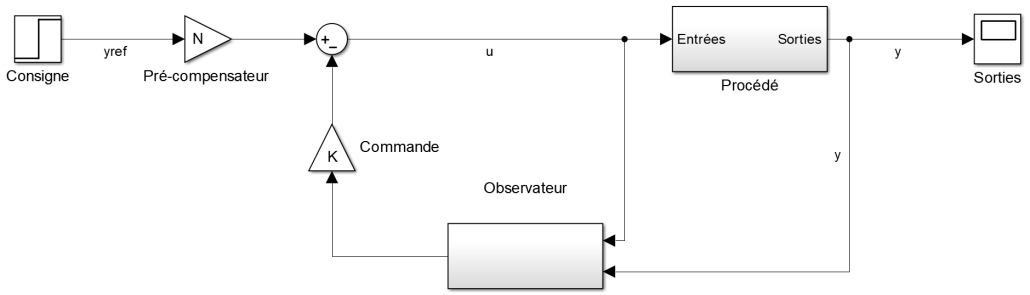
UNIVERSITÉ PAUL SABATIER

Conception et mise en œuvre de commande à temps réel

- Rapport 1 : ANALYSE THÉORIQUE -

Auteurs :
Lucien RAKOTOMALALA
David TOCAVEN

Encadrants :
Sylvain DUROLA
Frédéric GOUAISBAUT
Yann LABIT



Faculté des Sciences et d'Ingénierie

EEA TOULOUSE

14 décembre 2017

Table des matières

Introduction

Dans cet UE, nous devons réaliser la commande d'un système temps réel du prototypage à l'implémentation sur un microcontrôleur : nous devons asservir un banc de moteurs à courant continu à l'aide d'un micro-contrôleur C167.

Ce premier rapport contient toute la partie théorique. Celle-ci est décomposée dans les trois premiers chapitres et un dernier chapitre qui détaille la suite des étapes à réaliser.

Le *chapitre 1 : Modélisation* contient l'étude physique qui nous a été donné en cours, le modèle le plus précis, non linéaire et variant, les différentes simplifications de celui-ci.

Ensuite, dans le *chapitre 2 : Analyse*, nous avons effectué une analyse de nos différents modèles afin de maîtriser l'impact de nos simplifications, étudier les performances de notre systèmes et définir celles souhaitées. Nous avons réalisé cela grâce, autant que nous avons pu, a une approche théorique et grâce à des simulations.

Le *chapitre 3 : Synthèse de commande*, qui est le dernier chapitre de la partie théorique, contient la conception de l'asservissement et l'étude des performances de celle-ci sur les différents modèles de notre système.

Dans le *chapitre 4 : Planification de la suite de l'asservissement* , nous détaillons comment nous allons testé la validité de nos modèles par rapport au modèle physique et les différentes étapes de la mise en œuvre sur micro-contrôleur. Ce chapitre nous permettra d'organiser au mieux notre démarche afin que la commande implémentée respecte bien les contraintes temps réel, garantisse la stabilité et les performances attendues tout en étant adaptée au support d'implémentation et au moteur asservi.

Chapitre 1

Modélisation

1.1 Choix du formalisme et de la modélisation

Notre modélisation sera basée sur les modèles physiques qui décrivent les différents constituants de notre système de procédé : deux moteurs couplés l'un à l'autre par un arbre simple. L'un étant générateur de force mécanique et l'autre générateur de courant afin de faire office de charge (il dissipe son énergie sur une résistance). Il y a aussi un tachymètre couplé à l'arbre principal par un réducteur. Nos modèles seront donc des modèles de connaissances.

Nous avons choisis de faire une modélisation espace d'état pour différentes raisons. La première est que cette représentation permet d'étudier facilement la valeur des différents états (l'étude de la stabilité asymptotique, par exemple, en est simplifiée dans un modèle espace d'état). Elle permet aussi de garder les états non observables et non commandables dans le modèle, une modélisation par fonctions de transferts ne le permet pas. Ce choix nous permet aussi, pour la suite, de concevoir un asservissement par retour d'état basé observateur, qui est l'asservissement que nous avons choisis. Le choix d'un modèle de connaissance améliore aussi l'analyse de l'influence des différents paramètres du modèle, ce qui nous permettra d'affiner notre modèle lors des tests sur le système réel.

1.2 Maquette et équations physiques

1.2.1 Maquette :

Voici, figure 1.1, un schéma électrique et physique du banc qui fait office de procédé dans notre asservissement. Le circuit électrique de gauche correspond au Moteur à Courant Continue 1 (MCC) qui délivre la puissance

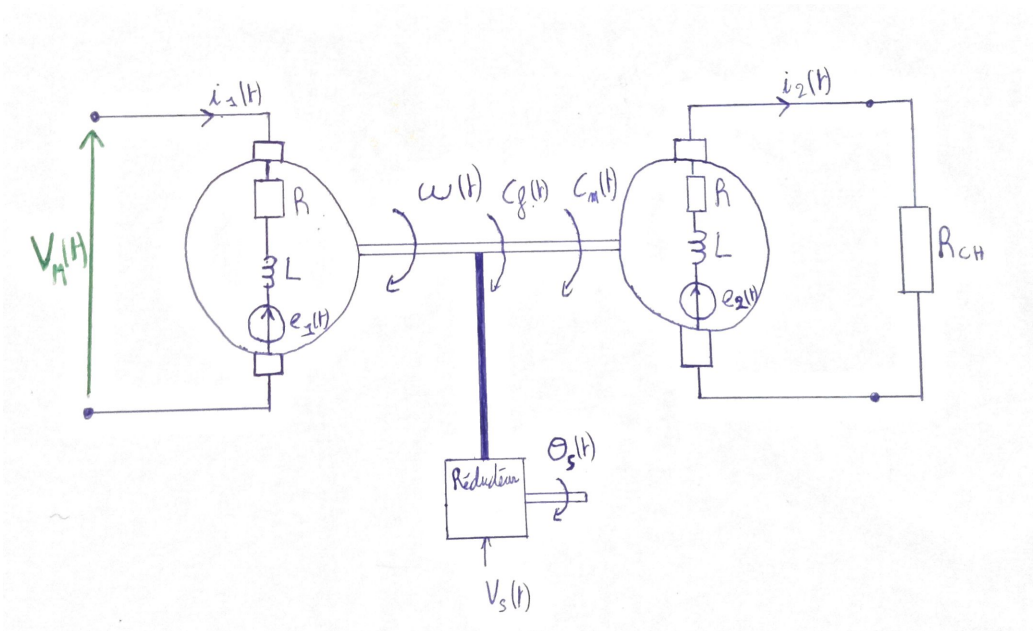


FIGURE 1.1 – Schéma électrique/physique du banc moteur

mécanique à partir d'une tension d'entrée V_M. Celle-ci est l'entrée de notre procédé. Le schéma électrique de droite représente le MCC 2, générateur de puissance électrique qui fait office de charge en alimentant une résistance R_{CH}. Entre ces deux schémas électriques, se trouve une représentation de l'arbre, des forces qu'il subit, du réducteur et du tachymètre qui délivre une tension proportionnelle à la position du moteur, c'est le signal V_s, nous l'étudierons en tant que sortie.

1.2.2 Équations physique

Voici les différentes équations décrivant notre procédé :

- Équations des moteurs :

$$V_m(t) = Ri_1(t) + L\frac{di_1(t)}{dt} + e_1(t) \tag{1.1}$$

$$e_2(t) = (R + R_{CH})i_2(t) + L\frac{di_2(t)}{dt} \tag{1.2}$$

- Équations banc :

$$J_2\frac{d\omega(t)}{dt} = C_m(t) + C_f(t) \tag{1.3}$$

$$\frac{d\theta_s(t)}{dt} = \omega(t) \tag{1.4}$$

$$e_1(t) = K_e\omega(t) \tag{1.5}$$

$$e_2(t) = K_e\omega(t) \tag{1.6}$$

$$C_m(t) = K_c i_1(t) - K_c i_2(t) \tag{1.7}$$

$$C_f(t) = -\mu\omega(t) - C_0(t) \tag{1.8}$$

$$R_{CH}(t) = R_{CHn} + rR_{CH}(t)\Delta_1 \tag{1.9}$$

$$C_0(t) = -sign(C_m(t)C \tag{1.10}$$

$$V_s(t) = K_r K_s \theta_s(t) \tag{1.11}$$

$$V_g(t) = K_g\omega(t) \tag{1.12}$$

Où :

R La résistance de l'induit aux moteurs.

L L'inductance de l'induite des moteurs.

$i_1(t), i_2(t)$ Respectivement le courant dans les moteurs 1 et 2.

$e_1(t), e_2(t)$ Respectivement la force électromotrice des moteurs 1 et 2.

$R_{CH}(t)$ Résistance de charge.

J_2 Inertie totale (somme des inerties du rotor, du réducteur et de la charge).

$\omega_s(t)$ Vitesse radiale de l'arbre.

θ_s Position de rotation de l'arbre.

$C_m(t)$ Couple de l'arbre.

$C_f(t)$ Couple de frottement.

K_e Constante de force électromagnétique.

K_c Constante de couple.

μ Coefficient de frottement visqueux.

$C_0(t)$ Couple de frottement sec si rotation.

R_{CHn} Résistance de charge nominale.

rR_{CH} Rayon de l'incertitude de la résistance de charge.

Δ_1 Perturbation de la résistance de charge bornée en $[-1; 1]$.

K_r Facteur de réduction.

K_s Constante du potentiomètre.

K_g Constante de la génératrice tachymétrique.

$V_g(t)$ Tension reflétant la vitesse de rotation de l'arbre. Sortie de performance non mesurée.

C Couple de frottement sec si $\omega(t)$ assez grande.

1.2.3 Linéarisation

Les équations ci-dessous permettent de former un modèle du procédé. Néanmoins, l'équation 1.10 exprime une non linéarité sur les frottements secs du banc moteur. Afin d'avoir un modèle linéaire du moteur, nous avons transformé cette non linéarité en incertitude. Ainsi nous exprimons :

$$C_0(t) = C_{0n} + rC_0\Delta_2 \tag{1.13}$$

Où :

C_{0n} Couple de frottement sec nominal.

rR_{CH} Rayon d'incertitude du couple de frottement sec.

Δ_1 Perturbation du couple de frottement sec bornée en $[-1;1]$.

Nous avons maintenant des équations linéaires mais deux équations présentent des incertitudes dues aux perturbations Δ_1 et Δ_2 respectivement liés à la température de la résistance de charge et à la non linéarité du couple de frottement sec.

1.2.4 Invariance

Nous souhaitons exprimer le modèle du procédé sous la forme d'un espace d'état linéaire et invariant, donc il faut rendre invariant nos équations. Pour cela, nous allons considérer que la résistance de charge $R_{CH}(t)$, exprimée dans l'équation 1.9 et $C_0(t)$, dans l'équation 1.13 valent leurs valeurs nominales.

$$R_{CH}(t) = R_{CHn} \tag{1.14}$$
$$C_0(t) = C_{0n} \tag{1.15}$$

Nous pouvons maintenant former un modèle linéaire et invariant.

1.3 Modèle de niveau 0, modèle EE0

Notre modèle présente 4 dynamiques, nous avons donc 4 états dans notre vecteur d'état.

Le vecteur d'état choisit est :

$$x(t) = \begin{bmatrix} i_1 \\ i_2 \\ \Omega_m \\ \Theta_m \end{bmatrix} \tag{1.16}$$

L'entrée $u(t)$ du modèle est $u(t) = V_m(t)$.

Les sorties sont $y(t) = \begin{bmatrix} V_g(t) \\ V_s(t) \end{bmatrix}$.

Le modèle d'ordre 4 vaut donc :

$$\begin{cases} \dot{x}(t) = \begin{pmatrix} -\frac{R}{L} & 0 & 0 & -\frac{K_e}{L} \\ 0 & -\frac{(R+R_{ch})}{L} & 0 & -\frac{K_e}{L} \\ 0 & 0 & 0 & 1 \\ \frac{K_c}{J_2} & \frac{K_c}{J_2} & 0 & \frac{\mu}{J_2} \end{pmatrix} x(t) + \begin{pmatrix} \frac{1}{L} \\ 0 \\ 0 \\ 0 \end{pmatrix} u(t) \\ y(t) = \begin{pmatrix} 0 & 0 & 0 & K_g \\ 0 & 0 & K_r K_s & 0 \end{pmatrix} x(t) \end{cases} \tag{1.17}$$

1.4 Espace d'état d'ordre 3, modèle EE1

Afin de facilité la création d'une commande, nous avons retiré l'état $i_2(t)$ de l'état de la modélisation. Il permettra une validation intermédiaire.

L'état vaut maintenant

$$x(t) = \begin{bmatrix} i_1 \\ \Omega_m \\ \Theta_m \end{bmatrix} \tag{1.18}$$

L'entrée et les sorties n'ont pas changées.

L'espace d'état, d'ordre 3, vaut :

$$\begin{cases} \dot{x}(t) = \begin{pmatrix} -\frac{R}{L} & 0 & -\frac{K_e}{L} \\ 0 & 0 & 1 \\ \frac{K_c}{J_2} & 0 & \frac{\mu}{J_2} \end{pmatrix} x(t) + \begin{pmatrix} \frac{1}{L} \\ 0 \\ 0 \end{pmatrix} u(t) \\ y(t) = \begin{pmatrix} 0 & 0 & K_g \\ 0 & K_r K_s & 0 \end{pmatrix} x(t) \end{cases} \tag{1.19}$$

1.5 Espace d'état d'ordre 2, modèle EE2

Pour correspondre avec un modèle de comportement, nous allons à nouveau réduire la représentation d'état (1.19) pour obtenir un modèle d'ordre 2. Pour cela, nous allons annuler l'effet des dynamiques des courants i_1 et i_2 qui sont beaucoup plus grandes que les dynamiques de ω et θ, qui sont celles que nous sommes capable de mesurer et que nous souhaitons asservir. Ainsi nous avons un espace d'état où il sera plus facile de concevoir un retour d'état. Le système d'ordre 3 servira donc de modèle permettant une validation intermédiaire entre celui d'ordre 2 et celui d'ordre 4.

Le vecteur d'état vaut maintenant :

$$x(t) = \begin{bmatrix} \Omega_m \\ \Theta_m \end{bmatrix} \tag{1.20}$$

Le modèle espace d'état d'ordre 2 vaut :

$$\begin{cases} \dot{X}(t) = \begin{pmatrix} 0 & 1 \\ 0 & -\frac{(K_c K_e)}{J_2 R} - \frac{\mu}{J_2} \end{pmatrix} X(t) + \begin{pmatrix} \frac{K_c}{J_2 R} \\ 0 \end{pmatrix} V_m(t) \\ Y(t) = \begin{pmatrix} 0 & K_g \\ K_r K_s & 0 \end{pmatrix} X(t) \end{cases} \tag{1.21}$$

Chapitre 2

Analyse

Dans ce chapitre, nous allons dans une première partie, étudier la stabilité de nos différentes modélisations (espace d'état d'ordre 4, 3 et 2), puis leurs commandabilités et observabilités. Dans une seconde partie, nous étudierons les performances dynamiques des différents modèles à travers une analyse temporelles et fréquentielle. Dans l'ensemble du chapitre sera abordé l'impact des simplifications effectués sur les modèles espace d'état d'ordre 3 et 2. Comme nous souhaitons asservir le procédé en vitesse et non en position, nous étudierons la sortie de performance $V_g(t)$ des modèles et non $V_s(t)$.

2.1 Analyse des modèles

2.1.1 Stabilité

Nous avons décidé d'étudier la stabilité asymptotique afin de savoir si l'ensemble des états de nos modèles sont stables et non uniquement ceux qui sont observables comme en stabilité BIBO.

Les valeurs propres de notre système d'ordre 4 sont : (calculé à l'aide de matlab)

$$0; -132749,8861; -4,0655; -7748,0483 \tag{2.1}$$

Nous remarquons que les valeurs propres sont toutes à partie réelle négatives, le système d'ordre 4 est donc asymptotiquement stable.
Les valeurs propres de notre système d'ordre 3 sont : (calculé à l'aide de matlab)

$$0; -7748,0484; -3,9516 \tag{2.2}$$

Nous remarquons que les valeurs propres sont toutes à parties réelles négatives, le système d'ordre 3 est donc asymptotiquement stable.

C'était prévisible car le modèle d'ordre 3 est une simplification du modèle d'ordre 4, qui est stable. Nous remarquons aussi que la troisième valeur propre du système d'ordre 3, qui normalement doit être similaire à la troisième valeur propre du système d'ordre 4 a légèrement variée. Cette différence est une première conséquence de la perte d'une dynamique engendrée par la simplification.
Les valeurs propres de notre système d'ordre 2 sont : (calculé à l'aide de matlab)

$$0; -3,9506 \tag{2.3}$$

Nous remarquons que les valeurs propres sont toutes à parties réelles négatives, le système d'ordre 2 est donc asymptotiquement stable.
Comme précédemment, cette conclusion était prévisible, néanmoins on remarque une autre conséquence de la simplification sur la seconde valeur propre qui est légèrement différente de celle du modèle d'ordre 3.

2.1.2 Commandabilité

L'étude de la commandabilité d'un système nous permettra de savoir quels états sont commandables, c'est à dire qu'il sera possible de modifier la dynamique qu'ils représentent par un asservissement. Cette étude se fera uniquement sur l'espace d'état d'ordre 4 car les deux autres modèles découlent de celui-ci. Un système (sous forme d'espace d'état) est commandable, d'après le critère de Kalman, si la matrice de commandabilité $\mathcal{C}_m$ est de rang plein donc égale à la dimension de A.

Où, $\mathcal{C}_m = \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix}$ Nous avons étudié la commandabilité sur matlab et le résultat est que :

$$rang(\mathcal{C}_m) = n = 4$$

Donc le système est commandable. Cela nous permet de mettre en place un retour d'état.

2.1.3 Observabilité

L'étude de l'observabilité d'un système nous permettra de savoir quels états sont observables, c'est à dire s'il est possible de déterminer la valeur des états à partir de mesures de la sortie. Cette étude se fera uniquement sur l'espace d'état d'ordre 4 car les deux autres modèles découlent de celui-ci. Un système (sous forme d'espace d'état) est observable, d'après le critère de Kalman, si la matrice d'observabilité $\mathcal{O}$ est de rang plein donc égale à la dimension de A.

Où, $\mathcal{O} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}$ Nous avons étudié l'observabilité sur matlab et le résultat est que :

$$rang(\mathcal{O}) = dim(A) = 4$$

Donc le système est observable, néanmoins intuitivement ce résultat semble faux.

2.2 Analyse temporelle et fréquentielle

Nous avons étudié les performances de nos modèles grâce à deux types de réponses :

— Une réponse à un échelon unité (voir figure 2.1).

— Une réponse fréquentielle représentée par un diagramme de Bode (voir figure 2.2).

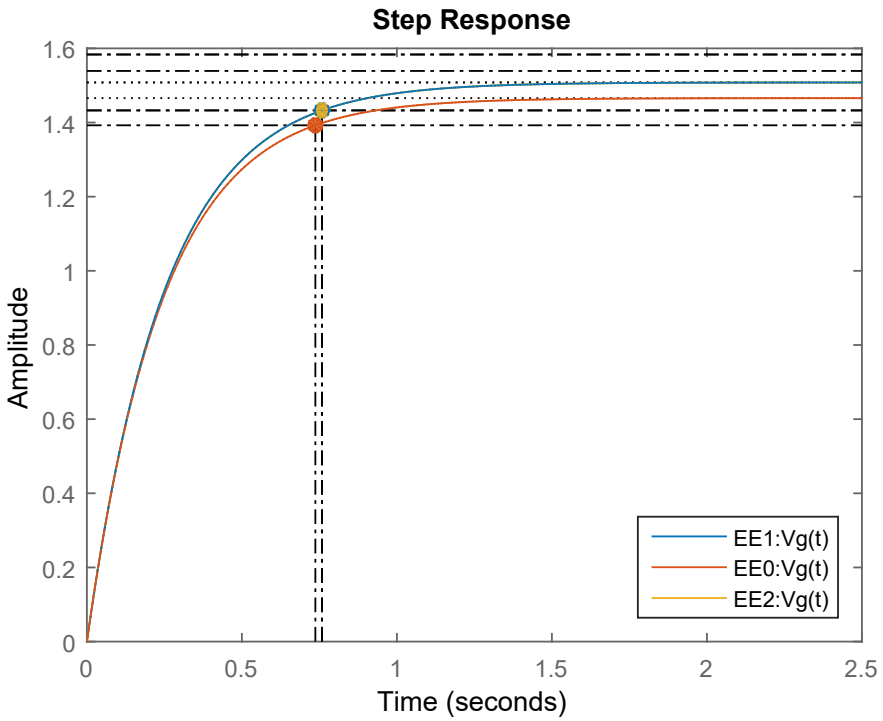


FIGURE 2.1 – Réponse à un échelon unité de $V_g(t)$ des modèles EE0, EE1 et EE2.

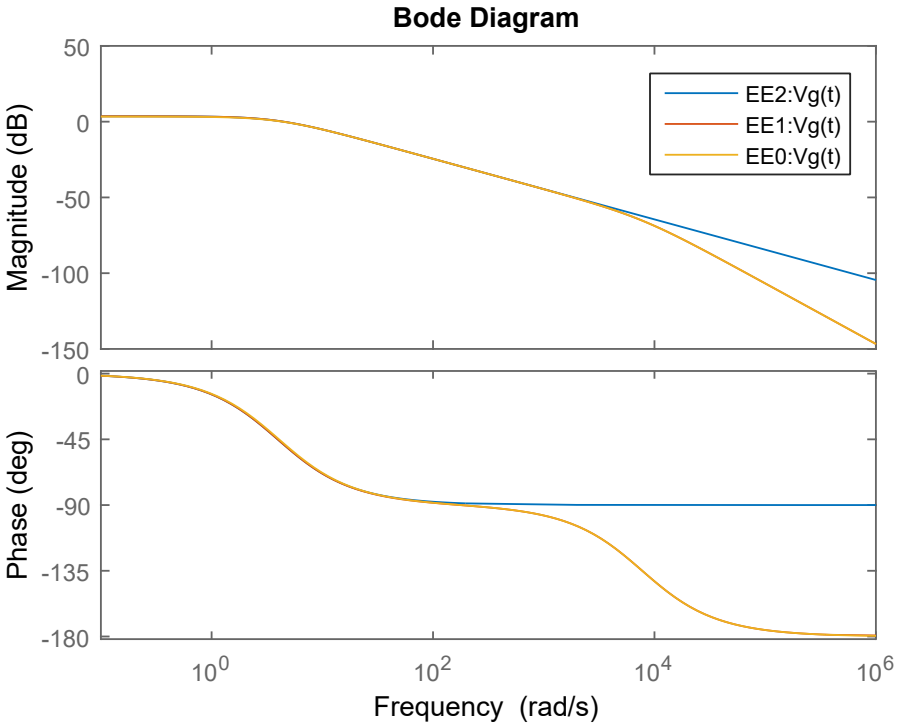


FIGURE 2.2 – Diagramme de Bode sur $V_g(t)$ des modèles EE0, EE1 et EE2.

2.2.1 Performances Statiques

Nous étudions $V_g(t)$ en temps que sortie de performance (figure 2.1).
Nos modèles présentent des gains statiques qui varient d'un modèle à l'autre, à cause des simplifications :

Gain statique de EE0 : $1,4658$

Gain statique de EE1 : 1.5080

Gain statique de EE0 : 1.5080

On peut constater qu'entre le modèle EE0 et le modèle EE1, il y a une erreur de 0.0423 et qu'en le modèle EE1 et le modèle EE2 l'erreur vaut $2.0241 * 10^{-8}$. La première simplification engendre une erreur d'environ 2% et la suivante de le l'ordre de 10^{-8}%. L'asservissement risquera donc de présenter une erreur statique causée par ces simplifications.

2.2.2 Performances dynamiques

Nous étudions $V_g(t)$ en temps que sortie de performance (figure 2.1).
Nous avons choisis d'étudier le temps de montée, le temps de réponse et le dépassement car ce sont des paramètres précis et déterminants en terme de performances.
Temps de montées :
Sur EE0 : 0.5404s
Sur EE1 : 0.5560s
Sur EE2 : 0.5561s
Erreur de EE0 à EE1 : 0.0156 s (2.8826%)
Erreur de EE1 à EE2 : 1.3830e-04 s(0.0249%)
Nous pouvons remarquer que le temps de montée est aux alentours d'une demie seconde. Les différences entre les temps de montées des différents modèles risquent de créer une erreur sur l'asservissement, en effet elle témoigne d'une différence dans les dynamiques des modèles (ce qui est une conséquence logique de la simplification). Le test de la commande sur EE0 risque de démontrer une erreur dynamique.

Temps de réponses à 5% :
Sur EE0 : 0.7370 s

Sur EE1 : 0.7582 s
Sur EE2 : 0.7583 s
Erreur de EE0 à EE1 : 0.0212 s (2.8821%)
Erreur de EE1 à EE2 : 5.9211e-05 s(7.8089e-05%)
Comme pour la performance précédente , la valeur du temps de réponse varie et cela entrainera de potentielles erreurs d'asservissement.
Dépassements :
Le modèle en boucle ouverte ne présente pas de dépassement.

2.2.3 Analyse fréquentielle

On peut remarquer sur le diagramme de Bode, figure 2.2, qu'il y a une compensation pôle zéro dans le modèle EE0 car nous avons uniquement trois changements d'allures. Cela se confirme sur une vue des pôles et des zéros sur un plan complexe (voir figure 2.3) ou une fonction de transfert. Cette figure met en évidence la compensation d'un pôle électrique qui agis à haute fréquence. Sur le diagramme de Bode, nous pouvons aussi constater que les modèles semblent avoir des réponses fréquentielles assez similaires. On peut aussi voir l'absence de la dernière dynamique de EE2 due à la simplification.

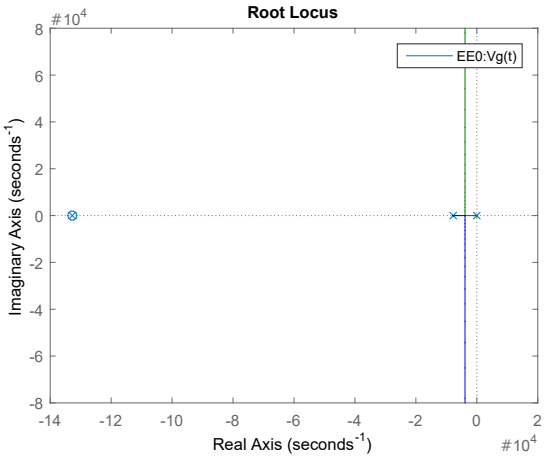


FIGURE 2.3 – Pôles et zéros dans le plan complexe du transfert vers $V_g(t)$ du modèles EE0.

Chapitre 3

Synthèse de commande

3.1 Commande du système d'ordre 2

3.1.1 Rédaction du cahier des charges et démarche de réponse

Après l'étude que nous venons de réaliser sur notre système, nous allons ici exprimé les attentes que doit réaliser la commande que nous allons implémenter. Nous souhaitons avoir :

— Erreur de position nulle.

— Pas d'oscillations

— Temps de réponse inférieur à 1 seconde.

La commande de notre système doit permettre d'asservir le système en vitesse, par rapport à une consigne. Pour respecter, nous allons réaliser un placement de valeurs propres par retour d'état.

3.1.2 Calcul du retour d'état

Pour garantir les performances dynamiques souhaitées, nous allons appliquer un placement de pôles par retour d'état. Ce réajustement des valeurs propres de la matrice dynamique du système nous permettra de répondre aux attentes du cahier de charges si le choix de celles-ci est correct. De même, nous devons choisir des valeurs propres qui ne sont pas être trop éloignées de celles du procédé, pour ne pas être trop exigeant avec la commande et le système. Nous avons avec ces spécifications choisi les valeur propres suivantes :

$$\lambda = \begin{pmatrix} -4 & -5 \end{pmatrix} \tag{3.1}$$

La commande par retour d'état est une technique d'asservissement qui permet de modifier le signal d'entrée du système en fonction de la sortie mesuré et une référence. Cette nouvelle loi de commande s'écrit :

$$u(t) = Ny_{ref} - Ky(t) \tag{3.2}$$

avec N un gain de pré-compensation, y_{ref} la vitesse de référence, $y(t)$ la sortie mesuré du système et K le gain de retour. Si l'on applique cette loi à notre système en espace d'état, on obtient :

$$\begin{cases} \dot{X}(t) = (A - BK)X(t) + BNy_{ref} \\ Y = CX(t) \end{cases}$$

Ainsi la nouvelle dynamique du système est donnée par la matrice $A' = (A - BK)$ et doit admettre les valeurs propres que nous désirons appliquer à notre système. A et B étant des paramètres du système, nous allons utiliser le gain K pour répondre à ce problème. Avec la fonction *place* de MATLAB, nous sommes capable de concevoir ce vecteur K.

3.1.3 Observateur ordre plein sur modèle d'ordre 2

Pour pouvoir réaliser notre commande par retour d'état, nous devons tous d'abord reconstruire l'ensemble des états du système dont nous n'avons pas accès. Dans notre cas, nous disposons d'une mesure de la vitesse Ω avec la tension de sortie V_s mais aucune information sur la position Θ, l'implémentation d'un observateur est donc nécessaire pour au minimum reconstruire cet état.

Nous préférons reconstruire Ω et θ à partir de V_s et de l'entrée du modèle d'ordre 2 pour simplifier les calculs nécessaire à sa construction. Il est représenté par :

$$\begin{cases} \dot{z}(t) = Fz(t) + Gy(t) + Hu(t) \\ \hat{x} = Mz(t) + Ny(t) \\ \epsilon = x - \hat{x} \end{cases}$$

où x représente l'état du système, $\hat{x}$ l'état du système reconstruit et ϵ l'erreur d'estimation à un temps t. Nous souhaitons contrôler la dynamique de ce paramètres pour pouvoir estimer correctement notre système. Pour cela, nous nous interressons à :

$$\begin{aligned} &\dot{\epsilon} = \dot{x} - \dot{\hat{x}} \\ &\Leftrightarrow \dot{\epsilon} = Ax + Bu - F\hat{x} - Gy - Hu \text{ en considérant } M = 1 \text{ et } N = 0 \\ &\Leftrightarrow \dot{\epsilon} = Ax - F\hat{x} - GCx + u(B - H) \\ &\dot{\epsilon} = (A - GC - F)x + F\epsilon + u(B - H) \end{aligned}$$

Il vient alors $F = A - GC$ et $B = H$ pour obtenir $\dot{\epsilon} = F\epsilon$. Ainsi l'erreur d'estimation est autonome et ne dépend pas des entrées et sorties du système, et il vient $\epsilon(t) = e^{Ft}\epsilon(0)$. Les valeurs propres de F vont ainsi déterminer la dynamique de l'erreur d'estimation, nous choisissons de les faire dépendre des valeurs propres désirées dans la partie 3.1.2 en les multipliant par 3, pour une convergence encore plus rapide.

3.1.4 Construction de l'asservissement

Maintenant que l'observateur et le gain du retour d'état sont construits, nous allons les assembler ensemble pour commander notre système. Pour ce modèle du système bouclé, nous prenons comme vecteur d'état $X = \begin{pmatrix} X & \epsilon \end{pmatrix}^T$ et nous obtenons par construction :

$$\begin{cases} \dot{X} = \begin{pmatrix} (A - BK) & -BK \\ 0 & F \end{pmatrix} X + \begin{pmatrix} B \\ 0 \end{pmatrix} y_{ref} \\ y(t) = \begin{pmatrix} C & 0 \end{pmatrix} X \end{cases}$$

Cependant, nous souhaitons asservir le système avec la vitesse ω, et la description de ce système bouclé va asservir le système en position à cause du changement de dynamiques de la position θ. Pour prévenir à cette mauvaise commande, nous avons annulé le retour de la mesure de position dans le système bouclé.

Le gain de pré-compensation N peut maintenant être calculé à partir gain statique du transfert entre Vg et y_{ref} noté $G_{Vg/y_{ref}}$:

$$N = \frac{1}{G_{Vg/y_{ref}}} \tag{3.3}$$

A partir de ce modèle, nous allons être capable d'éprouver la commande obtenu de manière théorique et avec une simulation SIMULINK.

3.2 Validation de la commande

3.2.1 Validation théorique

Pour commencer, nous allons identifier les valeurs propres de notre système **??** afin de de connaitre les effets de l'observateur sur le retour d'état, même si a priori celui ci est autonome. Nous obtenons les λ_i valeurs propres suivantes : -5, -4, -15 et -12. Nous remarquons que les λ_i n'ont pas été modifiées : nous avons les valeurs propres désirées par la commande et nous avons celles désirées pour l'erreur d'estimation de l'observateur.

3.2.2 Simulation SIMULINK

Pour compléter la validation de la commande, nous avons crée un prototype avec SIMULINK pour pouvoir simuler une réponse temporelle de la commande de notre système. La description du fichier se trouve en annexe 4.3. Nous observons sur les figures suivantes la réponse temporelle à une référence $y_{ref} = 1$:

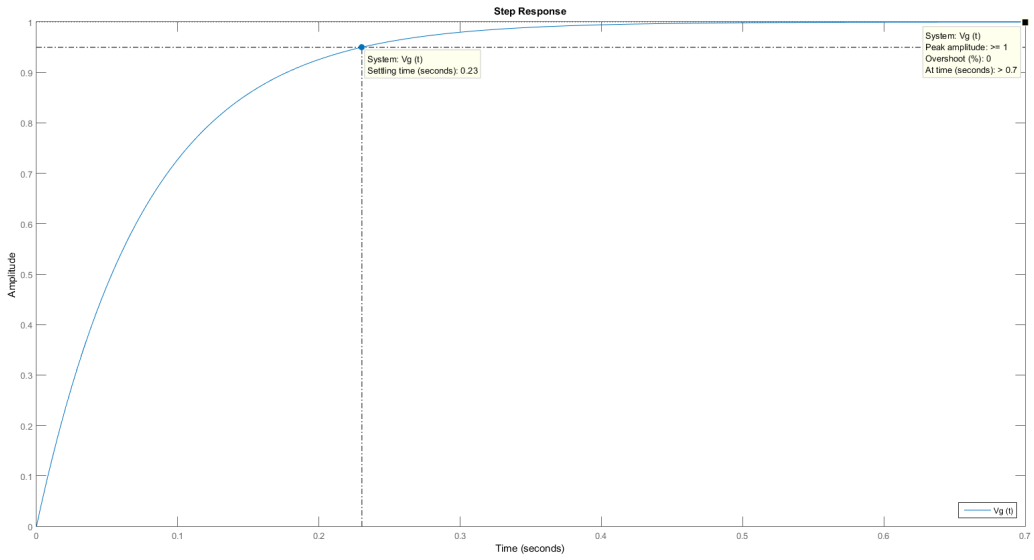


FIGURE 3.1 – Réponse du système asservi

La réponse du système Vs simulé correspond aux changement de dynamique que nous avons espéré. De plus, après une observation de la commande $u(t)$ appliquée sur le système, nous voyons que la tension maximale envoyée est borné dans l'ordre de grandeur des valeurs admises par le système physique.

3.2.3 Analyse boucle fermé du modèle d'ordre 3

Nous commande respecte le cahier des charges, nous validons ainsi la commande pour notre modèle d'ordre 2. Nous allons maintenant valider cette même commande sur des modèles supérieur et plus complexes pour compléter sa validation. En effet, notre modèle a été très simplifié pour permettre la création du commande facilement. Si nous arrivons a prouvé que cette commande marche sur des systèmes plus complexes, alors nous aurons prouvé que les dynamiques que nous avons simplifiés ne vont pas perturbé notre commande.

3.3 Validation sur les modèles d'ordre supérieur

Dans un premier temps, nous regarderons les effets de l'implantation de l'observateur d'ordre 2 sur les modèles d'ordre supérieur et plus précisément l'erreur d'estimation et le placement des valeurs propres, puis nous simulerons la commande obtenu pour observé les modifications des réponses.

Pour intégrer l'observateur dans un espace d'état d'ordre supérieur, nous avons du réorganiser les états du modèle, ici le modèle d'ordre 1, de façon à ce que les états reconstruits avec l'observateur soient en haut, x_1, et les non observable en bas x_2.

- Etats observables : Ω_m et Θ_m.
- Etat non observable : i_1
- L'espace d'état est donc :

$$\overline{X} = \begin{bmatrix} \theta \\ \omega \\ i_1 \end{bmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{3.4}$$

donc $x_1 = \begin{bmatrix} \theta \\ \omega \end{bmatrix}$ et $x_2 = i_1$. Pour passer de X à $\overline{X}$, il faut utiliser une matrice de passage P_X définit par :

$$P_X \quad / \quad \overline{X} = P_X \cdot X \tag{3.5}$$

$$P_X \quad = \quad \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.6}$$

$$\tag{3.7}$$

Maintenant, nous devons calculer $\overline{A}$, $\overline{B}$, $\overline{C}$, $\overline{D}$ pour le nouvel espace d'état lié à $\overline{X}$.

$$\begin{cases} \overline{A} &= P^{-1}AP \\ \overline{B} &= P^{-1}B \\ \overline{C} &= CP \end{cases} \tag{3.8}$$

Avec les résultats obtenus pendant les séances de TD et de cours, nous avons

$$\dot{\epsilon} = F\epsilon - A_2 x_2 \tag{3.9}$$

avec A_2 la partie qui lie les états x_1 et x_2 dans la nouvelle base. Cet élément modifie les résultats que nous avons obtenu pour le modèle d'ordre 2. Nous allons étudier les nouvelles valeurs propres du système bouclé pour vérifier si celles si correspondent toujours au cahier des charges, en modélisant un espace d'état qui admet comme vecteur d'état $x = \begin{pmatrix} \overline{X} & \epsilon \end{pmatrix}^T$. Nous obtenons : -7748, -20.61, -3.832 ± 6.931 et -2.784, la stabilité asymptotique en boucle fermé est respectée.

Les réponses temporelles du système en boucle fermé ont les caractéristiques suivantes :

— Le gain statique n'est plus respecté

— Le systèmes oscille à cause des valeurs propres complexes

— le temps de réponse reste dans la clause du cahier des charges

Pour terminer l'analyse de la boucle fermé, nous allons étudier le transfert de $\overline{X}$ avec la consigne et le comparer avec celui du modèle d'ordre 2. Cette analyse fréquentielle ne va pas utiliser le système bouclé utilisé pour calculer les valeurs propres dans le paragraphe précédent car nous allons ici prendre en compte que nous commandons le système en vitesse et non en position. Nous obtenons le diagramme de Bode suivant :

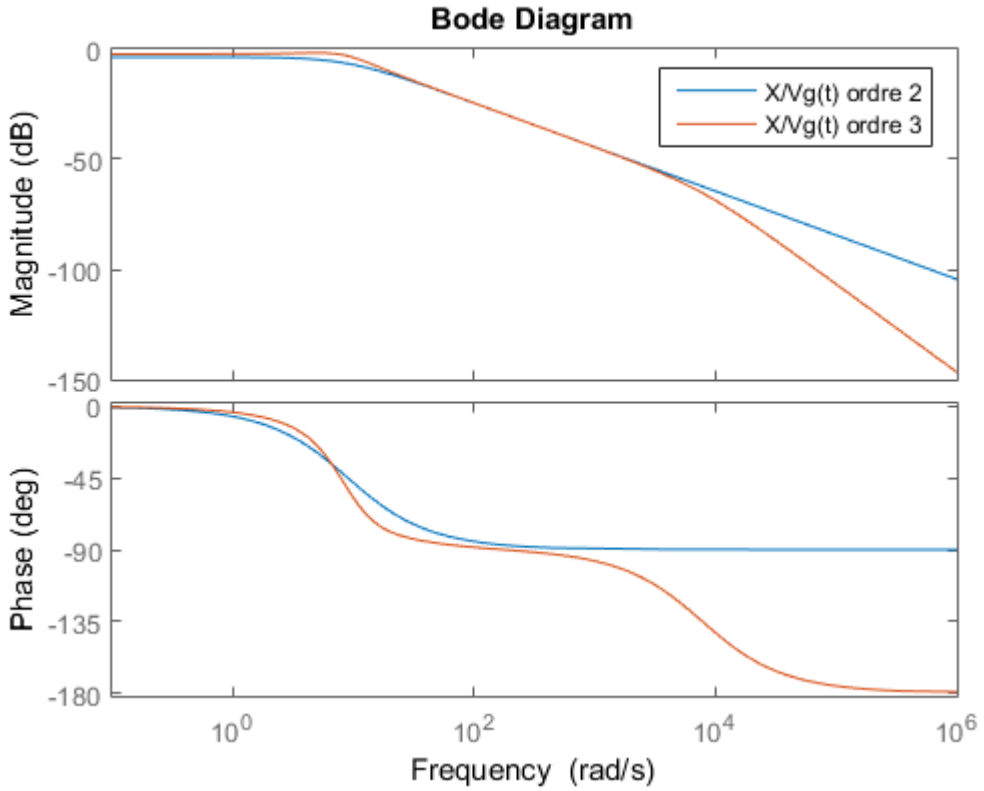


FIGURE 3.2 – Transfert des boucles fermées

Le résultat que nous obtenons confirmes la stabilité asymptotique établie précédemment, cependant les différences notables avec le transfert du modèle d'ordre 2 en basse fréquence explique pourquoi nous n'avons pas pu respecter exactement le cahier des charges.

3.3.1 Changement de base du modèle d'ordre 4 et intégration de la commande

De la même manière que pour le modèle d'ordre 3, nous effectuons un changement de base avec la matrice de passage :

$$P = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{3.10}$$

Les performances du système bouclé ont les mêmes problèmes que pour le modèle d'ordre supérieur. Avec le même résultat pour l'erreur d'estimation que pour le modèle d'ordre 3, ces résultats était prévisible.
De plus, les approximations de modélisation ont montré dans l'analyse des modèles une erreur statique que nous ne pouvons pas corriger avec une commande construite sur le modèle d'ordre 2. Nous avons toutefois voulu analyser le transfert entre Vg et la consigne et obtenons :

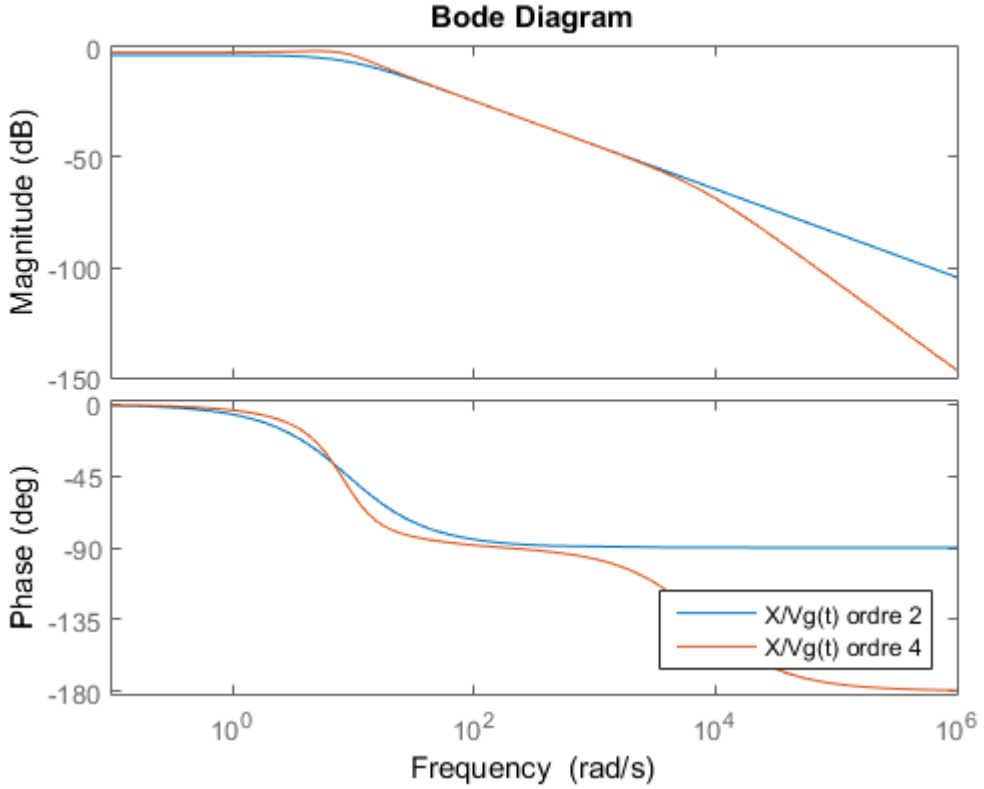


FIGURE 3.3 – Transfert des boucles fermées

Nous avons de plus amples écart en basse fréquence avec un dépassement encore plus grand que pour le modèle d'ordre 3. Ce dépassement joue beaucoup sur les oscillations du système en boucle fermé. Le cahier des charges n'est encore une fois pas respecté pour ce modèle mais admet un temps de réponse proche de celui du modèle d'ordre 2.
En sachant ceci, nous savons que les dynamiques qui ont été simplifiés ne sont pas l'origine des écarts que nous venons d'exposer, il s'agit de la dynamique de l'observateur qui n'est pas très optimale.

3.3.2 Simulation SIMULINK

Nous avons réutilisé la commande simulé précédemment pour l'implanter dans les modèles d'ordre 3 et 4. Ce prototypage rapide a eu les résultats suivant :

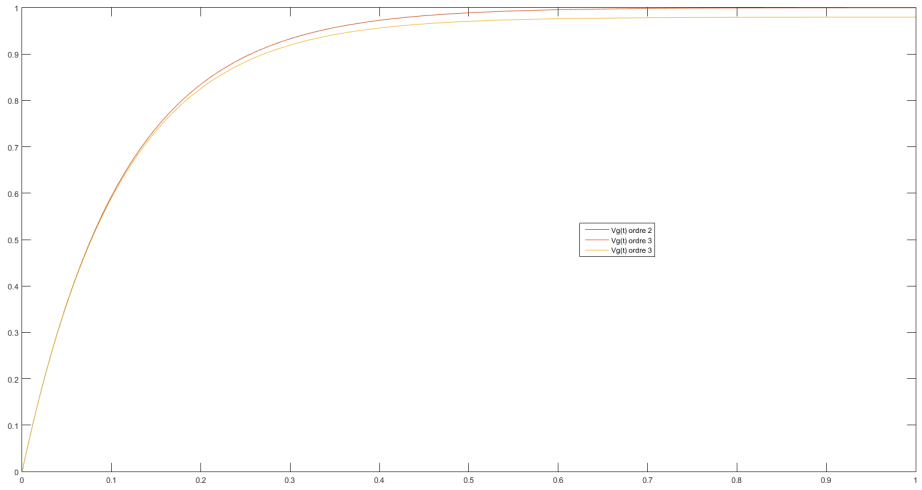


FIGURE 3.4 – Réponses temporelles des boucles fermées sous SIMULINK

Ces résultats nous permettent de regarder une simulation des signaux de $Vg(t)$. D'après ces résultats, la commande que nous souhaitons appliquer respecte beaucoup mieux le cahier des charges que d'après l'analyse précédente. Cependant, nous ne pouvons en tirer aucune conclusion , il reste encore d'autres modèles plus complexes sur lequel notre commande doit être tester.

Chapitre 4

Planification de la suite de l'asservissement

Dans ce chapitre, nous allons décrire les étapes suivantes à cette étude théorique. Dans un premier temps sera présenté la validation de notre commande par tests puis il sera abordé les étapes nécessaires à l'implémentation sur micro-contrôleur et la validation finale.

4.1 Validation de commande

Dans un premier temps, nous testerons notre asservissement sur un modèle plus riche (non linéaire, variant, plus de dynamique, ...) en simulation afin de voir si notre commande respecte toujours le cahier des charges. Le modèle sera fournie en seconde séance de TP. Ensuite, nous testerons le moteur dans différentes configurations en fonction des résultats et de notre vitesse d'avancement :

- Émulation sur le même ordinateur (test du temps concret).
- Test de la commande et du procédé sur un même ordinateur en sortant le signal de commande par les cartes d'entrées/sorties du prototypage rapide.
- Test de la commande simulée sur un ordinateur et le procédé simulé sur un second ordinateur afin de tester deux vitesses de fonctionnement de simulation (synchronisations) et les cartes entrées/sorties (communications).
- Test de la commande simulée et le procédé simulé sur un micro-contrôleur.
- Test de la commande simulée sur procédé réel : on éprouve ici la robustesse de la commande face aux imprécisions de modélisation.

Nous effectuerons aussi une estimation du modèle de comportement du moteur que nous asservirons afin d'avoir un modèle plus proche du comportement réel.

4.2 Implémentation sur micro-contrôleur

Dans cette partie, nous appréhenderons les problématiques de conversions de signaux numériques/analogiques et analogiques/numériques et essaierons de les corriger afin d'avoir des conversions les plus linéaires possibles. Nous calculerons les fréquences maximales et minimales des entrées et sorties du procédé, cela nous permettra de choisir des fréquences de conversions adaptées et la fréquence de fonctionnement du micro-contrôleur. Nous transformerons aussi notre modèle de commande en un modèle à temps discret afin de pouvoir l'implémenter. Une fois le modèle temps discret obtenu, nous programmerons les différentes fonctions nécessaires à l'asservissement (lecture des entrées, calcul des sorties, écriture des sorties). Nous aborderons aussi les problématiques d'ordonnancement afin que notre programme puisse s'exécuter dans le temps impartie afin de respecter les contraintes temps réels. Enfin, nous implémenterons notre programme sur le micro-contrôleur C167.

4.3 Validation finale

Une fois la commande implémentée, nous effectuerons les tests suivants :

- Test de la commande implémentée sur le procédé simulé sur ordinateur.
- Test de la commande implémentée sur le procédé simulé sur micro-contrôleur.

— Test de la commande implémentée sur la maquette du procédé réel.

Nous vérifierons le respect des spécifications du cahier des charges dans l'ensemble de ces tests afin de voir si notre asservissement est correct.

Annexes

Annexe 1 - Scripts Matlab

Modèles et analyses

```
clear
close all
MoteurScript
%% Modele Niveau 0 - Ordre 4 - EE
%%% Modele espace d'etat LINEAIRE
% etat    = [ i1 ; i2 ; theta ; omega ]
% entree  = [ Vm ]
% sorties = [ Vg ; Vs]

% Matrices EE

EE0.a   =[-R/L,     0,              0,   -Ke/L;
          0,        -(R+Rchn)/L,    0,   -Ke/L;
          0,        0,              0,   1;
          Kc/J2,    Kc/J2,          0,   -mu/J2];

EE0.b   =[1/L;
          0;
          0;
          0];

EE0.c   =[0, 0, 0,      Kg;
          0, 0, Kr*Ks,  0 ];

EE0.d   =[0;
          0];
% EE
EE0.ee= ss(EE0.a,EE0.b,EE0.c,EE0.d);
% Valeurs propres
EE0.vp = eig(EE0.ee);

% gain statique
EE0.gain = dcgain(EE0.ee(1));
%% Modele Niveau 1 - Ordre 3 - EE
%%% Modele espace d'etat LINEAIRE
% etat    = [ i1 ; omega ; theta ]
% entree  = [ Vm ]
% sorties = [ Vg ; Vs]

% Matrices EE
EE1.a = EE0.a([1,3,4],[1,3,4]);
EE1.b = EE0.b([1, 3, 4]);
EE1.c = EE0.c(:,[1, 3, 4]);
EE1.d = EE0.d;

% Espace Etat
EE1.ee= ss(EE1.a,EE1.b,EE1.c,EE1.d);
```

```
48 % Valeurs propres
49 EE1.vp = eig(EE1.ee);
50
51 % gain statique
52 EE1.gain = dcgain(EE1.ee(1));
53 %% Modele Niveau 2 - Ordre 2 - EE
54 %%% Modele espace d'etat LINEAIRE
55 % etat    = [ theta ; omega ]
56 % entree  = [ Vm ]
57 % sorties = [ Vg ; Vs]
58
59 % Matrices EE
60 % EE2.a = [ -(Ks*Ke)/(J2*R) , 0 ;
61 %            1               , 0 ]; % ca marche pas...
62 EE2.a = [  0   ,   1 ;
63            0   ,   -(Kc*Ke)/(R*J2)-(mu/J2)];
64 EE2.b = [ 0          ;
65           Kc/(J2*R) ];
66
67 EE2.c = EE1.c(:,[2, 3]);
68 %EE2.c =[];
69 EE2.d = EE1.d;
70
71 % EE
72 EE2.ee= ss(EE2.a,EE2.b,EE2.c,EE2.d);
73 % Valeurs propres
74
75 EE2.vp = eig(EE2.ee);
76
77 % gain statique
78 EE2.gain = dcgain(EE2.ee(1));
79
80 % Commandabilitee
81 Controlabilite = ctrb(EE0.a,EE0.b);
82 disp('Controlable ?')
83 disp(rank(Controlabilite) == size(EE0.a,1))
84
85 % Observabilite
86
87 [ABAR,BBAR,CBAR,T,K] = obsvf(EE0.a,EE0.b,EE0.c)
88
89 % etude de performance temporelle
90 EE0.stepChar = stepinfo(EE0.ee(1),'SettingTime',0.05)
91 EE1.stepChar = stepinfo(EE1.ee(1),'SettingTime',0.05)
92 EE2.stepChar = stepinfo(EE2.ee(1),'SettingTime',0.05)
```

Observateur et Asservissement

```
1 %% Creation de notre observateur
2 % reconstruction de tous les atats :
3 espaceEtat1
4
5 %% Cahier des charges
6 vp_desire = [-8; -5];
7 %% observateur
8 obsver.H = EE2.ee.b;
9 obsver.M = eye(size(EE2.ee.a));
10 obsver.vp = vp_desire*3; %*3
11 obsver.G = place(EE2.ee.a', EE2.ee.c(2,:)', obsver.vp)'; % Ne pas utiliser
       acker
12
```

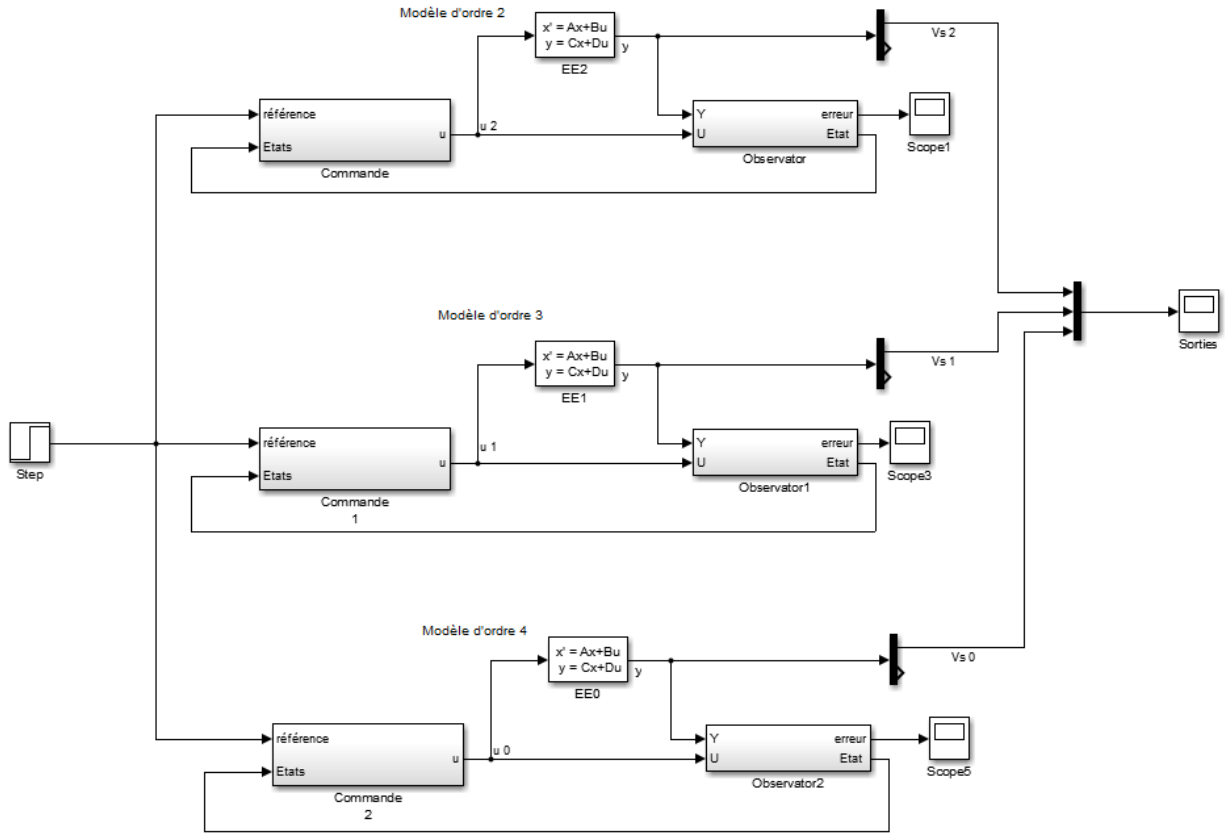
```
obsver.F = EE2.ee.a - obsver.G*EE2.ee.c(2,:);
           %   a        b               c       D
obsver.ee = ss(obsver.F, [obsver.G obsver.H], obsver.M, 0);

%% commande

K = place(EE2.ee.a, EE2.ee.b, vp_desire);
K = [0 K(2)];
EE2_bf.a = [EE2.ee.a-EE2.ee.b*K     -EE2.ee.b*K;
            [0 0; 0 0]              obsver.F];

EE2_bf.b = [EE2.ee.b; 0;0];

EE2_bf.c = [EE2.ee.c [0 0;0 0]];
EE2_bf.ee = ss(EE2_bf.a, EE2_bf.b, EE2_bf.c, EE2.ee.d);

EE2_bf.gain = dcgain(EE2_bf.ee(1));
%%  Analyse du retour d'etat base observateur
% Pour EE1 :
%Changement de base de EE1
P = [0     0     1;
     1     0     0;
     0     1     0];
EE1_c.a = inv(P)*EE1.ee.a*P;
EE1_c.b = inv(P)*EE1.ee.b;
EE1_c.c = EE1.ee.c*P;

EE1_c.ee = ss(EE1_c.a,EE1_c.b,EE1_c.c,EE1.ee.d);

% Partitionnement de l'EE
EE1.A1 = EE1_c.ee.a(1:2,1:2);
EE1.A2 = EE1_c.ee.a(1:2,3);
EE1.A3 = EE1_c.ee.a(3,1:2);
EE1.A4 = EE1_c.ee.a(3,3);

EE1.B1 = EE1_c.ee.b(1:2);
EE1.B2 = EE1_c.ee.b(3);

EE1.C = [EE1_c.ee.c];


% Construction des matrices
EE1_obsver.A = [EE1.A1      EE1.A2     [0 0;0 0]   ;
                EE1.A3      EE1.A4      [0 0 ] ;
                [0 0;0 0]   -EE1.A2       obsver.F];
EE1_obsver.B = [EE1.B1; EE1.B2; [0;0]];
EE1_obsver.C = [EE1.C [0 0;0 0];
                    [0 0 0 0 1]];   % Pour affficher de l'erreur
                                    % de reconstrcution de la

% Espace d'etat
EE1_obsver.ee = ss(EE1_obsver.A ,EE1_obsver.B, EE1_obsver.C, [0;0;0]);


% gain statique
EE1_obsver.gain = dcgain(EE1_obsver.ee(1));
% comparaison de gain statique

err_gain_stat=EE2.gain - EE1_obsver.gain;

% Analyse pour EE0
```

```
74 % Changement de base [theta, omega, i1, i2]
75 P_0 = [0 0 1 0; 0 0 0 1; 1 0 0 0; 0 1 0 0];
76
77 EE0_c.a = inv(P_0)*EE0.ee.a*P_0;
78 EE0_c.b = inv(P_0)*EE0.ee.b;
79 EE0_c.c = EE0.ee.c*P_0;
80
81 EE0_c.ee = ss(EE0_c.a,EE0_c.b,EE0_c.c, EE0.d);
82 % Partitionnement de l'EE
83 EE0.A1 = EE0_c.ee.a(1:2,1:2);
84 EE0.A2 = EE0_c.ee.a(1:2,3:4);
85 EE0.A3 = EE0_c.ee.a(3:4,1:2);
86 EE0.A4 = EE0_c.ee.a(3:4,3:4);
87
88 EE0.B1 = EE0_c.ee.b(1:2);
89 EE0.B2 = EE0_c.ee.b(3:4);
90
91 EE0.C = [EE0_c.ee.c];
92
93
94 % Construction des matrices
95 EE0_obsver.A = [EE0.A1      EE0.A2     [0 0;0 0]   ;
96                 EE0.A3      EE0.A4      [0 0;0 0 ] ;
97                 [0 0;0 0]   -EE0.A2       obsver.F];
98 EE0_obsver.B = [EE0.B1; EE0.B2; [0;0]];
99 EE0_obsver.C = [EE0.C [0 0;0 0]];   % Pour afffiche de l'erreur
100                                    % de reconstrcution de la
101
102 EE0_obsver.ee = ss(EE0_obsver.A, EE0_obsver.B, EE0_obsver.C, EE0_c.ee.d);
103
104 % Analyse du trasfert de epsilon
105 EE0_obsver.vp = eig(EE0_obsver.ee);
106 % bodemag(EE1.ee(1), EE1_obsver.ee(1))
107
108 %%
109
110
111 %% Calcul du systeme en boucle ferme base observateur de EE1
112 %       cf Cours de GOUAISBAULT
113 % Etat EE0 : x = [ i1    ; i2    ; theta ; omega ]
114 % Etat EE1 : x = [ i1    ; theta ; omega ]
115 % Etat EE2 : x = [ theta ; omega ]
116
117 % EE1
118
119 EE1_bf.a = [EE1.A1-EE1.B1*K      EE1.A2     -EE1.B1*K   ;
120             EE1.A3-EE1.B2*K      EE1.A4     -EE1.B2*K;
121                 [0 0;0 0]       -EE1.A2       obsver.F];
122 EE1_bf.b = EE1_obsver.B;
123 EE1_bf.c = EE1_obsver.C;
124
125 EE1_bf.ee = ss(EE1_bf.a, EE1_bf.b, EE1_bf.c, EE1_obsver.ee.d);
126 EE1_bf.vp = eig(EE1_bf.ee);
127
128 % EE0
129
130
131 EE0_bf.a = [EE0.A1-EE0.B1*K      EE0.A2     -EE0.B1*K   ;
132             EE0.A3-EE0.B2*K      EE0.A4     -EE0.B2*K;
133                 [0 0;0 0]       -EE0.A2       obsver.F];
134 EE0_bf.b = EE0_obsver.B;
```

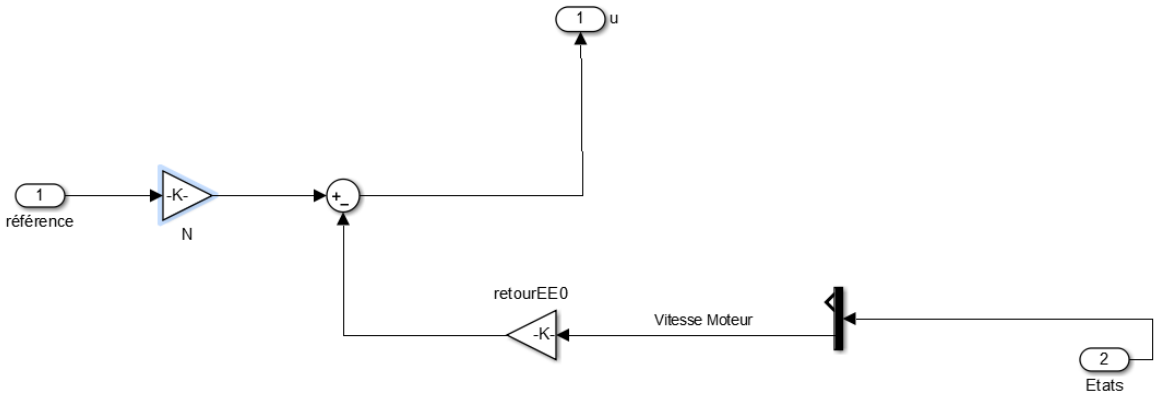
```
135 EE0_bf.c = EE0_obsver.C;
136
137 EE0_bf.ee = ss(EE0_bf.a, EE0_bf.b, EE0_bf.c, EE0_obsver.ee.d);
138 EE0_bf.vp = eig(EE0_bf.ee);
```

Annexe 2 - Modèles SIMULINK

Modèle Global



Sous-système de commande



Sous-système d'observateur