TP Asservissement Echantillonés : Mini Projet Asservissement de Vitesse et de Position

Boris Alexandre Baudel - ENS Rennes - Departement Mécatronique

Octobre 2023

1 Introduction

Le TP a pour but de faire un asservissement échantillonés d'un moteur à courant continue. La maquette est composée d'un ensemble moteur à courant continu d'un codeur de référence et d'un réducteur Pololu 2822 dont on connait les caractéristiques. On peut voire à quoi ressemble le moteur dans la figure 1. Le but du TP sera d'implémenter un code Arduino pour faire l'asservissement en vitesse et en position, choisir un correcteur pour controler ainsi le moteur en vitesse et en position.

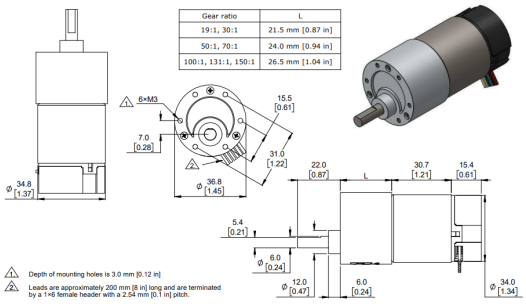

Figure 1: Caption de la figure

2 Activité 1 : Etude des variables et du codeur

Les documents des moteurs 37D de Pololu explique le fonctionnement du codeur. Il s'agit d'un codeur qui fonctionne en quadrature intégré et qui possède une résolution de 64 comptes par révolution. Comme il y a deux voie cela revient à 16 comptes par révolution par voie pour un front montant. On note P le nombre d'impulsion par révolution et G le rapport de réduction.Par ailleurs, en sortie nous avons 1200 points/révolution, ce qui nous donne aussi un indice de la résolution du codeur. Nous pouvons donc calculé la résolution en dégré par :

$$R = \frac{360}{P * G} = \frac{360}{64 * 18,75} = 0,3 \tag{1}$$

Pour le calcule de la période minimale des impulsions du codeur, on a :

$$Tmin = \frac{1}{P * Wmax} = \frac{1}{64 * 560 * 2\pi/60} = 1,79ms \tag{2}$$

Le code fourni fait référence à la mesure de la vitesse du moteur et a son asservissement. Dans un premier temps, DTstimulius, DTdebugus, DTasserus définissent les variationbs de temps qui servent pour les déclanchement des stimulis, le débogage, et finalement l'asservissement.

Pou les variables MOTDIR, MOTPWM, LED, bcA, bcB il s'agit des numéros des broches de Arduino : la direction du moteur, la modulation en largeur d'impulsion,la LED, et les voies A et B de l'encodeur.

nc, lastinc, currenttime, lasttimeasser, lasttimestimuli, lasttimedebug, currentinc, vitesse, coefpassebas, etat, pwm, sont utilisées pour permettre de stocker les états du programme, le compteur de l'encodeuret la vitesse du moteur.

Pour le lien avec le cablage, MOTDIR et MOTPWM controlent la direction et la vitesse du moteur. bcA et bcB sont à la sortie de l'encodeur et servent à mesurer la vitesse du moteur. LED sert pour le débogage s'il le temps de de boucle est trop élevé.

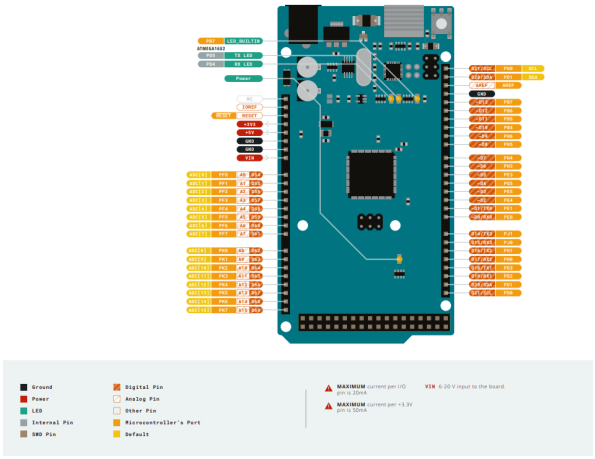

Figure 2: Caption de la figure

3 Activité 2 : Fonctionnement du codeur à deux voies et mesure des vitesses

Le codeur transforme une position angulaire en un signal électrique de deux voie A et B. Les deux voies A et B en sortie qui sont déphasé de 90 degré. Avec ce système, le microcontroleur peut déterminer un sens de rotation positif ou négatif. et compter les impulsion et mésurer ainsi la postion. Nous avons 64 points par tour. Cela revient d'utiliser la voie A à 16 point par tour pour chacun des fronts montant.

La fonction Encoder() est appelée quand une interruption est déclanchée par une variation de l'état sur la voie A.

Figure 3: Caption de la figure

attachInterrupt(digitalPinToInterrupt(bcA), Encoder, CHANGE) sert à gérer l'interruption. Lorsque nous avons un changement d'état de la Voie A, de High à Low, la fonction encoder va être appelée. Cela, va permettre de vérifier si A et B sont les des signaux identiques. Si c'est le cas, le programme ajoute 1 à l'incrément inc sinon -1.

Pour la mesure de la vitesse et du Filtrage,la vitesse est calculée avec un filtre passe bas du premier ordre. Le filtre sert à lisser le signal.La valeur de coefpassebas est calculé à partir de la constante de temps et du temps d'échantillonage. La vitesse dépend de la vitesse précédente et de nouvelle mesure car il s'agit d'une combinaison pondéré.

Pour la gestion du Stimuli et pwm, celle-ci est réalisé dans la boucle loop toutes les variations de Dtstimulis qui coresponde à des microsecondes. Ainsi, l'état et le sens du moteur est changé. Les lignes analogWrite(MOTPWM,pwm); et digitalWrite(MOTDIR,sens); appliquent ces changements au matériel.À chaque intervalle de 2 secondes, pwm alterne entre 0 et 100, contrôlant ainsi la puissance fournie au moteur et, par conséquent, sa vitesse.

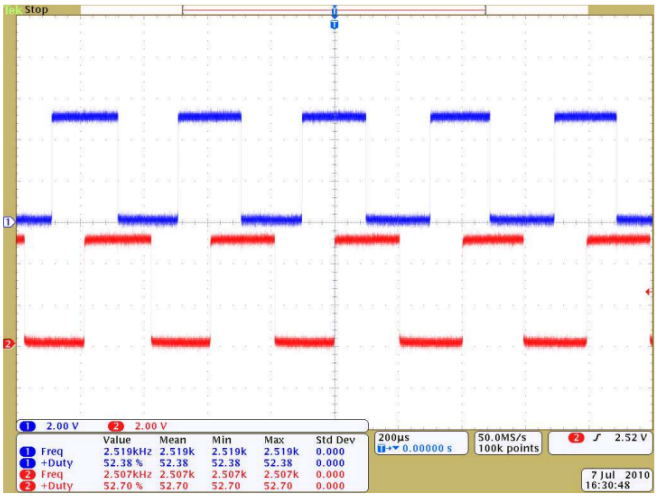

Figure 4: Caption de la figure

Par aileurs, lorsque l'on évalue la tension Vpwm, on obtient :

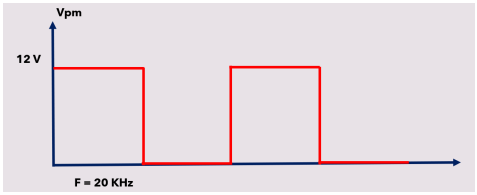

Figure 5: Caption de la figure

4 Activité 3 : Analyse du Code et prémier modèle

Tout d'abord, nous devons avoir le modèle du process, c'est à dire de comment est la fonction de transfert du système d'après ce que nous avons vu précédemment.

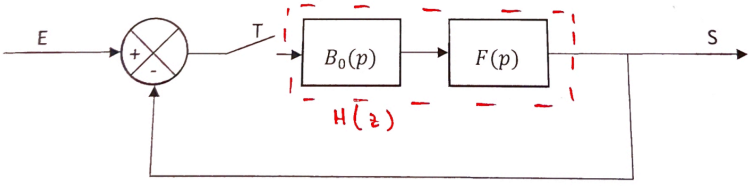

Figure 6: Caption de la figure

Les fonctions de transfert respective sont :

$$B_0(p) = \frac{1 - e^{-pT}}{pT} \tag{3}$$

$$H(p) = \frac{K}{\tau p + 1} \tag{4}$$

$$B_0(p)H(p) = \left(\frac{1 - e^{-pT}}{pT}\right)\left(\frac{K}{\tau p + 1}\right) \tag{5}$$

$$H(z) = \mathcal{Z}\{B_0(p)H(p)\} \tag{6}$$
$$= K(1-z^{-1})\mathcal{Z}\left\{\frac{H(p)}{p}\right\} \tag{7}$$
$$= K(1-z^{-1})\mathcal{Z}\left\{\frac{1}{(1+\tau p)p}\right\} \tag{8}$$
$$= K(1-z^{-1})\frac{1-a}{(z-a)(1-z^{-1})} \tag{9}$$
$$FTBO = \frac{K(1-a)}{z-a} \tag{10}$$

Ansi, on applique alors la définition de la fonction de transfert en boucle ouverte et on obtient alors :

$$Ht(z) = \frac{\frac{K(1-a)}{z-a}}{1+\frac{K(1-a)}{z-a}} \tag{11}$$
$$= \frac{K(1-a)}{z-a+K-Ka} \tag{12}$$
$$= \frac{K(1-a)}{z+K(1-a)-a} \tag{13}$$

Le code suivant est écrit pour Arduino et gère la vitesse et la direction d'un moteur à l'aide d'un encodeur. Il utilise diverses techniques, notamment la modulation de largeur d'impulsion (PWM) et les interruptions, pour contrôler le moteur et lire les données de l'encodeur. Le code n'a pas encore été adapté pour l'instant pour implémenter un correcteur, il s'agit du code qui a été fourni et dont les commentaires on été rajouté.

```
const unsigned long DTstimuli_us = 2000000;  //  stimuli
const unsigned long DTdebug_us = 10000;  // Intervalle pour le debogage
const unsigned long DTasser_us = 5000;  // Intervalle pour l'asservissemekt
// Broche de l'ARDUINO
const int MOT_DIR = 12;  // direction du moteur
const int MOT_PWM = 3;  //  PWM du moteur
const int LED = 13;  //  LED sur la carte Arduino
const int bcA=18;  // voie A de l'encodeur
const int bcB=19;  // voie B de l'encodeur

volatile long inc=0;  // Compteur d'impulsions
volatile long lastinc=0;  // Compteur d'impulsions a l'echantillon pr c dent

// temps
unsigned long current_time;  // Temps actuel
unsigned long last_time_asser = 0; //temps d'execution de l'asservissement
unsigned long last_time_stimuli = 0;  // temps d'execution du stimuli
unsigned long last_time_debug = 0;  // temps d'execution du d bogage

long current_inc;  // Impulsions actuelles
float vitesse=0;  // Vitesse du moteur
float coef_passebas = exp(-(DTasser_us/1000000.)/0.02);  // filtre passe-bas
char etat;  // etat actuel du systeme
int pwm;  // Valeur de la PWM
bool sens;  // Direction du moteur

// Configuration initiale
void setup() {
    Serial.begin(115200);  // serie a 115200 bauds

    pinMode(LED, OUTPUT);  // LED comme sortie
    pinMode(MOT_DIR, OUTPUT);  // Direction du moteur comme sortie
    pinMode(MOT_PWM, OUTPUT);  // PWM du moteur comme sortie
    pinMode(bcA, INPUT_PULLUP);  // Voie A de l'encodeur comme entr e avec pull-up
    pinMode(bcB, INPUT_PULLUP);  // Voie B de l'encodeur comme entr e avec pull-up

    // Attacher l'interruption pour la voie A de l'encodeur
```

```
    attachInterrupt(digitalPinToInterrupt(bcA), Encoder, CHANGE);  // Appeler Encoder
    () sur changement d' tat
}

// Fonction appel e sur interruption par changement d' tat  de la voie A de l'
    encodeur
void Encoder() {
    // Si les voies A et B sont  gales , incr menter; sinon, d cr menter
    if(digitalRead(bcB) == digitalRead(bcA)) inc++;
    else inc--;
}

// Boucle principale
void loop() {
    // Mise    jour du temps actuel
    current_time = micros();

    // Bloc pour l'asservissement
    if(current_time - last_time_asser >= DTasser_us) {
        last_time_asser = current_time;
        current_inc = inc;  // Conserver la valeur actuelle m me si une interruption
    survient apr s
        // Calcul de la vitesse avec filtrage passe-bas
        vitesse = vitesse*coef_passebas + (1-coef_passebas)*(current_inc-lastinc);
        lastinc = current_inc;
    }

    // Bloc pour la gestion du stimuli
    if(current_time - last_time_stimuli >= DTstimuli_us) {
        last_time_stimuli = current_time;
        // Alternance de l' tat , de la PWM et du sens du moteur
        if(etat == 0) {
            etat = 1;
            pwm = 100;
            sens = LOW;
        } else {
            etat = 0;
            pwm = 0;
            sens = HIGH;
        }
        // Application des nouvelles valeurs de PWM et de direction au moteur
        analogWrite(MOT_PWM, pwm);
        digitalWrite(MOT_DIR, sens);
    }

    // Bloc pour le d bogage
    if(current_time - last_time_debug >= DTdebug_us) {
        last_time_debug = current_time;
        // Envoi de la vitesse    la console s rie
        Serial.println(vitesse);
    }

    // V rification que la boucle ne d passe pas le temps DTasser_us et allumage de
    la LED si c'est le cas
    digitalWrite(13, (micros() - current_time >= DTasser_us));
}
```

5 Activité 4

Dans cette partie, nous devons faire l'identification de la fonction de Transfert pour trouver K et te le temps caractéristiques. Pour cela, nous lançons une commande et nous observons donc, en connnaissant ou se situe les valeurs caractéristiques pour un premier ordre dans la figure . Nous identifions ainsi les valeurs de notre fonction de transfert du premier ordre.

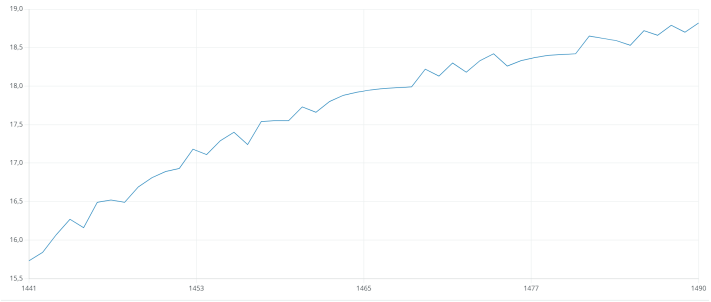

Figure 7: Caption de la figure

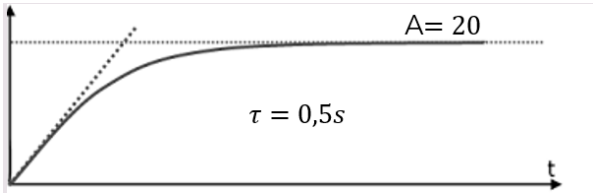

Figure 8: Caption de la figure

6 Activité 5

Pour asservir la vitesse tels que l'erreur statique soit nul il nous faut implémenter un correcteur PID don les coéffcient on été choisi expérimentalement. Dans notre cas, nous avons, 150,1/15 et 0 comme valeurs pour ce correcteur. Pour les valeurs spécifiques $K_p = 150$, $K_i = \frac{1}{15}$, et $K_d = 0$, la formule du correcteur PID devient :

$$C(s) = 150 + \frac{\frac{1}{15}}{s} + 0 \cdot s$$

$$C(s) = 150 + \frac{1}{15s}$$

```
const unsigned long DTstimuli_us = 2000000;
const unsigned long DTdebug_us = 10000;
const unsigned long DTasser_us = 5000;

// Param tres du correcteur PID
const float Kp = 10 * 15; // Gain proportionnel
const float Ki = 1 / 15; // Gain int gral
const float Kd = 0.0; // Gain d rivatif

// Variables pour le correcteur PID
float erreur = 0;
float erreur_precedente = 0;
float somme_erreurs = 0;
float vitesse_souhaitee_tr_min = 50.0; // Consigne en tr/min
float vitesse_souhaitee_tps = vitesse_souhaitee_tr_min / 60.0; // Conversion en tps

const int MOT_DIR = 12;
const int MOT_PWM = 3;
const int LED = 13;
const int bcA = 18;
const int bcB = 19;

volatile long inc = 0;
volatile long lastinc = 0;
unsigned long current_time;
unsigned long last_time_asser = 0;
unsigned long last_time_stimuli = 0;
unsigned long last_time_debug = 0;
long current_inc;
```

```
float vitesse = 0;
float coef_passebas = exp(-(DTasser_us / 1000000.) / 0.02);
int pwm;

void setup() {
  Serial.begin(115200);
  pinMode(LED, OUTPUT);
  pinMode(MOT_DIR, OUTPUT);
  pinMode(MOT_PWM, OUTPUT);
  pinMode(bcA, INPUT_PULLUP);
  pinMode(bcB, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(bcA), Encoder, CHANGE);
  digitalWrite(MOT_DIR, HIGH); // Definir le sens du moteur ici
}

void Encoder() {
  if (digitalRead(bcB) == digitalRead(bcA)) inc++;
  else inc--;
}

void loop() {
  current_time = micros();

  if (current_time - last_time_asser >= DTasser_us) {
    last_time_asser = current_time;
    current_inc = inc;
    vitesse = vitesse * coef_passebas + (1 - coef_passebas) * (current_inc - lastinc);
    lastinc = current_inc;

    // Calcul de l'erreur
    erreur = vitesse_souhaitee_tps - vitesse;

    // Calcul de la sortie du correcteur PID
    float sortie_correcteur = Kp * erreur + Ki * somme_erreurs + Kd * (erreur -
    erreur_precedente);

    // Mise    jour des variables pour le correcteur PID
    somme_erreurs += erreur;
    erreur_precedente = erreur;

    // Ajustement de la commande du moteur
    pwm = constrain(sortie_correcteur, 0, 255);
    analogWrite(MOT_PWM, pwm);
  }

  if (current_time - last_time_debug >= DTdebug_us) {
    last_time_debug = current_time;
    Serial.print("PWM: "); Serial.print(pwm);
    Serial.print(", Vitesse: "); Serial.println(vitesse);
  }

  // Verification du temps d'ex cution de la boucle
  digitalWrite(LED, (micros() - current_time >= DTasser_us));
}
```

Le code semble marcher et les commandes des vitesses en tours par minutes sont satisfaisantes. Cependant, de nombreux problèmes on été trouver pour trouver les valeurs du correcteur (oscillations, instabilités). De même, le sens de rotation était un peut penible et donc un sens de rotation a été mit dans un unique sens afin de ne pas prendre en compte la rapidité du système (En plus du fait que notre correcteur ne comporte pas de partie dérviateur ce qui affecte la rapidité)

7 Acitivités complémentaires

L'asservissement en position n'a pas pu être effectué par manque de temps. Sinon, pour effectuer un asservissement en position il faudrait placer un intégrateur.

8 Conclusion

Dans ce TP, nous avons pu mettre en place l'asservissement en vitesse d'un moteur à courant continu via un asservissement échantillonée. Nous avons pu aussi anaylser son comportement et sont fonctionnement en proposant un correcteur pour celui ci.