

Rapport

Bureau d'étude - Trottinette électrique

Bureau d'étude Trottinette à l'INSA de Toulouse en 4ème année De Bastian KROHG et Nicolas SIARD Automatique Electronique - Systèmes Embarqués 2022-2023



Sommaire

Introduction

Objectifs et compétences développées

1 - Asservissement de couple

1.1 - Premier approche et prise en main du système

1.2 - Asservissement dans le domaine continu

1.3 - Passage au discret - Asservissement dans le domaine discret

1.4 - Asservissement avec correcteur numérique par µcontrôleur

1.4.1 - De la fonction de transfert en z vers l'équation récurrente du correcteur

1.4.2 - Étapes de la conception du correcteur numérique

1.4.3 - Vérifier et débugger le code - mauvais calcul des macros

1.4.4 - Vérification en simulation - comparaison Simulink/Keil

1.4.5 - Implémentation et essais du correcteur

2 - Régulation de vitesse

3 - Autres évolutions du projet

Conclusion

Introduction

La trottinette Electrique est depuis quelques années très tendances et les ventes ne cessent d'augmenter. Les trottinettes sont aujourd'hui très permormantes et fonctionnent avec un moteur brushless, un moteur directement mis dans la roue. Cependant, cela n'a pas toujours été le cas. En effet, la trottinette apparaît dans les années 2000. On peut alors en acheter une pour 30 euros, elles étaient équipées de moteur à courant continu, d'une batterie 12V, d'un bouton marche arret et de frein à tambour. Quelques années plus tard, le département génie électrique informatique acquiert une quinzaine de trottinettes et propose de faire des améliorations. En 2004, il propose de concevoir un variateur de vitesse à développer en BE. La régulation de couple était faite entièrement en analogique, la boucle de vitesse avec un µC PIC18F458, 8 bits 12MHz. De plus, le système permettait la récupération d'énergie au freinage pour une plus grande autonomie. A cette époque aucune trottinette ne proposait cela. Puis en 2012, le STM32F103 entre au GEI. Sa puissance permet d'intégrer les deux boucles entièrement en numérique, c'est sur cette version de la trottinette que nous avons travaillé cette année lors de ce BE. Au travers de plusieurs scéances nous allons étudier ce système et proposer une implémentation de correcteur pour contrôler la trottinette. Ce Be nous permet d'acquérir de nouvelles compétences et d'apprendre plus de choses sur la conception et l'analyse des systèmes électroniques.

Objectifs et compétences développées

L'objectif de ce BE Trottinette est d'arriver à comprendre et à analyser un système électronique complexe dans le but d'implémenter une commande en couple et en vitesse d'une trottinette électrique.

Pour cela nous allons devoir mettre en application les connaissances que nous avons en architecture électronique mais aussi celles que nous avons en automatique, notamment pour la stabilité du système et la conception d'un correcteur. Nous allons également devoir apprendre à commander et comprendre un élément de puissance qui sera pour nous le hacheur et le moteur électrique, dans le but d'implémenter un asservissement. Enfin il faudra aller d'une vision continue à une vision discrète et implémenter le correcteur dans un microcontrôleur grâce à la transformée bilinéaire.

Ce BE comprend beaucoup de concepts qu'il nous faut comprendre pour pouvoir réaliser le meilleur asservissement possible. Pour cela nous avons accès à toutes les documentation techniques et à tous les schémas électroniques du système.

1 - Asservissement de couple

1.1 - Premier approche et prise en main du système

Tout d'abord il faut avoir une vue d'ensemble du système et comprendre de quels éléments il est composé et comment ces derniers interagissent entre eux. Pour cela nous avons le dossier technique à notre disposition. Nous pouvons commencer par analyser le diagramme en bloc qui indique comment les blocs de la commande électronique sont reliés les uns aux autres. Cela nous permet d'identifier les différents blocs que nous aurons dans le schéma bloc du système.

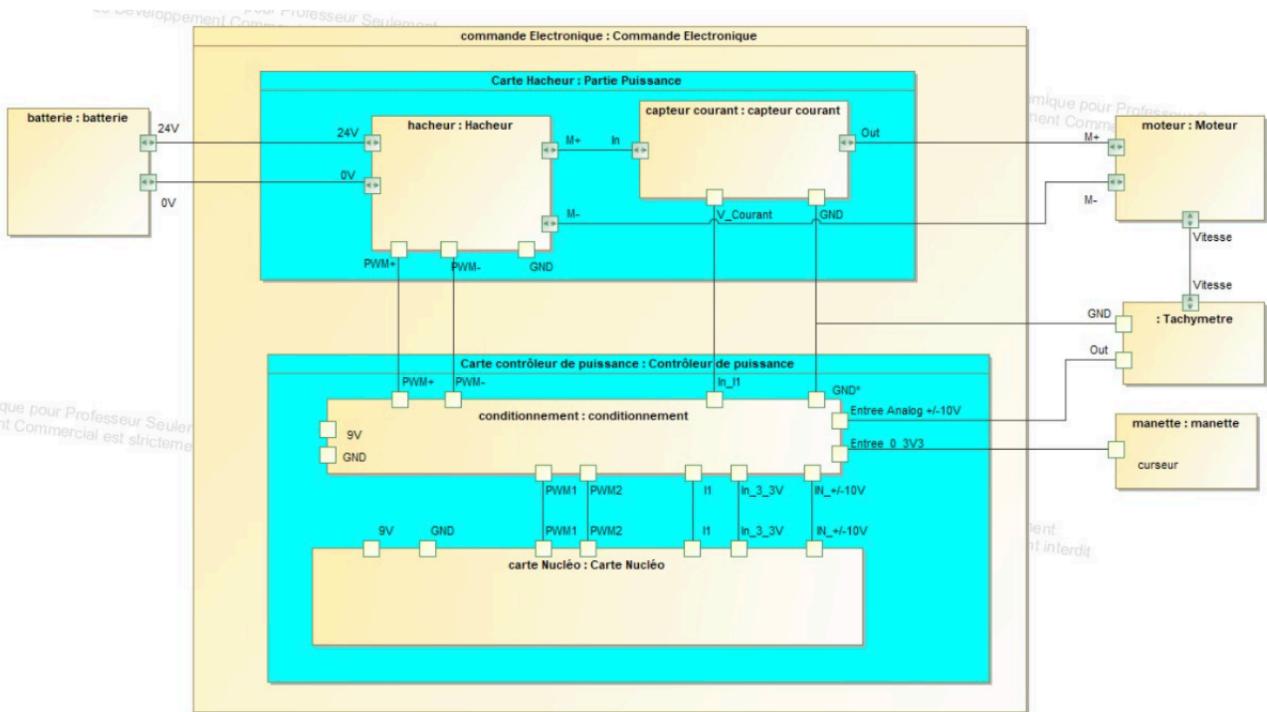


Figure 1.1.1 - Diagramme en bloc de la commande électronique

Il a donc les différents blocs:

Le hacheur : H(p)

Le moteur M(p)

Le capteur de courant : N(p)

Le conditionnement et les filtres : F(p)

Ensuite, grâce à la documentation technique nous pouvons trouver quelles sont les grandeurs physiques que nous devons prendre en compte, quels sont les points de repos et quelles sont les amplitudes de chacun des signaux. Nous arrivons au schéma bloc suivant:

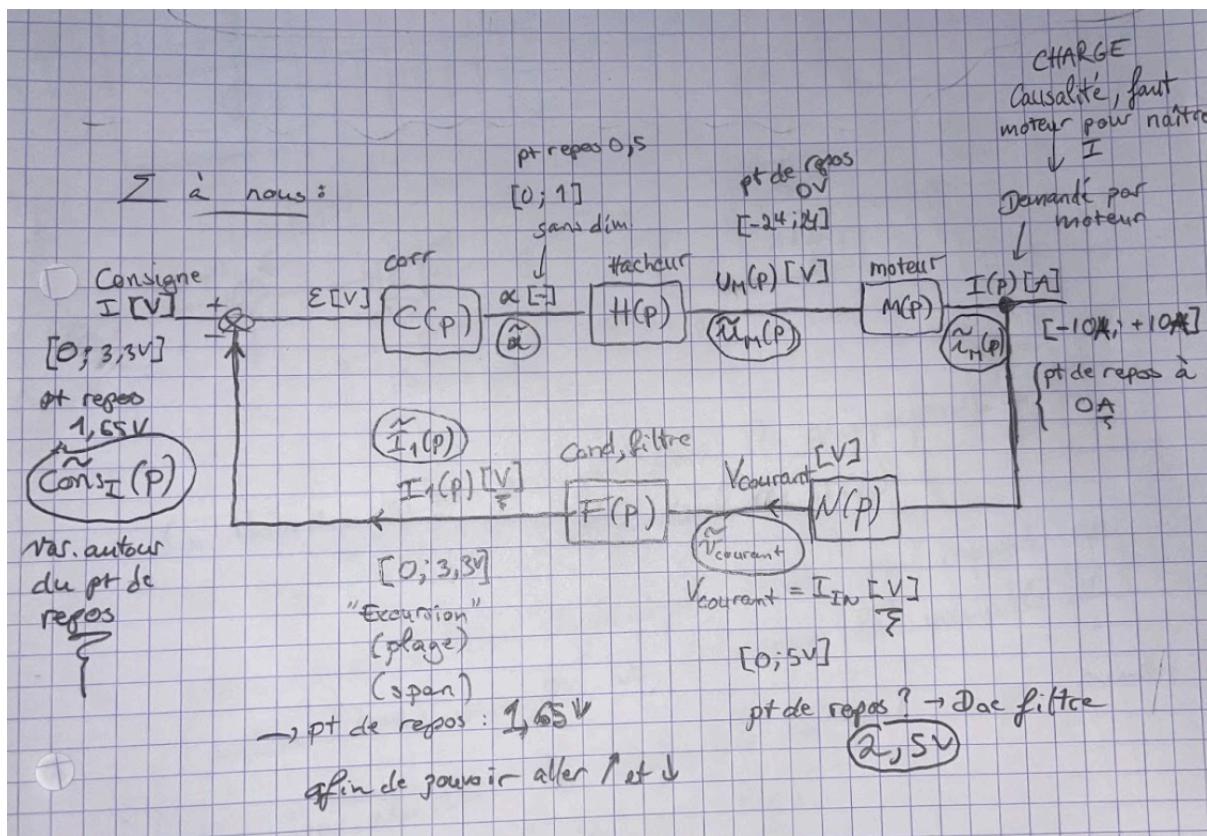


Figure 1.1.2 - Schéma bloc qui simplifie et fusionne entièrement le système de régulation de couple, avec unités physiques ainsi que les plages de valeurs admises.

On notera $C(p)$, le microcontrôleur dans lequel on implémentera notre correcteur discret.

Il suffit maintenant d'exprimer les fonctions de transfert des différents blocs.

- Le hacheur peut juste se traduire par un gain K_{hacheur} qui vaut 48 d'après la fiche technique. Cet élément permet à partir d'un rapport cyclique alpha, de fournir une tension continue. Cependant, si on analyse le spectre fréquentiel du signal généré entre +/- 24 Volts, la tension de la batterie, on peut voir la fondamental mais aussi des harmoniques (de 20kHz). Or, on aimeraient garder uniquement la composante continue à 0Hz. Il faudrait donc rajouter un filtre passe bas pour éliminer les harmoniques. Cela n'est pas nécessaire dans notre cas car on attaque un moteur à courant continu en sortie qui est semblable à un filtre LR, un filtre passe bas avec une fréquence de coupure à 80Hz, ce qui permet de supprimer les harmoniques. La tension de sortie du hacheur U_M est donc une tension continue avec une valeur de tension moyenne induite par alpha.
- La fonction de transfert du moteur se traduit par un fonction de transfert du premier ordre :

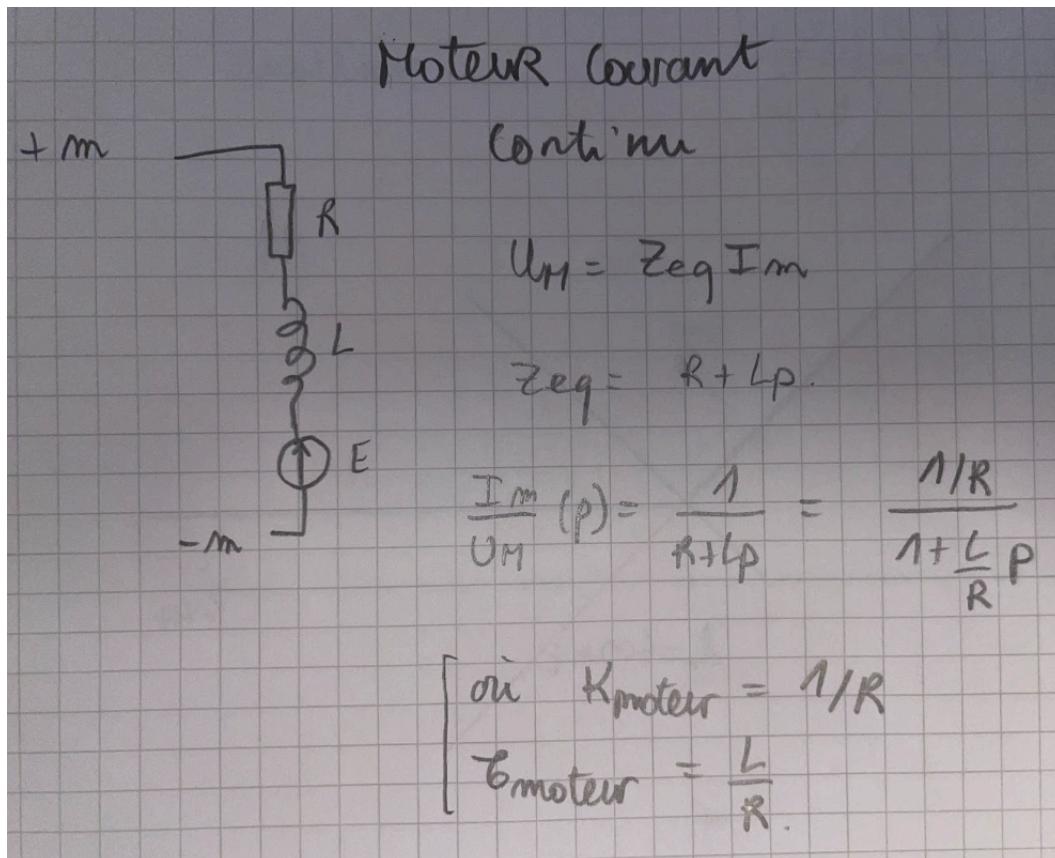
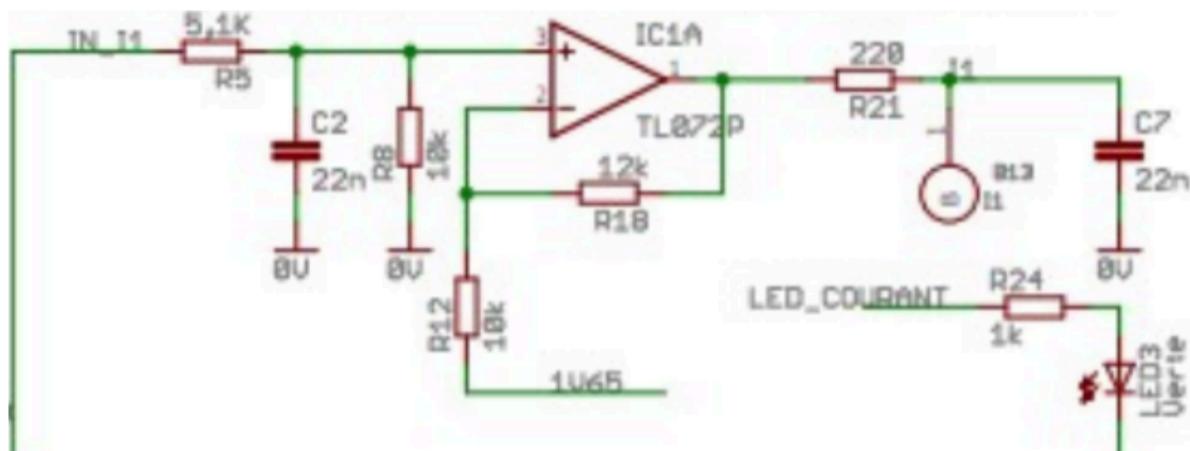


Figure 1.1.3 - Schéma

simplifié et calcul des caractéristiques du moteur

On ne prend pas en compte la perturbation $E(p)$ car on $E(p)$ à une dynamique lente par rapport au moteur. Donc on considérera que les variables sont indépendantes. Pour étudier la stabilité on peut ne pas prendre en compte cette perturbation. C'est donc pour cela qu'elle n'apparaît pas dans les calculs.

- Le capteur de courant, selon le document technique, correspond à un gain de $K_{\text{courant}} = 0.1042$.
- La fonction de transfert du bloc de conditionnement peut se calculer à partir du schéma électrique dans la documentation technique:



Figure

1.1.4 - Schéma électrique du filtre $F(p)$ de conditionnement

Pour analyser ce schéma électronique on peut d'abord analyser la partie statique et ensuite la partie dynamique.

La partie statique servira à contrôler le gain statique du montage et la partie dynamique à calculer la fonction de transfert dynamique du système. Après calcul, on obtient la fonction de transfert finale suivante:

$$\frac{1.4569}{1+4.855 \cdot 10^{-6} p + 7 \cdot 10^{-14} p^2}$$

1.2 - Asservissement dans le domaine continu

Pour asservir notre système nous devons respecter le cahier des charges suivant:

- Marge de phase supérieure ou égale à 45° dans le pire des cas
- Fréquence de transition en boucle ouverte 300 et 500Hz
- Erreur statique nulle en boucle fermée.

Par la suite on notera $G(p)$ le bloc qui comprend le hacheur et le moteur et $F(p)$ le bloc qui comprend le capteur de courant et le conditionnement.

On cherche donc à asservir le système en boucle ouverte suivant : $G(p)F(p)$.

Il faut d'abord analyser le système en Boucle Ouverte, car si le système n'est pas stable ici, il ne le sera pas non plus en Boucle Fermée. L'enjeu est donc d'avoir une marche de phase positive à la fréquence de transition pour que le système soit stable. Si la marge de phase est négative, le système oscille et on perd la stabilité. Regarder la Boucle Ouverte peut également nous renseigner sur la vivacité du système.

On comprend donc que si on assure une marche de phase supérieure à 45 degrés en Boucle Ouverte on assure que le système soit stable en Boucle Fermée.

Maintenant, il nous faut aussi une erreur statique nulle en boucle fermée. Quel correcteur choisir ? Nous allons essayer plusieurs correcteurs, du plus simple au plus complexe pour trouver celui qui nous convient et qui nous permet de respecter le cahier des charges.

- Le correcteur proportionnel: $\$C(p) = k \$$

Si on a par exemple une marge de phase de 45 degrés à 200Hz on peut trouver un gain k tel que on arrive à avoir cette marge de phase pour la fréquence de transition que l'on veut, ici

400Hz. En effet un gain élevé shiftera la courbe de gain vers le haut et un gain faible à l'inverse la shifera vers le bas. Cependant l'erreur statique ne sera pas nulle et ne pourra jamais l'être avec ce type de correcteur à moins d'avoir un gain immense, ce qui entraînerait des instabilités et ce n'est pas ce que l'on recherche. Ce correcteur n'est donc pas le bon.

- L'intégrateur pur: $\frac{1}{p}$

Ce correcteur ajoute -20dB/décade à la courbe de gain. Cela permet d'avoir une erreur statique nulle, cependant cela fait descendre la phase de 90 degrés pour toute la courbe de phase et notre marge de phase est donc maintenant de 0 degré, ce qui n'est pas satisfaisant. On ne peut pas non plus prendre ce correcteur pour notre système.

- Le correcteur proportionnel intégral : $K_0 + \frac{K}{p} = \frac{1 + \tau_i p}{\tau_i p}$

Ce correcteur nous permet d'avoir une erreur statique nulle grâce à l'intégrateur et le gain K_0 permet de faire remonter la phase. On peut illustrer ce correcteur grâce au schéma de bode suivant pour plus de compréhension:

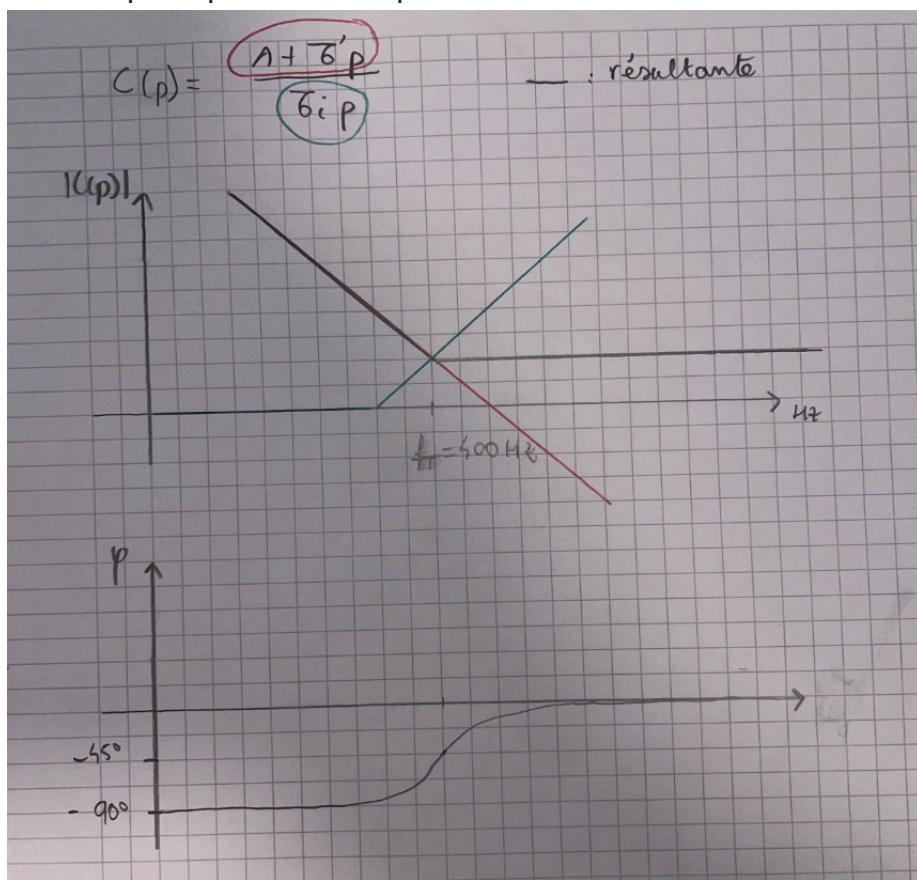


Figure 1.2.1 - Tracé de bode du gain et de la phase pour montrer le comportement du correcteur PI

Ce correcteur est donc parfait pour nos besoins. Il suffit de trouver les différents paramètres

pour le calibrer au plus proche de ce que l'on veut. Le calcul dépendra de ce que l'on veut pour le système.

- La première approche consiste à compenser le pôle du moteur (pôle dominant), ce qui simplifie l'équation. De plus, comme on se place aux alentours de la fréquence de transition qui vaut 400Hz, on peut négliger le pôle qui se trouve à 30kHz car sa dynamique n'entre pas en jeu ici. Il nous reste une expression simple dans laquelle il nous suffit de calculer K, le gain total du système et de trouver tau_i.
- La deuxième est celle sans compensation de pôle. Dans ce cas l'équation est bien plus fastidieuse et bien plus compliquée à résoudre pour trouver tau_i. Cependant, nous pouvons faire des simplifications car on travaille autour d'une fréquence particulière et ainsi l'équation est bien plus rapide à résoudre.

Dans ce BE, nous avons choisi d'appliquer la compensation de pôle. Nous allons donc compenser le pôle du moteur ($\tau' = 0.0020$) et nous avons donc cette équation à résoudre:

$$\left| \frac{K}{\tau_i p} \right| = 1 \Leftrightarrow \left| \frac{K}{j * \tau_i * 2 * \pi * 400} \right|$$

où $K = K_{moteur} * K_{hacheur} * K_{capteur-courant} * K_{bloc-F} = 7.28$

Ainsi, on peut facilement calculer tau_i:

$$\tau_i = \frac{7.28}{2 * 400 * \pi} = 2.9 \cdot 10^{-3}, \text{ soit une fréquence de } 55\text{Hz}$$

Nous avons donc tous les éléments pour calculer notre correcteur et faire les simulations sur Matlab et simulink !

Version 1 :

Dans notre première version du système bouclée avec simulink, on a la configuration suivante:

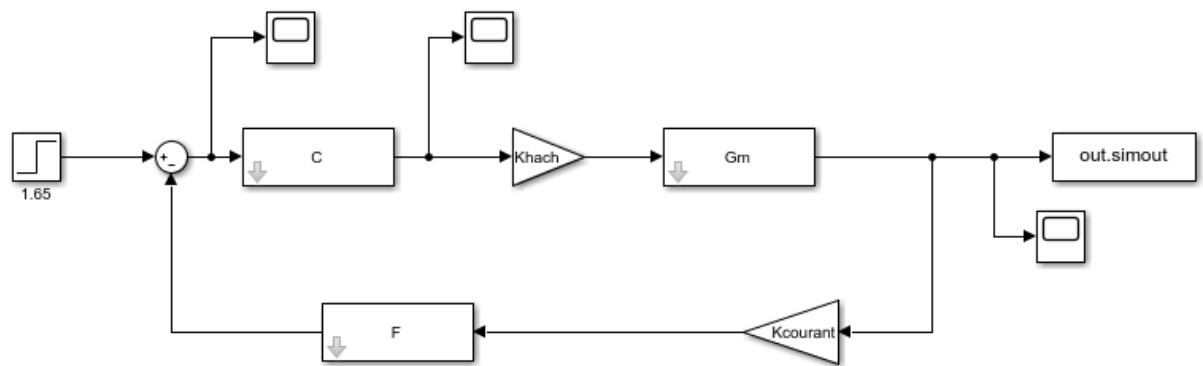


Figure 1.2.2 - Simulink système asservi version 1

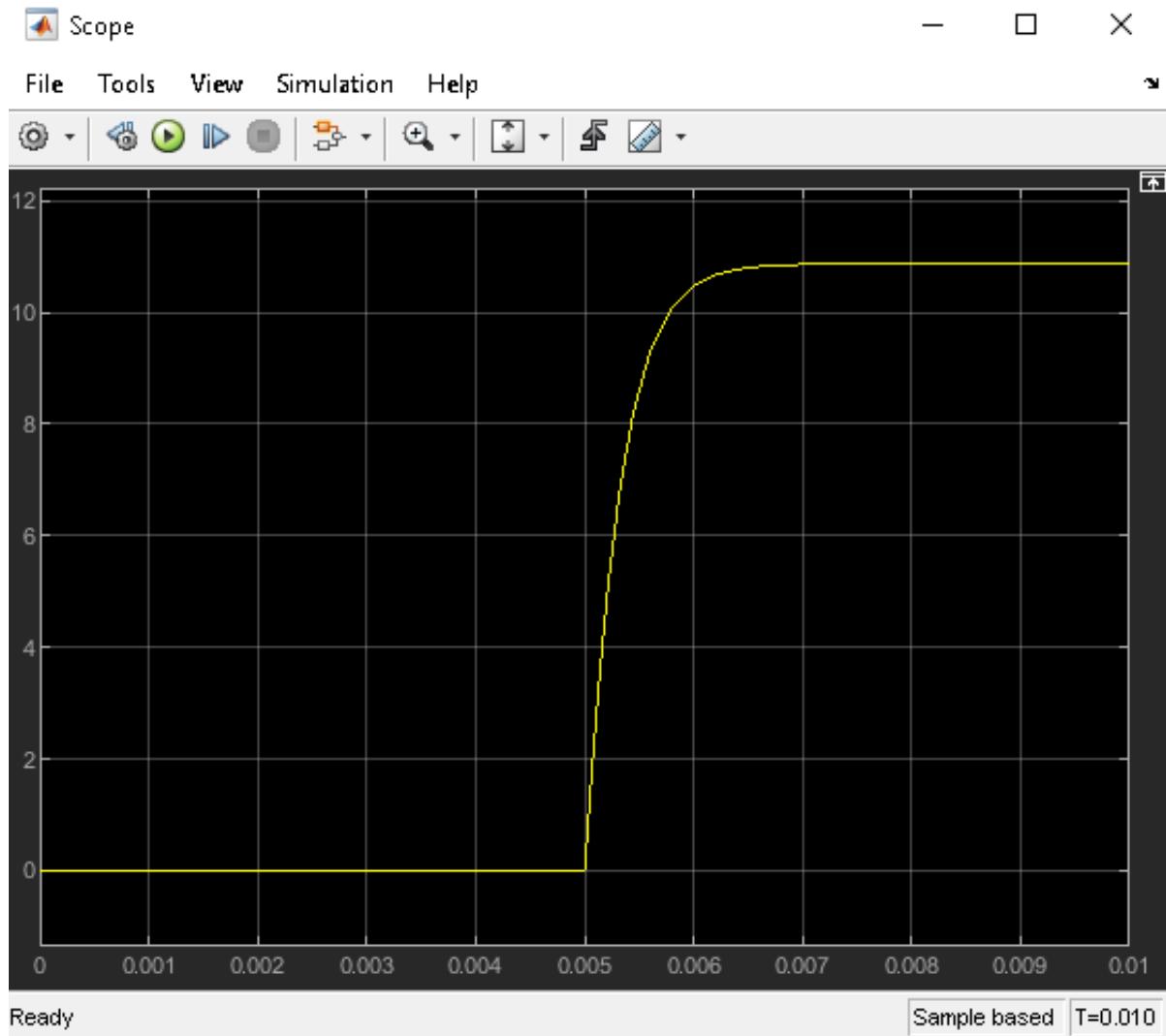


Figure 1.2.3 - Simulink - réponse à un échelon d'amplitude 1.65

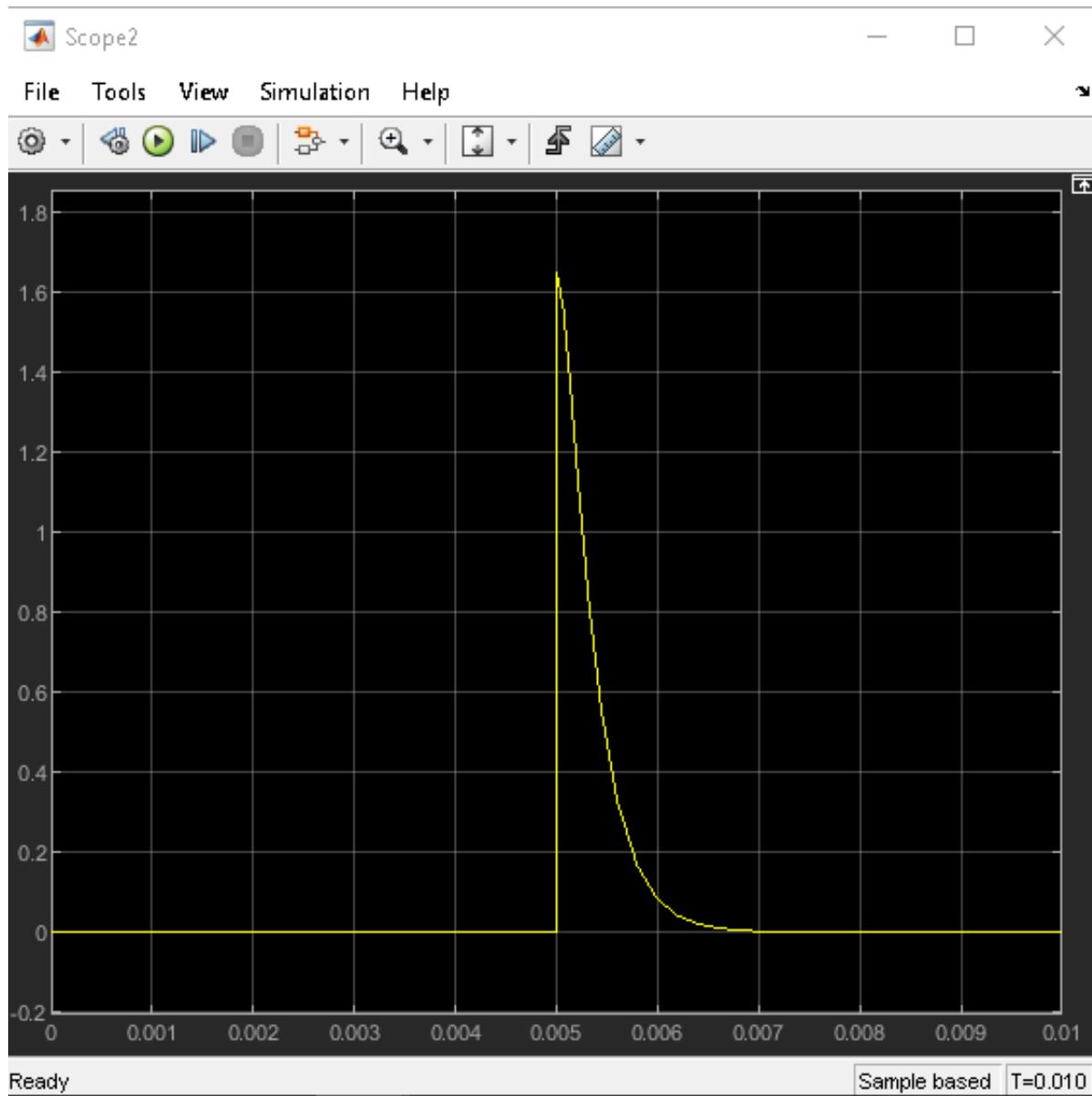
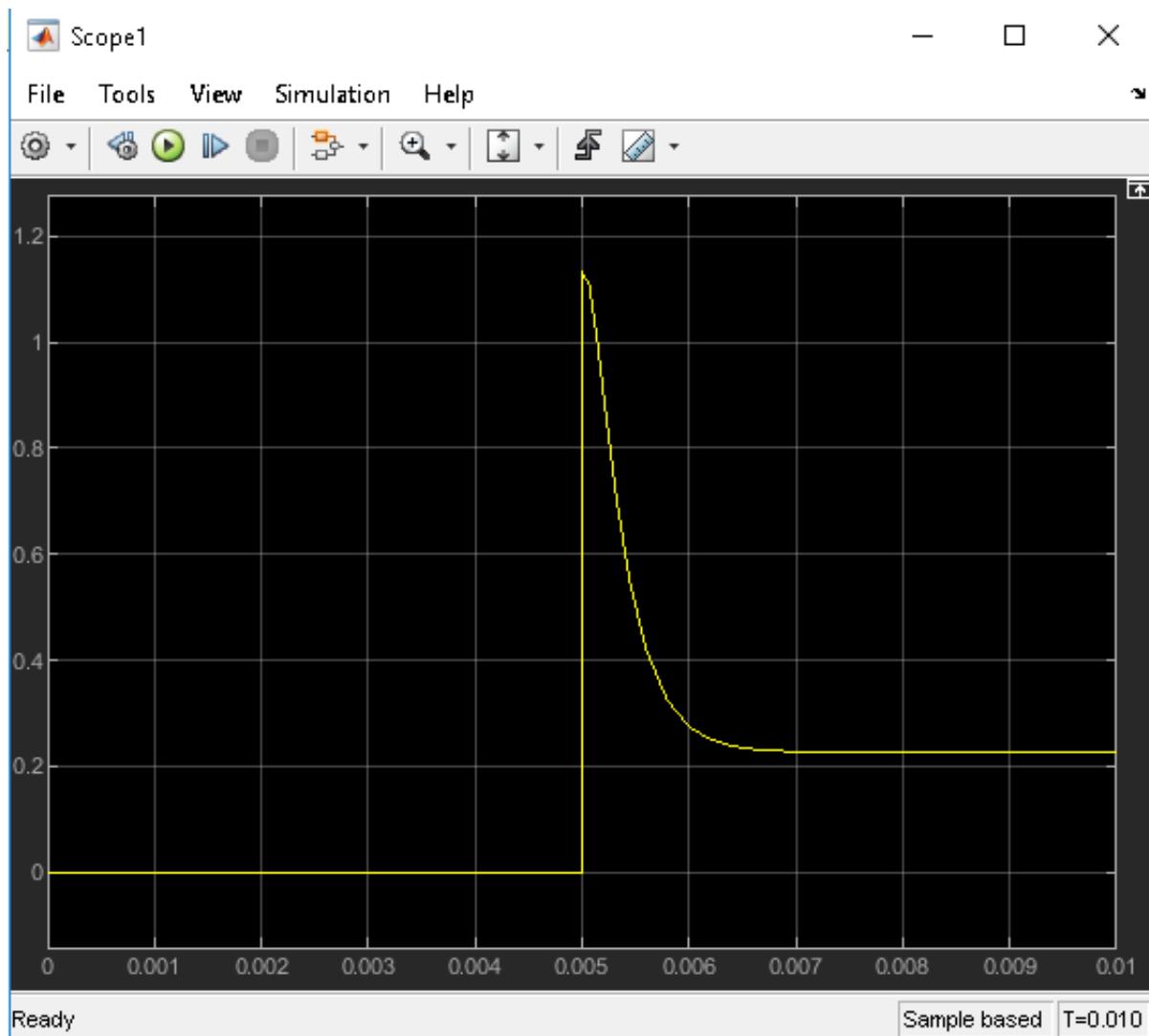


Figure 1.2.4 - Simulink - erreur du système - On peut voir que le système en boucle fermée arrive à se réguler et après un temps très court l'erreur devient nulle. Le système répond très rapidement.



Ready

Sample based T=0.010

Figure 1.2.5 - Simulink - Le signal alpha - Sur ce graphique, on peut voir que le signal pwm qui est envoyé à l'issue du bloc correcteur du système dépasse à un moment les valeurs admises. Normalement, alpha, le rapport cyclique est censé être compris dans l'intervalle [-0.5; +0.5].

On peut également voir que alpha ne reste pas dans l'intervalle des valeurs admissibles. Cela est dû au fait que l'on voit une valeur très grande et donc le système envoie un signal pwm de rapport cyclique supérieur à 100% pour répondre le plus rapidement possible. Cependant, un rapport cyclique supérieur à 100% n'a aucun sens physique. Dans la réalité, le système sera en saturation. Pour modéliser la saturation et être au plus proche de la réalité nous avons réjouté un bloc saturateur dans le schéma simulink pour contraindre les valeurs de alpha.

Version 2 :

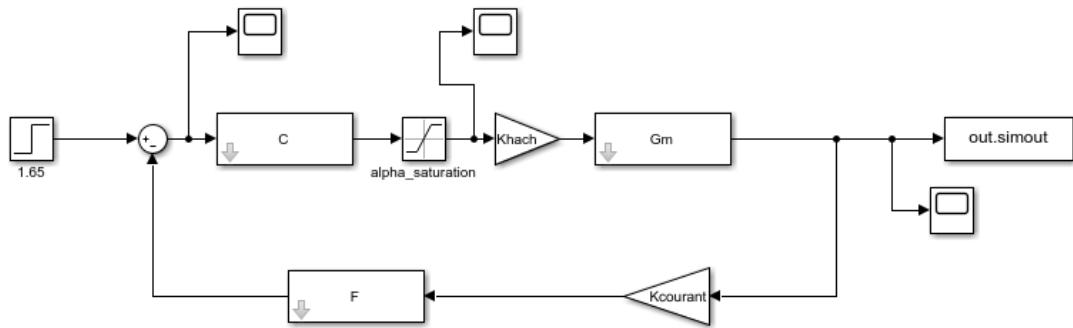


Figure 1.2.6 - Version 2 du système sous simulink - on fait évoluer notre système avec un bloc saturateur qui fait que l'alpha est à nouveau compris dans l'intervalle $[-0.5; +0.5]$ afin d'assurer que le signal PWM de alpha ait un sens physique, qu'il ne dépasse pas $\pm 100\%$.

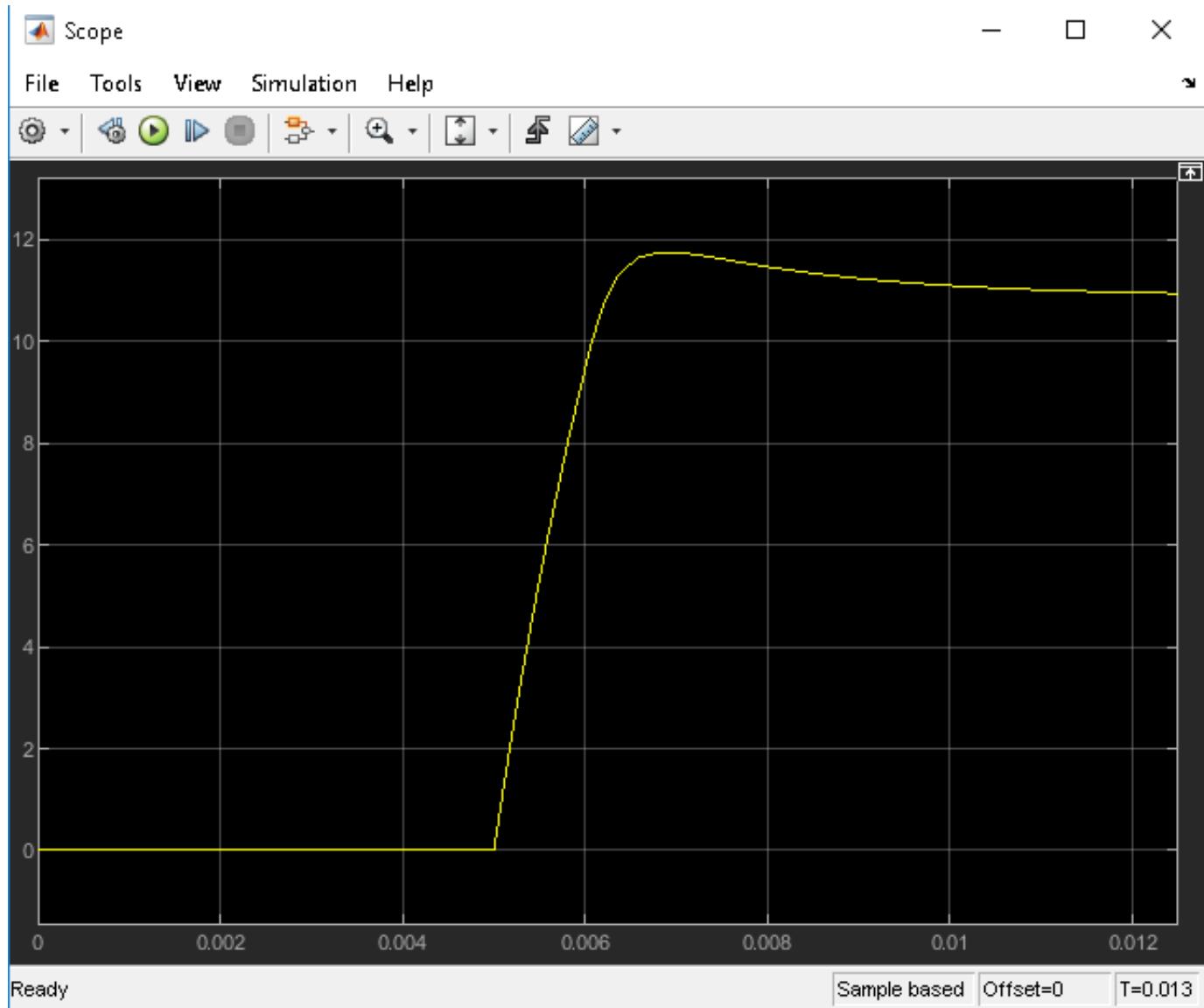


Figure 1.2.7 - La réponse à un échelon de 1.65 du système avec saturateur. On voit l'apparition

du phénomène de dépassement lié à la saturation lorsque l'on essaie d'exciter un système plus rapidement que son slew rate.

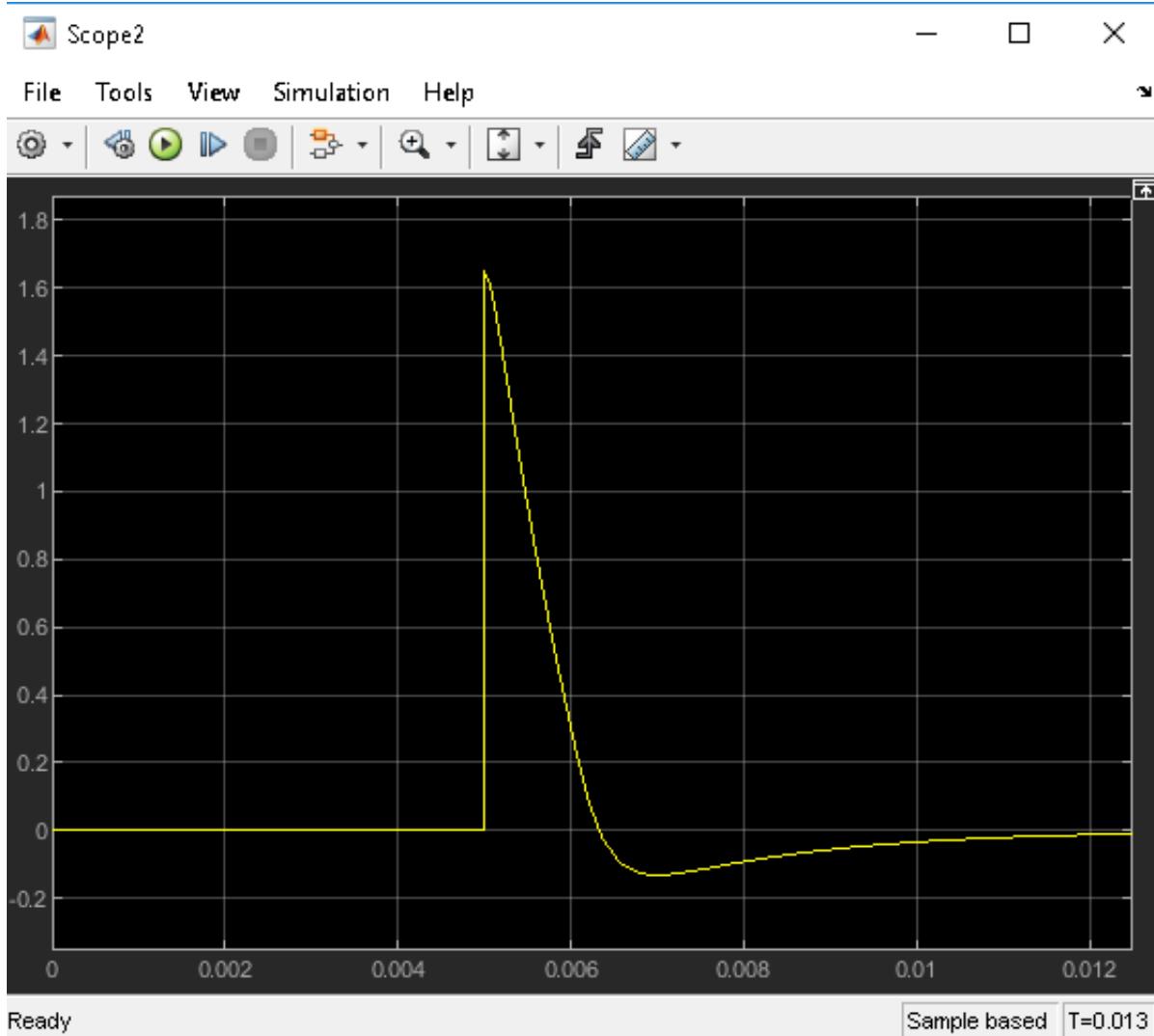


Figure 1.2.8 - L'erreur pour la version 2 du système (avec saturateur) On peut voir que non a une erreur nulle en régime permanent. Cependant, le temps pour l'obetnir est un peu plus long que dans la version 1. Ceci est tout à fait normal étant donné que nous avons contraint la valeur de alpha, le système répond un peu plus lentement. D'où cette allure de courbe.

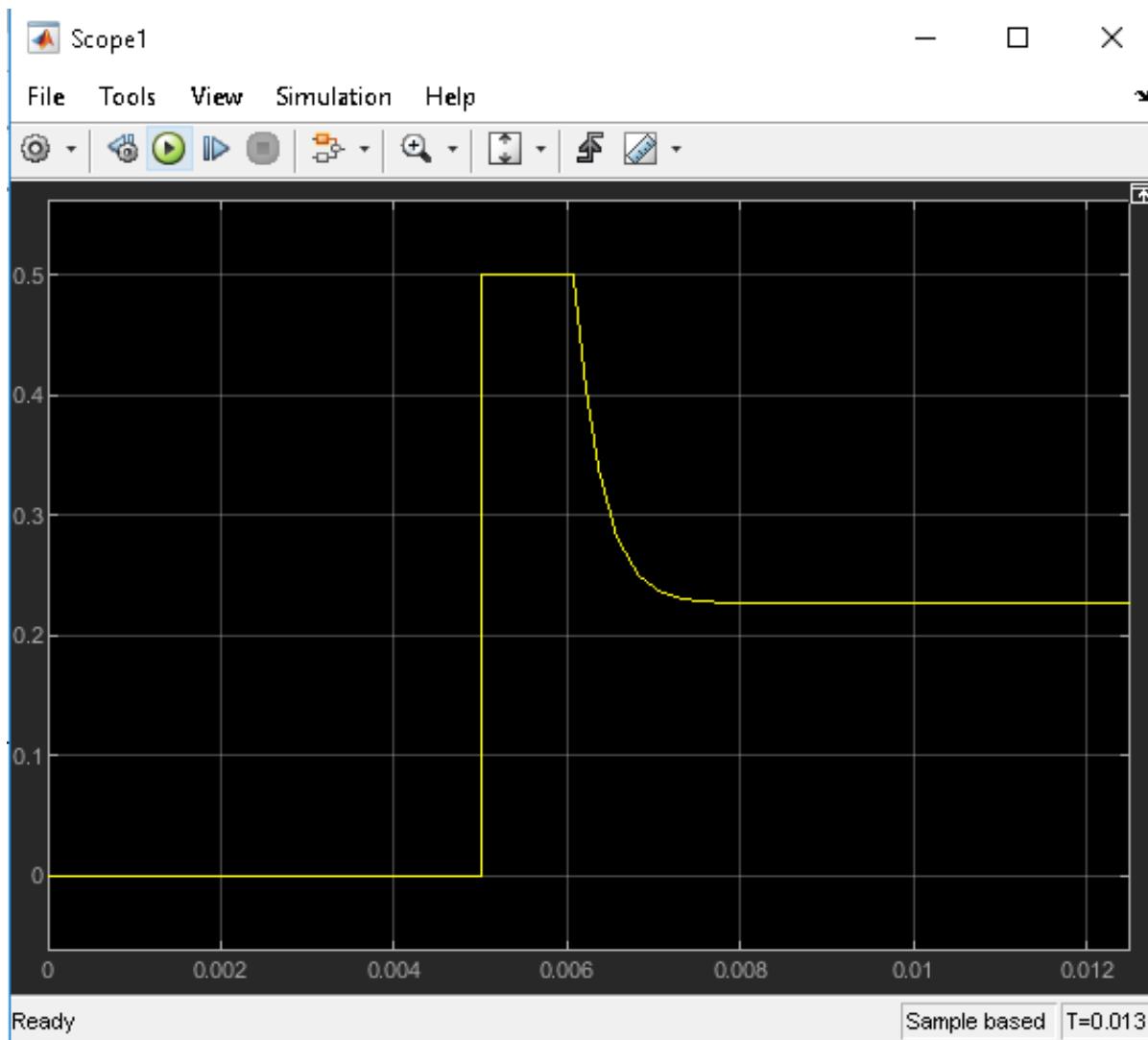


Figure 1.2.9 - L'alpha quand on ajoute un bloc de saturation à ± 0.5 . On voit que alpha sature pendant un court instant jusqu'à que le système réponde et que alpha se stabilise.

1.3 - Passage au discret - Asservissement dans le domaine discret

Lorsque l'on passe du continu vers le discret, on passe d'une représentation en continu comme ceci : $C(p) = \frac{1 + \tau_c p}{\tau_i p} = \frac{K}{p} + \frac{\tau_c}{\tau_i}$. À une représentation discrète comme ceci : $C(z) = \frac{a_0 z - a_1}{z - 1}$. On fait cette passage en gardant tous les propriétés du système comme les pôles et les zéros grâce à la méthode de Tustin, autrement dit la transformée bilinéaire. Pour cette transformée, on pose : $p = \frac{2}{T_e} \frac{z - 1}{z + 1}$ et on insère l'expression de p dans l'expression de $C(p)$ afin de retrouver la fonction de transfert de C discrétisée, maintenant avec la variable z :

- On pose a_0 et a_1 tels que :
- $$a_0 = \frac{T_e + 2\tau_c}{2\tau_i} \quad a_1 = \frac{-2\tau_c}{2\tau_i}$$

Donc on obtient la fonction de transfert suivante pour $C(z)$:

$$\$\$C(z) = \frac{a_0 z - a_1}{z - 1}\$\$$$

Version 3 :

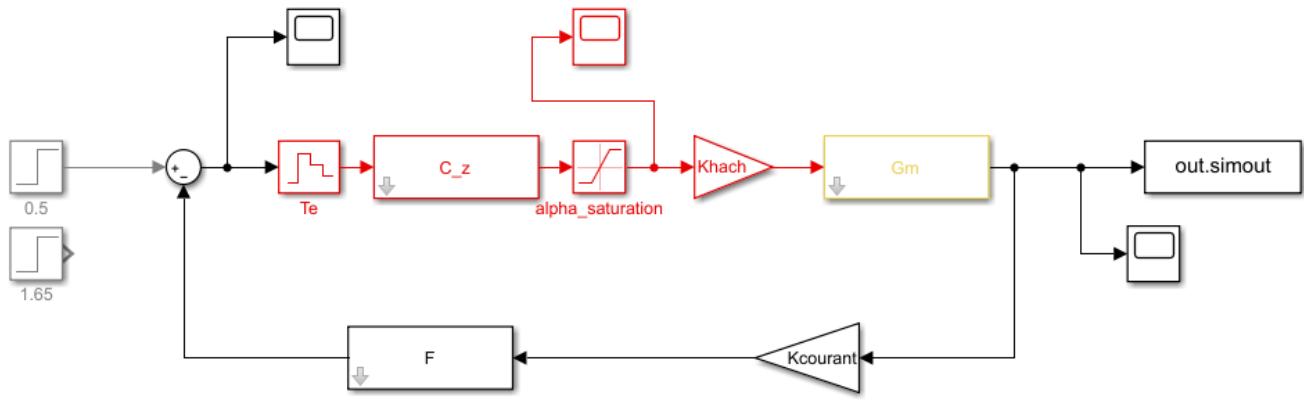


Figure 1.3.1 - Simulink - Version 3 du système, on a enlevé le bloc correcteur continu de la version 2 et on a ajouté un bloc correcteur discret. Ici les couleurs correspondent aux différents domaines des signaux, c'est-à-dire que le noir est en continu, le rouge est en discret, et le bloc jaune correspond à un bloc qui convertit un signal discret en continu. On a choisi de rajouter le bloc échantillonneur Te en amont du bloc C_z afin de discréteriser avec la bonne période d'échantillonnage Te .

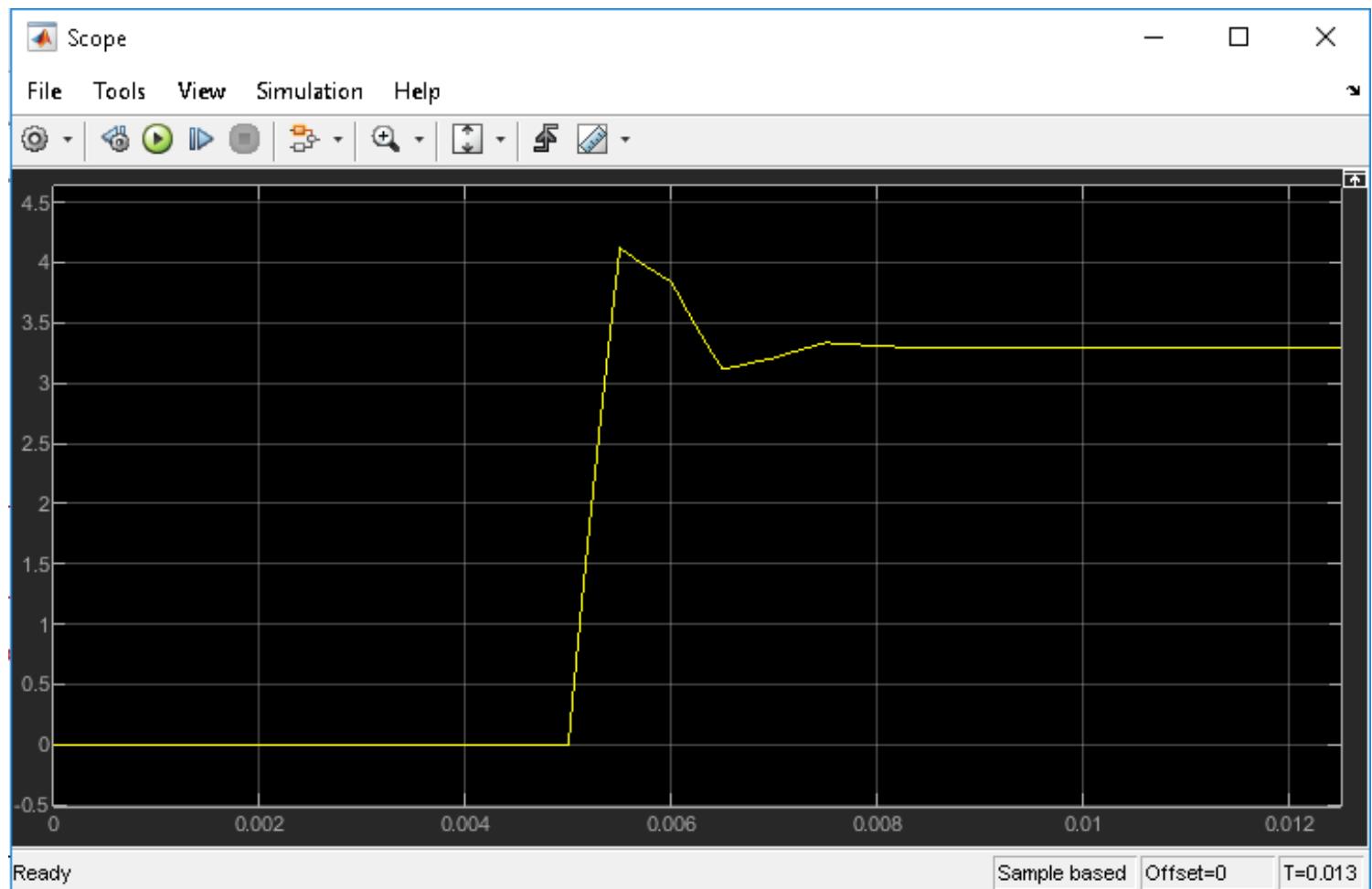


Figure 1.3.2 - Réponse à un échelon de 1.65 avec le correcteur discret. On remarque que le phénomène de dépassement lié à la saturation y est toujours. On remarque que le phénomène de dépassement lié à la saturation y est toujours. De plus, on peut voir que la courbe de la version 3 est bien moins lisse que celle de la version 2. Cela est dû à la période d'échantillonage. Cependant, les allures sont les mêmes.

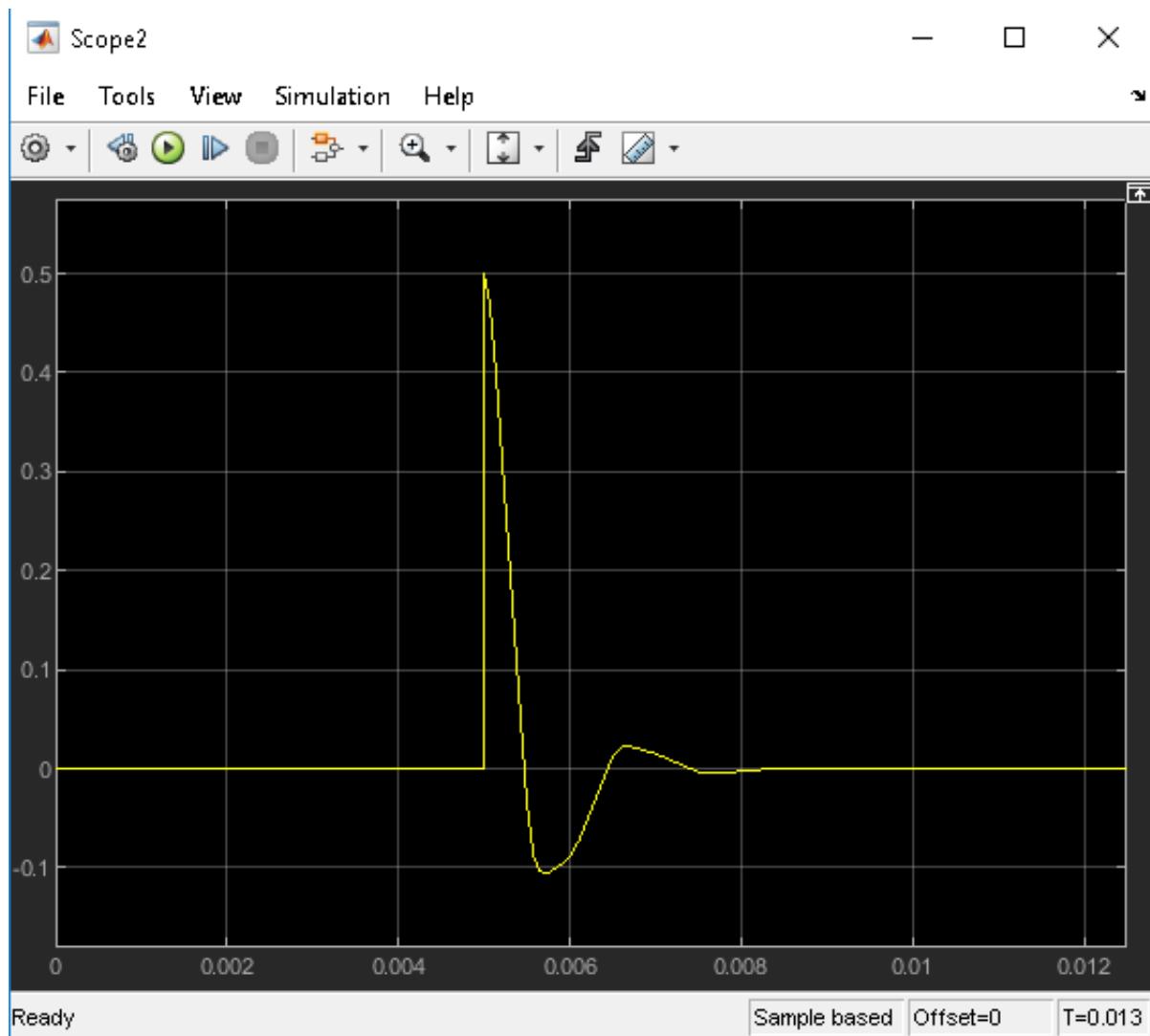


Figure 1.3.3 - L'erreur lorsque l'on passe en discret. On remarque une faible oscillation du à l'échantillonage. Cependant l'erreur est également nulle en régime permanent et la réponse est rapide.

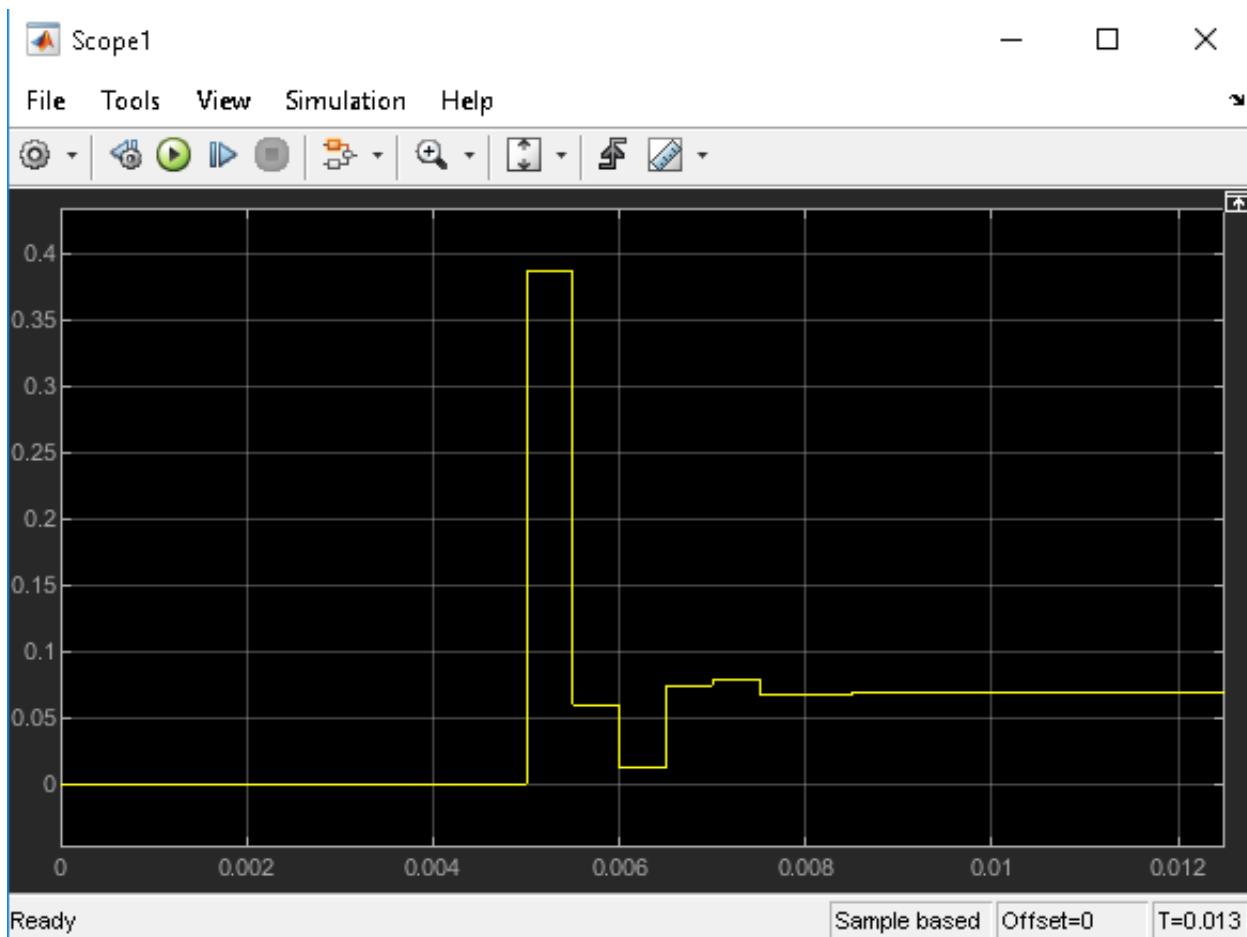


Figure 1.3.4 - Le signal PWM de alpha lorsque le correcteur est discrétré.

Version 4 et 5 :

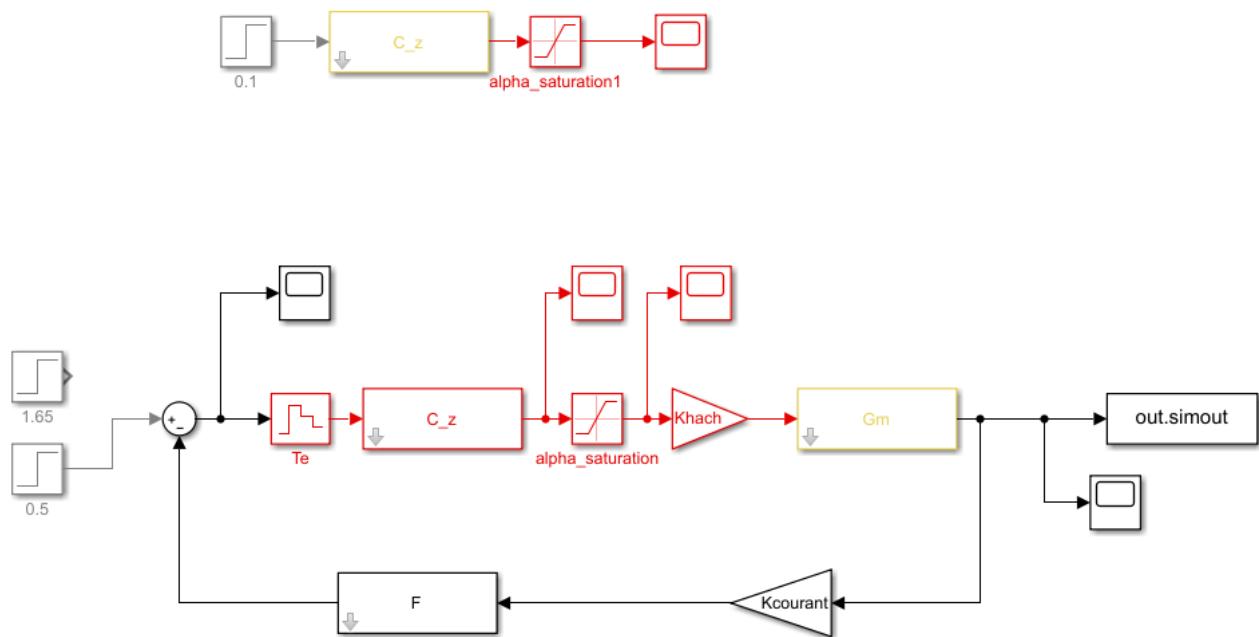


Figure 1.3.5 - Simulink quand on souhaitait vérifier le bon comportement du correcteur discret avec le correcteur codé avec Keil. On excite le bloc correcteur avec un petit échelon d'entrée

afin de mettre en évidence le K du premier pas de la réponse, qui est une des caractéristiques de notre correcteur PI.

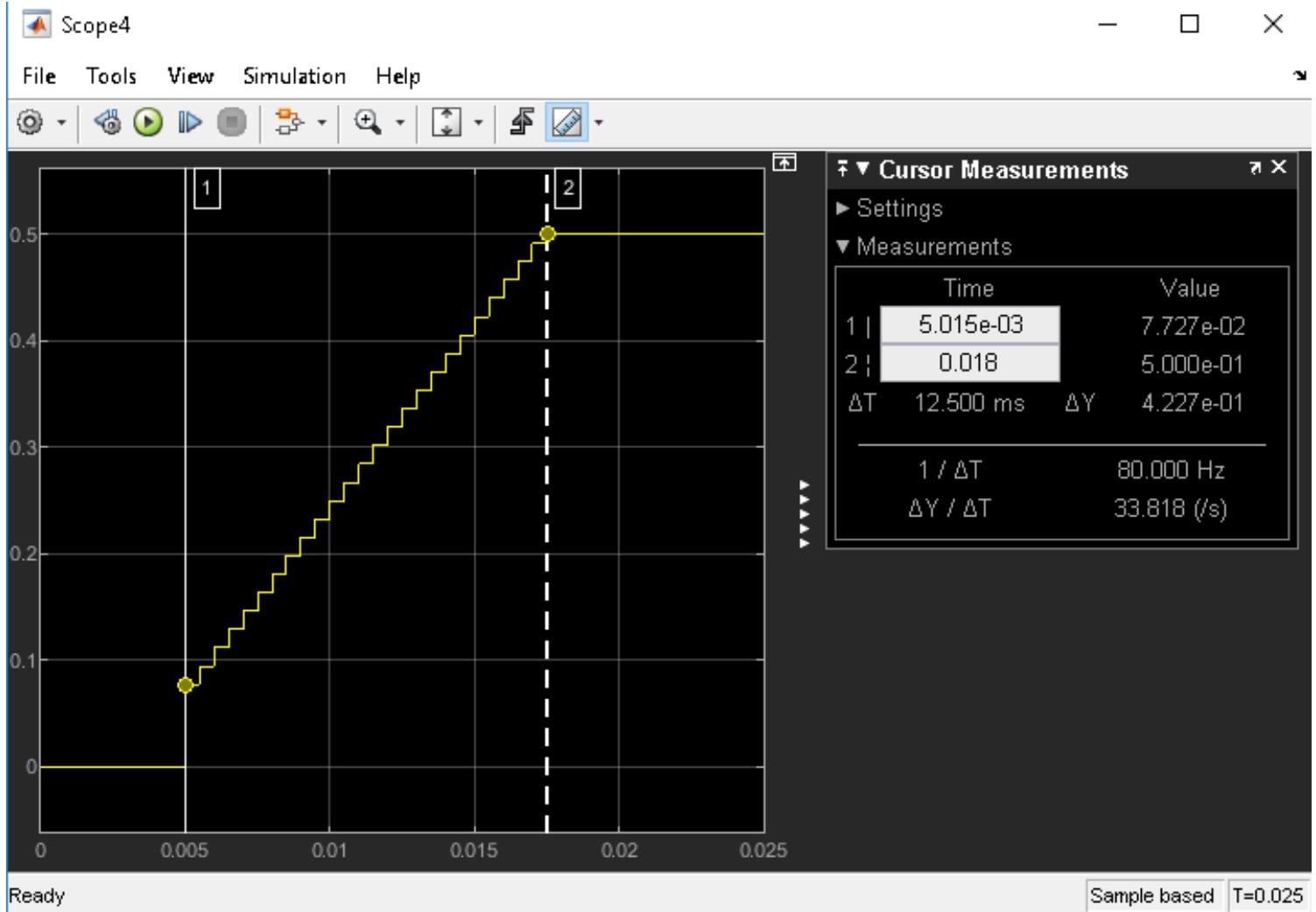


Figure 1.3.6 - Réponse à un échelon du bloc correcteur seul pour vérifier le comportement dans keil par rapport à celui dans simulink. Ici on voit un premier pas de $K=0.077$, avant de monter jusqu'à +0.5 après 12.5ms. On s'arrête à $\alpha=0.5$ grâce au bloc saturateur en aval du correcteur. Ce comportement va servir de test pour notre correcteur numérique afin qu'on puisse tester le bon fonctionnement du correcteur avant de le câbler physiquement. Cette vérification évite d'asservir le banc de trottinette avec un comportement qui n'est pas prévu qui peut engendrer des conséquences sur le matériel ou peut mettre l'utilisateur en risque.

1.4 - Asservissement avec correcteur numérique avec µcontrôleur

Jusqu'ici, on a vu comment on peut modéliser et asservir ce couple en continu puis en discret avec des correcteurs théoriques (analogiques) sur papier et avec Matlab et Simulink. Dans cette partie, on va voir comment on peut faire cette même conception, de manière numérique, avec un STM32/carte nucléo. De plus, le fait d'attaquer l'asservissement en numérique permet d'éviter quelques phénomènes qui se montrent lorsque l'on modélise le système

analogiquement avec Simulink, comme le dépassement lié à la saturation(figure 1.2.7) lorsque le slew rate du système n'est pas "respecté", mais aussi de faire en sorte que la configuration du correcteur soit plus finement réglée pour chaque système physique en recalculant les caractéristiques de ce dernier pour chaque trottinette.

Pour ce travail de bureau d'étude, on nous a fourni un "Toolbox", une boîte à outils contenant toute la couche service et driver pour configurer et mettre en œuvre le correcteur numérique avec un STM32. Lors de notre réalisation, on a tout simplement défini des macros (#define ...) contenant les grandeurs physiques et les gains dont on avait besoin, issues de la modélisation théorique à la main et sur matlab et simulink. Cela nous permettait ensuite de définir des formules qui calculent automatiquement les nouveaux coefficients du correcteur spécifique lorsque les valeurs de temps d'échantillonnage "Te"/fréquence d'échantillonnage Fe, différents gains, résistances etc. changent. Cela permettait également de plus facilement voir et comprendre comment le comportement du système pourrait changer lorsque ces grandeurs physiques changent, surtout le temps d'échantillonnage qui a un grand impact sur l'asservissement numérique.

Pour rentrer plus dans les détails de l'implémentation du correcteur numérique: Afin de libérer les ressources du microcontrôleur, ainsi de faire intervenir périodiquement les actions du correcteur chaque période d'échantillonnage "Te", on implémente le correcteur avec une interruption principale basée sur un Timer. Cette interruption est configurée afin d'être levée chaque fois qu'un temps Te s'écoule. Le bon choix de cette période est impératif afin d'assurer le bon fonctionnement du correcteur numérique. Regardons cela de plus près:

- Si on choisit Te trop petit, alors :
 - On va avoir des énergies trop élevées lorsque la fréquence augmente, ce qui dissipe davantage la puissance et la chaleur. De plus, on peut avoir un phénomène d'interruptions qui ne s'effectuent pas assez rapidement, et la même interruption va être pending et la commande ne s'effectuera plus en temps réel.
- et si on choisit Te trop grand, alors :
 - On va voir l'apparition d'instabilités parce qu'on n'a pas assez de points pour bien discréteriser/échantillonner le signal en entrée du bloc correcteur de notre système.

1.4.1 - De la fonction de transfert en z vers l'équation récurrente du correcteur

On a retrouvé l'équation récurrente pour le calcul de l'alpha de la manière suivante : $\$C(z) = \frac{Y(z)}{U(z)} = \frac{a_0 z - a_1}{z - 1} \quad Y(z)(z - 1) = U(z)(a_0 z - a_1)$ Avec la transformée inverse en z, on obtient:

$$y_{n+1} - y_n = a_0 e_{n+1} - a_1 e_n$$

On pose n=n+1, et on obtient l'équation récurrente :

$$y_n = y_{n-1} + a_0 e_n - a_1 e_{n-1}$$

Avec alpha = y et l'erreur = e

Cette fonction récurrente nous permet de calculer le nouveau alpha / nouveau rapport cyclique avec lequel on va commander le système à la sortie du bloc correcteur C(z) numérique.

1.4.2 - Étapes de la conception du correcteur numérique

L'utilisation du périphérique ADC fait que l'on doit réfléchir à ce à quoi correspondent les différents signaux et leurs plages de valeurs. Par exemple, pour le calcul de l'erreur où on prend l'écart entre le courant et la consigne, il faut penser à diviser ce résultat avant de l'utiliser dans le calcul de alpha. Sinon, il faut bien garder en tête ce que ces valeurs représentent. Par défaut, ce que renvoie la fonction Entrée_3V3() de l'ADC est compris entre [0;4096]. Pour cela, il faut penser à ramener ces valeurs à la plage [0;3.3] si on souhaite avoir un alpha qui est bien compris entre [-0.5;+0.5]. Cela permet de faire le bon calcul de l'erreur, qui est utilisée pour le calcul de l'équation récurrente des nouvelles valeurs de alpha.

Afin de coder l'interruption principale de notre correcteur en C on a suivi les étapes suivants:

1. Initialisation

On initialise les valeurs de l'alpha et de l'erreur à n=0 à 0.0, la valeur de repos. Celles-ci sont importantes pour pouvoir commencer le calcul de l'équation récurrente

2. Acquisition de la consigne

On fait cela grâce à la fonction fournie dans le toolbox Entrée_3V3().

3. Acquisition du courant

On retrouve la mesure du courant grâce à la fonction I1(), également fournie.

4. Calcul de la valeur courante d'epsilon, l'erreur entre le courant et la consigne

On commence par faire le calcul de l'écart : epsilon = (Cons_In - Courant_1)

On ramène cette valeur à la plage de valeurs correspondante : Il faut multiplier par 3.3 et diviser par 4096 pour avoir une valeur qui correspond à la plage de valeurs de notre erreur afin de pouvoir calculer un alpha compris entre [-0.5;+0.5]. La valeur issue de cette opération est une float : epsilon = 3.3*(Cons_In - Courant_1)/4096

5. Calcul de la nouvelle valeur analogique de alpha avec l'expression déduite de l'équation récurrente. L'alpha analogique signifie la sortie comprise entre [-0.5;+0.5]. On utilise la formule déduite de l'équation récurrente:

$$\text{alphaAnalogique}_n = \text{alphaAnalogique}_{n-1} + a_0 \text{ erreur}_n + a_1 \text{ erreur}_{n-1}$$

Ici, a_0 et a_1 correspondent aux valeurs des coefficients que l'on a calculés pour la fonction de transfert en z , et sont exprimés ci-dessous :

$$\begin{aligned} \$\$a_0 &= \frac{T_e + 2\tau_c}{2\tau_i} \\ \$\$a_1 &= \frac{T_e - 2\tau_c}{2\tau_i} \\ \$\$C(z) &= \frac{a_0 z - a_1}{z - 1} \end{aligned}$$

6. Saturation

On sature la sortie alpha pour s'assurer qu'il est bien compris dans la plage [-0.5;+0.5]. Cela est fait pour être sûr que la sortie du bloc correcteur, qui est sous forme de signal PWM - pulse width modulation, a un sens physique. Quand alpha atteint (resp.) +0.5 et -0.5, cela correspond à, respectivement, un signal PWM de 100% (toute la période en état haut) et un signal PWM de 0% (toute la période en état bas). Cette saturation est faite tout simplement avec un if else if qui redéfinit la valeur de alpha à la borne supérieure ou inférieure lorsqu'il dépasse.

7. Mise à jour de l'ancienne valeur d'alpha et d'erreur

Le fait de les affecter après la saturation fait que l'on arrive à éviter le phénomène de dépassement lié à la saturation que l'on a vu lors de la simulation avec Simulink en discret (avec $C(Z)$). Faire réf à l'ancienne courbe/figure avec lien.

8. Recalcule la valeur numérique de alpha, qui est comprise dans la plage [0;4096] permettant de commander la fonction R_Cyc_1 / R_Cyc_2 qui mettent à jour les signaux PWM en sortie.

1.4.3 - Vérifier et débugger le code - mauvais calcul des macros

Après le premier essai lorsque l'on a lancé la simulation du correcteur numérique on a constaté la présence d'erreurs qui faisaient que l'on n'obtenait pas le bon comportement en sortie. Au début, notre alpha n'avait pas l'air d'augmenter, mais c'était tout simplement dû au fait que les macros que l'on avait définis pour les coefficients du contrôleur ne donnaient pas la bonne valeur. On se rendait compte après quelques essais-erreurs que c'était pas bon, mais on a réussi à résoudre le problème en décomposant les calculs et en regardant les résultats que cela donnait. On avait bien utilisé la bonne expression, néanmoins les macros ne calculaient pas le résultat que l'on avait prévu. Ce problème a été résolu en reformulant les expressions de ces coefficients (a_0, a_1), ainsi que de définir la valeur de la période d'échantillonnage T_e directement, au lieu de l'exprimer comme l'inverse de la fréquence d'échantillonnage.

Suite à cela, on a à nouveau regardé la sortie du bloc correcteur. Cette fois-ci, on avait presque le bon résultat qui colle avec la simulation en Simulink. Cependant, on a vu l'apparition d'une espèce de bâtonnets quand alpha saturait à 0.5 qui allaient au-delà. On peut voir ces bâtonnets sur la figure 1.4.4.3 lorsque alpha = 0.5. On n'a pas encore compris d'où ils viennent.

1.4.4 - Vérification en simulation - comparaison Simulink/Keil

Après avoir codé le correcteur numérique discret, c'était important de vérifier que l'on obtient bien le comportement souhaité de ce bloc. Sous Simulink on a donc isolé le bloc $C(z)$ avec seulement un petit échelon en entrée d'amplitude 0.1, et un bloc saturateur et un scope en aval.



Figure 1.4.4.1 - Schéma bloc du correcteur seul sous simulink lors de la vérification du comportement du simulink vs celui obtenu avec Keil. Le comportement prévu était le suivant:

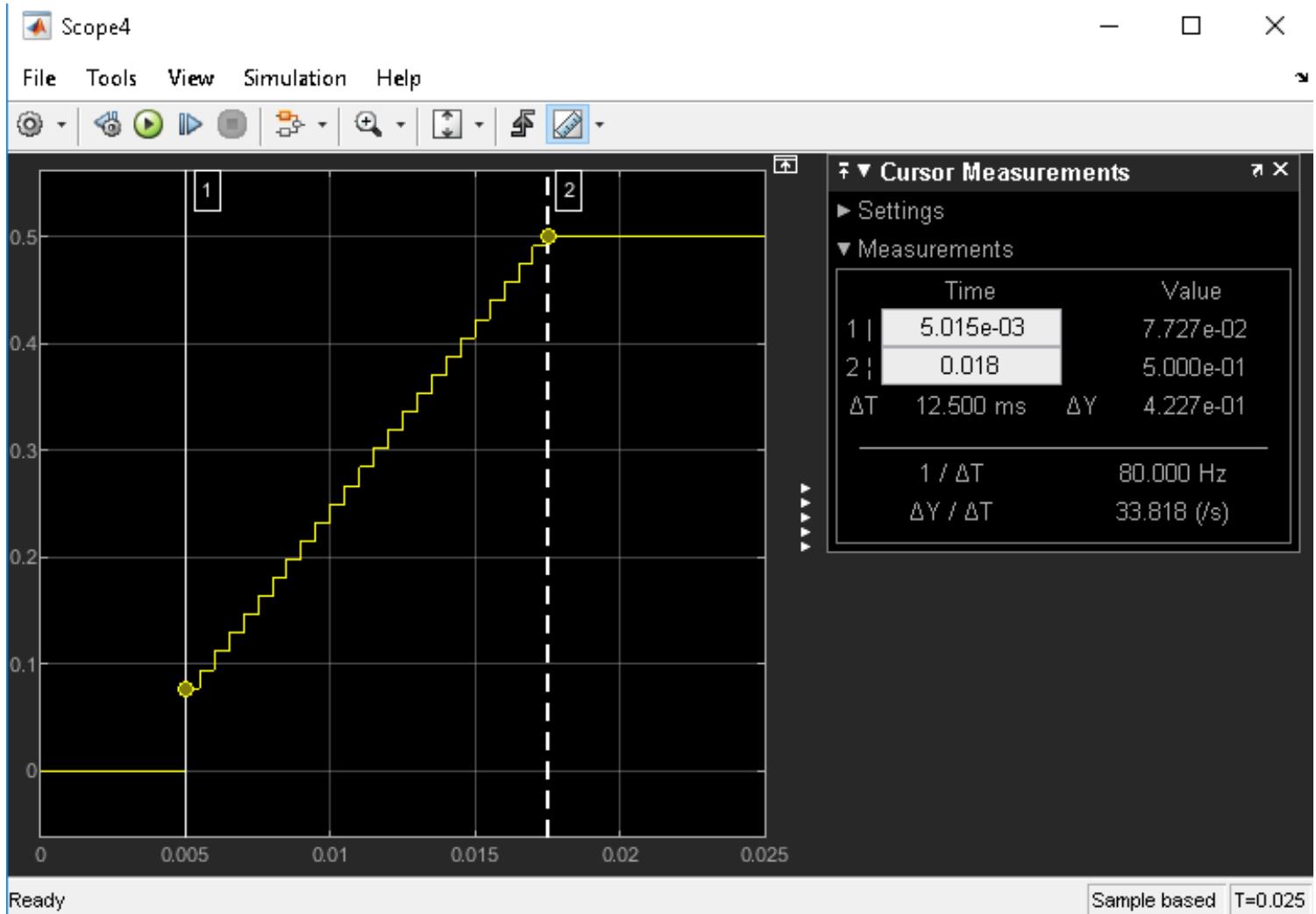


Figure 1.4.4.2 - Tracé de la réponse à un échelon de 0.1 du bloc correcteur sous Simulink, qui

met en évidence le comportement d'intégrateur souhaité.

Dans ce schéma on constate que le comportement observé était composé d'un premier step égale à $K=\tau_c/\tau_i=0.077$, suivi de petits pas en escalier qui monte (pense: intégrateur) jusqu'à l'obtention de $\alpha=0.5$. Le temps que met l'intégrateur à monter jusqu'à $\alpha=0.5$ depuis le premier pas K est ici égale au delta $\Delta T=12.5\text{ms}$. Il fallait vérifier que ces valeurs, et donc le comportement du bloc correcteur (si on excite avec la même entrée) était exactement la même sous Keil et Simulink. On a commencé par réécrire l'expression de $C(p)$, qui nous donnait une idée de ce à quoi on s'attendait, avant de faire une application numérique. En regardant les courbes obtenues en simulation avec Keil et Simulink, les valeurs de K et tau collent bien avec notre calcul théorique. On obtient bien le même $K=0.077$ (premier pas) et temps de réponse de $\alpha=K=0.077$ jusqu'à $\alpha=0.5$, $\Delta T=12.5\text{ms}$.

Réécriture de l'expression du correcteur PI : $\frac{1 + \tau_c p}{\tau_i p} = \frac{K}{p} + \frac{\tau_c}{\tau_i}$

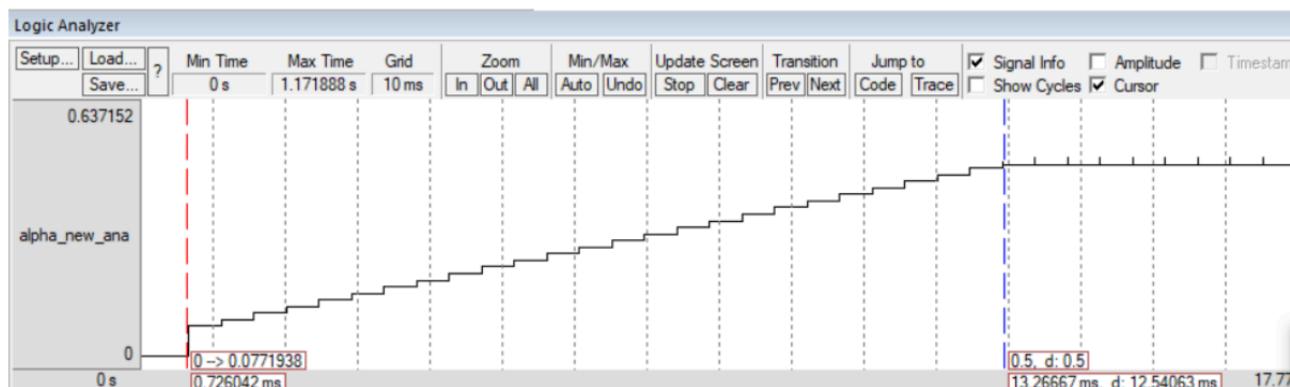


Figure 1.4.4.3 - Tracé de la réponse à un échelon de 0.1 du bloc correcteur sous keil

1.4.5 - Implémentation et essais du correcteur

Avant de tester la carte de puissance que l'on a conçue, on a vérifié que le banc trottinette fonctionnait. Pour cela on a branché le banc à une carte de puissance analogique qui a été conçue préalablement. Quand on a vu que cette carte et donc le correcteur fonctionnait comme prévu, c'est-à-dire qu'on réussissait à commander la trottinette vers l'avant, l'arrière, et arriver à maintenir une couple nulle. On pouvait donc passer à l'étape de test du correcteur numérique en réel. Lors de l'implémentation du correcteur, on a commencé par configurer la carte de contrôle de puissance contenant la carte STM32. Ensuite, on a branché un générateur basse-fréquence à l'entrée In de la carte. De plus, on a relié les deux PWM en opposition de phase au banc trottinette avec deux fils blancs. On a également récupéré la tension Vcourant, venant du banc trottinette (à l'issue du capteur courant après conditionnement par le bloc de filtre F(p)), et l'a branché à la carte de contrôle de puissance à l'entrée du courant. On a relié la carte à la masse

du banc trottinette, et on a repéré la commande du potentiomètre au niveau de l'entrée 3V3 (même endroit que le GBF) permettant de régler la consigne d'entrée.

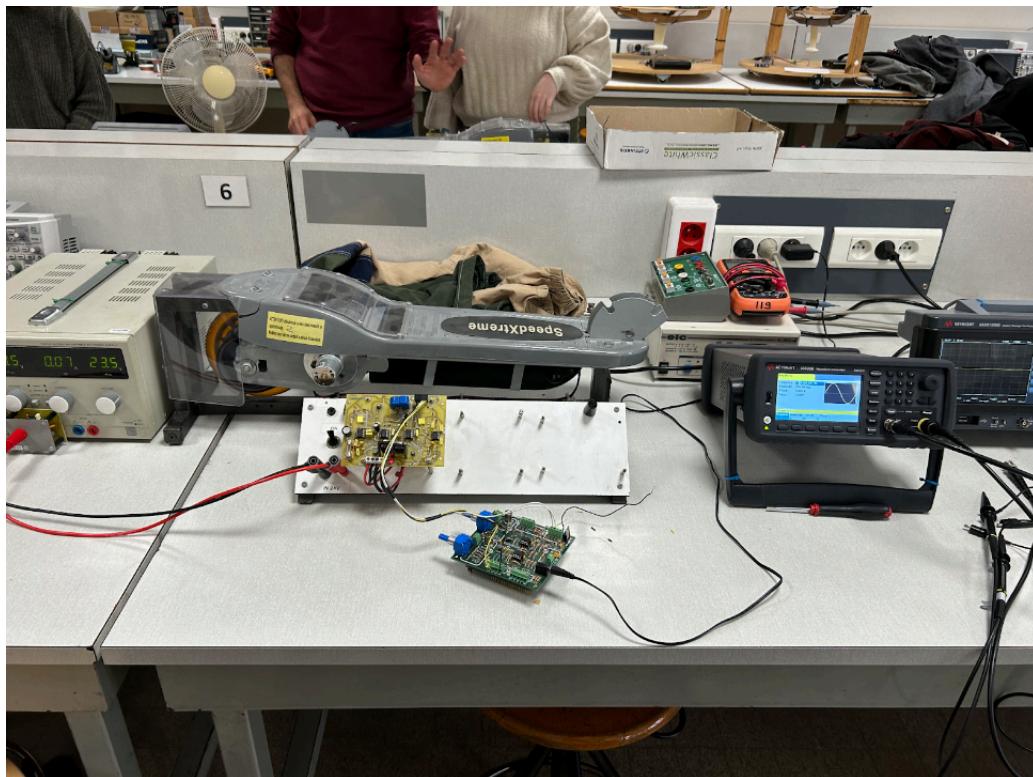


Figure 1.4.5.1 - Banc trottinette câblé avec la carte de puissance codée.

Le système récupérateur d'énergie est relié au banc trottinette avec un système de contrôle de tension avec un transistor et une partie dissipatrice permettant de dissiper de la chaleur lorsqu'il y a trop de récupération d'énergie afin d'éviter une augmentation/accumulation de tension. La trottinette est alimentée à travers le système de récupération d'énergie qui sert de mécanisme de protection entre l'alimentation et la trottinette.

Finalement, on a relié deux sondes de l'oscilloscope pour visualiser les signaux que l'on envoie et l'on reçoit du système. Cela nous permettait de comprendre le comportement du système quand on appliquait des entrées différentes (sinus / créneaux / basse fréquence / haute fréquence)

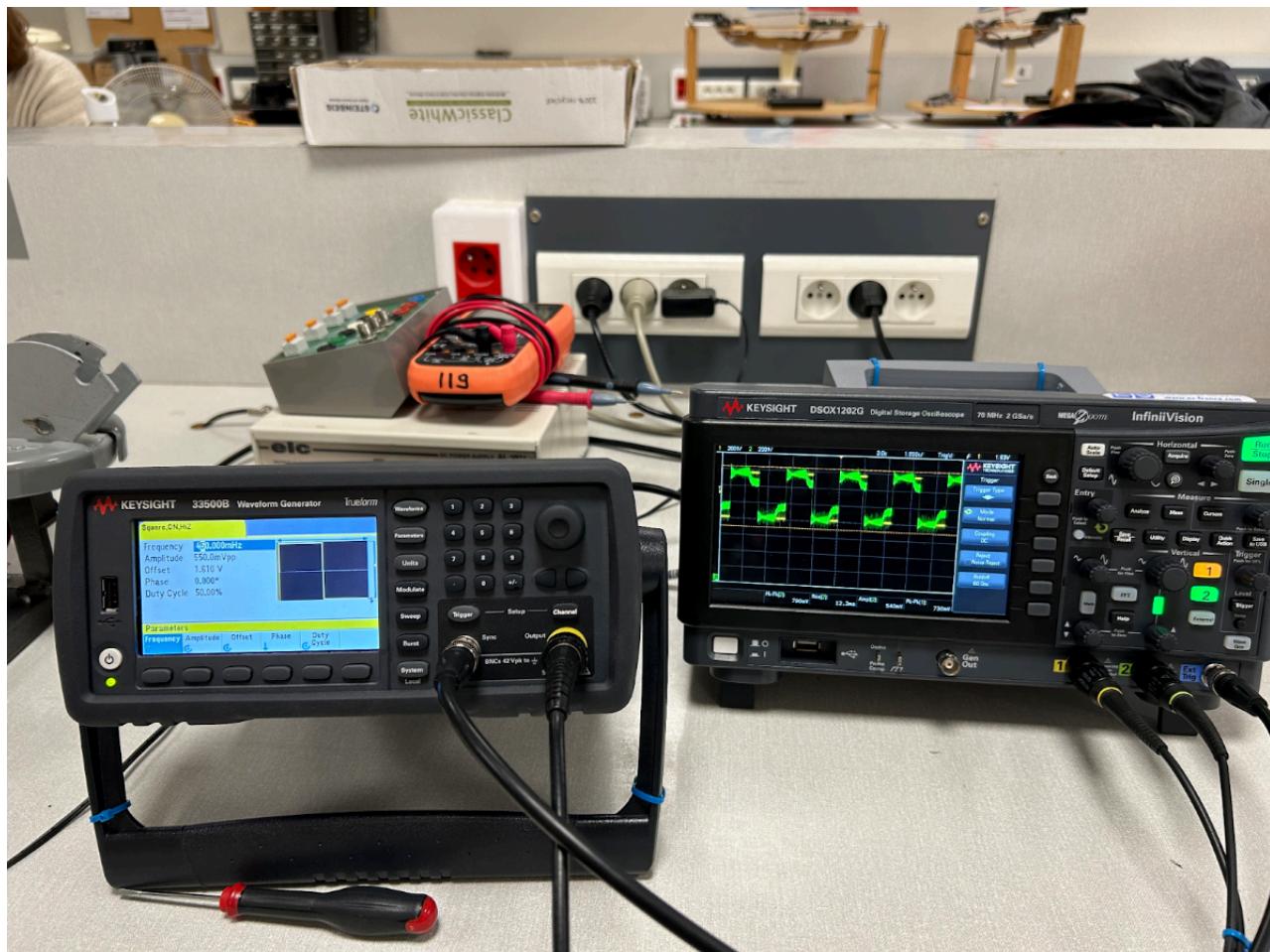


Figure 1.4.5.2 - Observations sur l'oscilloscope lorsque l'on a mis un signal créneaux en entrée, basse fréquence ($f=450\text{mHz}$). On observe le phénomène de dégradation de la sortie à cause du long temps que met le système pour répondre. Le système n'arrive plus à accélérer, et on voit une dégradation du signal en sortie vers la fin du créneau/carré.

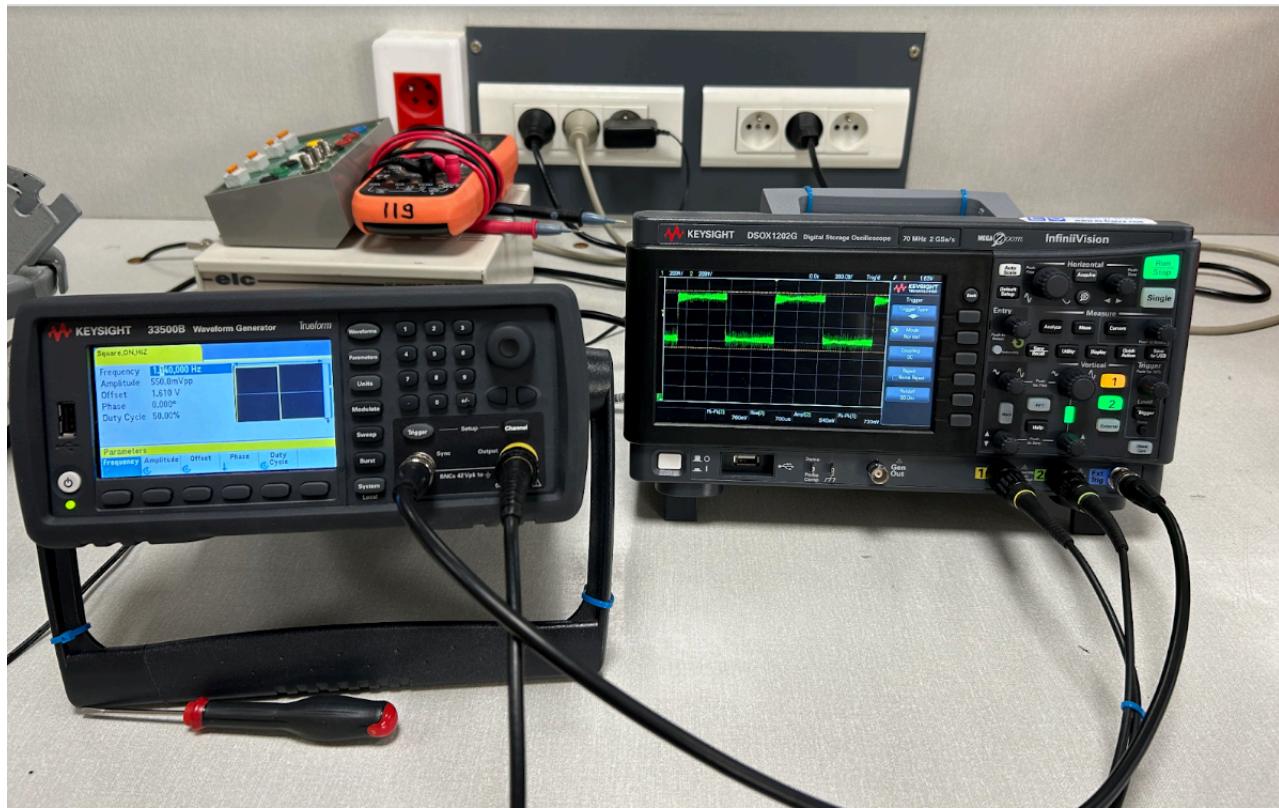


Figure 1.4.5.3 - Observations sur l'oscilloscope lorsque l'on a mis un signal créneaux en entrée, à $\approx 1\text{Hz}$, on voit que la sortie a à peu près la même amplitude que l'entrée, c'est-à-dire que l'on retrouve le gain de 0dB en basse fréquence.

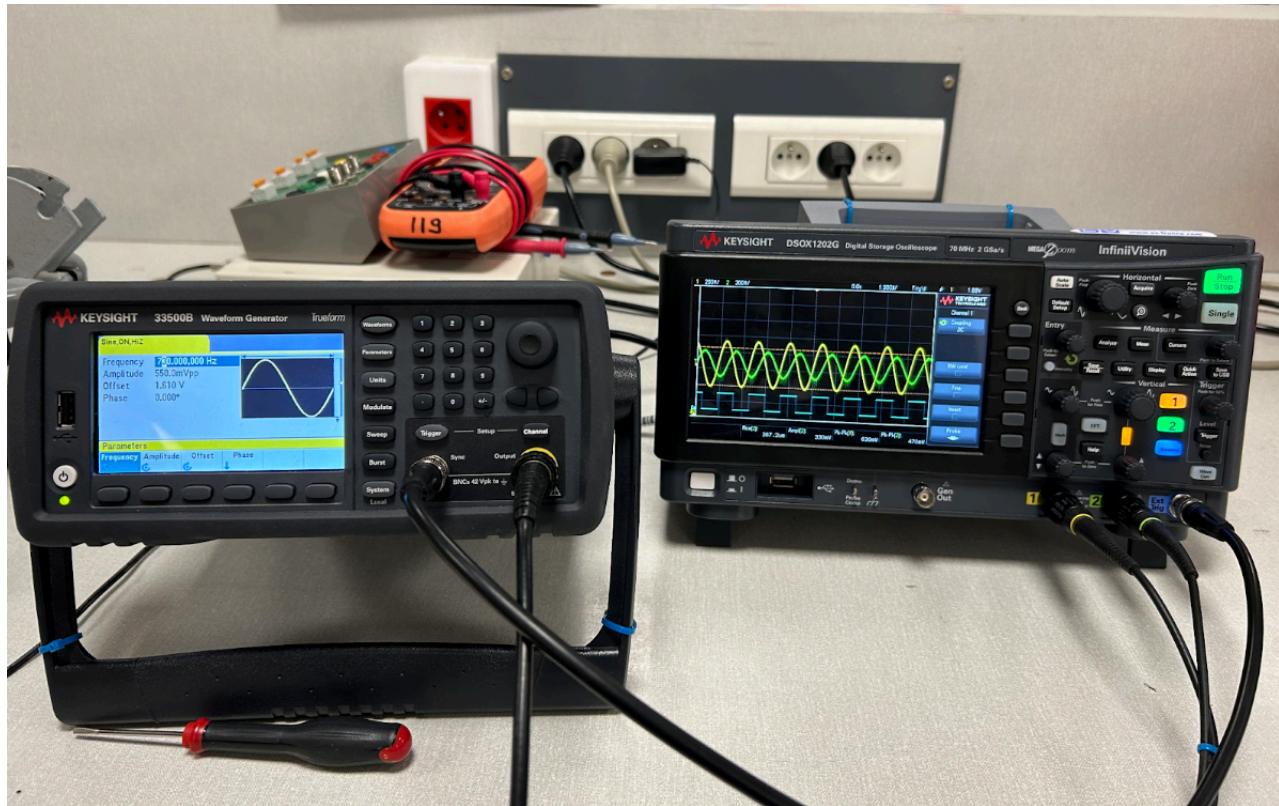


Figure 1.4.5.4 - Observations sur l'oscilloscope lorsque l'on a mis un sinus en entrée, sortie à 0,74 (presque 0,707...) lorsque l'on est près de la fréquence de coupure à 730Hz.
On n'a pas pris de photo lors de la fréquence de coupure exacte, mais on l'a mesurée à $f \approx$

750Hz. Pour tester la trottinette lors de l'implémentation en réel, on veut vérifier trois choses :

- La fréquence de coupure
- L'erreur statique nulle
- La marge de phase

Comme on l'a vu ci-dessus, on a trouvé une fréquence de coupure égale à ≈ 750 Hz. Dans nos simulations, on s'attendait à une fréquence de coupure à 500Hz en boucle fermée, donc on n'est pas loin. Cette différence peut être dûe à des petites différences de gain dans la modélisation par rapport au système réel. Le gain du système réel est donc à un facteur $\approx 1,5$ de gain par rapport au système modélisé, qui apparaît à cause des différences physiques des systèmes. Par rapport au comportement global du système, on constate que cette différence ne change pas beaucoup le comportement observé, car avec cette fréquence de coupure de même ordre de grandeur que celle prévue, on ne perd pas la stabilité du système.

Pour vérifier l'erreur statique nulle, on essaie de commander le système avec une consigne nulle qui maintient le système à un point de repos où la roue reste immobile. On arrive bien à avoir un tel comportement, et on peut donc constater que l'erreur statique est nulle car le système ne se met pas en route tout seul si on arrive à garder ce niveau de consigne. De plus, on voit en relativement basse fréquence, on arrive à avoir une sortie qui est l'image de l'entrée. En boucle fermée à basse fréquence, on sait que le comportement prévu est un gain en dB de 0dB, autrement dit un gain de 1. Cela veut dire que l'erreur est ≈ 0 car on obtient le même signal en sortie qu'en entrée.

Pour vérifier la marge de phase, il faut estimer la stabilité en boucle fermée, car on n'a pas de possibilité de mesurer la marge de phase à ce point. Vu que la marge de phase correspond à une grandeur physique en boucle ouverte, on ne peut pas le vérifier en réel, cependant on peut tester la stabilité en boucle fermée qui nous indique la marge de phase qui conduit à un tel comportement. On a déjà constaté que l'on arrive à maintenir une consigne nulle qui nous donne une sortie nulle, c'est-à-dire une roue immobile. En réglant une entrée créneaux, on peut regarder la réponse pour voir si on voit de dépassement lorsque le système va répondre à sa commande de couple. Sur la figure 1.4.5 on voit qu'il n'y a pas de dépassement lors de la réponse à un échelon (une période du signal créneaux). On a donc gardé le comportement stable souhaité avec ≈ 60 degrés de marge de phase, car on sait qu'avec 45 degrés de marge, la réponse à un échelon contient un premier dépassement de 5%. Vu que l'on n'a pas de dépassement, cela veut dire que l'on a bien >45 degrés, et probablement vers 60+ degrés vu que l'on n'a pas du tout de dépassement en sortie du système.

2 - Régulation de vitesse

On n'a pas eu le temps de passer à l'étape de la conception de la boucle de vitesse, mais on va expliquer l'idée et comment on aurait fait si on avait le temps de le faire pour la trottinette. La boucle de vitesse est une boucle extérieure qui va commander la consigne donnée en entrée de la boucle interne qui est celle que l'on a mise en œuvre préalablement. Pour l'étape de conception du correcteur de la boucle de vitesse, on va pouvoir simplifier la boucle interne. Cela est dû au fait qu'entre la sortie et l'entrée, puisque la régulation du courant en sortie du système est très rapide devant la régulation de vitesse nécessaire, on constate le comportement d'un simple bloc de gain qui permet de passer une consigne comprise entre [0;3.3] volts à un courant compris entre [-10;+10] ampères. De ce fait, on peut modéliser le système de régulation de vitesse de la trottinette de la manière suivante :

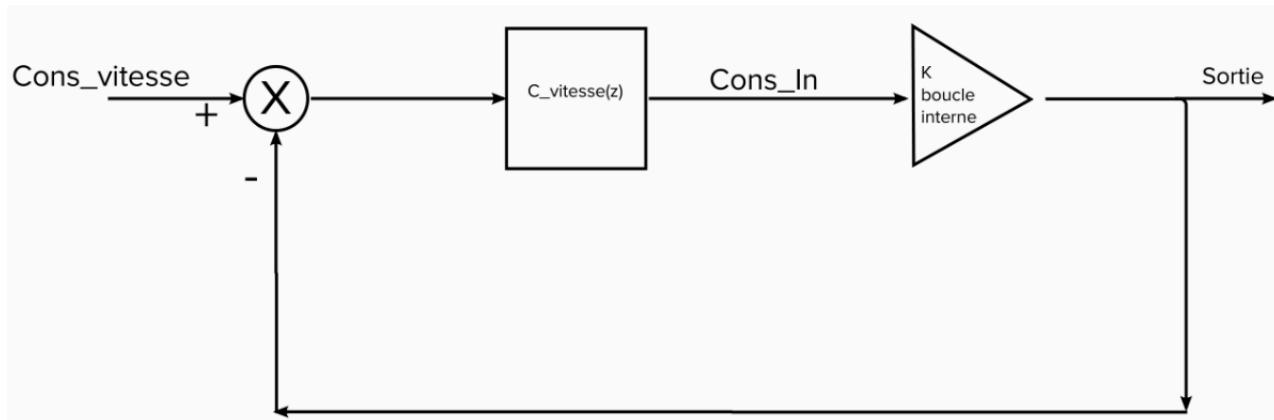


Figure 2.1.1 - Schéma bloc simplifié permettant de plus facilement déterminer la fonction de transfert du correcteur de vitesse de la boucle de régulation de vitesse.

3 - Autres évolutions du projet

Le projet de la trottinette peut évoluer encore plus, car il serait possible d'implémenter une courte phase d'initialisation lorsque l'utilisateur allume la trottinette, qui permet de configurer plus finement l'asservissement (numérique) du système. Pendant cette initialisation, le microcontrôleur STM32 pourrait exciter le système pendant quelques millisecondes afin d'analyser et calculer les grandeurs physiques de chaque trottinette en analysant la réponse à un échelon, permettant de mettre en évidence quelques caractéristiques comme le temps de réponse à 5%/2% / temps de montée etc. Vu qu'il y a de légères différences physiques entre chaque trottinette, comme par exemple les valeurs exactes des résistances du moteur ou les valeurs des inductances du système, on pourrait donc les calculer pour avoir des valeurs plus exactes par rapport à la vraie vie, permettant ainsi à concevoir un correcteur encore plus performant, avec des coefficients optimisés pour chaque trottinette.

Conclusion

Lors de ce BE, nous avons dû mobiliser de nombreuses connaissances en électronique et en automatique pour implémenter un correcteur pour une régulation de couple et de vitesse sur une trottinette électrique au travers d'un micro-controlleur. Nous avons commencé par l'analyse du système au travers de schémas et des documentations techniques des différents éléments nécessaires à son fonctionnement. Grâce à cette analyse et à cette compréhension du système, nous avons été capable d' extraire un schéma bloc de Laplace qui fut notre base de travail pour la recherche du bon correcteur. Après l'étude de stabilité du système en boucle ouverte et d'après les spécifications du cahier des charges nous avons décidé de prendre un correcteur proportionnel intégral qui nous permettait de répondre au cahier des charges et d'avoir un système stable. Après calculs et vérifications avec simulink, nous avons implémenté ce correcteur dans un STM32 et nous avons finalement testé sur le matériel à disposition. Ce BE a été compliqué à certains moments car il demandait d'avoir une excellente compréhension théorique de certains éléments mais nous a permis de mener une étude approfondie d'un système électronique et de mettre en lien des connaissances de plusieurs domaines pour répondre à une problématique, en suivant une démarche scientifique et professionnelle. Ce fut un BE très intéressant pour nous et très enrichissant, tant par l'apport de connaissance que par la mise en application de celles-ci.