MASTER OF SCIENCE IN ENGINEERING

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Master of Science HES-SO in Engineering
Av. de Provence 6
CH-1007 Lausanne

Master of Science HES-SO in Engineering

Orientation: Technologies industrielles (TIN)

Modélisation et simulation en boucle fermée des systèmes Beam Wire Scanner du CERN à moteur haute performance basé sur la commande vectorielle par FPGA

Proposé par:
Jonathan Emery
CERN

Fait par:
Arnaud Yersin

Sous la direction de:
Prof. Yann Thoma
REDS, HEIG-VD

Expert externe:
Stéphane Humbert
CSEM SA

Lausanne, HES-SO//Master, 2023

Remerciements

Je voudrais remercier M. Yann Thoma qui a supervisé le projet. Son suivi régulier et ses conseils m'ont permis de résoudre certains problèmes (notamment liés aux logiciels de simulation). Ses suggestions pour la réalisation du testbench m'ont également été très utiles.

Je remercie également M. Jonathan Emery (CERN) qui m'a donné toutes les informations nécessaires concernant le système étudié. Il m'a notamment permis d'effectuer un stage à mi-temps durant trois mois au CERN. Sa disponibilité et ses réponses à mes nombreuses questions m'ont permis d'avancer lorsqu'il a fallu revoir le modèle du système à régler.

Je voudrais aussi remercier les professeurs Raoul Herzog et Christophe Besson qui se sont rendus disponible pour répondre à des questions.

Résumé

Le projet consiste à mettre en place un testbench permettant de vérifier un régulateur utilisé dans les systèmes « Beam Wire Scanner » du CERN. Ce régulateur codé en VHDL est simulé de manière fonctionelle en boucle fermée avec un modèle du système à régler (modèle compatible avec les outils de simulation VHDL). Un régulateur de référence est également modélisé pour permettre de comparer les résultats et de juger si le design est correct ou non. Plusieurs testcases pour plusieurs cas différents sont implémentés et un rapport automatique est généré pour observer les résultats des simulations.

La première section **Introduction** (1) présente le projet avec le contexte, l'énoncé et une liste détaillée des tâches à effectuer pour ce travail.

Dans la seconde section **Modélisation du système à régler** (2), il y a une analyse détaillée du système à régler. Le modèle analogique sur Matlab/Simulink a pu être amélioré et un système discret a été modélisé en SystemVerilog pour les besoins du testbench.

La troisième section **Régulateur** (3) présente le régulateur idéal. À partir d'un modèle Matlab/Simulink, un modèle (de référence) a pu être modélisé en SystemVerilog. les différences (liées aux contraintes d'implémentation) avec le régulateur à vérifier et le modèle de référence ont également été étudiées.

La quatrième section **Testbench** (4) explique la structure du testbench, la manière dont sont exploités les résultats et les différents aspects logiciels. Des résultats plus ou moins satisfaisants sont présentés et analysés.

Enfin, la dernière section **Conclusion** (5) résume les résultats obtenus et la manière dont s'est passé ce projet. Enfin, un avis personnel est donné sur le déroulement de ce travail.

Dans ce rapport, l'analyse du système à régler (surtout sur la partie moteur) est volontairement très détaillée. La raison est que dans les travaux précédents, de nombreuses incohérences étaient présentes dans les modèles. Il a été nécessaire de revoir la quasi-totalité du modèle du moteur pour obtenir une cohérence entre les deux représentations du modèle (triphasée et "dq") et la théorie.

Contents

1 Introduction

1.1 Contexte

Ce sujet est proposé dans le cadre du projet Beam Wire Scanner (BWS) pour le Large Hadron Collider (LHC) par le groupe d'instrumentation des faisceaux de l'organisation européenne pour la recherche en physique des particules (CERN) qui est responsable de développer, construire, qualifier et maintenir des instruments permettant de vérifier et optimiser la qualité des faisceaux de particules. Une quarantaine de systèmes ont été construits et sont en phase d'installation dans les accélérateurs du CERN à Genève et dans la source de neutrons internationale ESS à Malmö (Suède). Ce projet concerne plus particulièrement le système de contrôle de trajectoire haute performance par FPGA qui va être adapté à un nouveau type de mécanisme.

1.2 Rappel de l'énoncé

Dans le cadre de l'instrument « Beam Wire scanner » pour le Large Hadron Collider (LHC), et dans l'optique d'amélioration continue de son système de contrôle par FPGA, le projet propose de modéliser puis simuler le comportement de la partie mobile de cet instrument à l'aide d'un langage permettant une intégration aisée et peu coûteuse avec les outils de simulation Questa. Il pourra s'agir de description matériel comme VHDL et SystemVerilog, ou un autre langage en fonction d'une pré-étude.

La simulation fonctionnelle de contrôleur par FPGA au niveau RTL « Register Transfer Level » en boucle fermée nécessite la modélisation des systèmes physiques à régler pour en observer leur comportement en situation quasi réelle. Cette approche doit permettre de travailler dans le même environnement logiciel que la vérification RTL, et d'ainsi effectuer parallèlement la validation fonctionnelle de la régulation. Il sera dès lors possible d'expérimenter des architectures ou des détails de réalisation de système de contrôle de manière itérative assistée par ordinateur (trial and error, optimisation, etc.). Il sera par exemple possible d'observer l'effet de la précision des opérations mathématiques sur la précision du réglage ou encore le comportement d'une architecture de contrôleur particulière dans divers scénarios de fonctionnements, lors de variation de l'environnement ou lors de panne du système à régler.

L'approche proposée dans ce projet est de décrire le système à régler avec son comportement électrique et mécanique dans un langage compréhensible par les outils de simulation RTL (ou pouvant s'y interfacer facilement). Le modèle du système physique étant directement décrit en VHDL ou SystemVerilog, il permettra une grande flexibilité de simulation devenant compatible de fait avec les outils de développement/simulation utilisés pour la description du contrôleur. Cela ouvre également la possibilité d'utiliser différents outils et technologies du domaine de la simulation digitale pour le contrôle de système physique.

Pour la description du système physique dans le cadre de ce projet, l'étudiant pourra s'appuyer sur un modèle existant Simulink utilisé dans le passé pour de la co-simulation [1]. Cette approche utilise deux logiciels en parallèle et nécessite une mise en place ainsi qu'une maintenance durant le cycle du produit qui s'avère peu flexible et coûteuse. Pour le contrôleur digital, il existe une implémentation qui pourra être utilisée comme base expérimentale. A noter que la génération de la trajectoire du système est actuellement calculée offline et chargée en mémoire lors du boot de la FPGA. Pour ce projet, il sera intéressant d'améliorer cette partie en intégrant l'algorithme [2] directement dans la FPGA pour permettre une génération à la demande en fonction de critères d'entrées, comme la vitesse à atteindre, la position finale, etc.

Le concept développé sera utilisé pour plusieurs systèmes physiques et algorithmes de contrôle. Actuellement, il y a deux systèmes physiques assez différents à modéliser pour cette étude.

1.3 Travail à effectuer

Selon l'énoncé, le travail à effectuer pour ce projet comprend l'étude du système complet, la modélisation du système à régler et la mise en place du testbench. Si le temps le permet, la génération de la trajectoire directement dans la FPGA peut également être réalisée. Après une réflexion plus approfondie, une liste plus détaillée des tâches à effectuer est établie ci-dessous.

Étude du système à régler
Dans la perspective de devoir re-créer un nouveau modèle du système à régler pour le testbench, il est nécessaire d'étudier, de comprendre et, si besoin, de re-modéliser les différents éléments qui composent ce système à régler. La documentation des travaux précédents et les simulations (Matlab) à disposition permettent de réaliser ce travail d'analyse.

Étude du régulateur (théorique)
Ce projet a pour but de mettre en place un testbench permettant de vérifier le bon fontionnement du régulateur VHDL. Celui-ci est relativement complexe, il est possible qu'il comporte des erreurs et surtout, il s'agit d'un design à tester (pas encore validé) qui ne doit en aucun cas être pris comme une référence. Il est donc intéressant de se familiariser avec un modèle simplifié du régulateur (sur Matlab) afin d'étudier le comportement idéal du design à vérifier et d'avoir une référence.

Étude de la simulation Matlab existante
Le régulateur et le système à régler intéragissent ensemble de manière à optimiser la trajectoire du système à régler. Souvent proche de l'instabilité, un faible écart de paramètre dans l'une des deux entités peut entraîner une instabilité dans la boucle de réglage. Il est donc important de comprendre l'aspect "régulation" de ce projet afin de faciliter les phases de "debug" lorsque le système est instable.

Étude du régulateur à vérifier (VHDL)
Le régulateur à vérifier (VHDL) est celui qui est implémenté dans la FPGA sur le système réel. L'implémentation dans un système réel impose un certain nombre de contraintes qui n'apparaissent pas dans une simulation idéale. Le but de cette étape n'est pas d'étudier en détail tous les aspects du régulateur à vérifier mais de comprendre quelles sont les contraintes et les différences par rapport au régulateur de référence qui est idéal.

Modélisation du système à régler (compatible avec les outils de simulation HDL)
Puisque le régulateur à vérifier est décrit en VHDL, le testbench nécessite un logiciel de simulation VHDL. Le régulateur étant interfacé avec le système à régler, ce dernier doit donc également être décrit dans un language compatible VHDL ou SystemVerilog.

Interfaçage entre le régulateur à vérifier et le modèle du système à régler
Le régulateur VHDL contient un certain nombre d'entrées/sorties de types prédéfinis. Il est important de connaître les conventions, les grandeurs et les unités des entrées/sorties afin d'interfacer correctement les deux entités et si besoin, d'effectuer les conversions.

Mise en place de la structure du testbench

En plus de connecter le régulateur au système à régler, il faut gérer l'environnement autour de la boucle fermée. La vérification du régulateur nécessite la génération de stimulis, l'observation des signaux de sorties et la détection de fin de simulation. L'environnement du testbench à mettre en place doit être suffisamment complet pour implémenter les scénarios voulus et en ajouter. Si besoin, il doit aussi pouvoir être modifié sans trop de difficulté.

Choix des scénarios

La vérification du régulateur ne peut pas être exhaustive. Il faut établir une liste de scénarios et stimuler le design à vérifier en fonction de ceux-ci. Les scénarios doivent être choisis avant de lancer la simulation et il est important de pouvoir les modifier ou en ajouter des nouveaux facilement.

Acquisition des données

Après la simulation du testbench, certains signaux doivent être analysés. L'observation peut se faire avec l'interface graphique du logiciel de simulation mais celui-ci se limite à analyser les résultats d'une seule simulation à la fois et il est difficile de comparer précisement plusieurs courbes. Afin d'analyser toutes les données souhaitées après la simulation, les signaux peuvent être enregistrés dans des fichiers durant la simulation. Le format de ces fichiers doit être défini de manière à pouvoir exploiter les données facilement.

Exploitation des résultats

Après la simulation, les données enregistrées doivent être analysées en observant les courbes sur des graphiques, en calculant des écarts entre deux courbes ou simplement en effectuant des calculs. Ce traitement post-simulation doit permettre de juger si la vérification du design à tester est un succès ou non.

Génération de la trajectoire

Avec le régulateur actuel, les trajectoires sont pré-calculées et chargées en mémoire. Les trajectoires dépendent de paramètres physiques du système à régler et de certaines grandeurs qui doivent être choisies telles que la vitesse de rotation et la durée du mouvement. Si le temps le permet, ces calculs pourraient très bien être réalisés directement dans le régulateur intégré dans la FPGA.

Le planning initial a été réalisé aux environs de la deuxième semaine mais celui-ci a dû être réadapté régulièrement à cause des retards du projets. L'annexe A montre le planing initial et les tâches réalisées pour chaque semaine. L'analyse et la modélisation du système à régler a pris bien plus de temps que prévu. Cela a retardé les autres tâches et comme l'indique le planing, la génération de la trajectoire n'a pas pu être effectuée.

2 Système à régler

L'étude du système à régler réel est une étape essentielle de ce travail. Le régulateur à vérifier est conçu de manière à interagir avec l'installation électromécanique située dans l'accélérateur de particules et de ce fait, la simulation (testbench du régulateur) nécessite un modèle numérique précis de ce système à régler.

La section 2.1 présente brièvement le principe de fonctionnement et les différents éléments de l'installation. La modélisation du système à régler est étudiée à la section 2.2. Enfin, le modèle du système à régler est discrétisé dans la section 2.3.

2.1 Présentation du système à régler

Le système étudié dans ce travail s'appelle le « Beam Wire Scanner » (BWS). Le but de cette installation est de mesurer le profil du faisceau dans des accélérateurs de particules. Il existe plusieurs versions de ce système. La version utilisée dans ce travail et décrite ci-dessous est celle du « Rotary wire scanner ».

La mesure s'effectue à l'aide d'un fil de carbone (d'environ 30 $[\mu m]$ de diamètre) qui traverse le faisceau de particules. L'interaction entre le fil de carbone et le faisceau envoie des particules en direction d'un capteur (« Scintillator » sur la figure 2.1).

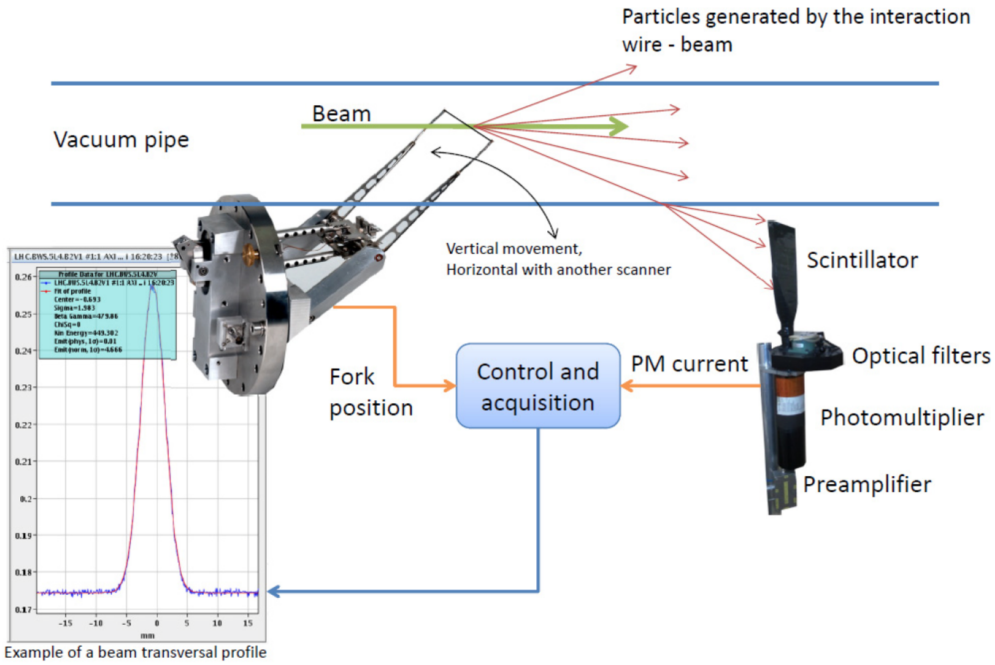


Figure 2.1: Principe de fonctionnement du « Beam Wire Scanner »

Le mouvement du fil qui traverse le faisceau est effectué à l'aide d'un moteur triphasé. Dans le cas de l'installation du « Rotary wire scanner », Le moteur effectue un demi-tour (de haut en bas) pour le mouvement « in ». Un deuxième passage est effectué (de bas en haut) pour le mouvement « out » en revenant à sa position initiale.

La mesure exige que la vitesse et la position du fil de carbone soient contrôlées précisément. En fonction des propriétés mécaniques et électriques du système, des trajectoires ont été calculées pour estimer la position, la vitesse et le courant électrique durant le mouvement du moteur. Le système réel ne pouvant pas être parfaitement modélisé avec tous ses paramètres exacts (bruit aléatoire, changement de température, imprécisions dans la mesure des propriétés physiques), la trajectoire réelle du moteur peut être optimisée avec un régulateur afin de corriger les éventuels écarts.

Pour corriger les erreurs de position, de vitesse et de courant électrique, le régulateur a besoin de comparer les trajectoires pré-calculées avec les mesures de ces grandeurs réelles. Dans ce travail, le système à régler comprend tous les éléments physiques entre la sortie du régulateur (« Command ») et le retour des informations mesurées (« Feedback »). Dans le but de faciliter la compréhension de ce système lors de l'analyse et surtout lors de la modélisation, il peut être décomposé en plusieurs blocs individuels.

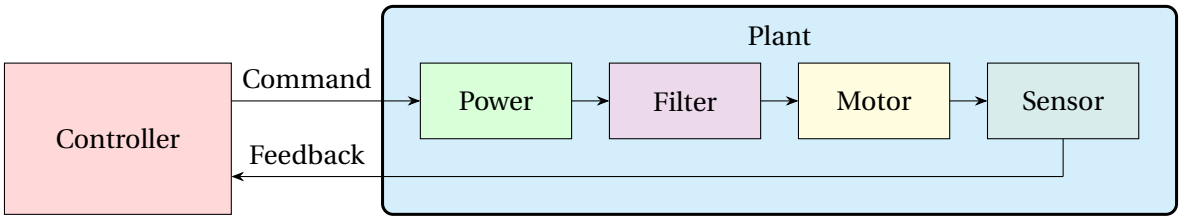


Figure 2.2: Décomposition du système à régler (« plant »)

2.1.1 Bloc « Power »

Comme le montre la figure 2.2, c'est le bloc « Power » qui est directement commandé par le régulateur. Cette commande est constituée de six signaux binaires (deux pour chaque phase) modulés sous forme de PWM (« Pulse With Modulation »). Les signaux de commande numérique n'apportent aucune puissance significative au système à régler. C'est le rôle de ce bloc « Power » de générer des signaux de puissance permettant d'actionner le moteur à partir de la commande numérique du régulateur. Ceci se fait avec un onduleur (« power inverter ») qui permet de générer des signaux de tensions (avec une puissance significative) à partir d'une tension d'alimentation continue et de transistors commandés.

2.1.2 Bloc « Filter »

Ici, le bloc « Filter » fait le lien entre l'étage de puissance et le moteur. Dans le système réel, les installations électroniques (régulateur et onduleur) sont à la surface alors que l'installation mécanique (le moteur) est placée dans l'accélérateur de particules sous terre. La distance qui sépare ces installations peut varier entre 55 et 232 $[m]$. Elles sont donc reliées par des câbles électriques triphasés. Ces câbles ont un impact sur les signaux de tension lorsque ceux-ci sont formés de PWM avec des composantes hautes fréquences. En effet, les câbles sont sujets aux effets de « cross-talk » qui induisent un comportement indésirable de filtre passe-bas. Le problème étant que ces effets dépendent de la longeur des câbles et donc de l'emplacement de l'installation.

Afin de remédier à ces effets non désirés, un filtre a été mis en place juste après l'onduleur et donc juste avant les câbles. Il s'agit du filtre Schaffner FN 5045-10-44 de type « Sine wave filter ». Il permet de filtrer le signal PWM de manière à obtenir un signal de sortie sinusoïdal. Ainsi, les signaux circulant dans les câbles contiennent moins de hautes fréquences et les effets non désirés de « cross-talk » sont réduits.

2.1.3 Bloc « Motor »

Le bloc « Motor » correspond à toute la partie électromécanique de l'installation. Celle-ci se situe dans l'accélérateur de particules. La figure 2.3 montre les différents éléments de cette installation.

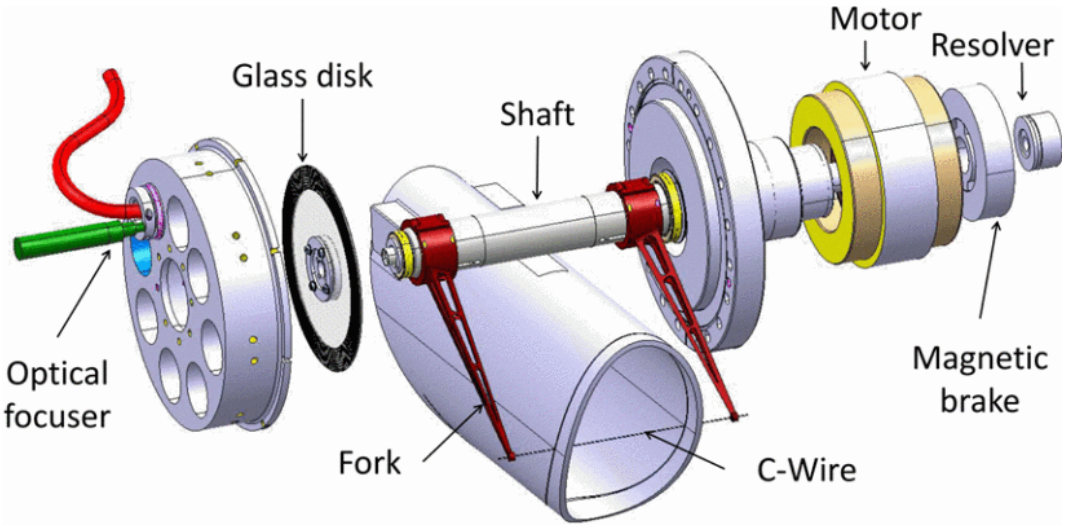


Figure 2.3: [1] L'installation électromécanique du « Rotary Wire Scanner »

Le moteur utilisé est un moteur de type synchrone à aimants permanents aussi appelé PMSM (« Permanent Magnet Synchronous Motor »). Le modèle est une version modifiée du Alxion 145ST2M (1500 rpm). Il s'agit de la variante 145ST2M023 dont les caractéristiques seront vues plus en détail lors de la modélisation. La charge mécanique est composée d'une fourche (« Fork ») qui permet de tenir le fil de carbone. L'accélérateur de particules fonctionne sous vide et donc, la partie mécanique avec la fourche se trouve également sous vide.

Lorsque le moteur est à l'arrêt, la régulation n'est pas active. Pour que le rotor ne bouge pas lorsqu'il est à l'arrêt, le moteur est équipé d'un frein magnétique (« Magntic brake »). Celui-ci exerce un couple magnétique de manière à replacer le rotor à la position souhaitée.

2.1.4 Bloc « Sensor »

La mesure de la position et de la vitesse s'effectue à l'aide d'un resolver. Celui-ci est positionné sur le moteur (voir figure 2.3) pour détecter la position du rotor en fonction de son champ magnétique. Le resolver fournit deux signaux analogiques (déphasés de 90° entre eux). Pour calculer la position du rotor à partir de ces deux signaux, il faut utiliser un RDC (« Resolver-to-Digital Converter »). Le composant utilisé est le AD2S1210 (Analog Devices).

Pour la mesure de courant, le capteur utilisé est le LEM LA 100-P. Celui-ci permet de mesurer le courant circulant dans un câble et il est donc nécessaire d'en avoir plusieurs pour mesurer le courant des trois phases. En réalité, contrairement à ce que laisse penser le schéma de la figure 2.2, la mesure du courant ne s'effectue pas sous terre au niveau du moteur. Les capteurs de courant sont situés à la surface juste après le filtre FN5045 et donc avant les câbles. Si ces capteurs sont placés après le moteur sur la figure 2.2, c'est parce que du point de vue du modèle, le courant doit être mesuré après le moteur. Il faut cependant garder à l'esprit qu'ils sont en réalité placés à la surface avant le moteur.

2.2 Modélisation du système à régler

L'accessibilité au système réel étant limitée, il est indispensable d'avoir à disposition un modèle numérique fiable afin d'effectuer les étapes de développement et de vérification du régulateur sans avoir besoin d'accéder au système réel. Le travail de modélisation a déjà été effectué auparavant ([3]) et le modèle est notamment utilisé dans des simulations Matlab/Simulink afin d'effectuer la synthèse du régulateur.

Plusieurs travaux sur la modélisation du système à régler ont été effectués précédemment et il existe plusieurs versions de la modélisation de ce système à régler. Une version du modèle est basée sur l'utilisation des outils Simscape de Simulink. Cette toolbox permet de construire et simuler un système physique avec différents composants (notamment électriques) qui se connectent entre eux comme un schéma électrique. Le niveau d'abstraction élevé permet une modélisation intuitive et rapide, sans se préoccuper des équations mathématiques qui régissent le comportement des composants. Avec Simscape, le système peut être entièrement décrit dans Simulink tout en gardant une bonne lisibilité et compréhension. Mais le défaut principal de ce type d'outils est le manque de portabilité, le système ne pouvant être simulé que dans Simulink avec une toolbox spécifique.

Une version plus traditionnelle, basée sur des fonctions de transfert, a également été modélisée. Dans cette seconde version, certaines parties du système ont été simplifiées mais tout en gardant une bonne représentation mathématique du système réel. Dans un premier temps, c'est cette version simplifiée qui est utilisée comme référence afin d'appréhender le système avec des équations simplifiées. Dans un second temps, dans le but d'avoir un modèle le plus proche possible du système réel, certains composants plus complexes peuvent venir compléter ce modèle.

Dans cette section (2.2), le modèle du système à régler est analysé dans le domaine continu. C'est-à-dire que le modèle utilise des fonctions de transfert continues et il ne peut donc être simulé qu'avec des outils numériques à pas de temps variable (comme Simulink).

2.2.1 Bloc « Power »

Comme expliqué à la section 2.1.1, le premier élément du système à régler est l'onduleur qui est commandé par les six signaux de commande venant du régulateur. Le schéma d'un onduleur triphasé typique est visible à la figure 2.4.

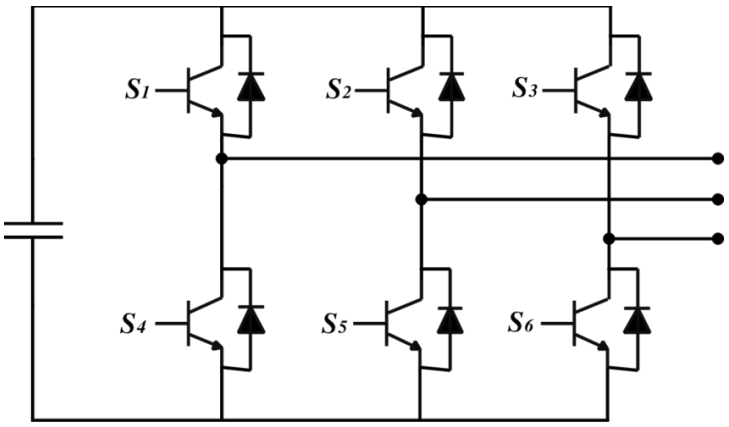


Figure 2.4: Schéma d'un onduleur triphasé
https://www.researchgate.net/figure/Three-phase-voltage-source-inverter_fig5_319391013

La figure 2.4 représente le schéma (au plus simple) d'un onduleur triphasé. Tout au long de ce travail, la tension continue qui alimente l'onduleur est appelée U_e et elle est fixée à 390 $[V]$. Pour chacune des trois phases, il y a deux signaux de commande associés (« up » et « down »). La tension de sortie de chacune des phases peut ainsi prendre les valeurs 0 V, U_e, ou haute impédance (utilisée lors des transitions). Le cas où les deux transistors d'une phase seraient conducteurs au même moment ne doit jamais se présenter.

Puisqu'il s'agit de signaux binaires, la commande est envoyée au système à régler sous forme de PWM et la valeur de la commande est donc proportionnelle au rapport cyclique (« duty cycle »). À la sortie de l'onduleur, ces signaux sont ensuite moyennés avec un filtre passe bas (section 2.2.2.1) et deviennent des signaux analogiques pouvant prendre n'importe quelle valeur de potentiel entre 0 V (rapport cyclique de 0 %) et U_e (rapport cyclique de 100 %).

2.2.1.1 PWM et fréquence d'échantillonnage

Pour un système à régler commandé par un régulateur numérique, la période d'échantillonnage correspond au temps entre chaque nouvelle donnée échangée. Dans le cas d'une commande par PWM, chaque valeur doit être transmise sur la durée d'un cycle de PWM et celle-ci correspond donc à la période d'échantillonnage. La précision (ou résolution) de cette valeur de commande dépend du rapport entre la fréquence d'horloge du régulateur et la fréquence d'échantillonnage (fréquence du PWM). Pour le système étudié ici, les fréquences sont les suivantes:

$F_{clk} = 40 \cdot 10^6\,[Hz]$
$F_{PWM} = 16129\,[Hz]$

Le rapport entre ces fréquences vaut donc:

$$\frac{F_{clk}}{F_{PWM}} = \frac{40 \cdot 10^6}{16129} = 2480.005 \tag{2.1}$$

Cela signifie qu'une donnée calculée par le régulateur est envoyée au système à régler sur une durée de 2480 coups d'horloge. En théorie avec un PWM idéal, la commande (pour une phase) peut ainsi prendre 2480 valeurs différentes correspondant à une résolution de $U_e/2480$. Ceci n'est pas tout à fait vrai en pratique car le PWM n'est pas idéal. Le cas où les deux transistors d'une phase sont activés au même moment doit absolument être évité. Afin de s'assurer que ce cas critique ne survienne jamais, un temps-mort (« dead-time ») est introduit durant le changement d'état des deux transistors d'une phase. Ainsi, il y a un court instant où aucun des deux transistors ne conduit (état haute impédance). La figure 2.5 montre la commande PWM d'une phase qui est générée par le régulateur. Les temps-morts sont bien visibles.

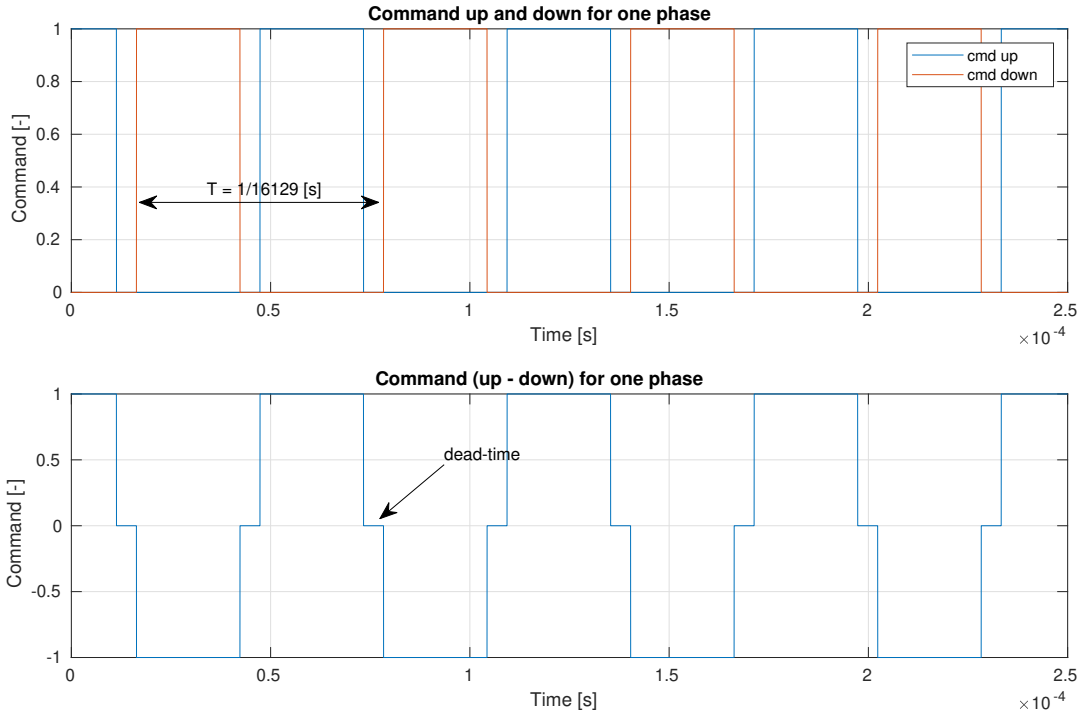


Figure 2.5: Exemple de commande PWM réelle (à 50%) pour une phase

2.2.1.2 Potentiels et tensions

Les lignes de transmission ne sont constituées que des trois phases et ne contiennent donc pas de potentiel de référence. Cela signifie qu'il n'est pas relié physiquement à l'alimentation mais, dans le cas du moteur triphasé (montage en étoile), ce potentiel de référence peut être vu comme une masse virtuelle au point milieu entre les trois phases de la charge (figure 2.6). Si cette charge est composée de trois branches égales, dont les impédances sont purement résistives (aucune dynamique, qui ne dépend pas des valeurs du passé), alors le potentiel de référence (V_{ref}) est égal à la moyenne des trois potentiels (équation 2.2).

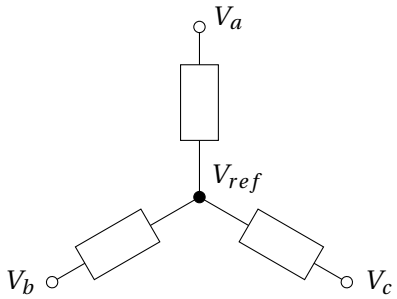


Figure 2.6: Schéma électrique d'une charge idéale en étoile

$$V_{ref}(t) = \frac{V_a(t) + V_b(t) + V_c(t)}{3} \tag{2.2}$$

En réalité, le moteur est constitué d'éléments réactifs et d'une tension induite (section 2.2.3.1) qui engendrent des variations et rendent l'équation 2.2 un peu plus compliquée. Cependant, avec une dynamique faible et une tension induite de mouvement qui est correctement prise en compte dans la modélisation du moteur, les variations du potentiel de référence sont négligeables (exemple avec le filtre à la figure 2.13). De plus, utiliser l'équation 2.2 simplifie grandement les calculs et la compréhension du système. Pour la suite du travail, le potentiel de référence est supposé stable et fixé à:

$$V_{ref} = \frac{U_e}{2}$$

Pour que cette hypothèse reste valide et que ce potentiel V_{ref} ne varie pas, il est indispensable que la commande soit bien équilibrée (équation 2.2). La moyenne des trois potentiels au niveau de l'onduleur doit donc toujours être égale à $V_{ref} = U_e/2$ correspondant à un « duty cycle » de 50 %. C'est le rôle du régulateur de générer une commande correctement équilibrée.

Une tension électrique est définie comme étant la différence entre deux potentiels électriques. Une tension de phase (« phase voltage ») est donc égale à la différence entre le potentiel d'une phase et le potentiel de référence V_{ref}:

$$\begin{cases} U_a = V_a - V_{ref} = V_a - \frac{U_e}{2} \\ U_b = V_b - V_{ref} = V_b - \frac{U_e}{2} \\ U_c = V_c - V_{ref} = V_c - \frac{U_e}{2} \end{cases} \tag{2.3}$$

Définies ainsi, les tensions de phases peuvent varier entre $-U_e/2$ (« duty cycle » de 0 %) et $+U_e/2$ (« duty cycle » de 100 %).

Les tensions de phases ne sont pas la seule représentation possible des tensions dans une ligne triphasée et elles présentent d'ailleurs un inconvénient en pratique. La tension de phase se mesure par rapport au potentiel de référence (V_{ref} sur la figure 2.6) mais celui-ci n'est pas accessible physiquement sur un moteur triphasé réel et cette tension n'est donc pas directement mesurable.

Dans le cas d'une mesure expérimentale, c'est donc la tension entre phases (« phase-to-phase voltage ») qui est utilisée. Il s'agit de la tension entre deux bornes du moteur. La relation entre les tensions de phase et les tensions entre phases n'est pas évidente si celle-ci doit être exprimée en valeur instantanée, en tenant compte de la dynamique de la charge réactive. Cependant, lorsque les trois tensions de phases sont exprimées en valeur efficace et que ces trois tensions efficaces sur chacune des phases sont égales (ce qui est le cas pour une commande bien équilibrée), la relation est définie avec un simple facteur donné à l'équation 2.6. Une tension de phase sinusoïdale est définie en fonction du temps selon l'équation 2.4.

$$u_p(t) = \widehat{u}_p \cdot sin(\omega \cdot t) \tag{2.4}$$

Où:
$u_p(t)\,[V]$ est la tension de phase en valeur instantanée en fonction du temps
$\widehat{u}_p\,[V]$ est la valeur crête du signal sinusoïdal
$\omega\,[rad/s] = 2 \cdot \pi \cdot f$ est la vitesse de rotation liée à la fréquence du signal sinusoïdal

La tension efficace de phase est définie selon l'équation 2.5:

$$U_p = \frac{1}{\sqrt{2}} \cdot \hat{u}_p \tag{2.5}$$

Où:
$U_p\,[V]$ est la tension de phase en valeur efficace
$\hat{u}_p\,[V]$ est la valeur crête du signal sinusoïdal

Et dans le cas où les trois tensions de phases sont égales, les tensions efficaces entre phases sont définies selon l'équation 2.6:

$$U_{pp} = \sqrt{3} \cdot U_p \tag{2.6}$$

Où:
$U_{pp}\,[V]$ est la tension entre phases en valeur efficace
$U_p\,[V]$ est la tension de phase en valeur efficace

Cette relation entre les tensions d'entre phases et tension de phase (équation 2.6) est issue de la différence géométrique entre deux phases (figure 2.7 et équation 2.7).

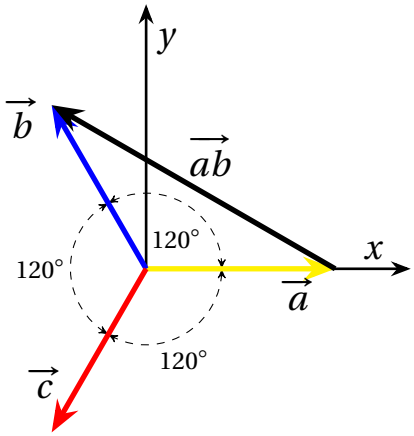


Figure 2.7: Représentation géométrique de la relation entre tension de phase et tension d'entre phases

$$\begin{cases} a_x = cos(0°) = 1 \\ a_y = sin(0°) = 0 \end{cases}$$

$$\begin{cases} b_x = cos(120°) = -\frac{1}{2} \\ b_y = sin(120°) = \frac{\sqrt{3}}{2} \end{cases}$$

$$\begin{cases} c_x = cos(240°) = -\frac{1}{2} \\ c_y = sin(240°) = -\frac{\sqrt{3}}{2} \end{cases}$$

$$\begin{cases} ab_x = b_x - a_x = -\frac{3}{2} \\ ab_y = b_y - a_y = \frac{\sqrt{3}}{2} \end{cases}$$

$$\left|\vec{ab}\right| = \sqrt{\left(-\frac{3}{2}\right)^2 + \left(\frac{\sqrt{3}}{2}\right)^2} = \sqrt{3} \tag{2.7}$$

2.2.1.3 Retard de l'onduleur

La commande PWM qui commande l'onduleur génère un retard. Ce retard de la commande a été étudié dans un travail précédent ([3], page 48) et celui-ci correspond à une demi-période d'échantillonnage. Il a été modélisé sous forme de fonction de transfert en utilisant l'approximation de Padé d'ordre 2:

$$e^{s\cdot\tau_d} \approx \frac{1-k_1\cdot s+k_2\cdot s^2}{1+k_1\cdot s+k_2\cdot s^2} = G_{delay}(s) \tag{2.8}$$

Avec:

$$\tau_d = \frac{1}{2\cdot F_{PWM}} = \frac{1}{2\cdot 16129} = 31\,[\mu s]$$
$$k_1 = \frac{\tau_d}{2} = \frac{1}{2\cdot 16129\cdot 2} = 15.5\cdot 10^{-6}$$
$$k_2 = \frac{\tau_d^2}{12} = \frac{1}{(2\cdot 16129)^2\cdot 12} = 80.1\cdot 10^{-12}$$

La fonction de transfert avec les valeurs numériques est donnée à l'équation 2.9.

$$G_{delay}(s) = \frac{1-15.5\cdot 10^{-6}\cdot s+80.1\cdot 10^{-12}\cdot s^2}{1+15.5\cdot 10^{-6}\cdot s+80.1\cdot 10^{-12}\cdot s^2} = \frac{s^2-194\cdot 10^3\cdot s+12.5\cdot 10^9}{s^2+194\cdot 10^3\cdot s+12.5\cdot 10^9} \tag{2.9}$$

2.2.1.4 Modélisation du bloc « Power »

La commande du régulateur à vérifier est composée de six signaux PWM (avec une valeur entre 0 et 1). Cependant, comme expliqué auparavant, ces signaux vont par pairs et il est plus pertinent de représenter la commande du système à régler avec un seul signal pour chaque phase. Pour connecter la sortie du régulateur avec l'entrée du modèle du système à régler, il suffit de faire la différence entre les signaux "up" et "down" de chacune des phases. Les trois phases sont ainsi commandées par des signaux pouvant prendre les valeurs -1, 0 ou 1 avec une modulation PWM.

Pour le gain de ce bloc « Power », il suffit de multiplier le signal par $U_e/2$, la tension d'une phase peut ainsi varier entre $-U_e/2$ et $+U_e/2$ comme expliqué à la section 2.2.1.2.

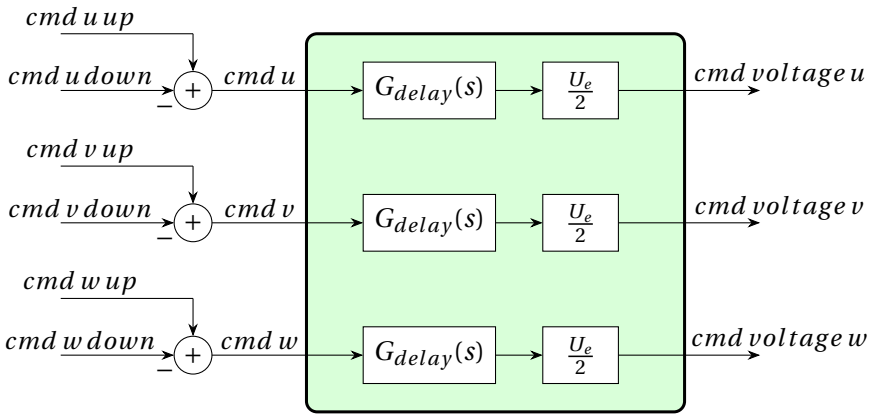


Figure 2.8: Modélisation de l'étage de puissance (bloc « Power »)

2.2.2 Bloc « Filter »

Les lignes de transmission sont constituées de trois phases dont les tensions sont modulées par PWM. L'onduleur et le moteur sont séparés par des câbles d'une distance pouvant varier entre 55 et 232 $[m]$. Des câbles d'une telle longueur créent des problèmes de « cross-talk » lorsque les signaux sont composés de hautes fréquences. Pour atténuer ce problème, un filtre a été placé entre l'onduleur et les câbles.

Le filtre et les câbles sont des éléments relativement complexes à modéliser sous forme de fonctions de transfert. Dans les travaux précédents, la modélisation du filtre et des câbles qui semble la plus précise et la plus intuitive est celle réalisée avec les outils Simscape de Simulink. Celle-ci n'utilise pas de fonctions de transfert mais un assemblage de composants électroniques passifs (voir figure 2.9) permettant de simuler le comportement électrique.

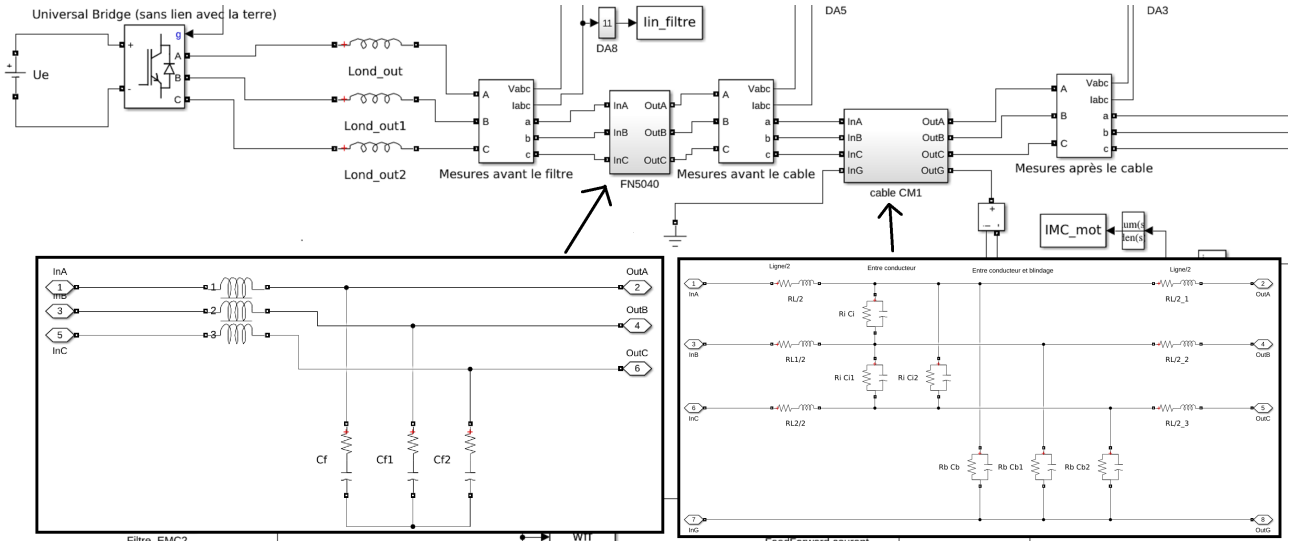


Figure 2.9: Modélisation du filtre et des câbles avec l'outil Simscape
`dev/01_plant_model/simulink_model/51_simscape_modified/CM_FILTRE_FN5040.mdl`

Cette simulation Simscape permet de bien observer les effets du filtre et des câbles. La figure 2.10 montre l'allure des signaux avant et après les éléments modélisés.

Les résultats de cette simulation permettent de constater que le filtre élimine les hautes fréquences du PWM pour obtenir le signal souhaité. Avec un signal déjà filtré, les câbles ont un faible impact sur l'allure du signal de sortie. Selon ces résultats, la modélisation du filtre serait suffisante et la modélisation des câbles ne serait donc pas indispensable.

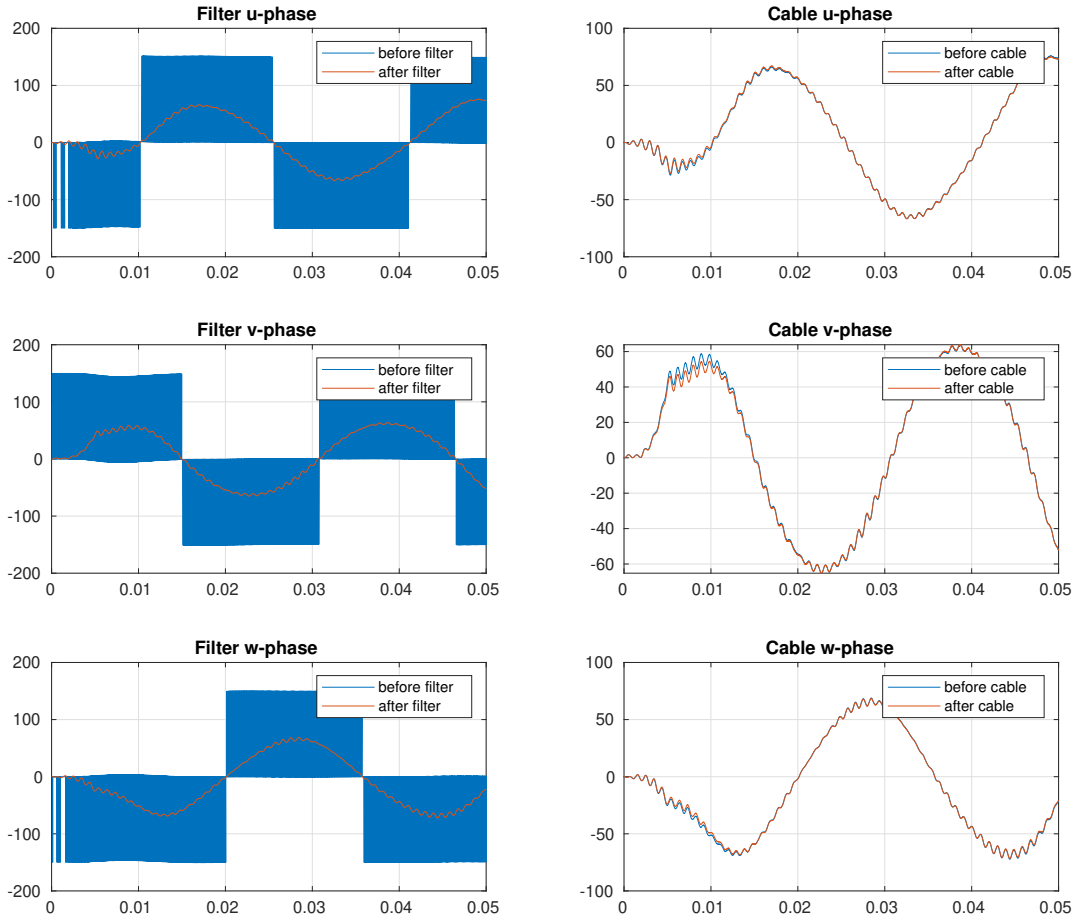


Figure 2.10: Effets du filtre et des câbles (simulation Simscape)
`dev/01_plant_model/simulink_model/51_simscape_modified/ParametreCM_FILTRE_FN5040.m`

Afin de pouvoir simuler le système sans être dépendant de la toolbox Simscape, un travail précédent ([3]) a été réalisé dans lequel le filtre et les câbles sont modélisés sous forme de fonctions de transfert. Ces fonctions de transfert sont regroupées et calculées ensemble (à l'aide des outils Matlab) pour être représentées par une seule fonction de transfert (une seule par phase).

Cette modélisation donne de bons résultats proches de la réalité mais cette manière de procéder présente un inconvénient évident. Avec cette méthode, il est nécessaire de recalculer toute la fonction de transfert dans le cas où il faut changer un paramètre du filtre, des câbles ou même du moteur (qui est vu comme une charge absorbant le courant). Le calcul de cette fonction de transfert global peut se faire automatiquement avec Matlab mais ce type de calcul est un point fort de Matlab qui n'est pas toujours aussi efficace avec d'autres logiciels concurrents (créant ainsi une dépendance à Matlab). De plus, cette modélisation ne permet pas de distinguer les différents éléments et d'effectuer des mesures entre ceux-ci. Pour ces raisons, il peut être intéressant de modéliser le filtre et les câbles séparément avec une analyse plus approfondie (et de ce fait, plus complexe) des équations physiques de ces éléments électriques.

La difficulté réside dans le fait qu'une fonction de transfert représentant un comportement électrique ne peut pas toujours être modélisée avec une seule entrée et une seule sortie. Dans le cas du filtre et des câbles, l'idéal serait d'avoir juste une entrée en tension et une sortie en tension, pour chacune des deux fonctions de transfert. Ceci n'est pas possible car le comportement électrique des composants ne dépend pas que de la tension mais aussi du courant électrique circulant au travers de ces composants. Ces fonctions de transfert doivent être vues comme des filtres dont la tension de sortie ne dépend pas que de la tension d'entrée mais aussi du courant que la charge doit absorber. Il est donc nécessaire d'ajouter un « feedback » à la fonction de transfert de manière à prendre en compte le courant de sortie. Les fonctions de transfert du filtre et des câbles peuvent ainsi être connectées comme sur la figure 2.11.

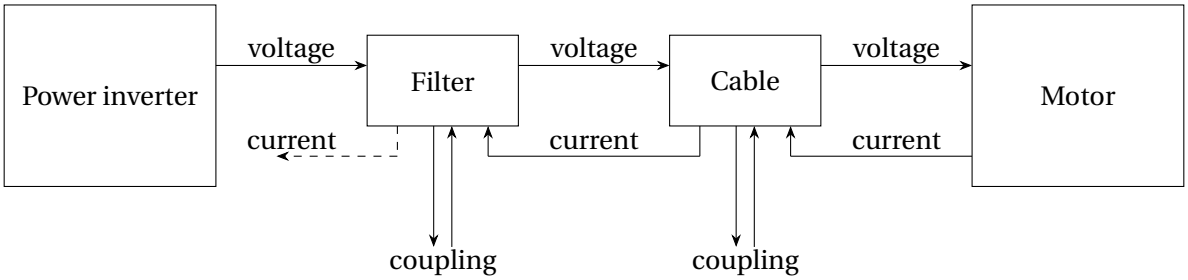


Figure 2.11: Schéma bloc des fonctions de transfert du filtre et des câbles avec « feedback »

Cette manière de représenter les fonctions de transfert avec un « feedback » pour prendre en compte la contribution de la charge est une approche similaire à la représentation EMR (« Enegetic Macroscopic Representation »). Cette représentation offre l'avantage d'observer la puissance du système. En effet, avec une valeur de tension et de courant entre chaque élément, il est possible de quantifier la puissance reçue, transmise et consommée par l'élément lui-même pour estimer les pertes.

Modélisées et connectées de cette manière, les fonctions de transfert du filtre et des câbles peuvent être calculées indépendamment sans avoir besoin de connaître précisement tous les composants des autres blocs. Ceci permet également de supprimer ou d'ajouter un nouveau bloc sans avoir à recalculer les autres.

2.2.2.1 Filtre

Le filtre utilisé est le Schaffner FN 5045-10-44. C'est un « Sine wave filter » qui a pour but de lisser le signal PWM pour ne garder que la fréquence fondamentale et ainsi obtenir un signal de sortie sinusoïdal. Le schéma électrique équivalent de ce filtre est donné ci-dessous:

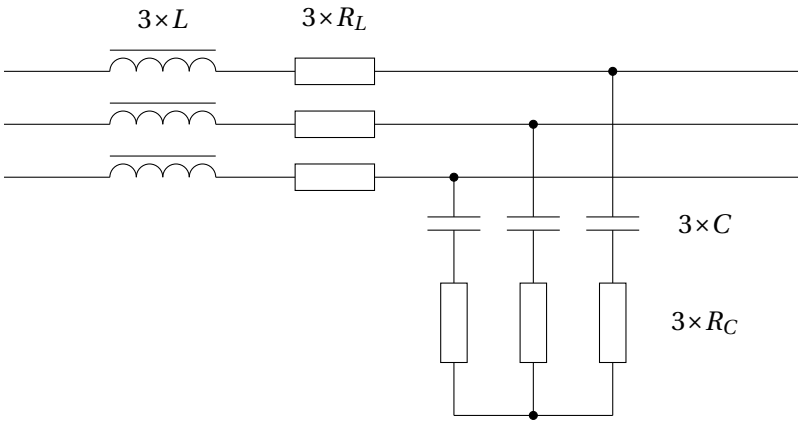


Figure 2.12: Schéma électrique équivalent du filtre FN 5045-10-44

Pour réussir à modéliser ce filtre sans que les calculs deviennent trop complexes, il a été nécessaire de faire quelques hypothèses. Sur le schéma 2.12, les trois phases sont connectées en étoile au niveau des condensateurs. Cette connexion en étoile rend les calculs bien plus complexes car chaque tension et courant de chacune des branches dépend des tensions et des courants des autres branches. Les équations de chacune des phases deviennent dépendantes des autres phases et les équations deviennent trop compliquées pour être aisément implémentées sous forme de fonction de transfert.

Comme expliqué à la section 2.2.1.2, si la commande est bien balancée et que les impédances sur chacune des phases sont identiques, les variations de potentiel du noeud central ne dépendent que de la dynamique de la charge. La simulation avec Simscape permet d'observer ces variations de potentiel au niveau du noeud central (figure 2.13).

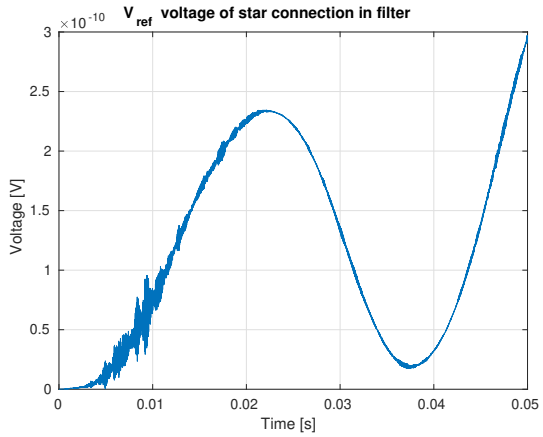


Figure 2.13: Potentiel de référence du noeud central dans le filtre (simulation Simscape)
`dev/01_plant_model/simulink_model/51_simscape_modified/ParametreCM_FILTRE_FN5040.m`

Le potentiel observé ci-dessus est inférieur à 1 nV alors que les tensions de phases oscillent au-delà de 50 V. Cette observation justifie l'hypothèse que ce potentiel est nul (0 V). Les calculs peuvent ainsi être simplifiés.

Avec l'hypothèse que le potentiel électrique du noeud entre les condensateurs ne varie pas, la fonction de transfert pour une seule phase devient bien plus simple et le couplage entre chaque phase ne dépend plus que des inductances mutuelles et des courants circulant dans celles-ci.

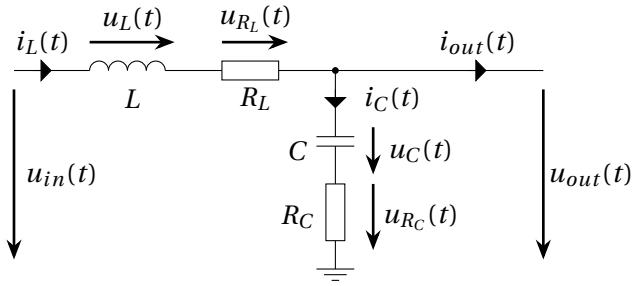


Figure 2.14: Schéma électrique du filtre FN 5045-10-44 simplifié pour une seule phase

La grandeur recherchée est la tension de sortie $U_{out}(t)$ qui est calculée ainsi:

$$u_{out}(t) = u_{R_C}(t) + u_C(t) = i_C(t) \cdot R_C + \frac{1}{C} \int i_C(t)\, dt \tag{2.10}$$

Avec la forme de Laplace:

$$U_{out} = I_C \cdot \left(R_C + \frac{1}{s \cdot C} \right) \tag{2.11}$$

La différence de tension entre l'entrée et la sortie correspond à la chute de tension au niveau de l'inductance (et de sa résistance). La tension aux bornes de l'inductance dépend aussi des courants des autres phases, i_{L2} et i_{L3}. M est le facteur de couplage inductif entre deux phases.

$$u_{in}(t) - u_{out}(t) = u_{R_L}(t) + u_L(t) = i_L(t) \cdot R_L + L \cdot \frac{di_L(t)}{dt} - M \cdot \left(\frac{di_{L2}(t)}{dt} + \frac{di_{L3}(t)}{dt} \right) \tag{2.12}$$

$$U_{in} - U_{out} = I_L \cdot (R_L + s \cdot L) - s \cdot M \cdot (I_{L2} + I_{L3}) \tag{2.13}$$

En retournant l'équation 2.13, le courant circulant dans l'inductance peut être exprimé ainsi:

$$I_L = \frac{U_{in} - U_{out} + s \cdot M \cdot (I_{L2} + I_{L3})}{R_L + s \cdot L} \tag{2.14}$$

Le courant devié I_C correspond à la différence entre le courant d'entrée et celui de sortie:

$$I_C = I_L - I_{out} \tag{2.15}$$

Le courant I_{out} qui sort de ce filtre est celui qui circule dans le moteur et celui-ci est connu. Cela signifie qu'il est possible d'exprimer la fonction de transfert en fonction de I_{out}. C'est ce qui est fait en remplaçant I_C de l'équation 2.11 par l'équation 2.15. Il faut également remplacer I_L par l'équation 2.14.

$$U_{out} = \left(\frac{U_{in} - U_{out} + s \cdot M \cdot (I_{L2} + I_{L3})}{R_L + s \cdot L} - I_{out}\right) \cdot \left(R_C + \frac{1}{s \cdot C}\right)$$

$$U_{out} \cdot s \cdot C \cdot (R_L + s \cdot L) = (U_{in} - U_{out} + s \cdot M \cdot (I_{L2} + I_{L3}) - I_{out} \cdot (R_L + s \cdot L)) \cdot (R_C \cdot s \cdot C + 1)$$

U_{out} peut ainsi être exprimé en fonction de I_{out}, U_{in}, I_{L2} et I_{L3}.

$$U_{out} \cdot \left(s^2 \cdot L \cdot C + s \cdot C \cdot (R_L + R_C) + 1\right) = -I_{out} \cdot \left(s^2 \cdot R_C \cdot L \cdot C + s \cdot (L + R_C \cdot R_L \cdot C) + R_L\right)$$
$$+U_{in} \cdot (s \cdot R_C \cdot C + 1)$$
$$+M \cdot (I_{L2} + I_{L3}) \cdot \left(s^2 \cdot R_C \cdot C + s\right)$$

La fonction de transfert à quatre entrées qui permet de calculer U_{out} peut être décomposée en une somme de quatre fonctions de transfert.

$$\frac{U_{out}(s)}{I_{out}(s)} = -\frac{s^2 \cdot R_C \cdot L \cdot C + s \cdot (L + R_C \cdot R_L \cdot C) + R_L}{s^2 \cdot L \cdot C + s \cdot C \cdot (R_L + R_C) + 1} \tag{2.16}$$

$$\frac{U_{out}(s)}{U_{in}(s)} = \frac{s \cdot R_C \cdot C + 1}{s^2 \cdot L \cdot C + s \cdot C \cdot (R_L + R_C) + 1} \tag{2.17}$$

$$\frac{U_{out}(s)}{I_{L2}(s)} = \frac{U_{out}(s)}{I_{L3}(s)} = \frac{M \cdot \left(s^2 \cdot R_C \cdot C + s\right)}{s^2 \cdot L \cdot C + s \cdot C \cdot (R_L + R_C) + 1} \tag{2.18}$$

En respectant le schéma suivant:

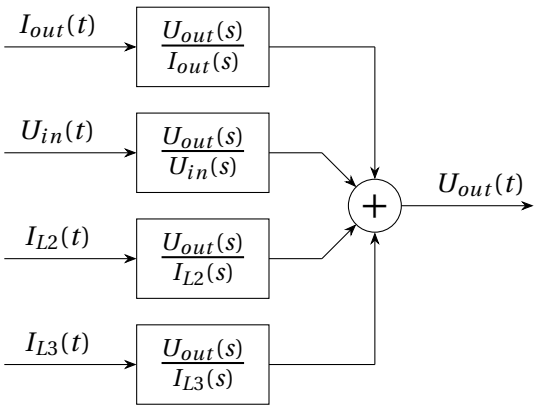


Figure 2.15: Somme des fonctions de transfert pour le calcul de U_{out} du filtre

Cependant, les courants I_{L2} et I_{L3} sont les courants d'inductance des autres phases et ils ne sont pas connus. Il est donc nécessaire de calculer également le courant I_L afin de l'utiliser pour le couplage. Il faut aussi noter que les deux dernières fonctions de transfert (U_{out}/I_{L2} et U_{out}/I_{L3}) sont identiques et peuvent ainsi être regroupées.

En décomposant l'équation 2.14, le courant I_L peut s'écrire selon l'équation 2.19.

$$I_L = \frac{U_{in} - U_{out}}{R_L + s \cdot L} + \frac{s \cdot M \cdot (I_{L2} + I_{L3})}{R_L + s \cdot L} \tag{2.19}$$

La modélisation de cette équation peut être exprimée à l'aide de deux fonctions de transfert:

$$\frac{I_L(s)}{U_{in}(s) - U_{out}(s)} = \frac{I_L(s)}{U_{diff}(s)} = \frac{1}{R_L + s \cdot L} \tag{2.20}$$

$$\frac{I_L(s)}{I_{L23}(s)} = \frac{s \cdot M}{R_L + s \cdot L} \tag{2.21}$$

La figure 2.16 montre le schéma complet de la modélisation du filtre pour une seule phase.

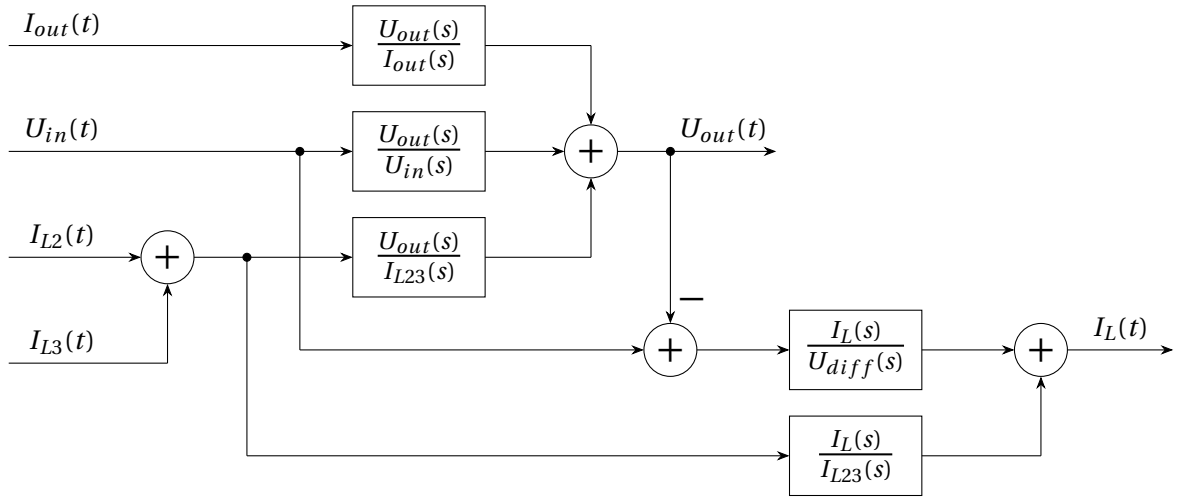


Figure 2.16: Modélisation du filtre pour une seule phase

Pour observer la contribution des différentes fonctions de transfert, il faut remplacer les valeurs analytiques par les valeurs numériques du filtre FN 5045-10-44. Celles-ci sont données à la table 2.1.

Parameter	Value
L (donnée fabricant)	5.2 [mH]
R_L (estimation)	50 [mΩ]
C (donnée fabricant)	6.8 $[\mu$F$]$
R_C (estimation)	1 $[\mu\Omega]$
M (estimation)	0.8 [mH]

Table 2.1: Valeurs des paramètres du filtre FN 5045-10-44
`https://www.schaffner.com/fileadmin/user_upload/pim/products/datasheets/FN5040_FN5045.pdf`

Ci-dessous, les fonctions de transfert du filtre (équations 2.16, 2.17, 2.18, 2.20, 2.21) sont données avec leurs valeurs numériques.

$$\frac{U_{out}(s)}{U_{in}(s)} = \frac{6.8 \cdot 10^{-12} \cdot s + 1}{35.4 \cdot 10^{-9} \cdot s^2 + 340 \cdot 10^{-9} \cdot s + 1}$$

$$\frac{U_{out}(s)}{I_{out}(s)} = \frac{-35.4 \cdot 10^{-14} \cdot s^2 - 5.2 \cdot 10^{-3} \cdot s - 50 \cdot 10^{-3}}{35.4 \cdot 10^{-9} \cdot s^2 + 340 \cdot 10^{-9} \cdot s + 1}$$

$$\frac{U_{out}(s)}{I_{L23}(s)} = \frac{5.44 \cdot 10^{-15} \cdot s^2 + 800 \cdot 10^{-3} \cdot s}{35.4 \cdot 10^{-9} \cdot s^2 + 340 \cdot 10^{-9} \cdot s + 1}$$

$$\frac{I_L(s)}{U_{diff}(s)} = \frac{1}{5.2 \cdot 10^{-3} \cdot s + 50 \cdot 10^{-3}}$$

$$\frac{I_L(s)}{I_{L23}(s)} = \frac{800 \cdot 10^{-3} \cdot s}{5.2 \cdot 10^{-3} \cdot s + 50 \cdot 10^{-3}}$$

Sur la figure 2.17 sont affichés les diagrammes de Bode de ces 5 fonctions de transfert. C'est principalement la tension U_{out} qui est intéressante car le courant I_L n'est utilisé que pour le couplage qui est moins important et plus compliqué à analyser.

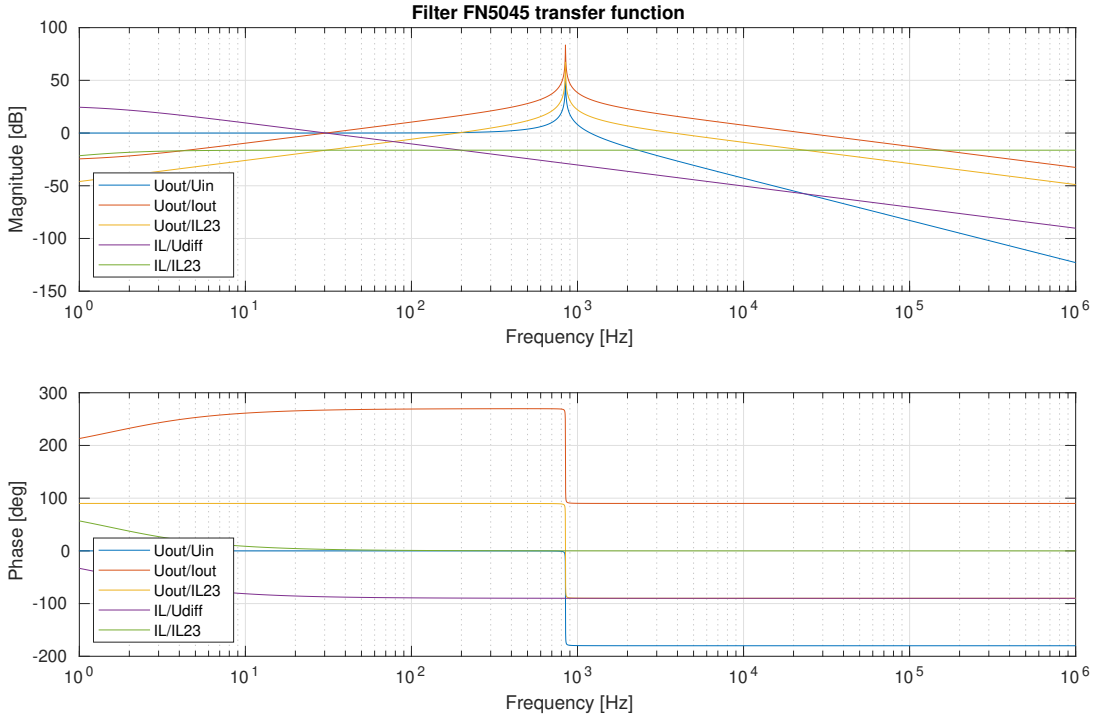


Figure 2.17: Diagrammes de Bode des fonctions de transfert du filtre FN5045
`dev/90_test/10_plant_model/filter_model/filter_model_num.m`

Le diagramme de Bode de la figure 2.17 permet de voir qu'il s'agit bien d'un filtre passe-bas avec une résonnance. La résonnance théorique se situe à 846 $[Hz]$.

Pour vérifier que le filtre est bien modélisé, il faut le comparer avec un modèle fiable en simulant les deux filtres avec les mêmes signaux d'entrée. L'autre modèle de filtre qui peut être utilisé comme référence est celui de la simulation Simscape (figure 2.9). Ce filtre de référence a été modélisé directement avec des composants électriques et les équations physiques qui régissent ce filtre sont résolues par le logiciel Matlab/Simulink.

La simulation Simscape dans laquelle est modélisé le filtre de référence contient un régulateur et un système à régler. Il est ainsi possible de lancer la simulation en boucle fermée. De cette manière, les tensions de commande et les courants électriques appliqués au filtre sont pertinents et ne devraient pas être très différents de la réalité.

Cependant, il s'agit d'une version plus ancienne du système et certains paramètres de cette simulation ne sont plus d'actualité. Par exemple, la tension du PWM de la commande (figure 2.19) ne correspond pas tout à fait à la tension actuelle de $390/2\,[V]$. Malgré quelques différences de gain et d'amplitude, les signaux générés par le régulateur ont une forme identique à ceux utilisés dans les simulations plus récentes. Et de toute manière, le but de cette simulation est de comparer le comportement du filtre et celui-ci ne devrait pas dépendre de l'amplitude mais plutôt des composantes fréquentielles du signal d'entrée.

La simulation Simscape permet de mesurer les tensions et les courants avant et après le filtre. Ceux-ci peuvent être appliqués aux entrées du nouveau modèle de filtre comme sur la figure 2.18.

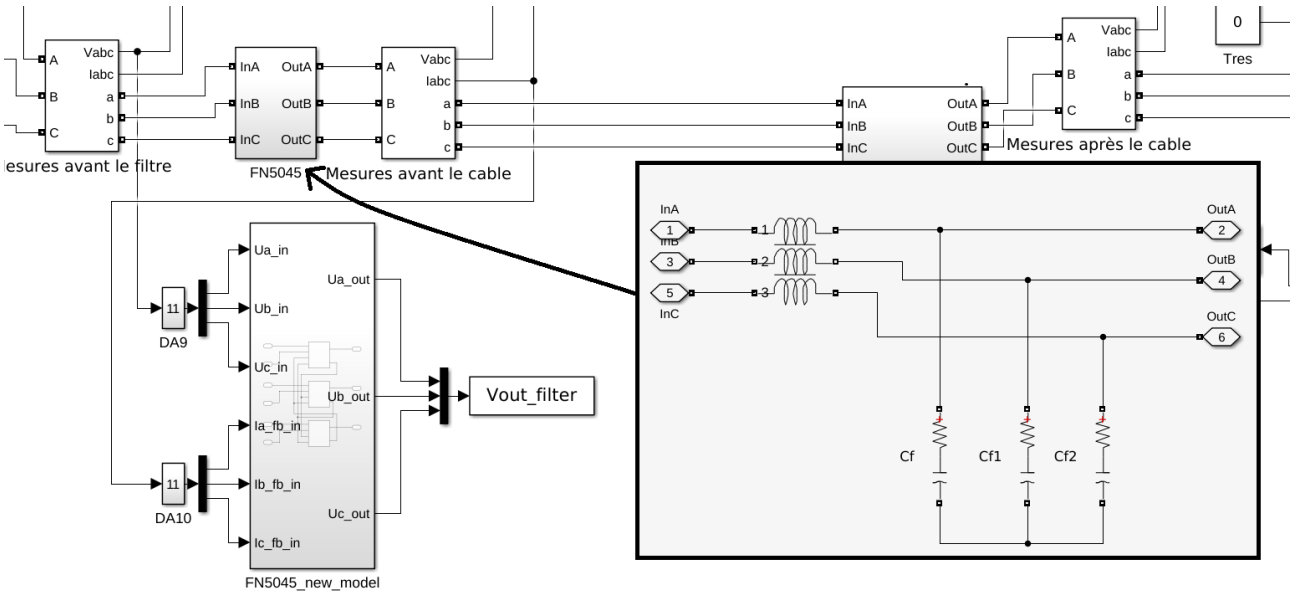


Figure 2.18: Schéma Simulink/Simscape pour la comparaison des modèles de filtre
`dev/90_test/10_plant_model/filter_model_compare/CM_FILTRE_FN5040.mdl`

L'allure des tensions d'entrée et des courants de sortie (appliqués en entrée du filtre comme feedback) sont visibles sur les figures 2.19 et 2.20.

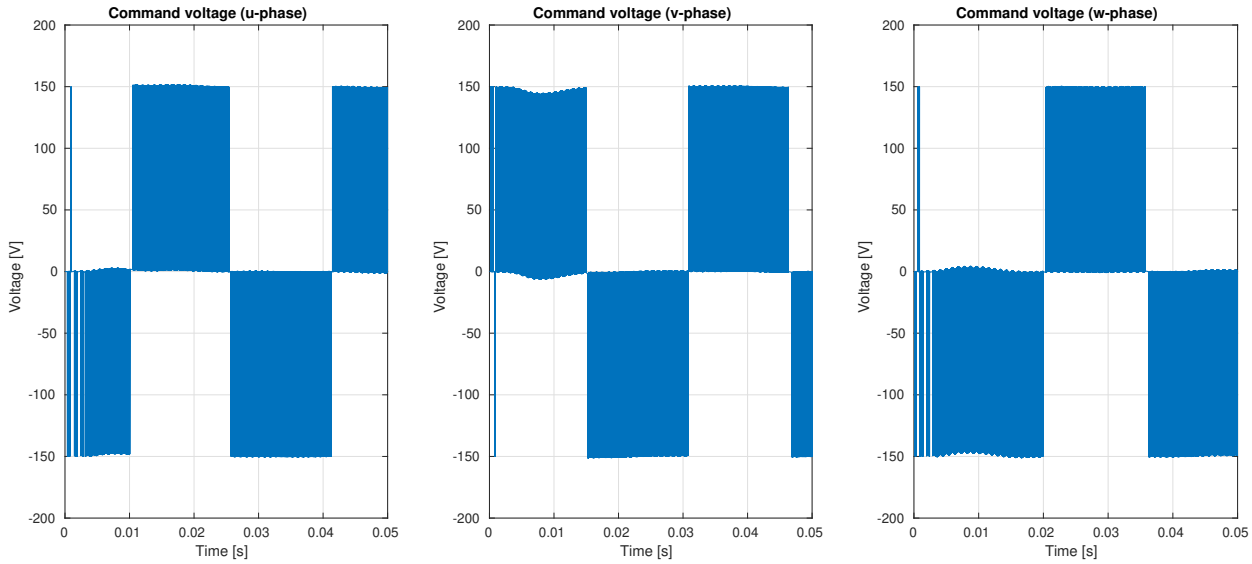


Figure 2.19: Tensions d'entrée du filtre pour la comparaison des modèles du filtre
`dev/90_test/10_plant_model/filter_model_compare/filter_model_compare.m`

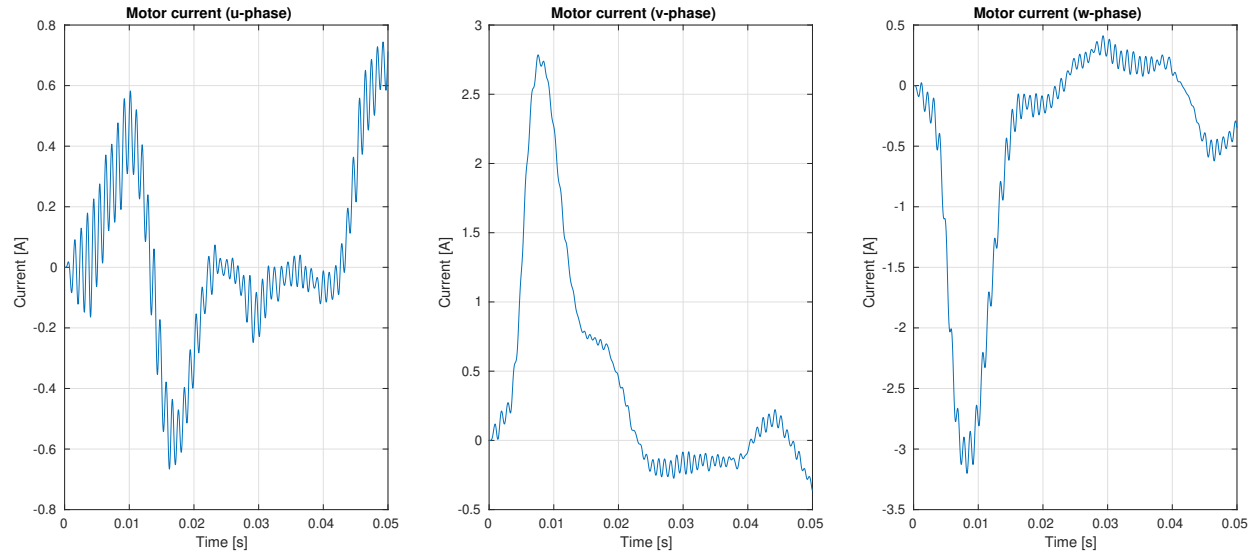


Figure 2.20: Courants de sortie du filtre pour la comparaison des modèles du filtre
`dev/90_test/10_plant_model/filter_model_compare/filter_model_compare.m`

Sur la figure 2.21 sont affichées les comparaisons des tensions de sortie du filtre pour chacune des phases. En observant l'erreur MSE (« Mean squarred error »), la différence entre les deux modèles semble importante. Cependant, l'allure des signaux montre que cette erreur importante est causée par des variations d'amplitude proches de la fréquence de résonance. L'ordre de grandeur des amplitudes, les déphasages et l'allure générale des signaux sont relativement proches. Il ne semble pas que les différences observées soient causées par une erreur de modélisation mais plutôt par des méthodes de calculs ou une précision différentes. Les comportements de manière générale de ces deux modèles semblent cohérents.

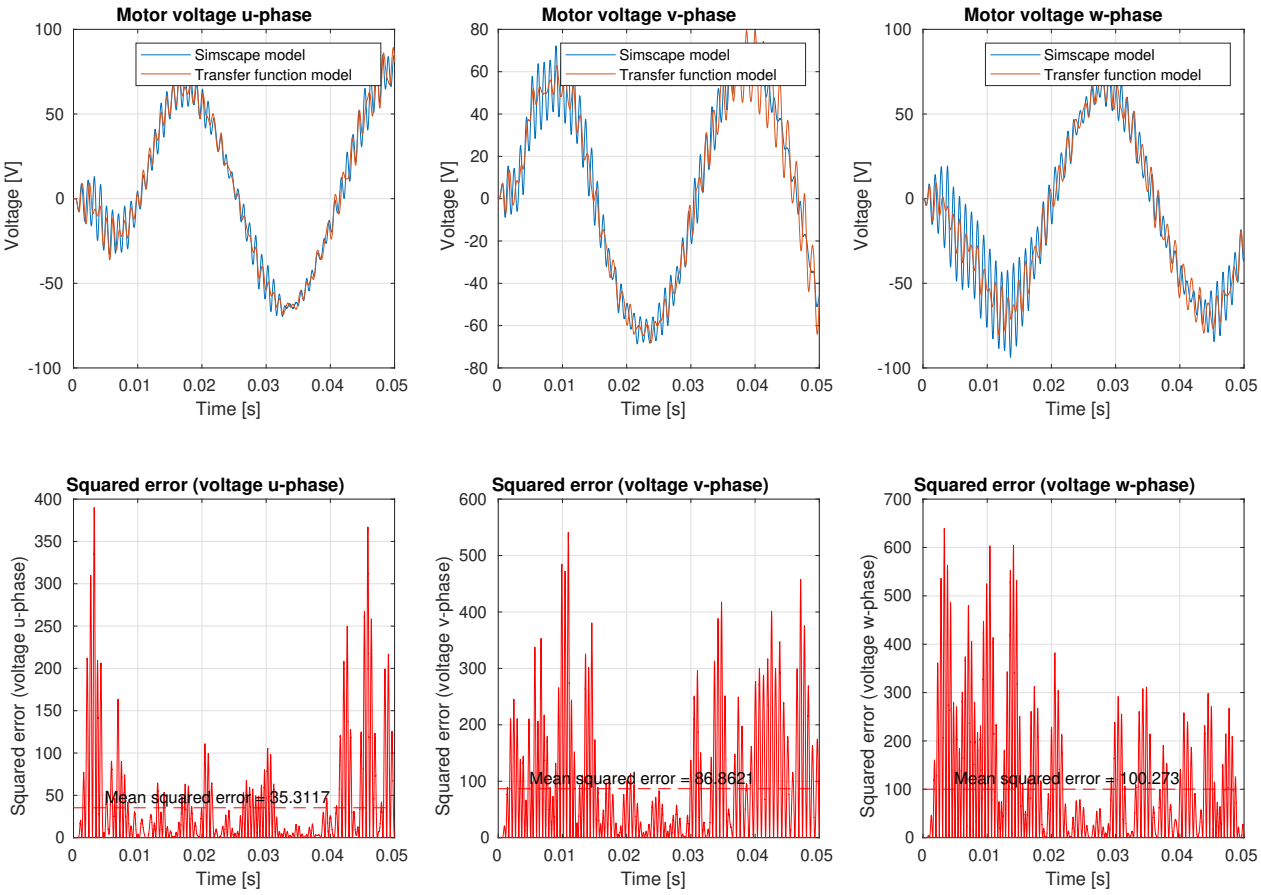


Figure 2.21: Comparaison des modèles du filtre (tensions de sortie)
`dev/90_test/10_plant_model/filter_model_compare/filter_model_compare.m`

2.2.2.2 Câbles

Pour la modélisation des câbles, le schéma électrique équivalent a déjà été étudié dans des travaux précédents et est donné ci-dessous:

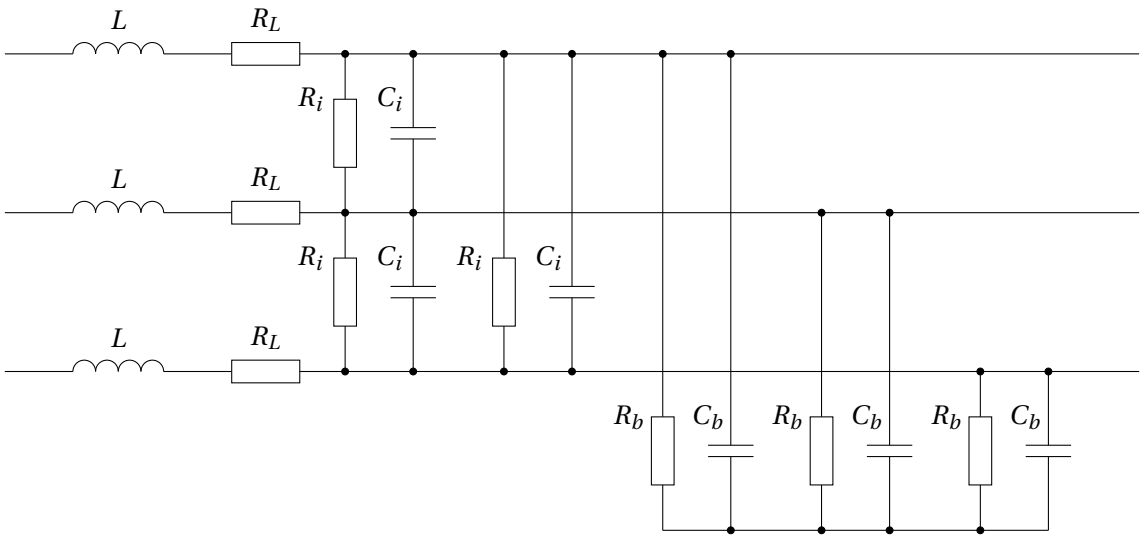


Figure 2.22: Schéma électrique équivalent des câbles

Ce schéma électrique a un comportement passe-bas comme pour le filtre FN 5045-10-44. Le résultat de la simulation (figure 2.10) a montré que la modélisation des câbles n'est pas indispensable si le signal est déjà filtré par le filtre prévu à cet effet. Pour ce travail, les câbles ne sont donc pas modélisés.

2.2.2.3 Modélisation du bloc « Filter »

Il a été décidé, d'après les résultats de simulation de la figure 2.10, de ne modéliser que le filtre. La modélisation du filtre pour une seule phase est donnée à la figure 2.16. La modélisation complète avec les trois phases couplées est visible à la figure 2.23. Les fonctions de transfert utilisées correspondent aux équations 2.16, 2.17, 2.18, 2.20 et 2.21.

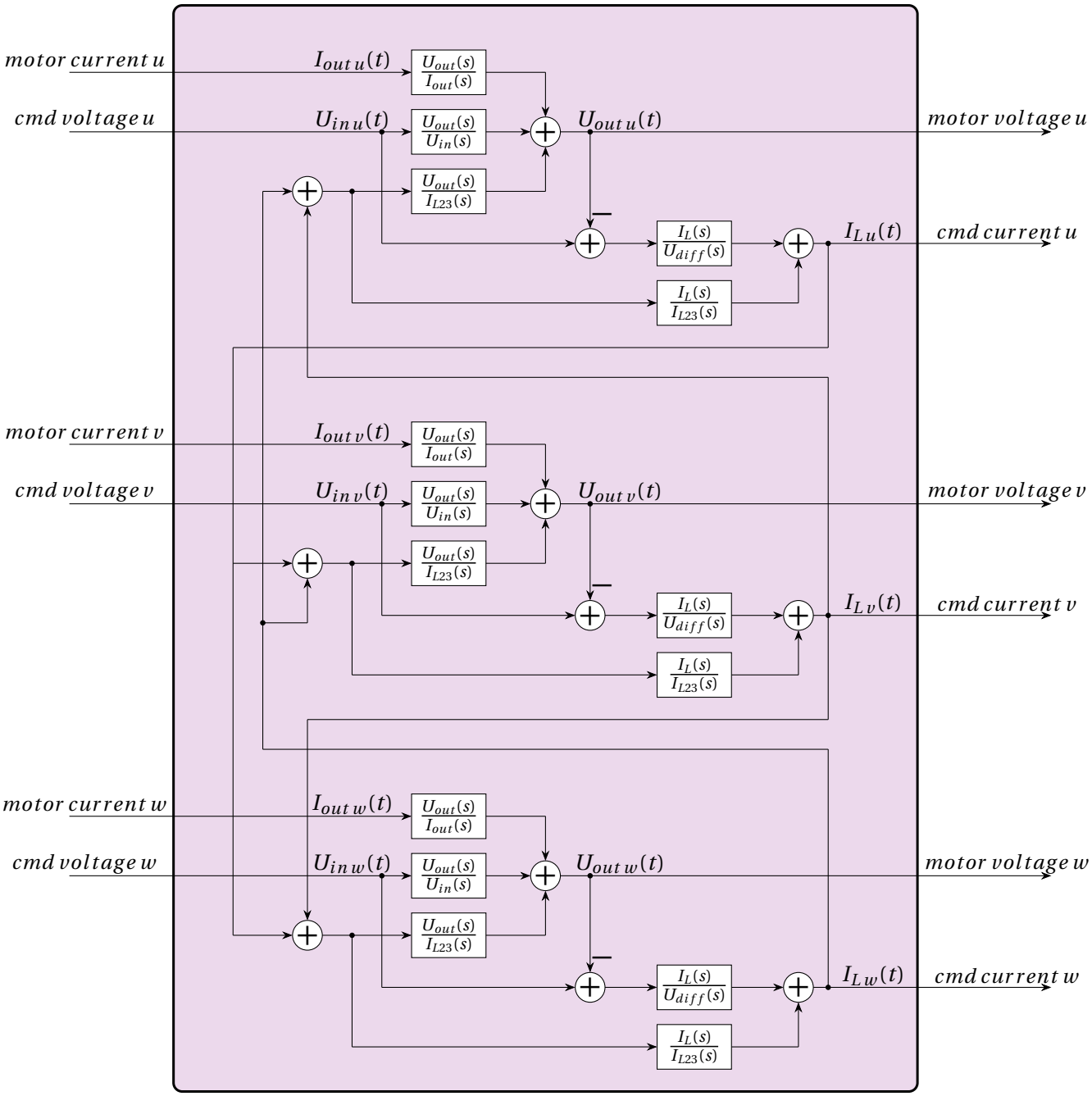


Figure 2.23: Modélisation du bloc « Filter »

2.2.3 Bloc « Motor »

Le moteur utilisé est un moteur de type synchrone à aimants permanents aussi appelé PMSM (« Permanent Magnet Synchronous Motor »). Le modèle est un Alxion 145ST2M (1500 rpm) à 6 paires de pôles avec une connexion en étoile. Il s'agit d'une version spéciale (145ST2M023) avec le bobinage qui a été modifié. Ces caractéristiques sont données au tableau 2.2.

Phase resistance (R)	0.839 [Ω]
Phase inductance (L)	3.8 [mH]
Back EMF constant (K_e)	0.98 $\left[\frac{V}{rad \cdot s}\right]$

Table 2.2: Caractéristiques du moteur 145ST2M023

Dans la section 2.2.1.2, il a été expliqué que pour avoir une référence de potentiel stable, il est nécessaire que la moyenne des trois tensions de phase soit toujours nulle. Grâce à la géométrie du moteur triphasé, une moyenne nulle des tensions peut quand même créer une composante vectorielle non nulle à l'intérieur du moteur. Par exemple, avec les trois tensions suivantes:

$$\begin{cases} U_a = 0.58 \cdot \frac{U_e}{2} \\ U_b = 0.21 \cdot \frac{U_e}{2} \\ U_c = -0.79 \cdot \frac{U_e}{2} \end{cases}$$

La moyenne de ces trois tensions est nulle mais en prenant en compte le déphasage entre chaque phase du moteur, le vecteur résultant n'est pas nul.

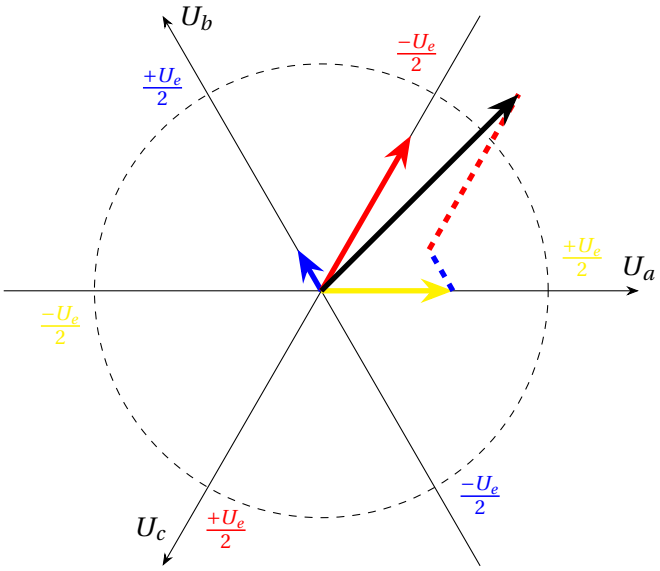


Figure 2.24: Illustration de la somme vectorielle d'un moteur triphasé

Les trois tensions de phases sont calculées de manière à obtenir le couple souhaité, à la position souhaitée, en ayant toujours une moyenne nulle entre les trois tensions de phases. Ceci se fait "automatiquement" grâce à la transformée de Park expliquée à la section 2.2.3.2.

2.2.3.1 Équations du moteur triphasé

Pour comprendre les équations du moteur triphasé, il faut déjà observer le schéma électrique d'une phase, figure 2.25.

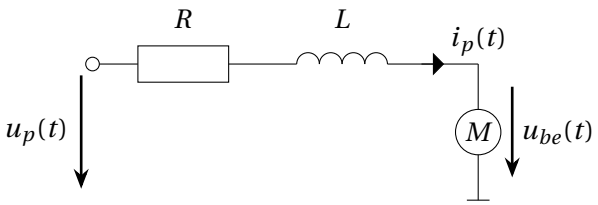


Figure 2.25: Schéma électrique d'une phase du moteur

La tension u_p est la tension de phase (« phase voltage ») par rapport au noeud de référence qui est supposé stable (hypothèse émise à la section 2.2.1.2). Le bobinage du moteur est modélisé par une inductance et une résistance en série. La tension u_{be} est la tension induite de mouvement, appelée aussi force contre-électromotrice ou Back EMF (« Back Electro-motive Force »). D'un point de vue purement électrique, le schéma de la figure 2.25 peut être représenté par l'équation 2.22.

$$u_p(t) = u_R(t) + u_L(t) + u_{be}(t) = R \cdot i_p(t) + L \cdot \frac{di_p}{dt} + u_{be}(t) \tag{2.22}$$

Avec la forme de Laplace:

$$U_p = R \cdot I_p + s \cdot L \cdot I_p + U_{be} \tag{2.23}$$

Dans un moteur électrique, le couple mécanique est proportionel au courant électrique et c'est donc cette grandeur qui est recherchée. Le comportement électrique d'une phase du moteur est ainsi exprimé comme une fonction de transfert, défini à l'équation 2.24.

$$\frac{I_p(s)}{U_p(s) - U_{be}(s)} = \frac{1}{R + s \cdot L} = G_{mot}(s) \tag{2.24}$$

Il est important de noter que l'inductance du moteur entraîne un déphasage entre la tension et le courant circulant dans le moteur. Ce déphasage courant-tension est appelé γ et il est donc modélisé à travers la fonction de transfert de l'équation 2.24.

L'entrée de cette fonction de transfert est une différence entre la tension de phase (grandeur qui est connue) et la tension induite de mouvement. Cette dernière doit être calculée et il est donc nécessaire de comprendre son origine.

La tension induite de mouvement est un phénomène électromagnétique qui génère une tension dans le bobinage afin de s'opposer au mouvement. Elle est présente dans tout moteur électrique et dans le cas du moteur triphasé, elle caractérise également le couplage entre les trois phases. La cause de cette force est liée à la variation du flux magnétique. Lorsque que celui-ci augmente, la tension induite diminue et vice-versa.

$$U_{be} = -N \cdot \frac{d\Phi}{dt} \tag{2.25}$$

Où:
$u_{be}\,[V]$ est la tension induite de mouvement
$N\,[-]$ est le nombre de spires dans le moteur
$\Phi\,[Wb]$ est le flux magnétique

Pour ce travail de modélisation, il n'est pas nécessaire d'analyser le flux magnétique en détail. Pour la modélisation, la seule formule à retenir concernant cette tension u_{be} est la suivante:

$$U_{be_p} = K_{e_p} \cdot \omega_m \tag{2.26}$$

Où:
U_{be_p} $[V]$ est la tension induite de mouvement en valeur efficace (pour une phase)
K_{e_p} $[V/(rad/s)]$ est la constante de tension induite (pour une tension de phase)
ω_m $[rad/s]$ est la vitesse de rotation mécanique du moteur

La constante de tension induite de mouvement K_e est donnée par le fabricant du moteur (table 2.2). Cependant, l'unité des constantes du moteur et ce qu'elles représentent n'est pas toujours clair. Il y a des incohérences liées à ces constantes entre les différents modèles des travaux précédents. Afin d'éviter toutes confusions et erreurs de modélisation, il est important de se poser les questions suivantes à propos de cette constante:

- Quelle est son unité ?
- Est-elle définie pour une tension de phase ou une tension entre phases ?
- Est-elle définie pour une valeur efficace ou une valeur instantanée (crête) ?

L'unité dans laquelle est donnée cette constante peut varier. Dans le cas présent, elle est donnée dans l'unité souhaitée, en $[V/(rad/s)]$.

Cette constante K_e peut être définie pour une tension de phase (« phase voltage ») ou tension entre phases (« phase-to-phase voltage »). Comme expliqué à la section 2.2.1.2, le facteur entre ces deux types de tension est de $\sqrt{3}$ (équation 2.7). Ici, le fabricant précise qu'il s'agit de la tension entre phases.

Il faut également savoir si cette constante est donnée pour une tension induite de mouvement en valeur efficace ou pour une valeur instantanée. Lorsque le moteur tourne, la tension induite de mouvement varie pour chaque phase en fonction de la différence de position entre le rotor et la phase. À vitesse constante, la tension induite est sinusoïdale et le rapport entre sa tension de crête et sa tension efficace est de $\sqrt{2}$ (équation 2.5). Le fabricant ne précise pas cette information. Pour s'assurer qu'il n'y ait aucune confusion, il faut s'appuyer sur les résultats d'une mesure expérimentale qui avait été réalisée précedemment au CERN ([4]).

Sur la figure 2.26, les résultats de la mesure expérimentale du moteur 145ST2M023 sont visibles (graphiques du bas). À gauche, les signaux sinusoïdaux des trois tensions de phase à la vitesse de $103[rad/s]$. À droite, une approximation linéaire afin d'en déduire la valeur de K_e. Sur ce graphique, il est bien précisé qu'il s'agit des tensions entre phases en valeur efficace. Avec une valeur mesurée de 1.018, K_e est proche de la valeur donnée par le fabricant (tableau 2.2, $K_e = 0.98$).

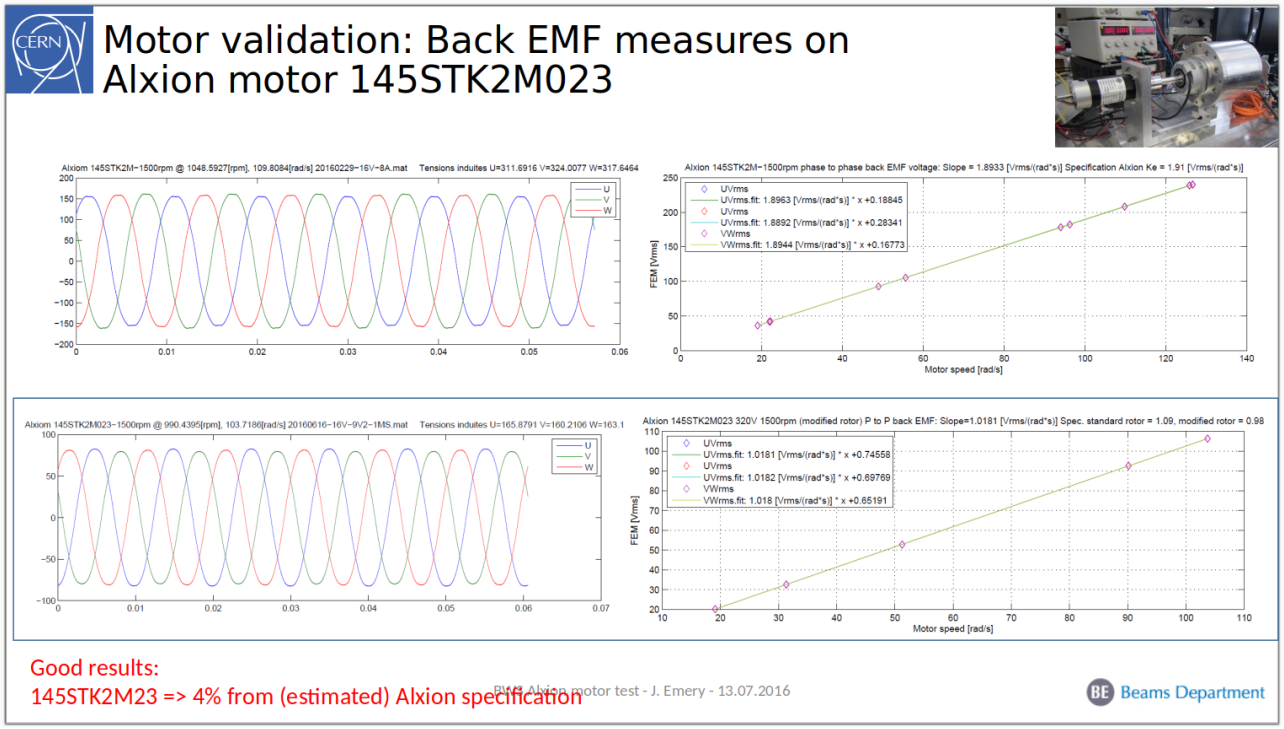


Figure 2.26: [4] Résultats de mesures expérimentales de K_e

Avec ces informations supplémentaires, il est important de rappeler que l'équation 2.26 n'est valable que pour une tension de phase en valeur efficace et que cette équation n'est donc pas cohérente avec la constante k_e donnée par le fabricant. La constante k_e du fabricant (définie pour une tension entre phases) vaut:

$$K_e = \sqrt{3} \cdot K_{e_p} \tag{2.27}$$

Et l'équation 2.26 peut être redéfinie avec la constante du fabricant:

$$U_{be_p} = \sqrt{\frac{1}{3}} \cdot K_e \cdot \omega_m \tag{2.28}$$

Cependant, cette équation (2.28) ne convient toujours pas tout à fait à la modélisation car le modèle doit être simulé au cours du temps. Pour prendre en compte les oscillations en fonction de la position du moteur, il est nécessaire de calculer la tension induite de mouvement avec une valeur instantanée en fonction de la position. Celle-ci est donnée à l'équation 2.29.

$$u_{be_p}(\theta_e) = -\sqrt{\frac{2}{3}} \cdot K_e \cdot \omega_m \cdot sin\left(\theta_e - n \cdot \frac{2\pi}{3}\right) \qquad n = 0,1,2 \tag{2.29}$$

La tension induite de mouvement varie en fonction de la position électrique du moteur. Lorsque le rotor est aligné avec la phase, cette tension induite est nulle. Le signe négatif de l'équation 2.29 est lié aux conventions utilisées dans le modèle et n'a aucune importance d'un point de vue de la théorie. Le paramètre n de l'équation 2.29 permet de distinguer les 3 phases séparées chacune par 120° (du point de vue électrique).

La relation entre position électrique et mécanique est donnée par l'équation 2.30.

$$\theta_e = p \cdot \theta_m \tag{2.30}$$

Où:
$\theta_e\,[rad]$ est la position électrique (du point de vue du signal électrique)
$p\,[-]$ est le nombre de paires de pôles du moteur
$\theta_m\,[rad]$ est la position mécanique du moteur

Cette relation est également valable pour la vitesse (entre ω_e et ω_m). Avec $p = 6$, cela signifie que si le moteur tourne d'un sixième de tour mécaniquement, le signal électrique d'une phase (et donc la tension induite) fait une période complète.

Avec la fonction de transfert du moteur (équation 2.24) et le calcul de la tension induite (équation 2.29), il est possible de calculer le courant électrique circulant dans chacune des phases en fonction de u_p, ω_m et θ_e. Ce courant de phase peut être exprimé en valeur instantanée en fonction de sa valeur efficace (équation 2.31). Le paramètre γ permet de distinguer le déphasage entre courant et tension.

$$i_p(\theta_e) = \sqrt{2} \cdot I_p \cdot sin\left(\theta_e - n \cdot \frac{2\pi}{3} + \gamma\right) \qquad n = 0,1,2 \tag{2.31}$$

Il faut maintenant introduire le couple électromagnétique du moteur, cette grandeur physique correspond à une puissance divisée par une vitesse. Avec des grandeurs électriques, la puissance est donnée par le produit de la tension et du courant.

$$T_{em} = \frac{P_{em}}{\omega} = \frac{U \cdot I \cdot cos(\gamma)}{\omega} \tag{2.32}$$

Où:
$T_{em}\,[Nm]$: est le couple électromagnétique
$P_{em}\,[W]$: est la puissance électromagnétique
$\omega\,[rad/s]$ est la vitesse angulaire
$U\,[V]$ est la tension électrique
$I\,[A]$ est le courant électrique
$\gamma\,[rad]$ est le déphasage entre courant et tension

Dans le cas du moteur triphasé, la puissance est donnée par le courant électrique circulant dans le moteur multiplié par la tension induite de mouvement. L'équation 2.33 correspond au couple moyen généré par une seule phase (courant et tension donnés en valeur efficace).

$$\overline{T_p} = \frac{I_p \cdot U_{be_p} \cdot cos(\gamma)}{\omega_m} \tag{2.33}$$

En remplaçant U_{be_p} avec l'équation 2.28, la vitesse angulaire ω_m disparait de l'équation.

$$\overline{T_p} = \sqrt{\frac{1}{3}} \cdot \frac{I_p \cdot K_e \cdot \omega_m \cdot cos(\gamma)}{\omega_m} = \sqrt{\frac{1}{3}} \cdot I_p \cdot K_e \cdot cos(\gamma) \tag{2.34}$$

Si le courant de phase efficace est identique sur les trois phases, le couple électromagnétique total est égal à trois fois le couple d'une phase (équation 2.35).

$$T_{em} = 3 \cdot \overline{T_p} = \sqrt{3} \cdot I_p \cdot K_e \cdot cos(\gamma) \tag{2.35}$$

Pour la modélisation, le couple doit être exprimé en fonction du courant et de la tension induite en valeur instantanée pour chacune des phases.

$$T_p(\theta_e) = \frac{i_p(\theta_e) \cdot u_{be_p}(\theta_m)}{\omega_m} \tag{2.36}$$

$$= \frac{-i_p(\theta_e) \cdot \sqrt{\frac{2}{3}} \cdot K_e \cdot \omega_m \cdot sin\left(\theta_e - n \cdot \frac{2\pi}{3}\right)}{\omega_m} = -i_p(\theta_e) \cdot \sqrt{\frac{2}{3}} \cdot K_e \cdot sin\left(\theta_e - n \cdot \frac{2\pi}{3}\right) \quad n = 0,1,2$$

La somme du couple électromagnétique en valeur instantanée est la somme des couples de chaque phase.

$$T_{em}(\theta_e) = -\sqrt{\frac{2}{3}} \cdot K_e \cdot \left(i_a(\theta_e) \cdot sin(\theta_e) + i_b(\theta_e) \cdot sin\left(\theta_e - \frac{2\pi}{3}\right) + i_c(\theta_e) \cdot sin\left(\theta_e + \frac{2\pi}{3}\right)\right) \tag{2.37}$$

La constante K_e est utilisée pour exprimer la tension induite en fonction de la vitesse de rotation. Même si les équations 2.35 et 2.37 sont tout à fait correctes, le couple est souvent exprimé à l'aide d'une autre constante. Cette seconde constante du moteur est appelée K_t et elle permet donc de lier le couple au courant électrique. Dans le cas d'un moteur triphasé en étoile, la relation entre K_e et K_t est donnée à l'équation 2.38.

$$K_t = 3 \cdot K_e \tag{2.38}$$

$K_t\,[Nm/A]$ est la constante de couple

Les équations qui définissent le couple en fonction du courant peuvent être exprimées en fonction de K_t:

$$T_{em} = \sqrt{\frac{1}{3}} \cdot I_p \cdot K_t \cdot cos(\gamma) \tag{2.39}$$

$$T_{em}(\theta_e) = -\sqrt{\frac{2}{3}} \cdot \frac{K_t}{3} \cdot \left(i_a(\theta_e) \cdot sin(\theta_e) + i_b(\theta_e) \cdot sin\left(\theta_e - \frac{2\pi}{3}\right) + i_c(\theta_e) \cdot sin\left(\theta_e + \frac{2\pi}{3}\right)\right) \tag{2.40}$$

La constante de couple K_t représente donc le rapport entre le courant électrique et le couple électromagnétique. Si elle n'est pas donnée par le fabricant, elle peut être calculée avec l'équation 2.38 en fonction de K_e.

Cependant, cette constante K_t n'est en réalité pas tout à fait constante. La valeur du couple en fonction du courant varie selon l'intensité du courant électrique dans le moteur. En fait, lorsqu'il y a beacoup de courant, le flux d'induction magnétique à travers le fer du moteur fait apparaître un phénomène de saturation. C'est la perméabilité du matériau qui diminue. Cela a pour effet de diminuer le paramètre K_t. Avec les données du fabricant, cette variation de K_t peut être calculée avec les valeurs nominales et maximales du couple et du courant (table 2.3).

	Couple $[Nm]$	Courant $[A]$	$K_t\,[Nm/A]$ calculé
$3 \cdot K_e$	-	-	2.94
nominal	14.6	$\sqrt{\frac{1}{3}} \cdot 9.1$	2.78
maximal	55	$\sqrt{\frac{1}{3}} \cdot 40.6$	2.35

Table 2.3: Calcul de K_t en fonction des valeurs de couple et de courant données

Pour prendre en compte cette saturation dans la modélisation, le calcul du couple en fonction du courant peut se faire avec une « Lookup table » à partir des données de la table 2.3.

Le couple électromagnétique T_{em} qui dépend de la partie électrique du moteur peut ainsi être modélisé. Mais du point de vue mécanique, il y a d'autres couples à prendre en compte.

Tout d'abord, il y a la masse de la charge qui génère une force de pesanteur et un couple en fonction de la position. Le couple de la charge a été calculé dans un travail précédent ([3], page 42) et vaut:

$$T_{gravity} = 2 \cdot c_f \cdot m_f \cdot g \cdot sin(\theta_m) = 2 \cdot 17.5 \cdot 10^{-3} \cdot 0.066 \cdot 9.81 \cdot sin(\theta_m) \tag{2.41}$$

Où:
$T_{gravity}\,[Nm]$ est le couple généré par la charge sous l'effet de la gravité
$c_f = 17.5 \cdot 10^{-3}\,[m]$ est la distance entre l'axe du moteur et le centre de masse de la charge
$m_f = 0.066\,[kg]$ est la masse de la charge
$g = 9.81\,[m/s^2]$ est la constante de gravité de la terre
$\theta_m\,[rad]$ est la position mécanique du moteur

En plus du couple électromagnétique et de celui de la charge, il y a le frein magnétique qui génère aussi un couple. Ce couple est utilisé pour replacer correctement la position du moteur lorsque la régulation de position est désactivée. La courbe du couple en fonction de la position est donnée par une « Lookup Table » et est visible sur la figure 2.27. En sachant qu'un couple positif contribue à faire augmenter la position et qu'un couple négatif contribue à la diminuer, il y a deux points d'équilibres, à 0° et 180°. Il s'agit des positions de départ et d'arrivée où il est justement souhaitable que le rotor se repositionne automatiquement.

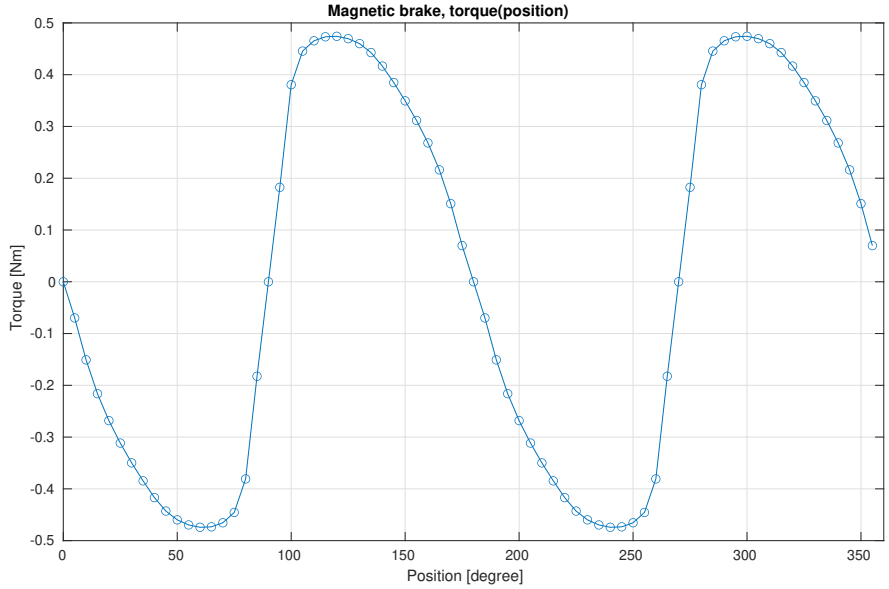


Figure 2.27: Couple du frein magnétique en fonction de la position
`dev/90_test/10_plant_model/motor_model/magnetic_brake/magnetic_brake.m`

La courbe visible à la figure 2.27 est définie avec un pas de 5° entre chaque valeur de position. Afin d'obtenir une plus grande précision, il est possible d'approximer cette courbe avec l'algorithme « spline » au lieu d'une simple interpolation linéaire.

La figure 2.28 compare l'interpolation « spline » avec une simple interpolation linéaire. Cet algorithme « spline » est un peu plus compliqué mais celui-ci donne un résultat plus pertinent pour ce type de courbe.

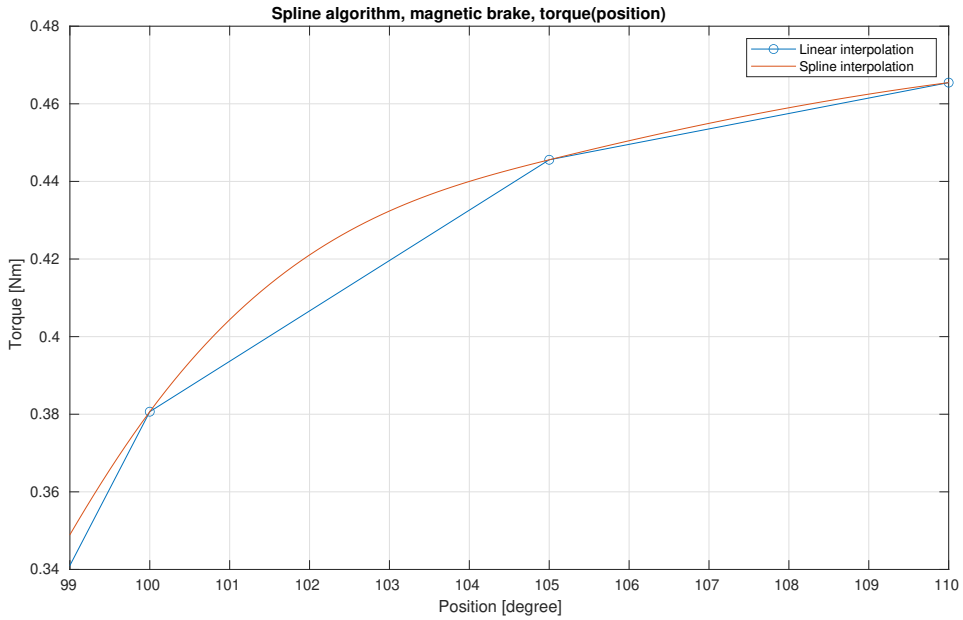


Figure 2.28: Comparaison entre « spline » et l'interpolation linéaire
`dev/90_test/10_plant_model/motor_model/magnetic_brake/magnetic_brake.m`

Les couples électromagnétiques, le couple de la gravité et le couple du frein magnétique peuvent être sommés ensemble.

$$T_{sum} = T_{em} + T_{gravity} + T_{brake} \tag{2.42}$$

Pour compléter le calcul du couple, il reste encore le couple de friction à prendre en compte. Celui-ci est différent dans le sens où il dépend des autres couples en s'y opposant.

$$T_{tot} = T_{sum} + T_{friction}(T_{sum}) \tag{2.43}$$

Pour calculer le couple de friction, il faut distinguer le cas statique du cas dynamique. En statique, le couple de friction annule la somme des autres couples tant que celle-ci est inférieure à la constante de friction statique T_{sf}. En dynamique, le couple de friction varie en fonction de la vitesse de rotation et il s'oppose au sens de rotation. Les différents cas sont listés à l'équation 2.44.

$$\begin{cases} T_{friction} = T_{sf} & \text{if } (\omega_m == 0) \text{ and } (T_{sum} < -T_{sf}) \\ T_{friction} = -T_{sf} & \text{if } (\omega_m == 0) \text{ and } (T_{sum} > T_{sf}) \\ T_{friction} = -T_{sum} & \text{if } (\omega_m == 0) \text{ and } (-T_{sf} < T_{sum} < T_{sf}) \\ T_{friction} = -T_{df} - \mu_v \cdot \omega_m & \text{if } (\omega_m > 0) \\ T_{friction} = T_{df} - \mu_v \cdot \omega_m & \text{if } (\omega_m < 0) \end{cases} \tag{2.44}$$

Où:
T_{sf} $[Nm]$ est le couple de friction statique
T_{df} $[Nm]$ est le couple de friction dynamique initial
μ_v $[Nm/(rad/s)]$ est le coefficient de friction cinétique

Des mesures ont été effectuées au CERN dans le cadre d'autres projets afin de déterminer les valeurs numériques des constantes ci-dessus. Dans le cas statique, les résultats des mesures sont trop dispersés pour pouvoir s'y fier. En revanche, dans le cas dynamique, le couple de friction a pu être linéarisé et les valeurs numériques sont données au tableau 2.4. Par manque d'informations, le couple de friction statique (T_{sf}) est fixé arbitrairement à la même valeur que T_{df}.

Couple de friction statique (T_{sf})	0.197 $[Nm]$
Couple de friction dynamique initial (T_{df})	0.197 $[Nm]$
Coefficient de friction cinétique (μ_v)	0.023 $[Nm/(rad/s)]$

Table 2.4: Constantes du couple de friction

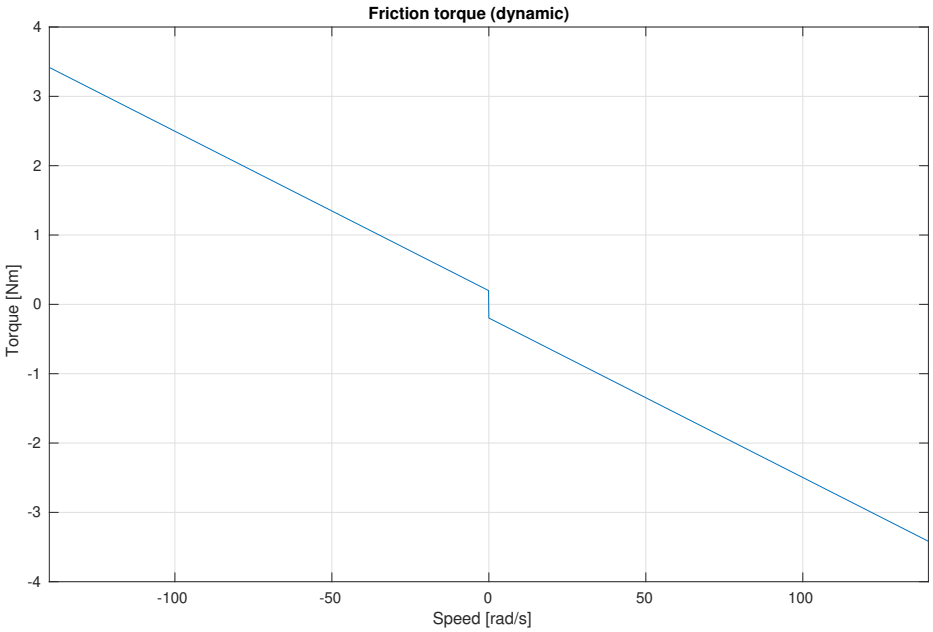


Figure 2.29: Couple de friction dynamique en fonction de la vitesse
`dev/90_test/10_plant_model/motor_model/friction/show_friction.m`

À partir du couple total, l'accélération du rotor se calcule avec l'inertie de la charge. Dans ce travail, l'inertie est estimée à:

$$J = 3.01 \cdot 10^{-3} \, [kg \cdot m^2] \tag{2.45}$$

L'accélération est définie avec l'équation 2.46.

$$\alpha(t) = \frac{T_{tot}(t)}{J} = \frac{T_{tot}(t)}{3.01 \cdot 10^{-3}} \, [m/s^2] \tag{2.46}$$

La vitesse correspond à l'intégrale de l'accélération, équation 2.47. De la même manière, la position correspond à l'intégrale de la vitesse, équation 2.48.

$$\omega_m(t) = \int \alpha(t) \, dt + \omega_{m_0} \tag{2.47}$$

$$\theta_m(t) = \int \omega_m(t) \, dt + \theta_{m_0} \tag{2.48}$$

La vitesse initiale ω_{m_0} est toujours nulle. En revanche, la position initiale θ_{m_0} n'est pas nécessairement nulle. Pour le mouvement de retour, la position initiale doit être mise à π (180°). De plus, il serait possible de simuler le système à régler en prenant en compte une erreur de position initiale afin d'en observer les conséquences sur la régulation de position.

2.2.3.2 Transformée de Park

Dans la section précédente (2.2.3.1), les équations qui lient les grandeurs électriques aux grandeurs mécaniques ont été exprimées indépendamment pour chaque phase. La contribution au couple pour chacune des phases dépend de la position relative du rotor et nécessite donc des équations exprimées en valeur instantanée en fonction de la position. Avec l'équation 2.40 qui exprime le couple électromagnétique total en fonction des courants de chaque phase et de la position électrique, il est possible de définir le courant "utile", nommé i_q.

$$T_{em}(\theta_e) = \frac{K_t}{3} \cdot i_q(\theta_e) \tag{2.49}$$

$$i_q(\theta_e) = -\sqrt{\frac{2}{3}} \cdot \left(i_a(\theta_e) \cdot sin(\theta_e) + i_b(\theta_e) \cdot sin\left(\theta_e - \frac{2\pi}{3}\right) + i_c(\theta_e) \cdot sin\left(\theta_e + \frac{2\pi}{3}\right)\right) \tag{2.50}$$

Si la commande est générée en prenant en compte la position du rotor et le déphasage de chaque phase, la somme des courants "utile" (définie ci-dessus comme "i_q") peut être invariante en fonction de la position afin de générer une accélération constante. Vu de cette manière, il est possible de calculer directement le courant "utile" sans passer par les courants de chacune des phases.

Dans la géométrie du moteur, le courant i_q peut être vu comme un vecteur qui est déphasé de 90° par rapport à la position du rotor. L'amplitude de ce vecteur dépend de la valeur calculée avec l'équation 2.50.

Puisque i_q définit le courant ayant un impact sur le couple, il faut définir l'autre vecteur complémentaire qui se nomme i_d. Celui-ci n'a aucun impact direct sur le couple. Géométriquement, ce vecteur est aligné à la position du rotor. Ces deux vecteurs sont représentés géométriquement sur la figure 2.30.

Le courant n'est pas la seule grandeur à pouvoir être calculée de cette manière. Cette représentation (appelée "dq" dans ce travail) peut être étendue à d'autres grandeurs triphasées comme, par exemple, la tension de commande.

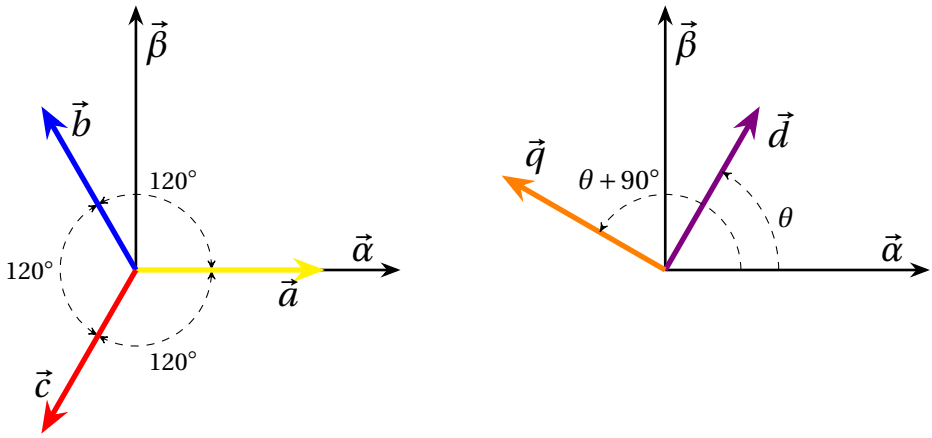


Figure 2.30: Comparaison géométrique entre la représentation triphasée et dq

La figure 2.30 montre les deux manières de représenter les grandeurs de courant ou de tension dans un moteur triphasé. Le passage de la représentation triphasée à la représentation dq dépend de la position θ du moteur. Le calcul est donné par la transformée de Park (équation 2.51).

$$\begin{bmatrix} d \\ q \\ 0 \end{bmatrix} = \sqrt{\frac{2}{3}} \cdot \begin{bmatrix} cos(\theta) & cos\left(\theta - \frac{2\pi}{3}\right) & cos\left(\theta + \frac{2\pi}{3}\right) \\ -sin(\theta) & -sin\left(\theta - \frac{2\pi}{3}\right) & -sin\left(\theta + \frac{2\pi}{3}\right) \\ \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{2.51}$$

Si les phases sont correctement équilibrées (moyenne nulle entre les trois phases), alors la composante 0 est toujours nulle. Le calcul de la transformée de Park peut passer par l'intermédiaire des composantes α et β (visibles sur la figure 2.30). Cette étape intermédiaire s'appelle la transformée de Clarke (équation 2.52).

$$\begin{bmatrix} \alpha \\ \beta \\ 0 \end{bmatrix} = \sqrt{\frac{2}{3}} \cdot \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{2.52}$$

$$\begin{bmatrix} d \\ q \\ 0 \end{bmatrix} = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ 0 \end{bmatrix} \tag{2.53}$$

Pour convertir les grandeurs d et q vers les grandeurs triphasées, c'est la transformée inverse de Park qui est utilisée (équation 2.54).

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \sqrt{\frac{2}{3}} \cdot \begin{bmatrix} cos(\theta) & -sin(\theta) & \sqrt{\frac{1}{2}} \\ cos\left(\theta - \frac{2\pi}{3}\right) & -sin\left(\theta - \frac{2\pi}{3}\right) & \sqrt{\frac{1}{2}} \\ cos\left(\theta + \frac{2\pi}{3}\right) & -sin\left(\theta + \frac{2\pi}{3}\right) & \sqrt{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} d \\ q \\ 0 \end{bmatrix} \tag{2.54}$$

Elle peut également être calculée par l'intermédiaire de la transformée inverse de Clarke (équation 2.56).

$$\begin{bmatrix} \alpha \\ \beta \\ 0 \end{bmatrix} \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d \\ q \\ 0 \end{bmatrix} \tag{2.55}$$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \sqrt{\frac{2}{3}} \cdot \begin{bmatrix} 1 & 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ 0 \end{bmatrix} \tag{2.56}$$

L'équation de la transformée de Park (équation 2.51) permet de retrouver le résultat obtenu à l'équation 2.50. C'est-à-dire:

$$q(\theta_e) = -\sqrt{\frac{2}{3}} \cdot \left(a \cdot sin(\theta_e) + b \cdot sin\left(\theta_e - \frac{2\pi}{3}\right) + c \cdot sin\left(\theta_e + \frac{2\pi}{3}\right)\right)$$

La composante complémentaire à q qui est la composante d peut s'exprimer ainsi:

$$d(\theta_e) = \sqrt{\frac{2}{3}} \cdot \left(a \cdot cos(\theta_e) + b \cdot cos\left(\theta_e - \frac{2\pi}{3}\right) + c \cdot cos\left(\theta_e + \frac{2\pi}{3}\right)\right)$$

Comme expliqué auparavant, le courant i_d n'a aucun impact direct sur le couple. Cependant, il est quand même important de le prendre en compte lors d'une modélisation. Ce courant i_d représente une quantité de courant dans le moteur. Même si celui-ci ne crée pas de mouvement, il crée un flux magnétique dans le moteur qui crée une tension induite sur la phase q. De la même manière, le courant de la phase q a une influence sur la phase d. Le calcul du flux dans le moteur n'est pas étudié dans ce travail. Les équations qui définissent les tensions induites sur les phases d et q sont directement données ci-dessous:

$$U_{be_d} = -L_s \cdot i_q \cdot \omega_m \cdot p \tag{2.57}$$

$$U_{be_q} = K_e \cdot \omega_m + L_s \cdot i_d \cdot \omega_m \cdot p \tag{2.58}$$

Dans cette section, les différentes grandeurs représentées sous différentes formes peuvent facilement entraîner une confusion. Dans le but de clarifier ces différentes représentations, un résumé à la table 2.5 montre les équivalences entre chaque manière de représenter ces grandeurs. La phase d n'est pas prise en compte ci-dessous et le couplage (équation 2.58) est donc simplifié.

RMS value (q-phase)	RMS value (one phase)	Angle variant value (one phase), n = 0,1,2
$U_q = \sqrt{3} \cdot U_p$	U_p	$u_p(\theta_e) = U_p \cdot \sqrt{2} \cdot sin\left(\theta_e - n \cdot \frac{2\pi}{3}\right)$
$I_q = \sqrt{3} \cdot I_p \cdot cos(\gamma)$	I_p	$i_p(\theta_e) = I_p \cdot \sqrt{2} \cdot sin\left(\theta_e - n \cdot \frac{2\pi}{3} + \gamma\right)$
$U_{be_q} = K_e \cdot \omega_m$	$U_{be_p} = \sqrt{\frac{1}{3}} \cdot K_e \cdot \omega_m$	$u_{be_p}(\theta_e) = \sqrt{\frac{2}{3}} \cdot K_e \cdot \omega_m \cdot sin\left(\theta_e - n \cdot \frac{2\pi}{3}\right)$
$T_{em_q} = \frac{K_t}{3} \cdot I_q$	$\overline{T_p} = \sqrt{\frac{1}{3}} \cdot \frac{K_t}{3} \cdot I_p \cdot cos(\gamma)$	$T_p(\theta_e) = \sqrt{\frac{2}{3}} \cdot \frac{K_t}{3} \cdot i_p(\theta_e) \cdot sin\left(\theta_e - n \cdot \frac{2\pi}{3}\right)$

Table 2.5: Équivalence entre valeur RMS de la phase q et les valeurs instantanées d'une seule phase

2.2.3.3 Modélisation du bloc « Motor »

Il existe deux manières de modéliser le moteur. La première consiste à calculer le courant dans chaque phase indépendamment. À partir de ces courants, le couple de chaque phase peut être calculé en fonction de la position du rotor en prenant en compte le déphasage de chaque phase.

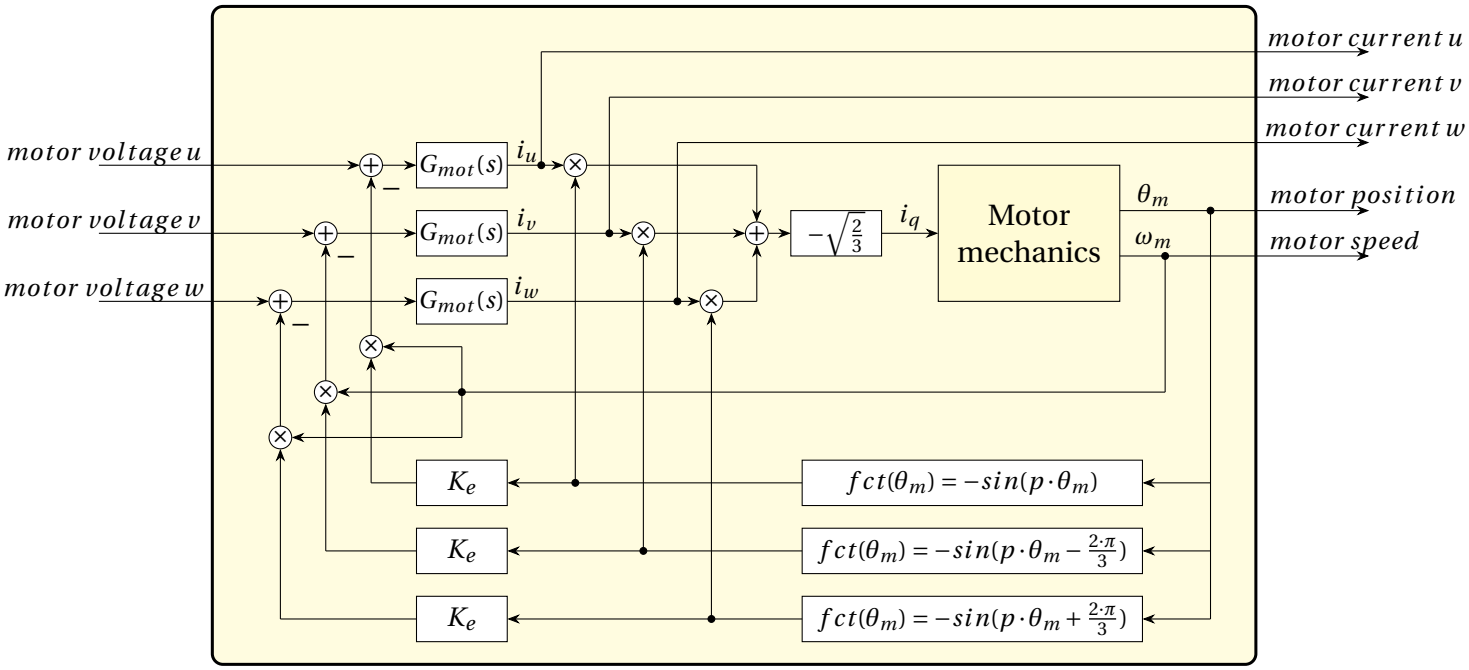


Figure 2.31: Modélisation du moteur

Le courant iq est le courant "utile" décrit à la section 2.2.3.2 qui permet d'avoir un impact sur la partie mécanique. La modélisation de la partie mécanique du moteur (bloc « Motor mechanics ») est donnée à la figure 2.32.

Sur la figure 2.32, le bloc LUT_{Tem} correspond à l'équation 2.49 où K_t est défini avec une « Lookup Table » en fonction de la table 2.3 pour prendre en compte la saturation du fer.

Le bloc $Friction$ correspond aux calculs conditionnels de l'équation 2.44.

Le bloc LUT_{Tbrake} correspond à la « Lookup Table » montrée à la figure 2.27 qui doit être interpolée avec l'algorithme « spline ».

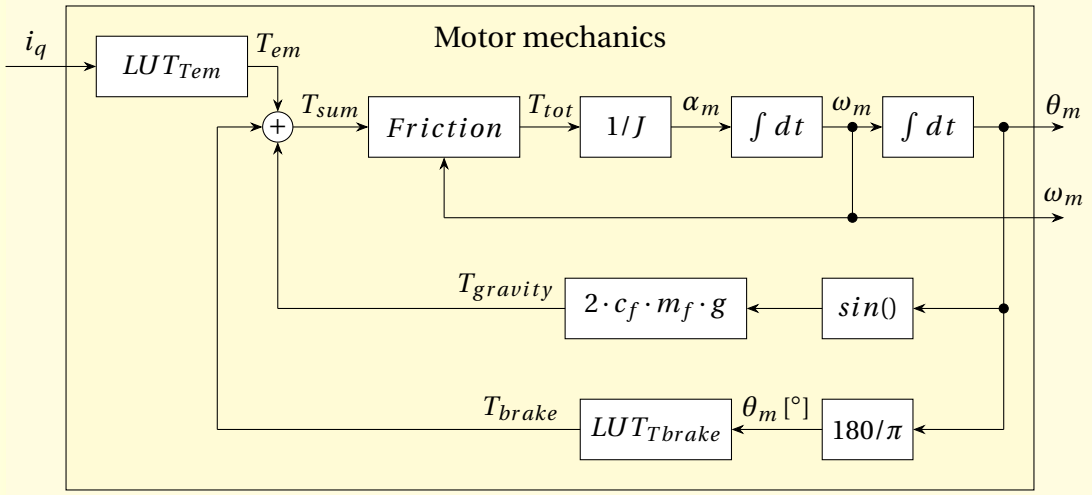


Figure 2.32: Modélisation du moteur (mécanique)

La seconde manière de modéliser le moteur est d'utiliser la transformée de Park pour effectuer les calculs sur deux phases. La phase q correspond à la composante utile qui permet de générer un couple sur le moteur. L'autre phase d ne génère pas de couple utile mais sa modélisation est importante pour prendre en compte les effets de couplage.

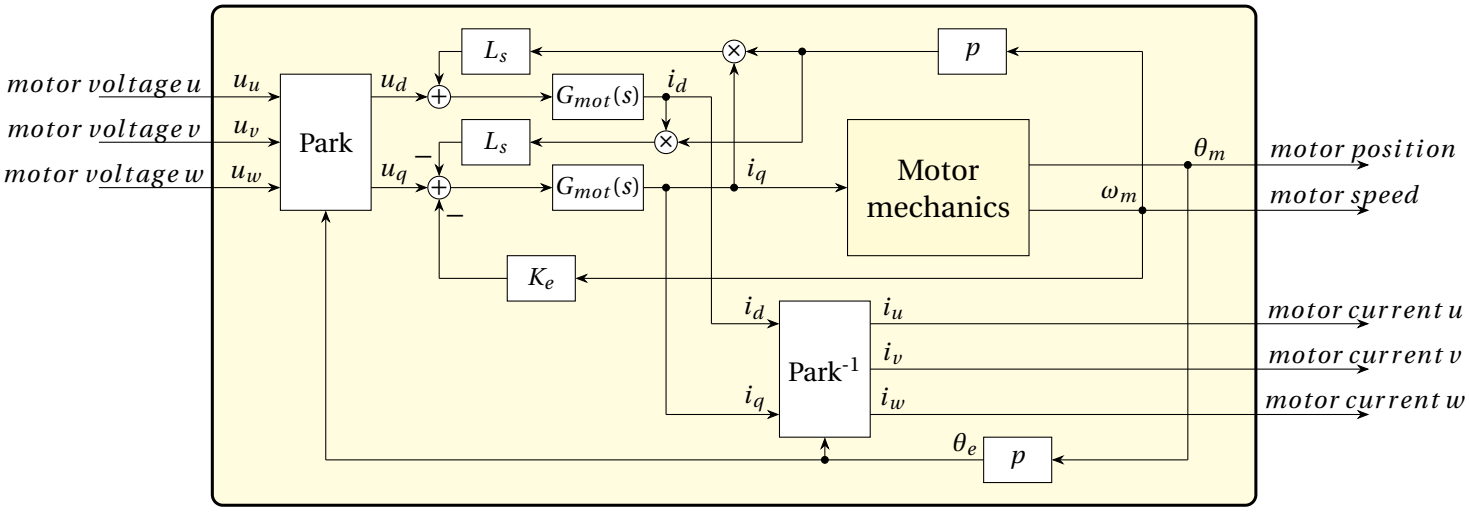


Figure 2.33: Modélisation du moteur

2.2.3.4 Comparaison des modèles du moteur

Pour vérifier que les deux modèles (visibles aux figures 2.31 et 2.33) ont bien un gain équivalent, il suffit de simuler les deux systèmes en parallèle avec la même commande et comparer les signaux de sortie. Cette commande a été générée et sauvegardée lors d'une simulation en boucle fermée (fichier ci-dessous) et correspond donc à un cas réaliste.

`dev/01_plant_model/simulink_model/08_analogic_modified/analogic_modified_AY8.m`

La commande utilisée pour stimuler le système à régler est visible à la figure 2.34

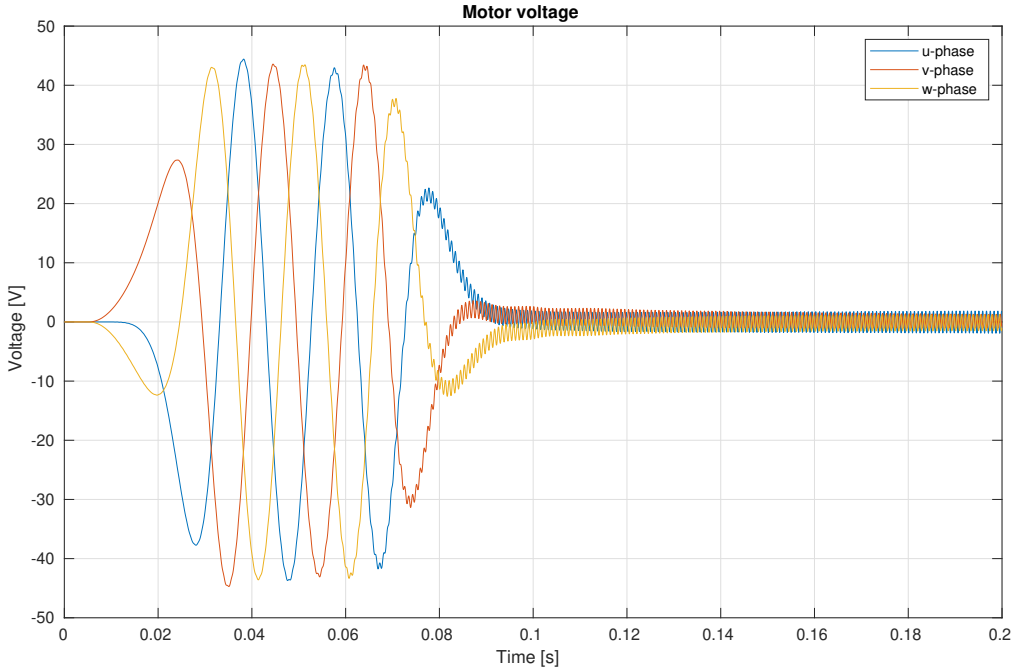


Figure 2.34: Tensions d'entrée pour la comparaison des modèles du moteur
`dev/90_test/10_plant_model/motor_model/motor_model_compare/motor_model_compare.m`

Les signaux de sortie sont comparés sur les figures 2.35, 2.36 et 2.37. Pour chaque grandeur mesurée, l'erreur MSE (« Mean squared error ») entre les signaux des deux modèles est calculée. Pour mieux visualiser à quel endroit de la trajectoire l'erreur est la plus grande, l'erreur au carré est affichée sur un graphique pour chaque comparaison.

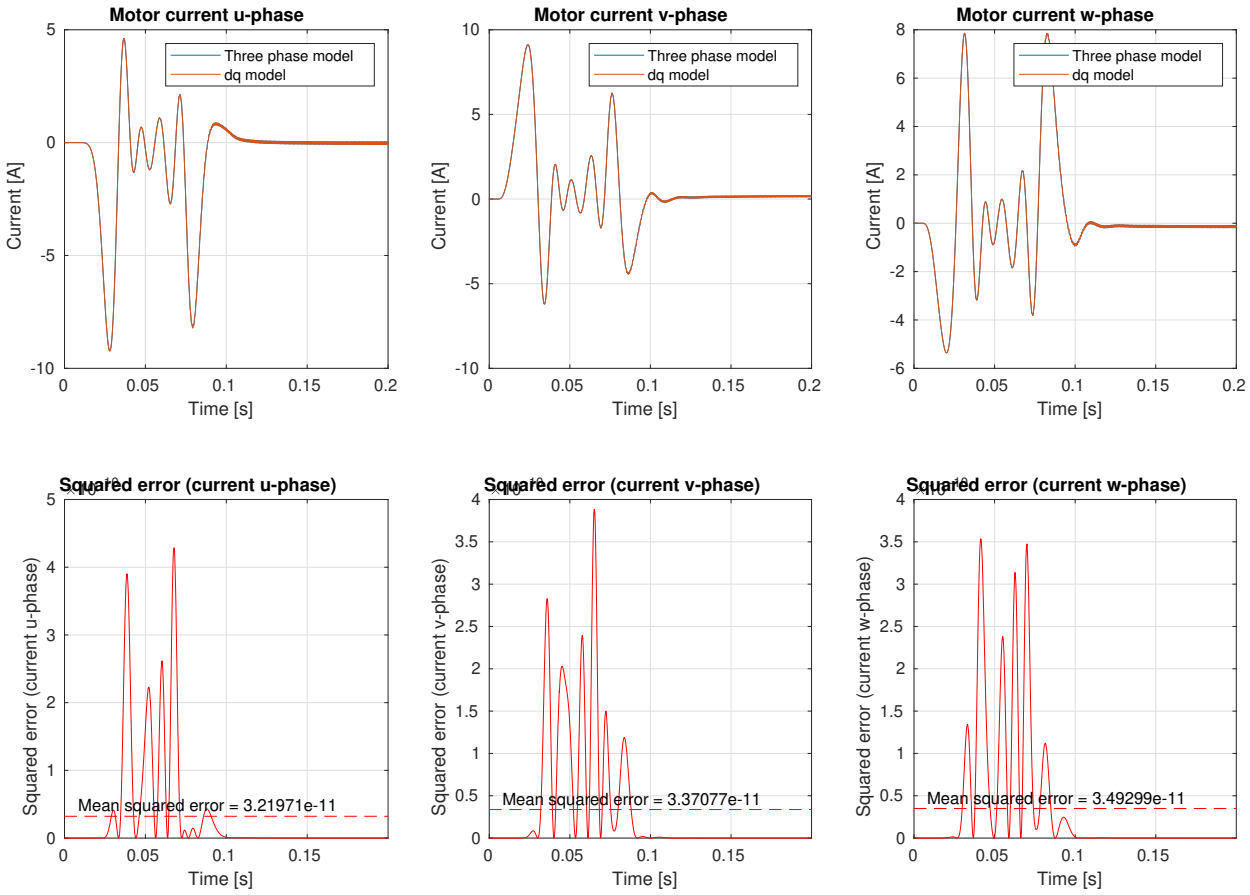


Figure 2.35: Comparaison des modèles du moteur (courants triphasés)
`dev/90_test/10_plant_model/motor_model/motor_model_compare/motor_model_compare.m`

Sur la figure 2.35, ce sont les courants de chaque phase du moteur qui sont affichés. Dans le cas du modèle triphasé, chaque courant est calculé indépendamment en fonction de la tension de phase, et de la position. Pour le modèle dq, les trois courants de phase sont obtenus avec une transformée de Park inverse à partir des courants i_d et i_q et de la position.

Les résultats ci-dessus montre une erreur très faible et cela prouve que le calcul des courants de phase est équivalent pour les deux modèles. De plus, le calcul du courant dépend de la tension de commande et de la tension induite. Les résultats valident également le calcul de la tension induite pour les deux modèles.

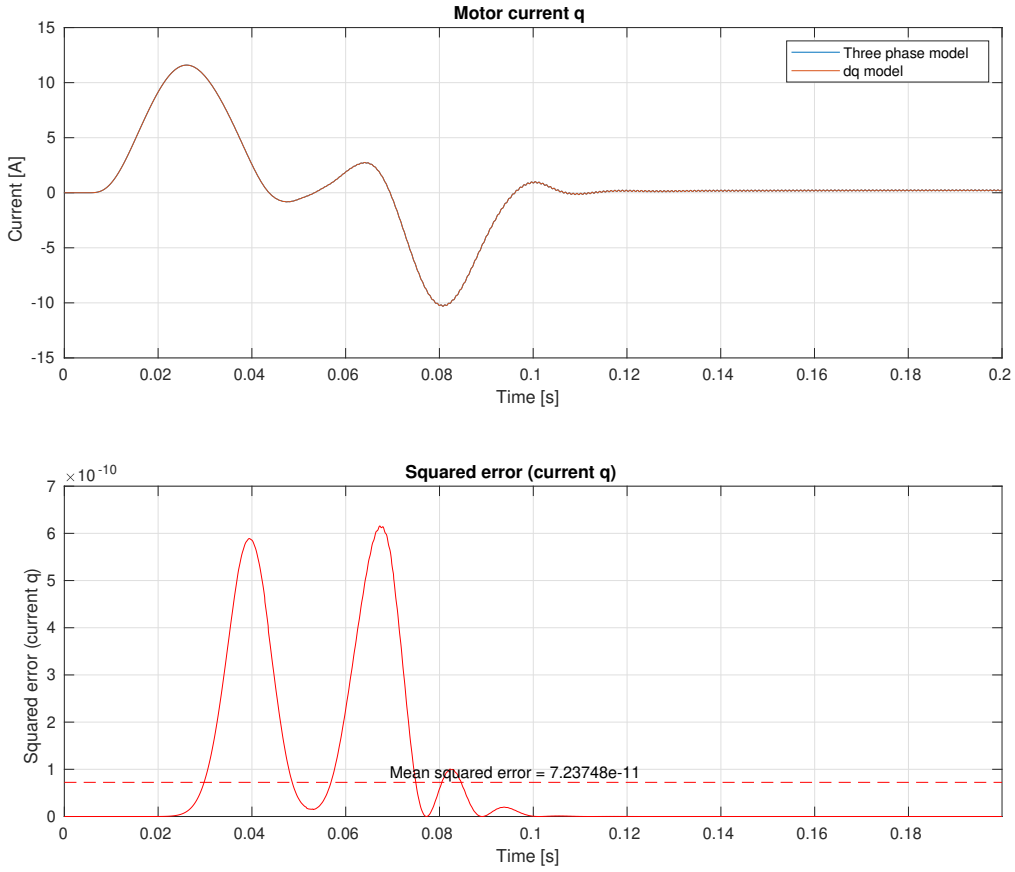


Figure 2.36: Comparaison des modèles du moteur (courant q)
`dev/90_test/10_plant_model/motor_model/motor_model_compare/motor_model_compare.m`

La figure 2.36 montre le courant i_q. Dans le modèle triphasé, ce courant i_q doit être calculé à partir des trois phases en prenant en compte la géométrie du moteur. Pour le modèle dq, le courant i_q est calculé directement avec la fonction de transfert G_{mot}.

Les résultats ci-dessus montrent une erreur très faible et cela permet de valider le calcul de i_q entre les deux modèles.

Sur la figure 2.37, ce sont la vitesse et la position des deux modèles qui sont comparées. Ces deux grandeurs dépendent de la partie mécanique (qui est identique pour les deux modélisations) et du courant i_q. Puisque le courant i_q a déjà pu être validé à la figure 2.36, il n'est pas surprenant que les erreurs de vitesse et de position entre les deux modèles soient également très faibles.

Les résultats obtenus dans cette simulation montrent bien que les gains entre les deux types de modélisation sont identiques. Mais il est important de noter que la simulation présente ne tient pas compte de la dynamique du système.

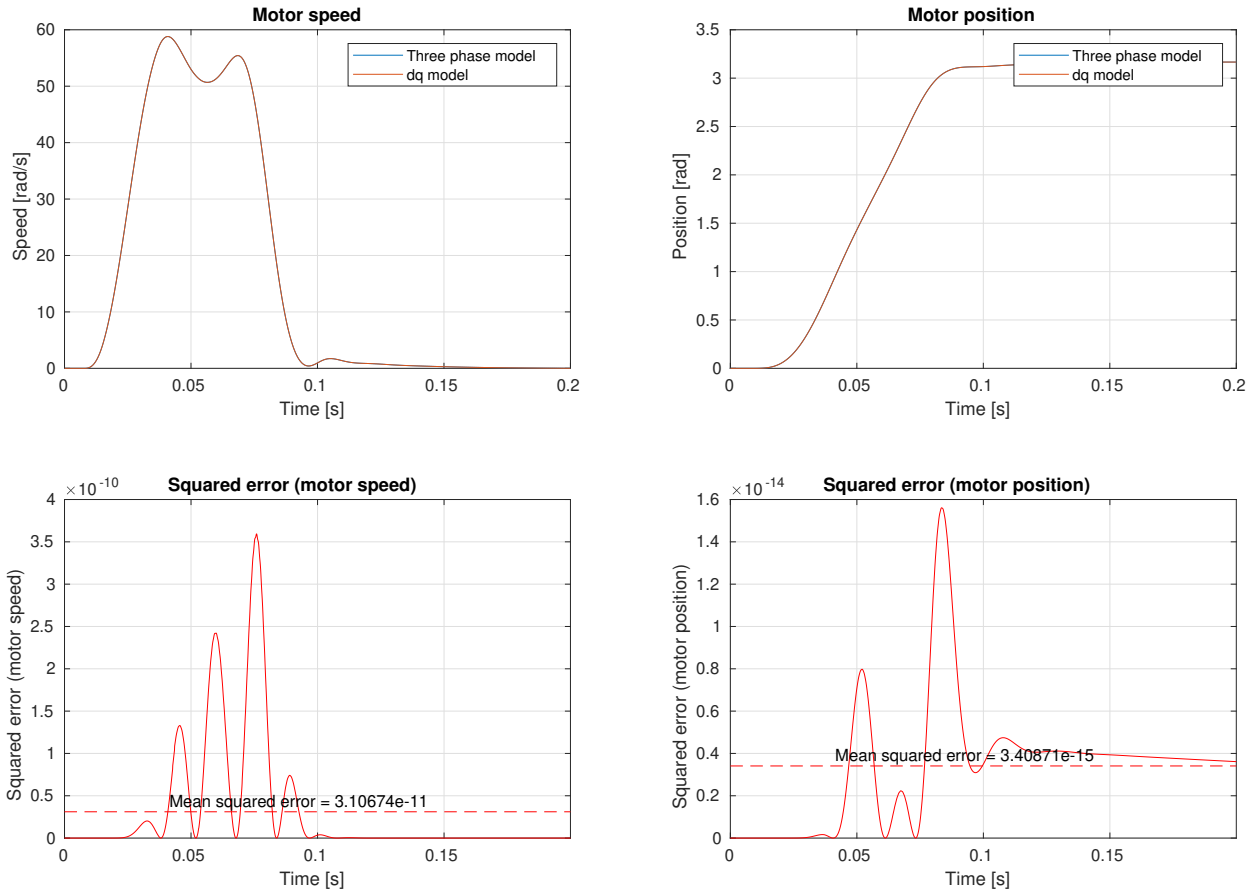


Figure 2.37: Comparaison des modèles du moteur (vitesse et position)
`dev/90_test/10_plant_model/motor_model/motor_model_compare/motor_model_compare.m`

Pour la suite de ce travail, c'est le modèle triphasé qui sera utilisé. Cela permet d'observer chacune des phases indépendamment des autres.

De plus, avec le modèle triphasé, la transformée de Park n'a pas besoin d'être utilisée dans le système à régler. Celle-ci ne pose aucun problème dans le cas analogique avec une dynamique "parfaite" mais dans le cas de la modélisation discrète, la transformée de Park pourrait provoquer des erreurs. En effet, la discrétisation peut entraîner de légers retards entre chaque fonction de transfert. Puisque la transformée de Park utilise la position du moteur qui est calculée à travers des fonctions de transfert (et donc retardée), les tensions de commande dq qui dépendent de la position peuvent être légèrement faussées. Cependant, avec les méthodes de discrétisation utilisées par la suite, le retard lié à la discrétisation est très limité et ce phénomène n'est donc pas très important.

De toute manière, l'utilisation du modèle triphasé permet de ne pas se soucier du problème de la transformée de Park et c'est pourquoi ce problème n'est pas présenté en détail dans ce document.

2.2.4 Bloc « Sensor »

Le régulateur a besoin d'un « feedback » pour connaître la position, la vitesse et le courant électrique du moteur. Ce bloc « Sensor » permet de modéliser les capteurs qui permettent d'effectuer les mesures de ces grandeurs. La mesure du courant électrique nécessite un capteur de courant pour chacune des phases. La position et la vitesse peuvent être mesurées avec un seul capteur qui est le resolver.

2.2.4.1 Capteur de courant

En réalité, les capteurs de courant sont situés juste après le filtre et donc avant les câbles (à l'intérieur du bloc « Filter »). Mais puisque les câbles ne sont pas pris en compte dans le modèle, les courants à la sortie du filtre correspondent aux courants circulant dans le moteur. Ceux-ci sont calculés et fournis par le bloc « Motor ». Il est donc justifié de modéliser les capteurs de courant après le bloc « Motor ».

Le capteur de courant utilisé est le LEM LA 100-P. Il a été étudié et modélisé dans un travail précédent ([3], page 40). Il s'agit d'un filtre passe-bas du premier ordre dont la constante de temps est donnée à l'équation 2.59. La fonction de transfert correspondante est donnée à l'équation 2.60.

$$\tau_c = \frac{1}{2 \cdot \pi \cdot f_c} = \frac{1}{2 \cdot \pi \cdot 350 \cdot 10^3} = 455\,[ns] \tag{2.59}$$

$$G_{cur\_sens}(s) = \frac{1}{\tau_c \cdot s + 1} = \frac{2.2 \cdot 10^6}{s + 2.2 \cdot 10^6} \tag{2.60}$$

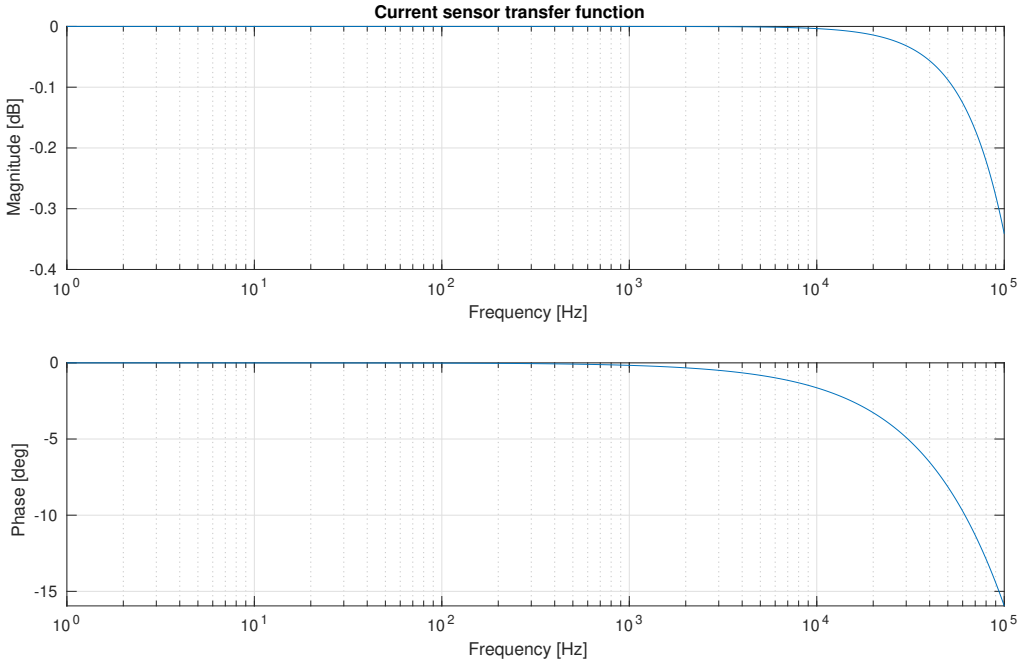


Figure 2.38: Diagramme de Bode de la modélisation du capteur de courant

2.2.4.2 Resolver

La modélisation du resolver a également été étudiée auparavant ([3], page 31). Comme expliqué à la section 2.1.4, la mesure de la position et de la vitesse nécessite l'utilisation d'un RDC (« Resolver-to-Digital Converter »). La résolution du composant utilisé (AD2S1210 de Analog Devices) est configurable (10, 12, 14 ou 16 bits). Une plus grande précision nécessite un temps de calcul plus élevé et impacte la dynamique du système. Dans le système actuel, il est configuré en 14 bits et son modèle est donné directement sous forme de fonction de transfert avec les valeurs numériques à l'équation 2.61.

$$G_{resolver}(s) = \frac{2.03 \cdot 10^{-5} \cdot s^2 + 0.1372 \cdot s + 63.57}{5.031 \cdot 10^{-14} \cdot s^4 + 6.298 \cdot 10^{-10} \cdot s^3 + 2.227 \cdot 10^{-5} \cdot s^2 + 0.1372 \cdot s + 63.57} \tag{2.61}$$

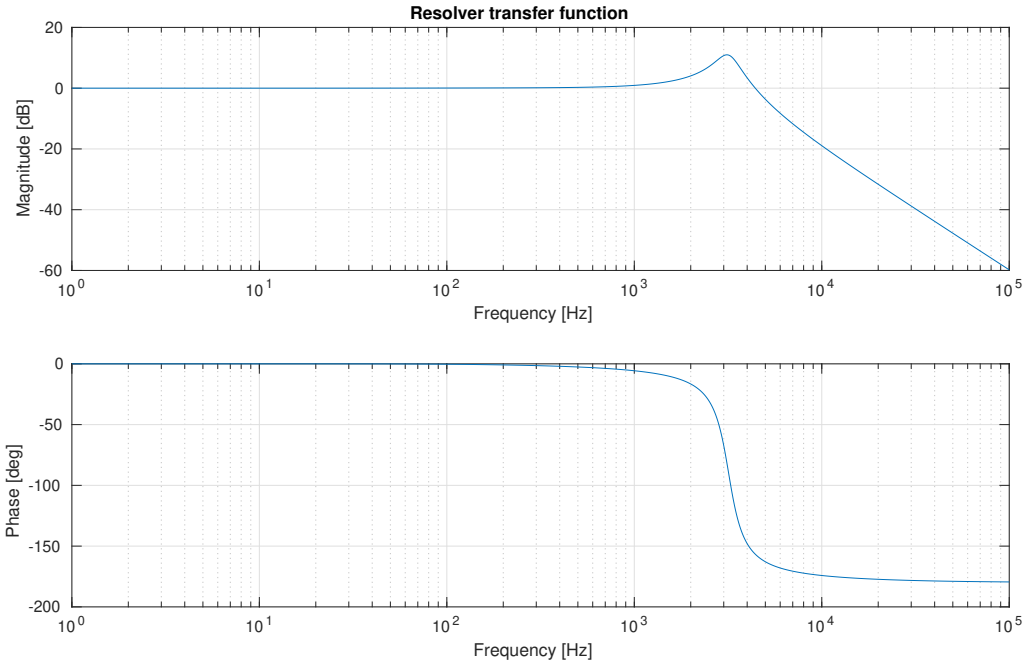


Figure 2.39: Diagramme de Bode de la modélisation du resolver

Cette fonction de transfert (équation 2.61) a été calculée pour le capteur de position mais elle est aussi valable pour le capteur de vitesse.

Puisque le resolver est configuré pour une précision de 14 bits, il faut aussi modéliser cette quantification. Avec 14 bits, la résolution de la position est donnée à l'équation 2.62.

$$\theta_{res} = \frac{2 \cdot \pi}{2^{14}} = 383 \cdot 10^{-3} \, [rad] \tag{2.62}$$

La quantification du signal peut être réalisée avec le calcul de l'équation 2.63.

$$\theta_{out} = \theta_{res} \cdot floor\left(\frac{\theta_{in}}{\theta_{res}}\right) \tag{2.63}$$

2.2.4.3 Modélisation du bloc « Sensor »

Les capteurs de courant doivent être modélisés sur les trois phases avec la fonction de transfert de l'équation 2.60. Les capteurs de position et de vitesse sont modélisés avec la fonction de transfert de l'équation 2.61. Pour la position, il faut aussi ajouter la quantification avec le calcul de l'équation 2.63.

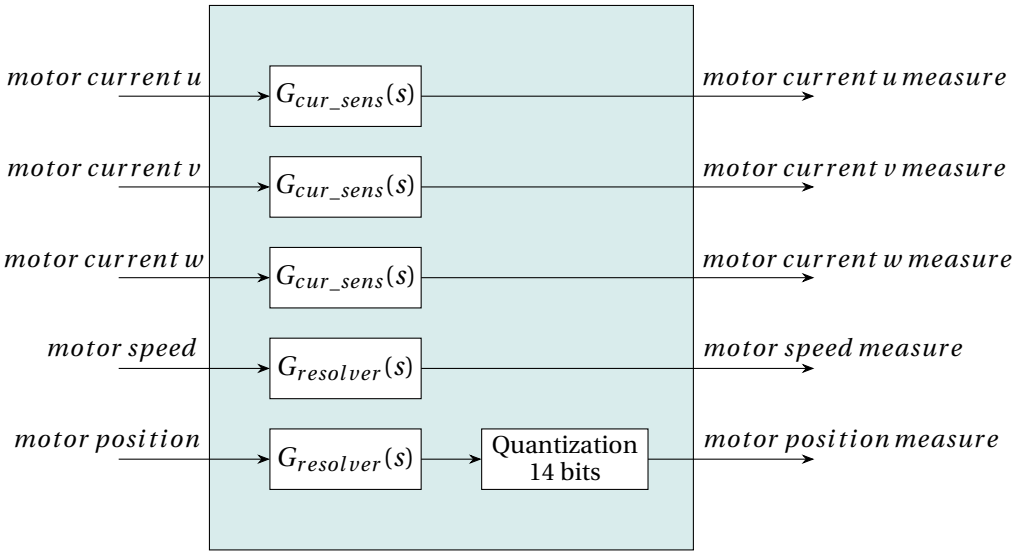


Figure 2.40: Modélisation du bloc « Sensor »

2.3 Discrétisation

Dans la section 2.2, le système à régler a été modélisé avec des fonctions de transfert analogiques (fonction en "s") qui peuvent être simulées à l'aide d'un outil de simulation à pas variable. Une simulation à pas variable ajuste la fréquence d'échantillonnage du système simulé de manière à trouver un compromis entre précision et performance (temps de simulation). Losque le signal subit de fortes variations, la fréquence d'échantillonnage augmente pour obtenir une meilleure précision. Dans le cas inverse, lorsque le signal subit peu de variations, la fréquence d'échantillonnage diminue pour ne pas sur-échantillonner inutilement.

Simulink est un outil de simulation à pas variable (méthode ode23 par défaut). Le système à régler modélisé à la section 2.2 peut donc être simulé en combinant une bonne précision et une bonne performance grâce à Simulink. Cependant, le régulateur à vérifier est implémenté en VHDL et il est donc préférable que le système à régler soit également simulé en VHDL ou en SystemVerilog.

Dans ce type de langage de description de matériel, le temps est géré avec une horloge de fréquence fixe. Les signaux sont synchronisés sur l'horloge à l'aide de bascules D. Ces langages offrent aussi la possibilité d'utiliser du code non-synthétisable qui offre des fonctionnalités un peu plus permissives. Mais même avec du code non-synthétisable, il n'est pas possible de réaliser une simulation à pas de temps variable dans ce type de langage.

Pour simuler la dynamique d'un système physique en VHDL ou en SystemVerilog, il est nécessaire de modéliser ce système de manière à ce qu'il soit évalué à intervalle de temps fixe (synchronisé sur une horloge). Cette section a pour but de discrétiser les fonctions de transfert analogiques de la section 2.2 afin d'obtenir des fonctions de transfert discrètes (fonction en "z") qui puissent être simulées avec un pas de temps fixe.

2.3.1 Fonction de transfert discrète

Une fonction de transfert discrète d'ordre N s'exprime sous la forme suivante:

$$G(z) = \frac{Y(z)}{U(z)} = \frac{b_0 \cdot z^0 + b_1 \cdot z^{-1} + \cdots + b_N \cdot z^{-N}}{a_0 \cdot z^0 + a_1 \cdot z^{-1} + \cdots + a_N \cdot z^{-N}} = \frac{b_0 \cdot z^N + b_1 \cdot z^{N-1} + \cdots + b_N \cdot z^0}{a_0 \cdot z^N + a_1 \cdot z^{N-1} + \cdots + a_N \cdot z^0} \tag{2.64}$$

Où:
z est la variable discrète (équivalent de "s" dans le domaine continu)
$Y(z)$ est la transformée en Z du signal de sortie
$U(z)$ est la transformée en Z du signal d'entrée
b_n sont les coefficients au numérateur
a_n sont les coefficients au dénominateur

L'équation 2.64 peut aussi s'écrire avec la forme ci-dessous:

$$Y(z) \cdot \left(a_0 \cdot z^0 + a_1 \cdot z^{-1} + \cdots + a_N \cdot z^{-N}\right) = U(z) \cdot \left(b_0 \cdot z^0 + b_1 \cdot z^{-1} + \cdots + b_N \cdot z^{-N}\right) \tag{2.65}$$

Puisque les fonctions de transfert discrètes agissent avec des signaux discrets, ces derniers ne doivent pas être exprimés en fonction du temps mais plutôt avec des indices d'échantillons discrets. Le passage du temps continu aux échantillons discrets se fait avec l'équation 2.66.

$$t = k \cdot h\,[s] \tag{2.66}$$

Où:
$t\,[s]$ est le temps continu
$k \in \mathbb{Z}$ est l'indice de l'échantillon d'un signal
$h\,[s]$ est la période d'échantillonnage

En sachant que z^{-n} correspond à un retard de n échantillons sur un signal numérique, l'équation 2.65 peut être exprimée en fonction des échantillons de $y[k]$ et $u[k]$ comme ci-dessous.

$$a_0 \cdot y[k] + a_1 \cdot y[k-1] + \cdots + a_N \cdot y[k-N] = b_0 \cdot u[k] + b_1 \cdot u[k-1] + \cdots + b_N \cdot u[k-N] \tag{2.67}$$

Où:
$y[k]$ est la valeur de sortie de l'instant $h \cdot k\,[s]$
$y[k-n]$, $n \in \mathbb{N}^*$ sont les valeurs de sortie précédentes, aux instants $h \cdot (k-n)\,[s]$
$u[k]$ est la valeur d'entrée de l'instant $h \cdot k\,[s]$
$u[k-n]$, $n \in \mathbb{N}^*$ sont les valeurs d'entrée précédentes, aux instants $h \cdot (k-n)\,[s]$

Puisque $y[k]$ est la valeur de sortie qui doit être calculée, l'équation permettant d'évaluer une fonction de transfert discrète prend la forme suivante:

$$y[k] = \frac{b_0}{a_0} \cdot u[k] + \frac{b_1}{a_0} \cdot u[k-1] + \cdots + \frac{b_N}{a_0} \cdot u[k-N] - \frac{a_1}{a_0} \cdot y[k-1] - \cdots - \frac{a_N}{a_0} \cdot y[k-N] \tag{2.68}$$

À partir de l'équation 2.68, il est possible de décrire un système logique synchronisé sur une horloge permettant de résoudre le calcul d'une fonction de transfert. Le schéma de ce système logique est donné à la figure 2.41.

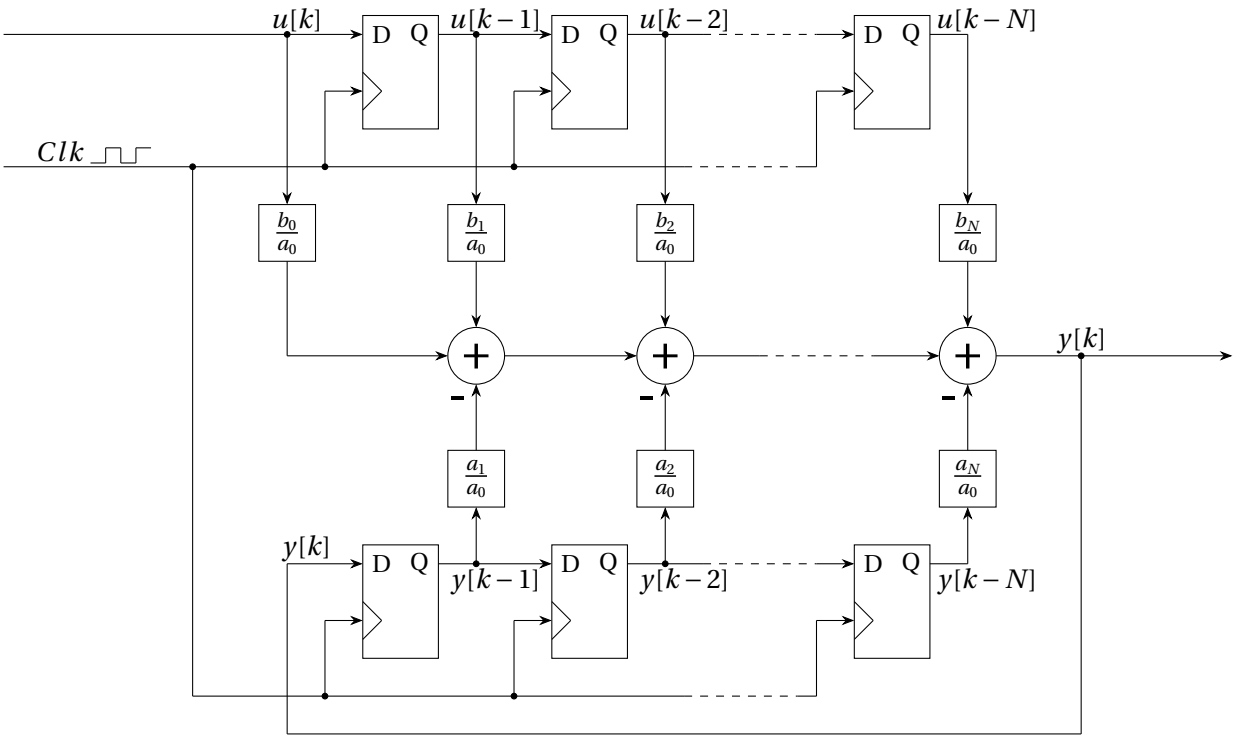


Figure 2.41: Système logique pour le calcul d'une fonction de tranfert discrète (filtre IIR)

Le schéma de la figure 2.41 correspond à un filtre IIR (« Infinite Impulse Response ») et il permet de simuler la fonction de transfert discrète de l'équation 2.64. Il est important de noter que dans ce cas là, si b_0 n'est pas nul, la sortie $y[k]$ n'est pas entièrement synchronisée sur l'horloge (aucune bascule entre $u[k]$ et $y[k]$). De plus, dans le cas d'un design synthétisable qui doit être implémenté dans une FPGA, la multiplication par les coefficients a un coût temporel d'au moins un coup d'horloge. Pour ces raisons, un design synthétisable synchronisé sur horloge doit avoir une bascule D supplémentaire à la sortie $y[k]$. La conséquence de cette bascule D supplémentaire est un signal de sortie qui est retardé d'un coup d'horloge.

2.3.2 Méthodes de discrétisation

Il existe plusieurs manières de discrétiser une fonction de transfert. Pour cause, une fonction de transfert analogique ne possède pas d'équivalent exact dans le domaine numérique dans le cas général. Des équivalents exacts existent lorsque certaines méthodes sont utilisées dans certains cas spécifiques. Dans cette section, trois méthodes de discrétisation différentes sont étudiées afin de déterminer laquelle convient le mieux à la situation. Il s'agit des méthodes de Tustin, ZOH et FOH.

L'évaluation d'une méthode de discrétisation dépend beaucoup de la nature du système à discrétiser et de la fréquence à laquelle ce système doit être échantillonné. La section présente se contente donc de présenter brièvement les méthodes sans exemples concrets. Une étude plus approfondie avec des simulations sera effectuée à la section 2.3.3 afin d'évaluer concrètement chaque méthode en fonction de la fréquence d'échantillonnage.

2.3.2.1 Méthode Tustin

La méthode de Tustin (aussi appelée transformation bilinéaire) est une méthode de discrétisation relativement simple qui offre une très bonne approximation lorsque la fréquence d'échantillonnage est faible. Cependant, lorsque cette méthode est utilisée pour discrétiser des systèmes à une fréquence d'échantillonnage élevée, le modèle discret peut montrer des signes d'instabilité.

Dans la théorie de la régulation numérique, la relation entre les pôles des systèmes analogiques et ceux des systèmes discrets est définie à l'équation 2.69.

$$z = e^{s \cdot Ts} \tag{2.69}$$

Où:
z est le pôle numérique
s est le pôle analogique
$Ts\,[s]$ est la période d'échantillonnage

Avec la relation ci-dessus, la correspondance entre le domaine en s et le domaine en z est exacte. Cependant, une fonction de transfert doit être exprimée sous forme de polynômes. Pour s'approcher de l'équation 2.69, la méthode de Tustin utilise l'approximation de Padé du premier ordre (équation 2.70).

$$e^{s \cdot Ts} = \frac{e^{s \cdot Ts/2}}{e^{-s \cdot Ts/2}} \approx \frac{1 + s \cdot Ts/2}{1 - s \cdot Ts/2} \tag{2.70}$$

Avec cette méthode, la discrétisation d'une fonction de transfert consiste à remplacer la variable s par:

$$s = \frac{2}{Ts} \cdot \frac{z-1}{z+1} \tag{2.71}$$

L'approximation de Padé n'étant qu'une approximation, la correspondance entre domaine continu et discret n'est pas exacte. Mais dans un grand nombre de cas, cette approximation donne de bons résultats.

2.3.2.2 Méthode ZOH

La méthode ZOH (« Zero-Order Hold ») consiste à modéliser une fonction de transfert discrète pour laquelle le signal d'entrée est stable durant une période d'échantillonnage.

$$u(t) = u[k] \qquad , k \cdot Ts \leqslant t \leqslant (k+1) \cdot Ts \tag{2.72}$$

Par exemple, un signal généré par un convertisseur DAC (« Digital-to-Analog Converter ») est stable durant une période d'échantillonnage. Cela signifie qu'un système à régler utilisant des convertisseurs (pour s'interfacer avec un régulateur numérique) a une correspondance exacte s'il est discrétisé avec la méthode ZOH. De cette manière, la fonction de transfert discrète prend en compte dans sa modélisation un retard lié au signal qui est bloqué pendant une période d'échantillonnage.

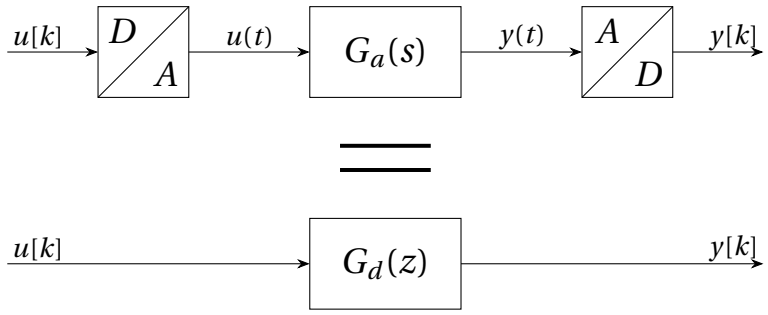


Figure 2.42: Équivalence entre système analogique et discret selon la méthode ZOH

L'équation permettant de passer directement d'une fonction de transfert analogique à une fonction de transfert discrète avec la méthode ZOH est donnée à l'équation 2.73.

$$G_d(z) = (1 - z^{-1}) \cdot \mathcal{Z}\left\{\mathcal{L}^{-1}\left\{\frac{G_a(s)}{s}\right\}\Big|_{t=k\cdot h}\right\} \tag{2.73}$$

Cette équation (2.73) peut être décomposée en plusieurs étapes afin de faciliter sa compréhension. Tout d'abord, à l'aide de la transformée inverse de Laplace, il est possible de trouver le signal temporel continu correspondant:

$$g_a(t) = \mathcal{L}^{-1}\left\{\frac{G_a(s)}{s}\right\} \tag{2.74}$$

Le signal continu obtenu grâce à la transformée inverse de Laplace peut être discrétisé avec l'équivalence suivante:

$$t = k \cdot h\,[s]$$

Où:
$t\,[s]$ est le temps continu
$k \in \mathbb{Z}$ est l'indice de l'échantillon d'un signal
$h\,[s]$ est la période d'échantillonnage

$$g_d[k] = g_a(t)\Big|_{t=k\cdot h} \tag{2.75}$$

À partir du signal discret, la fonction de transfert discrète peut être obtenue avec la transformée en Z multipliée par $(1-z^{-1})$.

$$G_d(z) = (1-z^{-1}) \cdot \mathcal{Z}\left\{g_d[k]\right\} \tag{2.76}$$

Ce processus de discrétisation peut être réalisé automatiquement à l'aide d'outils numériques (Matlab ou python) pour une période d'échantillonnage définie. Cependant, pour réaliser cette discrétisation avec une période d'échantillonnage variable (comme paramètre générique), il est nécessaire de réaliser les étapes une par une pour écrire les coefficients des polynômes sous forme symbolique.

Puisque cette méthode est très répandue dans le domaine de la régulation, les transformées de Laplace et les transformée en Z les plus courantes sont facilement trouvables dans des tables disponibles en ligne (exemple avec le lien ci-dessous) ou dans les cours de régulation numérique ([6]).

`https://www.pdffiller.com/preview/101/736/101736884/large.png`

Dans le cas où la fonction de transfert n'existe pas dans les tables et qu'elle doit être exprimée en fonction de la période d'échantillonnage, il est nécessaire de refaire les calculs de la transformée inverse de Laplace et de la transformée en Z. Il n'est pas réaliste de vouloir résoudre de tels calculs "à la main". Il existe une toolbox Matlab "Symbolic Math Toolbox" pemettant de réaliser des calculs symboliques (avec des paramètres variables). Même si cet outil permet de simplifier grandement les calculs, il est quand même nécessaire de réécrire ou de revoir la forme des résultats. L'utilisation du calcul symbolique avec Matlab a donc ses limites. La complexité des calculs dépend du nombre de pôles et de zéros de la fonction de transfert analogique et, si cela est possible, il est souhaitable de simplifier au maximum la fonction de transfert (pôles/zéros qui s'annulent ou qui sont proches de 0 ou de l'infini).

2.3.2.3 Méthode FOH

La méthode FOH (« First-Order Hold ») utilise la même approche que la méthode ZOH mais avec un signal d'entrée qui n'est pas bloqué. Entre deux échantillons, le signal est interpolé selon l'équation 2.77.

$$u(t) = u[k] + \frac{t-k\cdot Ts}{Ts} \cdot (u[k+1]-u[k]) \qquad , k\cdot Ts \leqslant t \leqslant (k+1)\cdot Ts \tag{2.77}$$

Cette méthode permet de modéliser avec une grande précision les systèmes analogiques dont le signal d'entrée n'est pas bloqué par des convertisseurs. Cette méthode étant encore plus compliquée que la méthode ZOH, l'algorithme permettant de discrétiser une fonction de transfert avec cette méthode n'est pas étudié dans ce travail.

2.3.3 Fréquence d'échantillonnage

La fréquence à laquelle le système à régler doit être échantillonné n'est pas un choix évident. Il y a des contraintes liées aux méthodes de discrétisation mais aussi des contraintes liées à la commande entre le régulateur et le système à régler.

Habituellement, un système à régler analogique asservi par un régulateur numérique est discrétisé par la méthode ZOH à la fréquence d'échantillonnage des convertisseurs. De cette manière, le modèle discret est parfaitement équivalent au modèle analogique. Cependant, cette manière de procéder n'est pas possible dans ce travail pour plusieurs raisons.

Pour procéder à la discrétisation d'un système analogique, il est nécessaire que celui-ci soit modélisé en une seule fonction de transfert (ou plusieurs en parallèle). Ici, le problème est que le modèle est trop complexe pour pouvoir le mettre sous forme d'une seule fonction de transfert. Il est modélisé sous forme de plusieurs fonctions de transfert et il serait faux de procéder à la discrétisation de celles-ci individuellement pour les mettre en série après discrétisation. En effet, puisque la méthode ZOH prend en compte le retard lié au bloqueur d'ordre zéro (figure 2.42), mettre en série plusieurs fonctions de transfert discrétisées avec la méthode ZOH revient à prendre en compte le retard de plusieurs convertisseurs en série.

Un autre problème est lié à la commande du régulateur. Il ne s'agit pas d'une commande d'un convertisseur d'ordre zéro mais d'un PWM. Le signal d'entrée du système à régler n'est donc pas stable durant une période d'échantillonnage. Comme expliqué à la section 2.2.1.1, une période de PWM est constituée de 2480 coups d'horloge de la FPGA (équation 2.1). Si la fréquence d'échantillonnage du système à régler est inférieure à la fréquence d'horloge de la FPGA (40 $[MHz]$), il y a une perte d'informations. Dans le pire des cas où cette fréquence d'échantillonnage du système à régler est égale à la fréquence du PWM (16129 $[Hz]$), aucune information n'est transmise. En effet, si la fréquence du système à régler est synchronisée sur la fréquence du PWM, la valeur d'entrée est toujours prise au début du cycle de PWM et celle-ci est donc toujours nulle. En tenant compte de ces informations, il est nécessaire que la fréquence d'échantillonnage du système à régler soit plus élevée que la fréquence d'échantillonnage du régulateur.

Il faut aussi garder à l'esprit qu'une fréquence d'échantillonnage élevée augmente la durée de la simulation. Le temps de simulation peut être un paramètre à prendre en compte si les durées de simulation deviennent trop importantes.

En sachant que la fréquence doit se situer entre 16129 $[Hz]$ et 40 $[MHz]$ avec plusieurs méthodes de discrétisation possibles, ce choix peut être fait en simulant le système avec plusieurs méthodes et plusieurs fréquences et en observant l'erreur liée à la discrétisation. En effet, les outils de simulation Matlab permettent de discrétiser une fonction de transfert analogique avec une simple fonction (pour une période d'échantillonnage donnée):

```
Gd_zoh = c2d(Ga, Ts, 'zoh')
Gd_tustin = c2d(Ga, Ts, 'tustin')
Gd_foh = c2d(Ga, Ts, 'foh')
```

Ainsi, il est possible de simuler le système analogique, puis le système discret et de quantifier l'erreur en comparant les résultats. Ceci peut être fait avec les différentes méthodes pour différentes fréquences.

Le choix a été fait de réaliser cette comparaison en boucle fermée et en boucle ouverte. C'est-à-dire, que les deux systèmes sont simulés une première fois en boucle fermée, où la commande du système analogique est sauvegardée. Ensuite, une deuxième simulation est réalisée en boucle ouverte avec la même commande pour les deux systèmes. La raison de cette comparaison en deux étapes est qu'en boucle fermée, un système peut être instable à cause d'un retard trop important dans le modèle discretisé. En boucle ouverte, les retards n'ont pas d'impact sur la sortie du système (mis à part le fait que les échantillons sont décalés). Cette seconde simulation en boucle ouverte permet donc d'observer le comportement du système avec une commande normale sans que les retards provoquent une instabilité.

`dev/01_plant_model/simulink_model/08_analogic_modified/analogic_modified_AY8.m`

Dans cette simulation, les deux modèles sont simulés (en boucle ouverte et en boucle fermée) puis l'erreur MSE est calculée et sauvegardée dans un fichier. Le choix a été fait de comparer la commande, la vitesse et la position en boucle fermée. En boucle ouverte, avec une commande identique pour les deux modèles, ce sont les courants (i_d et i_q), la vitesse et la position qui sont comparés.

Après avoir effectué plusieurs simulations avec plusieurs méthodes et fréquences différentes, les erreurs peuvent être affichées sur un graphique pour comparer les résultats. Dans le cas où la boucle est instable, les erreurs peuvent être énormes et pour cette raison, les erreurs sont comparées sur une échelle logarithmique.

Sur la figure 2.43 sont affichées les erreurs calculées lorsque le système à régler a été discrétisé avec la méthode ZOH. Puisque ces graphiques ne montrent que les erreurs, il est important de préciser à partir de quelle valeur d'erreur le système est jugé instable. D'après les courbes (qui ne sont pas montrées ici), la commande montre des signes d'instabilité en boucle fermée lorsque l'erreur MSE de la commande s'approche de la valeur 1. Par exemple avec la méthode ZOH en boucle fermée, le système est instable pour des fréquences en-dessous de 640 $[kHz]$ et au-dessus de 5120 $[kHz]$. Entre ces deux fréquences, le système en boucle fermée est stable.

Les résultats montrés à la figure 2.43 confirment que la méthode ZOH n'est pas adaptée lorsque ce système à régler est échantillonné dans les basses fréquences. Mais dans cette simulation Matlab, une fréquence d'échantillonnage élevée est également problèmatique. Les meilleurs résultats sont obtenus autour de 2560 kHz.

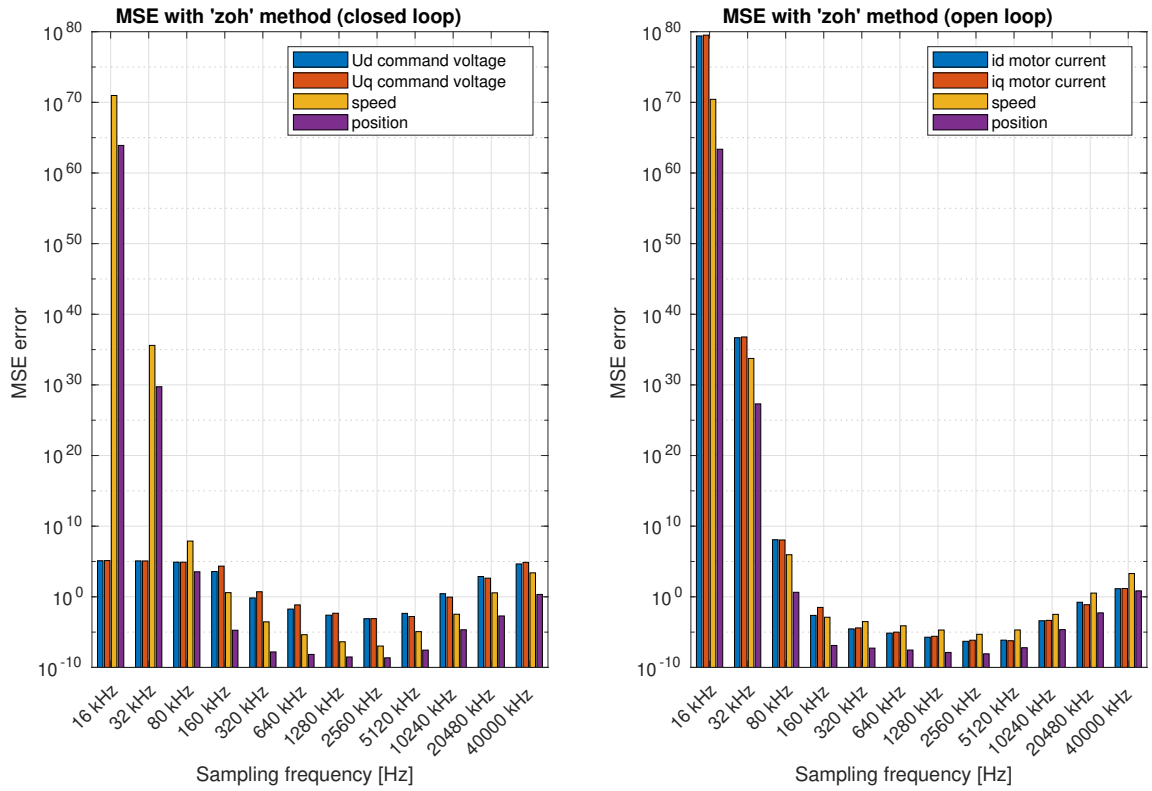


Figure 2.43: Comparaison des erreurs de discrétisation (ZOH) pour différentes fréquences d'échantillonnage

`dev/01_plant_model/simulink_model/08_analogic_modified/show_mse.m`

Avec la méthode de Tustin (figure 2.44), le système en boucle fermée est instable au-delà de 80 $[kHz]$. L'utilisation de la méthode de Tustin se limite donc aux fréquences d'échantillonnage basses.

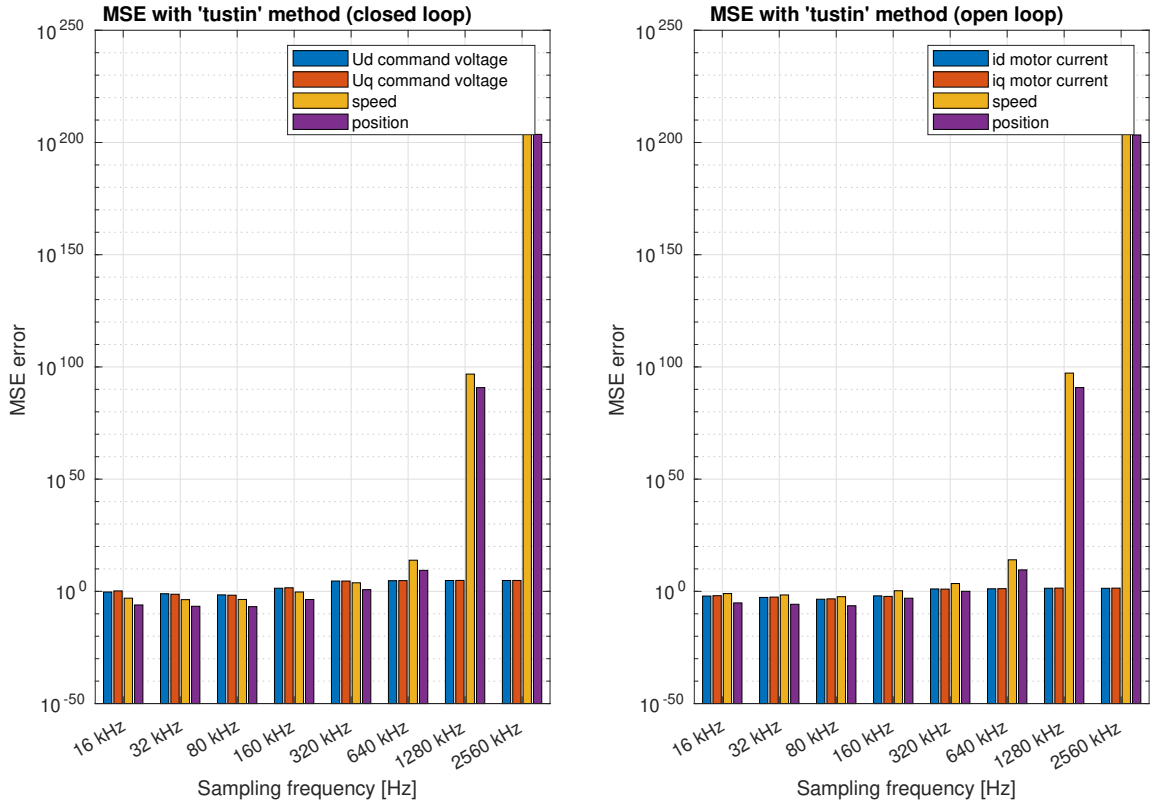


Figure 2.44: Comparaison des erreurs de discrétisation (Tustin) pour différentes fréquences d'échantillonnage

`dev/01_plant_model/simulink_model/08_analogic_modified/show_mse.m`

Sur la figure 2.45 sont affichés les résultats obtenus avec la méthode FOH. Les résultats sont très bons comparés à ceux des deux autres méthodes. En fait, la méthode FOH donne des bons résultats dans les basses fréquences (comme pour Tustin) et des bons résultats également dans les fréquences plus élevées (comme pour ZOH). La limite de la stabilité s'arrête, comme pour la méthode ZOH, à 5120 $[kHz]$.

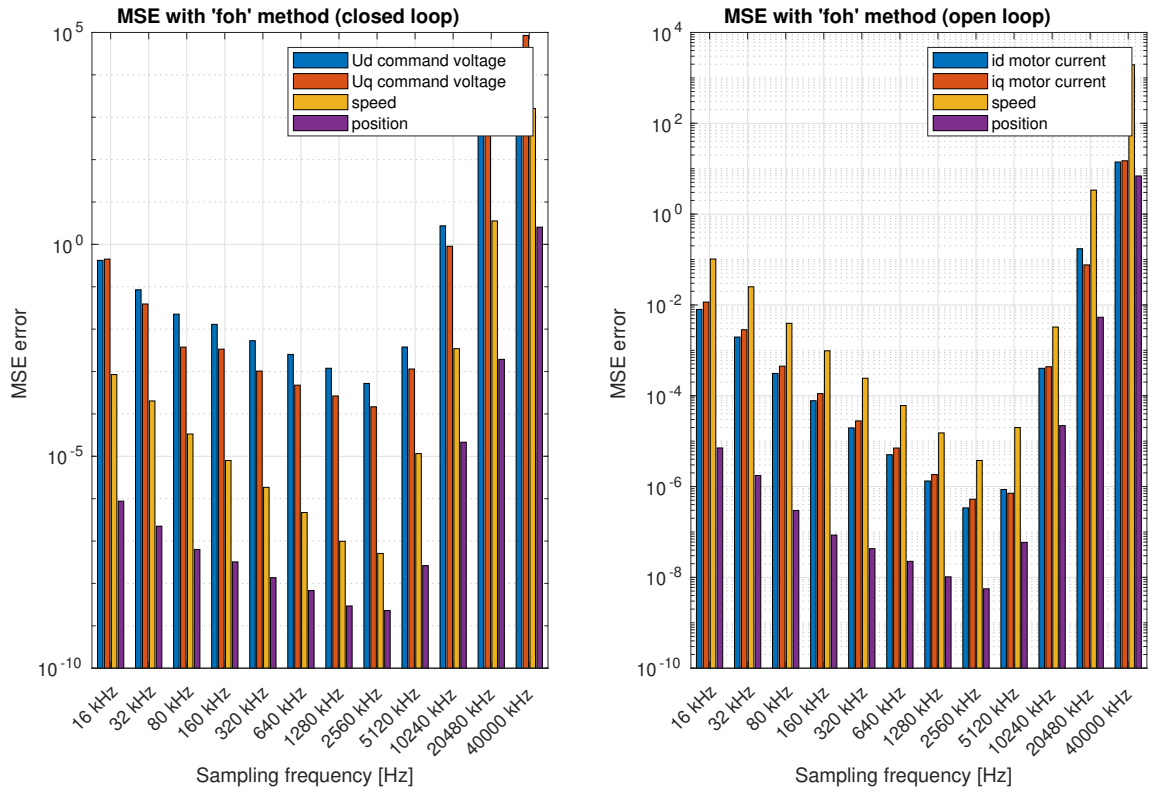


Figure 2.45: Comparaison des erreurs de discrétisation (FOH) pour différentes fréquences d'échantillonnage

`dev/01_plant_model/simulink_model/08_analogic_modified/show_mse.m`

Pour mieux visualiser les différences entre les méthodes, la figure 2.46 compare directement les erreurs de chaque méthode sur un seul graphique.

Précédemment, l'erreur était calculée pour huit grandeurs (commande, courant, position et vitesse en boucle fermée et ouverte). Sur la figure 2.46, dans le but de faciliter la comparaison, toutes les erreurs ont été quantifiées avec une seule valeur au lieu de huit. Puisque les erreurs des différentes grandeurs n'ont pas toutes les mêmes ordres de grandeurs, le choix arbitraire a été fait de sommer les valeurs d'un point de vue logarithmique. Ainsi, pour une méthode à une fréquence précise, l'erreur totale correspond au produit des huit erreurs calculées précédemment (la somme des valeurs logarithmiques étant le produit des valeurs linéaires).

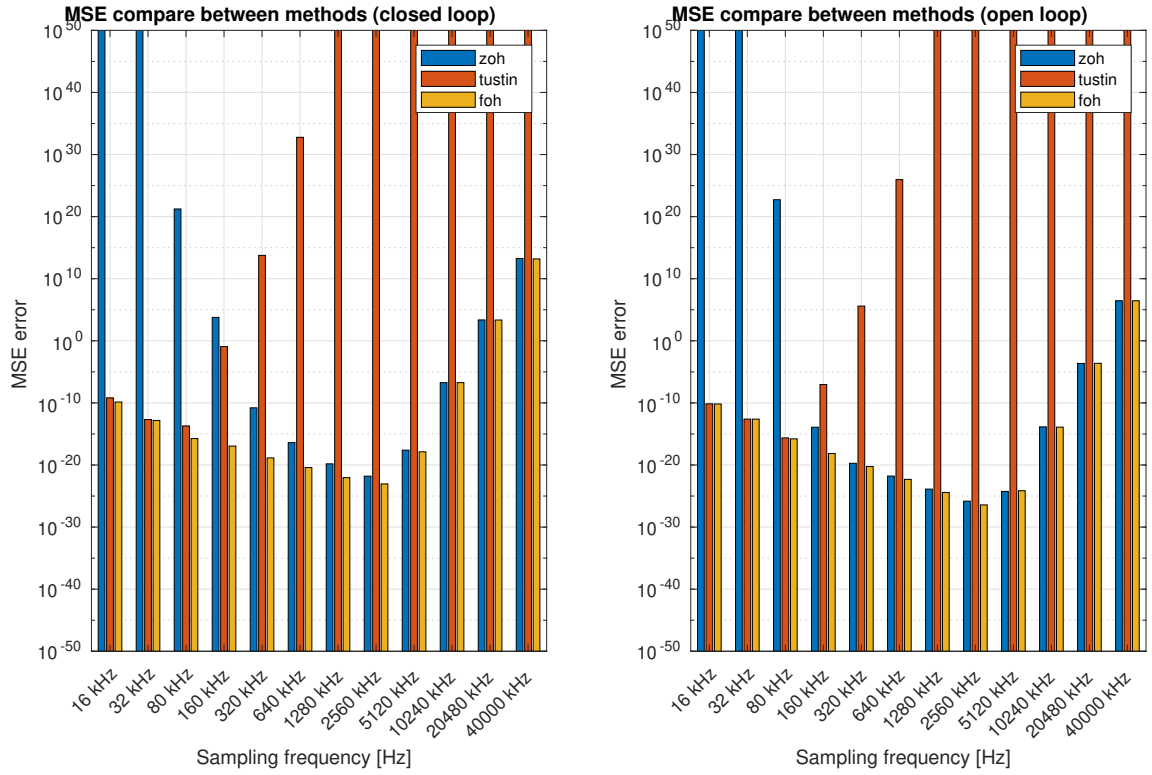


Figure 2.46: Comparaison des erreurs de discrétisation entre les différentes méthodes pour différentes fréquences d'échantillonnage
`dev/01_plant_model/simulink_model/08_analogic_modified/show_mse.m`

La figure 2.46 résume bien les résultats obtenus sur les graphiques des figures 2.43, 2.44 et 2.45. En basses fréquences, les méthodes de Tustin et FOH permettent d'obtenir de bons résultats. Vers la fréquence de 320 $[kHz]$, seule la méthode FOH est acceptable et les meilleurs résultats obtenus se situent entre 640 $[kHz]$ et 5120 $[kHz]$ avec les méthodes ZOH et FOH.

Plus haut dans cette section, il a été expliqué que la commande sous forme de PWM peut provoquer des pertes d'informations (diminution de la résolution de la commande) si la fréquence d'échantillonnage du système à régler est trop faible. Dans la simulation Matlab utilisée pour générer les graphiques ci-dessus, la commande est parfaite (sans PWM) et ce problème n'est donc pas pris en compte. Cela signifie qu'avec le régulateur réel et sa commande en PWM, les résultats en basses fréquences risquent de se dégrader par rapport aux résultats montrés ici. En prenant en compte toutes ces informations, il est préférable de choisir une fréquence d'échantillonnage plutôt élevée autour de 2560 $[kHz]$.

L'autre facteur qui peut jouer un rôle important est la durée de simulation du testbench. Il peut être souhaitable de pouvoir ajuster la fréquence d'échantillonnage du système à régler en fonction des résultats obtenus et de la durée de simulation lorsque le testbench sera simulé avec les outils VHDL/SystemVerilog. Pour cette raison, il a été décidé de modéliser les fonctions de transfert discrètes du système à régler de manière à ce que la fréquence d'échantillonnage soit un paramètre générique.

Pour discrétiser les fonctions de transfert en laissant la fréquence d'échantillonnage comme paramètre générique, le choix a été fait d'utiliser la méthode ZOH. Le principal défaut de cette méthode est qu'elle n'est pas adaptée à un système où plusieurs fonctions de transfert en série doivent être discrétisées individuellement (à cause des retards introduits dans le modèle discret). Mais avec une fréquence d'échantillonnage plus élevée, le retard ajouté qui est proportionnel à la fréquence d'échantillonnage devient très faible et donc négligeable.

Selon la figure 2.46, la méthode FOH donne des résultats encore meilleurs mais les calculs de cette méthode sont trop complexes pour pouvoir effectuer la discrétisation sans l'aide d'un outil de calcul numérique puissant tel que Matlab. De plus, la différence entre ZOH et FOH sur la figure 2.46 reste très faible dans les fréquences proches de 2560 $[kHz]$.

2.3.4 Discrétisation du système à régler

Dans cette section, chacune des fonctions de transfert du système à régler est discrétisée avec la méthode ZOH. En effectuant les calculs avec la transformée inverse de Laplace et la transformée en Z, il est possible de paramétriser ces fonctions de transfert discrètes en fonction de la période d'échantillonnage. La simplification des fonctions de transfert analogiques joue un rôle important pour limiter au maximum la complexité des calculs.

En exprimant les fonctions de transfert discrètes sous forme de polynômes en fonction des paramètres, ces modèles discrets seront facilement implémentables dans les simulations sans avoir recours à des outils de discrétisation (non disponibles en VHDL ou SystemVerilog). Il sera ainsi très simple de modifier des paramètres du modèle physique ou de changer la fréquence d'échantillonnage pour en observer les effets.

2.3.4.1 Discrétisation de Gdelay du bloc « Power »

La fonction de transfert correspondant au délai de l'onduleur a été modélisée en utilisant l'approximation de Padé avec les paramètres k_1 et k_2.

$$e^{s\cdot\tau_d} \approx \frac{k_2\cdot s^2 - k_1\cdot s + 1}{k_2\cdot s^2 + k_1\cdot s + 1} = G_{delay}(s)$$

Avec:

$$\tau_d = 31\cdot 10^{-6} \qquad k_1 = \frac{\tau_d}{2} \qquad k_2 = \frac{\tau_d^2}{12}$$

Cette fonction de transfert peut s'écrire en fonction de τ_d:

$$G_{delay}(s) = \frac{\frac{\tau_d^2}{12}\cdot s^2 - \frac{\tau_d}{2}\cdot s + 1}{\frac{\tau_d^2}{12}\cdot s^2 + \frac{\tau_d}{2}\cdot s + 1} = \frac{\tau_d^2\cdot s^2 - 6\cdot\tau_d\cdot s + 12}{\tau_d^2\cdot s^2 + 6\cdot\tau_d\cdot s + 12} \tag{2.78}$$

Pour discrétiser une fonction de transfert analogique avec la méthode ZOH, il est préférable d'exprimer cette fonction de transfert sous forme de pôles et de zéros (équation 2.79).

$$G_{delay}(s) = \frac{(s-z_1)\cdot(s-z_2)}{(s-p_1)\cdot(s-p_2)} \tag{2.79}$$

Il s'agit de deux zéros complexes conjugués et de deux pôles complexes conjugués.

$$z_{1,2} = \frac{6 \cdot \tau_d \pm \sqrt{36 \cdot \tau_d^2 - 48 \cdot \tau_d^2}}{2 \cdot \tau_d^2} = \frac{3}{\tau_d} \pm \frac{\sqrt{3}}{\tau_d} \cdot j$$

$$p_{1,2} = \frac{-6 \cdot \tau_d \pm \sqrt{36 \cdot \tau_d^2 - 48 \cdot \tau_d^2}}{2 \cdot \tau_d^2} = -\frac{3}{\tau_d} \pm \frac{\sqrt{3}}{\tau_d} \cdot j$$

Pour simplifier au maximum les calculs, il faut réduire le nombre de paramètres différents au minimum. Dans le cas présent, les zéros et les pôles étant quasi similaires, il est possible d'écrire cette fonction de transfert sous la forme ci-dessous:

$$G_{delay}(s) = \frac{(s - a - b \cdot j) \cdot (s - a + b \cdot j)}{(s + a - b \cdot j) \cdot (s + a + b \cdot j)} \tag{2.80}$$

Avec:

$$a = \frac{3}{\tau_d} \qquad\qquad b = \frac{\sqrt{3}}{\tau_d}$$

Cette fonction de transfert sous cette forme peut être discrétisée avec l'exemple à l'annexe B.1. La fonction de transfert discrète est donnée ci-dessous sous forme de polynômes avec des coefficients.

$$G_d(z) = \frac{b_0 \cdot z^2 + b_1 \cdot z + b_2}{a_0 \cdot z^2 + a_1 \cdot z + a_2} \tag{2.81}$$

$b_0 = 1$
$b_1 = -2 \cdot e^{-h \cdot a} \cdot \left(cos(h \cdot b) + 2 \cdot \frac{a}{b} \cdot sin(h \cdot b)\right)$
$b_2 = e^{-2 \cdot h \cdot a} + 4 \cdot \frac{a}{b} \cdot e^{-h \cdot a} \cdot sin(h \cdot b)$
$a_0 = 1$
$a_1 = -e^{-h \cdot a} \cdot 2 \cdot cos(h \cdot b)$
$a_2 = e^{-2 \cdot h \cdot a}$

Avec les valeurs numériques, la fonction de transfert discrète (à 16129 $[Hz]$) est donnée à l'équation 2.82.

$$G_d(z) = \frac{z^2 + 10.1 \cdot 10^{-3} \cdot z - 5.44 \cdot 10^{-3}}{z^2 + 4.7 \cdot 10^{-3} \cdot z + 6.14 \cdot 10^{-6}} \tag{2.82}$$

Les diagrammes de Bode des fonctions de transfert continue et discrète sont comparés à la figure 2.47.

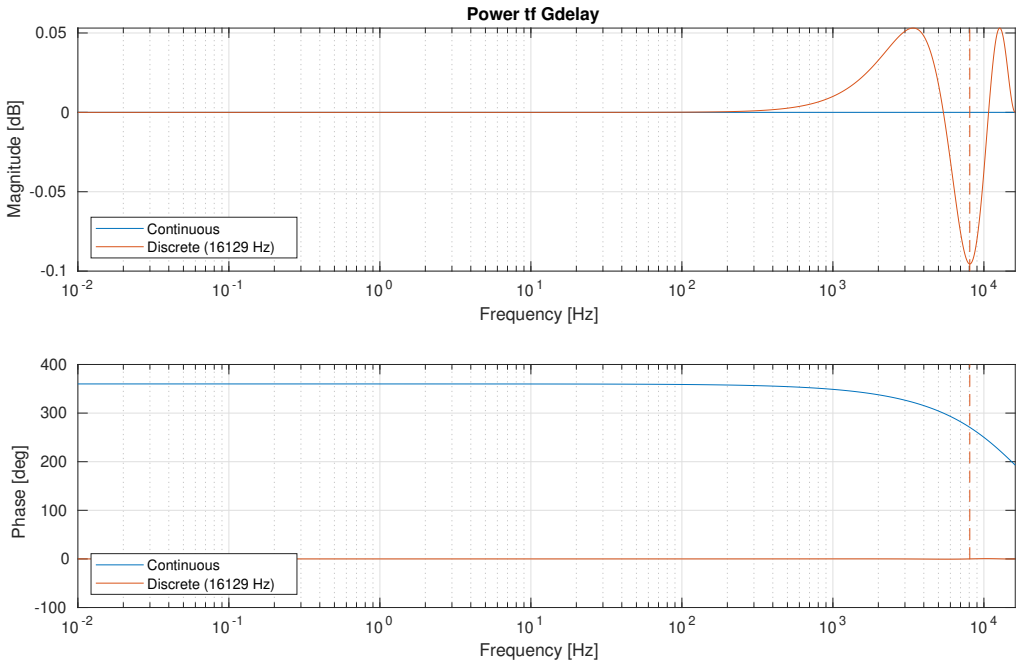


Figure 2.47: Comparaison avec la fonction de transfert discrète de Gdelay
`dev/90_test/10_plant_model/power_model/power_model_num.m`

2.3.4.2 Discrétisation de Uout/Iout du bloc « Filter »

La fonction de transfert qui donne une partie de la tension U_{out} en fonction de I_{out} a été modélisée avec les paramètres ci-dessous:

$$\frac{U_{out}(s)}{I_{out}(s)} = -\frac{s^2 \cdot R_C \cdot L \cdot C + s \cdot (L + R_C \cdot R_L \cdot C) + R_L}{s^2 \cdot L \cdot C + s \cdot C \cdot (R_L + R_C) + 1}$$

Pour la discrétiser, il faut exprimer cette fonction de transfert sous forme de pôles/zéros (équation 2.83).

$$\frac{U_{out}(s)}{I_{out}(s)} = \frac{-R_C \cdot L \cdot C \cdot (s - z_1) \cdot (s - z_2)}{L \cdot C \cdot (s - p_1) \cdot (s - p_2)} = -R_C \cdot \frac{(s - z_1) \cdot (s - z_2)}{(s - p_1) \cdot (s - p_2)} \tag{2.83}$$

$$z_1 = -\frac{R_L}{L} \qquad z_2 = -\frac{1}{R_C \cdot C}$$

$$p_{1,2} = -\frac{C \cdot (R_L + R_C)}{2 \cdot L \cdot C} \pm \frac{\sqrt{4 \cdot L \cdot C - (C \cdot (R_L + R_C))^2}}{2 \cdot L \cdot C} = a \pm b \cdot j$$

Pour connaître l'ordre de grandeur de ces pôles/zéros, il faut remplacer les paramètres par leurs valeurs numériques données à la table 2.1.

$$z_1 = -9.62$$

$$z_2 = -147 \cdot 10^9$$

$$p_{1,2} = -4.81 \cdot 10^{-3} \pm 6.3 \cdot 10^3 \cdot j$$

Avec les valeurs numériques, z_2 est très grand. Un pôle ou un zéro qui est trop grand ou trop petit peut ne pas avoir d'effet dans la bande passante du système. Pour vérifier l'impacte de z_2, le diagramme de Bode de la figure 2.48 compare la fonction de transfert avec et sans le zéro z_2.

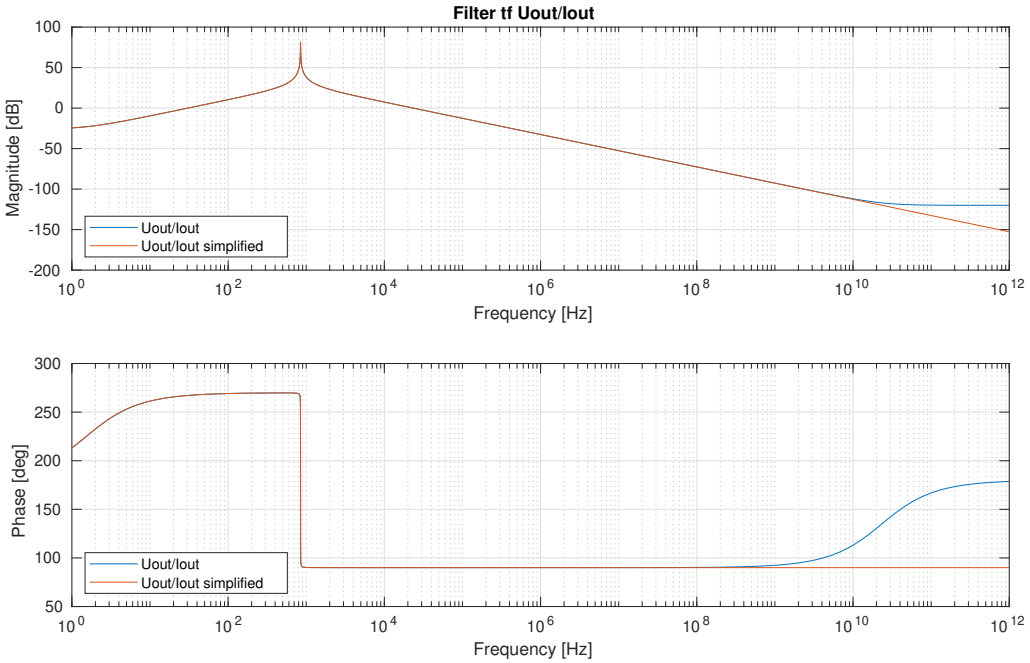


Figure 2.48: Comparaison avec la fonction de transfert U_{out}/I_{out} simplifiée
`dev/90_test/10_plant_model/filter_model/filter_model_num.m`

La figure 2.48 montre que le zéro z_2 a un effet uniquement au-delà de $1\,[GHz]$. Il est donc inutile de le prendre en compte dans cette modélisation. La fonction de transfert simplifiée est donnée à l'équation 2.84.

$$\frac{U_{out}(s)}{I_{out}(s)} \approx -\frac{1}{C} \cdot \frac{(s-z_1)}{(s-a-b\cdot j)\cdot(s-a+b\cdot j)} \tag{2.84}$$

Cette fonction de transfert sous cette forme peut être discrétisée avec l'exemple de l'annexe B.2. La fonction de transfert discrète est donnée ci-dessous sous forme de polynômes avec des coefficients.

$$\frac{U_{out}(z)}{I_{out}(z)} = -\frac{1}{C} \cdot \frac{b_0\cdot z^2 + b_1\cdot z + b_2}{a_0\cdot z^2 + a_1\cdot z + a_2}$$

$b_0 = 0$
$b_1 = \frac{-b\cdot z1 + e^{h\cdot a}\cdot(b\cdot z1\cdot cos(h\cdot b) + (a^2+b^2-a\cdot z1)\cdot sin(h\cdot b))}{b\cdot(a^2+b^2)}$
$b_2 = \frac{-b\cdot z1\cdot e^{2\cdot h\cdot a} - e^{h\cdot a}\cdot((a^2+b^2-a\cdot z1)\cdot sin(h\cdot b) - b\cdot z1\cdot cos(h\cdot b))}{b\cdot(a^2+b^2)}$
$a_0 = 1$
$a_1 = -e^{h\cdot a}\cdot 2\cdot cos(h\cdot b)$
$a_2 = e^{2\cdot h\cdot a}$

Avec les valeurs numériques, cette fonction de transfert discrète (à 16129 $[Hz]$) est donnée à l'équation 2.85.

$$\frac{U_{out}(z)}{I_{out}(z)} = \frac{-8.95 \cdot z + 8.95}{z^2 - 1.89 \cdot z + 1} \tag{2.85}$$

Les diagrammes de Bode des fonctions de transfert continue et discrète sont comparés à la figure 2.49.

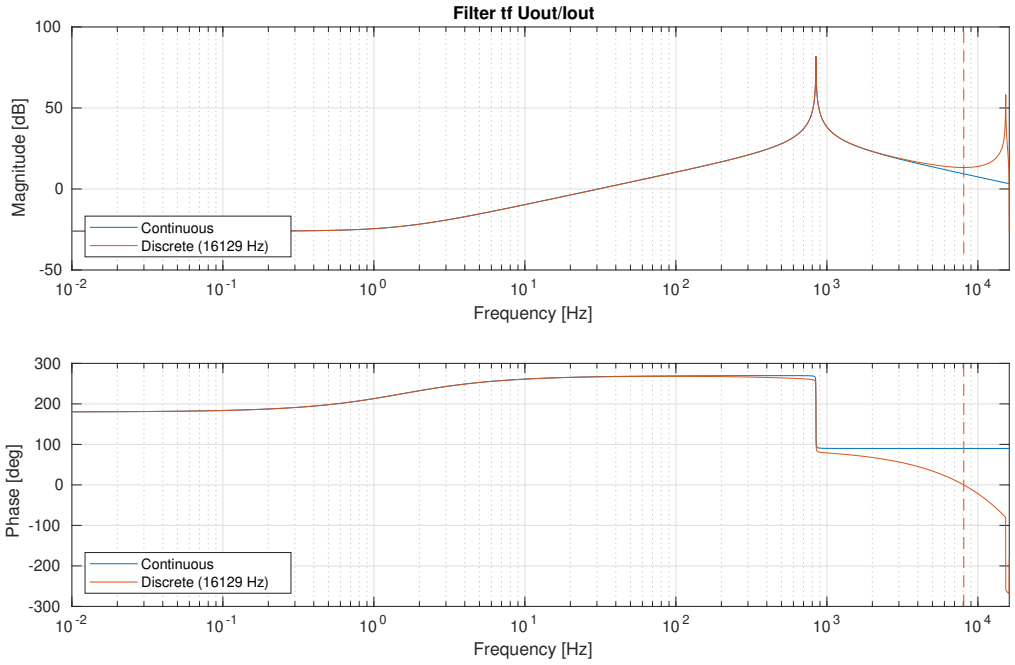


Figure 2.49: Comparaison avec la fonction de transfert discrète de Uout/Iout
`dev/90_test/10_plant_model/filter_model/filter_model_num.m`

2.3.4.3 Discrétisation de Uout/Uin du bloc « Filter »

La fonction de transfert qui donne une partie de la tension U_{out} en fonction de U_{in} a été modélisée avec les paramètres ci-dessous:

$$\frac{U_{out}(s)}{U_{in}(s)} = \frac{s \cdot R_C \cdot C + 1}{s^2 \cdot L \cdot C + s \cdot C \cdot (R_L + R_C) + 1}$$

À l'équation 2.86, cette fonction de transfert est donnée sous forme de pôles/zéros.

$$\frac{U_{out}(s)}{U_{in}(s)} = \frac{R_C \cdot C \cdot (s - z_1)}{L \cdot C \cdot (s - p_1) \cdot (s - p_2)} = \frac{R_C}{L} \cdot \frac{(s - z_1)}{(s - p_1) \cdot (s - p_2)} \tag{2.86}$$

$$z_1 = -\frac{1}{R_C \cdot C}$$

$$p_{1,2} = -\frac{C \cdot (R_L + R_C)}{2 \cdot L \cdot C} \pm \frac{\sqrt{4 \cdot L \cdot C - (C \cdot (R_L + R_C))^2}}{2 \cdot L \cdot C} = a \pm b \cdot j$$

Le zéro ci-dessus est le même que z_2 de l'équation 2.83. La figure 2.48 montre que ce zéro est négligeable. La fonction de transfert simplifiée est donnée à l'équation 2.87.

$$\frac{U_{out}(s)}{U_{in}(s)} \approx \frac{1}{C \cdot L} \cdot \frac{1}{(s-a-b \cdot j) \cdot (s-a+b \cdot j)} \tag{2.87}$$

La fonction de transfert de l'équation 2.87 peut être discrétisée avec l'exemple de l'annexe B.3. La fonction de transfert discrète est donnée à l'équation 2.88 sous forme de polynômes avec des coefficients.

$$\frac{U_{out}(z)}{U_{in}(z)} = \frac{1}{C \cdot L} \cdot \frac{b_0 \cdot z^2 + b_1 \cdot z + b_2}{a_0 \cdot z^2 + a_1 \cdot z + a_2} \tag{2.88}$$

$b_0 = 0$
$b_1 = \frac{b+a \cdot e^{h \cdot a} \cdot sin(h \cdot b) - b \cdot e^{h \cdot a} \cdot cos(h \cdot b)}{b \cdot (a^2+b^2)}$
$b_2 = \frac{b \cdot e^{2 \cdot h \cdot a} - b \cdot e^{h \cdot a} \cdot cos(h \cdot b) - a \cdot e^{h \cdot a} \cdot sin(h \cdot b)}{b \cdot (a^2+b^2)}$
$a_0 = 1$
$a_1 = -e^{h \cdot a} \cdot 2 \cdot cos(h \cdot b)$
$a_2 = e^{2 \cdot h \cdot a}$

Avec les valeurs numériques, cette fonction de transfert discrète (à 16129 $[Hz]$) est donnée à l'équation 2.89.

$$\frac{U_{out}(z)}{U_{in}(z)} = \frac{53.9 \cdot 10^{-3} \cdot z + 53.8 \cdot 10^{-3}}{z^2 - 1.89 \cdot z + 1} \tag{2.89}$$

Les diagrammes de Bode des fonctions de transfert continue et discrète sont comparés à la figure 2.50.

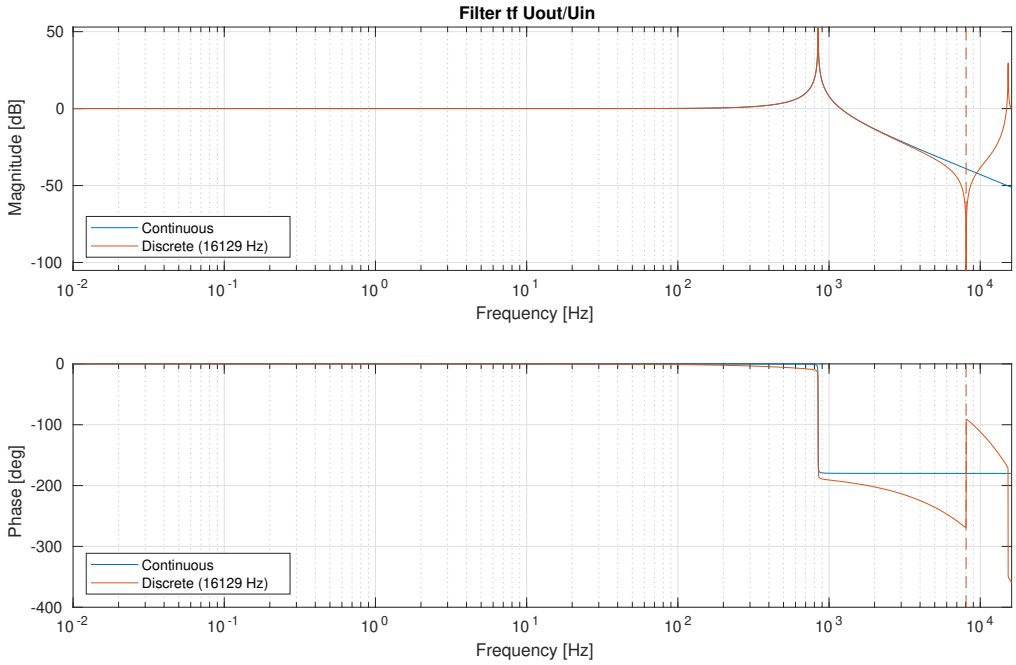


Figure 2.50: Comparaison avec la fonction de transfert discrète de Uout/Uin
`dev/90_test/10_plant_model/filter_model/filter_model_num.m`

2.3.4.4 Discrétisation de Uout/IL23 du bloc « Filter »

La fonction de transfert qui donne une partie de la tension U_{out} en fonction de I_{L2} et I_{L3} a été modélisée avec les paramètres ci-dessous:

$$\frac{U_{out}(s)}{I_{L23}(s)} = \frac{M \cdot \left(s^2 \cdot R_C \cdot C + s\right)}{s^2 \cdot L \cdot C + s \cdot C \cdot (R_L + R_C) + 1}$$

À l'équation 2.90, cette fonction de transfert est donnée sous forme de pôles/zéros.

$$\frac{U_{out}(s)}{I_{L23}(s)} = \frac{M \cdot R_C \cdot C \cdot s \cdot (s - z_1)}{L \cdot C \cdot (s - p_1) \cdot (s - p_2)} = \frac{M \cdot R_C}{L} \cdot \frac{s \cdot (s - z_1)}{(s - p_1) \cdot (s - p_2)} \tag{2.90}$$

$$z_1 = -\frac{1}{R_C \cdot C}$$

$$p_{1,2} = -\frac{C \cdot (R_L + R_C)}{2 \cdot L \cdot C} \pm \frac{\sqrt{4 \cdot L \cdot C - (C \cdot (R_L + R_C))^2}}{2 \cdot L \cdot C} = a \pm b \cdot j$$

Le zéro ci-dessus est le même que z_2 de l'équation 2.83. La figure 2.48 montre que ce zéro est négligeable. La fonction de transfert simplifiée est donnée à l'équation 2.91.

$$\frac{U_{out}(s)}{I_{L23}(s)} \approx \frac{M}{C \cdot L} \cdot \frac{s}{(s - a - b \cdot j) \cdot (s - a + b \cdot j)} \tag{2.91}$$

La fonction de transfert de l'équation 2.91 peut être discrétisée avec l'exemple de l'annexe B.4. La fonction de transfert discrète est donnée à l'équation 2.92 sous forme de polynômes avec des coefficients.

$$\frac{U_{out}(z)}{I_{L23}(z)} = \frac{M}{C \cdot L} \cdot \frac{b_0 \cdot z^2 + b_1 \cdot z + b_2}{a_0 \cdot z^2 + a_1 \cdot z + a_2} \tag{2.92}$$

$b_0 = 0$
$b_1 = \frac{e^{h \cdot a} \cdot sin(h \cdot b)}{b}$
$b_2 = \frac{-e^{h \cdot a} \cdot sin(h \cdot b)}{b}$
$a_0 = 1$
$a_1 = -e^{h \cdot a} \cdot 2 \cdot cos(h \cdot b)$
$a_2 = e^{2 \cdot h \cdot a}$

Avec les valeurs numériques, cette fonction de transfert discrète (à 16129 $[Hz]$) est donnée à l'équation 2.93.

$$\frac{U_{out}(z)}{I_{L23}(z)} = \frac{1.38 \cdot z + -1.38}{z^2 - 1.89 \cdot z + 1} \tag{2.93}$$

Les diagrammes de Bode des fonctions de transfert continue et discrète sont comparés à la figure 2.51.

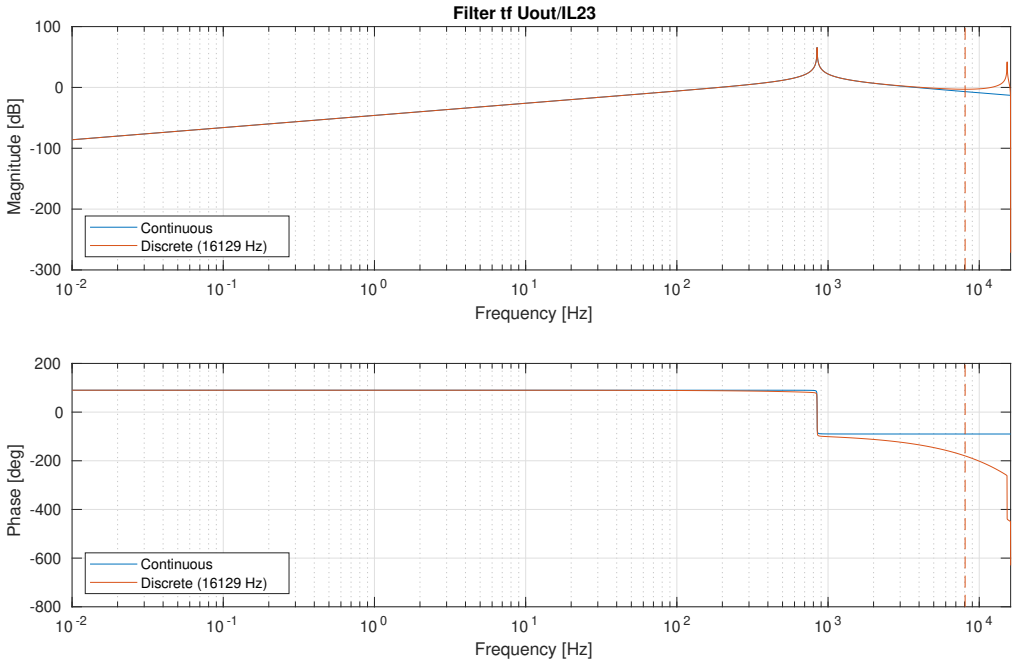


Figure 2.51: Comparaison avec la fonction de transfert discrète de Uout/IL23
`dev/90_test/10_plant_model/filter_model/filter_model_num.m`

2.3.4.5 Discrétisation de IL/Udiff du bloc « Filter »

La fonction de transfert qui donne une partie du courant I_L en fonction de U_{diff} (différence de tension entre l'entrée et la sortie du filtre) a été modélisée avec les paramètres ci-dessous:

$$\frac{I_L(s)}{U_{diff}(s)} = \frac{1}{R_L + s \cdot L}$$

À l'équation 2.94, cette fonction de transfert est donnée sous forme de pôles/zéros.

$$\frac{I_L(s)}{U_{diff}(s)} = \frac{1}{L} \cdot \frac{1}{(s - p_1)} \tag{2.94}$$

$$p_1 = -\frac{R_L}{L} = -9.62$$

La fonction de transfert de l'équation 2.94 peut être discrétisée avec l'exemple de l'annexe B.5. La fonction de transfert discrète est donnée à l'équation 2.95 sous forme de polynômes avec des coefficients.

$$\frac{I_L(z)}{U_{diff}(z)} = \frac{1}{L} \cdot \frac{b_0 \cdot z + b_1}{a_0 \cdot z + a_1} \tag{2.95}$$

$b_0 = 0$
$b_1 = \frac{e^{h \cdot p_1} - 1}{p_1}$
$a_0 = 1$
$a_1 = -e^{h \cdot p_1}$

Avec les valeurs numériques, cette fonction de transfert discrète (à 16129 $[Hz]$) est donnée à l'équation 2.96.

$$\frac{I_L(z)}{U_{diff}(z)} = \frac{11.9 \cdot 10^{-3}}{z - 1} \tag{2.96}$$

Les diagrammes de Bode des fonctions de transfert continue et discrète sont comparés à la figure 2.52.

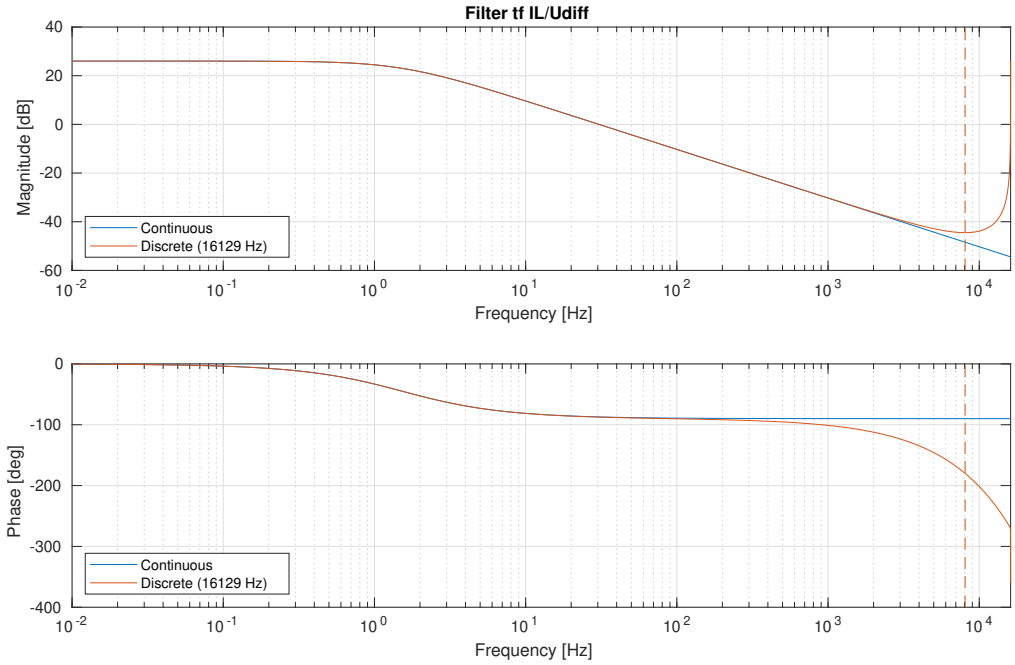


Figure 2.52: Comparaison avec la fonction de transfert discrète de IL/Udiff
`dev/90_test/10_plant_model/filter_model/filter_model_num.m`

2.3.4.6 Discrétisation de IL/IL23 du bloc « Filter »

La fonction de transfert qui donne une partie du courant I_L en fonction de I_{L2} et I_{L3} a été modélisée avec les paramètres ci-dessous:

$$\frac{I_L(s)}{I_{L23}(s)} = \frac{s \cdot M}{R_L + s \cdot L}$$

À l'équation 2.97, cette fonction de transfert est donnée sous forme de pôles/zéros.

$$\frac{I_L(s)}{I_{L23}(s)} = \frac{M}{L} \cdot \frac{s}{(s - p_1)} \tag{2.97}$$

$$p_1 = -\frac{R_L}{L} = -9.62$$

La fonction de transfert de l'équation 2.97 peut être discrétisée avec l'exemple de l'annexe B.6. La fonction de transfert discrète est donnée à l'équation 2.98 sous forme de polynômes avec des coefficients.

$$\frac{I_L(z)}{I_{L23}(z)} = \frac{M}{L} \cdot \frac{b_0 \cdot z + b_1}{a_0 \cdot z + a_1} \tag{2.98}$$

$b_0 = 1$
$b_1 = -1$
$a_0 = 1$
$a_1 = -e^{h \cdot p_1}$

Avec les valeurs numériques, cette fonction de transfert discrète (à 16129 $[Hz]$) est donnée à l'équation 2.99.

$$\frac{I_L(z)}{I_{L23}(z)} = \frac{0.154 \cdot z - 0.154}{z - 1} \tag{2.99}$$

Les diagrammes de Bode des fonctions de transfert continue et discrète sont comparés à la figure 2.53.

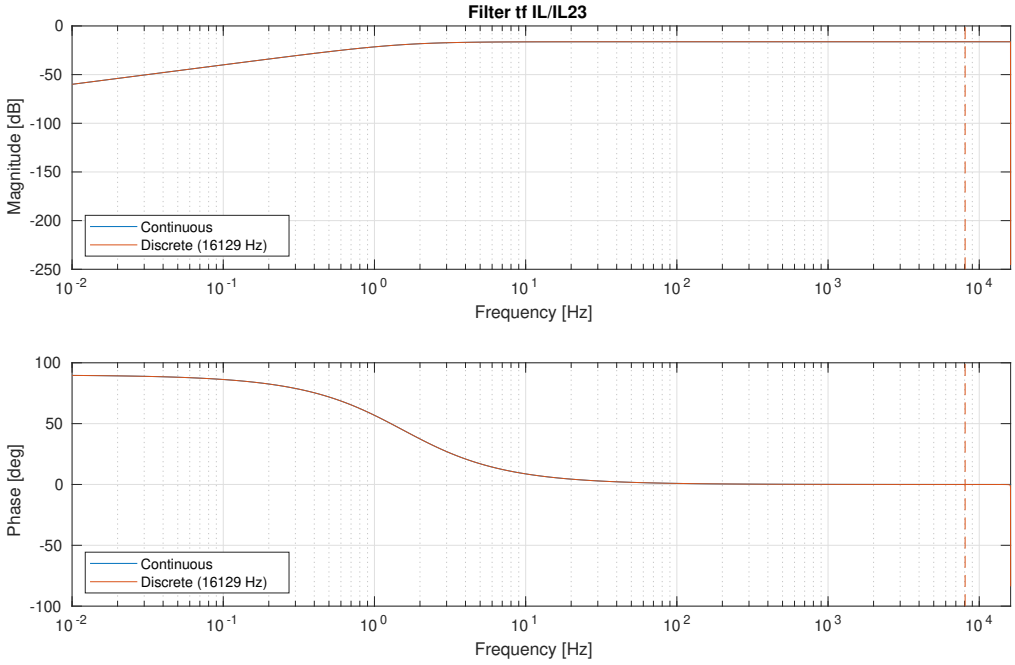


Figure 2.53: Comparaison avec la fonction de transfert discrète de IL/IL23
`dev/90_test/10_plant_model/filter_model/filter_model_num.m`

2.3.4.7 Discrétisation de Gmot du bloc « Motor »

La fonction de transfert correspondant à la partie électrique du moteur a été modélisée avec les paramètres ci-dessous:

$$G_{mot}(s) = \frac{1}{R + s \cdot L}$$

Les valeurs numériques de ces paramètres sont données à la table 2.2. À l'équation 2.100, cette fonction de transfert est donnée sous forme de pôles/zéros.

$$G_{mot}(s) = \frac{1}{L} \cdot \frac{1}{(s - p_1)} \tag{2.100}$$

$$p_1 = -\frac{R}{L} = 221$$

La fonction de transfert de l'équation 2.100 peut être discrétisée avec l'exemple de l'annexe B.5. La fonction de transfert discrète est donnée à l'équation 2.101 sous forme de polynômes avec des coefficients.

$$G_{mot}(s) = \frac{1}{L} \cdot \frac{b_0 \cdot z + b_1}{a_0 \cdot z + a_1} \tag{2.101}$$

$b_0 = 0$
$b_1 = \frac{e^{h \cdot p_1} - 1}{p_1}$
$a_0 = 1$
$a_1 = -e^{h \cdot p_1}$

Avec les valeurs numériques, cette fonction de transfert discrète (à 16129 $[Hz]$) est donnée à l'équation 2.102.

$$G_{mot}(s) = \frac{16.2 \cdot 10^{-3}}{z - 0.986} \tag{2.102}$$

Les diagrammes de Bode des fonctions de transfert continue et discrète sont comparés à la figure 2.54.

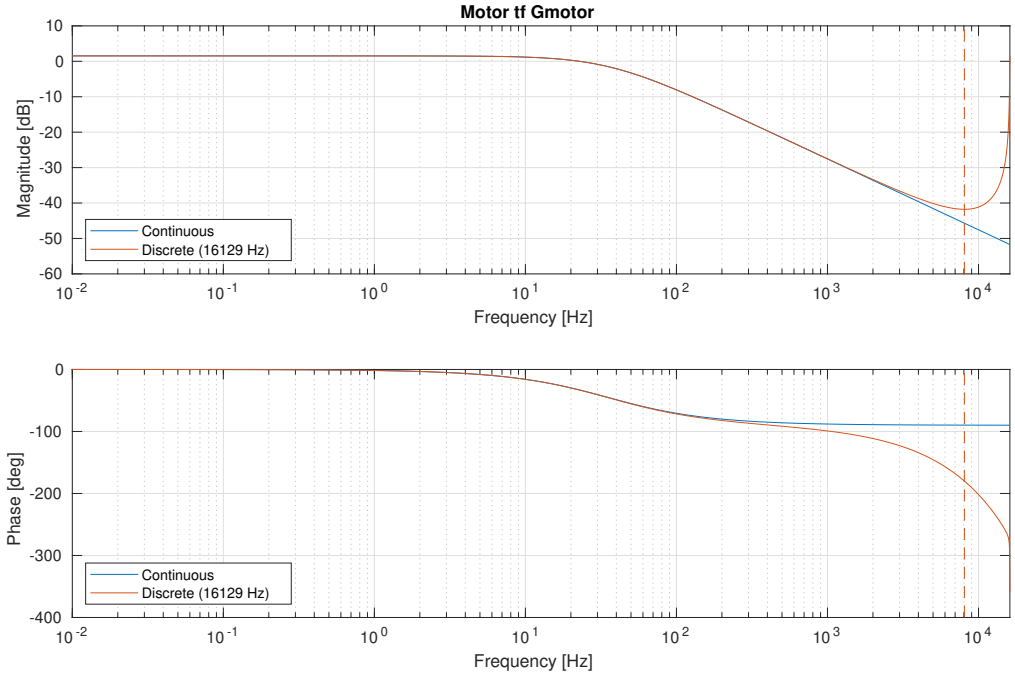


Figure 2.54: Comparaison avec la fonction de transfert discrète de Gmot
`dev/90_test/10_plant_model/motor_model/motor_model_num.m`

2.3.4.8 Discrétisation de Gcur_sens du bloc « Sensor »

La fonction de transfert correspondant au capteur de courant a été modélisée avec les paramètres ci-dessous:

$$G_{cur\_sens}(s) = \frac{1}{\tau_c \cdot s + 1}$$

$$\tau_c = \frac{1}{2 \cdot \pi \cdot f_c} = \frac{1}{2 \cdot \pi \cdot 350 \cdot 10^3} = 455\,[ns]$$

À l'équation 2.103, cette fonction de transfert est donnée sous forme de pôles/zéros.

$$G_{mot}(s) = \frac{1}{\tau_c} \cdot \frac{1}{(s - p_1)} \tag{2.103}$$

$$p_1 = -\frac{1}{\tau_c} = -2.2 \cdot 10^6$$

La fonction de transfert de l'équation 2.103 peut être discrétisée avec l'exemple de l'annexe B.5. La fonction de transfert discrète est donnée à l'équation 2.104 sous forme de polynômes avec des coefficients.

$$G_{cur\_sens}(z) = \frac{1}{\tau_c} \cdot \frac{b_0 \cdot z + b_1}{a_0 \cdot z + a_1} \tag{2.104}$$

$b_0 = 0$
$b_1 = \frac{e^{h \cdot p_1} - 1}{p_1}$
$a_0 = 1$
$a_1 = -e^{h \cdot p_1}$

Avec les valeurs numériques, cette fonction de transfert discrète (à 16129 $[Hz]$) est donnée à l'équation 2.105.

$$G_{sens\_cur}(z) = \frac{1}{z - 6.63 \cdot 10^{-60}} \tag{2.105}$$

Les diagrammes de Bode des fonctions de transfert continue et discrète sont comparés à la figure 2.55.

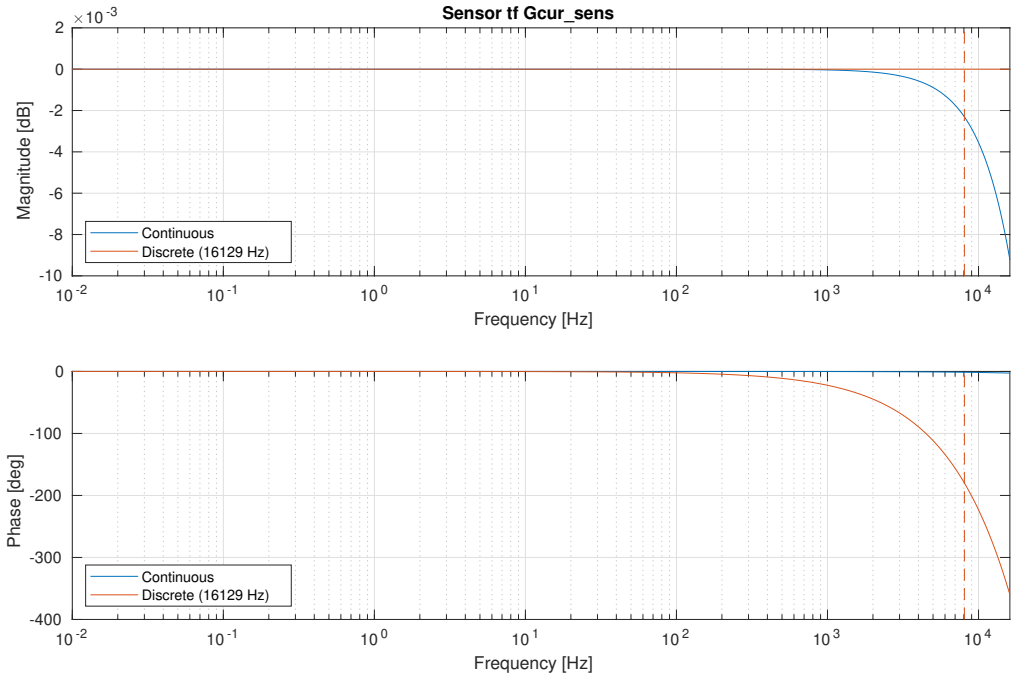


Figure 2.55: Comparaison avec la fonction de transfert discrète de Gsens_cur
`dev/90_test/10_plant_model/sensor_model/sensor_model_num.m`

2.3.4.9 Discrétisation de Gresolver du bloc « Sensor »

La fonction de transfert correspondant au resolver a été modélisée avec les valeurs numériques ci-dessous:

$$G_{resolver}(s) = \frac{2.03 \cdot 10^{-5} \cdot s^2 + 0.1372 \cdot s + 63.57}{5.031 \cdot 10^{-14} \cdot s^4 + 6.298 \cdot 10^{-10} \cdot s^3 + 2.227 \cdot 10^{-5} \cdot s^2 + 0.1372 \cdot s + 63.57}$$

À l'aide de Matlab, les zéros et les pôles de cette fonction de transfert peuvent être retrouvés:

$$z_1 = -6.26 \cdot 10^3$$

$$z_2 = -500$$

$$p_1 = -2.88 \cdot 10^3 + 19.8 \cdot 10^3 \cdot j$$

$$p_2 = -2.88 \cdot 10^3 - 19.8 \cdot 10^3 \cdot j$$

$$p_3 = -6.26 \cdot 10^3$$

$$p_4 = -504$$

Les valeurs ci-dessus permettent de voir qu'il y a des pôles et des zéros qui sont très proches. Dans une fonction de transfert, si un pôle et un zéro ont une valeur égale, ils s'annulent et l'équation peut être simplifiée. L'équation 2.106 correspond à la fonction de transfert simplifiée sous forme de pôles/zéros.

$$G_{resolver}(s) \approx p_1 \cdot p_2 \cdot \frac{1}{(s-p_1)\cdot(s-p_2)} = (a^2+b^2)\cdot\frac{1}{(s-a-b\cdot j)\cdot(s-a+b\cdot j)} \tag{2.106}$$

$$a = -2.88\cdot 10^3 \qquad b = 19.8\cdot 10^3$$

La figure 2.56 permet de comparer le diagramme de Bode de la fonction de transfert simplifiée avec l'originale.

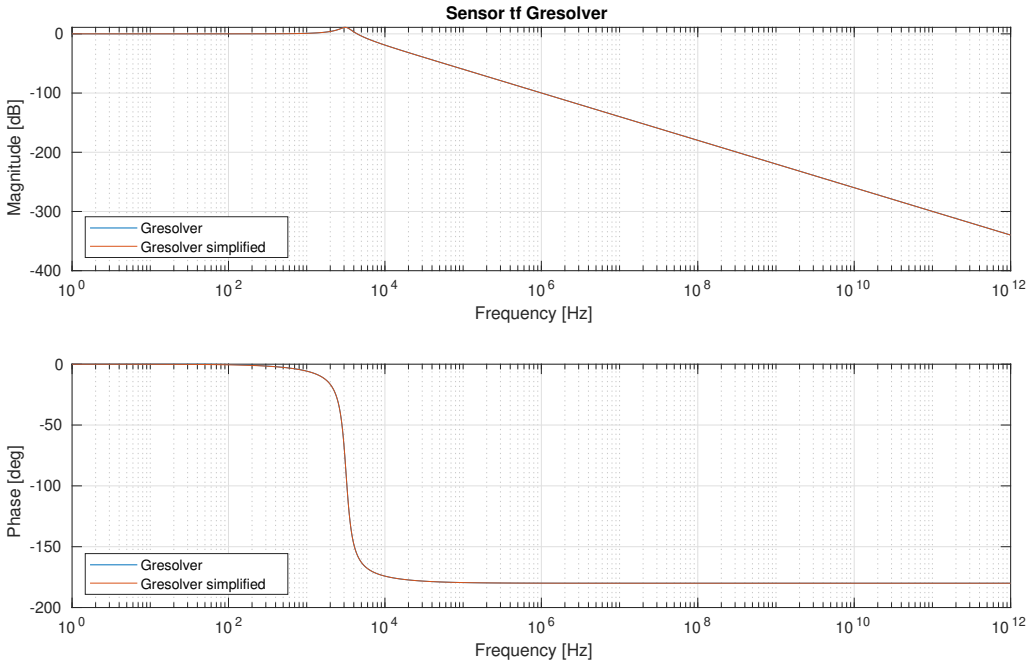


Figure 2.56: Comparaison avec la fonction de transfert Gresolver simplifiée
`dev/90_test/10_plant_model/sensor_model/sensor_model_num.m`

La figure 2.56 montre qu'avec seulement 2 pôles, la fonction de transfert du resolver est identique à celle qui a été modélisée avec 4 pôles. La fonction de transfert simplifiée de l'équation 2.106 peut être discrétisée avec l'exemple de l'annexe B.3. La fonction de transfert discrète est donnée à l'équation 2.107 sous forme de polynômes avec des coefficients.

$$G_{resolver}(z) = (a^2+b^2)\cdot\frac{b_0\cdot z^2 + b_1\cdot z + b_2}{a_0\cdot z^2 + a_1\cdot z + a_2} \tag{2.107}$$

$b_0 = 0$
$b_1 = \frac{b+a\cdot e^{h\cdot a}\cdot sin(h\cdot b)-b\cdot e^{h\cdot a}\cdot cos(h\cdot b)}{b\cdot(a^2+b^2)}$
$b_2 = \frac{b\cdot e^{2\cdot h\cdot a}-b\cdot e^{h\cdot a}\cdot cos(h\cdot b)-a\cdot e^{h\cdot a}\cdot sin(h\cdot b)}{b\cdot(a^2+b^2)}$
$a_0 = 1$
$a_1 = -e^{h\cdot a}\cdot 2\cdot cos(h\cdot b)$
$a_2 = e^{2\cdot h\cdot a}$

Avec les valeurs numériques, cette fonction de transfert discrète (à 16129 $[Hz]$) est donnée à l'équation 2.108.

$$G_{resolver}(z) = \frac{0.604 \cdot z + 0.533}{z^2 - 0.562 \cdot z + 0.7} \tag{2.108}$$

Les diagrammes de Bode des fonctions de transfert continue et discrète sont comparés à la figure 2.57.

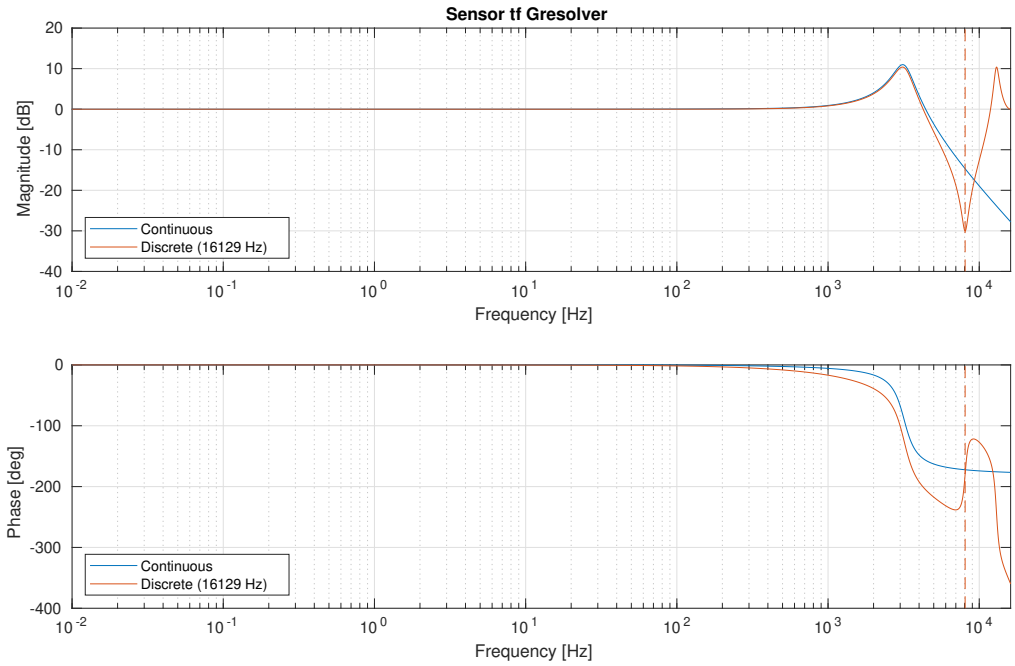


Figure 2.57: Comparaison avec la fonction de transfert discrète de Gresolver
`dev/90_test/10_plant_model/sensor_model/sensor_model_num.m`

2.4 Modélisation en SystemVerilog

Pour que le système à régler puisse être simulé dans le testbench avec le régulateur, il faut le modéliser en SystemVerilog. Ce modèle est réalisé selon les schémas des figures 2.8, 2.23, 2.31 et 2.40 avec les fonctions de transfert discrètes définies à la section 2.3.4.

2.4.1 Filtre IIR

Les fonctions de transfert discrètes doivent être simulées en étant synchronisées sur une horloge. La fréquence de l'horloge du système à régler correspond donc à la fréquence d'échantillonnage. Un module est créé (en code synthétisable) pour effectuer le calcul d'une fonction de transfert discrète selon le schéma de la figure 2.41 (filtre IIR). Au détail près que la sortie est synchronisée sur l'horloge et qu'elle est donc retardée d'une période d'échantillonnage par rapport à l'équation 2.68. Avec une fréquence d'échantillonnage élevée, ce retard ne doit pas avoir de conséquences.

En modélisant le filtre IIR sous forme de module, il peut être utilisé comme dans l'exemple ci-dessous. Le paramètre générique "IIR_ORDER" permet de définir l'ordre et donc la taille du filtre en fonction des polynômes donnés en paramètre.

```
/*************** Motor ***************/
const real  R_mot = 0.839;
const real  L_mot = 3.8e-3;
// gain/zeros/poles (of the analogic tf)
const real  gain_mot = 1.0/L_mot;
const real  p_mot = -R_mot/L_mot;
// compute tf
const real  den0_mot = 1.0;
const real  den1_mot = -$exp(p_mot*Ts);
const real  num0_mot = gain_mot*(0.0);
const real  num1_mot = gain_mot*(($exp(p_mot*Ts)-1.0)/p_mot);
real  Gmotor_num[2] = '{num0_mot,num1_mot};
real  Gmotor_den[2] = '{den0_mot,den1_mot};

IIR_filter #(.IIR_ORDER(1)) Gmotor_a(
    .clk(clk),
    .rst(rst),
    .b_coef(Gmotor_num),
    .a_coef(Gmotor_den),
    .x_in(Uin),
    .y_out(Iout));
```

Le code source de ce module est disponible dans le fichier suivant:

`dev/01_plant_model/SV_model/src/plant_model_modules.sv`

2.4.2 Lookup table

Dans le modèle du système à régler, il y a deux « Lookup Table ». La première est pour le gain entre le courant i_q et le couple électromagnétique avec une interpolation linéaire. La seconde est pour le couple du frein magnétique en fonction de la position avec une interpolation selon l'algorithme « spline ». Cette seconde technique d'interpolation exige une étape importante de calcul (algorithme dans le lien ci-dessous).

`https://en.wikipedia.org/wiki/Spline_(mathematics)`

Le choix a été fait de modéliser ces LUTs comme des objets avec des méthodes (donc non synthétisable). Ainsi, une phase d'initialisation permet de charger les données (depuis des fichiers), de choisir la technique d'interpolation et d'effectuer les calculs nécessaires en début de simulation. Ensuite, une simple méthode "interp" permet d'interpoler la LUT en fonction de la technique préalablement choisie. Un exemple d'utilisation est donné ci-dessous:

```
LUT Tem_lut, brake_lut;

initial begin
    Tem_lut = new("Kt_torque_current.dat", "LINEAR");
    Tem_lut.init();
    brake_lut = new("brake_torque_pos.dat", "SPLINE");
    brake_lut.init();
end

always_comb begin
    Tem <= Tem_lut.interp(Iq);
    Tbrake <= brake_lut.interp(pos);
end
```

Le code source de la classe "LUT" est disponible dans un « package » dans le fichier suivant:

`dev/01_plant_model/SV_model/src/plant_model_pkg.sv`

2.4.3 Comparaison entre modèle Matlab et SystemVerilog

Pour s'assurer que le modèle du système à régler est correctement modélisé, une simulation en boucle ouverte peut être réalisée pour comparer le modèle Matlab avec le modèle SystemVerilog. Dans cette simulation, la fréquence d'échantillonnage du système à régler est de 1.6 $[MHz]$ (pour les deux modèles).

Pour simuler le système à régler en boucle ouverte, la commande et les valeurs de sortie ont été sauvegardées dans un fichier lors de la simulation Matlab (lien ci-dessous). La commande sauvegardée et utilisée pour la comparaison est montrée à la figure 2.58.

`dev/01_plant_model/simulink_model/07_filter_modified/filter_modified_AY7.m`

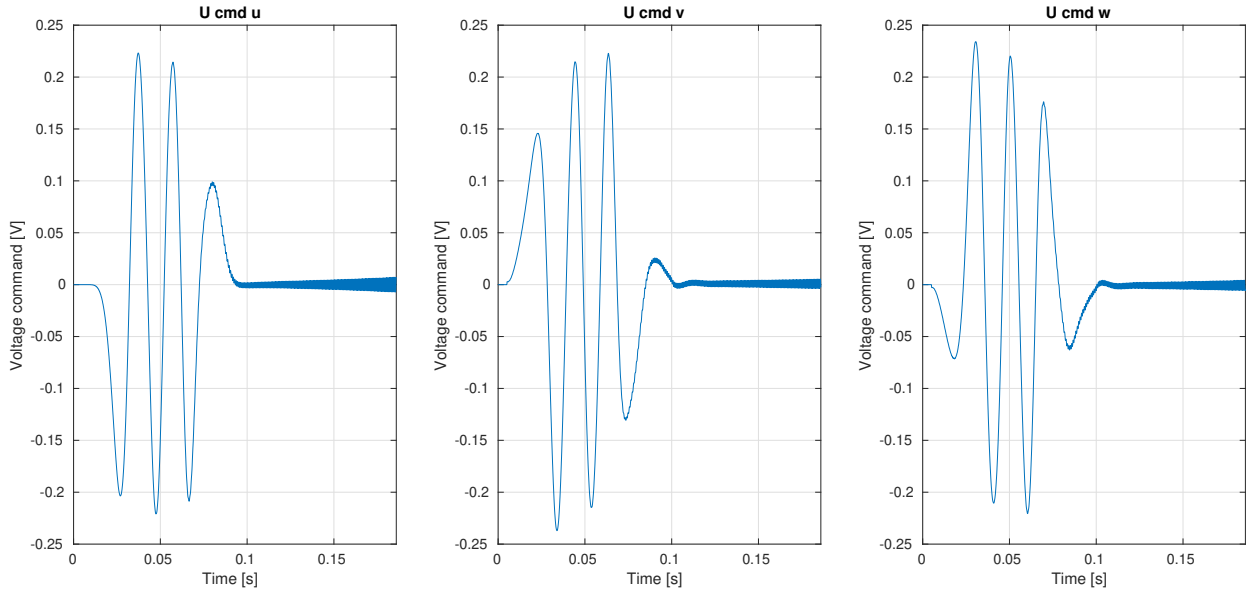


Figure 2.58: Tensions de commande pour la comparaison en boucle ouverte entre les modèles du système à régler (Matlab - SystemVerilog)
`dev/01_plant_model/simulink_model/07_filter_modified/filter_modified_AY7.m`

Dans la simulation SystemVerilog, la commande est importée depuis le fichier pour effectuer la simulation. Les valeurs de sortie du modèle SystemVerilog sont également enregistrées dans un fichier. Après la simulation, un script (python) compare les signaux de sortie des deux modèles. La simulation SystemVerilog et la comparaison des signaux sont automatiquement lancées depuis le script suivant:

`dev/01_plant_model/SV_model/tb/plant_model_tb.py`

La figure 2.59 montre la comparaison des courants de sortie entre les deux modèles simulés.

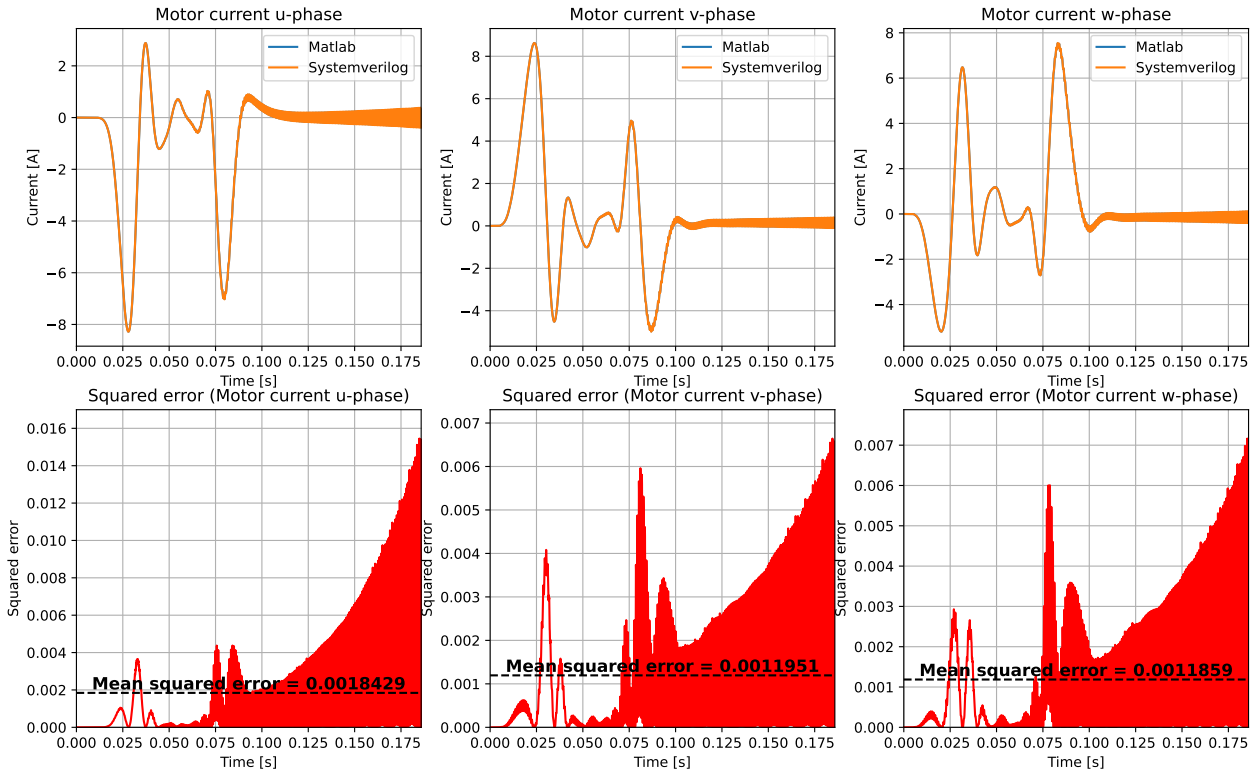


Figure 2.59: Comparaison du courant en boucle ouverte entre les modèles du système à régler (Matlab - SystemVerilog)
`dev/01_plant_model/SV_model/tb/script/signals_compare.py`

Ci-dessus, les graphes du haut montrent une bonne cohérence entre les modèles. L'allure générale et les gains semblent corrects. Les graphes du bas permettent d'observer l'erreur plus en détail. Avec une erreur MSE qui ne dépasse pas 0.002 comparée à la valeur maximale du signal de 8 $[A]$, cette différence semble tout à fait acceptable. Il est également important de noter que l'erreur la plus importante se situe à la fin du mouvement, lorsque la commande (qui a été générée dans une simulation en boucle fermée) montre des signes d'instabilités.

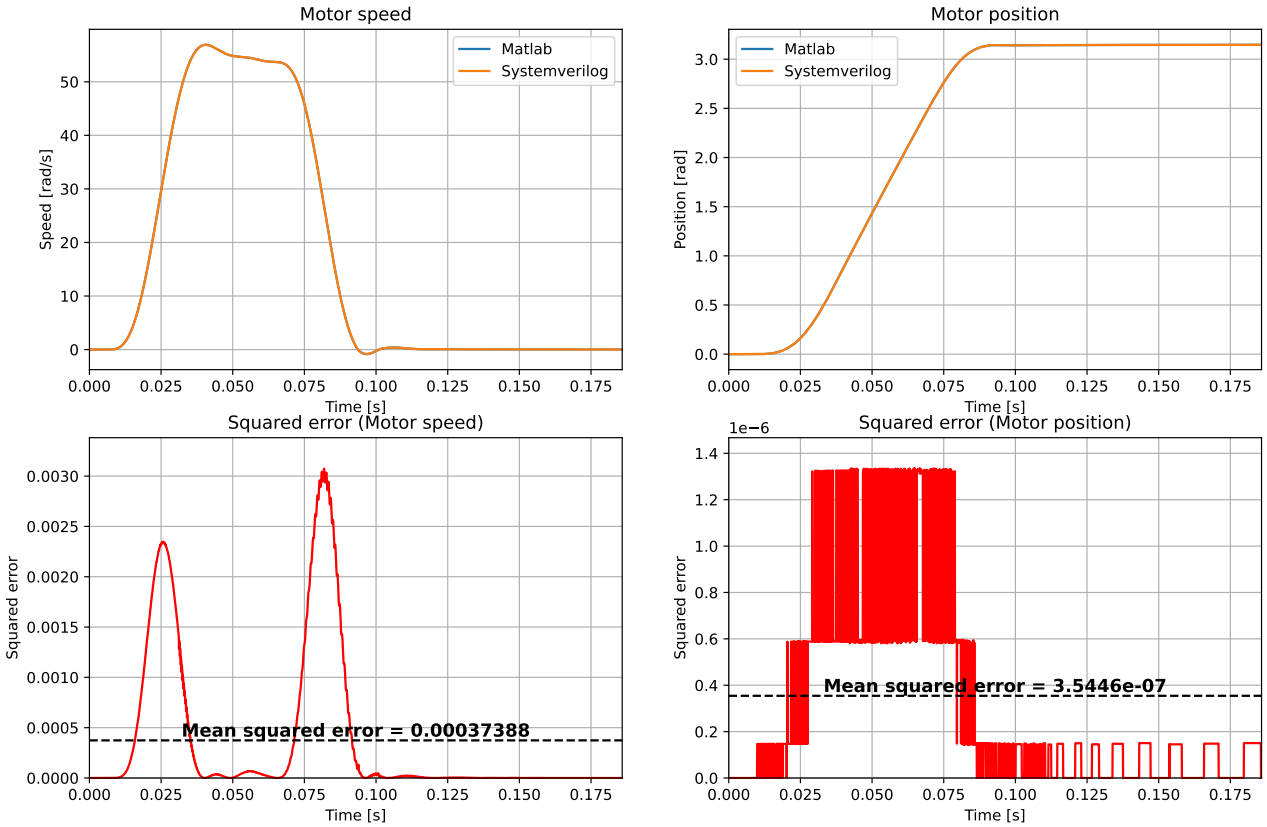


Figure 2.60: Comparaison de la vitesse et de la position en boucle ouverte entre les modèles du système à régler (Matlab - SystemVerilog)
`dev/01_plant_model/SV_model/tb/script/signals_compare.py`

Sur la figure 2.60, ce sont la vitesse et la position des deux modèles qui sont comparées. Pour ces deux grandeurs, l'erreur est très faible. Pour la vitesse, l'erreur augmente durant l'accélération et elle est très faible lorsque la vitesse est constante. Pour la position, il est important de noter que l'erreur fait des sauts. La raison de ces sauts est que dans le modèle du système à régler, la position est quantifiée (sur 14 bits) pour simuler la résolution du RDC (équation 2.63). Comme pour la vitesse, c'est lorsque le signal varie que l'erreur est la plus importante.

Les résultats visibles sur les figures 2.59 et 2.60 sont suffisamment bons pour dire que le modèle du système à régler en SystemVerilog est identique (en boucle ouverte) au modèle Matlab.

3 Régulateur

Le régulateur est l'élément qui devra être vérifié dans le testbench. Pour vérifier le bon fonctionnement de ce régulateur, il est intéressant d'avoir un autre modèle de régulateur utilisé comme référence. Jusqu'à présent, le modèle idéal du régulateur était simulé avec Matlab/Simulink. Ce modèle idéal est étudié à la section 3.1.

La section 3.2 concerne le régulateur à vérifier (en VHDL). N'ayant pas été validé jusqu'à présent, le code de celui-ci ne doit pas être pris comme référence et il n'est pas nécessaire, dans un premier temps, d'étudier précisement ce design à vérifier. Dans la simulation, il peut être vu comme une boîte noire sans se soucier de la manière dont les fonctionnalités ont été implémentées. Cependant, il est quand même important de comprendre quelles sont les contraintes et les différences par rapport au régulateur de référence qui est idéal. De plus, il est nécessaire d'étudier les entrées et les sorties de ce composant afin de l'interfacer correctement dans le testbench.

Enfin dans la section 3.3, un régulateur de référence est modélisé en SystemVerilog. Celui-ci doit être idéal comme le régulateur de référence utilisé jusqu'à maintenant (section 3.1). Ce modèle SystemVerilog pourra être simulé dans le testbench en parallèle du régulateur à vérifier.

3.1 Présentation du régulateur idéal (Matlab)

La tâche du régulateur est de fournir une commande triphasée au système à régler de manière à effectuer le mouvement mécanique souhaité. Connaissant à l'avance le mouvement mécanique et les propriétés physiques du moteur, les trajectoires ont été pré-calculées.

Les grandeurs pré-calculées dans les trajectoires sont la position angulaire mécanique, la vitesse angulaire mécanique et le courant électrique du moteur. Le courant électrique du moteur calculé correspond au courant "utile" nommé i_q qui a un impact sur le couple (voir section 2.2.3.2). L'autre courant qui est complémentaire à i_q est le courant i_d qui n'a aucun impact sur le couple et qui doit être idéalement nul. Cette manière de réguler le courant dans un moteur triphasé s'appelle la commande vectorielle.

3.1.1 Commande vectorielle

Un schéma de la boucle de courant d'une commande vectorielle est montré à la figure 3.1. Pour que le régulateur exécute les calculs de courant dans la représentation dq en ayant un feedback et une commande triphasée, il est nécessaire de passer par la transformée de Park et son inverse (voir section 2.2.3.2). Le schéma de la figure 3.2 montre la manière dont le régulateur dq est interfacé avec les signaux du système à régler.

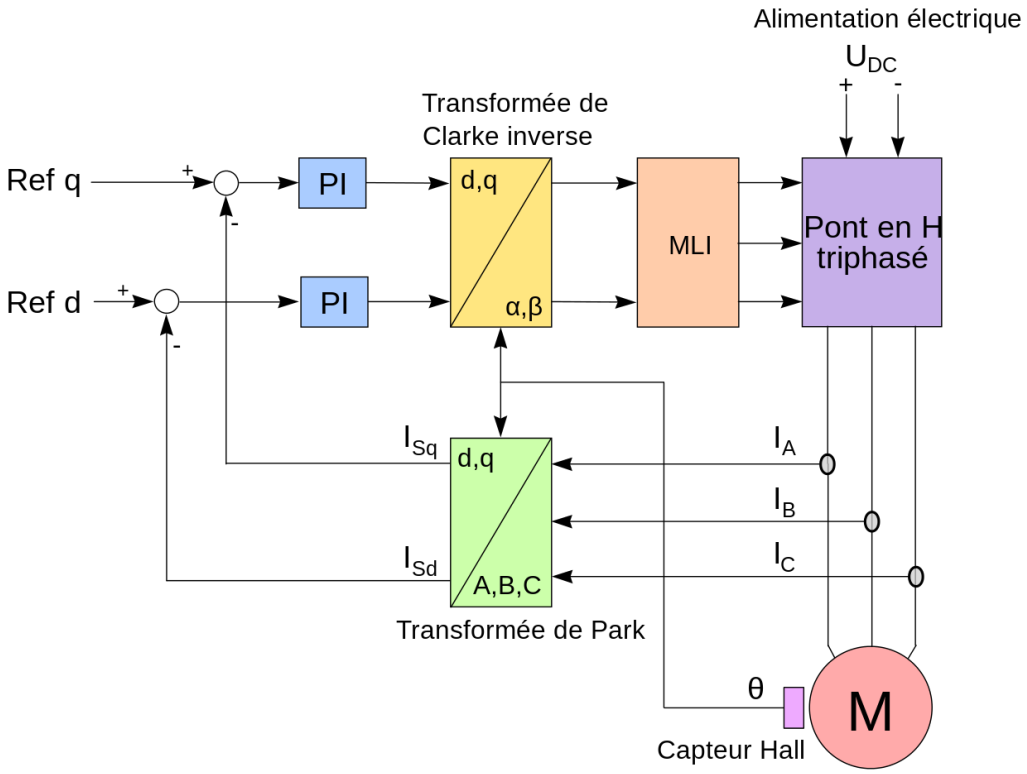


Figure 3.1: Exemple de commande vectorielle d'un moteur triphasé
https://fr.wikipedia.org/wiki/Commande_vectorielle

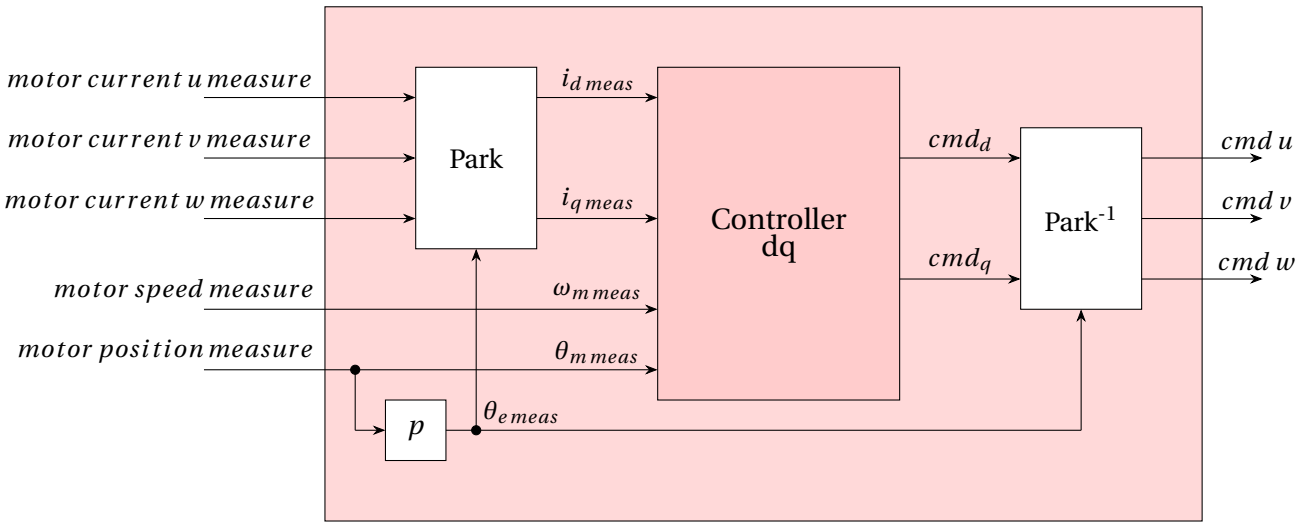


Figure 3.2: Interface du régulateur idéal avec les transformées de Park

Avec la structure de la figure 3.2, le régulateur peut être modélisé avec des courants i_d et i_q pour lesquels la régulation est plus intuitive qu'avec les trois courants de phase. Il est important de noter que dans la modélisation ci-dessus, la saturation de la commande n'y est pas. Si elle doit être modélisée, il faut placer la saturation au niveau de la commande triphasée, sur chacune des phases.

3.1.2 Structure du régulateur

Le schéma du régulateur dq est montré à la figure 3.3. Dans ce régulateur, il y a trois boucles de régulation imbriquées en plus de la régulation de i_d qui est placée en parallèle. Comme le montre la modélisation du moteur de la figure 2.33, les phases d et q sont couplées. Afin d'optimiser la régulation, les blocs « decoupling » doivent compenser le couplage du système à régler. Les signaux sortants des blocs « decoupling » sont sommés à la commande à la sortie des blocs « PI » (figure 3.6).

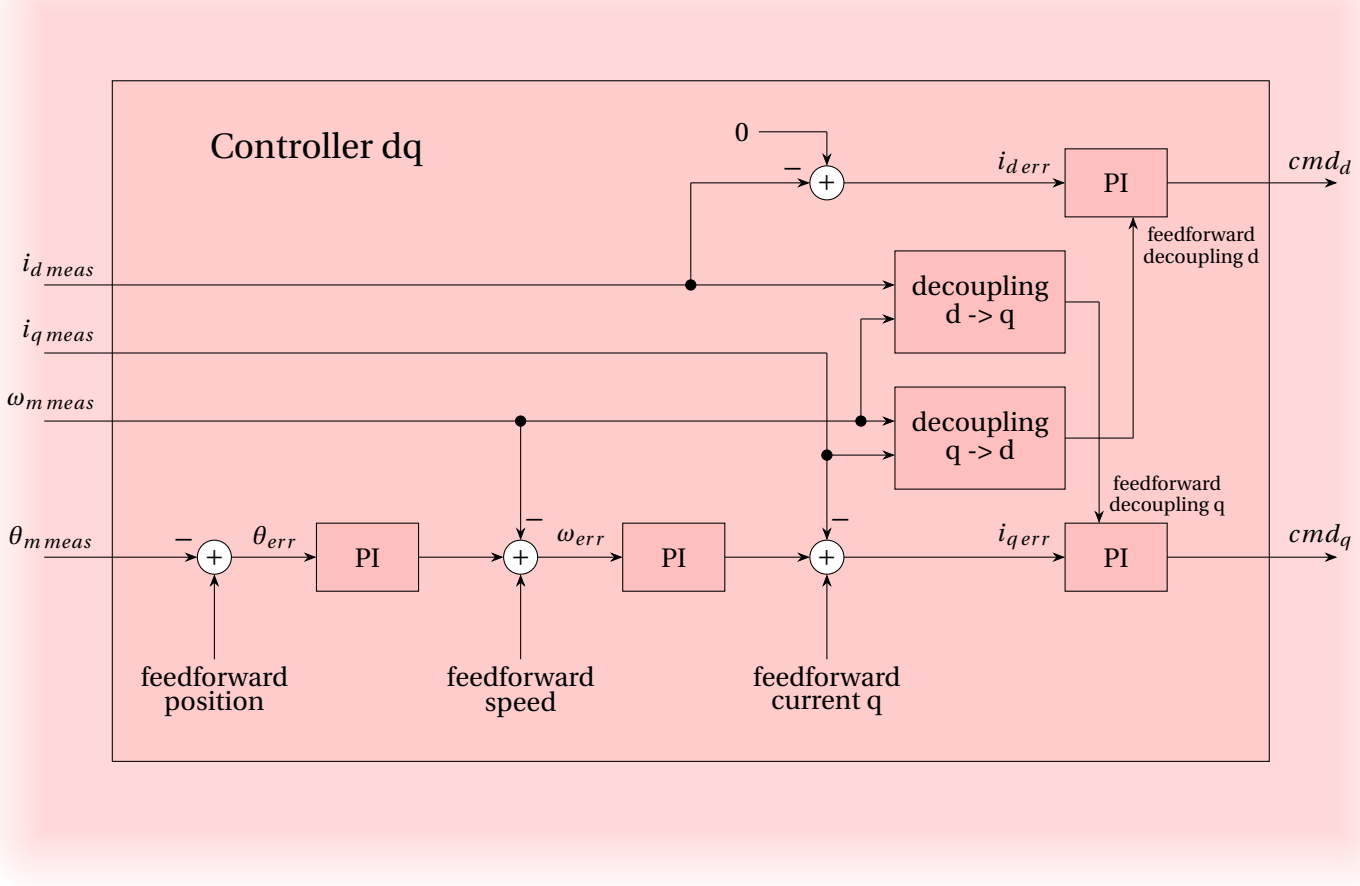


Figure 3.3: Structure du régulateur idéal (représentation dq)

3.1.3 Découplage

Pour comprendre comment compenser le couplage, il faut observer le schéma de la figure 2.33. Sur la phase d, le couplage a été défini à l'équation 2.57 et vaut:

$$\omega_m \cdot p \cdot i_q \cdot L_s$$

Pour annuler ce couplage depuis la sortie du régulateur, il faut prendre en compte tous les gains qui se situent entre ces deux signaux. En prenant en compte les gains, le découplage pour la phase d est donné à la figure 3.4

Le bloc « Filter » est modélisé avec des fonctions de transfert dont le gain varie en fonction de la fréquence. Mais à basses fréquences, le gain du filtre est unitaire. Il n'y a donc pas besoin de le prendre en compte.

Dans le bloc « Power » il y a une fonction de transfert avec un gain unitaire et un gain de $U_e/2$. Pour annuler ce gain lorsque le signal passe par ce bloc, il faut multiplier la compensation du couplage par $2/U_e$. Un signe négatif est ajouté pour que ce découplage puisse annuler le couplage.

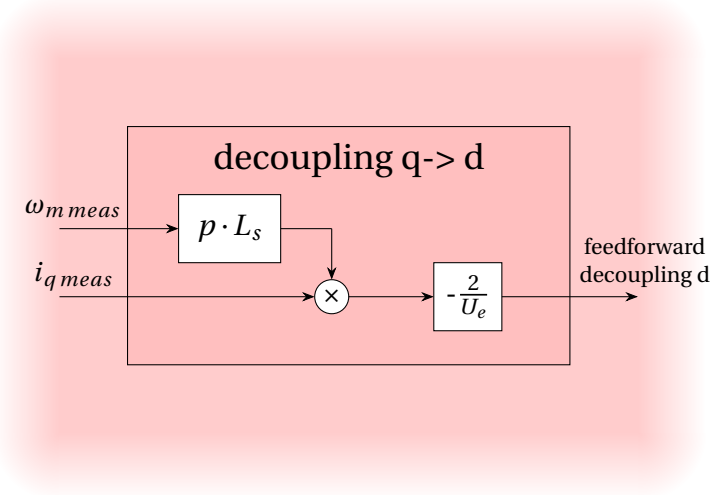


Figure 3.4: Découplage q -> d

En se référant à la figure 2.33 ou à l'équation 2.58, le couplage pour la phase q vaut:

$$-\omega_m \cdot (K_e + p \cdot i_d \cdot L_s)$$

Comme pour la phase d, il faut inverser tous les gains et inverser le signe pour annuler ce couplage. Le découplage pour la phase q est donné à la figure 3.5.

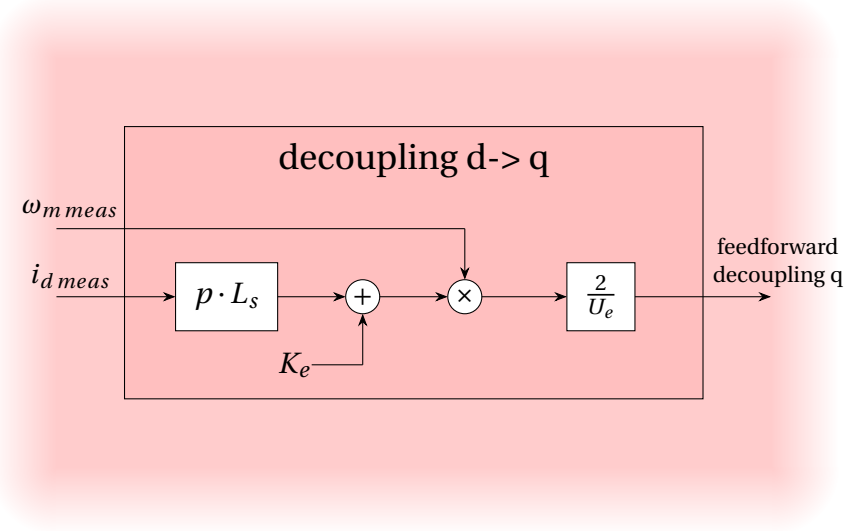


Figure 3.5: Découplage d -> q

3.1.4 Régulateur PI

Les blocs PI sont les éléments principaux du régulateur. Ils permettent de compenser une erreur. Il y en a un pour chaque grandeur à régler (position, vitesse et les courants d et q). Chaque bloc a une structure identique mais avec des paramètres différents.

Un régulateur PI (Proportional-Integral) est composé d'une action proportionnelle et d'une action intégrale. L'action proportionnelle permet d'amplifier le signal avec un gain sans aucun impact sur la dynamique du système. L'action intégrale permet d'intégrer les échantillons d'entrée et cela a pour effet de lisser le signal. D'un point de vue fréquentiel, un régulateur PI permet de filtrer les hautes fréquences et donc d'effectuer la correction sur les basses fréquences. L'équation de ce type de régulateur est la suivante:

$$u[k] = e[k] \cdot \left(K_p + K_i \cdot \frac{Ts}{z-1} \right) \tag{3.1}$$

Où:
$u[k]$ est le signal de sortie
$e[k]$ est le signal d'entrée (représentant une valeur d'erreur)
K_p $[-]$ est le gain proportionnel
K_i $[-]$ est le gain intégral
Ts $[s]$ est la période d'échantillonnage

L'équation 3.1 correspond au régulateur PI numérique de base. Dans le cas du régulateur utilisé dans ce travail, il est un peu plus complexe. Premièrement, la figure 3.3 montre que les signaux de découplage sont utilisés dans les blocs « PI » à la sortie du régulateur. Ces signaux de découplage sont simplement ajoutés à la valeur de commande. Cette somme pourrait être faite en dehors du bloc (juste après) mais elle est faite à l'intérieur car le découplage doit être pris en compte dans la saturation.

En effet, il est souhaitable que chaque régulateur contienne une saturation à la sortie. De cette manière, un régulateur qui est trop agressif (avec un gain trop élevé) est limité par la saturation. Cela empêche de propager une erreur trop importante au travers des autres boucles de régulation. De plus, la commande étant de toute façon limitée, la saturation de cette commande peut déjà être prise en compte dans les régulateurs de courant.

Dans le cas où l'erreur est très grande et que le régulateur n'arrive pas à compenser l'erreur (par exemple à cause d'une saturation), les valeurs de l'erreur s'accumulent dans l'intégrale. Lorsque l'erreur revient à une valeur proche de 0, l'intégrale contient toujours une valeur très grande. Dans un tel cas, l'intégrale met du temps à revenir à un niveau normal et cela a un impact négatif sur le comportement du système. Ce problème est bien connu et se nomme « Integral windup ».

Pour remédier à ce problème, il existe une technique nommée « Anti-windup ». Cette technique consiste à bloquer l'intégrateur si la sortie est saturée. De cette manière, l'erreur n'est pas accumulée dans l'intégrateur et elle n'a donc pas à être compensée par la suite. Le schéma de la figure 3.6 correspond au régulateur PI avec le "feedforward" (utilisé pour le couplage), la saturation et l' « anti-windup ». La bascule D doit être synchronisée avec un signal d'horloge correspondant à la fréquence d'échantillonnage du régulateur (donc celle du PWM: 16129 $[Hz]$).

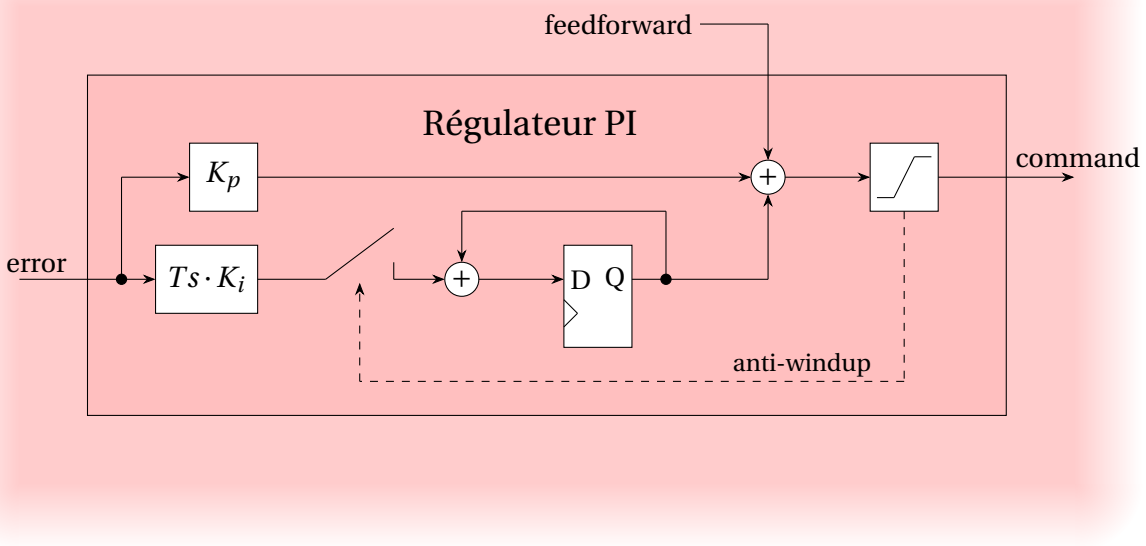


Figure 3.6: Schéma du régulateur PI

3.2 Régulateur à vérifier (VHDL)

Le régulateur à vérifier est donné en language VHDL. Il s'agit du composant qui doit être évalué dans le testbench. Pour juger si le design de ce régulateur est correct, il faut le comparer au régulateur idéal (section 3.1). Puisque le régulateur à vérifier est un design synthétisable avec un certain nombre de contraintes, il faut de toute façon s'attendre à des différences entre ce modèle et celui de référence.

3.2.1 Commande réelle

La différence la plus évidente entre le régulateur réel et le modèle idéal est la manière dont est générée la commande. Dans le régulateur idéal, la sortie du régulateur (pour chacune des trois phases) est mise sous forme de nombre réel entre -1 et 1. Dans le cas du régulateur réel, la commande est composée de six signaux binaires. Comme expliqué à la section 2.2.1, chaque phase contient deux signaux ("up" et "down") permettant de générer une tension positive ou négative (par rapport à la tension moyenne) à travers l'onduleur du système à régler.

Pour obtenir une valeur entre -1 et 1, les signaux de commande sont modulés sous forme de PWM. La résolution de cette commande dépend du rapport entre la fréquence d'horloge de la FPGA et la fréquence du PWM (équation 2.1). La modulation PWM est composée de temps-morts (figure 2.5). Ces temps-morts entraînent une diminution de la résolution mais aussi une diminution de l'amplitude maximale du signal. En effet, les temps-morts mettent le signal PWM à 0 durant un instant à chaque période de PWM. Cela signifie qu'il n'est pas possible d'avoir une période complète de PWM à l'état 1 et que la valeur maximale de 1.0 n'est pas atteignable.

La modulation PWM est un peu trop complexe pour être modélisée aisément dans le modèle de référence du régulateur. Cependant, cette différence devra être prise en compte lors de l'instanciation du régulateur à vérifier dans le testbench et lors de l'analyse des résultats.

3.2.2 Synchronisation des régulateurs PI

Dans le régulateur idéal, les données sont reçues et transmises entre chaque cycle du PWM (16129 $[Hz]$). Les régulateurs PI sont également échantillonnés à la fréquence du PWM.

Dans le système réel, la fréquence d'horloge de la FPGA est de 40 $[MHz]$. Cela signifie que les PIs ne peuvent pas être synchronisés directement sur la fréquence d'horloge. Dans le design à vérifier, les signaux qui traversent les PIs sont accompagnés d'informations permettant de savoir quand la valeur du signal est valide ou non. De cette manière, le PI peut être synchronisé sur l'horloge à 40 $[MHz]$ mais le calcul n'est effectué que lorsque le signal d'entrée est considéré comme valide. Il y a donc un flux de données synchronisées entre les différents PIs. Avec une horloge de 40 $[MHz]$, les données traversent le régulateur (à travers les PIs) en moins d'une période de PWM. Ensuite les données sont transmises à la sortie par le modulateur PWM.

Ce flux de données à travers les régulateurs PI ne correspond pas tout à fait au modèle idéal vu à la section 3.1. Sur le schéma du PI idéal (figure 3.6), l'action intégrateur est sychronisée sur l'horloge à la fréquence du PWM. Cela signifie que l'information (à travers l'action intégrale) est bloquée entre chaque cycle de PWM. De cette manière, l'erreur de position a besoin de trois périodes de PWM pour que l'information arrive à la sortie du régulateur (sans prendre en compte l'action proportionnelle). Dans le cas du régulateur à vérifier, l'information entre l'entrée et la sortie du régulateur ne prend qu'une période de PWM (l'information n'est bloquée dans l'action intégrale des PIs que durant quelques coups d'horloge de la FPGA). Dans le cas où l'action intégrale est importante, il peut y avoir une différence dans la dynamique du système en boucle fermée sur la régulation de position et de vitesse.

Ces deux manières d'implémenter l'action intégrale du régulateur PI sont différentes mais toutes les deux correctes. Il s'agit de méthodes d'intégration différentes. Dans le cas du régulateur qui bloque le signal (équation 3.1), la méthode d'intégration utilisée se nomme « Forward Euler ». En décomposant l'équation sous forme de signal discret, la sortie peut être exprimée de la manière suivante:

$$u[k] = u[k-1] + e[k] \cdot K_p + e[k-1] \cdot K_i \cdot Ts$$

Dans l'équation ci-dessus, l'action intégrale fait intervenir $e[k-1]$ qui est l'erreur retardée d'un coup d'horloge. L'autre méthode s'appelle « Backward Euler » et l'équation est donnée ci-dessous:

$$u[k] = e[k] \cdot \left(K_p + K_i \cdot \frac{Ts \cdot z}{z-1} \right) \tag{3.2}$$

En exprimant la sortie de ce régulateur PI en fonction des échantillons discrets de l'erreur, cette équation devient:

$$u[k] = u[k-1] + e[k] \cdot K_p + e[k] \cdot K_i \cdot Ts$$

Cette fois-ci, l'action intégrale est directe. Les données ne sont pas bloquées à travers l'action intégrale et cette façon de calculer le régulateur PI correspond à ce qui est implémenté dans le régulateur à vérifier.

3.2.3 Interface

L'entité du régulateur à vérifier avec toutes ces entrées/sorties est donnée ci-dessous:

```
entity bws_controller_top is
    port(
        clk_i                                       : in    std_logic;
        rst_i                                       : in    std_logic;

        resolver_angle_sfixed_16_16_i               : in    std_logic_vector(31 downto 0);
        resolver_angle_sfixed_dv_i                  : in    std_logic;
        integrated_pos_rst_i                        : in    std_logic;
        resolver_spd_sfixed_16_16_i                 : in    std_logic_vector(31 downto 0);
        resolver_spd_sfixed_dv_i                    : in    std_logic;

        mechanical_offset_sfixed_i                  : in    std_logic_vector(31 downto 0);
        electrical_angle_sfixed_8_8_o               : out   std_logic_vector(15 downto 0);

        motor_current_u_sfixed_16_16_i              : in    std_logic_vector(31 downto 0);
        motor_current_u_sfixed_dv_i                 : in    std_logic;
        motor_current_v_sfixed_16_16_i              : in    std_logic_vector(31 downto 0);
        motor_current_v_sfixed_dv_i                 : in    std_logic;
        motor_current_w_sfixed_16_16_i              : in    std_logic_vector(31 downto 0);
        motor_current_w_sfixed_dv_i                 : in    std_logic;

        trajectory_speed_selection_i                        : in    std_logic_vector(31 downto 0);
        trajectory_start_scan_in_i                          : in    std_logic;
        trajectory_start_scan_out_i                         : in    std_logic;
        trajectory_dq_frame_q_mult_in_sfixed_16_16_i        : in    std_logic_vector(31 downto 0);
        trajectory_dq_frame_q_mult_out_sfixed_16_16_i       : in    std_logic_vector(31 downto 0);

        pwm_modulator_enable_i                      : in    std_logic;
        pwm_modulator_offset_phase_u_i              : in    std_logic_vector(15 downto 0);
        pwm_modulator_offset_phase_v_i              : in    std_logic_vector(15 downto 0);
        pwm_modulator_offset_phase_w_i              : in    std_logic_vector(15 downto 0);
        pwm_modulator_outputs                       : out   std_logic_vector(5 downto 0);

        controller_feedback_spd_enable_i        : in std_logic;
        controller_feedback_pos_enable_i        : in std_logic;

        avalon_q_frame_current_offset_i         : in std_logic_vector(31 downto 0);
        avalon_d_frame_current_offset_i         : in std_logic_vector(31 downto 0);

        cur_pi_controller_prop_gain_set_i       : in std_logic_vector(31 downto 0);
        cur_pi_controller_integ_gain_set_i      : in std_logic_vector(31 downto 0);
        cur_pi_controller_sat_high_set_i        : in std_logic_vector(31 downto 0);
        cur_pi_controller_sat_low_set_i         : in std_logic_vector(31 downto 0);
        spd_pi_controller_prop_gain_set_i       : in std_logic_vector(31 downto 0);
        spd_pi_controller_integ_gain_set_i      : in std_logic_vector(31 downto 0);
        spd_pi_controller_sat_high_set_i        : in std_logic_vector(31 downto 0);
        spd_pi_controller_sat_low_set_i         : in std_logic_vector(31 downto 0);
        pos_pi_controller_prop_gain_set_i       : in std_logic_vector(31 downto 0);
        pos_pi_controller_integ_gain_set_i      : in std_logic_vector(31 downto 0);
        pos_pi_controller_sat_high_set_i        : in std_logic_vector(31 downto 0);
        pos_pi_controller_sat_low_set_i         : in std_logic_vector(31 downto 0)
    );
end bws_controller_top;
```

dev/02_controller/HDL-synthetised-2022/bws_controller_top.vhd

Parmi les signaux d'entrées du régulateur à vérifier, ceux qui contiennent le suffixe "_dv_i" sont des signaux qui indiquent que les données sont valides (« data valid »). Ils doivent être actifs durant un coup d'horloge (de la FPGA) et permettent donc de synchroniser les données d'entrées.

Le signal d'horloge de la FPGA à l'entrée "clk_i" doit être de 40 $[MHz]$. Un signal de reset doit être mappé à l'entrée "rst_i" et doit obligatoirement être actif durant au moins un coup d'horloge en début de simulation pour initialiser certains composants.

Sur le schéma du régulateur idéal de la figure 3.2, les signaux d'entrées sont les trois courants de phases, la vitesse et la position. Dans l'entité du régulateur à vérifier, ces signaux sont respectivement aux lignes 15, 17, 19, 9 et 6. Ils sont définis en virgule fixe (sfixed_16_16).

Les signaux qui contiennent le préfixe "trajectory" sont utilisés pour la génération des trajectoires. L'entrée "trajectory_speed_selection_i" permet de choisir entre les quatre trajectoires possibles. Les quatre valeurs possibles sont 33, 55, 110 et 133 (en référence à la vitesse de rotation en $[rad/s]$). Les signaux "start_scan" permettent de démarrer le mouvement dans le sens choisi. Enfin, les signaux "dq_frame_q_mult" permettent d'ajuster la consigne du courant q en ajoutant un gain. Si les trajectoires sont correctement générées, ces deux entrées peuvent être mise à 1.0 (sfixed_16_16).

Les signaux entre les lignes 40 et 51 sont utilisés pour paramétrer les régulateurs PI. Pour chaque type de régulateur (courant, vitesse et position), il faut définir le gain proportionnel, le gain intégral, et les niveaux de saturations. Ces données doivent être définies en virgule fixe (sfixed_16_16).

Les deux signaux avec le préfixe "controller_feedback" permettent d'activer ou de désactiver les boucles de régulation de la position et de la vitesse.

Enfin, le signal de sortie "pwm_modulator_outputs" est défini sur 6 bits. Ces signaux vont par paires (pour chacune des phases) et sont modulés sous forme de PWM (comme expliqué à la section 2.2.1). Le signal d'entrée "pwm_modulator_enable_i" permet d'activer la modulation PWM.

Les autres signaux qui n'ont pas été cités ci-dessus ne sont pas utilisés.

3.3 Modélisation en SystemVerilog

Le régulateur de référence est modélisé en SystemVerilog. Celui-ci pourra être simulé dans le testbench en parallèle du régulateur à vérifier. Le code de ce régulateur de référence se trouve dans les fichiers suivants:

```
dev/02_controller/SV_controller/src/controller.sv
dev/02_controller/SV_controller/src/controller_modules.sv
```

La modélisation de ce régulateur de référence en SystemVerilog ne contient pas de spécialités, ce code peut être entièrment synthétisable. Ce modèle est conforme aux schémas des figures 3.2, 3.3, 3.4, 3.5 et 3.6. Sauf que, comme pour le régulateur à vérifier, l'horloge est donnée à 40 $[MHz]$ alors que les régulateurs PI doivent être échantillonnés à la fréquence du PWM qui est de 16129 $[Hz]$.

De plus, les signaux d'entrées du régulateur viennent du système à régler et la fréquence de ceux-ci n'est pas définie de manière fixe. Il est donc nécessaire d'ajouter un signal de synchronisation en entrée afin d'indiquer lorsque les données doivent être traitées. Puisque le régulateur idéal est censé fonctionner à la fréquence du PWM (16129 $[Hz]$), ce signal de synchronisation doit correspondre à cette fréquence. À l'entrée du modèle de ce régulateur, les valeurs des données sont mises à jour et maintenues à chaque fois que le signal de synchronisation est actif (comme un bloqueur d'ordre zéro).

Ensuite, ce signal de synchronisation peut être utilisé pour faire circuler les informations à travers les régulateurs PI. Le modèle du régulateur PI contient donc des signaux de synchronisation en entrée et en sortie. De cette manière, chaque régulateur PI indique au régulateur PI suivant lorsque les données sont valides. Ce flux de données à travers les PIs correspond à ce qui est fait dans le régulateur à vérifier. Comme expliqué dans la section 3.2.2, cette manière d'implémenter ce flux de données entre les PIs est différente de ce qui a été montré à la section 3.1. Cette implémentation correspond à des régulateurs PI dont l'intégrale est calculée avec la méthode « Backward Euler ».

Une autre modification a été faite au niveau du gain des régulateurs de courant. Dans le régulateur à vérifier, les paramètres du régulateur PI de courant sont multipliés par 100 par rapport au modèle Matlab. La raison est que le modulateur PWM du régulateur à vérifier a un signal d'entrée qui doit varier entre -100 et 100. Ce facteur n'a donc aucun impact sur le fonctionnement. Pour pouvoir comparer les signaux internes entre les deux régulateurs, ce gain de 100 est également présent dans le modèle de référence. Après la transformée inverse de Park, à la sortie du régulateur, les signaux de commande sont divisés par 100 pour retrouver une valeur entre -1 et 1. Puisque les signaux de découplage sont connectés à la sortie des régulateurs PI de courant, il est également nécessaire de multiplier le découplage par 100.

3.3.1 Comparaison entre le modèle Matlab et SystemVerilog

Pour valider le modèle du régulateur en SystemVerilog, il peut être simulé en boucle ouverte et comparé avec celui de Matlab (comme pour le modèle du système à régler).

Les signaux de courants, de vitesse et de position peuvent être obtenus lors d'une simulation en boucle fermée sur Matlab. Ces signaux sont les mêmes que ceux qui étaient utilisés pour la vérification du système à régler. Ainsi, les signaux d'entrées du régulateur sont ceux qui sont comparés aux figures 2.59 et 2.60 (courbes de la simulation Matlab).

```
dev/01_plant_model/simulink_model/07_filter_modified/filter_modified_AY7.m
```

Comme pour la vérification du système à régler, un script python permet de lancer la simulation et de comparer les résultats. Dans le cas du régulateur, les signaux de sortie à comparer sont ceux de la commande (triphasée).

`dev/02_controller/SV_controller/tb/controller_ref_tb.py`

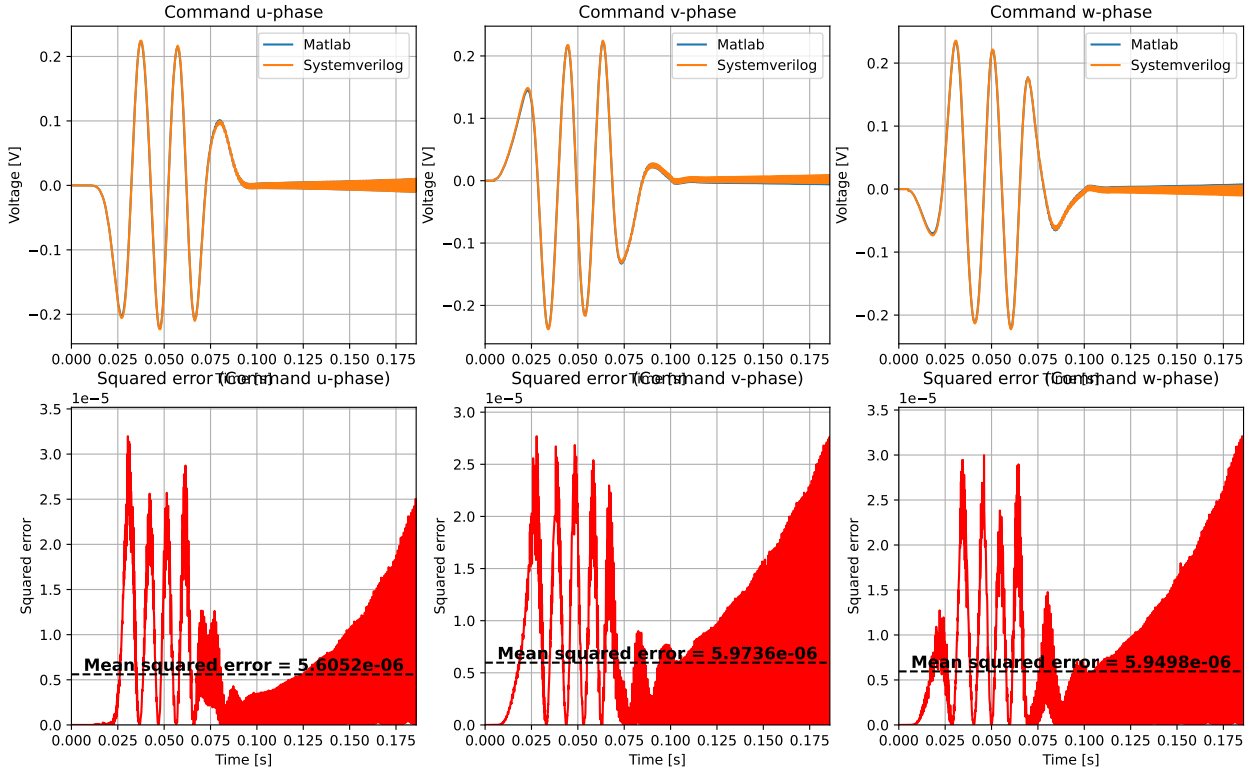


Figure 3.7: Comparaison de la commande en boucle ouverte entre les modèles du régulateur (Matlab - SystemVerilog)
`dev/02_controller/SV_controller/tb/script/signals_compare.py`

La figure 3.7 compare les valeurs de la commande à la sortie du régulateur pour le modèle Matlab et SystemVerilog. Les valeurs d'erreurs sont relativement faibles.

Il est intéressant de rappeler que le modèle SystemVerilog est échantillonnée à 40 $[MHz]$. Comme expliqué à la section 3.3, les régulateurs PIs ne sont pas tout à fait modélisés de la même manière qu'avec le modèle Matlab. Si la différence n'est pas significative, c'est que l'action intégrateur des PIs de position et de vitesse est faible. De plus, en boucle ouverte, la dynamique du système a moins d'importance.

Les résultats sont suffisamment bons pour dire que le modèle de référence du régulateur en SystemVerilog correspond au régulateur idéal (Matlab).

4 Testbench

Cette section présente le testbench. Celui-ci a pour but de vérifier le régulateur codé en VHDL. Pour vérifier ce régulateur, un autre régulateur a été modélisé à la section 3.3. Ce second régulateur utilisé comme référence est considéré comme idéal. Des différences entre le régulateur à vérifier et le régulateur de référence sont connues et sont expliquées à la section 3.2.

Ces deux régulateurs ont besoin d'interagir avec un modèle du système à régler pour simuler un comportement semblable au cas réel. Le système à régler a été étudié à la section 2.1 et il a été modélisé en SystemVerilog à la section 2.2.

La structure du testbench est expliquée à la section 4.1. La section 4.2 explique la manière dont les données issues du testbench sont enregistrées et exploitées. Les testcases et les différents paramètres de la simulation sont décrits à la section 4.3. La section 4.4 est consacrée aux scripts permettant d'automatiser le testbench. La section 4.5 présente brièvement les logiciels de simulation utilisés avec quelques remarques et observations. Enfin, les résultats du testbench sont analysés à la section 4.6.

4.1 Structure

La structure générale du testbench est affichée à la figure 4.1

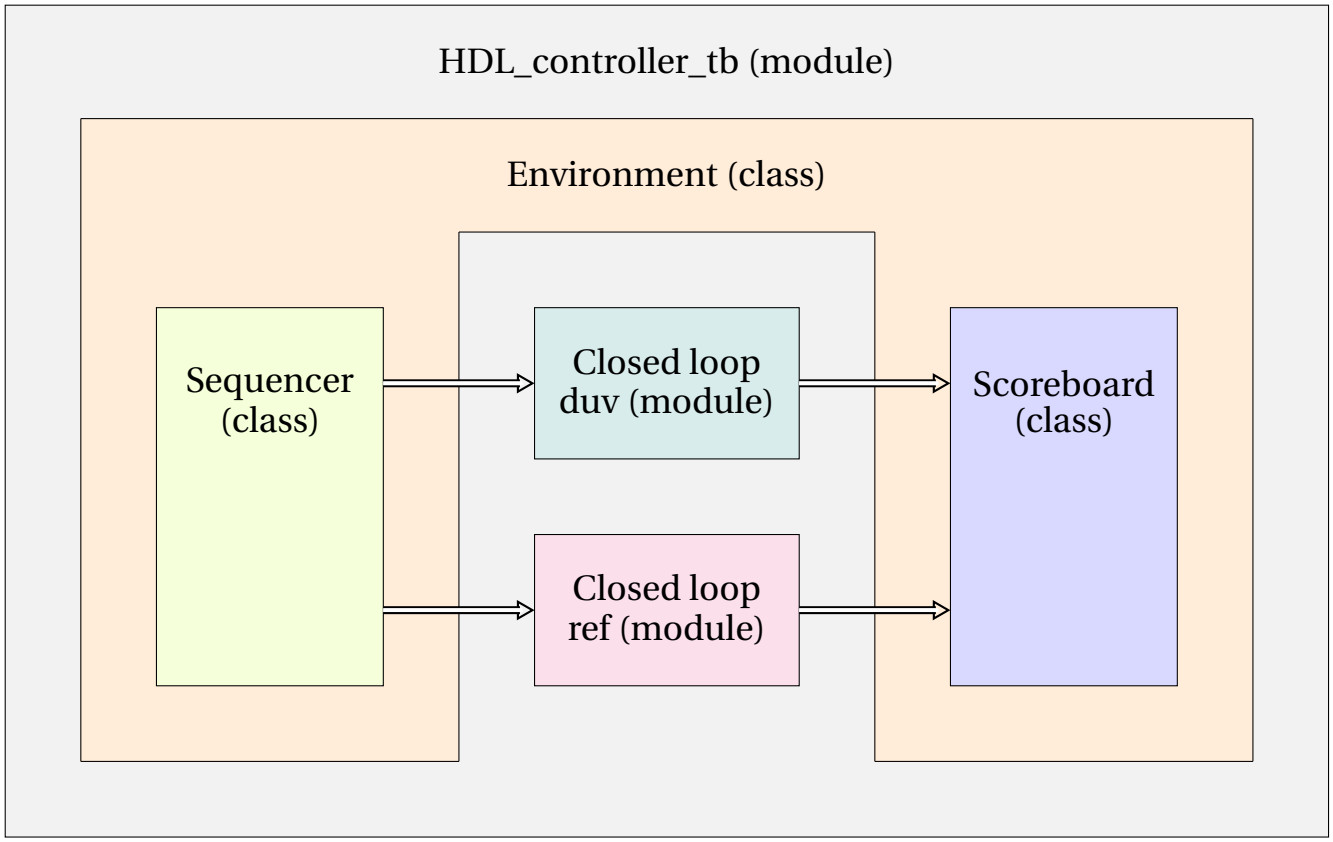


Figure 4.1: Structure du testbench

Dans le « top level » de la simulation, un processus « initial » permet d'exécuter la partie non-synthétisable. Dans un premier temps, un « logger » est utilisé pour sauvegarder tous les paramètres de la simulation dans un fichier. Ensuite, un objet de la classe « Environment » est créé et initialisé.

À côté du processus et de l'environnement, le régulateur et le système à régler sont instanciés dans des modules separés. Un module « Closed loop duv » contient la boucle fermée avec le régulateur à vérifier et une instance du système à régler. L'autre module « Closed loop ref » contient la boucle fermée avec le régulateur de référence et une autre instance du système à régler.

Les deux modules sont connectés à l'environement à l'aide d' « interface ». La génération du signal d'horloge (40 $[MHz]$) est également implémentée dans le « top level ».

`dev/10_testbench/tb/src_tb/HDL_controller_tb.sv`

4.1.1 Module « Closed loop duv »

Dans ce module se trouve la boucle de régulation fermée avec le régulateur à vérifier. Il faut mapper les signaux de commande entre la sortie du régulateur et l'entrée du système à régler. Il faut effectuer une soustraction avec chaque paire de signaux pour obtenir une commande entre -1 et 1.

```
1 assign cmd_u_from_controller = (real'(pwm_modulator[0]) - real'(pwm_modulator[1]));
2 assign cmd_v_from_controller = (real'(pwm_modulator[2]) - real'(pwm_modulator[3]));
3 assign cmd_w_from_controller = (real'(pwm_modulator[4]) - real'(pwm_modulator[5]));
```

Il est intéressant de pouvoir observer les signaux de commande lors de l'analyse post-simulation. Pour cela, ces signaux devront être observés à une fréquence fixe pour être sauvegardés. Si l'échantillonnage de ces signaux est effectuée à la fréquence du PWM, les valeurs seront toujours celles du début du cycle et les signaux sauvegardés ne contiendront aucune informations. Pour contourner ce problème, des modules « moving_average » ont été implémentés pour lisser les signaux de commande. Ceux-ci sont hors de la boucle fermée et ils n'ont donc aucun impact sur le fonctionnement. Ils sont uniquement utilisés pour l'acquisition de données et ils sont paramétrés pour moyenner une période entière de PWM (2480 coups d'horloge à 40 $[MHz]$).

Les autres signaux (feedback de courant, vitesse et position) doivent être mappés avec des fonctions de conversion pour passer de nombres réels à des nombres en virgule fixe. En plus de cela, le module est conçu de manière à pouvoir inverser les phases (entre v et w), la position et la vitesse. La raison est que les conventions liées à la position des phases et au sens du resolver peuvent varier. En cas de problème de convention entre le design à vérifier et les autres composants du testbench, des paramètres génériques permettent d'inverser facilement les interfaces.

`dev/10_testbench/tb/src_tb/closed_loop_duv.sv`

4.1.2 Module « Closed loop ref »

Ce module contient le régulateur de référence. Puisque celui-ci a été modélisé en SystemVerilog, l'interfaçage avec le système à régler ne contient aucune subtilité.

Pour le régulateur à vérifier, les trajectoires sont générées à l'intérieur du composant et il n'est donc pas nécessaire de lui donner les consignes de régulation. Ce n'est pas le cas du régulateur de référence. Celui-ci a besoin que les trajectoires lui soit données en entrée. Le choix a été fait de générer les trajectoires à l'intérieur du module « Closed loop ref ». En réalité, ces trajectoires ne sont pas générées mais simplement lues à partir de fichiers. Ce module contient donc un processus non-synthétisable qui va lire les fichiers des trajectoires. Ces trajectoires sont envoyées au régulateur de référence mais également en sortie du module (en direction du scoreboard).

En effet, il est également intéressant de pouvoir enregistrer les trajectoires avec les autres données pour l'analyse post-simulation. De plus, la durée de simulation dépend de la trajectoire choisie. De ce fait, un signal « running » est également envoyé au scoreboard de manière à indiquer à celui-ci lorsque la trajectoire commence et se termine. Un autre signal « update_cycle » permet de synchroniser l'acquisition de données du scoreboard.

`dev/10_testbench/tb/src_tb/closed_loop_ref.sv`

4.1.3 Classe « Environment »

La classe « Environment » contient toute la partie nécessaire à la vérification. Il s'agit de la « Factory » qui permet d'instancier les autres classes et de démarrer la vérification. Lors de la création des autres classes, l'environement doit mapper les interfaces entre le séquenceur, les deux modules de boucle fermée et le scorboard.

`dev/10_testbench/tb/src_tb/environment.sv`

4.1.4 Classe « Sequencer »

Le séquenceur a pour tâche de générer les stimulis envoyés aux régulateurs. Ces stimulis sont transmis aux régulateurs par une interface nommée « control_signals ».

```
interface itf_control_signals(
        input clk_controller);
    logic   rst;
    logic   update_cycle;
    logic   feedback_pos_enable;
    logic   feedback_spd_enable;
    int     trajectory_selection;
    logic   trajectory_start_scan_in;
    logic   trajectory_start_scan_out;
    real    rotor_position_offset;
```

Parmi ces stimulis se trouvent des paramètres passifs (qui ne varient pas durant la simulation) comme l'activation des boucles de vitesse et de position ou encore la sélection de la trajectoire. Mais cette interface contient également des signaux actifs comme le signal d'horloge, le reset, un signal de synchronisation ou encore le signal de démarrage de la trajectoire. À noter que le signal d'horloge est généré depuis l'extérieur (dans le top level) et que le séquenceur utilise également ce signal d'horloge.

Tous ces signaux de contrôle sont générés en fonction du testcase choisi. La liste des testcases est donnée à la section 4.3.1. Au début de chaque testcase, le signal de reset doit être activé pour initialiser le régulateur VHDL. Après le reset, les signaux liés aux paramètres sont initialisés. Pour démarrer la trajectoire, un des deux signaux "trajectory_start_scan" est activé. Après avoir lancé la simulation, le rôle du séquenceur se limite à générer un signal de synchronisation "update_cycle" à la fréquence du PWM dans une boucle infinie.

`dev/10_testbench/tb/src_tb/sequencer.sv`

4.1.5 Classe « Scoreboard »

Le scoreboard a pour tâche de sauvegarder les données qui seront ensuite analysées. Parmi ces données, il y a la commande triphasée donnée par le régulateur ainsi que les mesures de courant, de vitesse et de position issues du système à régler. Ceci pour les deux boucles fermées et dans le cas de la référence, il y a également les trajectoires qui doivent être enregistrées. Le scoreboard a besoin des signaux « running » et « update_cycle » pour être synchronisé.

```
interface itf_observable_signals;
    real    cmd_u_from_controller;
    real    cmd_v_from_controller;
    real    cmd_w_from_controller;
    real    mot_cur_u_from_plant;
    real    mot_cur_v_from_plant;
    real    mot_cur_w_from_plant;
    real    resolver_pos_from_plant;
    real    resolver_spd_from_plant;
```

```
interface itf_feedforward_signals;
    real    feedforward_position;
    real    feedforward_speed;
    real    feedforward_current;
    logic   update_cycle;
    logic   running;
```

En début de simulation, le scoreboard attend le signal « running ». Lorsque celui-ci s'active, le scoreboard commence à enregistrer les données. Celles-ci sont enregistrées à la fréquence du PWM (donnée par le signal « update_cycle ») jusqu'à ce que le signal « running » soit désactivé. Lorsque c'est le cas, la tâche « run » du scoreboard se termine et cela met fin à la simulation.

`dev/10_testbench/tb/src_tb/scoreboard.sv`

4.2 Exploitation des résultats

Le but du testbench est de pouvoir vérifier si le régulateur VHDL est correct. Une simulation est mise en place avec un autre régulateur (de référence) pour pouvoir les comparer. Cette comparaison se fait par l'observation des signaux sortants des régulateurs (commande) et des grandeurs physiques simulées du système à régler.

Le régulateur est trop complexe pour être évalué avec une simple formule mathématique. Il existe des méthodes de calcul d'erreur permettant d'avoir une estimation globale du résultat mais celles-ci ne sont pas suffisantes pour pouvoir s'y fier aveuglément. Il est impératif que ce testbench offre la possiblité d'observer les signaux en question de manière graphique.

Les signaux sont sauvegardés dans des fichiers depuis le scoreboard. À chaque cycle, une nouvelle valeur est ajoutée à la fin du fichier. Les données sont ajoutées au fur et à mesure pour ne pas perdre les données en cas d'interruption de la simulation.

4.2.1 Format des données

La sauvegarde des données nécessite une réflexion sur le format à utiliser pour les stocker. Le choix du format peut dépendre de la taille mémoire, de la précision ou de la portabilité des données enregistrées.

4.2.1.1 ASCII

Les données peuvent être enregistrées au format ASCII dans un fichier texte. Cette manière de procéder a l'avantage de pouvoir visualiser directement les valeurs en ouvrant le fichier avec un simple éditeur de texte. Elles peuvent facilement être importées dans des programmes externes pour être analysées.

En terme de quantité de mémoire utilisée, le format ASCII n'est pas du tout optimal. Dans le cas des nombres à virgule, la taille dépend de la précision. De plus, même en choisissant une grande précision (avec beaucoup de décimales), il peut quand même y avoir une perte d'informations. Ce format n'est pas adapté aux applications qui exigent l'intégrité des données sauvegardées.

4.2.1.2 HDF5

Le format HDF5 (« Hierarchical Data Format 5 ») est un format adapté aux grandes quantités de données permettant de créer une structure avec une arborescence. Des méta-données permettent d'ajouter des informations aux objets. Les bibliothèques permettant d'utiliser ce format sont sous licence libre (BSD). Les fichiers HDF5 peuvent être traités par de nombreux langages comme C++, Matlab et Python.

Ce format est déjà utilisé au CERN pour certaines mesures du système réel. Il est donc souhaitable que les données du testbench soient sauvegardées dans ce format. Ainsi, les mesures de la simulation pourront directement être comparées aux mesures du système réel.

Cependant, le langage SystemVerilog ne fait pas partie des langages permettant de traiter le format HDF5. Pour remédier à ce problème, deux solutions ont été étudiées.

La première est l'utilisation de DPI (« Direct Programming Interface »). Il s'agit d'une interface permettant d'établir une communication à travers plusieurs langages de programmation. L'utilisation de DPI permet d'exécuter du code C ou C++ dans une simulation SystemVerilog. Les librairies HDF5 existent en C++ et la sauvegarde des données dans ce format pourrait se faire en important des fonctions C++ dans la simulation du testbench. Cependant, la compilation des librairies HDF5 en C++ et l'importation de celles-ci dans le code SystemVerilog ajoutent un niveau de complexité non négligeable (en prenant en compte que la simulation doit être compatible avec plusieurs logiciels de simulation et plusieurs systèmes d'exploitation).

La seconde solution consiste à enregistrer les données dans un autre format depuis la simulation SystemVerilog. Ensuite, les données sauvegardées peuvent être importées dans un autre programme (en C++, Matlab ou Python) pour être converties en format HDF5. Cette manière de procéder nécessite donc un traitement post-simulation. Ceci n'est pas un problème dans le cas présent puisque les données de la simulation doivent de toute façon être traitées après la simulation pour être analysées. Cette étape supplémentaire n'est pas contraignante.

4.2.1.3 Binaire

Enregistrer les données au format binaire permet d'optimiser la quantité de mémoire utilisée et aucune information n'est perdue. Aucune librairie n'est nécessaire et les données peuvent être traitées par n'importe quel language permettant de lire un fichier. La seule contrainte est de devoir gérer soi-même l'écriture et la lecture des données en choisissant une convention.

Les données issues de la simulation n'ont pas une structure fixe. C'est-à-dire que des signaux peuvent être ajoutés ou enlevés et qu'il est nécessaire de savoir à quoi ils correspondent. Ces signaux peuvent également être accompagnés d'informations complémentaires permettant de faciliter leur analyse. Il est donc nécessaire d'avoir une en-tête au fichier (en format ASCII) afin de savoir à quoi correspondent les données binaires. Ce format personnalisé n'est pas détaillé ici car il n'est pas nécessaire d'exploiter ces données sous cette forme. Elles sont converties automatiquement au format HDF5 juste après la simulation.

4.2.2 Analyse des données

La section 4.2.1 explique sous quelle forme sont stockées les données de sortie du testbench. La section présente explique comment ces données peuvent être analysées.

4.2.2.1 Données binaires

La simulation créer un fichier binaire, celui-ci est converti directement après la simulation en format HDF5. Cependant, si pour une raison ou pour une autre, il est nécessaire d'observer les courbes depuis le fichier binaire, le script suivant permet de le faire:

`dev/10_testbench/tb/script/signals_analysis.py`

```
>>> python3 script/signals_analysis.py sim_out/Test_55_base/55_1
```

4.2.2.2 Données HDF5

Les données HDF5 sont générées après la simulation à partir des fichiers binaires. Le script permettant de faire la conversion est donné ci-dessous:

`dev/10_testbench/tb/script/convert_data_to_HDF5.py`

```
>>> python3 script/convert_data_to_HDF5.py sim_out/Test_55_base
```

Pour analyser les données HDF5, un GUI (« Graphical User Interface ») en python a déjà été réalisé au CERN par Jonathan Emery.

`dev/10_testbench/HDF5_data/bwsmcu_bwsidc_gui.py`

```
>>> python3 bwsmcu_bwsidc_gui.py
```

Pour les fichiers qui se trouvent dans le même répertoire que le programme, les signaux peuvent être affichés sous l'onglet "Load_vectors". Pour comparer plusieurs courbes sur le même graphique, il faut cocher la case "keep multiple plots on the graph". Dans cette interface, les signaux de simulation peuvent être comparés avec les mesures réelles qui sont, elles aussi, au format HDF5.

4.2.2.3 Génération de rapport

Pour visualiser tous les résultats de la simulation rapidement, il est intéressant de générer un rapport contenant toutes les courbes et toutes les comparaisons pertinentes.

`dev/10_testbench/tb/script/generate_reports.py`

```
>>> python3 script/generate_reports.py sim_out/Test_55_base
```

Une première page résume tous les paramètres de la simulation afin de savoir de quel test il s'agit et de pouvoir réitérer la simulation. Ensuite, les comparaisons sont faite entre les trajectoires (feedforward), les valeurs simulées du design à vérifier (duv) et celles de la référence (ref). Les signaux internes aux régulateurs sont également comparés entre la référence et le design à vérifier s'ils ont été enregistrés (voir 4.3.3).

4.3 Testcases et paramètres

Pour tester les différentes fonctionnalités du régulateur à vérifier, il est nécessaire de réaliser différentes simulations avec différents paramètres. Les paramètres de la simulation peuvent être séparés en deux catégories. Il y a les paramètres génériques à la simulation, qui sont définis à l'extérieur du testbench et donnés en paramètre au lancement de la simulation. Les autres sont des paramètres fixes, définis dans un fichier ".svh". Ces derniers peuvent être changés entre deux lancements de simulation mais ils sont identiques entre les testcases qui sont simulés en parallèle.

4.3.1 Testcases

Dans l'installation réelle, l'asservissement en position n'est pas toujours active. La boucle de vitesse peut également être désactivée pour ne réguler que le courant. Il peut être intéressant de simuler le régulateur dans les cas où les boucles de régulation de la postion ou de la vitesse sont ouvertes. Il s'agit des testcases 1 à 3 de la table 4.1.

Comme expliqué à la section 2.1, le moteur effectue deux mouvements (« in » et « out ») correspondant à un aller et un retour. Il est donc important de vérifier le régulateur aussi pour le mouvement retour. Il s'agit des testcases 4 à 6.

L'installation réelle est placée dans un environnement pouvant contenir du bruit. À cause de ce bruit, le système de régulation en boucle fermée pourrait subir des perturbations sur les signaux de commande et de feedback. Le bruit est simulé dans les testcases 7 à 9.

Number	Name	Position control	Speed control	Trajectory scan	Noise
1	testcase_pos_scan_in	yes	yes	in	no
2	testcase_spd_scan_in	no	yes	in	no
3	testcase_cur_scan_in	no	no	in	no
4	testcase_pos_scan_out	yes	yes	out	no
5	testcase_spd_scan_out	no	yes	out	no
6	testcase_cur_scan_out	no	no	out	no
7	testcase_pos_scan_out_noise	yes	yes	in	yes
8	testcase_spd_scan_out_noise	no	yes	in	yes
9	testcase_cur_scan_out_noise	no	no	in	yes

Table 4.1: Listes des testcases implémentés dans la classe « Sequencer »

La table 4.1 correspond aux testcases implémentés dans le séquenceur du testbench. En plus des cas cités ci-dessus, le régulateur peut être utilisé pour plusieurs profils de trajectoire différents. Les quatre trajectoires utilisées dans ce travail correspondent à une vitesse de rotation de 33, 55, 110 et 133 $[rad/s]$. Les neuf testcases du séquenceur et les quatre trajectoires disponibles offrent 36 combinaisons possibles.

Le numéro du testcase et la trajectoire sélectionnée doivent être envoyés au testbench comme paramètres génériques lors du lancement de la simulation.

4.3.2 Paramètres

Le testbench est conçu de manière à tester plusieurs fonctionnalités dans plusieurs cas différents. Il est intéressant de paramétriser tout ce qui peut l'être. Les paramètres du testbench sont définis dans le fichier ".svh" ci-dessous. Certains paramètres sont cités dans cette section.

`dev/10_testbench/tb/src_tb/HDL_controller_tb.svh`

Fréquence d'échantillonnage du sytème à régler

Comme expliqué à la section 2.3, le système à régler a été modélisé de manière à pouvoir changer facilement sa fréquence d'échantillonnage.

Paramètres du régulateur

Si les résultats du testbench sont suffisamment proches des mesures réelles, le testbench pourrait être utilisé comme outil de synthèse de la régulation. C'est-à-dire que les paramètres du réglateur pourraient être ajustés dans cette simulation avant d'être testés sur le système réel. Le testbench permet donc de modifier facilement les paramètres des régulateurs PI dans le design à vérifier et le design de référence.

Bruit

Lorsque les testcases 7, 8 ou 9 sont sélectionnés (table 4.1), il faut générer du bruit qui est ajouté aux signaux entre le régulateur et le système à régler (commande et feedback). C'est le séquenceur qui génère ce bruit et il est transmis aux systèmes en boucle fermée par une interface « noise_signals ». Le langage SystemVerilog permet de générer des nombres aléatoires selon plusieurs lois de distribution.

Dans le testbench, le bruit peut être généré avec une distribution uniforme ou avec une distribution normale. Pour générer des nombres aléatoires en SystemVerilog, un exemple est donné ci-dessous. La figure 4.2 montre les résultats sous forme de signaux et d'histogrammes pour visualiser les distributions. Il est ainsi possible de choisir la distribution et l'amplitude du bruit simulé.

```
for(i = 0; i < `LENGTH; i++) begin
    random_signals[0][i] = real'($dist_uniform(seed,-1e4,1e4))/1e4;
    random_signals[1][i] = real'($dist_normal(seed,0,1e4))/1e4;
end
```

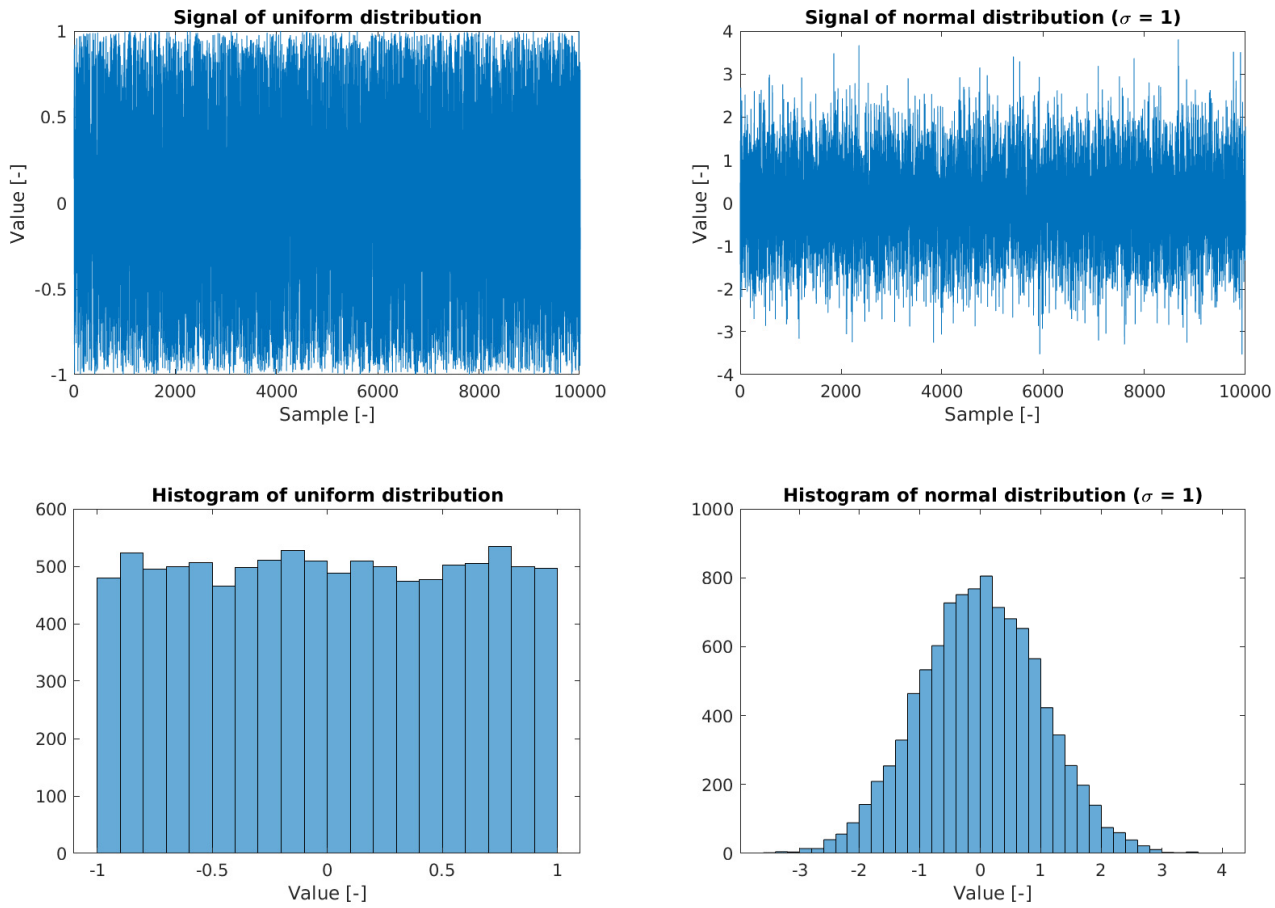


Figure 4.2: Distributions uniformes et normales de nombres aléatoires

4.3.3 Debug

Jusque-là, le testbench a été conçu de manière à vérifier le régulateur en tant que boîte noire. C'est-à-dire que mis à part les entrées/sorties, le code du régulateur n'a pas besoin d'être connu et le testbench est conçu indépendamment de ce code du design à vérifier. Cependant, lorsque les résultats ne sont pas ceux attendus et que la cause du problème n'est pas évidente, il peut être intéressant d'aller observer les signaux à l'intérieur du régulateur. C'est ce qui est fait avec l'option de debug décrite ici.

Lorsque cette option est activée, un objet d'une classe « Debug_signals » est créé à l'intérieur du scoreboard. Cette classe n'utilise pas d'interfaces comme pour les autres signaux car les informations vont être cherchées directement dans les composants sans passer par les entrées/sorties.

Si les signaux du régulateur à vérifier sont observés, il est intéressant de les comparer à une référence. Les signaux internes du régulateur de référence peuvent donc aussi être observés. L'accès direct à des signaux d'un module depuis la classe peut être fait avec un chemin hiérarchique à partir de $*root* comme ci-dessous:

```
1 $root.HDL_controller_tb.closed_loop_ref.controller_ref.command_u
```

Cela fonctionne très bien pour accéder à des modules SystemVerilog mais l'accès au code VHDL ne fonctionne pas toujours. Pour pouvoir observer les signaux à l'intérieur du design à vérifier codé en VHDL, il a fallu créer une copie de ces signaux à l'intérieur d'un module SystemVerilog.

Il existe des fonctions (non-standards qui dépendent du logiciel utilisé) qui permettent de mapper directement des signaux entre différents modules. Ces fonctions peuvent être utilisées pour établir une connexion entre un signal de code VHDL et un signal de code SystemVerilog. Ainsi, dans le « top level » du testbench, un module « Duv_debug_signals » est implémenté (sans entrées/sorties) et il contient une liste de signaux correspondant aux signaux voulant être observés dans le design à vérifier.

Lors de l'exécution du testbench avec l'option debug activée, l'objet de la classe « Debug_signals » va s'initialiser et appeler la fonction permettant de mapper les signaux entre le design à vérifier et le module implémenté en SystemVerilog. Ces fonctions sont les suivantes:

```
1 $init_signal_spy("signal_duv", "signal_systemverilog",0);    // Questa
2 $signal_agent("signal_duv", "signal_systemverilog",0);       // Riviera
```

De cette manière, les signaux du régulateur à vérifier pourront être lus en accédant au module SystemVerilog comme un "miroir" du régulateur à vérifier.

Les signaux internes aux régulateurs sont ensuite enregistrés dans un fichier durant la simulation comme pour les autres signaux. Cependant, le design à vérifier peut être changé et les noms des signaux aussi. De plus certains signaux peuvent être ajoutés ou enlevés selon les besoins. Pour éviter les erreurs liées à la lecture d'un signal inexistant, toute la partie acquisition dans la classe « Duv_debug_signals » est faite de manière conditionnelle à l'aide de macros. C'est-à-dire que pour chaque signal, la compilation du code d'acquisition de ce signal se fait uniquement si le signal en question est défini dans le fichier ".svh". De cette manière, si le design change et qu'une partie des signaux n'existent plus, il suffit de supprimer ou mettre en commentaire les « #define » correspondant à ces signaux pour éviter les erreurs de compilation.

4.4 Scripts et automatisation

Comme expliqué à la section 4.3.1, plusieurs testcases différents sont implémentés dans ce testbench. Lorsqu'il est souhaitable d'observer plusieurs cas différents avec les mêmes paramètres de simulation, les testcases peuvent être simulés en parallèle. Pour lancer plusieurs simulations en parallèle, l'utilisation d'un script permet de simplifier la procédure. De plus, plusieurs traitement post-simulation doivent être effectués pour ce testbench. L'utilisation d'un script pour gérer les testcases, le traitement des données et la génération de rapports automatiques permet de simplifier grandement les tâches de l'utilisateur. Le script permettant de lancer la simulation du testbench est donné ci-dessous:

`dev/10_testbench/tb/HDL_controller_tb.py`

Ce script permet de choisir le logiciel de simulation, le nom de la simulation, la trajectoire, les testcases et le nombre de « thread ». Ce dernier paramètre correspond au nombre de testcases qui peuvent être simulés simultanément. Puisque chaque lancement de simulation nécessite une licence, le nombre de « thread » correspond également au nombre de licences qui seront utilisées.

Lorsque toutes les simulations sont terminées, les scripts permettant de générer le fichier HDF5 (section 4.2.2.2) et le script permettant de générer les rapports (section 4.2.2.3) sont tous les deux exécutés.

4.5 Logiciels de simulation

Pour vérifier le régulateur avec le testbench, il y a deux logiciels de simulation qui peuvent être utilisés. Il s'agit des logiciels QuestaSim de Mentor Graphics et Riviera-Pro de Aldec.

Durant ce travail, la licence permettant d'utiliser le logiciel Riviera-Pro était une version "EDU" qui ralentit volontairement l'exécution de la simulation. Pour cette raison, le logiciel QuestaSim a été privilégié. Par la suite, sur les ordinateurs du CERN, une simulation a montré qu'avec les licences complètes, les durées de simulation entre ces deux logiciels sont équivalentes.

De manière générale, QuestaSim est plus permissif. Le code compilé avec QuestaSim ne compile pas forcément avec Riviera-Pro. Ce second logiciel demande plus de rigueur dans la manière de coder. Au cours de ce travail, plusieurs problèmes sont apparus avec le logiciel Riviera-Pro où celui-ci générait une erreur (lors de l'exécution et non pas de la compilation) sans donner d'informations sur la cause de cette erreur. Ce type de problème a été résolu en revoyant la structure d'une partie du code (en ne sachant toujours pas où était le problème).

Dans la section 4.3.3, il a été expliqué que l'accès à un signal pouvait se faire depuis $\$root$ mais que dans le cas d'un composant VDHL, cela ne fonctionnait pas toujours. En réalité, c'est avec le logiciel Riviera-Pro que cela ne fonctionne pas. En utilisant uniquement QuestaSim, le code permettant d'accéder aux signaux du design à vérifier aurait pu être simplifié.

Les commandes utilisées pour compiler et lancer les simulations sont compatibles avec les deux logiciels. Les scripts de compilation sont donc communs dans les deux cas. Le script principal qui permet de compiler et de lancer la simulation est donné ci-dessous:

`dev/10_testbench/tb/script/sim_HDL_controller_tb.tcl`

Le testbench et les scripts permettant d'automatiser la simulation et l'exploitation des résultats doivent pouvoir fonctionner sur Windows et GNU/Linux.

4.6 Résultats

Le nombre de cas possibles à tester est important et les comparaisons de signaux qui sont générés pour chaque cas sont également nombreuses. La durée des simulations n'étant pas négligeable, le temps restant à la fin de ce travail n'a pas permis d'explorer pleinement toutes les options de ce testbench. Cette section ne montre que quelques cas intéressants qui ont été observés.

4.6.1 Fréquence d'échantillonnage du système à régler

La première observation intéressante est liée à la fréquence d'échantillonnage du système à régler. Comme expliqué à la section 2.3.3, la discrétisation des fonctions de transfert a été réalisée avec la méthode ZOH et de manière à pouvoir changer la fréquence d'échantillonnage selon les besoins et les résultats obtenus. Des simulations Matlab/Simulink avaient été effectuées afin d'observer l'effet de la fréquence d'échantillonnage sur la stabilité de la boucle fermée. Ces résultats (figure 2.46) avaient montré que la simulation en boucle fermée n'était stable qu'avec des fréquences d'échantillonnage entre 640 et 5120 $[Hz]$.

Dans le testbench, avec du code VHDL/SystemVerilog, une fréquence d'échantillonnage élevée du système à régler ne provoque aucun effet indésirable. Plus la fréquence d'échantillonnage est élevée, plus la précision est bonne (se rapprochant du système analogique). Pour les autres simulations, la fréquence d'échantillonnage peut être égale à la fréquence d'horloge du régulateur (40 $[MHz]$) afin de ne perdre aucune information au niveau de la commande. Ci-dessous, le premier rapport montre les résultats pour une fréquence d'échantillonnage de 1.6 $[MHz]$ et le second pour 40 $[MHz]$.

`dev/10_testbench/tb/sim_out/Test_55_Fs_1600kHz/55_1/testcase_report.pdf`
`dev/10_testbench/tb/sim_out/Test_55_base/55_1/testcase_report.pdf`

4.6.2 Différence de convention

À l'observation de certains signaux, il est clair que des conventions d'inversion de phases ou de signes sont différentes entre les deux régulateurs. La figure 4.3 montre une comparaison des signaux de commande à la sortie des régulateurs.

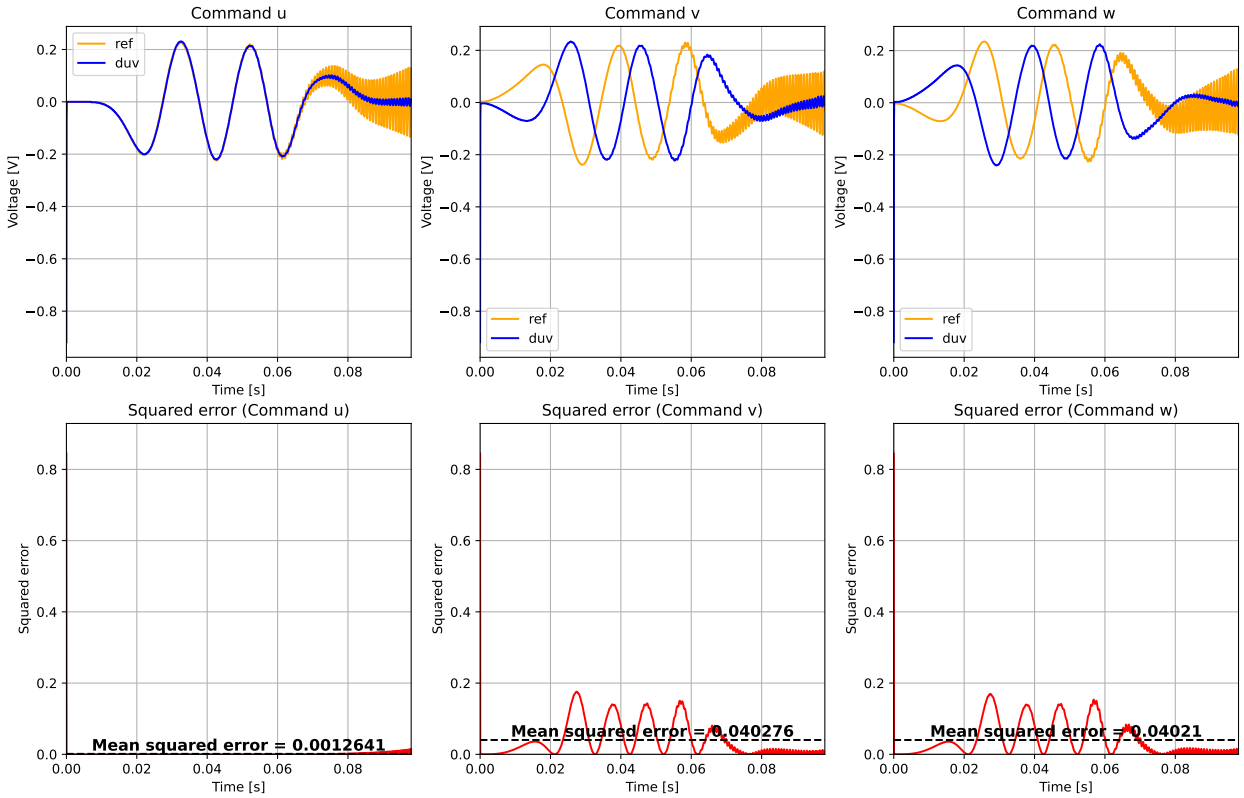


Figure 4.3: Observation de la commande inversée (phases v et w) entre les deux modèles de régulateurs
`dev/10_testbench/tb/sim_out/Test_55_base/55_1/testcase_report.pdf`

Cette différence de convention était déjà connue et l'inversion de ces deux phases était déjà implémentée entre le régulateur à vérifier et le modèle du système à régler. En revanche, ce qui n'était pas prévu est que le courant i_d à l'intérieur du régulateur est également inversé. Puisque ce courant est inversé à l'entrée et à la sortie, il pourrait ne pas avoir d'effet sur le fonctionnement. Sauf que ce courant est aussi utilisé pour le découplage ce qui introduit une différence par rapport au modèle de référence. Cette différence au niveau du découplage est étudiée à la section 4.6.3.

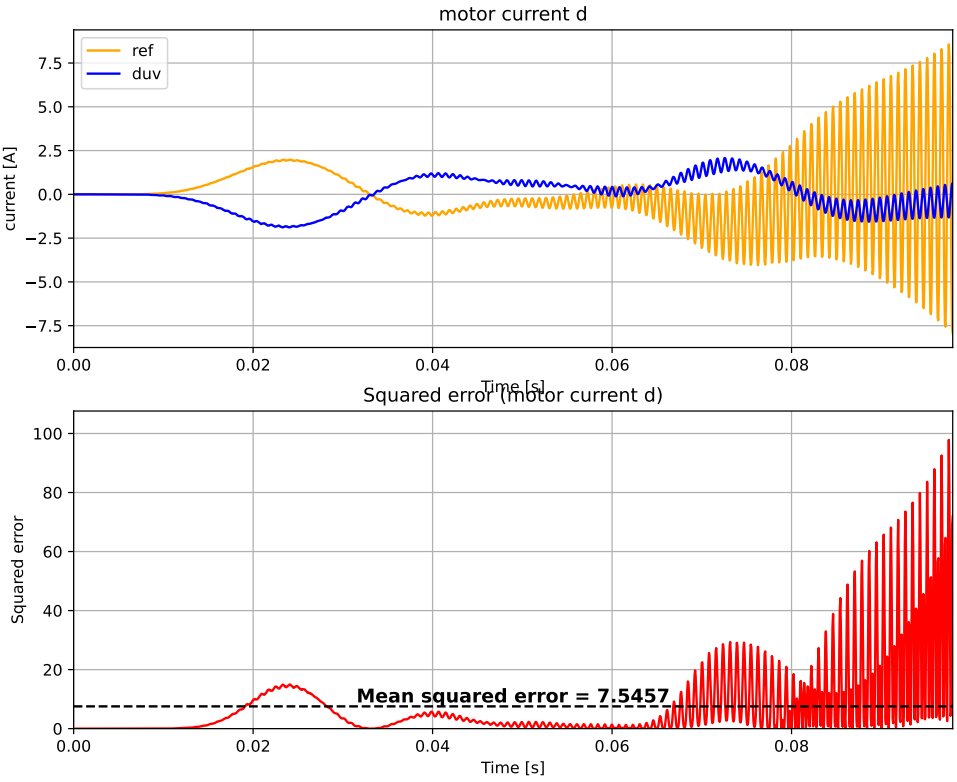


Figure 4.4: Observation de l'inversion du courant i_d dans les régulateurs
`dev/10_testbench/tb/sim_out/Test_55_base/55_1/testcase_report.pdf`

4.6.3 Découplage

La figure 4.4 montre que le courant i_d du régulateur à vérifier est inversé par rapport au régulateur de référence. Cela a une conséquence sur le découplage. Pour observer l'importance du découplage, un test a été réalisé avec le découplage du régulateur de référence qui a été adapté pour être pareil que celui du régulateur à vérifier.

Sur les figures 4.3 et 4.4, le système de référence montrait toujours des signes d'instabilité en fin de trajectoire. Avec le découplage qui a été changé pour être similaire à celui du design à vérifier, les résultats sont bien meilleurs.

`dev/10_testbench/tb/sim_out/Test_55_same_gain_coupling_d_q_change/55_1/testcase_report.pdf`

Les problèmes d'instabilité sont moins importants et surtout, les deux régulateurs ont le même comportement avec des erreurs très faibles sur toutes les grandeurs mesurées. L'exemple de la comparaison de vitesse est montré à la figure 4.5. Avant, l'erreur MSE était de 0.12759 et elle est passée à 0.00048.

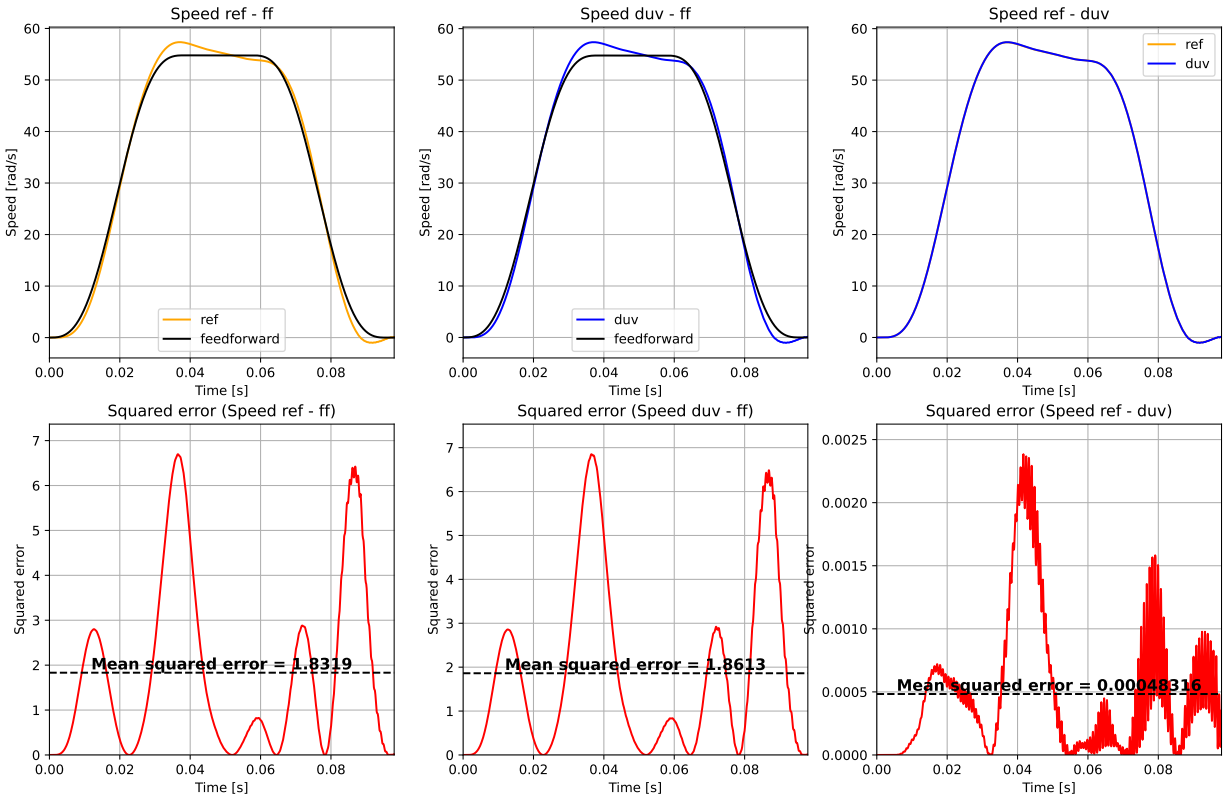


Figure 4.5: Observation de la vitesse avec le même découplage pour les deux régulateurs
`dev/10_testbench/tb/sim_out/Test_55_same_gain_coupling_d_q_change/55_1/testcase_report.pdf`

Ces résultats signifient que lorsque le découplage est calculé de la même manière, la cohérence entre les deux régulateurs est très bonne. Le problème est de sa voir quelle est la bonne manière de calculer le découplage. Dans la section 3.1.3, les équations ont été reprisent des anciennes simulations et semblaient cohérentes avec la théorie.

$$-\omega_m \cdot (K_e + p \cdot i_d \cdot L_s)$$

Cependant, les résultats du testbench montrent que le système a une meilleure stabilité lorsque le découplage vers l'axe q est calculé ainsi:

$$-\omega_m \cdot (K_e - p \cdot i_d \cdot L_s)$$

5 Conclusion

La première partie du travail sur le système à régler a été bien plus longue que prévu. Un modèle a rapidement pu être réalisé en s'inspirant du modèle existant mais par la suite, des erreurs (notamment au niveau du gain) ont été remarquées. Il a été nécessaire d'analyser à nouveau ce système à régler et cela a retardé les autres tâches. La partie sur la discrétisation a également pris un peu de temps jusqu'à obtenir de bons résultats. Finalement, le modèle semble correspondre à peu près au modèle réel même s'il y a certainement encore des choses à améliorer.

La seconde partie sur le régulateur a été plus rapide à étudier. Le modèle était donné et celui-ci est moins complexe que le système à régler (le régulateur est purement numérique). Cependant, après avoir observé les résultats du testbench, il reste une incertitude autour du découplage.

La structure du testbench a pu être réalisée assez rapidement (pour la simulation avec QuestaSim). Cependant, la compatibilité avec le logiciel Riviera a nécessité de refaire plusieurs parties du code. Ne sachant pas d'où venait l'erreur, la phase de debug a pris beaucoup de temps (et la cause n'est toujours pas connue). Malgré les problèmes rencontrés, le testbench a pu finalement être testé avec QuestaSim et Riviera sur Windows et GNU/Linux.

Les scripts avec la génération des graphiques ont également nécessités un temps non négligeable. Finalement, peu de temps a pu être investi à l'analyse des résultats.

5.1 Résultats de simulation

Contrairement à la simulation Matlab/Simulink, une fréquence d'échantillonnage élevée du système à régler permet de s'approcher d'un système analogique parfait. C'est un résultat qui est cohérent avec la théorie (selon la méthode ZOH, plus la fréquence est élevée, plus le retard est faible) mais qui n'était pas observé dans les simulations Matlab/Simulink. Avec ces résultats, il est préférable d'utiliser une fréquence d'échantillonnage élevée pour le système à régler dans le testbench.

Les problèmes de convention avec les inversions de phases et les signes de certains signaux inversés étaient connus. Cependant, il n'était pas prévu que cela ait une influence sur le comportement du régulateur.

Le testbench a mis en évidence une différence d'implémentation entre les deux régulateur liée au découplage. En observant les résultats obtenus pour ces deux régulateurs, il semblerait que c'est le design à vérifier qui est juste. Après correction du régulateur de référence, le testbench montre une très bonne cohérence entre les deux régulateurs. Cependant, la correction qui a été faite sur le régulateur de référence n'est pas bien comprise.

D'une manière générale, le testbench qui a été mis en place a permis de détecter des différences entre la référence et le régulateur à vérifier. Même si ces différences remettent en question le modèle "idéal" du régulateur, la structure générale du testbench fonctionne bien.

5.2 Améliorations possibles

Le système à régler qui a longuement été étudié durant ce travail peut probablement encore être amélioré.

Le modèle de régulateur a besoin d'être revu au niveau du découplage. Il est important de comprendre d'où vient cette différence de calcul.

Les scripts permettant de générer automatiquement les graphiques aurait pu être mieux faits et de manière plus complète. Il aurait été intéressant de pouvoir générer des graphiques montrant les valeurs d'erreur entre les différents testcases. Cela permettrait de voir par exemple, la différence lorsque la régulation de position est active ou non.

Le calcul de trajectoire à l'intérieur de la FPGA était une tâche à réalisée dans ce travail, si le temps le permettait. Avec les contre-temps rencontrés, cette tâche n'a pas pue être réalisée.

5.3 Avis personnel

D'une manière générale, ce projet était très intéressant. Le mixte de la régulation et de la vérification m'a permis d'approfondir mes connaissances dans deux domaines quand même bien différents.

Le stage au CERN que j'ai pu réaliser durant ce travail m'a permis de m'investir encore plus dans ce projet et en même temps, de découvrir un nouvel environnement.

Il n'était pas prévu de passer autant de temps sur le modèle du système à régler. Durant le travail, il était très frustrant de se rendre compte, après des simulations, que le système modélisé comportait des incohérences. À plusieurs reprises, il a fallu reprendre les équations théoriques, refaire les schémas et refaire un nouveau modèle.

Il est dommage d'avoir perdu autant de temps et en voyant les premiers résultats de simulation, je suis un peu déçu de ne pas avoir eu l'occasion d'analyser les résultats de manière plus approfondie.

Malgré tout, j'espère que le travail qui a été fait durant ce projet a permis d'éclaircir certains aspect du système à régler et que cela sera utile pour les futurs projets autour de ce système.

Concernant le testbench, j'ai finalement passé assez peu de temps à travailler sur la structure. Ce sont surtout les problèmes rencontrés avec le logiciel Riviera qui ont nécessité d'investir du temps. Globalement, je suis plutôt satisfait du résultat si je prends en compte les problèmes rencontrés.

Yverdon-les-Bains, le 10 février 2023 Arnaud Yersin

List of Figures

List of Tables

Bibliographie

[1] "Design and validation methodology of the control system for a particle beam size measurement instrument at the CERN laboratory", 2017 American Control Conference (ACC), Seattle, WA, 2017, pp. 4221-4228, doi: 10.23919/ACC.2017.7963604
http://cds.cern.ch/record/2318407

[2] "A low fluctuation control strategy for PMSM direct drive system targeting Particle Beam Instrumentation Application", 2019 IEEE Conference on Control Technology and Applications (CCTA), Hong Kong, China, 2019, pp. 437-443, doi: 10.1109/CCTA.2019.8920688
http://cds.cern.ch/record/2749730

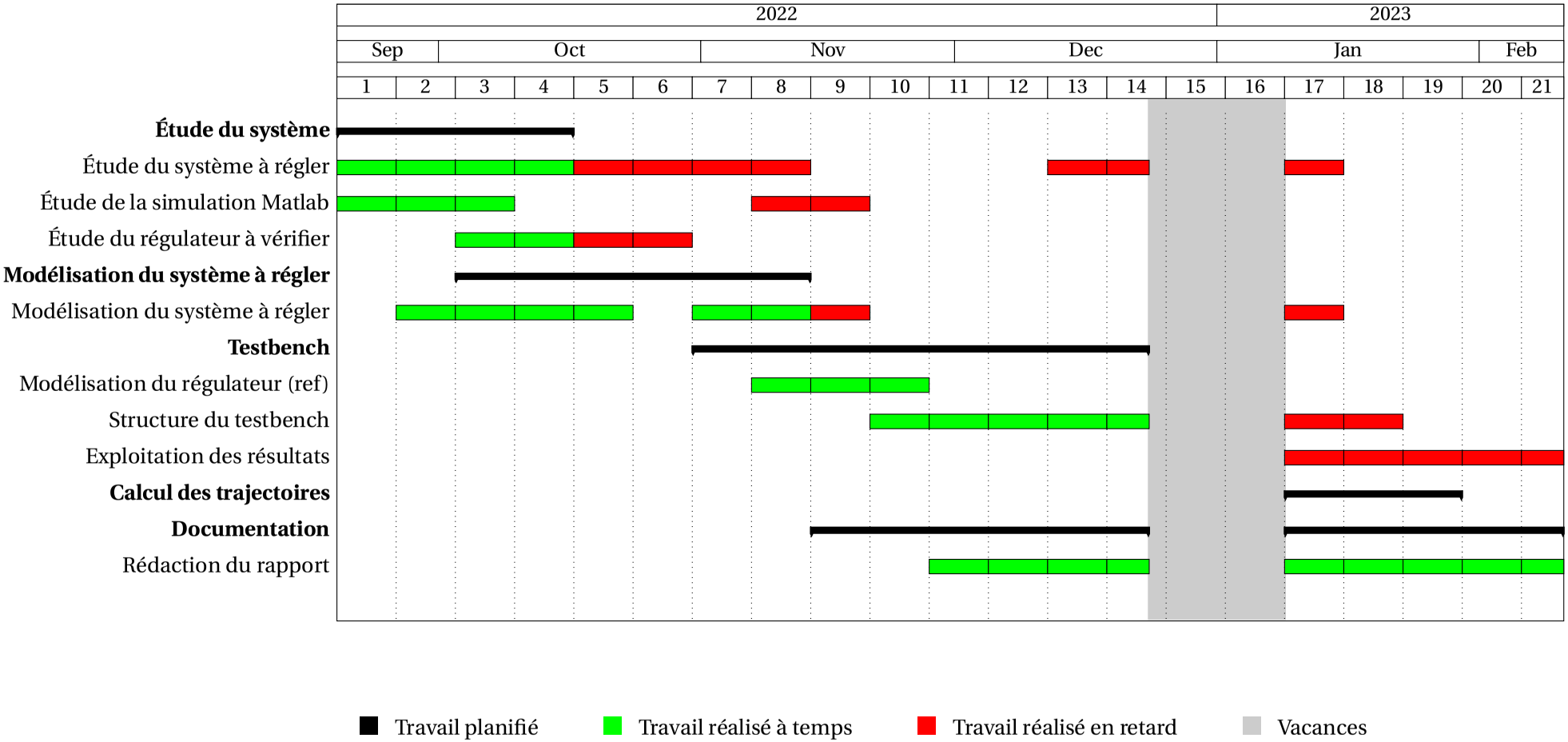
[3] "Motion control design of a PMSM and FPGA implementation for the Beam Wire Scanner at CERN", Master thesis dissertation, Matteo Macchini, Section BE-BI-BL, CERN, Geneva, CH, 2015
http://cds.cern.ch/record/2021089

[4] "BWS Alxion motor test", Jonathan Emery, CERN, 2016
https://indico.cern.ch/event/543918

[5] Support de cours "Machine synchrone", Christophe Besson, HEIG-VD, 2014

[6] Support de cours "Régulation numérique", Michel Etique, HEIG-VD, 2017

A Planning


2022
2023
Sep
Oct
Nov
Dec
Jan
Feb
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
Étude du système
Étude du système à régler
Étude de la simulation Matlab
Étude du régulateur à vérifier
Modélisation du système à régler
Modélisation du système à régler
Testbench
Modélisation du régulateur (ref)
Structure du testbench
Exploitation des résultats
Calcul des trajectoires
Documentation
Rédaction du rapport
Travail planifié
Travail réalisé à temps
Travail réalisé en retard
Vacances

B.1 Discrétisation par la méthode ZOH, exemple 1

Une fonction de transfert analogique est définie comme ci-dessous:

$$G_a(s) = \frac{(s-a-b\cdot j)\cdot(s-a+b\cdot j)}{(s+a-b\cdot j)\cdot(s+a+b\cdot j)}$$

Cette forme de fonction de transfert ne se trouve pas dans les tables. Il faut la calculer avec l'outil "Symbolic Math Toolbox" de Matlab (le code est donné ci-dessous).

```
syms a b s t k h z j
Ga = (s-a-b*1j)*(s-a+b*1j)/(s+a-b*1j)/(s+a+b*1j);
ga = ilaplace(Ga/s,s,t)
>>> (a*exp(-t*(a - b*1i))*2i)/b - (a*exp(-t*(a + b*1i))*2i)/b + 1
```

```
gd = subs(ga,t,(h*k));
Gd = (z-1)/z*ztrans(gd,k,z)
>>> ((z - 1)*(z/(z - 1) + (a*z*2i)/(b*(z - exp(-h*(a - b*1i)))) - (a*z*2i)/(b*(z - exp(-h*(a +
    b*1i))))))/z
```

$$G_d(z) = \frac{z-1}{z}\cdot\left(\frac{z}{z-1} + \frac{2\cdot a\cdot z\cdot j}{b\cdot\left(z-e^{-h\cdot(a-b\cdot j)}\right)} - \frac{2\cdot a\cdot z\cdot j}{b\cdot\left(z-e^{-h\cdot(a+b\cdot j)}\right)}\right)$$

Pour exprimer cette fonction de transfert numérique sous forme de polynômes, il est nécessaire de changer la forme de l'équation pour avoir tous les termes au même dénominateur. Ensuite, en regroupant les termes de chaque degré de z^n, les nombres complexes devraient s'annuler.

$$G_d(z) = 1 + \frac{2\cdot a\cdot(z-1)\cdot j}{b}\cdot\frac{\left(z-e^{-h\cdot(a+b\cdot j)}\right)-\left(z-e^{-h\cdot(a-b\cdot j)}\right)}{\left(z-e^{-h\cdot(a-b\cdot j)}\right)\cdot\left(z-e^{-h\cdot(a+b\cdot j)}\right)}$$

$$G_d(z) = 1 + \frac{2\cdot a\cdot(z-1)\cdot j}{b}\cdot\frac{-e^{-h\cdot a}\cdot\left(cos(h\cdot b)-j\cdot sin(h\cdot b)\right)+e^{-h\cdot a}\cdot\left(cos(h\cdot b)+j\cdot sin(h\cdot b)\right)}{\left(z-e^{-h\cdot a}\cdot\left(cos(h\cdot b)+j\cdot sin(h\cdot b)\right)\right)\cdot\left(z-e^{-h\cdot a}\cdot\left(cos(h\cdot b)-j\cdot sin(h\cdot b)\right)\right)}$$

$$G_d(z) = 1 + \frac{2\cdot a\cdot(z-1)\cdot j}{b}\cdot\frac{e^{-h\cdot a}\cdot 2\cdot sin(h\cdot b)\cdot j}{z^2 - z\cdot e^{-h\cdot a}\cdot 2\cdot cos(h\cdot b)+e^{-2\cdot h\cdot a}}$$

$$G_d(z) = \frac{z^2 - z\cdot e^{-h\cdot a}\cdot 2\cdot cos(h\cdot b)+e^{-2\cdot h\cdot a}}{z^2 - z\cdot e^{-h\cdot a}\cdot 2\cdot cos(h\cdot b)+e^{-2\cdot h\cdot a}} + \frac{-4\cdot\frac{a}{b}\cdot(z-1)\cdot e^{-h\cdot a}\cdot sin(h\cdot b)}{z^2 - z\cdot e^{-h\cdot a}\cdot 2\cdot cos(h\cdot b)+e^{-2\cdot h\cdot a}}$$

$$G_d(z) = \frac{z^2 - z\cdot 2\cdot e^{-h\cdot a}\cdot\left(cos(h\cdot b)+2\cdot\frac{a}{b}\cdot sin(h\cdot b)\right)+\left(e^{-2\cdot h\cdot a}+4\cdot\frac{a}{b}\cdot e^{-h\cdot a}\cdot sin(h\cdot b)\right)}{z^2 - z\cdot e^{-h\cdot a}\cdot 2\cdot cos(h\cdot b)+e^{-2\cdot h\cdot a}}$$

Sous forme de polynômes, la fonction de transfert s'écrit:

$$G_d(z) = \frac{b_0\cdot z^2 + b_1\cdot z + b_2}{a_0\cdot z^2 + a_1\cdot z + a_2}$$

$b_0 = 1$
$b_1 = -2\cdot e^{-h\cdot a}\cdot\left(cos(h\cdot b)+2\cdot\frac{a}{b}\cdot sin(h\cdot b)\right)$
$b_2 = e^{-2\cdot h\cdot a}+4\cdot\frac{a}{b}\cdot e^{-h\cdot a}\cdot sin(h\cdot b)$
$a_0 = 1$
$a_1 = -e^{-h\cdot a}\cdot 2\cdot cos(h\cdot b)$
$a_2 = e^{-2\cdot h\cdot a}$

B.2 Discrétisation par la méthode ZOH, exemple 2

Une fonction de transfert analogique est définie comme ci-dessous:

$$G_a(s) = \frac{(s - z_1)}{(s - a - b \cdot j) \cdot (s - a + b \cdot j)}$$

Cette forme de fonction de transfert ne se trouve pas dans les tables. Il faut la calculer avec l'outil "Symbolic Math Toolbox" de Matlab (le code est donné ci-dessous).

```
syms a b z1 s t k h z j
Ga = (s-z1)/(s-a-b*1j)/(s-a+b*1j);
ga = ilaplace(Ga/s,s,t)
>>> - z1/((a - b*1i)*(a + b*1i)) + (exp(t*(a - b*1i))*(a*1i + b - z1*1i))/(2*b*(a - b*1i)) + (
    exp(t*(a + b*1i))*(b - a*1i + z1*1i))/(2*b*(a + b*1i))
```

```
gd = subs(ga,t,(h*k));
Gd = (z-1)/z*ztrans(gd,k,z)
>>> ((z - 1)*((z*(a*1i + b - z1*1i))/(2*b*(a - b*1i)*(z - exp(h*(a - b*1i)))) + (z*(b - a*1i +
     z1*1i))/(2*b*(a + b*1i)*(z - exp(h*(a + b*1i)))) - (z*z1)/((a - b*1i)*(a + b*1i)*(z - 1))
    ))/z
```

$$G_d(z) = \frac{z-1}{z} \cdot \left(\frac{z \cdot (b + j \cdot (a - z_1))}{2 \cdot b \cdot (a - b \cdot j) \cdot \left(z - e^{h \cdot (a - b \cdot j)}\right)} + \frac{z \cdot (b - j \cdot (a - z_1))}{2 \cdot b \cdot (a + b \cdot j) \cdot \left(z - e^{h \cdot (a + b \cdot j)}\right)} - \frac{z \cdot z_1}{(a - b \cdot j) \cdot (a + b \cdot j) \cdot (z - 1)} \right)$$

Pour exprimer cette fonction de transfert numérique sous forme de polynômes, il est nécessaire de changer la forme de l'équation pour avoir tous les termes au même dénominateur. Ensuite, en regroupant les termes de chaque degré de z^n, les nombres complexes devraient s'annuler.

$$G_d(z) = \frac{(z-1) \cdot (b + j \cdot (a - z_1))}{2 \cdot b \cdot (a - b \cdot j) \cdot \left(z - e^{h \cdot (a - b \cdot j)}\right)} + \frac{(z-1) \cdot (b - j \cdot (a - z_1))}{2 \cdot b \cdot (a + b \cdot j) \cdot \left(z - e^{h \cdot (a + b \cdot j)}\right)} - \frac{z_1}{(a - b \cdot j) \cdot (a + b \cdot j)}$$

$$\begin{aligned} G_d(z) = & \frac{(z-1) \cdot (b + j \cdot (a - z_1)) \cdot (a + b \cdot j) \cdot \left(z - e^{h \cdot (a + b \cdot j)}\right)}{2 \cdot b \cdot (a - b \cdot j) \cdot (a + b \cdot j) \cdot \left(z - e^{h \cdot (a - b \cdot j)}\right) \cdot \left(z - e^{h \cdot (a + b \cdot j)}\right)} \\ & + \frac{(z-1) \cdot (b - j \cdot (a - z_1)) \cdot (a - b \cdot j) \cdot \left(z - e^{h \cdot (a - b \cdot j)}\right)}{2 \cdot b \cdot (a - b \cdot j) \cdot (a + b \cdot j) \cdot \left(z - e^{h \cdot (a - b \cdot j)}\right) \cdot \left(z - e^{h \cdot (a + b \cdot j)}\right)} \\ & - \frac{z_1 \cdot 2 \cdot b \cdot \left(z - e^{h \cdot (a - b \cdot j)}\right) \cdot \left(z - e^{h \cdot (a + b \cdot j)}\right)}{2 \cdot b \cdot (a - b \cdot j) \cdot (a + b \cdot j) \cdot \left(z - e^{h \cdot (a - b \cdot j)}\right) \cdot \left(z - e^{h \cdot (a + b \cdot j)}\right)} \end{aligned}$$

Après résolution des calculs, la fonction de transfert peut s'écrire sous forme de polynômes:

$$G_d(z) = \frac{b_0 \cdot z^2 + b_1 \cdot z + b_2}{a_0 \cdot z^2 + a_1 \cdot z + a_2}$$

$b_0 = 0$
$b_1 = \frac{-b \cdot z1 + e^{h \cdot a} \cdot (b \cdot z1 \cdot cos(h \cdot b) + (a^2 + b^2 - a \cdot z1) \cdot sin(h \cdot b))}{b \cdot (a^2 + b^2)}$
$b_2 = \frac{-b \cdot z1 \cdot e^{2 \cdot h \cdot a} - e^{h \cdot a} \cdot ((a^2 + b^2 - a \cdot z1) \cdot sin(h \cdot b) - b \cdot z1 \cdot cos(h \cdot b))}{b \cdot (a^2 + b^2)}$
$a_0 = 1$
$a_1 = -e^{h \cdot a} \cdot 2 \cdot cos(h \cdot b)$
$a_2 = e^{2 \cdot h \cdot a}$

B.3 Discrétisation par la méthode ZOH, exemple 3

Une fonction de transfert analogique est définie comme ci-dessous:

$$G_a(s) = \frac{1}{(s-a-b\cdot j)\cdot(s-a+b\cdot j)}$$

Cette forme de fonction de transfert ne se trouve pas dans les tables. Il faut la calculer avec l'outil "Symbolic Math Toolbox" de Matlab (le code est donné ci-dessous).

```
syms a b s t k h z j
Ga = 1/(s-a-b*1j)/(s-a+b*1j);
ga = ilaplace(Ga/s,s,t)
>>> 1/((a - b*1i)*(a + b*1i)) + (exp(t*(a - b*1i))*1i)/(2*b*(a - b*1i)) - (exp(t*(a + b*1i))*1
    i)/(2*b*(a + b*1i))
```

```
gd = subs(ga,t,(h*k));
Gd = (z-1)/z*ztrans(gd,k,z)
>>> ((z - 1)*((z*1i)/(2*b*(a - b*1i)*(z - exp(h*(a - b*1i)))) - (z*1i)/(2*b*(a + b*1i)*(z -
    exp(h*(a + b*1i)))) + z/((a - b*1i)*(a + b*1i)*(z - 1))))/z
```

$$G_d(z) = \frac{z-1}{z}\cdot\left(\frac{z\cdot j}{2\cdot b\cdot(a-b\cdot j)\cdot\left(z-e^{h\cdot(a-b\cdot j)}\right)} - \frac{z\cdot j}{2\cdot b\cdot(a+b\cdot j)\cdot\left(z-e^{h\cdot(a+b\cdot j)}\right)} + \frac{z}{(a-b\cdot j)\cdot(a+b\cdot j)\cdot(z-1)}\right)$$

Pour exprimer cette fonction de transfert numérique sous forme de polynômes, il est nécessaire de changer la forme de l'équation pour avoir tous les termes au même dénominateur. Ensuite, en regroupant les termes de chaque degré de z^n, les nombres complexes devraient s'annuler.

$$G_d(z) = \frac{(z-1)\cdot j}{2\cdot b\cdot(a-b\cdot j)\cdot\left(z-e^{h\cdot(a-b\cdot j)}\right)} - \frac{(z-1)\cdot j}{2\cdot b\cdot(a+b\cdot j)\cdot\left(z-e^{h\cdot(a+b\cdot j)}\right)} + \frac{1}{(a-b\cdot j)\cdot(a+b\cdot j)}$$

$$G_d(z) = \frac{(z-1)\cdot j\cdot(a+b\cdot j)\cdot\left(z-e^{h\cdot(a+b\cdot j)}\right) - (z-1)\cdot j\cdot(a-b\cdot j)\cdot\left(z-e^{h\cdot(a-b\cdot j)}\right) + 2\cdot b\cdot\left(z-e^{h\cdot(a-b\cdot j)}\right)\cdot\left(z-e^{h\cdot(a+b\cdot j)}\right)}{2\cdot b\cdot(a-b\cdot j)\cdot(a+b\cdot j)\cdot\left(z-e^{h\cdot(a-b\cdot j)}\right)\cdot\left(z-e^{h\cdot(a+b\cdot j)}\right)}$$

Après résolution des calculs, la fonction de transfert peut s'écrire sous forme de polynômes:

$$G_d(z) = \frac{b_0\cdot z^2 + b_1\cdot z + b_2}{a_0\cdot z^2 + a_1\cdot z + a_2}$$

$b_0 = 0$
$b_1 = \frac{b+a\cdot e^{h\cdot a}\cdot sin(h\cdot b)-b\cdot e^{h\cdot a}\cdot cos(h\cdot b)}{b\cdot(a^2+b^2)}$
$b_2 = \frac{b\cdot e^{2\cdot h\cdot a}-b\cdot e^{h\cdot a}\cdot cos(h\cdot b)-a\cdot e^{h\cdot a}\cdot sin(h\cdot b)}{b\cdot(a^2+b^2)}$
$a_0 = 1$
$a_1 = -e^{h\cdot a}\cdot 2\cdot cos(h\cdot b)$
$a_2 = e^{2\cdot h\cdot a}$

B.4 Discrétisation par la méthode ZOH, exemple 4

Une fonction de transfert analogique est définie comme ci-dessous:

$$G_a(s) = \frac{s}{(s-a-b\cdot j)\cdot(s-a+b\cdot j)}$$

Cette forme de fonction de transfert ne se trouve pas dans les tables. Il faut la calculer avec l'outil "Symbolic Math Toolbox" de Matlab (le code est donné ci-dessous).

```
syms a b s t k h z j
Ga = s/(s-a-b*1j)/(s-a+b*1j);
ga = ilaplace(Ga/s,s,t)
>>> (exp(t*(a - b*1i))*1i)/(2*b) - (exp(t*(a + b*1i))*1i)/(2*b)
```

```
gd = subs(ga,t,(h*k));
Gd = (z-1)/z*ztrans(gd,k,z)
>>> (((z*1i)/(2*b*(z - exp(h*(a - b*1i)))) - (z*1i)/(2*b*(z - exp(h*(a + b*1i)))))*(z - 1))/z
```

$$G_d(z) = \frac{z-1}{z}\cdot\left(\frac{z\cdot j}{2\cdot b\cdot\left(z-e^{h\cdot(a-b\cdot j)}\right)} - \frac{z\cdot j}{2\cdot b\cdot\left(z-e^{h\cdot(a+b\cdot j)}\right)}\right)$$

Pour exprimer cette fonction de transfert numérique sous forme de polynômes, il est nécessaire de changer la forme de l'équation pour avoir tous les termes au même dénominateur. Ensuite, en regroupant les termes de chaque degré de z^n, les nombres complexes devraient s'annuler.

$$G_d(z) = \frac{(z-1)\cdot j}{2\cdot b\cdot\left(z-e^{h\cdot(a-b\cdot j)}\right)} - \frac{(z-1)\cdot j}{2\cdot b\cdot\left(z-e^{h\cdot(a+b\cdot j)}\right)}$$

$$G_d(z) = \frac{(z-1)\cdot j\cdot\left(e^{h\cdot(a-b\cdot j)}-e^{h\cdot(a+b\cdot j)}\right)}{2\cdot b\cdot\left(z-e^{h\cdot(a-b\cdot j)}\right)\cdot\left(z-e^{h\cdot(a+b\cdot j)}\right)}$$

$$G_d(z) = \frac{1}{2\cdot b}\cdot\frac{(z-1)\cdot\left(e^{h\cdot a}\cdot 2\cdot sin(h\cdot b)\right)}{z^2 - z\cdot e^{h\cdot a}\cdot 2\cdot cos(h\cdot h) + e^{2\cdot h\cdot a}}$$

Après résolution des calculs, la fonction de transfert peut s'écrire sous forme de polynômes:

$$G_d(z) = \frac{b_0\cdot z^2 + b_1\cdot z + b_2}{a_0\cdot z^2 + a_1\cdot z + a_2}$$

$b_0 = 0$
$b_1 = \frac{e^{h\cdot a}\cdot sin(h\cdot b)}{b}$
$b_2 = \frac{-e^{h\cdot a}\cdot sin(h\cdot b)}{b}$
$a_0 = 1$
$a_1 = -e^{h\cdot a}\cdot 2\cdot cos(h\cdot b)$
$a_2 = e^{2\cdot h\cdot a}$

B.5 Discrétisation par la méthode ZOH, exemple 5

Une fonction de transfert analogique est définie comme ci-dessous:

$$G_a(s) = \frac{1}{s - p_1}$$

Cette forme de fonction de transfert peut se trouver facilement dans les tables. La fonction de transfert discrète est donnée ci-dessous.

$$\frac{G_a(s)}{s} = \frac{1}{a} \cdot \frac{a}{s \cdot (s + a)} \qquad \longrightarrow \qquad G_d(z) = \frac{1}{a} \cdot \frac{(z-1)}{z} \cdot \frac{(1 - e^{-a \cdot h}) \cdot z}{(z-1) \cdot \left(z - e^{-a \cdot h}\right)}$$

$$G_d(z) = \frac{1}{a} \cdot \frac{(1 - e^{-a \cdot h})}{\left(z - e^{-a \cdot h}\right)}$$

$$a = -p_1$$

$$G_d(z) = \frac{1}{-p_1} \cdot \frac{(1 - e^{p_1 \cdot h})}{\left(z - e^{p_1 \cdot h}\right)}$$

Elle peut s'écrire sous forme de polynômes:

$$G_d(z) = \frac{b_0 \cdot z + b_1}{a_0 \cdot z + a_1}$$

$b_0 = 0$
$b_1 = \frac{e^{h \cdot p_1} - 1}{p_1}$
$a_0 = 1$
$a_1 = -e^{h \cdot p_1}$

B.6 Discrétisation par la méthode ZOH, exemple 6

Une fonction de transfert analogique est définie comme ci-dessous:

$$G_a(s) = \frac{s}{s - p_1}$$

Cette forme de fonction de transfert peut se trouver facilement dans les tables. La fonction de transfert discrète est donnée ci-dessous.

$$\frac{G_a(s)}{s} = \frac{1}{(s+a)} \qquad \longrightarrow \qquad G_d(z) = \frac{(z-1)}{z} \cdot \frac{z}{\left(z - e^{-a \cdot h}\right)}$$

$$G_d(z) = \frac{z-1}{z - e^{-a \cdot h}}$$

$$a = -p_1$$

$$G_d(z) = \frac{z-1}{z - e^{p_1 \cdot h}}$$

Elle peut s'écrire sous forme de polynômes:

$$G_d(z) = \frac{b_0 \cdot z + b_1}{a_0 \cdot z + a_1}$$

$b_0 = 1$
$b_1 = -1$
$a_0 = 1$
$a_1 = -e^{h \cdot p_1}$