Sommaire

Objectifs

- Correction d'un système du 2nd ordre en BF
- Analyse et caractéristiques du correcteur P
- Analyse de la Précision/ Stabilité/ Rapidité du correcteur P
- La réponse à un échelon d'un système en 2nd ordre
- Etc.

Fonctionnement

Le tuto est dédié à l'analyse des performances, limitations et implémentation d'un correction proportionnel (P). On fera la comparaison entre la réponse indicielle d'un système du second ordre corrigé et non corrigé. La carte Arduino Mega sera utilisé pour implémenter les deux systèmes aux même temps (voir le tuto pour plus des détails).

Caractéristiques du correcteur proportionnel

> *Fonction de transfert C(p)=Kc = constante*

Kc > 1

- Décalage du diagramme de gain de Bode vers le haut => augmentation la bande passante (wn) ⇒ augmentation de la rapidité
- Réduction de la marge de phase => dégradation de la stabilité en BF

Kc < 1

- Décalage du diagramme de gain de Bode vers le bas => Diminution diminution de la bande passante (wn) ⇒ diminution de la rapidité
- Augmentation de la marge de phase => Amélioration stabilité (lien du cours)

Kc=1 : Système non corrigé

Paramètre du système

```
#define   Fn      10.00

#define   Zeta    0.70710678118

#define   K       1.0

#define   T_ms    2
```

Paramètre du correcteur

```
#define   Kc      2.0
```

Étapes d’implémentation

La consigne (l’entrée) x(n) pour les deux systèmes

```
  c++; c=c%c_step;

  if(!c)

  {
```

```
  Step=!Step;

  c=0;

}

x_nn=A_step*(double)Step; // Réponse à un échelon x(n)=cte

//x_nn=(double)c;           // Réponse à une rampe x(n)=n
```

Sortie du capteur: Retour unitaire

```
y_capt[0]=y_n[0];

y_capt[1]=y_n[1];
```

Soustracteur: Calcul de l’erreur eps(n)

```
eps_n[0]=x_nn-y_capt[0];

eps_n[1]=x_nn-y_capt[1];
```

Correcteur

```
y_corr[0]=eps_n[0];     // Système non Corrigé

y_corr[1]=Kc*eps_n[1];  // Système Corrigé
```

Calcul de la sortie: Système non corrigé

```
  y_n[0]=Sys2All(y_corr[0], x1, y1, Zeta, Wn, K, T_s);
```

Calcul de la sortie: Système corrigé

```
  y_n[1]=Sys2All(y_corr[1], x2, y2, Zeta, Wn, K, T_s);
```

Affichage des signaux

```
  Serial.print(x_nn); Serial.print(",");

  Serial.print(y_n[0]); Serial.print(",");

  Serial.println(y_n[1]);
```

Code complet

```
/*
 * 1. Correction d'un système du 2nd ordre en BF
 * 2. Analyse du correcteur P
 * 3. Précision/ Stabilité/ Rapidité du correcteur P
 * 4. La réponse à un échelon d'un système en 2nd ordre
 * 5. Etc.
 *


                -------------    -------------
 x(n) --[-]------     Kc     -----     SYS2    ---------- y(n): Sortie Corrigée
```

```
         -       -------------    -------------   -
         -                                        -
         ---------------------<--------------------
         Correcteur: H(p)= Kc = Constante
                           -------------
 x(n) --[-]---------------    SYS2     ---------- y(n): Non Corrigée
         -                 -------------           -
         -                                         -
         ---------------------<--------------------

*/


#define   Fn      10.00
#define   Zeta    0.70710678118
#define   K       1.0
#define   T_ms    2

#define   Kc      2.0    // le Correcteur proportionnel


#define   A_step  10.0    // Amplitude
#define   c_step  200     // Période = 2*c_step*T_ms

double Wn=2.0*PI*Fn;
double T_s=(double)T_ms/1000.0;


double x_nn=0.0;      // Consigne (entrée)
double y_n[2];        // "0" Non corrigé, "1": Corrigé
double eps_n[2];      // Erreur
double y_capt[2];    // Sortie du capteur
double y_corr[2];    // Sortie du correcteur

// Variables internes des systèmes
double x1[2], y1[3]; // Système Non Corrigé
double x2[2], y2[3]; // Système Corrigé

// Paramètres de l'échelon
unsigned long c=0; // Compteur (période)
bool Step=false;

void setup()
{
```

```
  // Port série de la réponse du système
  Serial.begin(9600);
}

void loop()
{
  // 1. La consigne (l'entrée) x(n) pour les deux systèmes
  c++; c=c%c_step;
  if(!c)
  {
    Step=!Step;
    c=0;
  }
  x_nn=A_step*(double)Step; // Réponse à un échelon x(n)=cte
  //x_nn=(double)c;           // Réponse à une rampe x(n)=n
  // 2. Sortie du capteur: Retour unitaire
  y_capt[0]=y_n[0];
  y_capt[1]=y_n[1];
  // 3. Soustracteur: Calcul de l'erreur eps(n)
  eps_n[0]=x_nn-y_capt[0];
  eps_n[1]=x_nn-y_capt[1];
  // 4. Correcteur
  y_corr[0]=eps_n[0];     // Système non Corrigé
  y_corr[1]=Kc*eps_n[1];  // Système Corrigé

  // 5. Calcul de la sortie: Système non corrigé
  y_n[0]=Sys2All(y_corr[0], x1, y1, Zeta, Wn, K, T_s);

  // 5. Calcul de la sortie: Système corrigé
  y_n[1]=Sys2All(y_corr[1], x2, y2, Zeta, Wn, K, T_s);

  // Affichage des signaux
  Serial.print(x_nn); Serial.print(",");
  Serial.print(y_n[0]); Serial.print(",");
  Serial.println(y_n[1]);

  // Période d'échantillonnage
  delay(T_ms);
}



double Sys2All(double x_nn, double *x, double *y, double zeta, double wn, double k,
double T)
```

```
{
  // Paramètre du système
  double a1=2.0*zeta/wn;
  double a2=1.0/(wn*wn);

  const double b0=(a1/(2.0*T))+(a2/(T*T));
  const double b1=-2.0*a2/(T*T);
  const double b2=(-1.0*a1/(2.0*T))+(a2/(T*T));
  const double b[3]={b0,b1,b2};

  // Variables de l'entrée et la sortie
  double y_nn=0.0;
  // Calcul de la nouvelle sortie
  y_nn= -(y[0]*(1.0+b[1]))-(y[1]*b[2])+(k*x[0]); // y[1]: y(n-2), y[0]: y(n-1)
  y_nn/=b[0];
  // Mise à jour de la sortie
  y[1]=y[0];
  y[0]=y_nn;

  // Mise à jour de la sortie
  x[0]=x_nn;
  // Renvoie du résultat
  return y_nn;
}
```

Accueil Asservissement avec Arduino

Click to rate this post!
[Total: 1 Average: 5]