Automatic Control 2

Sampling

Prof. Alberto Bemporad

University of Trento

Academic year 2010-2011
Time-discretization of continuous-time controllers

- We have designed an analog controller $C(s)$ in the \textit{continuous-time} domain (by loop shaping, pole-placement, etc.)
- We want to implement $C(s)$ in digital form (for example, in a microcontroller). We need to convert the design to \textit{discrete-time}
Relations between continuous/discrete-time signals

Let the digital controller operate with sampling time $T$

- tracking error $\tilde{e}(k) \triangleq e(kT)$ for $k \in \{0, 1, \ldots\}$
- control input $u(t) = \tilde{u}(k) \triangleq u(kT)$ for $t \in [kT, (k+1)T)$
Alternative scheme: more often the reference signal $r(t)$ is already given in digital form by its samples $\hat{r}(k)$, $\hat{r}(k) = r(kT)$, $k = 0, 1, \ldots$, $T=$sampling time

**Problem**

How to synthesize a *discrete-time* control law $\hat{C}(z)$ that in closed-loop behaves as the given *continuous-time* controller $C(s)$?
Time-discretization

- Consider the continuous-time controller $C(s)$

  $$ u(t) = C(s)e(t) = \frac{N(s)}{D(s)}e(t) = \frac{b_{n-1}s^{n-1} + \ldots + b_0}{s^n + a_{n-1}s^{n-1} + \ldots + a_0}e(t) $$

- We can look at the control law as at a differential equation linking $u(t)$ to $e(t)$

  $$ \frac{d^n}{dt^n}u(t) + a_{n-1}\frac{d^{n-1}}{dt^{n-1}}u(t) + \ldots + a_0u(t) = b_{n-1}\frac{d^{n-1}}{dt^{n-1}}e(t) + \ldots + b_0e(t) $$

- A **time-discretization** of $u(t) = C(s)e(t)$ is nothing else than a **numerical approximation** (with constant **integration-step** $T$)

  $$ \tilde{u}(k) = \tilde{C}(z)\tilde{e}(k) $$

  linking the input samples $\tilde{u}(k) \triangleq u(kT)$ to the error samples $\tilde{e}(k) \triangleq e(kT)$

- There are many numerical methods to integrate a differential equation (constant step, variable step, $n$-th order approximations, etc.)
Numerical integration in Simulink

**Fixed-step integration panel**

**Variable-step integration panel**
Example of time-discretization

- Consider the continuous-time controller
  \[ u(t) = \frac{1}{s + 2} e(t) \implies \frac{d}{dt} u(t) + 2u(t) = e(t) \]

- Use *Euler method* to approximate the derivative
  \[ \frac{d}{dt} u(t) \approx \frac{u((k + 1)T) - u(kT)}{T} = \frac{1}{T} (\tilde{u}(k + 1) - \tilde{u}(k)) \]

- Recall that \( z \) represents the unit-shift operator \( z\tilde{u}(k) = \tilde{u}(k + 1) \)
  \[ \tilde{u}(k) = \frac{1}{\left( \frac{z-1}{T} \right) + 2} \tilde{e}(k) \triangleq \tilde{C}(z)\tilde{e}(k) \]

- Note that formally this is equivalent to replace \( s = \left( \frac{z-1}{T} \right) \) in \( C(s) \) to get \( \tilde{C}(z) \)
Finite-difference approximation of the controller

Consider a state-space realization of \( C(s) \)

\[
\begin{align*}
\frac{dx_c}{dt} &= A_c x_c + B_c e \\
u &= C_c x_c + D_c e
\end{align*}
\]

\( C(s) = C_c (sI - A_c)^{-1} B_c + D_c \)

Integrate between time \( kT \) and \( (k+1)T \)

\[
x_c((k+1)T) - x_c(kT) = \int_{kT}^{(k+1)T} \frac{dx_c(\tau)}{dt} d\tau
\]

gives

\[
x_c((k+1)T) - x_c(kT) = A_c \int_{kT}^{(k+1)T} x_c(\tau) d\tau + B_c \int_{kT}^{(k+1)T} e(\tau) d\tau
\]

Unfortunately, in general both \( x(\tau) \) and \( e(\tau) \) are not constant between consecutive sampling instants, so we can’t use exact discretization (i.e., the exponential matrix \( e^{A_c T} \))
Approximation of the integral of $\dot{x}_c(\tau)$

**Forward Euler method**

$$
\int_{kT}^{(k+1)T} \dot{x}_c(\tau) d\tau \approx T \dot{x}_c(kT) \\
\Rightarrow s = \frac{z - 1}{T}
$$

**Backward Euler method**

$$
\int_{(k-1)T}^{kT} \dot{x}_c(\tau) d\tau \approx T \dot{x}_c(kT) \\
\Rightarrow s = \frac{1 - z^{-1}}{T}
$$

**Trapezoidal rule**

$$
\int_{kT}^{(k+1)T} \dot{x}_c(\tau) d\tau \approx \frac{T}{2} \left[ \dot{x}_c((k+1)T) + \dot{x}_c(kT) \right] \\
\Rightarrow s = \frac{2(z - 1)}{T(z + 1)}
$$
Finite-difference approximation of the controller

\[
(z - 1) x_c = T(A_c x_c + B_c e) \quad \rightarrow \quad \left[ \left( \frac{z - 1}{T} \right) I - A_c \right] x_c = B_c e
\]

\[
\tilde{C}(z) = \frac{U(z)}{E(z)} = C_c \left[ \left( \frac{z - 1}{T} \right) I - A_c \right]^{-1} B_c + D_c = C \left( \frac{z - 1}{T} \right)
\]

Forward Euler method

\[
(1 - z^{-1}) x_c = T(A_c x_c + B_c e) \quad \rightarrow \quad \left[ \left( \frac{z - 1}{zT} \right) I - A_c \right] x_c = B_c e
\]

\[
\tilde{C}(z) = \frac{U(z)}{E(z)} = C_c \left[ \left( \frac{1 - z^{-1}}{T} \right) I - A_c \right]^{-1} B_c + D_c = C \left( \frac{1 - z^{-1}}{T} \right)
\]

Backward Euler method

\[
\tilde{C}(z) = C_c \left[ \left( \frac{2(z - 1)}{T(z + 1)} \right) I - A_c \right]^{-1} B_c + D_c = C \left( \frac{2(z - 1)}{T(z + 1)} \right)
\]

Tustin's method
Finite-difference approximation of the controller

Formally, we just replace $s$ in $C(s)$ with the corresponding function of $z$:

- **forward Euler method**

  \[
  s = \frac{z - 1}{T} \quad \rightarrow \quad \dot{x}_c(kT) \approx \frac{x_c(((k + 1)T) - x_c(kT)}{T}
  \]

- **backward Euler method**

  \[
  s = \frac{1 - z^{-1}}{T} \quad \rightarrow \quad \dot{x}_c(kT) \approx \frac{x_c(kT) - x_c((k - 1)T)}{T}
  \]

- **Trapezoidal rule (Tustin’s method)**

  \[
  s = \frac{2(z - 1)}{T(z + 1)} \quad \text{(bilinear transformation)}
  \]

Note that all three methods preserve the DC gain: $z = 1 \rightarrow s = 0$
(no approximation error exists in constant steady-state !)

Compare to the exact discretization method: $\tilde{A}_c = e^{TA_c}$, $\tilde{B}_c = \int_0^T e^{tA_c}dtB_c$,
$\tilde{C}_c = C_c$, $\tilde{D}_c = D_c$
Relations between poles in $s$ and $z$

- What is the relation between the poles $s_i$ of $C(s)$ and the poles $z_i$ of $\tilde{C}(z)$?

  - forward Euler differences:
    \[
    \frac{z_i - 1}{T} = s_i \Rightarrow z_i = 1 + Ts_i
    \]

  - backward Euler differences:
    \[
    \frac{z_i - 1}{Tz_i} = s_i \Rightarrow z_i = \frac{1}{1 - s_iT}
    \]

  - Tustin’s method:
    \[
    \frac{2(z_i - 1)}{T(z_i + 1)} = s_i \Rightarrow z_i = \frac{1 + s_iT/2}{1 - s_iT/2}
    \]

  - Exact method: $z_i = e^{s_iT}$

Note that the three (approximate) integration methods approximate the function $z = e^{sT}$ by a rational function.
Relations between poles in $s$ and $z$

forward Euler differences

stable poles in $s$ may be mapped to unstable poles in $z$

backward Euler differences

stable poles in $s$ are also stable poles in $z$, marginally stable poles may become asymptotically stable

Tustin’s rule

stable poles in $s$ are also stable poles in $z$
### Time-discretization in MATLAB

**MATLAB**

C2D Conversion of continuous-time models to discrete time.

```matlab
SYSD = C2D(SYSC, TS, METHOD) converts the continuous-time LTI model SYSC to a discrete-time model SYSD with sample time TS. The string METHOD selects the discretization method among the following:

- `'zoh'` Zero-order hold on the inputs.
- `'foh'` Linear interpolation of inputs (triangle appx.).
- `'tustin'` Bilinear (Tustin) approximation.
- `'prewarp'` Tustin approximation with frequency prewarping. The critical frequency \( W_c \) is specified as fourth input by `C2D(SYSC, TS, 'prewarp', Wc)`.
- `'matched'` Matched pole-zero method (for SISO systems only).

The default is `'zoh'` when `METHOD` is omitted.
```
Sampled-data systems

- **continuous-time control system** (Laplace transform/frequency analysis)

\[ r(t) \rightarrow e(t) \rightarrow C(s) \rightarrow u(t) \rightarrow G(s) \rightarrow y(t) \]

\[ t = \text{continuous time} \]

- **discrete-time control system** (Z-transform)

\[ \tilde{r}(k) \rightarrow \tilde{e}(k) \rightarrow \tilde{C}(z) \rightarrow \tilde{u}(k) \rightarrow \tilde{G}(z) \rightarrow \tilde{y}(k) \]

\[ k = \text{discrete-time step counter} \]

- **sampled-data system**

\[ r(t) \rightarrow e(t) \rightarrow \frac{A}{D} \rightarrow \tilde{C}(z) \rightarrow \tilde{u}(k) \rightarrow \frac{D}{A} \rightarrow G(s) \rightarrow y(t) \]

\[ \text{continuous-time and discrete-time signals} \]
Why analyzing sampled-data systems

- The model $G(s)$ of the process to be controlled is (very often) given in continuous time
  - differential equations
  - actuator signals (electrical voltages to motors, etc.) vary continuously in time
  - output variables (temperature, pressure, position, etc.) vary continuously in time

- On the other hand, the controller is (almost always) implemented in digital form, $\tilde{C}(z)$:
  - cheaper to implement (computer code)
  - easier to reconfigure
  - can exploit time-sharing (multiple controllers on the same hardware)
  - much more versatile (arbitrary nonlinear control laws)

- Hence the need to analyze sampled-data systems, namely a continuous process in closed loop with a digital controller
Ways to analyze sampled-data systems

1. Convert the system \( G(s) \) to its discrete-time equivalent \( \tilde{G}(z) \) (use for instance exact sampling), ignoring its inter-sampling behavior (controller’s point of view or “stroboscopic model”) \( \implies \) discrete-time analysis

2. Model the digital controller in continuous time

\[
U(s) \approx \frac{1 - e^{-sT}}{sT} \tilde{C}(e^{sT})E(s) \quad \text{(this can be shown ...)}
\]

(process’ point of view) \( \implies \) continuous-time analysis

3. Use numerical simulation (e.g., Simulink) (only provides an answer for a certain finite set of initial states)
Choosing the sampling time

- How to choose the sampling time $T$?
- Example: sampling of $\sin(t)$

- How can we say that a sampling time is “good”? 
Nyquist-Shannon sampling theorem

Claude Elwood Shannon (1916–2001)

Sampling theorem

Let $e(t)$ be a signal and $E(j\omega) = \int_{-\infty}^{+\infty} e(\tau)e^{-j\omega \tau} d\tau$ its Fourier transform $\mathcal{F}[e]$. Let $E(j\omega) = 0$ for $|\omega| \geq \omega_{\text{max}}$. For all $T$ such that $\omega_N \triangleq \frac{\pi}{T} > \omega_{\text{max}}$

$$e(t) = \sum_{k=-\infty}^{+\infty} e(kT) \frac{\sin(\omega_N(t-kT))}{\omega_N(t-kT)}$$

$\omega_N$ is called the Nyquist frequency, equal to half the sampling frequency $\omega_s = \frac{2\pi}{T}$
Shannon’s reconstruction

- We can look at sampling as at the modulation of a Dirac comb $\sum_{k=-\infty}^{\infty} \delta(t - kT)$
- Let’s go to Fourier transforms

$$\mathcal{F} \left[ \sum_{k=-\infty}^{\infty} e(kT)\delta(t - kT) \right] = \sum_{k=-\infty}^{\infty} e(kT)e^{-jkT\omega} = \frac{1}{T} \sum_{k=-\infty}^{\infty} E(j(\omega + k\omega_s)) \triangleq \frac{1}{T}E_s(j\omega)$$

- $E_s(j\omega)$ is the periodic repetition of $E(j\omega)$ with period $\omega_s$
- The ideal low-pass filter can only reconstruct $e(t)$ if $E(j\omega) = 0$ for $|\omega| \geq \omega_{\text{max}}$
Aliasing

\[ E_s(j\omega) = \frac{1}{T} \sum_{k=-\infty}^{+\infty} E(j(\omega + k\omega_s)) \]

- The value \( E_s(j\omega_1) \) at a certain frequency \( \omega_1 \) not only depends on \( E(j\omega_1) \), but also on all the (infinite) contributions \( E(j(\omega_1 \pm k\omega_s)) \)
- The frequency \( \omega_1 \pm k\omega_s \) is called an alias of \( \omega_1 \)
- Example: consider the signals \( \sin(\omega_1 t) \) and \( \sin(\omega_2 t) \) and let the sampling time \( T = 1 \) s

\[
\omega_1 = \frac{2\pi}{10} \\
\omega_2 = \frac{2\pi}{T} + \frac{2\pi}{10} = \frac{22\pi}{10}
\]
Aliasing

- What if \( e(t) \) has spectral contributions \( E(j\omega) \neq 0 \) for \( \omega > \omega_N = \frac{\pi}{T} \)?

- **Aliasing** is the phenomenon for which frequencies \( \omega > \omega_N \) make contributions in the frequency range \([-\omega_N, \omega_N]\).

- Under aliasing conditions it is impossible to reconstruct the original signal.

- An **anti-aliasing filter** is a low-pass filter that removes from \( e(t) \) its high-frequency contents before sampling it. It partially mitigates the problem.
Selecting the sampling time

- In *signal processing* one is interested in making the difference between the original and the reconstructed signal as small as possible (=high fidelity)
- In *control systems* one is interested that the closed-loop system behaves according to specs, not much in carefully reconstructing \( e(t) = y(t) - r(t) \)
- For control purposes, the sampling time is mainly related to the closed-loop bandwidth / settling-time
- The sampling time used in control is typically larger than the one used signal processing
- That’s why in control applications we are often ok with micro-controllers and don’t need DSPs (digital signal processors)
Selecting the sampling time

- Let $\omega_c$ be the desired bandwidth of the closed-loop system.
- To avoid aliasing effects (and satisfy the sampling theorem) we must set
  \[
  \frac{\pi}{T} > \omega_c
  \]
- A good choice is
  \[
  5 \omega_c \leq \frac{2\pi}{T} \leq 100 \omega_c
  \]
  so that the (non-ideal) anti-aliasing filter has cutoff frequency $\omega_f$ in between $\omega_c$ and $\omega_N$.
- By recalling the approximate relation $t_s \omega_c \approx \frac{5}{\zeta}$, where $t_s$ is the settling time (1%) and $\zeta$ is the damping factor, we get
  \[
  \frac{t_s}{100} \leq T \leq \frac{t_s}{5}
  \]
- A related good choice is also to let $T = \frac{t_r}{10}$, where $t_r$ is the rise time of the open-loop system.
Numerical errors

- Recall the relation between poles $s_i$ and $z_i$
  \[ z_i = e^{s_i T} \]
  
- For $T \to 0$, we have $z_i \to 1$ whatever the poles $s_i$ are!
  
- This can make troubles when working in finite precision (both in control design and in controller implementation)
  
- Consider the following example: we have two poles $s_1 = -1$ and $s_2 = -10$
  
  - Sample with $T = 1$ ms and get $z_1 = e^{-0.001} \approx 0.9990$, $z_2 = e^{-0.01} \approx 0.9900$. If we truncate after two digits, we get $z_1 = z_2$, and then $s_1 = \frac{1}{T} \ln z_1 = \frac{1}{T} \ln z_2 = s_2 \approx -10.05$
  
  - Sample with $T = 100$ ms and truncate, we get $z_1 = 0.90$, $z_2 = 0.36$, and then $s_1 \approx -1.05$, $s_2 \approx -10.21$
Final remarks on choice of sampling time

- Make the sampling time $T$ small enough to reproduce the open-loop time response enough precisely ($T = \frac{t_r}{10}$), and to avoid aliasing effects (Nyquist frequency $\frac{\pi}{T}$ larger than closed-loop bandwidth).
- Make the sampling time $T$ small enough to react enough readily to disturbances affecting the system.
- Make the sampling time large enough to avoid numerical issues.
- Make the sampling time large enough to avoid fast and expensive control hardware.
Digital PID control

Any of the time-discretization we have seen so far can be used to convert a PID design in digital form. The following is most used:

- **proportional part**

\[
P(t) = K_p (br(t) - y(t)) \quad \text{static relation, no need to approximate it!}
\]

- **integral part**

\[
I(t) = \frac{K_p}{T_i} \int_0^t e(\tau) \, d\tau
\]

is approximated by forward Euler as

\[
I((k + 1)T) = I(kT) + \frac{K_p T}{T_i} e(kT)
\]
Digital PID

- derivative part:

\[
\frac{T_d}{N} \frac{dD(t)}{dt} + D(t) = -K_p T_d \frac{dy(t)}{dt}
\]

is approximated by backward Euler as

\[
D(kT) = \frac{T_d}{T_d + NT} D((k - 1)T) - \frac{K_p T_d N}{T_d + NT} (y(kT) - y((k - 1)T))
\]

Note that the discrete pole \( z = \frac{T_d}{T_d + NT} \) is inside the unit circle.

- The complete control signal is

\[
u(kT) = P(kT) + I(kT) + D(kT)
\]

This type of approximation allows one to calculate P, I, and D actions separately.
Digital PID in incremental form

- The digital PID form we described provides the full signal $u(kT)$ (position algorithm).
- An alternative form is the so called incremental form (velocity algorithm), where the PID controller computes instead the input increments

$$\Delta u(kT) = u(kT) - u((k - 1)T) = \Delta P(kT) + \Delta I(kT) + \Delta D(kT)$$

where

$$\begin{align*}
\Delta P(kT) &= K_p(br(kT) - br((k-1)T) - y(kT) + y((k-1)T)) \\
\Delta I(kT) &= \alpha_1(r(kT) - y(kT)) + \alpha_2(r((k-1)T) - y((k-1)T)) \\
\Delta D(kT) &= \beta_1 \Delta D((k-1)T) - \beta_2(y(kT) - 2y((k-1)T) + y((k-2)T))
\end{align*}$$

- Advantage: increased numerical precision in the presence of finite word-length (signal quantization)
- This form cannot be used for P and PD controllers
Digital PID in incremental form

An integrator is needed to reconstruct the input signal $u(kT)$ from the incremental PID form

$$u(kT) = u((k - 1)T) + \Delta u(kT) \quad \rightarrow \quad u(kT) = \frac{1}{1 - z^{-1}} \Delta u(kT)$$
### English-Italian Vocabulary

<table>
<thead>
<tr>
<th>English</th>
<th>Italian</th>
</tr>
</thead>
<tbody>
<tr>
<td>forward Euler method</td>
<td>metodo di Eulero in avanti</td>
</tr>
<tr>
<td>backward Euler method</td>
<td>metodo di Eulero all’indietro</td>
</tr>
<tr>
<td>trapezoidal rule</td>
<td>regola dei trapezi</td>
</tr>
<tr>
<td>sampled-data system</td>
<td>sistema a dati campionati</td>
</tr>
<tr>
<td>Nyquist-Shannon sampling theorem</td>
<td>Teorema di Shannon</td>
</tr>
<tr>
<td>Dirac comb</td>
<td>pettele di Dirac</td>
</tr>
<tr>
<td>aliasing</td>
<td>aliasing</td>
</tr>
<tr>
<td>cutoff frequency</td>
<td>frequenza di taglio</td>
</tr>
</tbody>
</table>

Translation is obvious otherwise.