Discretization of Continuous Controllers

Thao Dang

VERIMAG, CNRS (France)

Discretization of Continuous Controllers

- One way to design a computer-controlled control system is to make a continuous-time design and then make a discrete-time approximation of this controller $\Rightarrow$ Analog Design Digital Implementation
- The computer-controlled system should now behave as the continuous-time system

Discretization of Continuous Controllers

- One way to design a computer-controlled control system is to make a continuous-time design and then make a discrete-time approximation of this controller $\Rightarrow$ Analog Design Digital Implementation
- The computer-controlled system should now behave as the continuous-time system

Discretization of Continuous Controllers

- One way to design a computer-controlled control system is to make a continuous-time design and then make a discrete-time approximation of this controller $\Rightarrow$ Analog Design Digital Implementation
- The computer-controlled system should now behave as the continuous-time system
- This is crucially dependent on choosing fairly short sampling periods.

Difference Approximations (1)

- When the continuous-time controller is specified as a transfer function $C(s)$, it is natural to look for methods that will transform the continuous transfer function $C(s)$ to a pulse transfer function $C_d(z)$ so that the corresponding behaviors of the two systems are close to each other.
- z and s are related as $z = exp(sT)$, where T is the sampling period.

Difference Approximations (2)

The difference approximations correspond to the series expansions

- $z = e^{sT} \approx 1 + sT$ (Forward difference or Euler's method)
- $z = e^{sT} \approx \frac{1}{1-sT}$ (Backward difference)
- $z = e^{sT} \approx \frac{1+sT/2}{1-sT/2}$ (Trapezoidal method, or Tustin's approximation)

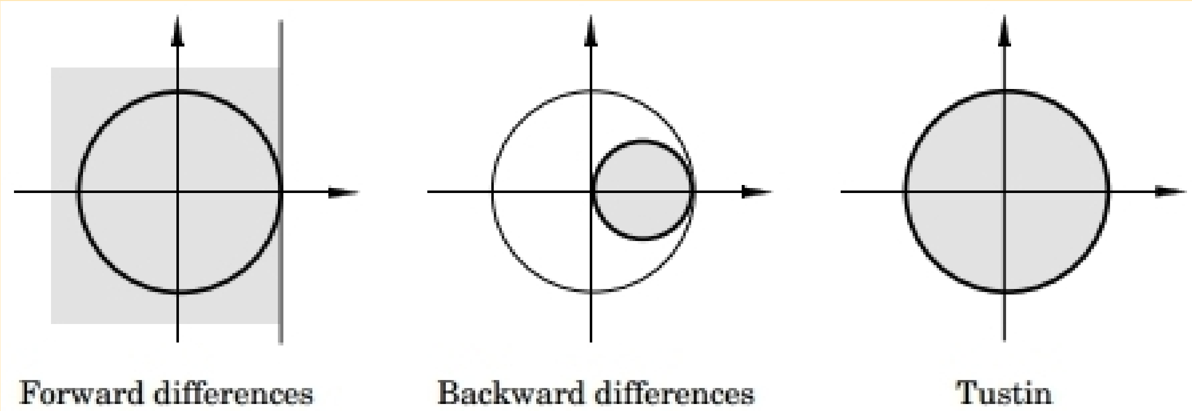
Computing transfer function $C_d(z)$

To calculate $C_d(z)$ we substitute s in $C(s)$ with the following:

- $s \approx \dfrac{z-1}{T}$ (Forward difference or Euler's method)
- $s \approx \dfrac{z-1}{zT}$ (Backward difference)
- $s \approx \dfrac{2}{T}\dfrac{z-1}{z+1}$ (Trapezoidal method, or Tustin's approximation)

Stability

the stability region (corresponding to the left half-plane $Re(s) \leq 0$) in the s-plane is mapped on the z-plane.



Stability - Remarks

- Forward-difference approximation: it is possible that a stable continuous-time system is mapped into an unstable discrete-time system.
- Backward approximation: a stable continuous-time system will always give a stable discrete-time system.
- Tustin's approximation: has the advantage that the left half s-plane is transformed into the unit disc in the z-plane.

Selection of Sampling Period and Anti-aliasing Filters

Choice of sampling rates and anti-aliasing flters are important

- Preserve stability
- Preserve performance

Anti-aliasing Filters: Example

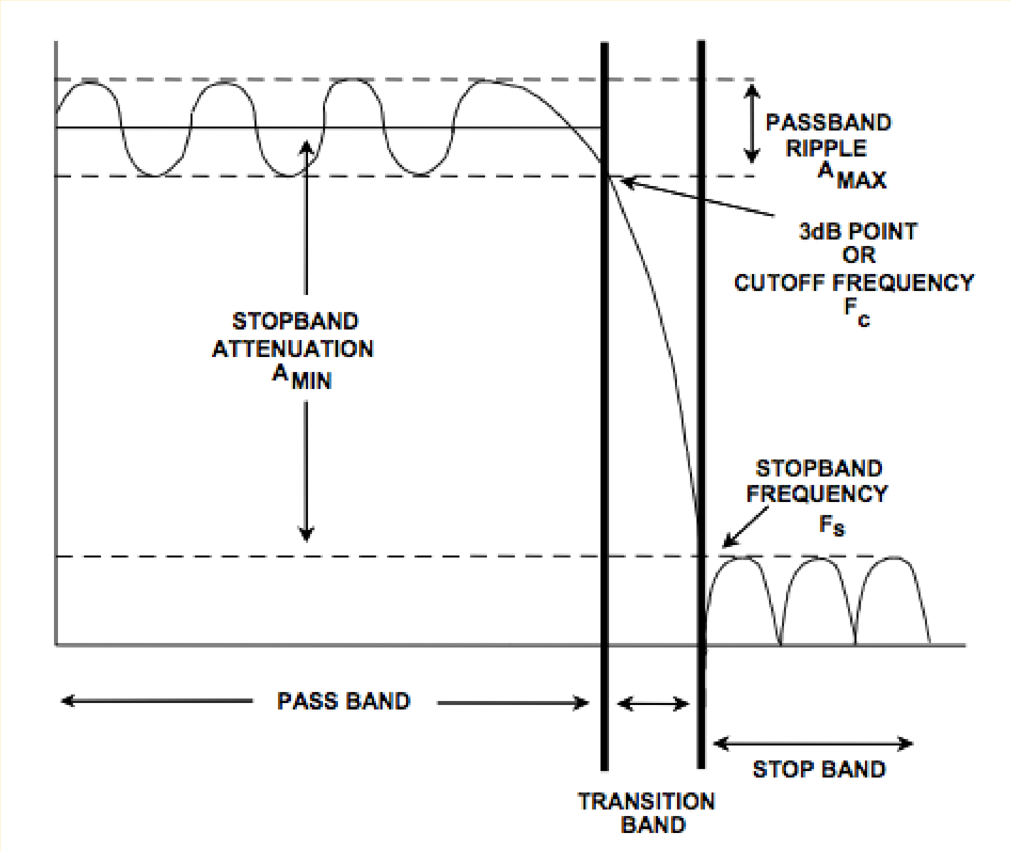
Example of a second order Butterwworth filter

$$G(s) = \frac{\omega_o^2}{s^2 + 2\sqrt{2}\omega_o s + \omega_o^2}$$

ω_o is the cut-off frequency.

The Butterworth filter: no ripple in the pass band or the stop band (maximally flat filter), at the expense of a relatively wide transition region from pass band to stop band

Anti-aliasing Filters: Example



The stop band frequency is generally $\frac{1}{2}$ the sample rate, i.e. $\frac{1}{2T_e}$

Designing Digital Anti-Aliasing Filters using Bilinear Transformation

Given a second order Butterworth filter, ω_o is the cut-off frequency:

$$G(s) = \frac{\omega_o^2}{s^2 + 2\sqrt{2}\omega_o s + \omega_o^2}$$

To discretize the filter, we can use the bilinear transformation from the s-domain to the $z-$-domain, T_e is the sampling period.:

$$s = \frac{2}{T_e}\,\frac{1 - z^{-1}}{1 + z^{-1}}$$

Relation between the analogue frequency ω_a and digital frequency ω_d:

$$\omega_a = \frac{2}{T_e}\, tan(\frac{\omega_d T_e}{2})$$

If we want the cutoff frequency after discretization to be ω_{do}, then from the above formula we can calculate the corresponding (analogue) ω_o. Then we apply the relation between s and z to compute the transfer function in z from $G(s)$

Designing Digital Anti-Aliasing Filters using matlab

Matlab offers functions to design digital digital anti-aliasing filter directly

Wp: Passband corner frequency Wp, the cutoff frequency, is a scalar or a two-element vector with values between 0 and 1, with 1 corresponding to the normalized Nyquist frequency, π radians per sample.

(the Nyquist frequency $\omega_N = 2\omega_e$)
Ws: Stopband corner frequency Ws, is a scalar or a two-element vector with values between 0 and 1, with 1 corresponding to the normalized Nyquist frequency.

Rp: Passband ripple in decibels.

Rs: Stopband attenuation in decibels. This value is the number of decibels the stopband is down from the passband.

Designing Digital Anti-Aliasing Filters using matlab

http://www.mathworks.fr/fr/help/signal/ref/buttord.html

Ex: For a signal sampled at $\omega_e = 1/T_e = 1000$ Hz, design a lowpass filter with no more than 3 dB of ripple in the passband from 0 to 40 Hz, and at least 60 dB of attenuation in the stopband. Plot the filter's frequency response.

```
Wp = 40/500; Ws = 150/500;
[n,Wn] = buttord(Wp,Ws,3,60);
% Returns n = 5; Wn=0.0810;
[b,a] = butter(n,Wn);
freqz(b,a,512,1000);
title('n=5 Butterworth Lowpass Filter')
```

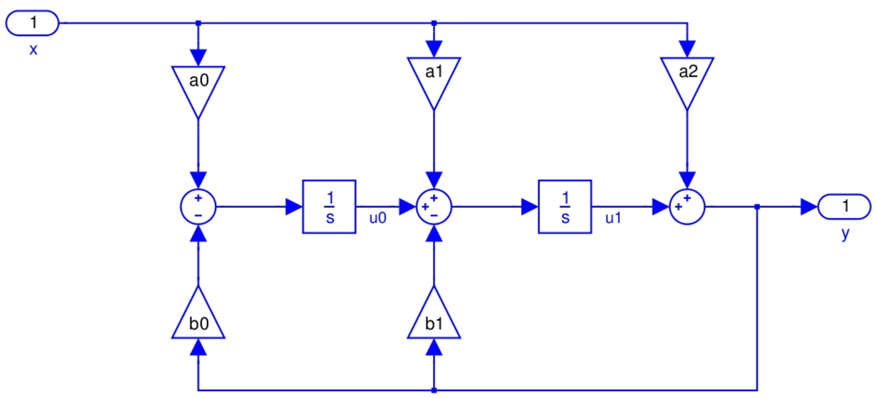
$$G(z) = \frac{b(1) + b(2)z^{-1} + \ldots b(n+1)z^{-n}}{1 + a(2)z^{-1} + \ldots a(n+1)z^{-n}}$$

Implementing in Simulink

Systèmes rationnels

Exemple :

$$\frac{a_2 s^2 + a_1 s + a_0}{s^2 + b_1 s + b_0}$$



Same principle for transfer functions in z!

Exercise: Application to LEGO robots

1. Compute the continuous-time transfer function of the open loop $F_{BO}(s)$

2. Estimate the crossover frequency ω_c of $F_{BO}(s)$ using the function "margin" in matlab.

3. Choose sampling period T_e according to the rule: $\omega_c T_e$ **is between** 0.05 **and** 0.14

4. Discretize the controllers using one of the approximation methods and the chosen sampling period. (Note that instead of replacing the whole continuous-time controller with its discretized version, we can replace only its components containing continuous-time blocs, such as the integrators $\frac{1}{s}$)

5 Add the digital anti-aliasing filter