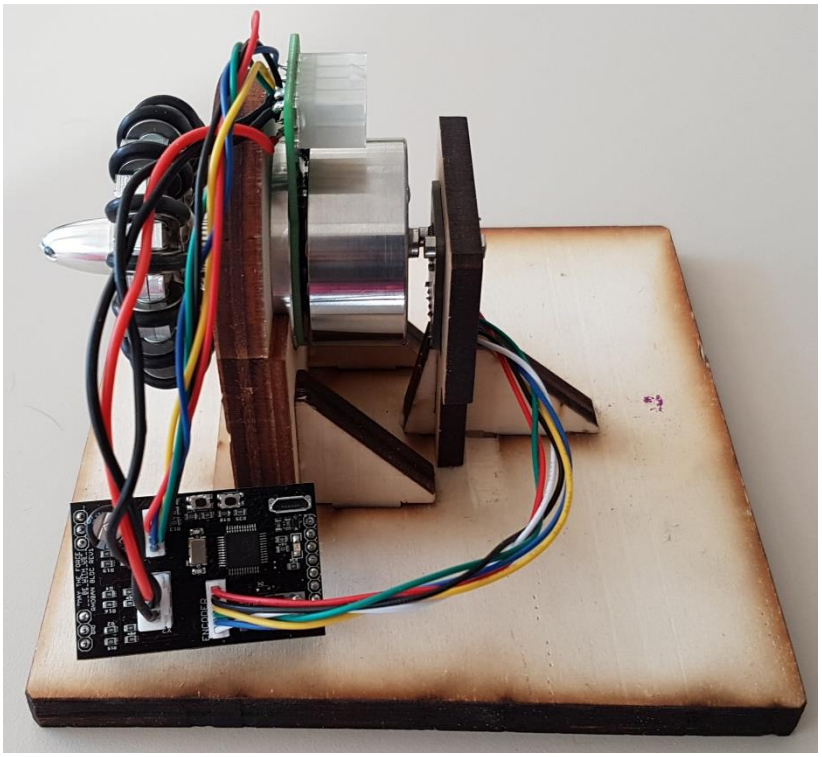
LOO Darren
SALIBA Thomas
LAKHAL Nada
SEHOUBA Victoire

ROBOCUP SSL

- PREMISSES A L'ASSERVISSEMENT D'UN ROBOT HOLONOME -



SOMMAIRE

Contexte

La Robocup est l'une des compétitions de robotique la plus connue dans le monde. Elle regroupe plusieurs catégories : Robocup Soccer (SSL, MSL, Humanoid, etc..), Robocup Rescue, Robocup @Home, Robocup Industrial et Robocup Junior. La catégorie qui nous intéresse ici est la SSL (Small Size League) de la Robocup Soccer.

Le principe est de se faire affronter deux équipes de six robots à roues qui sont pour la plupart holonomes dans un match de football. Les défis sont donc d'arriver à faire un robot maniable, précis et rapide mais sans qu'il soit incontrôlable. L'objectif de ce livret est de démarrer l'implémentation d'un asservissement d'un robot SSL par une approche rigoureuse et méthodologique de l'automatique.

Ce livret contient donc les parties suivantes :

- ➔ Décomposition du robot
- ➔ Création d'un TestBench
- ➔ Identification d'un moteur seul
- ➔ Asservissement d'un moteur seul
- ➔ Identification d'un moteur in situ
- ➔ Asservissement d'un moteur in situ

Chaque partie sera introduite et expliquée de manière simple et didactique. Cependant certaines explications seront succinctes car il existe des documents qui approfondissent beaucoup mieux le propos (Voir Bibliographie pour plus de détails). L'étude se fera sur un robot holonome de l'équipe Rhoban utilisant 4 moteurs brushless Maxon EC 45 flat en Direct Driving.

Décomposition du robot

Pour contrôler parfaitement un robot, il faut le connaître, c'est-à-dire identifier son comportement par rapport à certaines commandes. Cette phase, l'identification, doit être faite plusieurs fois à différents niveaux : une roue, toutes les roues ensemble puis le robot entier.

Dans un premier temps, il est nécessaire de pouvoir contrôler correctement chaque moteur. Pour cela, il faut effectuer une identification du moteur : il faut caractériser son comportement et sa réponse à une commande. Pour cela, un banc de test a été conçu pour accueillir un moteur et effectuer des tests plus facilement. Dans cette première version, le moteur tourne avec une roue holonome : il n'y a que la charge de la roue sur l'axe du moteur.

Une fois que le moteur a été identifié, que son comportement a été caractérisé, que son modèle de comportement a été établi, il devient possible de synthétiser un correcteur qui permettra d'obtenir du moteur une vitesse souhaitée et ce quel que soit les perturbations. On obtient donc la chaine suivante :

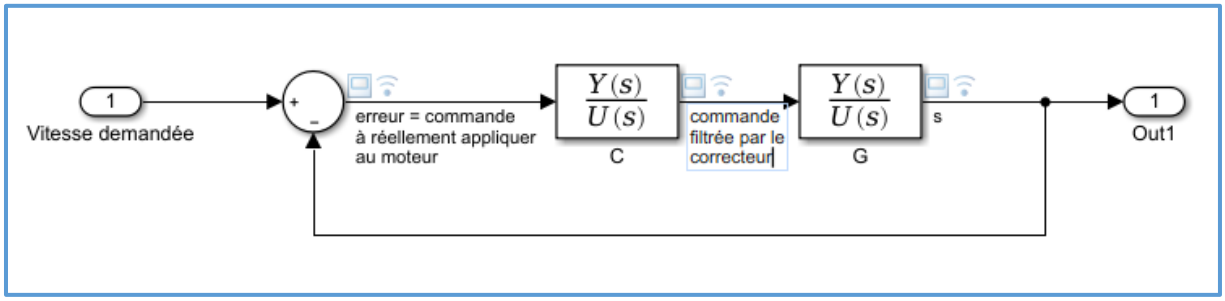


Figure 1 : Schéma bloc d'une chaîne asservie

G représentant le comportement du moteur sous forme de fonction de transfert (vitesse de rotation/tension appliquée), et *C* le correcteur que l'on synthétisera par la suite. Il est à noter que le retour sur la vitesse de la roue provient d'un encodeur magnétique captant le champ tournant d'un aimant fixé à l'arrière du moteur brushless.
Une fois cette chaîne implémentée et validée, on peut se concentrer sur un autre banc de test qui met en situation le moteur avec un certain poids appliqué en sa surface. L'idée est de refaire l'identification du moteur avec les contraintes qu'il aurait sur le robot. La chaine précédente est surtout pour valider tous les éléments nécessaires : le capteur, le calcul, l'implémentation. Car une fois tout ceci validé, refaire une identification et la mettre en place est très simple.

Maintenant, chaque processus va être détaillé.

Conception du premier banc de test

Il y a plusieurs manières d'effectuer un banc de test. Ce qui sera décrit par la suite se rapproche de ce qui a semblé être le plus simple à mettre en œuvre.
Les contraintes qui ont été fixées sont les suivantes :

-le moteur doit être soutenu dans son intégralité ;

-la roue ne doit pas toucher le sol ;

-l'ensemble doit être stable lors des tests ;

-il faut que l'encodeur magnétique soit parfaitement en vis-à-vis de l'aimant.

En effet, il faut un maximum de stabilité pour éviter que des perturbations viennent altérer les mesures. Le placement de l'encodeur magnétique est très important car c'est lui qui effectue les mesures sur lesquelles tout l'asservissement se base.

Globalement, le banc de test est composé de 5 pièces différentes. Toutes les pièces ont été conçues avec le logiciel Autodesk Inventor et ont été faite par découpage laser sur bois.

Concevoir une pièce

La conception d'une pièce est séquentielle. En effet, on ne peut généralement travailler que sur une surface à la fois. Il faut donc avoir une bonne idée des objectifs que la pièce devra remplir. Des ébauches au crayon sont toujours un bon début. Dans notre cas, puisque les pièces seront faîtes par découpage laser, il n'y aura qu'une face à faire. Toutefois, la démarche peut être réitérée autant de fois que possible pour effectuer des pièces plus compliquées. Dans tous les cas, il faut toujours garder à l'esprit le moyen qui sera utilisé pour fabriquer la pièce (découpage, impression 3D, pliage, emboutissage, usinage etc…), certaines formes étant impossible à réaliser selon le procédé d'usinage retenu.

Dans notre cas, les 5 pièces sont :

- le socle du banc de test ;
- un support pour le moteur ;
- un support pour la carte de l'encodeur ;
- des soutiens pour les supports (2 identiques pour le moteur et 1 pour la carte) ;

Dessiner une face

Pour dessiner une face plane, il faut créer une esquisse. Dans un premier temps, dans le volet de navigation du modèle, sélectionner un plan qui servira de référence à l'esquisse. Puis créer une esquisse. Le logiciel propose plusieurs fonctions qui permettent de créer des formes géométriques simples. L'outil « Ajuster » (icône de ciseaux) permet de supprimer des portions de trait non désirées.

Pour fixer des éléments par rapport à d'autres, il faut utiliser des contraintes. Il y a plusieurs types de contraintes : de dimensions (qu'on appelle des côtes), de positionnement (coïncidence de points, colinéarités d'axes ou parallélisme…). Lorsqu'un élément est totalement contraint, il apparaît en bleu sur l'esquisse, les éléments non contraints étant en vert. Contraindre totalement une esquisse a plusieurs avantages. Dans un premier temps, cela permet de maîtriser totalement les dimensions / orientations que prendra la pièce. De plus, si une esquisse est bien contrainte, il est aisé d'en changer les dimensions sans que cela ait d'impact néfaste sur tout le travail qui sera effectué après et qui se base sur l'esquisse même. Aussi, cela permet à un tiers de voir clairement les relations qu'il peut y avoir entre les éléments ne laissant plus aucune place au doute.

Enfin, une fois que l'esquisse est réalisée et prête, il est possible de l'extruder : de l'utiliser pour créer de la matière ou encore de l'utiliser pour enlever de la matière (sur une pièce déjà existante par exemple). Dans notre cas, une extrusion de 10mm a été effectuée sur toutes les pièces. En effet, les planches de bois qui passeront au découpage sont d'épaisseur 10mm.

Assembler les pièces

Une fois que toutes les pièces ont été réalisées, il est possible de les assembler afin d'avoir un bon aperçu du montage final.

Il faut créer un fichier de type « Ensemble » et y placer les différentes pièces qui ont été faites. Pour des raisons pratiques, il est préférable de commencer par la pièce centrale du montage. Puis, il faut bloquer la pièce dans l'espace pour inhiber ses déplacements. Ensuite il faut rajouter les autres pièces.

Pour placer une pièce, il faut aussi lui imposer des contraintes, de position (distance, colinéarité…) et de surface (placage en vis-à-vis ou affleurement etc…).

Pour qu'une pièce fixe soit bien contrainte, il faut lui retirer ses 6 degrés de liberté dans l'espace (3 translations sur les axes du repère et 3 rotations autour de ces axes). Si une pièce est mobile, il faut bien sûr lui laisser les degrés de liberté nécessaires.

Afin de vérifier qu'un assemblage se passe sans accroc, il est possible de changer le style visuel des pièces. Le paramètre « Filaire avec arêtes masquées » est plutôt bon pour cette tâche. Le logiciel n'informe pas lorsque deux pièces se superposent.

Découpage des pièces

Pour passer à la découpeuse laser, il faut exporter les faces de chaque pièce en format .dxf. Puis les ouvrir et les réenregistrer avec le logiciel QCad. Il faut ensuite les ouvrir avec Inkscape, tout sélectionner, tout dégrouper et appliquer les bons paramètres pour la découpe sur les contours et les fonds. Il ne reste plus qu'à laisser faire la machine et assembler les pièces.

Première identification du moteur

Principe

Le concept est de cette phase est donc de caractériser le comportement moteur. Pour cela il existe plusieurs approches : par représentation d'états ou par fonction de transfert. La première méthode consiste, à partir des équations différentielles régissant un moteur, à dresser un modèle basé sur des états (internes ou non) du système, représenté par des matrices. Cependant, cette méthode est compliquée à appréhender et l'implémentation numérique est plus complexe qu'à partir d'une fonction de transfert. La deuxième méthode, celle utilisée ici, consiste à modéliser le moteur par une fonction de transfert (domaine fréquentiel). On va donc exciter le moteur en lui appliquant une tension en entrée et observer sa réponse.

L'excitation la plus réputée est l'application d'un échelon. De fait, fréquentiellement, l'échelon comporte une richesse fréquentielle ce qui fait que l'on va pouvoir observer plusieurs choses :

- ➔ Le dépassement
- ➔ L'erreur statique
- ➔ Le gain statique
- ➔ Le temps de montée
- ➔ La pseudo-pulsation
- ➔ Etc…

Ces facteurs-là sont plus ou moins observables selon la forme de la réponse du moteur. La fonction de transfert a souvent la forme suivante :

$$G(p) = \frac{Gain\ statique}{\left(1+\frac{p}{w_{c1}}\right)\left(1+\frac{p}{w_{c2}}\right)\left(1+\frac{p}{w_{c2}}\right)\dots}$$

p étant jw avec j l'imaginaire et w la pulsation et wc les pulsations de coupures.

Dans la vie réelle, la plupart des systèmes sont des filtres de type passe-bas d'où la présence de pôles et non de zéros dans la fonction de transfert. Les pulsations de coupures représentent des fréquences caractéristiques du système étudié. On les retrouve grâce à l'identification. On peut également les retrouver leur expression analytique en passant par les équations théoriques.

Dans le modèle d'un moteur, il existe deux pulsations de coupures : la coupure mécanique et la coupure électrique. La première est due à tous les frottements et les inerties mécaniques présentes sur le moteur et la deuxième à la bobine et résistance du moteur. Généralement, la deuxième pulsation étant très grande devant la première, on a tendances à la négliger car le gain aux fréquences élevées est si bas que cette caractéristique est négligeable. On se retrouve donc souvent avec une fonction d'ordre 1.

Dans la partie suivante, il sera montré comment a été identifié le moteur.

Mesures

La première étape est l'observation de la réponse du système à une consigne. Pour cela, il faut récupérer à la fois les mesures de vitesse de rotation du moteur, la commande qui lui a été envoyé et traiter ces données (mise en forme et exploitation). Toute cette partie est compilée dans un script Matlab.

Les données relatives à la vitesse de rotation du moteur sont exprimées en centième de tour par seconde. Elles doivent être converties en rad/s. Aussi, les données relatives à la commande sont proportionnelles à cette dernière. Avant de pouvoir les exploiter correctement, il faut les ramener à des unités conventionnelles : rad/s pour la vitesse de rotation et V pour la commande.

Une série de deux échelons est envoyée au moteur afin de ne pas prendre dans les mesures les effets des frottements secs et autres perturbations agissant sur le moteur au démarrage.

Le second échelon correspond à une variation de PWM de 200 à 800 envoyée au moteur. Cela correspond à une variation de la tension de commande de $1.6V$ à $6.4V$.

Un premier tracé des données est effectué. Il permet dans un premier temps de choisir le modèle qui servira à l'identification du procédé, puis dans un second temps de définir les plages de données qui seront utilisées pour déterminer les éléments du modèle. En effet, un léger bruit interfère avec les données mesurées. Puisqu'en régime permanent la vitesse du moteur devrait être constante, une moyenne est effectuée sur beaucoup de points afin de diminuer les effets du bruit.

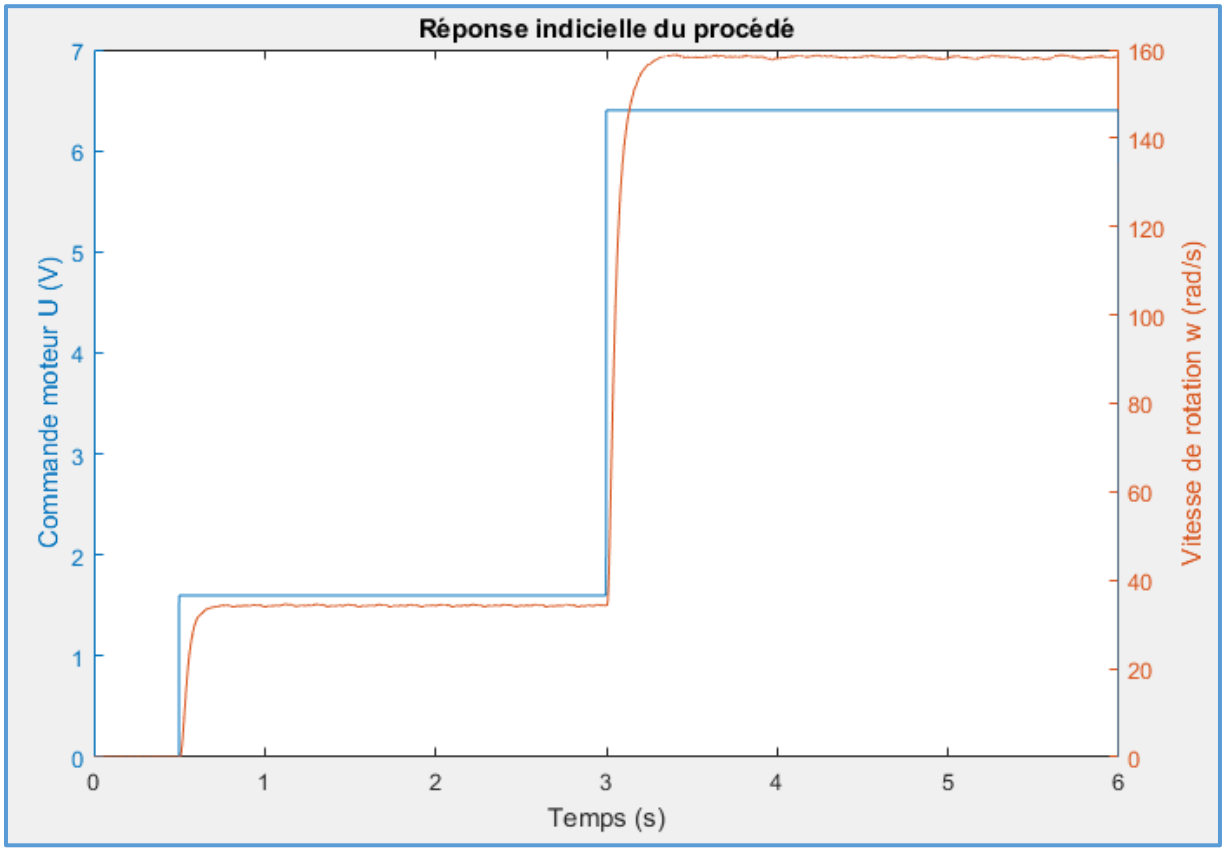


Figure 2 : Réponse indicielle du procédé – Tracé préliminaire

D'après l'allure de la courbe représentant la vitesse de rotation du moteur par rapport au temps, il est possible de modéliser le procédé par un système du second ordre apériodique (très amorti). En effet, elle s'apparente fortement à l'allure de la réponse indicielle d'un système du premier

ordre (pas de dépassement de la valeur finale) toutefois, la pente à l'origine (ici *3s*, origine du second échelon) n'est pas nulle. On en déduit la forme de la fonction de transfert du procédé suivante :

$$G(p) = \frac{K_0}{(1+\tau_1 p)(1+\tau_2 p)},$$

K_0 étant le gain statique, τ_1 et τ_2 les constantes de temps associées aux pulsations de coupure du procédé.

On mesure $K_0 = 25.8532$. Cela nous permet de tracer l'image en tension de la vitesse de rotation du moteur afin de l'avoir sur la même échelle que la tension de commande du moteur. Cette mise à l'échelle est nécessaire pour pouvoir mettre en œuvre la méthode des tangentes et obtenir les valeurs des constantes de temps.
En plaçant la tangente au point d'inflexion de la courbe, on a τ_1 à l'intersection de cette dernière et de la droite $y = 1.6$ (valeur de l'échelon bas). De plus, l'intersection entre cette tangente et la valeur finale du procédé donne $\tau_1 + \tau_2$. On a finalement $\tau_1 = 11ms$ et $\tau_2 = 78ms$ qui correspondent respectivement à des pulsations de coupure de $90.9\ rad/s$ et $12{,}8\ rad/s$.

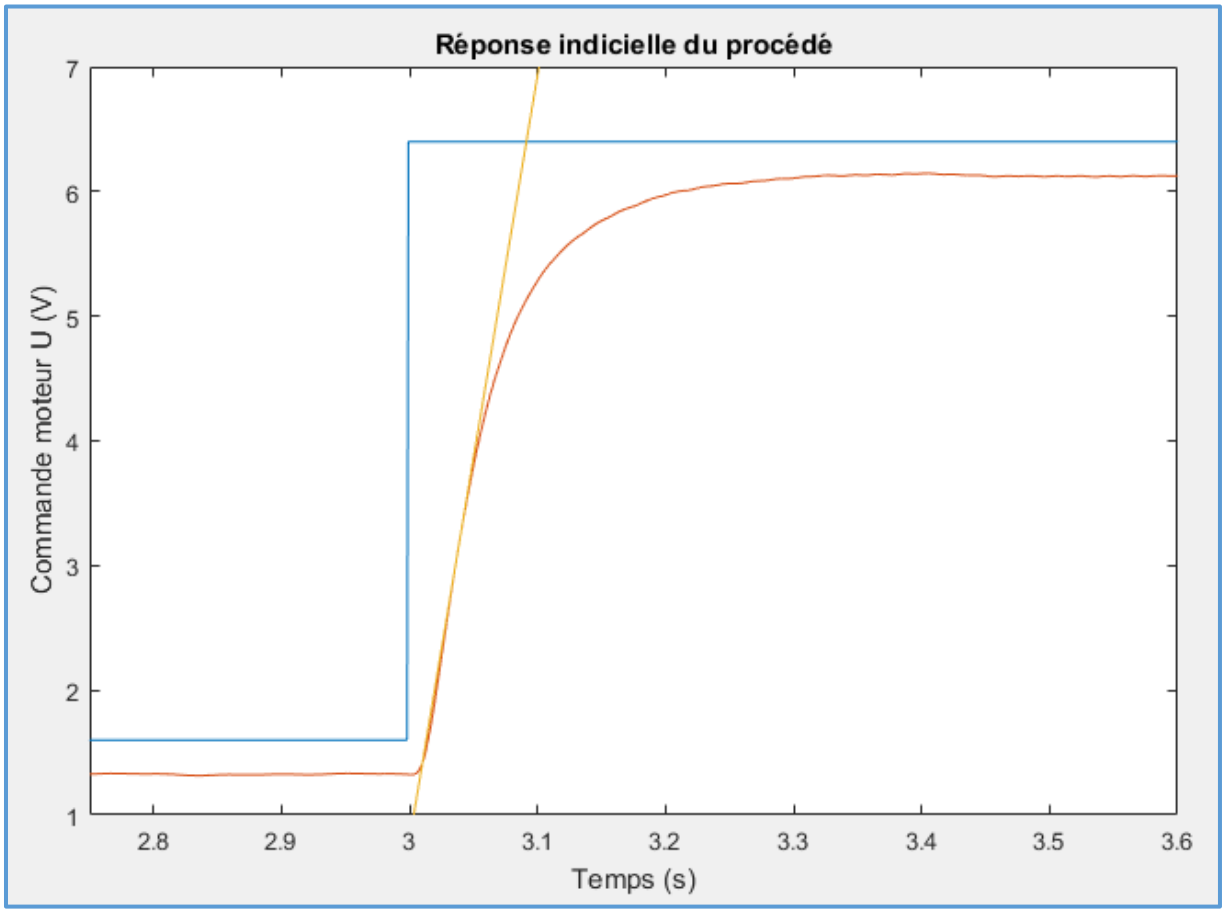


Figure 3 : Réponse indicielle du procédé

Finalement, le procédé est modélisé par la fonction de transfert :

$$G(p) = \frac{25.8532}{(1 + 0.011p)(1 + 0.078p)}$$

Calcul du premier correcteur

Principe

Le correcteur peut également être représenté par une fonction de transfert qui représente la tension à appliquer au moteur en fonction de l'erreur qu'il y a entre la consigne de vitesse et la vitesse réelle du moteur. Cette fonction de transfert se base sur la connaissance du moteur (identification). Le concept est de choisir une pulsation à laquelle l'on compte effectuer la commande. Généralement, on prend cette pulsation plus grande que la dernière fréquence de coupure. On va alors fixer le gain à 1 à cette fréquence et on va étudier la phase du procédé.

De fait, on désire que le correcteur suivi du procédé soit stable et rapide selon un cahier des charges. Pour cela, on identifie le point de non stabilité : la pulsation à laquelle le gain de la fonction de transfert en Boucle Ouverte (FTBO = correcteur * procédé) vaut 0dB et sa marge de phase. C'est-à-dire la différence entre sa phase et -180°. De fait, plus on est proche de -180° plus on se rapproche de l'instabilité. On cherche alors avec le correcteur à décaler la réponse fréquentielle au-dessus de -180°, au moins pour les gains forts. C'est pour cela que l'on fixe généralement la marge de phase à 55° environ à la fréquence de travail : cela favorise une réponse rapide et une stabilité correcte.

Dans la partie qui suit, il sera montré comment on arrive à faire ces calculs à l'aide de Matlab.

Synthèse du régulateur avec Matlab

La première étape est de définir le procédé à réguler et les conditions que l'on souhaite remplir. Dans notre cas, un simple correcteur à actions proportionnelle et intégrale a été synthétisé. En effet, le système étant déjà très amorti (réponse indicielle d'un second ordre amorti), un apport de phase par une action dérivée n'est pas nécessaire. Le régulateur aura donc la forme suivante :

$$C(p) = C_0 \frac{1 + \frac{p}{\omega_i}}{\frac{p}{\omega_i}}$$

La pulsation au gain unité sera imposée à $20\ rad/s$ et sera la même que la pulsation de la partie intégrale du régulateur ω_i.

Le gain C_0 servira à ajuster la réponse du système en boucle ouverte afin d'assurer un gain nul à $20\ rad/s$. On a donc $K_0 C_0 \frac{\omega_i}{\omega_u} \sqrt{1 + (\frac{\omega_u}{\omega_i})^2} = 1$ soit $C_0 = \frac{\omega_u}{\omega_i} \frac{1}{K_0 \sqrt{1+(\frac{\omega_u}{\omega_i})^2}}$.

Finalement, le régulateur à actions proportionnelle et intégrale s'écrit :

$$C(p) = 0.0519 \frac{1 + \frac{p}{20}}{\frac{p}{20}}$$

Simulations

Afin de vérifier que le régulateur synthétisé respecte bien le cahier des charges qui a été fixé, différentes simulations sont effectuées dans les domaines temporel et fréquentiel.

Dans un premier temps la réponse indicielle du procédé identifié précédemment et ses diagrammes de Bode sont observés.

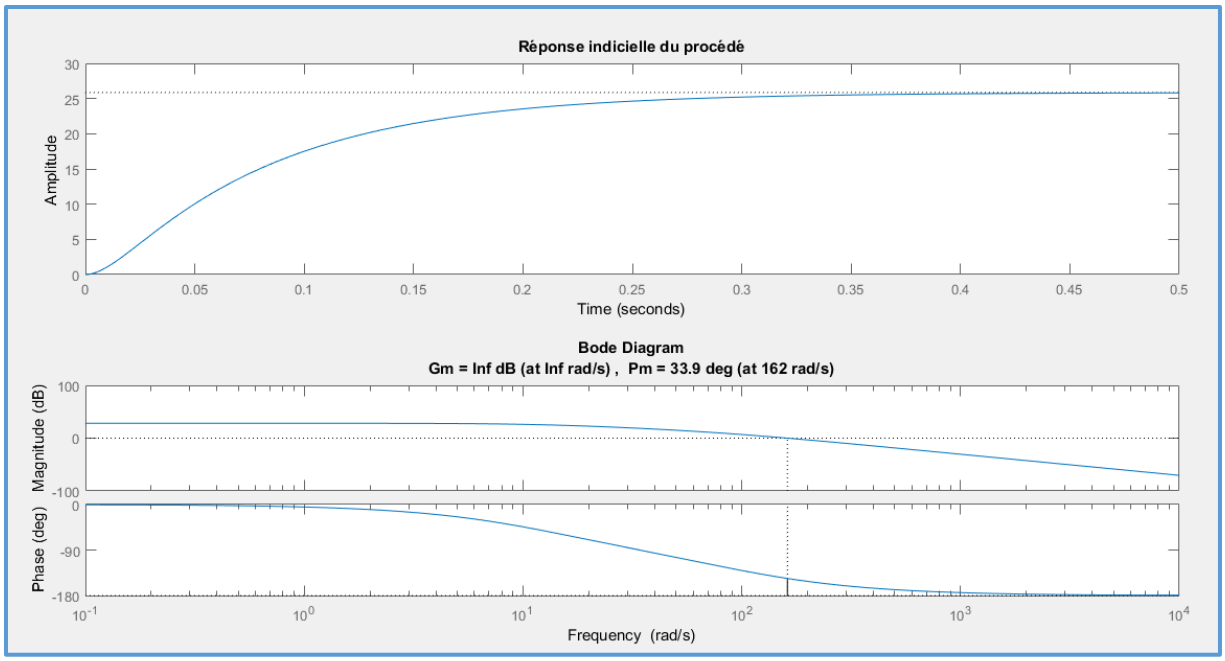


Figure 4 : Réponse indicielle et diagrammes de Bode du procédé

La simulation de la réponse indicielle du procédé permet d'effectuer une comparaison avec la courbe qui avait servi à l'identification. Globalement, cela permet de vérifier que le modèle qui a été identifié est proche de la réalité.
Puis, les tracés des diagrammes de Bode nous informent du comportement fréquentiel du modèle. On va chercher à améliorer la marge de phase afin d'augmenter le degré de stabilité du système en ajoutant un régulateur.

Il est aussi possible d'observer la réponse indicielle du régulateur et ses diagrammes de Bode.

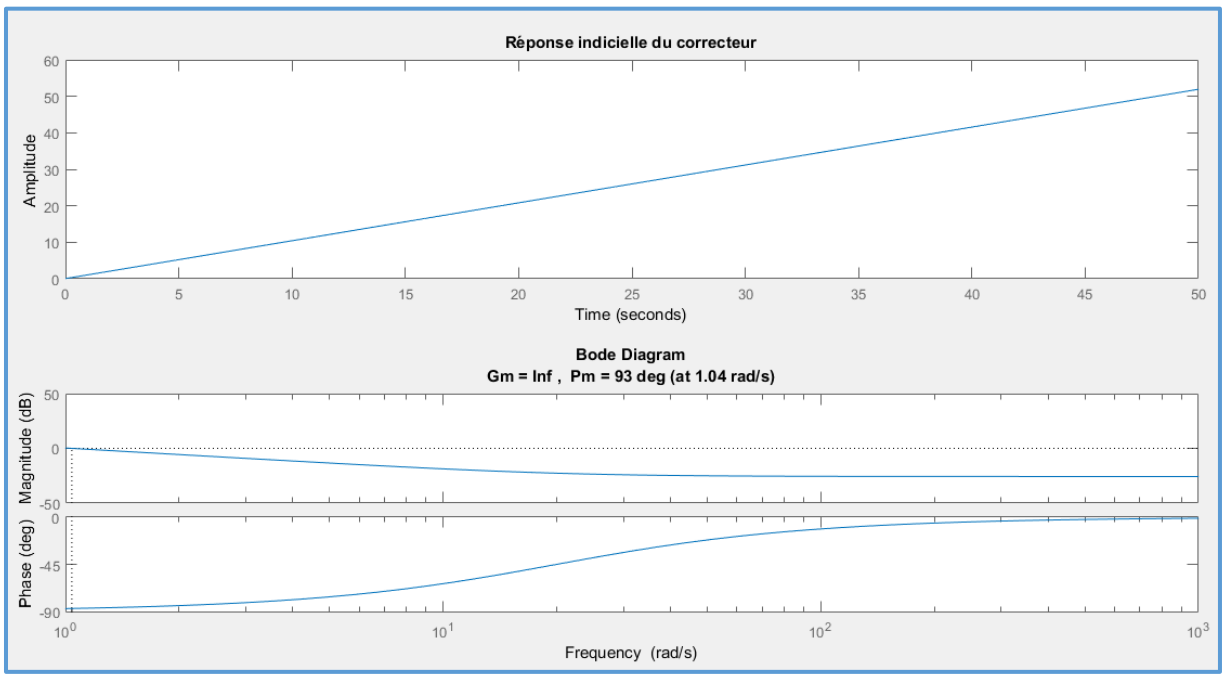


Figure 5 : Réponse indicielle et diagrammes de Bode du régulateur

Le régulateur étant à action proportionnelle et intégrale, une augmentation progressive (accumulation) est observée sur la réponse indicielle. Elle est due à l'action intégrale qui va permettre d'annuler l'erreur statique du moteur. La précision du système sera donc parfaite pour une consigne en échelon.

Tracer les diagrammes de Bode de la fonction de transfert en boucle ouverte permet de nous assurer que le cahier des charges qui a été fixé est bien respecté et donc que le régulateur ait été synthétisé correctement. En effet, comme le montre la figure suivante, la pulsation au gain unité est bien à $20\ rad/s$ et la marge de phase est de $65{,}3°$ ce qui assure une bonne stabilité du système.

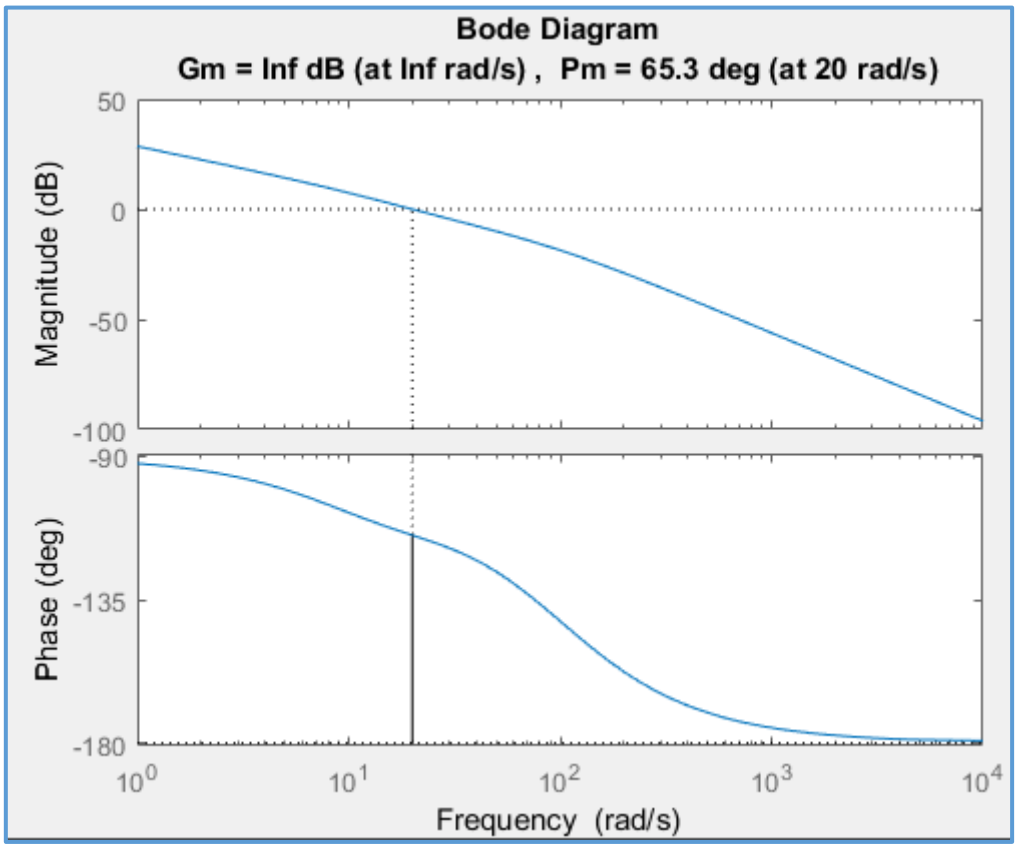


Figure 6 : Diagrammes de Bode de la fonction de transfert en boucle ouverte de l'asservissement

Il est maintenant intéressant d'observer la réponse indicielle de la fonction de transfert en boucle fermée, soit la réponse du procédé asservi, sur la figure suivante.
On y relève les caractéristiques suivantes :

-une absence d'erreur statique ;
-un dépassement de la valeur final inférieur à 10% ;
-un temps de réponse à 5% de $0.213s$;
-une constante de temps de $0.0526s$.

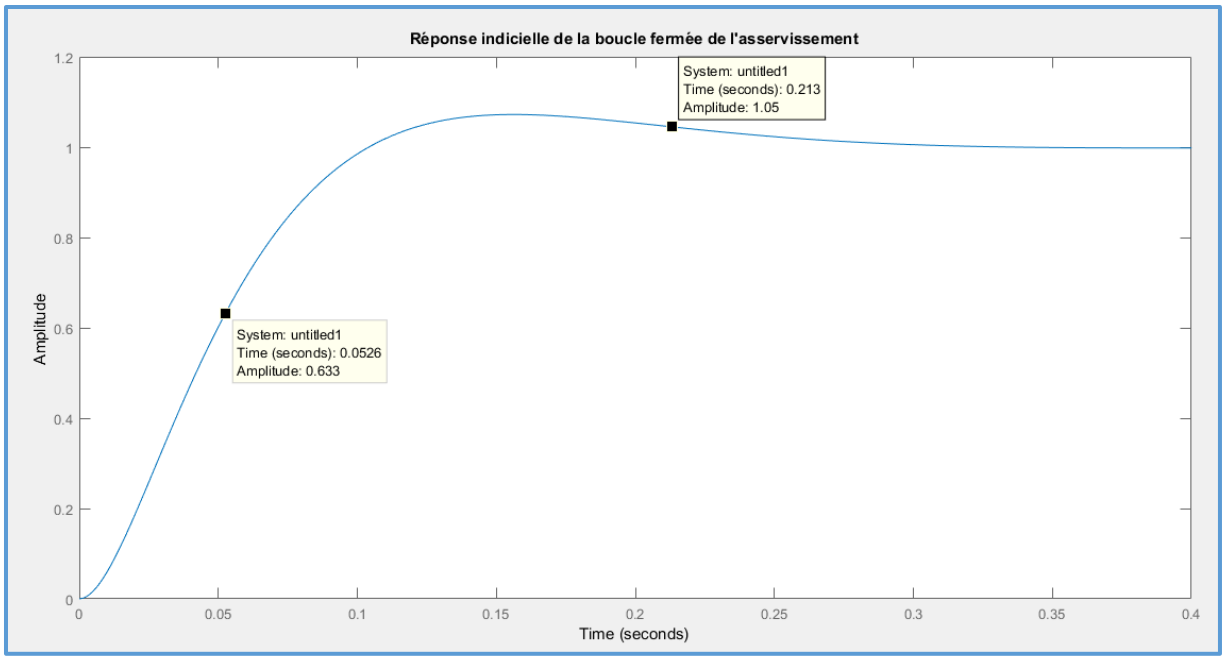


Figure 7 : Réponse indicielle de la fonction de transfert en boucle fermée de l'asservissement

Passage en numérique

La question que l'on peut se poser est comment implanter ce correcteur sur un processeur qui ne connait pas le domaine continu. De fait, cela ne semble pas naturel mais il existe plusieurs moyens : l'approximation en Delta et la transformée bilinéaire. Cependant les deux sont basés à un moment ou à un autre sur la transformé en z. L'approximation en Delta est valable uniquement si la fréquence d'échantillonnage est suffisamment grande devant les autres fréquences du procédé (x10 minimum). Alors il suffit de remplacer la variable p de la fonction de transfert par :

$$p \rightarrow \frac{1-z^{-1}}{T_e}$$, avec Te la période d'échantillonnage

La période d'échantillonnage est très importante à prendre en compte. Premièrement elle doit être plus grande que la fréquence de travail. Il faut au moins respecter le théorème de Shannon (x2 la fréquence de travail) mais en pratique, il est conseillé de la prendre au moins à x10. Cela permet notamment d'avoir une image du signal mesuré pertinente vis-à-vis de la dynamique du système Dans notre cas, l'échantillonnage est à 1kHz. Cela correspond à la fréquence la plus limitante de tout notre système (capteur inclus). De fait, le capteur, une fois traitée numériquement, renvoi une valeur de vitesse instantanée viable toutes les millisecondes. Généralement, cette fréquence tourne entre quelques Hz et 1kHz pour contrôler un robot.

Une fois la transformation faite (la fonction s'appelle *c2d*() sur Matlab en prenant comme paramètre '*zoh*'), on obtient une fonction de transfert en fonction de z qu'il faut passer en fonction de transfert en fonction de la variable z^{-1}, correspondant en fait à l'opérateur retard. Une fois la fonction de transfert obtenu, on exécute le raisonnement suivant.

La fonction de transfert du correcteur étant $\frac{U(p)}{\varepsilon(p)}$ avec *U(p)* la commande et *ε(p)* l'erreur, en considérant les coefficients précédents on obtient le passage suivant :

$$\frac{U(p)}{\varepsilon(p)} = \frac{\alpha_1 + \alpha_2 z^{-1} + \alpha_3 z^{-2} \dots}{1 + \beta_1 z^{-1} + \beta_2 z^{-2} \dots} \rightarrow u[n] = -\beta_1 * u[n-1] - \beta_2 * [n-2] + \alpha_1 * \varepsilon[n] + \alpha_2 * \varepsilon[n-1] + \alpha_3 * \varepsilon[n-2]$$

n représente l'instant actuel et n-1 l'instant précédent, etc… Ainsi on voit que la commande à chaque instant est déterminée par une combinaison linéaire des commandes précédentes et des erreurs actuelles et précédentes. C'est cette équation qui est implémentée dans le processeur et qui est itérée à la fréquence d'échantillonnage. Il faut donc à chaque instant garder en mémoire les commandes et les erreurs dans des FIFO de la taille du nombre de coefficients présents. Dans la partie suivante, nous allons voir comment nous avons implémenté cela dans le firmware développé par Rhoban.

Utilisation du Software et implémentation du correcteur

Le firmware par Rhoban étant assez étendu est complexe, pour notre partie il n'était pas nécessaire de tout comprendre. Nous nous sommes donc penchés sur les fichiers « servo.h », « servo.cpp », « motor.h » et « motor.cpp ». Dans ces fichiers il y a, entre autres, la réception de l'encodeur magnétique et le drive du pont en H qui alimente le moteur.

La version de servo.h est disponible sur le git associé. Les changements que nous avons effectués sont les suivants :

- ➔ Description d'une classe FIFO qui permettrait la mémorisation des commandes et des erreurs précédentes
- ➔ Implémentation des coefficients calculés
- ➔ Implémentation du debug afin de pouvoir visualiser les données à l'aide de gnuplot
- ➔ Implémentation d'une commande sur le terminal permettant d'envoyer un échelon de consigne.

Tout est assez clair sur le fichier. L'asservissement se déroule dans la condition « if servo_enable = true ; » et les paramètres sont situés surtout en haut du fichier.

Ci-dessous est présentée la démarche à suivre pour implémenter un nouveau firmware sur les brushless de Rhoban :

Dans un premier temps, cloner le git de Rhoban (s'ils vous l'autorisent évidemment). Ensuite, se placer dans le fichier brushless\firmware\. Ici se situent tous les fichiers présents sur le driver du moteur. Il suffit alors de modifier ce que l'on désire puis d'exécuter les commande « make » puis « make install » pour flasher le driver. Ensuite pour se connecter à la carte il faut taper la commande suivante : « cu -l \ttyACM0 » dans le terminal (Tout ceci sous Linux évidemment). Une fois connecté, voici les différentes commandes disponibles :

- ➔ pwm <valeur> : applique une pwm souhaitée (de 0 à 3000) au moteur sans passer par l'asservissement.
- ➔ set <valeur> : applique une consigne en tour/s à la boucle d'asservissement.
- ➔ step <valeur> : permet de donner une nouvelle consigne (tour/s) sans nettoyer les FIFOs de mémoires.
- ➔ fp=<phase> : force la pwm fixée dans uniquement une phase (allant de 0 à 5).
- ➔ sdb=1 : lance le programme d'identification définie dans servo.h. C'est une suite d'échelon avec visualisation de la vitesse et de la consigne.
- ➔ hall : visualise ce que renvoie le capteur à effet hall du brushless (codage de Gray).
- ➔ speed : visualise la vitesse instantanée du moteur.
- ➔ Et bien d'autres, il suffit de taper « help » pour avoir une liste plus complète.

Dans une future mise à jour, ce tutoriel sera un peu plus développé et générique.

Voici finalement les performances que nous avons obtenues sur le moteur avec la roue uniquement :

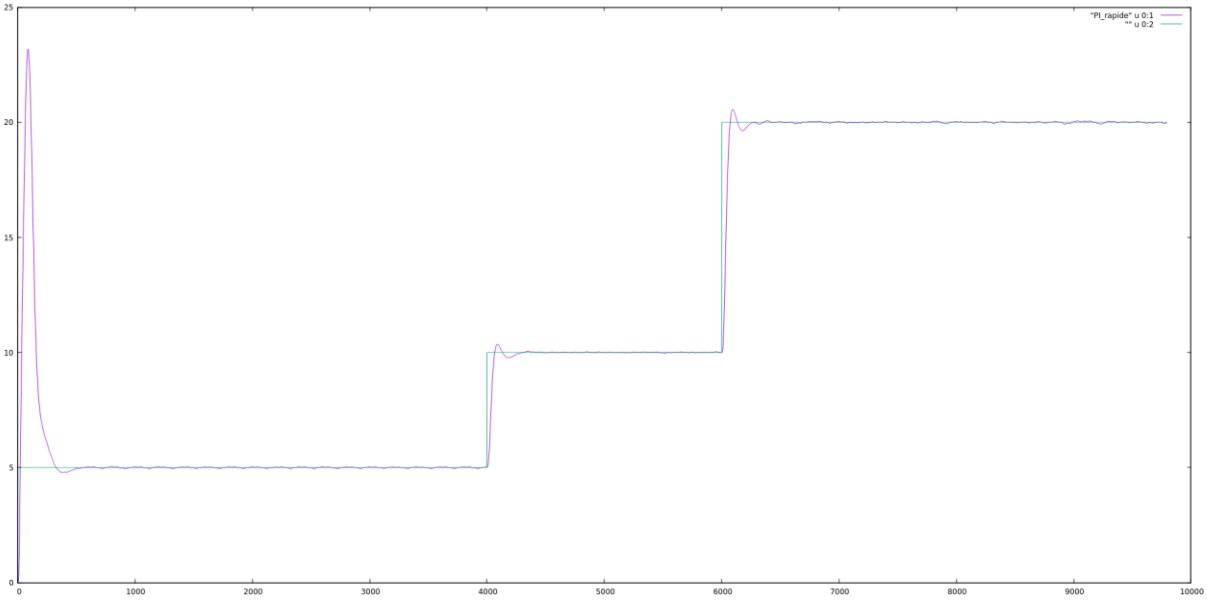


Figure 8 : Réponse du moteur à des échelons de 10 tours/secondes puis 20 tours/secondes

On observe que le moteur atteint la vitesse demandée en 0.2 secondes avec un léger dépassement. Le résultat est très satisfaisant. On en conclut que la chaîne de calcul et indentification est validée. Nous pouvons passer à l'étape d'au-dessus.

Conception du second banc de test

Toujours avec l'objectif de s'approcher au maximum de la situation dans laquelle les moteurs vont fonctionner, un second banc de test a été réalisé. Les mêmes méthodes ont été utilisées que lors de la conception du premier banc de test. L'utilisation de la découpeuse laser permet en effet un prototypage rapide.

Ce banc de test aura pour objectif principal de permettre mettre en œuvre le comportement d'un moteur avec roue qui touche le sol. Des tests en conditions presque réelles sont souhaités. Pour cela, la structure du banc ne sera pas statique comme précédemment mais dynamique. Le moteur entraînera son support avec lui autour d'un axe vertical. Une charge pourra être rajoutée sur l'ensemble et représentera l'effet d'un quart du poids du robot. On suppose que les charges seront uniformément réparties sur les 4 moteurs en situation réelle. Il devient donc possible de prendre en compte les inerties relatives à la masse du robot.

En plus de pouvoir observer le comportement du moteur lorsqu'il est en mouvement, ce banc nous permet d'effectuer les mesures lors du démarrage du moteur sur le même sol qui recouvre les terrains. En effet, il ne faut pas négliger les frottements de la roue sur le sol lors de l'étude : sans frottement, il n'y a pas de déplacement. Cela met aussi en évidence le problème de glissement des roues. Le patinage des roues sur le terrain peut s'avérer problématique. L'utilisation du banc peut donc permettre de s'assurer de la plage de commande où il n'y aura pas de glissement des roues.

Un problème persiste toutefois : les roues étant holonomes, elles permettent aussi des déplacements selon les axes de rotation des moteurs. Mettre en place un banc simple qui permet d'effectuer des tests sur ces mouvements n'est pas une mince affaire. On ne les considérera donc pas ici mais des tests sur le fonctionnement d'un robot complet peuvent combler ce manque.

Pour aller plus loin

Deuxième identification du moteur

La prochaine étape serait de continuer sur le second banc moteur et d'effectuer une nouvelle identification du procédé. En effet, sur le premier banc, l'absence de charge et de frottements fait que le système était certainement plus rapide. Il serait intéressant de comparer les deux comportements du moteur (à vide et chargé).

Synthèse et implantation d'un second correcteur

Si nous avions eu le temps, nous aurions synthétisé et implémenté le dernier correcteur avec le deuxième banc moteur afin de voir la différence d'efficacité sur le déplacement global. Il aurait été intéressant d'observer et de quantifier cette amélioration.

Etude mécanique et dynamique

Rhoban avait déjà un robot complet avec un correcteur calculé empiriquement. Ils ont déjà effectué l'étude dynamique consistant à envoyer une commande de vitesse à chaque roue considérant la vitesse désirée du robot et sa rotation sur lui-même. Il est important d'effectuer cette étude si le désir est de concevoir un robot holonome.

Suppression des « saut » dû aux roues holonomes

Comme il est possible de le voir sur les photos, une roue holonome est assez particulière. Celle-ci surtout possèdes des roues radiales assez longue et donc aux basses vitesses, le robot tend à vibrer car il passe d'une roue radiale à l'autre. L'idée de cette amélioration serait d'anticiper ces passages là en fonction de la position de la roue. Il faudrait donc superposer à l'asservissement de base une compensation qui rendrait plus fluide le déplacement.

Conclusion

Ce projet a été très enrichissant sur plusieurs aspects. Il nous a permis de mettre en pratique chaque semaine ce que nous avions appris la semaine d'avant en automatique ou en programmation. Cela nous a permis d'approfondir également nos connaissances en robotique et de voir les problématiques liées à l'implémentation numérique de correcteurs. Il est à noter que notre stage de deuxième année sera dans la continuité de ce travail car nous l'effectuerons au sein de l'IMS en collaboration avec Rhoban.

GLOSSAIRE

Consigne : *vitesse souhaitée du moteur.*

Commande : *tension réelle envoyée au moteur.*

Dépassement *: dépassement de la vitesse du moteur par rapport à sa valeur en régime permanent (valeur finale).*

Direct Driving *: les moteurs déplacent le robot sans passer par une réduction mécanique.*

Erreur statique *: erreur de vitesse entre celle souhaitée et celle atteinte.*

Gain statique *: coefficient entre la valeur atteinte par le moteur en régime permanent et la commande qui lui a été appliquée.*

Perturbation *: événement imprévu agissant sur le moteur (bruit de mesure, frottements supplémentaires, etc..).*

Procédé *: correspond dans notre cas au moteur et à toute la chaîne de traitement analogique et numérique. C'est la boite noire que l'on cherche à contrôler.*

Pseudo-pulsation *: pulsation liée aux dépassements successifs de la vitesse autour de sa valeur finale.*

Temps de montée *: temps pris par le moteur pour aller de 10% à 90% de sa vitesse finale.*

Bibliographie

• "*PID control of brushless DC motor and robot trajectory planning and simulation with Matlab/Simulink*" --- **Oludayo John Oguntoyinbo** – 2009

• "*Dynamical Models for Omni-directional Robots with 3 and 4 Wheels"* – **Helder P. Oliveira, Armando Jorge Sousa, A. Paulo Moreira, Paulo José Cerqueira Gomes da Costa** – 2008

• Datasheet of Maxon EC45 70W flat