Automatic Control 2

Advanced linear control techniques

Prof. Alberto Bemporad

University of Trento

Academic year 2010-2011

Advanced linear control design techniques

In this lecture we will consider three useful linear design techniques:

- Deadbeat control (discrete-time)
- Delay compensation (discrete-time)
- Internal model principle (both continuous and discrete-time)

Deadbeat control – Main idea

- Consider the continuous-time system

$$\begin{cases} \dot{x}_c(t) & = & A_c x_c(t) + B_c u_c(t) \\ y_c(t) & = & C x_c(t) + D u_c(t) \end{cases}$$

- Compute its discrete-time equivalent by exact sampling

$$\begin{cases} x(k+1) & = & Ax(k) + Bu(k) \\ y(k) & = & Cx(k) + Du(k) \end{cases}$$

with $u_c(t) \equiv u_c(kT) = u(k),\ \forall t \in [kT, (k+1)T), x(k) = x_c(kT), y(k) = y_c(kT)$

- Design a linear discrete-time controller $u(k) = Kx(k)$ by placing all the closed-loop poles in $z = 0$

$$\det(zI - A - BK) = z^n$$

Deadbeat control – Main idea

- By Cayley-Hamilton theorem $(A+BK)^n = 0$
- As a consequence, the state $x(k)$ vanishes after n steps

$$x(n) = (A+BK)^n x(0) = 0$$

  and remains at the origin, $x(k) = 0$, $\forall k \geq n$
- The state x_c of the original continuous time system also converges to zero and remains at zero, $x_c(t) = 0$, $\forall t \geq nT_s$

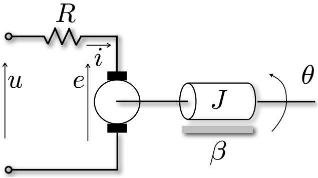
Deadbeat control brings the state of a continuous-time system to the origin in finite time nT, where T=sampling time, n=system order

Remarks on deadbeat control

- In continuous-time linear control systems closed-loop modes are exponentials $e^{\lambda_i t}$, $\Re\lambda_i < 0$
- There is no continuous-time linear controller that brings the state to zero in finite time !
- The tuning knob of the deadbeat controller is the **sampling time** T:
  - T small: the state converges quickly to the origin, but input effort may be large
  - T large: the state converges slowly, but input effort will be in general small

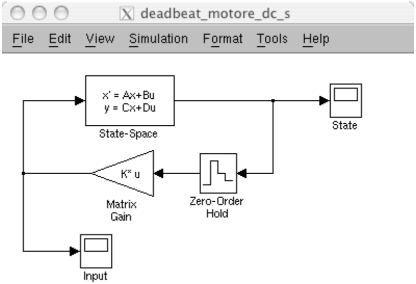
Example: deadbeat control of a DC motor

$$\begin{cases} \dot{x}_c = \begin{bmatrix} 0 & 1 \\ 0 & -(\frac{k^2}{R}+\beta)\frac{1}{J} \end{bmatrix} x + \begin{bmatrix} 0 \\ \frac{k}{JR} \end{bmatrix} u_c \\ y_c = \begin{bmatrix} 1 & 0 \end{bmatrix} x_c \end{cases}$$

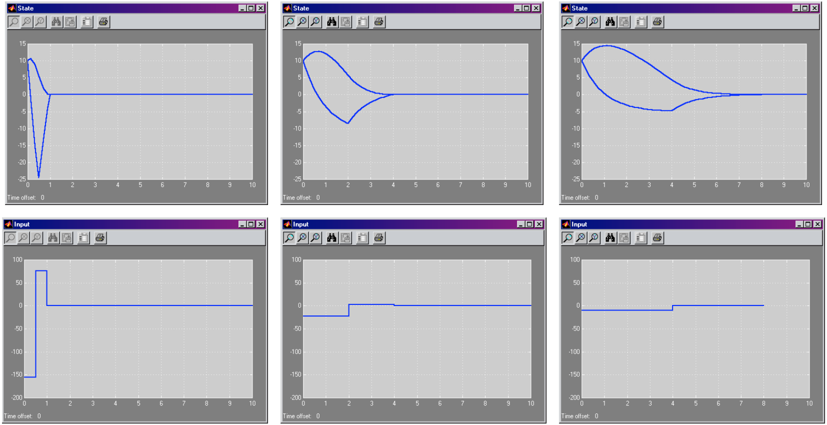


MATLAB

```
J=1; R=2;
k=1; beta=0.5;

% State:  x=[theta,omega]
Ac=[0 1; 0 -(k^2/R+beta)/J];
Bc=[0;k/J/R];
Cc=[1 0];
Dc=0;
sys=ss(Ac,Bc,Cc,Dc);

T=4; % sampling time
sysd=c2d(sys,T);
[A,B,C,D]=ssdata(sysd);

K=-acker(A,B,[0 0]);
```



Example: deadbeat control of a DC motor



$T = 0.5s$	$T = 2s$	$T = 4s$
$K = [-10.1660 \quad -5.4136]$	$K = [-1.1565 \quad -1.1075]$	$K = [-0.5093 \quad -0.5086]$

Deadbeat observer

- Let's design a discrete-time observer for $x(k)$

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + L(y(k) - C\hat{x}(k))$$

- Place now the *observer* poles in $z = 0$

$$\det(zI - A + LC) = z^n$$

- By Cayley-Hamilton theorem, $(A - LC)^n = 0 \Rightarrow$ the estimation error $\tilde{x}(k) = x(k) - \hat{x}(k)$ vanishes after n steps:

$$\tilde{x}(n) = (A - LC)^n \tilde{x}(0) = 0$$

  i.e., $\hat{x}(k) = x(k)$, $\forall k \geq n$

- Tuning considerations: the smaller the sampling time T, the faster the convergence of the estimate, but the worst typically the estimation error during the first $n-1$ steps

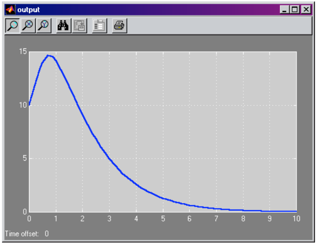
Deadbeat dynamic compensator

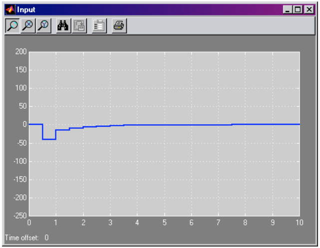
- Recall the overall dynamics of the closed-loop system under dynamic compensation

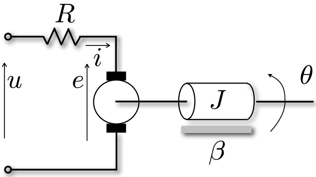
$$\begin{cases} x(k+1) &= Ax(k)+Bu(k) \\ \hat{x}(k+1) &= A\hat{x}(k)+Bu(k)+L(y(k)-C\hat{x}(k)) \\ u(k) &= K\hat{x}(k)+v(k) \\ y(k) &= Cx(k)+Du(k) \end{cases}$$

- We have seen that the observer poles do not appear in the transfer function from the reference to the output
- Let the feedback gain K be such that $(A+BK)$ has all zero eigenvalues (nilpotent)
- Will the dynamic compensator be deadbeat independently of the observer gain L ?

Example: DC motor under dynamic compensator







- Controller: $u(kT) = K\hat{x}(kT)$
- Controller poles: $0, 0$
- Observer poles: $0.5, 0.7$

The output does not converge to zero after $n = 2$ samples !

Dynamic (deadbeat) compensator

- The overall closed-loop dynamics is

$$\left\{ \begin{bmatrix} x(k+1) \\ \tilde{x}(k+1) \end{bmatrix} = \begin{bmatrix} A+BK & -BK \\ 0 & A-LC \end{bmatrix} \begin{bmatrix} x(k) \\ \tilde{x}(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} v(k) \right.$$

- The corresponding state evolution is

$$\begin{aligned} x(k) &= \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} A+BK & -BK \\ 0 & A-LC \end{bmatrix}^k \begin{bmatrix} x(0) \\ \tilde{x}(0) \end{bmatrix} \\ &= \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} (A+BK)^k & H(k) \\ 0 & (A-LC)^k \end{bmatrix} \begin{bmatrix} x(0) \\ \tilde{x}(0) \end{bmatrix} \\ &= \underbrace{(A+BK)^k x(0)}_{\text{goes to zero in } n \text{ steps}} \quad + H(k)\tilde{x}(0) \end{aligned}$$

where $H(k)$ is a matrix (dependent on k) that may not be zero for $k \geq n$.

Dynamic (deadbeat) compensator

- Let's place the eigenvalues of both $A+BK$ and $A-LC$ at zero
- Since

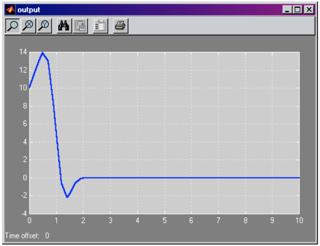
$$y(k) = \begin{bmatrix} C & 0 \end{bmatrix} \underbrace{\begin{bmatrix} (A+BK)^k & H(k) \\ 0 & (A-LC)^k \end{bmatrix}}_{2n \times 2n \text{ matrix with zero eigenvalues}} \begin{bmatrix} x(0) \\ \tilde{x}(0) \end{bmatrix}$$

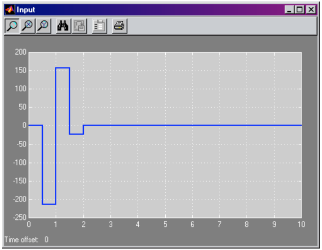
the output and state vectors converge to zero after at most $2n$ steps

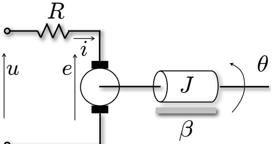
Proof:

Matrix $\left[\begin{smallmatrix} A+BK & -BK \\ 0 & A-LC \end{smallmatrix}\right]$ is nilpotent of order $2n$. By Cayley-Hamilton theorem $\left[\begin{smallmatrix} A+BK & -BK \\ 0 & A-LC \end{smallmatrix}\right]^{2n} = 0$, and hence $H(k) = 0$ for all $k \geq 2n$. $\square$

Example: DC motor under dynamic compensator



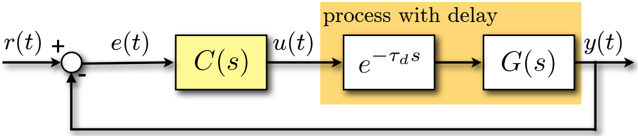




- Controller poles: $0, 0$
- Observer poles: $0, 0$

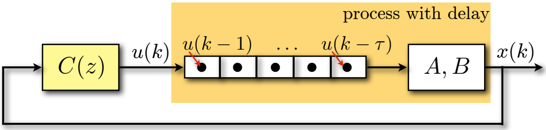
The output converges to zero after $2n = 4$ samples !

The problem of delay



- Control loops sometimes suffer the presence of delays, for example due to transport phenomena and buffers
- According to frequency-domain analysis, time delays introduce phase lag, and classical linear control techniques (like PID) may be unable to correct the phase margin adequately
- We will see two simple *discrete-time* control methods that compensate delays very effectively

Delay compensation: Method #1



- Assume the discrete-time model has delay of τ steps on the input and that (A,B) is completely reachable

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k-\tau) \\ y(k) &= Cx(k) \end{aligned}$$

- Map delays in τ poles in $z = 0$, introducing τ new states $w_j(k) \triangleq u(k-j)$, $j = 1, \ldots, \tau$
- The augmented system is

$$\begin{aligned} x(k+1) &= Ax(k) + Bw_\tau(k) \\ w_\tau(k+1) &= w_{\tau-1}(k) \quad = u(k-\tau+1) \\ &\vdots \\ w_2(k+1) &= w_1(k) \quad = u(k-1) \\ w_1(k+1) &= u(k) \end{aligned}$$

MATLAB

```
»sysnd = delay2z(sys)
```

Delay compensation: Method #1

- The extended system is

$$\begin{bmatrix} x \\ w_\tau \\ \vdots \\ w_2 \\ w_1 \end{bmatrix}(k+1) = \begin{bmatrix} A & B & 0 & \dots & 0 \\ 0 & 0 & I_m & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & I_m \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} x \\ w_\tau \\ \vdots \\ w_2 \\ w_1 \end{bmatrix}(k) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ I_m \end{bmatrix} u(k)$$

and is completely reachable:

$$R = \begin{bmatrix} 0 & 0 & \dots & 0 & B & AB & \dots & A^{n-1}B \\ 0 & 0 & \dots & I_m & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & I_m & \dots & 0 & 0 & 0 & \dots & 0 \\ I_m & 0 & \dots & 0 & 0 & 0 & \dots & 0 \end{bmatrix}$$

- Design a controller K for the extended system (by LQR, pole-placement, etc.)

$$\begin{aligned} u(k) &= K[x'(k)\, w'_\tau(k)\, \dots\, w'_1(k)]' + v(k) \\ &= K_x x(k) + K_\tau u(k-\tau) + \dots + K_1 u(k-1) + v(k) \end{aligned}$$

Delay compensation: Method #2

- Consider again the discrete-time model with delay of τ steps on the input

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k-\tau) \\ y(k) &= Cx(k) \end{aligned}$$

- Consider the delay-free model with state $\bar{x}(k) \triangleq x(k+\tau)$

$$\begin{aligned} \bar{x}(k+1) &= A\bar{x}(k) + Bu(k) \\ \bar{y}(k) &= C\bar{x}(k) \end{aligned}$$

- Design a controller $u(k) = K\bar{x}(k) + v(k)$ for the delay-free system
- Implementation: at time k predict the state τ steps ahead

$$x(k+\tau) = A^{\tau}x(k) + \sum_{j=0}^{\tau-1} A^j Bu(k-1-j)$$

Note that any other predictor of $x(k+\tau)$ would be ok, for instance a predictor based on a more accurate nonlinear model

- The complete control law is $u(k) = Kx(k+\tau) + v(k)$

Delay compensation: Method #2

- The delay-free closed-loop system is

$$\begin{aligned} \bar{x}(k+1) &= (A+BK)\bar{x}(k)+Bv(k) \\ \bar{y}(k) &= C\bar{x}(k) \end{aligned}$$

- The transfer function from $v(k)$ to $\bar{y}(k) = y(k+\tau)$ is

$$\frac{\bar{Y}(z)}{V(z)} = \frac{z^{\tau}Y(z)}{V(z)} = C(zI-A-BK)^{-1}B$$

and therefore

$$\frac{Y(z)}{V(z)} = C(zI-A-BK)^{-1}Bz^{-\tau}$$

- The characteristic polynomial $p_d(\lambda)$ of the resulting closed-loop system is

$$p_d(\lambda) = \det(\lambda I - A - BK)\lambda^{\tau} \quad \longleftarrow \tau \text{ closed-loop poles in } z=0$$

Example of delay compensation

- Open-loop system

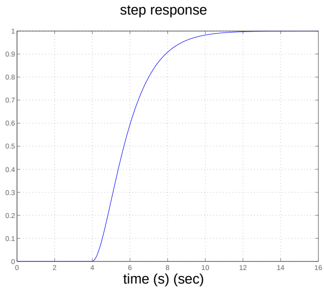
$$y(t) = \frac{1}{(s+1)^2} e^{-4s} u(t)$$

- Set

$$x(t) \triangleq \begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix}$$

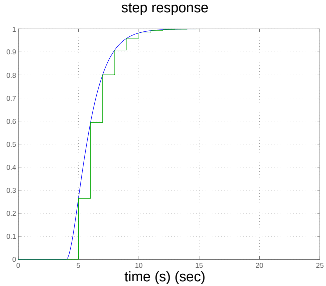
and obtain the state-space model

$$\begin{aligned} \dot{x}(t) &= \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t-4) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \end{aligned}$$



Example of delay compensation (cont'd)

- choose the sampling time $T = 1$ s
- convert the system to discrete-time
- compare step response



$$\begin{aligned} x(k+1) &= \begin{bmatrix} 0.7358 & 0.3679 \\ -0.3679 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 0.2642 \\ 0.3679 \end{bmatrix} u(k-4) \\ y(k) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(k) \end{aligned}$$

- By placing closed-loop poles in $0.4 \pm j0.4$ we get

$$K = [-0.3014 \;\; 0.3911]$$

Example of delay compensation (cont'd)

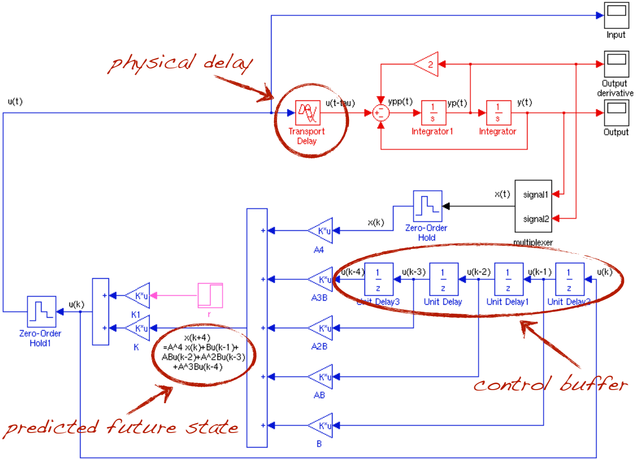
- The control law is $u(k) = K\bar{x}(k) + v(k)$, $v(k) = Hr(k)$
- Let's calculate H to have unit DC-gain

$$H = \frac{1}{C(I - A - BK)^{-1}B} = 1.3014$$

- The control law with delay compensation is

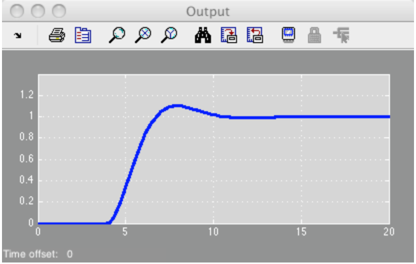
$$\left\{ \begin{array}{rcl} \hat{x}(k+4) & = & A^4 x(k) + A^3 Bu(k-4) + A^2 Bu(k-3) + \\ & & +ABu(k-2) + Bu(k-1) \\ & & \\ u(k) & = & K\hat{x}(k+4) + Hr(k) \end{array} \right.$$

Example of delay compensation (cont'd)

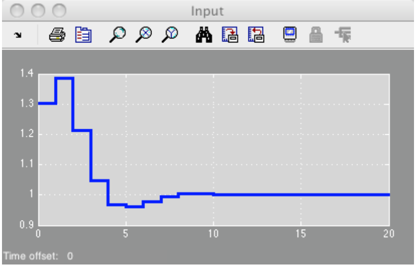


Example of delay compensation (cont'd)

output signal $y(t)$



input signal $u(t)$



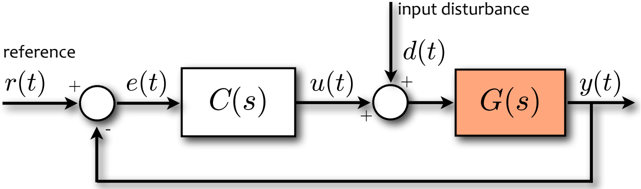
Remarks on delay compensation

- By using method #1, the extended state has dimension $n+\tau$. The computation of K can be complex if τ is large
- Instead, with method #2 the gain K has always n components, independently of τ
- On the other hand, method #2 is a particular case of method #1, as

$$K\hat{x}(k+\tau) = K[A^{\tau}\ A^{\tau-1}B\ \ldots AB\ B]\begin{bmatrix} x(k) \\ u(k-\tau) \\ \\ \vdots \\ u(k-1) \end{bmatrix}$$

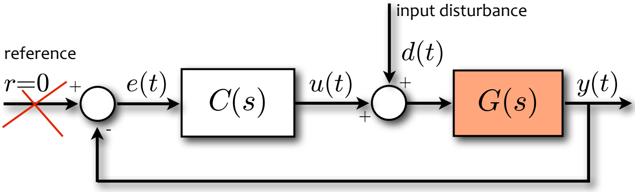
- Method #2 can only choose n closed-loop poles, by construction the remaining τ are in $z=0$: $p_d(\lambda) = \det(zI - A - BK)z^{\tau}$
- If the reference $r(k)$ is known in advance, it is possible to compensate the closed-loop tracking delay τ by sending $r(k+\tau)$ to the controller (*anticipative action* or *preview*)
- Both methods can be extended to observer design for compensating a delay $\sigma > 0$ on the output channel $y(k-\sigma)$

Internal model principle



- Let's come back to the problem of tracking a reference signal $r(t)$ under the possible perturbation of an input disturbance $d(t)$
- When $r(t)$, $d(t)$ are constant signals, we've seen that by introducing an *integral action* we can guarantee zero tracking errors in steady-state
- Can we generalize the idea of embedding the "internal model" $\frac{1}{s}$ of the (Laplace transform) of the reference and/or disturbance signal to other waveforms ?

IMP for noise models rejection



- Consider first the case of disturbance rejection only ($r(t) \equiv 0$)
- Let $d(t)$ be a signal whose Laplace transform $D(s)$ is a rational function

$$D(s) = \mathscr{L}[d(t)] = \frac{N_d(s)}{D_d(s)}$$

- Examples:

$$\begin{array}{rclcrcl} d(t) & = & 1\!\!1(t) \text{ (unit step)} & \Rightarrow & D(s) & = & \frac{1}{s} \\ d(t) & = & \sin(\omega t)\, 1\!\!1(t) & \Rightarrow & D(s) & = & \frac{\omega}{s^2+\omega^2} \end{array}$$

- Let $G(s) = \dfrac{N_p(s)}{D_p(s)}$ be the transfer function of the open-loop process

IMP for noise models rejection

- Let the transfer function of the controller $C(s)$ *include the disturbance model*

$$C(s) = \frac{N_c(s)}{D_c(s)D_d(s)}$$

- How to design $N_c(s)$, $D_c(s)$? Consider the extended system

$$G_e(s) = \frac{N_p(s)}{D_p(s)D_d(s)}$$

and design a stabilizing dynamic compensator $\frac{N_c(s)}{D_c(s)}$ (by state-feedback control + observer design on a realization A, B, C, D of $G_e(s)$, or by loop-shaping, etc.)

Theorem: Internal Model Principle (disturbance rejection)

A sufficient condition for the steady-state rejection of an input disturbance signal $d(t)$ with Laplace transform $N_d(s)/D_d(s)$ is that the denominator polynomial of $C(s)$ contains $D_d(s)$ (more generally: the denominator polynomial of the the loop function $L(s) = C(s)G(s)$ contains $D_d(s)$)

IMP for noise models rejection

Proof:

- For $r \equiv 0$, the Laplace transform $Y(s)$ of the output $y(t)$ is

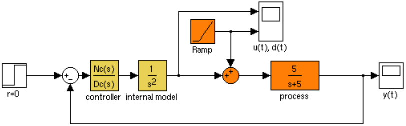
$$Y = \frac{N_d N_p D_c}{D_p D_d D_c + N_p N_c}$$

- The closed-loop poles are the roots of the polynomial

$$P_d = (D_p D_d) D_c + N_p N_c$$

  and therefore, by design, they have negative real part
- Then $y(t) = \mathscr{L}^{-1}[Y(s)]$ converges to zero asymptotically as $t \to \infty$

Example

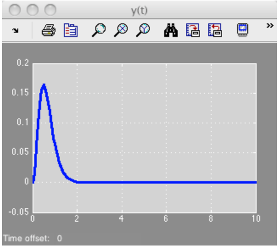


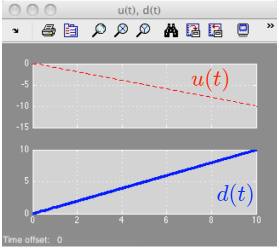
MATLAB

```
G=tf(5,[1 5]);
Gd=tf(1,[1 0 0]);

Ge=G*Gd;
ssGe=ss(Ge);

[A,B,C,D]=ssdata(ssGe);
K=-place(A,B,[-8 -5+j -5-j]);
L=place(A',C',[-10 -12 -15])';

Ce=-reg(ssGe,-K,L);

[Nc,Dc]=tfdata(Ce);
Nc=Nc{1};
Dc=Dc{1};
```

process model: $G(s) = \dfrac{5}{s+5}$

input disturbance model: $D(s) = \dfrac{1}{s^2}$ (ramp)





IMP for reference tracking

- Let $r(t)$ be a signal whose Laplace transform $R(s) = \mathscr{L}[r(t)] = \dfrac{N_r(s)}{D_r(s)}$
- consider the extended system $G_e(s) = \dfrac{N_p(s)}{D_p(s)D_r(s)}$
- design a stabilizing dynamic compensator $\frac{N_c(s)}{D_c(s)}$, and let $C(s) = \frac{N_c(s)}{D_c(s)D_r(s)}$

Theorem: Internal Model Principle (reference tracking)

A sufficient condition for tracking a reference signal $r(t)$ with Laplace transform $N_r(s)/D_r(s)$ with zero offset in steady-state is that the denominator polynomial of $C(s)$ contains $D_r(s)$ (more generally: the denominator polynomial of the the loop function $L(s) = C(s)G(s)$ contains $D_r(s)$)

Proof: Compute the Laplace transform $E(s) = R(s) - Y(s) = \frac{N_r N_p D_c}{D_p D_c D_r + N_p N_c}$. Its inverse Laplace transform $\mathscr{L}^{-1}[E(s)] = r(t) - y(t)$ tends to zero asymptotically $\square$

Example

- Problem data:

$$G(s) = \frac{s+1}{s(s+10)(s+20)}, \quad r(t) = 1 + \sin\frac{t}{2}, \quad d(t) = 20\cos\frac{t}{2}$$

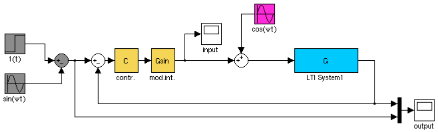
- Compute Laplace transforms of reference and disturbance signals

$$R(s) = \frac{1}{s} + \frac{\frac{1}{2}}{s^2 + \frac{1}{4}}, \quad D(s) = \frac{20s}{s^2 + \frac{1}{4}}$$

- We need to include the polynomial $s(s^2 + \frac{1}{4})$. Since $\frac{1}{s}$ already appears in $G(s)$, it's enough to augment by $\frac{1}{s^2+\frac{1}{4}}$ the system
- Design a regulator $C(s)$ to stabilize the extended system

$$G_e(s) = \frac{s+1}{s(s+10)(s+20)(s^2 + \frac{1}{4})}$$

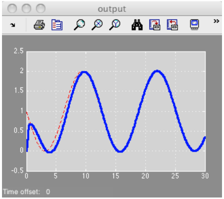
Example

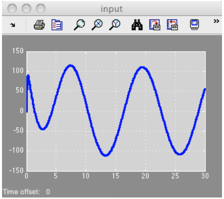


Dynamic compensator: LQR controller + state observer designed by pole placement

MATLAB

```
omega=0.5;

G=tf([1 1],[1 30 200 0]);
Gsin=tf(1,[1 0 omega^2]);
Ge=G*Gsin;

[A,B,C,D]=ssdata(ss(Ge));

%LQR state-feedback
K=-lqr(A,B,C'*C,.001);

%Observer
L=place(A',C',[-10 -15 -20 -25 -30])';

% Dynamic compensator (for augmented system)
C=-reg(ss(Ge),-K,L)

% The overall compensator is C*Gsin
```





Perfect tracking of $r(t) = 1 + \sin(0.5t)$ with complete rejection of $d(t) = 20\cos(0.5t)$

English-Italian Vocabulary

	
deadbeat controller	*controllore deadbeat*
internal model principle	*principio del modello interno*
time-delay system	*sistema con ritardo*

Translation is obvious otherwise.