Discretization of Continuous Controllers

Thao Dang

VERIMAG, CNRS (France)

Robot Controller Discretization - Sampling period

```
%la fonction de transfert C(s) = ki_teta/s + kp_teta
%la fonction de transfert H_teta(s) = 1/(ls)
%la fonction de transfert en boucle ouvert
%H_BO(s) = (kp_teta*s + ki_teta)/(ls^2 + ls)

sysbo = tf([0 kp_teta ki_teta], [ 1 1 0])

[g p f wc] = margin(sysbo)
%wc est la "crossover frequency"

%T = periode d'echantillonnage
T=0.05/wc
```

Robot Controller Discretization - Filter

Anti-aliasing Filter has little effect. If necessary increase ws or adjust $C(s)$ to obtain nearly the same performance as the continuous design.

```
%frequence d'echantillonage
ws = 2*pi/T;
%frequence de Nyquist
wn= 2*ws;

gn= 0.1;
xi_f = 0.707;
wf = wn*sqrt(gn);

filtre = tf([ wf^2 ], [ 1  2*xi_f*wf wf^2])
sysbo1 = sysbo*filtre;
[g p f wc1] = margin(sysbo1)
```

Sampling period

- It is also possible to choose a sample frequency, ws, based on the closed-loop bandwidth wd of the continuous system.
- Because of the approximate nature of this design method, it is best to choose $10 < ws/wd < 20$

PID

- Proportionnel (K): plus l'erreur est grande et plus la correction est grande. K règle le facteur multiplicatif sur l'erreur.
- Intégrale (Ti): la correction augmente si l'erreur reste constante. Ti règle la vitesse à laquelle est intégrée l'erreur. Plus Ti est petit, et plus on fait croître rapidement le signal de correction.
- Dérivée (Td et N): la correction est temporairement importante à chaque "saut" du signal d'erreur. Td règle le temps d'application de ce petit "plus" sur la correction globale et N sa hauteur. Ce paramètre doit évidemment être utilisé avec précaution: tout bruit hautes fréquences sur la mesure se traduit par des variations du même ordre sur la commande.

PID Tuning

```
>> sys = tf([ 1], [ 0.2 0]);
>> [C_pi_fast,info] = pidtune(sys, 'pi')
Continuous-time PI controller in parallel form:
           1
Kp + Ki * ---
           s
with Kp = 0.17321, Ki = 0.1

info =
                Stable: 1
    CrossoverFrequency: 1
           PhaseMargin: 60.0000
```

PID Tuning using Matlab

Typical PID tuning objectives include:

- **Stability**: output remains bounded for bounded input.
- **Adequate performance**: track reference changes and suppress disturbances as rapidly as possible. The larger the loop bandwidth (the first frequency at which the open-loop gain is unity), the faster the controller responds to changes in the reference or disturbances in the loop.
- **Adequate robustness**: enough phase margin and gain margin to allow for modeling errors or variations in system dynamics.
- Matlab algorithm for tuning PID controllers: balance performance (response time) and robustness (stability margins).
- By default, the algorithm chooses a crossover frequency (loop bandwidth) based upon the plant dynamics, and designs for a target phase margin of 60 degrees.
- If you specify the crossover frequency using wc or the phase margin using **pidtuneOptions**, the algorithm computes PID gains that best meet those targets.

Réglage de PID

Il est possible d'appliquer une procédure de réglage itérative de type essai erreur:

$$C(s) = K(1 + \frac{1}{Ti\,s} + \frac{Td\,s}{1 + \frac{Td\,s}{N}})$$

- Réglage de départ: K faible, action I et D à zéro (Ti maxi Td=0).
- Appliquer un échelon de consigne.
- Si le gain est suffisamment faible, la réponse doit être bien amortie.
- Multiplier K par 2, appliquer de nouveau un échelon de consigne et recommencer jusqu'à ce que des dépassements apparaissent sur la sortie. Régler alors K à la moitié de la valeur obtenue à l'apparition de dépassements.
- Appliquer la même procédure pour Ti en le réduisant à chaque fois d'un facteur 2, puis revenir à 2 fois la valeur provoquant les dépassements
- Même chose avec Td et revenir au tiers de la valeur provoquant les dépassements.

http://cerig.efpg.inpg.fr/tutoriel/automatique/page06.htm