Compte-rendu

Bureau d’étude Chaîne d’acquisition et Commande numérique

# 

# 

Table des matières

[**Introduction**](#_ruzxzhzhffye) **2**

[**Asservissement de courant**](#_asp8hn22isen) **3**

[1 - Modélisation de Laplace en continu](#_ftinilok7q97) 3

[2 - Synthèse du correcteur C(p) à la main…](#_k2k6gjpu3hex) 6

[A - C(p) correcteur proportionnel simple](#_q4qbv15miw40) 7

[B - C(p) correcteur PI](#_uklz1jxsu1up) 7

[3 - Travail sur Matlab](#_uafkt32w25m0) 8

[A - Confirmation de notre théorie en continu](#_qdbg64o39w68) 8

[B - Détermination de C(z) par la transformée bilinéaire](#_v1nxairxj6n2) 9

[C - Détermination de Te](#_4l0j61cz8p9e) 9

[D - Modélisation du système discret sur Matlab](#_qkqblmgd4b5r) 10

[4 - Travail sous Keil](#_mwdm4jb5hahn) 12

[A - Détermination de la relation de récurrence](#_q8c82fggg5e6) 12

[B - Implémentation sur Keil et simulation](#_98h4r58eld6m) 12

[C - Comparaison avec Matlab](#_gj068wbeexth) 13

[**Conclusion**](#_pydelupdhuk6) **14**

[**Annexe**](#_dugvtbo95orj) **15**

[Code Matlab de notre modèle (continu et discret)](#_nsh97em6j1vq) 15

## 

## Introduction

Le Bureau d'Études de chaîne d’acquisition et commande numérique avait pour objet une trottinette électrique, datant de 2007. La trottinette devait d’abord être propulsée par l’utilisateur pour pouvoir la démarrer. Ensuite, l’utilisateur pouvait actionner la commande et la trottinette allait à environ 10 km/h (sur du plat). L’utilisateur ne pouvait alors pas contrôler la vitesse ni le couple.

Le but de ce BE était de contrôler la trottinette en vitesse et en couple. La trottinette a été customisée avec des cartes électroniques de régulation, une carte de puissance, et différents capteurs et commandes.

Dans une première partie, nous avons analysé les différents groupes constituant la trottinette (customisée). Nous avons analysé : le groupe motopropulseur, la commande électrique, le hacheur quatre quadrants (qui est le cœur du convertisseur de puissance à haut rendement), la régulation de courant/couple, la régulation de vitesse (tout cela pour la première version de la trottinette), et enfin la carte contrôleur de puissance (nouvelle version de la trottinette).

Le cahier des charges est le suivant : à partir du système, on doit déterminer un correcteur satisfaisant les conditions suivantes : une marge de phase supérieure ou égale à 45° dans le pire des cas, une fréquence de coupure en boucle fermée entre 300 et 500 Hz, et enfin une erreur statique nulle en boucle fermée.

## Asservissement de courant

On va travailler sur le système entier pour mettre en place une régulation du système. Le bloc G(p) correspond à la fonction de transfert qui donne la grandeur à asservir à partir de la grandeur de commande. F(p) correspond à la fonction de transfert qui donne la grandeur mesurée et filtrée (entrée ADC) en fonction de la grandeur à asservir. C(p) est le correcteur que nous allons dimensionner.

### 1 - Modélisation de Laplace en continu

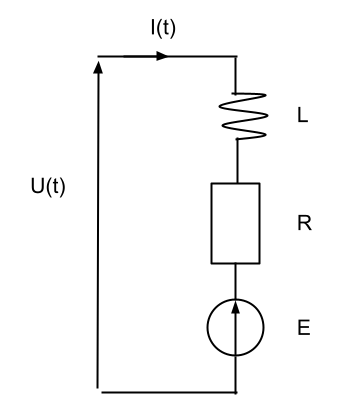
1. Identifier G(p): ensemble des blocs qui composent G(p) avec, pour chaque bloc:

* un nom
* une expression littérale en p
* les applications numériques de chaque constante de temps ou gain.

La fonction G(p) est composée de deux blocs : le hacheur et le moteur, qui constituent la partie opérative.

* ***Moteur***

Le moteur est constitué d’une charge LRE série, comme dessiné dans la figure suivante.



*Figure 2 : Circuit électrique du moteur*

La grandeur physique d’entrée est U(p) = L[U(t)], et la grandeur physique de sortie est I(p) = L[I(t)].

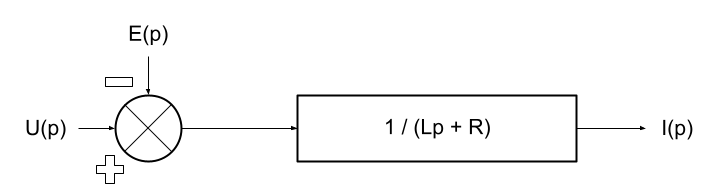
Lorsque la trottinette est en marche avant, on a E > 0, lorsqu’elle est en marche arrière E < 0, et on a E = Kφ. Ω

Toutes les grandeurs sont décrites par une partie continue et une partie alternative, qu’on va séparer lorsqu’on fait l’équation du système. Ainsi, on a :

et

On va appliquer la transformée de Laplace seulement aux parties alternatives (exemple : ) d’où : U(p) = E(p) + LpI(p) + RI(p) =>

Le schéma bloc correspondant est sur la figure ci-dessous. La fonction réalisée est un passe-bas d’ordre 1, qui va lisser la tension U(t) de la partie hacheur.



*Figure 3 : Schéma bloc du moteur*

* ***Hacheur***

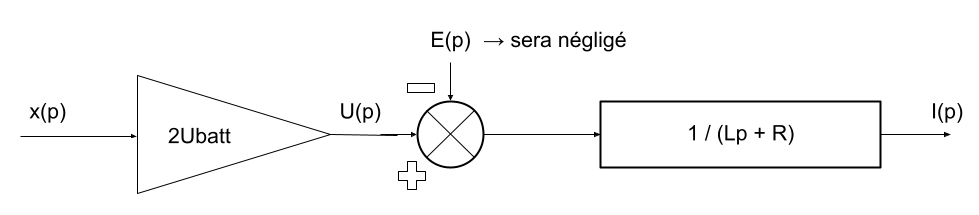
On a vu au cours des premiers TPs que la fonction réalisée par le hacheur est :

<U> = (2⍺-1) \* Ubatt

On va linéariser cette équation autour de , car à ce moment <U> = 0, la trottinette est à l’arrêt sur du plat.

De même que pour le moteur, on travaille avec les valeurs continues et alternatives des grandeurs : donc on a <U> = 2x(t)\*Ubatt ce qui implique que : <U(p)> = 2x(p)\*Ubatt.

On remplace U(p) dans le schéma bloc précédent et on obtient le schéma bloc suivant, qui représente donc G(p).



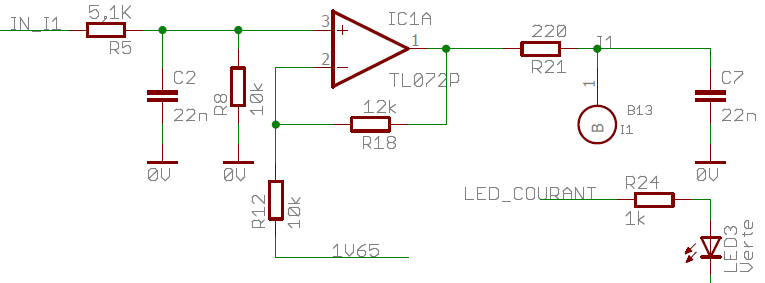
*Figure 4 : Schéma bloc de la fonction G(p)*

On va négliger E(p) : en effet, la constante de temps mécanique étant largement supérieure à la constante de temps électrique, on a : est considéré comme nul car trop lent. Donc quand on applique la transformée de Laplace, on a E(p) 0, voilà pourquoi on suppose que E(p) est nul.

|  |
| --- |
| On a donc : avec (d’après le sujet).  Et on obtient . |

2. Identifier F(p) de la même manière

Dans les questions préliminaires, nous avons étudié au cours de la partie 6 - Etude de la carte Contrôleur de puissance la fonction F(p) = .



*Figure 5 : Schéma électronique du filtre*

Nous avons obtenu :

Pour le filtre :

avec car SI = 0,104.

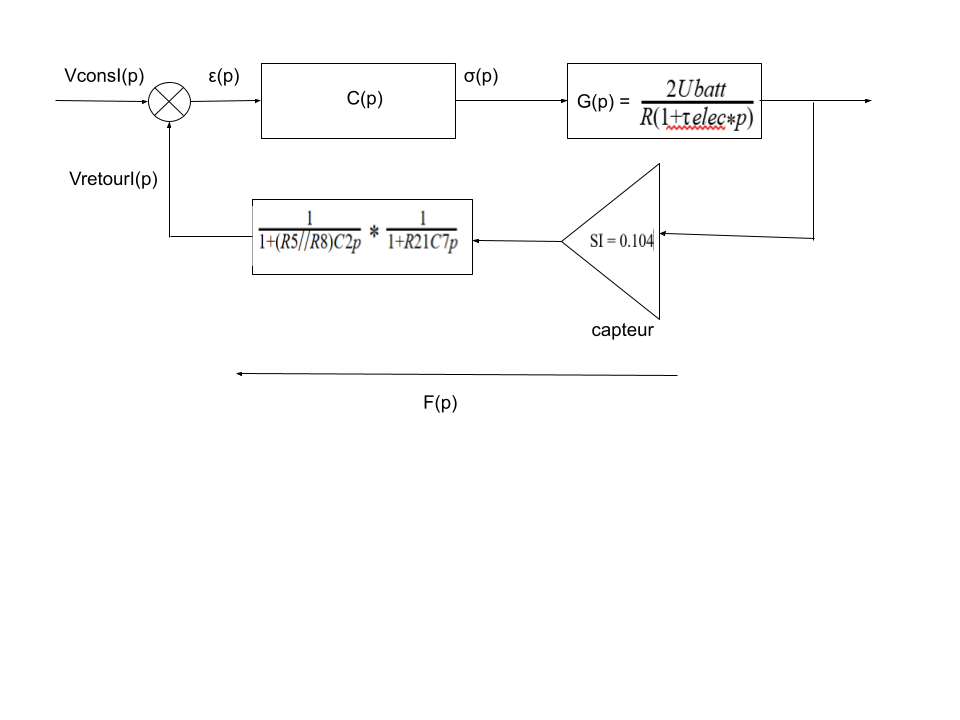
Pour le capteur :

Donc on a .

|  |
| --- |
| Ici :  Et on a : donc  Et : donc |

3. Dessiner le modèle du système complet sous forme de schéma-bloc de Laplace. On trouvera notamment:

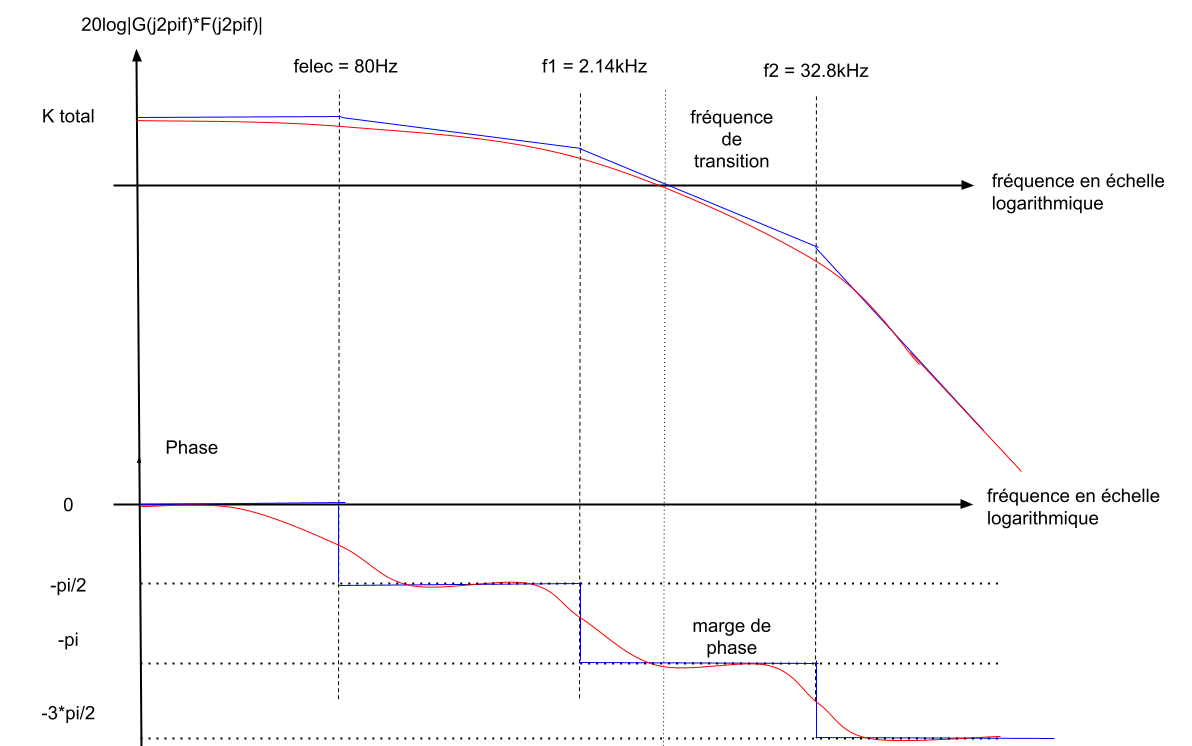
* C(p), G(p), F(p) éventuellement décomposés en plusieurs blocs.
* Toutes les grandeurs intermédiaires.



*Figure 6 : modèle du système complet sous forme de schéma-bloc de Laplace*

### 2 - Synthèse du correcteur C(p) à la main…

On trace G(p)\*F(p) “à la main”. On obtient les courbes suivantes :



*Figure 7 : diagramme de Bode de la fonction G(p)\*F(p)*

On remarque que pour la fréquence de transition, la marge de phase est très petite. Si on utilise un correcteur proportionnel simple, il faut que son gain soit inférieur à 1, car on veut pouvoir “baisser” la courbe pour déplacer la fréquence de transition à 400Hz (spécification du cahier des charges).

#### A - C(p) correcteur proportionnel simple

On a , donc on obtient

On utilise ensuite le théorème de la valeur finale : . Si l’erreur tend vers 0, alors on peut garder le correcteur proportionnel simple, sinon il faudra en choisir un plus sophistiqué.

On fait donc

Et =

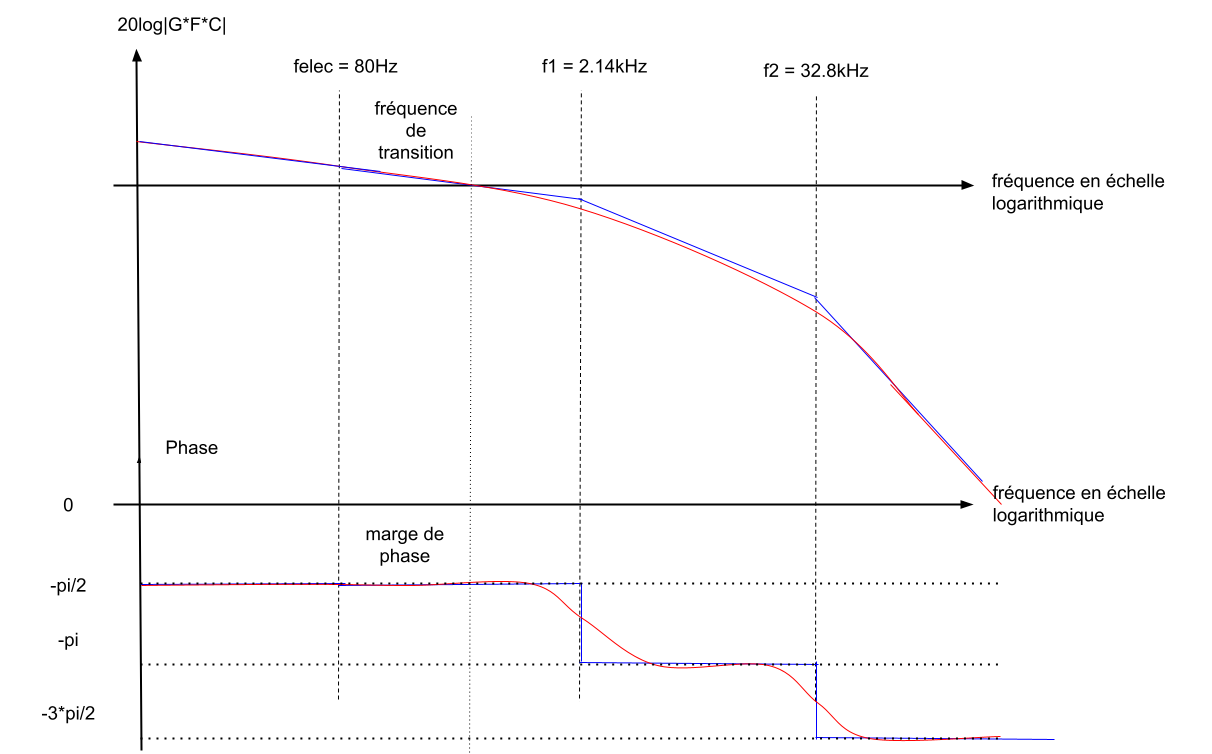
On conclut donc que le correcteur proportionnel simple ne suffit pas, on va donc prendre un correcteur PI.

#### B - C(p) correcteur PI

Un correcteur PI aura une équation suivante :

Dans un premier temps, on veut réaliser une compensation de pôle : on compense le pôle dominant (c’est-à-dire la fréquence la plus faible). On fait donc .

Dans un second temps, on veut que la fréquence de transition soit égale à 400Hz. On obtient donc les courbes et équations suivantes :



*Figure 8 : diagramme de Bode de la fonction G(p)\*F(p)\*C(p) pour C(p) correcteur PI*

On a donc, pour ce correcteur, la fonction en boucle ouverte :

On remarque bien que l’on a compensé le pôle dominant (puisqu’il n’apparaît plus). On cherche maintenant à exprimer littéralement le module de cette fonction : la condition pour avoir la marge de phase égale à il faut que ce module soit égal à 1. On pourra ainsi trouver la dernière constante de temps .

Pour rappel, on place la fréquence de transition à .

donc . Ici on va considérer que les fractions dans les racines sont négligeables (elles sont égales à environ 0.0349 et 1.48e-4). On en déduit :donc

|  |
| --- |
|  |

Donc on a . La marge de phase est bien supérieure à 45°, comme spécifié dans le cahier des charges.

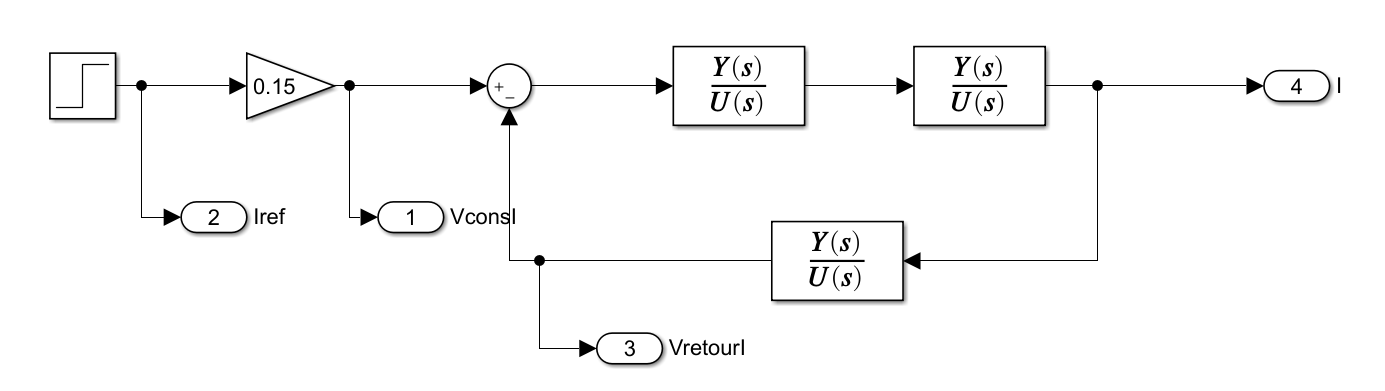
### 3 - Travail sur Matlab

#### A - Confirmation de notre théorie en continu

On implémente notre correcteur et notre système sur Matlab et Simulink. Le code Matlab sera donné en annexe. Sur la figure ci-dessous, on peut voir le modèle simulink de notre système. Pour pouvoir comparer notre sortie (un courant), nous avons créé une fonction step de 1 ampère qui sert de référence au courant.

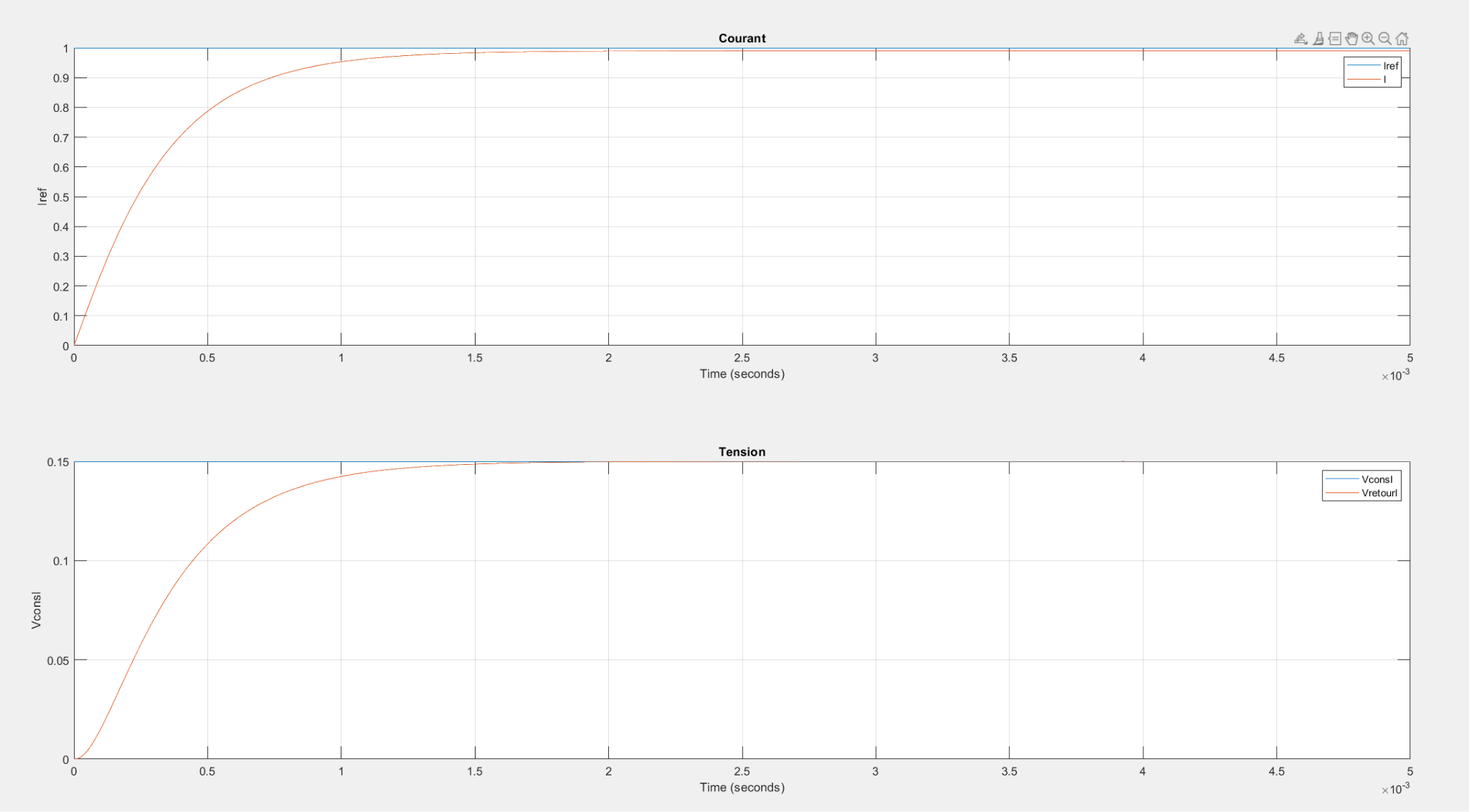
Pour avoir la tension de consigne, on multiplie ce step par 0.15, car c’est le gain en boucle ouverte.

Les fonctions sont respectivement C(s), G(s) et F(s).



*Figure 9 : modèle du système complet en continu sur simulink*

Les résultats que l’on obtient avec ce modèle sont affichés sur l’image suivante. On voit bien que l’erreur entre les consignes et les valeurs réelles tend vers 0, ce qui était demandé par le cahier des charges. On peut donc valider notre correcteur PI en continu.

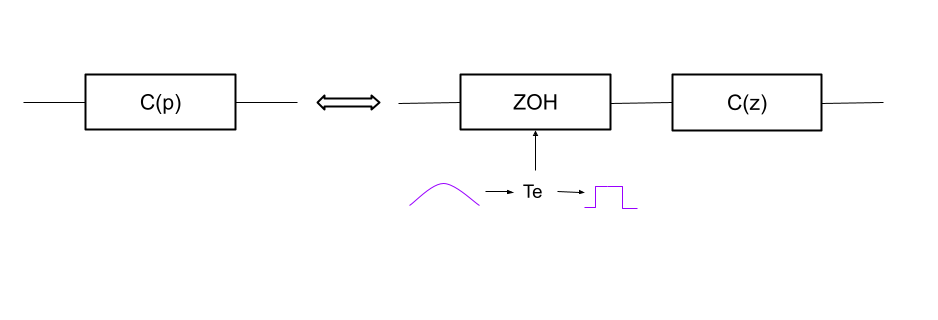


*Figure 10 : résultats de la modélisation sur Matlab en continu*

#### 

#### B - Détermination de C(z) par la transformée bilinéaire

On veut obtenir l’expression de C(p) discrétisée. Comme on peut le voir sur la figure suivante, on va échantillonner le signal en amont de C(z) grâce à un Zero-Order-Hold, au lieu de réaliser l’échantillonnage directement dans C(z). On utilise la méthode de Tustin pour faire la discrétisation.



*Figure 11 : but de la transformée*

L'équation du ZOH est :

De plus, . Comme on a déterminé l’équivalence entre p et z, on le remplace dans l’équation de C(p) pour obtenir C(z) :

#### C - Détermination de Te

L’échantillonnage introduit un retard pur de Te/2. On a donc .

On ajoute donc la fonction avant notre correcteur. La nouvelle fonction de transfert en boucle ouverte est donc : . Le module ne change pas, donc le gain non plus, car on a seulement un retard pur.

Le nouvel argument est donc :. La nouvelle courbe de phase est donc proportionnelle à la fréquence, et décroît quand la fréquence augmente.

La nouvelle marge de phase est donc . On la choisit inférieure à , car on peut se permettre un retard de phase de 30° environ.

On obtient donc :

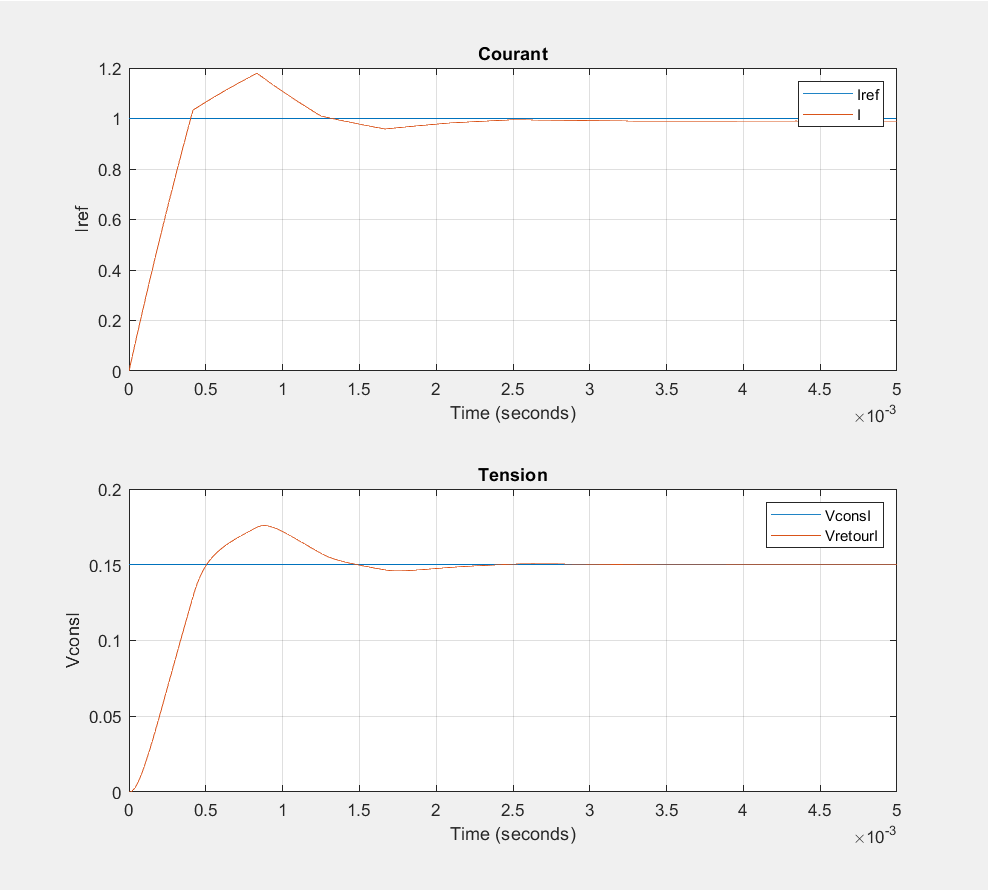
|  |
| --- |
| et donc |

#### D - Modélisation du système discret sur Matlab

On modélise ensuite notre système continu/discret sur Matlab et Simulink. On ajoute le ZOH et on modifie notre C(p). Pour les résultats suivants on a choisi .

#### 

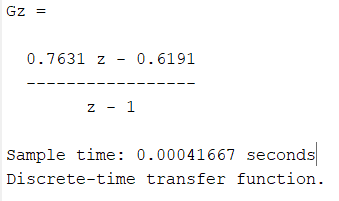
*Figure 12 : modèle du système complet discret sur simulink*



*Figure 13 : résultats de la modélisation du système discret sur Matlab*

On voit sur les figures précédentes que le correcteur numérique correspond bien aux attentes du cahier des charges : en effet, l’erreur statique diminue rapidement jusqu’à tendre vers 0.

On vérifie avec la commande c2d (continu to discrete) : on obtient C(z) égal à la fonction suivante, cela correspond à ce que nous avons calculé précédemment.



*Figure 14 : Discrétisation de C avec c2d en utilisant l’argument ‘tustin’*

### 4 - Travail sous Keil

#### A - Détermination de la relation de récurrence

On a . A partir de cette équation, on va calculer l’équation de récurrence qui correspond à notre correcteur :

Cette équation de récurrence est le seul moyen d’implémenter notre correcteur sur Keil, en effet, on ne peut pas y ajouter de fonction de transfert.

#### B - Implémentation sur Keil et simulation

A partir de l’équation de récurrence, on modélise notre système sous Keil. On se place en boucle ouverte et on envoie un échelon de 0.15V pour correspondre à notre simulation Matlab et observer ce qu’il se passe dans les deux cas. On cherche à voir si les pentes des réponses des systèmes implémentés sous Matlab et Simulink sont les mêmes; dans ce cas, le système sous Keil est correctement codé. Le code suivant est exécuté dans l’interruption “Systick” :

*Y = Ym1 + (Te+2\*t3)/(2\*t4)\*U + (Te-2\*t3)/(2\*t4)\*Um1 ;*

*if (Y < -0.5) {*

*Y = 0.5 ;*

*} else if (Y > 0.5) {*

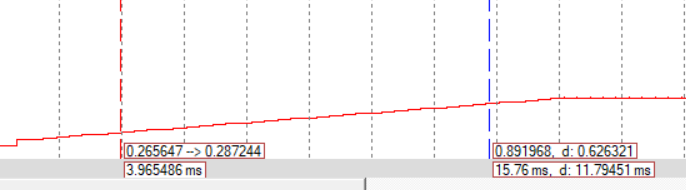
*Y = 0.5 ;*

*}*

*Ym1 = Y ;*

*Um1 = U ;*

*}*



*Figure 15 : résultats de la réponse de C(z) à un échelon sur Keil*

Après calcul, la pente que l’on observe est égale à 53.10 [V/s].

#### C - Comparaison avec Matlab

## 

*Figure 16 : résultats de la réponse de C(z) à un échelon sur Matlab*

Après calcul, la pente que l’on observe est égale à 51.83 [V/s].

Les pentes sont très proches : cela indique que le code Keil que nous avons écrit décrit bien le correcteur discret. Les conditions du cahier des charges sont donc bien respectées.

## 

## Conclusion

Ce BE nous a permis de comprendre comment dimensionner un correcteur à partir d’un système et d’un cahier des charges par rapport à ce système. Nous avons utilisé divers logiciels, comme Matlab et Keil. Nous avons utilisé la transformée bilinéaire pour pouvoir élaborer le correcteur numérique à partir du correcteur analogique.

Nous n’avons pas rencontré de difficultés particulières, nous n’avons juste pas eu le temps de tester notre code Keil sur le vrai système, ce qui s’est avéré frustrant puisque l’avantage de ce BE est que nous puissions avoir des résultats concrets.

La remarque que nous souhaitions faire par rapport à ce BE est qu’il est parfois difficile d’avoir une vue d’ensemble sur ce que nous faisons. On se concentre sur certaines questions/parties, donc on oublie parfois pourquoi on cherche à dimensionner ces parties. Quelques séances supplémentaires auraient été bénéfiques à une compréhension plus globale.

## Annexe

### Code Matlab de notre modèle (continu et discret)

clear all

close all

clc

%% CONSTANTES

Ubatt = 24 ;

R = 1 ;

L = 2e-3 ;

t\_elec = L/R ;

SI = 0.104 ;

R5 = 5.1e3 ;

R8 = 10e3 ;

R12 = 10e3 ;

R18 = 12e3 ;

R21 = 220 ;

C2 = 22e-9 ;

C7 = 22e-9 ;

K = SI \* (R8/(R5+R8)\*(1+R18/R12)) ;

t1 = (R5\*R8\*C2)/(R5+R8) ;

f1 = 1/(2\*pi\*t1) ;

t2 = R21\*C7 ;

f2 = 1/(2\*pi\*t2) ;

t3 = t\_elec ;

f3 = 1/(2\*pi\*t3) ;

ft = 400 ;

f4 = (ft\*R)/(2\*Ubatt\*K) ;

t4 = 1/(2\*pi\*f4) ;

Kretour = (R8/(R5+R8))\*(1+R18/R12)\*SI ;

Fe = 6\*ft ;

Te = 1/Fe ;

%% FONCTIONS DE TRANSFERT

numG = [2\*Ubatt] ;

denG = [R\*t\_elec R] ;

G =tf(numG,denG) ;

numF = [K] ;

denF = [t1\*t2 t1+t2 1] ;

F = tf(numF,denF) ;

numC = [t3 1] ;

denC = [t4 0] ;

C = tf(numC,denC) ;

%% DECLARACTION DES PARAMETRES DE SIMULATIONS

tstop = 20e-3 ;

tp = 10e-6 ;

%% SIMULATION

simu = sim('modele.mdl') ;

simu2 = sim('cz.mdl');

%% TRACÉ DES COURBES

figure(1)

subplot(2,1,1)

plot(simu.yout{2}.Values)

hold on

plot(simu.yout{4}.Values)

title('Courant')

grid ;

legend('Iref','I');

subplot(2,1,2)

plot(simu.yout{1}.Values)

hold on

plot(simu.yout{3}.Values)

title('Tension')

grid ;

legend('VconsI','VretourI');

figure(2)

plot(simu2.yout{3}.Values)

title('Réponse de C(z) à un échelon')

grid ;

legend('Cz')

%% DISCRETISATION

Gz = c2d(G,Te,'tustin') ;

## 

## 