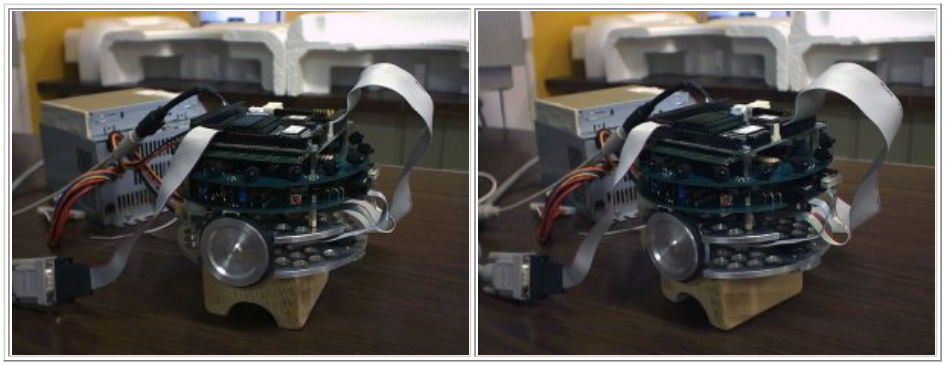
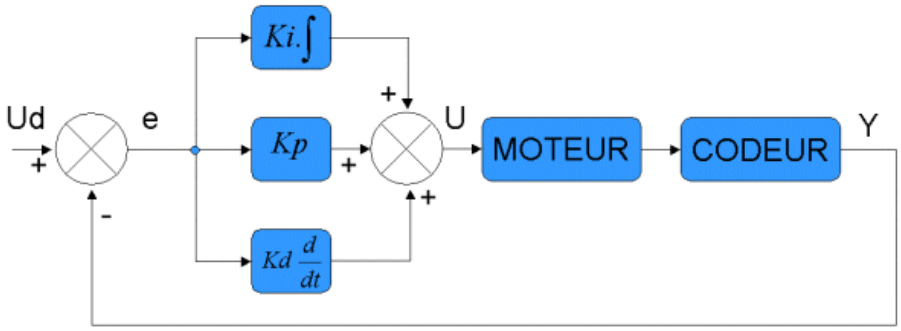
L'ASSERVISSEMENT PID

Vous avez sûrement déjà remarqué qu'il ne suffit pas d'appliquer une tension constante à un moteur pour qu'il tourne à la vitesse voulue. Et même avec des moteurs de bonne qualité, la vitesse dépend souvent de la charge des accus, du poids du robot et d'un tas d'autres paramêtres qui varient selon les circonstances. Pour pouvoir avoir une bonne répétabilité des trajectoire il faut réaliser un asservissement. Cela consiste à récupérer une information sur la vitesse du moteur (par exemple grâce aux codeurs) et s'en servir pour ajuster la tension de commande. Il existe un tas de méthodes d'asservissement, nous présentons ici la plus connue de toutes : le PID (Proportionnel Intégrale Dérivée).

Principe



Pour pouvoir tester notre asservissement, il nous faut d'abord une plateforme de dévelopement. Nous allons utiliser le robot Type 1 (photos ci-dessus) pour programmer notre asservissement. Mais pour commencer un brin de théorie. Le schéma de principe est représenté ci-dessous :

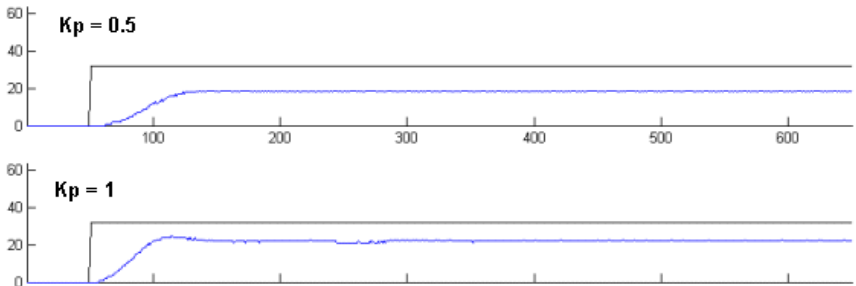


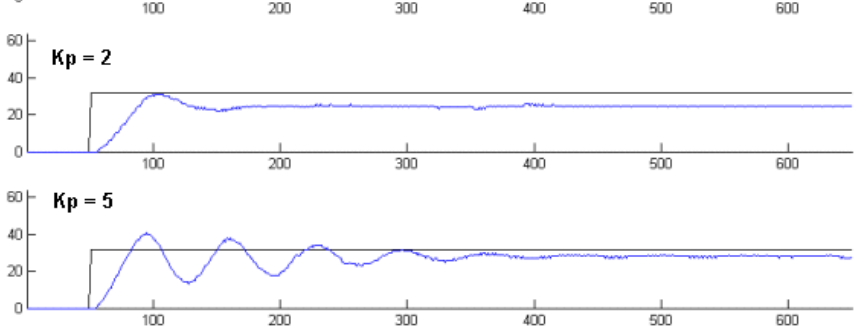
- Ud : vitesse désirée
- e : erreur entre la vitesse désirée et la vitesse réelle
- U : Consigne appliquée au moteur
- Y : Vitesse réelle

Nous avons bien un schéma en boucle fermée puisque la sortie du système est réinjectée dans l'entrée. Les trois blocs bleus correspondent chacun à une fonctionnalitée : Ki est le coefficient intégral, Kp le coefficient proportionnel et Kd le coefficient dérivé. Chacun de ces coefficients a une fonctionnalité bien précise. Commençons par le plus simple : le proportionnel.

Asservissement P

L'asservissement proportionnel est essentiel au fonctionnement du PID. Il permet essentiellement de donner de la puissance au moteur. Voici quelques résultats d'expérimentations avec Ki=0 et Kd=0.

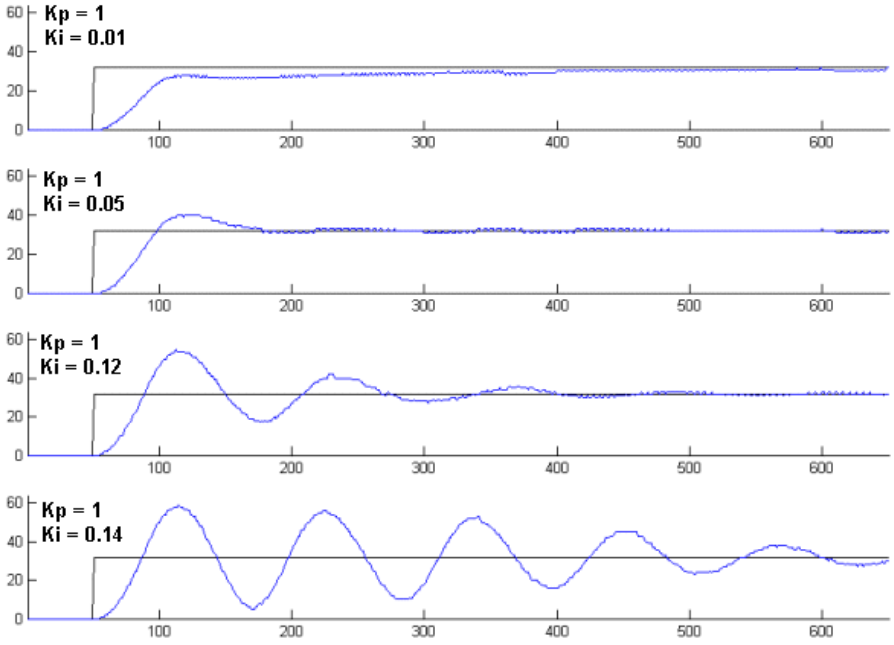




La courbe noire représente la consigne que l'on désire atteindre (Ud) c'est un échelon de valeur 32. La courbe bleue représente la vitesse réelle du système (Y). Comme vous pouvez le constater, plus Kp est grand, plus le système converge vite vers sa valeur finale. Mais en contre-partie, pour des valeurs de Kp trop grandes, le système oscille. Mais là n'est pas notre plus gros problème, en effet sur ces courbes on voit nettement que la vitesse du moteur n'atteind jamais la vitesse désirée. C'est ce que l'on appelle l'erreur statique, elle correspond à la différence entre la vitesse réelle et la vitesse désirée en régime établie (une fois que le système s'est stabilisé). Pour compenser cette erreur statique, on rajoute le terme intégral.

Asservissement PI

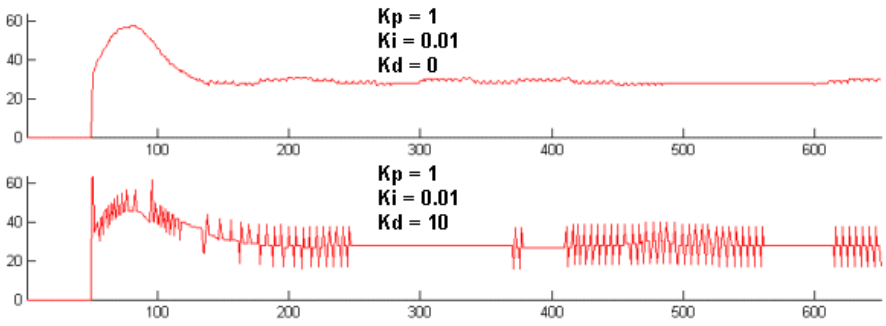
Le correcteur intégral sert principalement à supprimer l'erreur statique. L'idée principale est de "charger" ou intégrer l'erreur depuis le début et d'ajouter cette erreur à la consigne jusqu'à ce qu'elle devienne nulle. Lorsque cette erreur est nulle, le terme intégral se stabilise et il compense parfaitement l'erreur entre la consigne et la vitesse réelle. Démonstration en image :



On voit cette fois-ci que le terme integral a bien fonctionné et que l'erreur statique est nulle. On constate aussi que plus le gain Ki est grand, plus le système converge vite. En revanche, plus Ki est grand, plus le système oscille et plus le dépassement est grand. Sur des asservissements en position le terme dérivé permet de diminuer, le dépassement et les oscillations. Nous allons voir que dans notre cas, nous avons quelques problèmes.

Asservissement PID

J'ai donc rajouté le terme dérivé, comme tout bon livre d'automatique le préconise, et voici les résultats :



Cette fois ci, j'ai choisi de représenter la tension appliquée au moteur (U), le problème est bien plus flagrand ici. Vous voyez que lorsque l'on ajoute le terme dérivé, la commande est extrémement bruitée. En réalité, on somme à la commande la dérivée de la vitesse réelle, que l'on multiplie par un gain Kd. La dérivée d'une vitesse est une accélération, cela signifie que l'on amplifie tous les bruits d'accélération et en plus, on les multiplient par un gain Kd.

Donc rajoute du bruit amplifié dans la commande. Alors pourquoi mettent t-ils ce terme "dérivée" dans les livres ? Simplement parce que ce bruit est typique des asservissement en vitesse. Sur des asservissement en position, ce problème est moins fréquent, car la dérivée de la position est la vitesse.

Ici, le problème était visible, parce que nous avons réalisé toutes ces courbes sur un système réel (le robot Type 1). Si nous avions simulé ces résultats, l'accélération n'aurait pas été bruitée et les courbes auraient été parfaites. Pour implémenter ce PID sur notre robot, il a fallu programmer en numérique une dérivée et une intégrale.

PID numérique et réglage

Notre premier problème est de programmer le terme intégral. Il existe une méthode qui consiste à estimer la surface entre deux échantillons, mais dans notre cas, nous avons des problèmes dus aux bruits et à la fréquence d'échantillonnage. Alors, il faut sommer toutes les erreurs depuis de début.

Notre second problème est de programmer le terme dérivé. La méthode ici est de calculer la pente entre deux échantillons de e. Théoriquement cela revient à faire le calcul suivant :

$$\frac{de}{dt} = \frac{e(t) - e(t-1)}{\Delta t}$$

Avec Δt la période d'échantillonnage. Pour faire un asservissement numérique, il faut impérativement avoir une période fixe. Pour gagner du temps de calcul, généralement on intégre directement Δt dans le gain Kd.

Voici donc l'algorithme que nous avons utilisé dans notre robot :

```
Ti = 0;                         /* Initialise le terme intégral */
for ( i=1 ; i <= Nb_cycles ; i++ )
{
    Init_timers (1);            /* Initialise le timer à notre période : 1 ms */
    Y = Codeur ();              /* Stoque la valeur du codeur dans Y */
    Rst_codeur ();              /* Remet le codeur à zéro */
    Ud = Consigne ();           /* Mise à jour de la consigne */
    e_old=e;                    /* Stoque e(t-1) */
    e = Ud - Y;                 /* Calcul de l'erreur */
    Ti = Ti + e;                /* Calcul de l'intégral */

    /* Calcul du PID */

    U = Kp * e                  /* Terme proportionnel */
    + Ki * Ti                   /* Terme intégral */
    + Kd * ( e - e_old ) ;      /* Terme dérivé */

    Moteur (U);                 /* On applique la consigne */
    while ( ! end_timer() );    /* On attend la fin de la période */

}
```

Pour le réglage des gains Kp, Ki et Kd, il existe différentes méthodes. Ici nous l'avons fait de manière empirique et le résultats est plutôt bon. L'inconvénient de ces méthodes est qu'elles demandent un modèle du système, ce qui est généralement dur à produire. Il existe aussi des composants programmables qui permettent de faire directement l'asservissement numérique (HCTL 1100, LM629). Ils sont exactement basés sur la méthode présentée ici.

Quelques liens

- HCTL 1100 (datasheet) [477 Ko]
- LM 629 (datasheet) [631 Ko]
- PID Tutorial for Matlab (Carnegie Mellon)
- Tutorial complet sur les PID.

Contact

Pour toutes questions envoyer moi un mail: Sinclair ICQ# : 144345434