REGULATION ET ASSERVISSEMENT

COURS

TSSI cours asservissement.pdf

TP – Régulation de vitesse d’un moteur à courant continu avec PID

Objectifs du TP

* Mesurer la vitesse d’un moteur CC avec un codeur incrémental * Comprendre la différence entre boucle ouverte et boucle fermée * Mettre en œuvre progressivement un correcteur P, puis PI, puis PID * Observer l’erreur statique, le dépassement et la sensibilité au bruit

---

Travail demandé

* Décrire la réponse du moteur en boucle ouverte * Tracer ou décrire la courbe vitesse / consigne pour P * Expliquer pourquoi un écart statique persiste * Montrer comment I supprime cet écart * Comparer les dépassements pour P, PI et PID * Conclure sur l’intérêt des trois termes du PID

---

1) Commande du moteur en boucle ouverte

Manipulations :

* Envoyer une commande PWM fixe (ex : 50 %, puis 80 %) * Observer la vitesse indiquée par le programme * Bloquer légèrement l’axe du moteur avec le doigt

Observations attendues :

* La vitesse chute immédiatement lorsque l’axe est freiné * Le moteur ne corrige pas cette chute : c’est normal en boucle ouverte * La vitesse dépend de la charge, des frottements et de la tension

Conclusion : La boucle ouverte ne permet pas de maintenir une vitesse constante

---

2) Mise en place d’un correcteur P (Proportionnel)

Manipulations :

* Activer le correcteur proportionnel : $u = K_p \cdot e$ * Fixer une consigne (ex : 1500 tr/min) * Le

programme convertit automatiquement en ticks/s * Freiner légèrement le moteur avec le doigt * Augmenter progressivement K_p : 0.2 → 0.5 → 1.0 → 2.0

Observations attendues :

* Le moteur augmente la PWM pour compenser la perturbation * La vitesse remonte partiellement * Il reste un écart statique : $\text{vitesse réelle} < \text{consigne}$ * Si K_p devient trop grand : oscillations, vibrations, instabilité

Conclusion : Le correcteur P réduit l'erreur, mais ne la supprime pas

---

3) Mise en place du correcteur I (Intégral)

Manipulations :

* Ajouter le terme intégral : $u = K_p e; +; K_I \int e(t)\,dt$ * Débuter avec K_I = 0.05, puis 0.1 max * Freiner l'axe puis relâcher

Observations attendues :

* L'erreur statique disparaît * La vitesse atteint précisément la consigne * Si K_I trop fort : dépassement, oscillations lentes, instabilité

Conclusion : Le correcteur I supprime l'erreur statique, mais ne doit jamais être trop fort

---

4) Mise en place du correcteur D (Dérivé)

Manipulations :

* Ajouter le terme dérivé : $u = K_p e; +; K_I \int e\,dt; +; K_D \frac{de}{dt}$ * Tester avec K_D = 0.01, puis 0.05 * Freiner l'axe pour observer la réaction

Observations attendues :

* Le système est mieux amorti * Le dépassement diminue * La stabilité augmente

Attention : Si K_D trop élevé → bruit, vibrations, instabilité

Conclusion : Le terme D stabilise le système, mais n'améliore pas la précision

---

Synthèse des rôles P / I / D

Correcteur	Rôle principal	Risques si trop fort
P	réduit l'erreur	oscillations
I	supprime l'erreur statique	dépassement, instabilité
D	amortit, stabilise	amplification du bruit

Code Arduino du TP (consigne en tr/min, PID en ticks/s)

```
// === TP : PID régulation vitesse moteur CC ===
// Consigne entrée en tours/minute (tr/min)
// Le programme convertit en ticks/s pour le PID
// Mesure du codeur sur interruption

// --- Paramètres codeur ---
const int TICKS_PAR_TOUR = 20;   // à adapter selon votre codeur

// --- Pont en H ---
const int M_AV = 3;   // PWM forward
const int M_AR = 6;   // PWM reverse

// --- Codeur incrémental ---
const int canalA = 2;   // interruption 0
const int canalB = 11;

volatile long ticks = 0;  // compteur modifié par ISR

// === PID ===
float consigne = 0;        // consigne en ticks/s
float kp = 0.8;
float ki = 0.1;
float kd = 0.05;

float erreur, erreurPrec = 0;
float integral = 0;

// === Période mesure ===
unsigned long lastMeasure = 0;
const unsigned long period = 100;  // 100 ms

// === Conversion tr/min -> ticks/s ===
float trMinToTicksSec(float rpm) {
  return (rpm * TICKS_PAR_TOUR) / 60.0;
}

// === Prototypes ===
void ISR_codeur();
void commandeMoteur(float pwm);
float lireConsigne();
```

```
void setup() {
  Serial.begin(9600);

  pinMode(M_AV, OUTPUT);
  pinMode(M_AR, OUTPUT);

  pinMode(canalB, INPUT);

  attachInterrupt(digitalPinToInterrupt(canalA), ISR_codeur, RISING);

  Serial.println("=== TP : Regulation PID de vitesse ===");
  Serial.println("Entrez une consigne en tr/min (ex : 1500):");
}


// ====================== LOOP ===========================
void loop() {

  // --- Lecture consigne en tr/min ---
  if (Serial.available() > 0) {
    float rpm = lireConsigne();
    consigne = trMinToTicksSec(rpm);

    Serial.print("Consigne = ");
    Serial.print(rpm);
    Serial.print(" tr/min  ->  ");
    Serial.print(consigne);
    Serial.println(" ticks/s");
  }

  // --- PID toutes les 100 ms ---
  unsigned long now = millis();
  if (now - lastMeasure >= period) {
    lastMeasure = now;

    long ticksMesures = ticks;
    ticks = 0;

    float vitesse = ticksMesures * (1000.0 / period);  // ticks/s

    // === PID ===
    erreur = consigne - vitesse;
    integral += erreur * (period / 1000.0);
    float deriv = (erreur - erreurPrec) / (period / 1000.0);
    erreurPrec = erreur;

    float commande = kp * erreur + ki * integral + kd * deriv;

    // Saturation
```

```
    if (commande > 255) commande = 255;
    if (commande < -255) commande = -255;

    commandeMoteur(commande);

    // Affichage
    Serial.print("Consigne ticks/s = ");
    Serial.print(consigne);
    Serial.print(" | Vitesse = ");
    Serial.print(vitesse);
    Serial.print(" | PWM = ");
    Serial.println(commande);
  }
}


// === INTERRUPTIONS CODEUR ===
void ISR_codeur() {
  if (digitalRead(canalB))
    ticks++;
  else
    ticks--;
}


// === COMMANDE MOTEUR ===
void commandeMoteur(float pwm) {
  if (pwm >= 0) {
    digitalWrite(M_AR, LOW);
    analogWrite(M_AV, pwm);
  } else {
    digitalWrite(M_AV, LOW);
    analogWrite(M_AR, -pwm);
  }
}


// === LECTURE CONSIGNE ===
float lireConsigne() {
  String txt = Serial.readStringUntil('\n');
  txt.trim();
  return txt.toFloat();
}
```

From:
https://mistert.freeboxos.fr/dokuwiki/ - **Wiki de Sébastien TACK**

Permanent link:
https://mistert.freeboxos.fr/dokuwiki/doku.php?id=ssi_elec_regulation_asservissement&rev=1764425686

Last update: **2025/11/29 14:14**