Apports des outils numériques sur l'enseignement de l'automatique : travaux pratiques à distance à partir d'une solution numérique hardware à coût raisonnable

Culture Sciences de l'Ingénieur

Javier OJEDA

Édité le 23/04/2021

école normale supérieure paris—saclay

Cette ressource est issue d'une publication du numéro 104 de La Revue 3EI d'Avril 2021. *Javier Ojeda est Maître de Conférences au Département EEA (Électronique, Électricité et Automatique) de l'ENS Paris-Saclay.*

L'évolution des outils numériques aussi bien des logiciels que du matériel permet d'envisager l'enseignement de l'automatique de manière plus illustrative et participative. Illustrative, car des outils numériques, le plus souvent libres, sont disponibles pour les étudiants afin de mettre en image des notions parfois complexes de l'automatique et participative, car au travers d'ateliers réalisés par les étudiants, ceux-ci peuvent appréhender des notions principales ou annexes par leurs simulations et expérimentations. Par ailleurs, participative, par le fait que les outils numériques permettent des échanges enseignant-apprenant en ligne via un site web, des plateformes moodle, slack, etc.

C'est au travers d'une série de trois ressources que l'évolution des outils numériques est abordée du point de vue théorique, logiciel, hardware et pratique. Après les deux premiers volets [1] [2], cette troisième ressource aborde la notion d'enseignement pratique à distance en utilisant un hardware permettant de traiter des notions liées aux asservissements numériques.

1 – Introduction

L'évolution des différents supports numériques hardwares permet aujourd'hui d'envisager des manipulations sur les asservissements numériques à bas coût, mais également en autonomie. Les différents hardwares permettent soit de générer une connexion à distance [3] soit d'être suffisamment portatif pour que les apprenants puissent faire les manipulations depuis seulement un ordinateur sans générateurs de fonction, oscilloscopes, sondes, etc.

Dans cette ressource, nous allons traiter d'un asservissement à temps discret en dimensionnement un régulateur du type RST [4], [5], i.e. un correcteur polynomial sur un système analogique du second ordre. L'objectif est de présenter les différentes étapes du dimensionnement. Mais également de mettre en lumière l'intérêt de directement considérer un asservissement numérique tel quel plutôt que de chercher à numériser un correcteur dimensionné en temps continu.

Cette ressource est organisé suivant 3 paragraphes. Le premier présente les différentes solutions matérielles permettant d'envisager la réalisation de travaux pratiques en automatique à temps discret. Ces solutions seront analysées sous le prisme de l'enseignement et non selon des critères très généraux. Le second paragraphe traite du dimensionnement du régulateur RST à l'aide d'outils numériques facilitant ce dimensionnement. Le dernier paragraphe traite quant à lui de l'implémentation du correcteur et de sa caractérisation vis-à-vis de son jumeau numérique. L'un

des points forts des asservissements numériques étant de parfaitement contrôler le système et ainsi arriver à un degré de maitrise important en termes de performances.

2 – Les différents types de Hardware permettant de faire de l'enseignement à distance

Dans cette section, nous n'allons pas faire une liste exhaustive de tous les hardwares disponibles, mais nous essayons de distinguer les grandes familles et ce qu'elles peuvent apporter à l'enseignement de l'automatique à distance.

2.1 – Les spécifications attendues pour l'enseignement

En tout premier lieu, cet article traite de l'enseignement de l'automatique. Ainsi, la première spécification attendue est la simplicité de mise en œuvre. De préférence, les fonctions de conversion CAN, CNA doivent être transparentes du point de vue du code et faciles à appréhender. De même pour le typage et le codage des nombres.

Deuxièmement, la solution hardware doit permettre une communication entre un ordinateur et la manipulation via par exemple une liaison série usb, afin de pouvoir obtenir une manipulation autonome entre la board, le système et l'ordinateur. Dans ce cas, pas besoin de générateur de fonction ou d'oscilloscope, les étudiants pouvant ainsi effectuer les manipulations depuis chez eux.

Pour finir, les solutions doivent garantir un temps entre deux échantillonnages constants. Cela va imposer une fréquence d'horloge du processeur très grande devant la fréquence de conversion des convertisseurs.

2.2 – Les grandes familles d'hardware pour l'enseignement

Les grandes familles détaillées ci-dessous sont vues sous le prisme de l'enseignement de l'automatique.

- Les nano-ordinateurs type Raspberry pi. Les dernières versions du Raspberry pi 4 permettent d'avoir une plateforme puissante et polyvalente. Connectivité internet, GPIO, communications i2c, spi, etc. Le tout géré par un os choisi parmi plusieurs aux caractéristiques complémentaires. De posséder un os va rendre cette solution très flexible et simple du point de vue de la programmation, cependant, celui-ci va « cacher » la couche matérielle, ainsi il va être très difficile d'accéder, par exemple, aux différents timers et registres. Par conséquence, si le temps caractéristique est plutôt court (inférieure à la seconde), il va être difficile de garantir un échantillonnage à temps constant.
- Les cartes de développement basiques type Arduino. Ces cartes relativement bas coût possèdent de nombreux avantages (convertisseurs intégrés, langage de programmation simplifié, communication série) qui en font les références dès que l'on réalise un asservissement numérique. Nous les retrouvons dans les systèmes robotiques, domotiques, etc. L'inconvénient est que les algorithmes implémentés devront être relativement courts et simples.
- Les cartes de développement avancé type cartes microcontrôleurs STM32 [6]. Ce sont des cartes microcontrôleur évoluées possédant des caractéristiques remarquables en termes de complexité d'algorithmes pouvant être implémentés et de vitesse d'exécution. Néanmoins, ces solutions nécessitent une connaissance importante des architectures numériques et d'un langage de programmation proche de la machine.

- Les cartes programmables de type FPGA [7]. Elles possèdent des caractéristiques complémentaires aux cartes DSP. Elles permettent d'atteindre des performances en temps de calcul très importantes, mais pour des algorithmes peu complexes. De plus, elles nécessitent un codage en langage VHDL.

Bien entendu toutes ces différentes cartes peuvent convenir pour peu que le système soit à dynamique lente. Pour les systèmes à dynamique moyenne (autour de 10 ms) les cartes de développement basiques feront parfaitement l'affaire et leurs simplicités de programmation sont très appréciables. Enfin, pour des systèmes rapides, il sera nécessaire de passer par des cartes de développement avancées ou des cartes programmables FPGA qui vont avoir l'inconvénient de nécessiter un temps de développement du code plus important.

La notion de complexité de programmation est à prendre avec des pincettes, car de nombreuses solutions permettent de simplifier la programmation des cartes évoluées par des toolboxes dédiées.

2.3 – Présentation de la carte Pyboard en Micropython

Les précédents ateliers ont été réalisés en langage Python. Par cohérence pédagogique, il était nécessaire de chercher une carte de développement rapide, mais programmable en langage proche du Python, Micropython [8]. Ce langage est une adaptation et une optimisation du langage Python aux microcontrôleurs. Il permet donc de générer un environnement Python sur la board sans nécessité d'une installation sur l'ordinateur de pilotage. Parmi les boards compatibles, la Pyboard, **Erreur ! Source du renvoi introuvable.**, possède des caractéristiques idéales pour réaliser un asservissement numérique ; Convertisseurs ADC et DAC, fréquence du processeur élevée et une unité de calcul à virgule flottante.

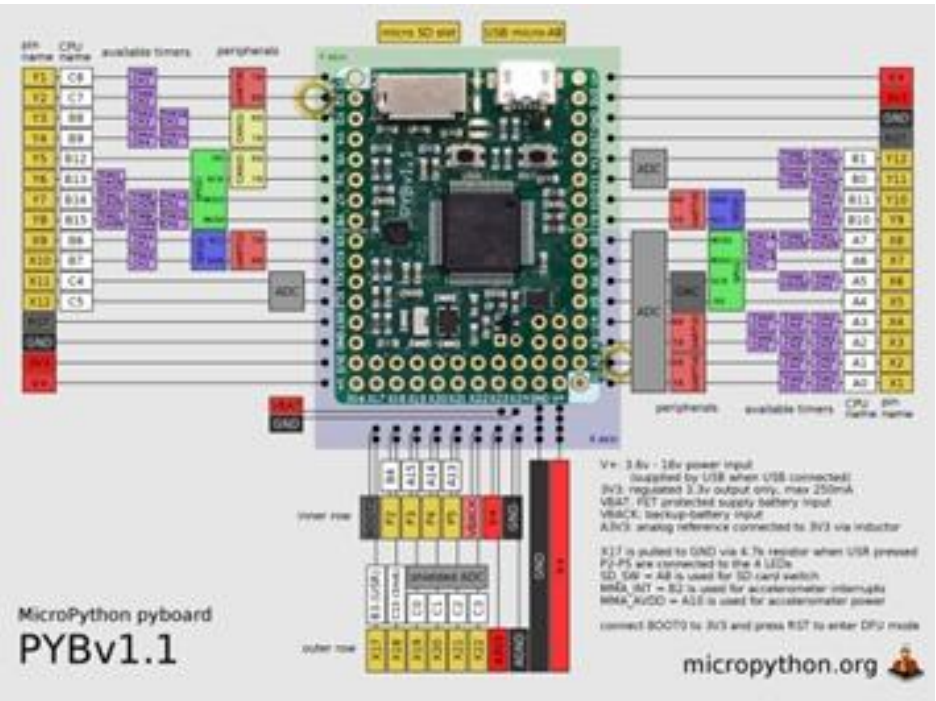

Figure 1 : Synopsis de la carte PYBOARD v1.1, source [8]

Les caractéristiques utiles pour l'article de la carte sont résumées ci-dessous :

- Microprocesseur Cortex M4F - STM32F405RGT6, ARM 32 bits - 168Mhz
- Ram 192 Ko, Flash 1Mo
- 16 ADC 12 bits et 2 DAC 12 bits
- Unité de calcul à virgule flottante

De plus, cette carte permet via une liaison série d'avoir accès aux variables internes en cours de traitement pour une visualisation externe.

3 – Dimensionnement du correcteur numérique

3.1 – Présentation du système étudié

L'étude d'un asservissement numérique passera par une étape de modélisation du système et particulièrement de la numérisation du système temps continu en un système temps discret. Cette étape consiste à calculer la transformée en z ou z^{-1}, donc calculer le système équivalent numérique de l'ensemble système et convertisseur analogique-numérique. Dans la grande majorité des cas, ce convertisseur analogique-numérique est modélisé par un bloqueur d'ordre 0, BOZ. Cette étape est relativement difficile et peut amener à des calculs parfois complexes. Ainsi, ce sont les systèmes du premier ordre qui sont le plus largement utilisés pour enseigner les asservissements numériques. Ce choix est judicieux, car il permet d'aborder tous les points des différents programmes ; stabilité, précision, etc. Pour être plus général, dans cet article, nous avons fait le choix d'étudier un système du second ordre pour présenter une méthode de numérisation du système à temps continu à l'aide de la toolbox control de Python. Ce système H(p) est un système du second ordre défini par :

$$H(p) = \frac{K_0}{1 + 2m\frac{p}{\omega_0} + \left(\frac{p}{\omega_0}\right)^2}$$

Le coefficient d'amortissement vaut m=0.707, la pulsation ω_0=2.π.110Hz et un gain statique K_0=1.586. Le système est réalisé par un filtre de Sallen-Key du second ordre dont le schéma électrique est donné sur la **Erreur ! Source du renvoi introuvable.**.

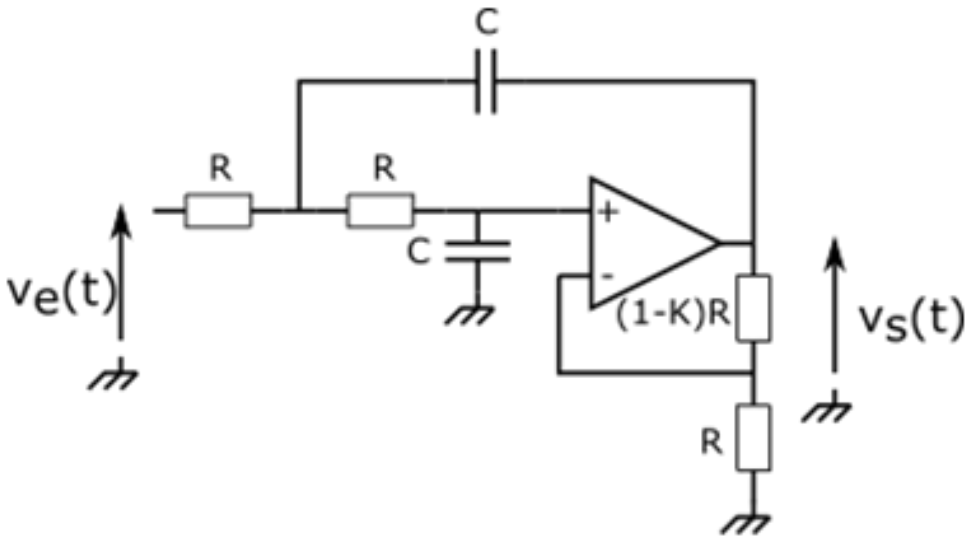

Figure 2 : Schéma électrique du système étudié (R=33kΩ, C=47nF, (1-K)R= 18kΩ, ALI =TSV321)

L'amplificateur opérationnel utilisé pour réaliser le système est un TSV321 qui est un AOP basse tension et asymétrique. Il peut ainsi être alimenté en 3,3V directement par la Pyboard sans alimentation externe. La carte réalisée à partir du schéma précédent est présentée sur la **Erreur ! Source du renvoi introuvable.**.

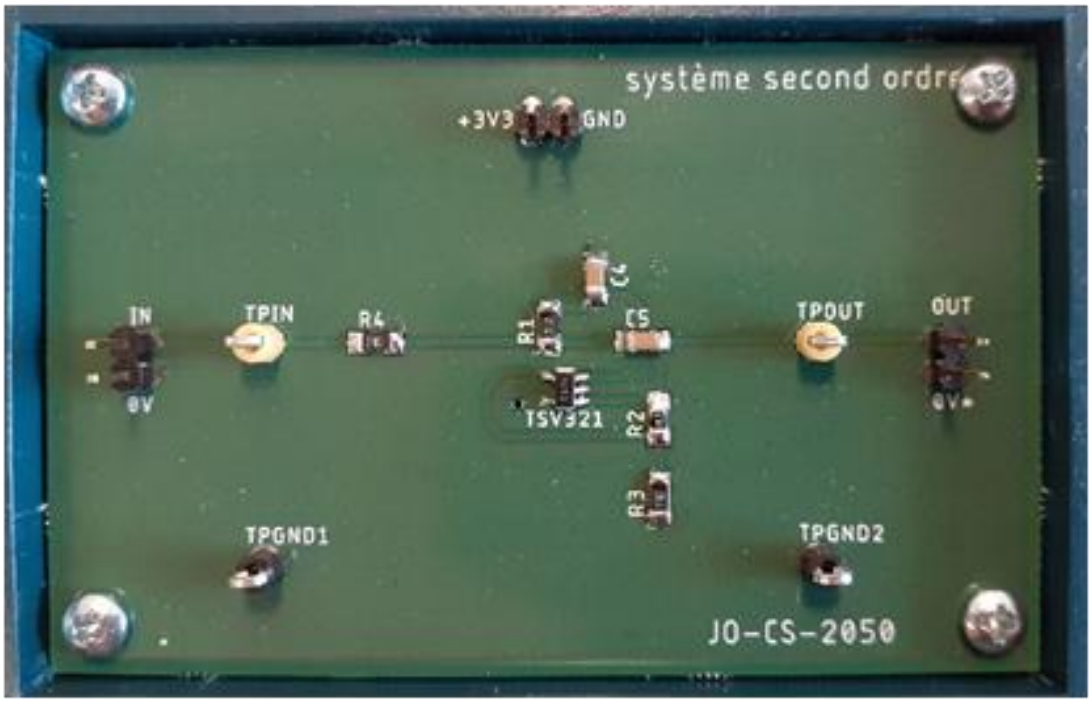

Figure 3 : Carte simulant un système du second ordre à partir d'un TSV321

Le bloqueur d'ordre 0 avec une période d'échantillonnage T_{ech}=1ms est réalisé par le convertisseur analogique-numérique de la Pyboard et il sera modélisé par :

$$BOZ(p) = \frac{1 - e^{-pT_{ech}}}{p}$$

Le calcul du correcteur numérique équivalent passe par le calcul de la transformée en z ou z^{-1} :

$$\overline{BOZ.H}(z^{-1}) \quad = \quad TZ\left[TL^{-1}[BOZ(p)H(p)]\right]$$

C'est bien la détermination de cette transformée qui inclut le calcul d'une transformée inverse de Laplace puis d'une transformée en z qui peut devenir problématique pour les apprenants, voire impossible pour des systèmes complexes. La toolbox control de python présentée dans l'article [2] permet de calculer cette fonction de transfert à partir de la fonction c2d.

```
T_ECH = 1e-3 # Période d'échantillonnage en secondes

# Transformation du système analogique en système numérique par BOZ

SYSTEME_NUM =  ctl.c2d(SYSTEME_TEMPS_CONTINU, T_ECH, method='zoh')
```

Le système équivalent numérisé est alors représenté sous la forme d'une fonction rationnelle et notamment pour le système étudié :

$$\overline{BOZ.H}(z^{-1}) = z^{-d}\frac{B(z^{-1})}{A(z^{-1})} = z^{-1}\frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Les diagramme du Bode du système temps continu et du système équivalent à temps discret est représenté sur la **Erreur ! Source du renvoi introuvable.**.

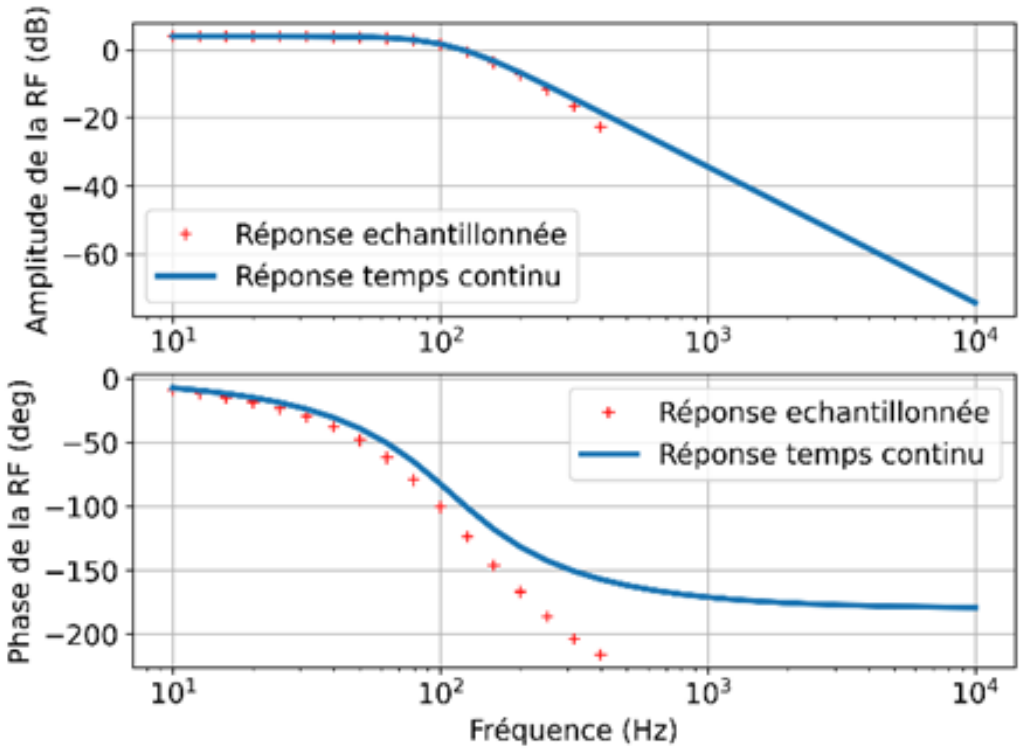

Figure 4 : Représentation en fréquence RF du système temps continu et du système équivalent temps discret

Nous avons délibérément choisi la période d'échantillonnage proche du temps caractéristique du système pour mettre en avant les effets de la numérisation. Sur la réponse en fréquence, nous voyons la modification du gain et de la phase lorsque la fréquence s'approche de la demi-fréquence d'échantillonnage dû à la contribution du bloqueur d'ordre 0.

3.2 - Choix du correcteur numérique

Le dimensionnement de la période d'échantillonnage par rapport à la constante de temps du système est crucial pour un asservissement numérique. C'est ce dimensionnement qui va aiguiller sur le choix de deux grandes familles de correcteur :

- La période d'échantillonnage est très grande (au moins deux ordres de grandeur) devant la période caractéristique du système. Dans ce cas de figure, il est envisageable de faire comme si le système était à temps continu et donc de calculer un correcteur temps continu qui sera ensuite numérisé par une méthode de numérisation (Euler, bilinéaire, etc.) [9].
- L'effet de la période d'échantillonnage ne peut être négligé. Dans ce cas, il n'est pas possible d'utiliser la méthode précédente. Il faut alors étudier le système équivalent numérique dans son ensemble. Dans cet article, nous allons nous placer dans ce cas. La période d'échantillonnage est de 1ms, T_{ech} = 1ms. Soit moins d'une décade plus faible que la période caractéristique du système.

Il existe plusieurs grandes familles de correcteurs numériques. Deux des plus importants types de correcteurs numériques sont la correction par retour d'état et observateur et également, les correcteurs dit polynomiaux. Nous allons dans cet article nous intéresser à un type de correcteur polynomial qu'est le correcteur RST. Son schéma bloc numérique est représenté sur la **Erreur ! Source du renvoi introuvable.**.

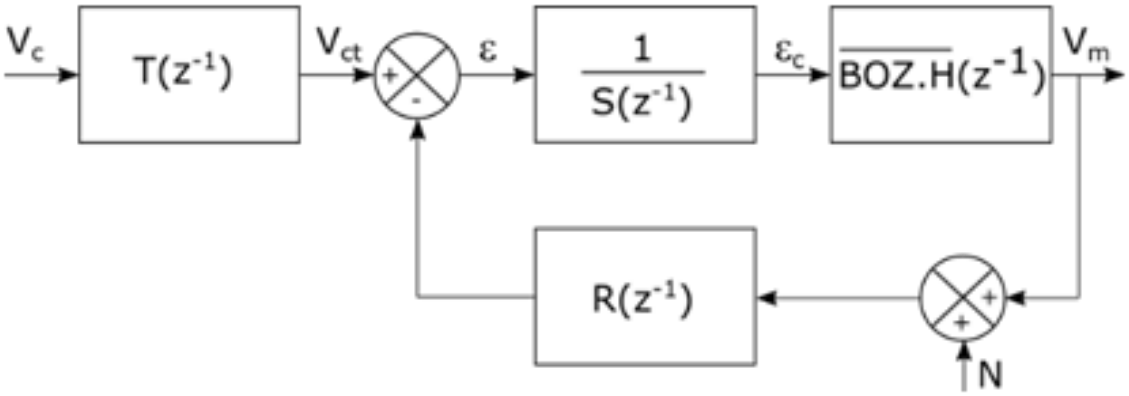

Figure 5 : Schéma bloc numérique du correcteur RST avec une perturbation en sortie

V_c représente l'entrée du système en asservissement et N, l'entrée de perturbation considérée en régulation. D'autres perturbations peuvent être prises en compte de la même manière. Le correcteur RST est un correcteur numérique à plusieurs degrés de liberté. Le dimensionnement des trois polynômes R, S et T permettent d'obtenir :

- Un placement de pôle en boucle fermée
- Une dynamique différente pour la régulation et l'asservissement
- La possibilité d'obtenir un correcteur à réponse pile
- Pas de restrictions quant aux retards dans la chaine directe ou des pôles / zéros instables

Nous pouvons alors définir deux fonctions de transfert en asservissement :

$$\left.\frac{V_m}{V_c}\right|_{N=0} = \frac{z^{-d}B(z^{-1})T(z^{-1})}{A(z^{-1})S(z^{-1}) + z^{-d}B(z^{-1})R(z^{-1})}$$

Et en régulation :

$$\left.\frac{V_m}{N}\right|_{V_c=0} = \frac{z^{-d}B(z^{-1})R(z^{-1})}{A(z^{-1})S(z^{-1}) + z^{-d}B(z^{-1})R(z^{-1})}$$

Pour contrôler la dynamique en boucle fermée, le dimensionnement du contrôleur RST se fait par placement de pôles et donc par une identification du dénominateur de la fonction de transfert en boucle fermée. Nous notons l'équation caractéristique souhaitée en boucle fermée $P_{BF}(z^{-1})$, ainsi, l'équation à résoudre est :

$$A(z^{-1})S(z^{-1}) + z^{-d}B(z^{-1})R(z^{-1}) = P_{BF}(z^{-1})$$

Cette équation est nommée équation de Bézout ou diophantienne. Cette équation dans le cadre général n'est pas triviale à résoudre, cependant, dans le cadre d'un asservissement numérique nous pouvons contraindre cette équation :

- Il est souhaitable d'avoir des polynômes R et S de degrés minimaux pour une implémentation et une utilisation des ressources matérielles optimisées.
- Le polynôme P_{BF} est composé d'une partie dominante déterminée par le cahier des charges et d'une partie auxiliaire de plus haute fréquence que la partie dominante et de degré le plus faible possible.

Ainsi, pour obtenir une solution unique et respectant les principes cités précédemment, il faut respecter les inégalités suivantes.

$$\begin{aligned} n_p &\leq n_A + n_B + d - 1 \\ n_s &= n_B + d - 1 \\ n_r &= n_A - 1 \end{aligned}$$

Avec n_S et n_R, les degrés des polynômes S et R, n_A et n_B, les degrés des polynômes A et B, d le retard du système et n_P le degré du polynôme de P_{BF}. Ces inéquations appliquées au système de cet article donnent :

$$\begin{aligned} n_p &\leq 3 \\ n_s &= 1 \\ n_r &= 1 \end{aligned}$$

Et donc deux polynômes R et S :

$$\begin{aligned} S(z^{-1}) &= 1 + s_1 z^{-1} \\ R(z^{-1}) &= r_0 + r_1 z^{-1} \end{aligned}$$

Il est à remarquer que selon les caractéristiques du système et du cahier des charges, des parties fixes peuvent être ajoutées. Nous pouvons citer la partie fixe qui est ajoutée à S afin d'obtenir une réjection des perturbations :

$$S(z^{-1}) = (1 - z^{-1})^k S^*(z^{-1})$$

À chaque degré de k, le système rejette un niveau de perturbation supplémentaire, échelon, rampe, etc. Il faut alors modifier les inégalités précédentes pour prendre en compte les parties fixes.

Le cahier des charges dans le cadre de cet article est choisi pour obtenir une partie dominante du polynôme en boucle fermée correspondant à un système du second ordre possédant le même amortissement qu'en boucle ouverte, mais une dynamique deux fois plus rapide. Le pôle auxiliaire est quant à lui un pôle du premier ordre avec une constante de temps égale à la période d'échantillonnage et par voie de conséquence plus grande que le temps caractéristique des pôles dominants. Le polynôme ainsi dimensionné est défini par :

$$P_{BF}(z^{-1}) = 1 + p_1 z^{-1} + p_2 z^{-2} + p_3 z^{-3}$$

La résolution de l'équation diophantienne revient à la résolution d'un système de 3 équations à 3 inconnues obtenues par identification. Dans notre cas :

$$\begin{pmatrix} 1 & b_0 & 0 \\ a_1 & b_1 & b_0 \\ a_2 & 0 & b_1 \end{pmatrix} \begin{pmatrix} s_1 \\ r_0 \\ r_1 \end{pmatrix} = \begin{pmatrix} p_1 - a_1 \\ p_2 - a_2 \\ p_3 \end{pmatrix}$$

Ce qui conduit à :

$$\begin{pmatrix} s_1 \\ r_0 \\ r_1 \end{pmatrix} = \begin{pmatrix} 0.645 \\ 4.54 \\ -0.979 \end{pmatrix}$$

Pour valider la dynamique du système, une réponse indicielle est réalisée avec un polynôme T de degré 0 permettant d'avoir un gain statique unitaire de l'ensemble. Cette réponse est représentée sur la **Erreur ! Source du renvoi introuvable.**.

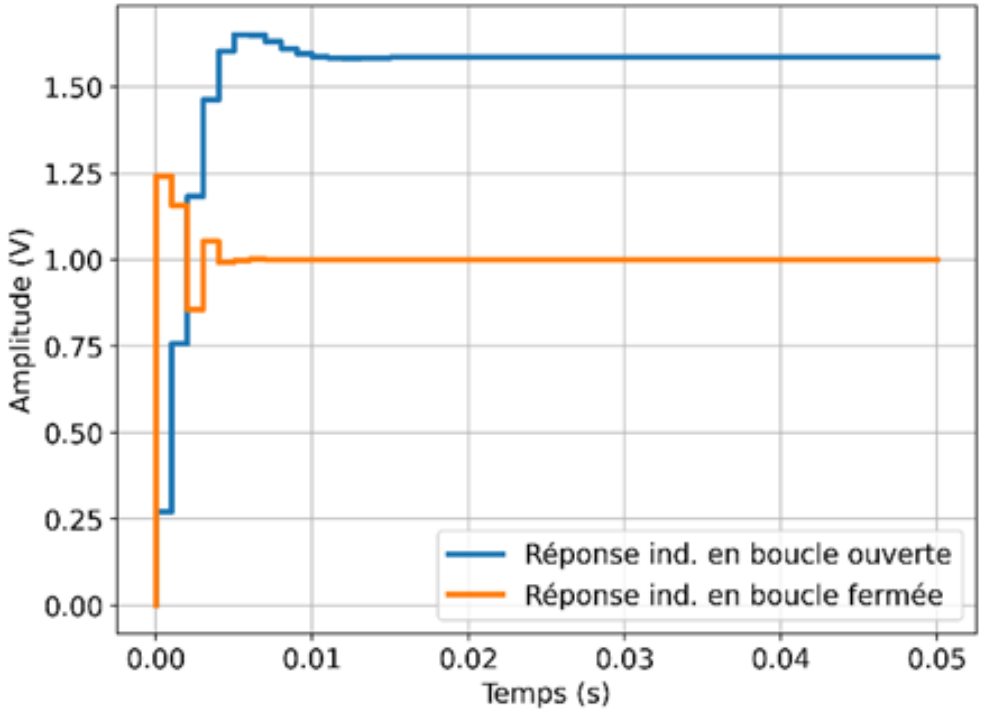

Figure 6 : Comparaison de la réponse indicielle simulée en boucle ouverte et boucle fermée

En comparant les temps de réponse à 5% des réponses indicielles en boucle ouverte et boucle fermée, nous retrouvons bien une amélioration par un facteur deux de la dynamique. Une vérification des pôles de la fonction de transfert en boucle fermée confirme le bon placement des pôles par rapport au cahier des charges.

L'écriture de la fonction de transfert en boucle fermée pour l'asservissement s'écrit :

$$\left.\frac{V_m}{V_c}\right|_{N=0} = \frac{z^{-d}B(z^{-1})T(z^{-1})}{A(z^{-1})S(z^{-1}) + z^{-d}B(z^{-1})R(z^{-1})} = \frac{z^{-d}B(z^{-1})T(z^{-1})}{P_{BF}(z^{-1})}$$

Le dimensionnement du polynôme T permet d'obtenir une réponse du système du type correcteur à réponse pile. Ici, nous allons dimensionner le polynôme T pour avoir une réponse en 4 tops d'horloge. En comptant, un top d'horloge dû au retard (d=1), il faut une fonction de transfert en boucle fermée :

$$\left.\frac{V_m}{V_c}\right|_{N=0} = z^{-d}\frac{1}{3}(1 + z^{-1} + z^{-2})$$

Puisqu'il n'y a pas de partie instable dans P_{BF} et dans B, nous pouvons identifier les deux équations pour obtenir la valeur de T :

$$T(z^{-1}) = \frac{\frac{1}{3}(1 + z^{-1} + z^{-2})P_{BF}(z^{-1})}{B(z^{-1})}$$

La réponse indicielle ainsi obtenue est représentée sur la **Erreur ! Source du renvoi introuvable.**.

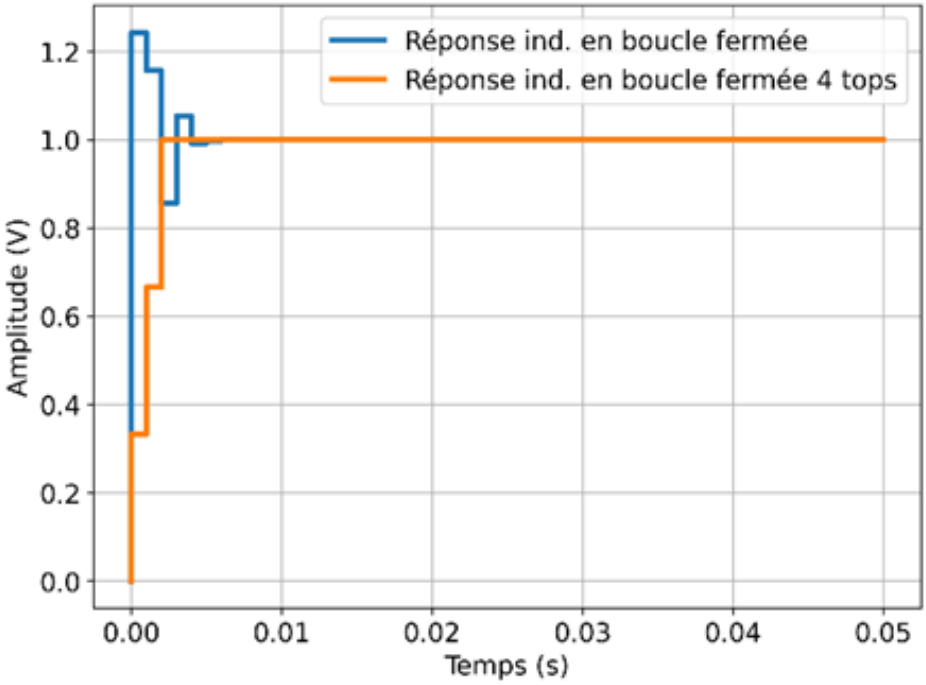

Figure 7 : Comparaison du correcteur sans polynôme T et avec une réponse en 4 tops d'horloge (la fonction de la toolbox control qui calcule la réponse indicielle omet le premier échantillon qui est nul)

Avec le dimensionnement du polynôme T, nous obtenons une réponse en 4 tops d'horloge (le premier top n'est pas représenté sur le graphique) comme demandé par le cahier des charges.

4 – Réalisation expérimentale de l'asservissement numérique

4.1 - Présentation de la manipulation

La manipulation est composée d'une Pyboard et d'une maquette électronique simulant un système physique, **Erreur ! Source du renvoi introuvable.**.

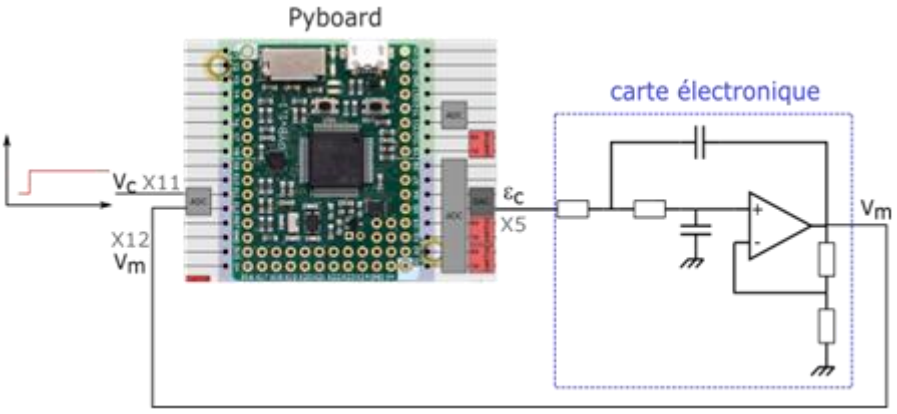

Figure 8 : Schéma de câblage de la manipulation. Les pins 3.3V et 0V de la Pyboard et de la carte sont reliées.

La Pyboard permet de gérer :

- La numérisation des signaux via les ADC en X11 (Consigne) et X12 (Mesure)
- L'implémentation de l'équation de récurrence du correcteur RST
- La conversion numérique de l'erreur corrigée via le DAC en X5
- Dans le cas d'une manipulation à distance, l'interface usb pourra être utilisée en invite de commande REPL (read-eval-print loop) pour la récupération des données et la visualisation sur un ordinateur.

La maquette électronique est quant à elle alimentée par le 3.3V de la Pyboard.

4.2 - Implémentation de l'asservissement numérique

4.2.1 - Gestion de l'échantillonnage

Garantir un temps d'échantillonnage constant entre deux échantillons n'est pas une tâche si aisée. Il y a plusieurs définitions possibles de l'échantillonnage :

- Un temps constant entre deux appels à une conversion ADC
- Un temps constant entre deux appels à une conversion DAC
- Un temps constant entre l'exécution d'une conversion ADC, algorithme de correction et d'une conversion DAC.

Si la fréquence du processeur est bien dimensionnée par rapport à la vitesse de conversion des convertisseurs et au temps d'exécution de l'algorithme, les trois définitions précédentes sont quasi équivalentes. Le choix d'une période d'échantillonnage de 1ms dans notre cas permet de se placer dans ce cas. Le temps de conversion de l'ADC ou du DAC est de l'ordre de 50µs pour la Pyboard et l'algorithme du régulateur possède une complexité très faible. Ainsi, nous pouvons utiliser une méthode simple pour gérer l'échantillonnage. Une fonction *correction()* est définie pour gérer les conversions ADC, le calcul de l'erreur corrigée et la conversion DAC. Alors, le diagramme du programme est représenté sur la **Erreur ! Source du renvoi introuvable.** :

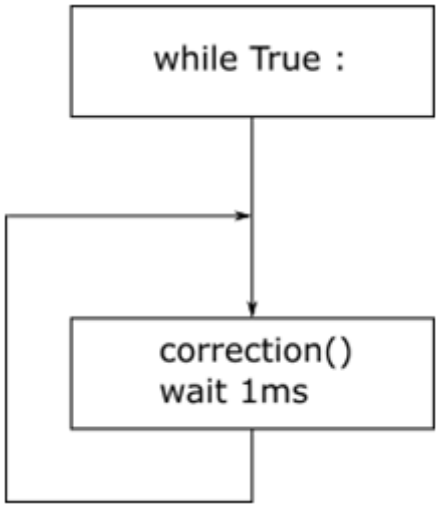

Figure 9 : Synopsis du programme gérant l'asservissement

Cette solution est simple, mais elle reste fortement dépendante du processeur utilisé pour effectuer les opérations. La Pyboard propose deux autres solutions pour obtenir une implémentation plus correcte :

- L'utilisation d'un timer : pyb.Timer. À chaque fin de boucle du Timer un appel à une fonction est passé Timer.callback(algorithme). Ainsi, le lancement de la fonction algorithme se fera toujours avec le même écart temporel. Attention de la fonction définie ne peut contenir d'attribution de mémoire et donc interdit les calculs sur les floatants.
- Une solution intermédiaire entre les deux précédemment citées. Le timer gère l'appel d'une fonction d'incrémentation d'une variable click. Le programme principal est composé d'une boucle while dans laquelle, nous venons comparer la valeur de click à une valeur définie par count. Si la valeur de click vaut la valeur de count, la fonction *correction()* est effectuée. Par cette méthode, nous nous assurons que le temps entre deux lancements de la fonction *correction()* est constant est égale à 1ms. Le schéma synoptique est représenté sur la **Erreur ! Source du renvoi introuvable.**.

C’est cette dernière version qui a été codée sur la Pyboard.

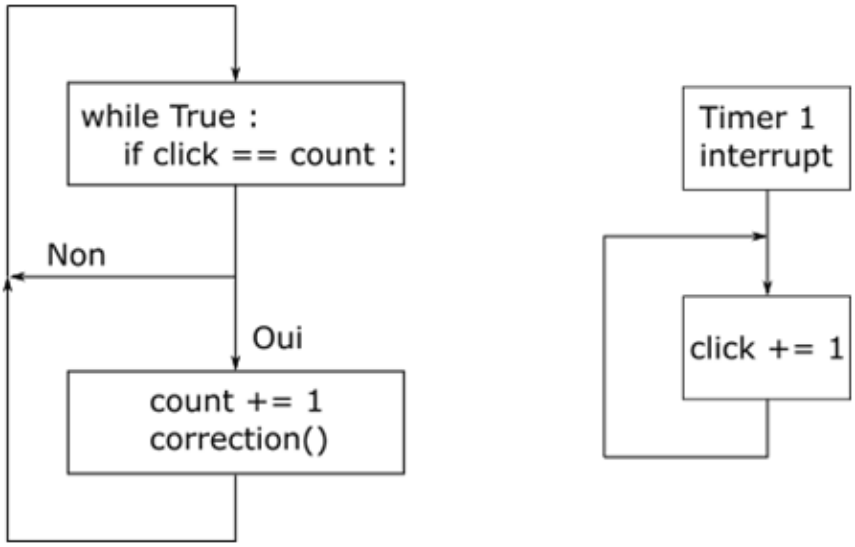

Figure 10 : Schéma de l'algorithme proposé pour le contrôle numérique (une fonction de remise à 0 de count n’est pas représentée sur le schéma)

4.2.2 - Comparaison entre le système expérimental et le système simulé

Dans les essais suivants, le polynôme sera pris constant et unitaire. Les polynômes R et S, sont ceux calculés dans les sections précédentes. La période d’échantillonnage est définie à 1ms. Sur la **Erreur ! Source du renvoi introuvable.**, un essai indiciel a été effectué sur la carte. L’échelon de consigne est compris entre 1V et 2V. Le choix des amplitudes de la consigne doit permettre d’éviter les saturations de la Pyboard et de la carte électronique.

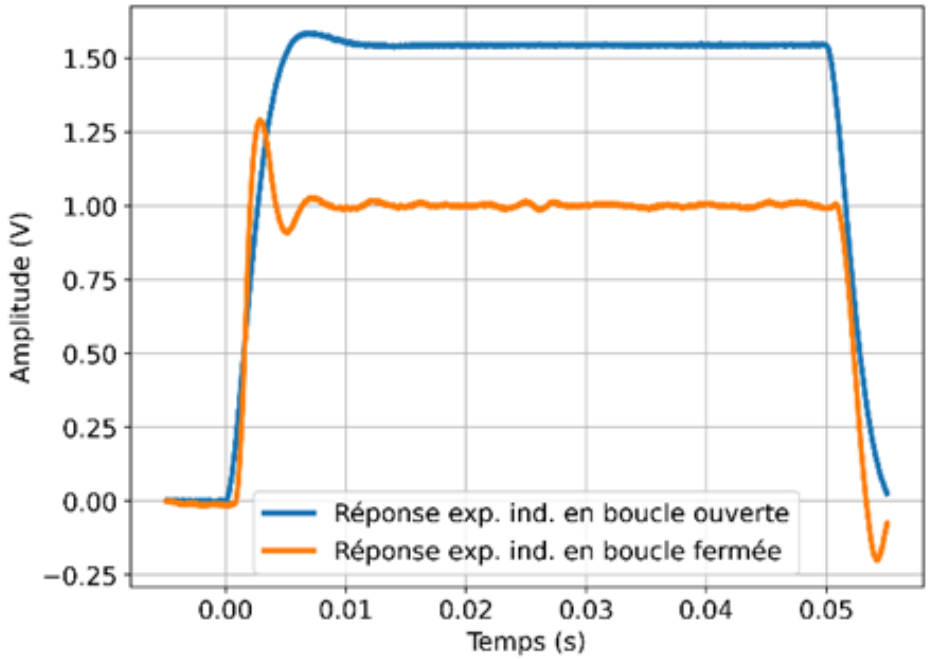

Figure 11 : Comparaison expérimentale de la réponse indicielle de la boucle ouverte et boucle fermée

La réponse obtenue en boucle fermée est celle attendue par le cahier des charges. Sur la **Erreur ! Source du renvoi introuvable.**, la réponse indicielle des deux systèmes permet de valider le cahier des charges et de conclure quant à la fidélité de la simulation numérique et de sa modélisation vis-à-vis du système réel.

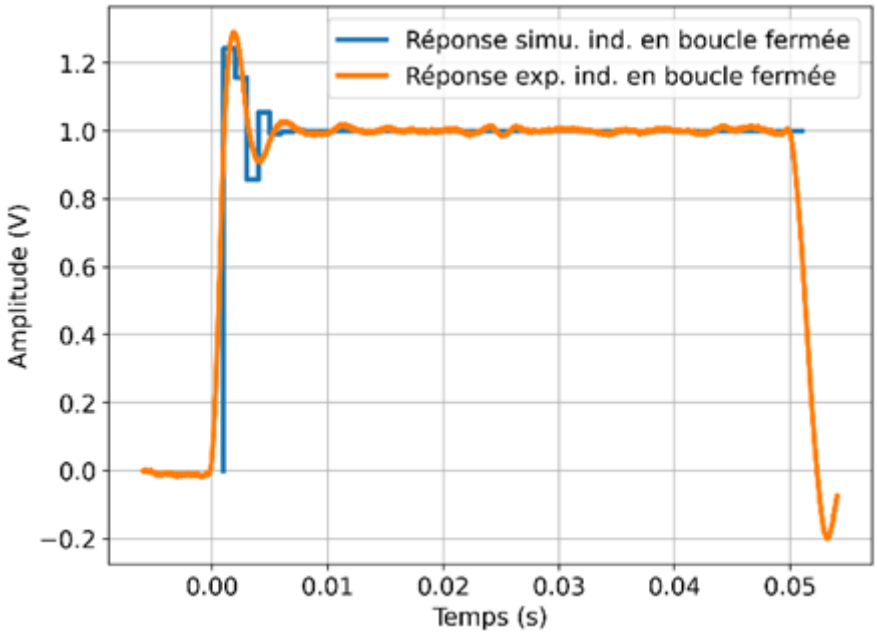

Figure 12 : Comparaison de la réponse indicielle en boucle fermée pour le système expérimental et le système simulé

Sur la **Erreur ! Source du renvoi introuvable.**, la réponse indicielle en boucle fermée a été représentée pour l'entrée du système (erreur corrigée) et la sortie du système.

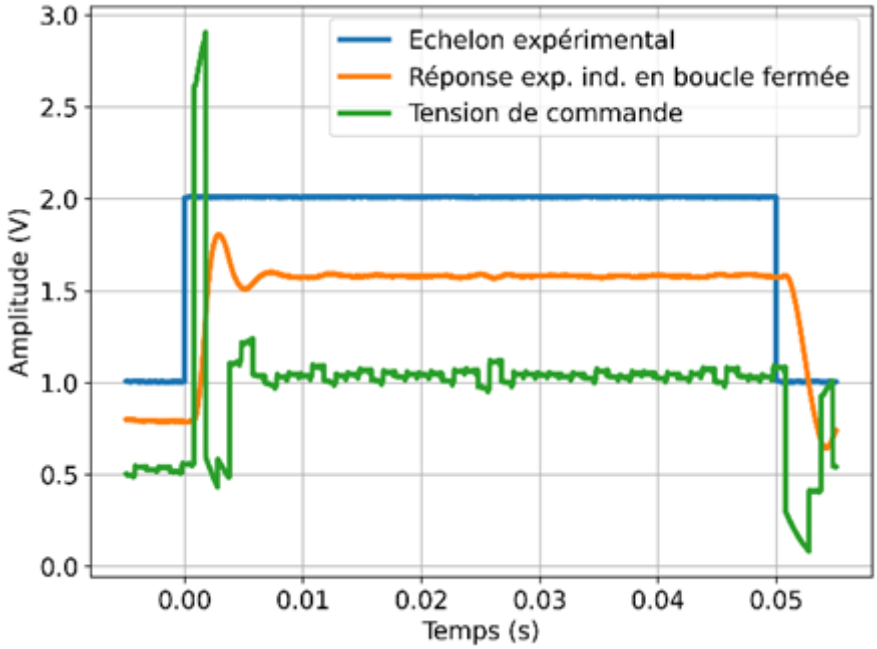

Figure 13 : Visualisation des grandeurs expérimentales en boucle fermée

Un zoom sur la tension de commande permet de vérifier que le temps entre deux échantillons est constant et égal à 1ms comme souhaité. De plus, pour l'échelon choisi, il n'y a pas de saturation de la tension de commande.

4.3 - Comparaison avec un correcteur PI numérisé

Dans ce paragraphe, nous allons comparer les solutions d'un asservissement par un régulateur RST et un PI numérisé par une méthode d'Euler. Le correcteur PI est dimensionné pour obtenir une erreur statique nulle et un temps de réponse deux fois plus rapide. Sur la **Erreur ! Source du**

renvoi introuvable., la réponse indicielle en boucle fermée du système avec un correcteur PI en temps continu est représentée.

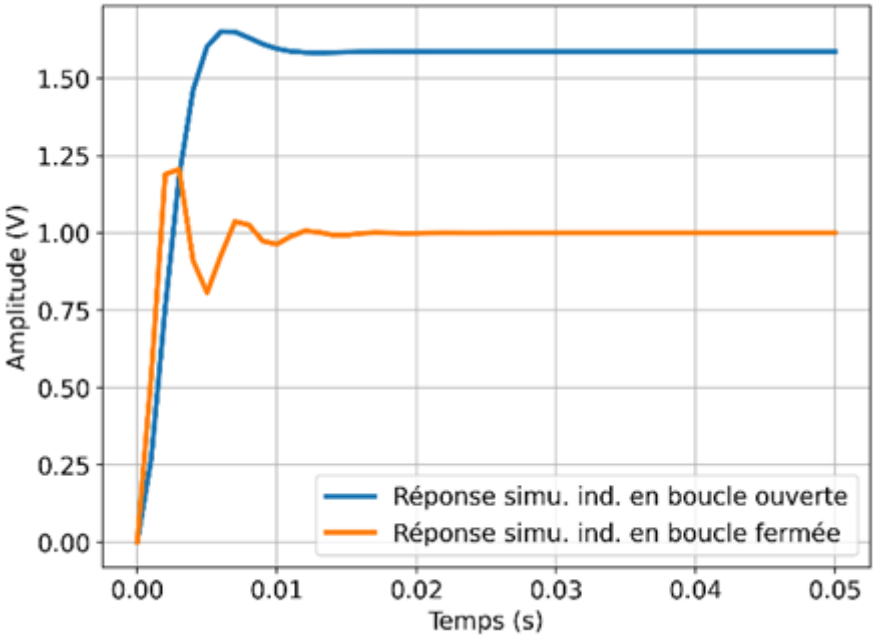

Figure 14 : Réponse à un échelon en temps continu d'un correcteur PI

Ce réglage permet de satisfaire le cahier des charges. À partir de ce correcteur dimensionné à partir du système à temps continu, nous en déduisons le correcteur en temps discret. Les réponses des correcteurs RST et du correcteur PI numérisé sont représentées sur la Figure 16.

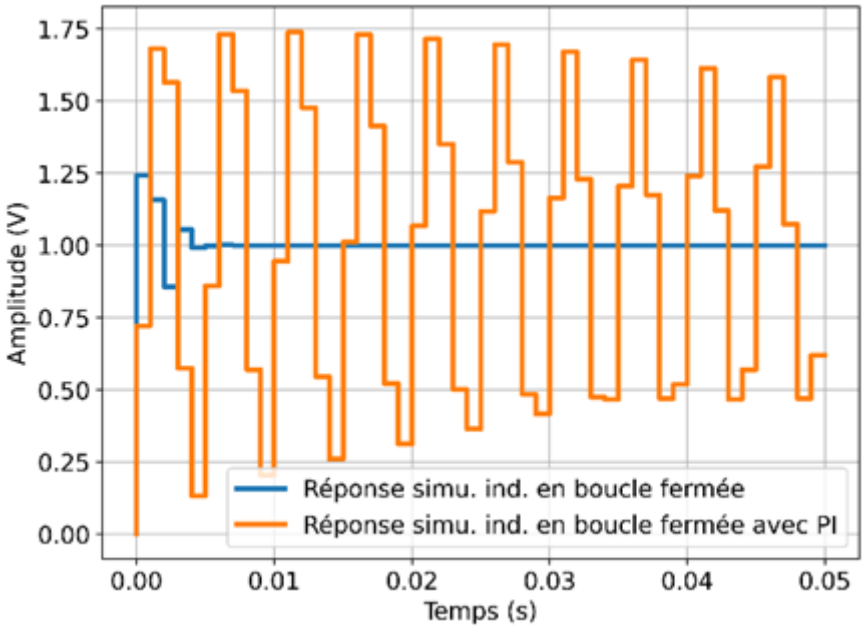

Figure 15 : Figure 16 : Comparaison des réponses en boucle fermée du correcteur RST et PI numérisé

La période d'échantillonnage a été dimensionnée dès le départ pour mettre en défaut l'idée de numériser un correcteur à temps continu en un correcteur temps discret. Ainsi, la réponse indicielle du système corrigé par un PI numérisé ne respecte plus le cahier des charges et est à la limite de l'instabilité.

5 – Conclusion

Dans cette ressource, nous avons tenté de mettre en avant l'intérêt de considérer les asservissements numériques tels qu'ils sont sans vouloir transposer des asservissements dimensionnés en temps continu vers le temps discret. En effet, la difficulté pour dimensionner directement en temps discret un correcteur tel que le RST est toute relative, notamment grâce à l'apport des méthodes numériques. De plus, l'apprenant pourrait faire un raccourci malheureux entre les asservissements à temps continu et ceux à temps discret. Même un correcteur aussi simple qu'un correcteur proportionnel à temps continu, n'est pas un gain de proportionnalité en temps discret. Le temps d'échantillonnage est aussi important que le gain proportionnel sur la stabilité du système. Pour finir, les asservissements à temps discret permettent d'obtenir des performances et des fonctionnalités très complémentaires à celles des asservissements à temps continu.

L'utilisation d'un hardware tel que la Pyboard permet d'envisager des séances de travaux pratiques en distanciel puisque chaque apprenant est autonome avec la carte de développement, le système considéré alimenté par la carte et un ordinateur servant d'oscilloscope. La gestion de la liaison série de la Pyboard à l'ordinateur n'a pas été traité dans cette ressource, cela pourra être développé dans une ressource future, notamment en lien avec des plateformes pédagogiques telles que Steeve [3].

L'atelier est téléchargeable librement en annexe de ce document.

Références :

[1]: A. Dupas and J. Ojeda, « Apports des outils numériques sur l'enseignement de l'automatique : séance de travaux pratiques distanciels asynchrones », juillet 2020, https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/apports-des-outils-numeriques-sur-lenseignement-de-lautomatique

[2]: J. Ojeda, « Apports des outils numériques sur l'enseignement de l'automatique : Ateliers à partir d'un notebook Jupyter », novembre 2020, https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/apports-des-outils-numeriques-sur-lenseignement-de-lautomatique-travaux-pratiques-distanciels-asynchrones

[3]: F. Louf, H. Horsin Molinaro, « Réalisez des TP à distance avec STEEVE », décembre 2020, https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/realiser-des-tp-a-distance-avec-steeve

[4]: I. D. Landau, "The R-S-T digital controller design and applications", Control Eng. Pract., vol. 6, no. 2, pp. 155–165, Feb. 1998, doi: 10.1016/S0967-0661(98)00016-1

[5]: G. Alegrin, « Méthodes de synthèse de correcteurs numériques », Tech. l'ingénieur, vol. 2, no. R7420, 1996

[6]: P. Varoqui and A. Juton, « Utilisation de télé-TP en informatique industrielle », Janvier 2021, https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/utilisation-de-tele-tp-en-informatique-industrielle2021

[7]: E. Monmasson and M. N. Cirstea, "FPGA Design Methodology for Industrial Control Systems—A Review", IEEE Trans. Ind. Electron., vol. 54, no. 4, pp. 1824–1842, Aug. 2007, doi: 10.1109/TIE.2007.898281

[8]: « Micropython », https://micropython.org/

[9]: A. Besancon-voda and S. Gentil, « Régulateurs PID analogiques et numériques », Tech. lingénieur, vol. R7416, 1999

Ressource publiée sur Culture Sciences de l'Ingénieur : https://eduscol.education.fr/sti/si-ens-paris-saclay