Topic #17

16.30/31 Feedback Control Systems

- Improving the transient performance of the LQ Servo
- Feedforward Control Architecture Design
- DOFB Servo
- Handling saturations in DOFB

LQ Servo Revisited

- Earlier (13-??) we described how to use an integrator to achieve zero steady state error that is a bit more robust to modeling errors than the $\overline{N}$ formulation.
  - An issue with this architecture is that the response to the reference is only driven by the integrated error $\Rightarrow$ there is no direct path from the reference input to the system
    $\Rightarrow$ transient might be slow.

- If the relevant system output is $\mathbf{y} = C_y\mathbf{x}$ and reference $\mathbf{r}$, previously added extra states $\mathbf{x}_I$, where

$$\dot{\mathbf{x}}_I = \mathbf{e}$$

  - Then penalize both $\mathbf{x}$ and $\mathbf{x}_I$ in the cost
  - Caveat: note we are free to define $\mathbf{e} = \mathbf{r} - \mathbf{y}$ or $\mathbf{e} = \mathbf{y} - \mathbf{r}$, but the implementation must be consistent

- As before, if the state of the original system is $\mathbf{x}$, then the dynamics are modified to be

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{x}}_I \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C_y & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_I \end{bmatrix} + \begin{bmatrix} B_u \\ 0 \end{bmatrix} \mathbf{u} + \begin{bmatrix} 0 \\ I \end{bmatrix} \mathbf{r}$$

  and define $\overline{\mathbf{x}} = \begin{bmatrix} \mathbf{x}^T & \mathbf{x}_I{}^T \end{bmatrix}^T$

- The optimal feedback for the cost

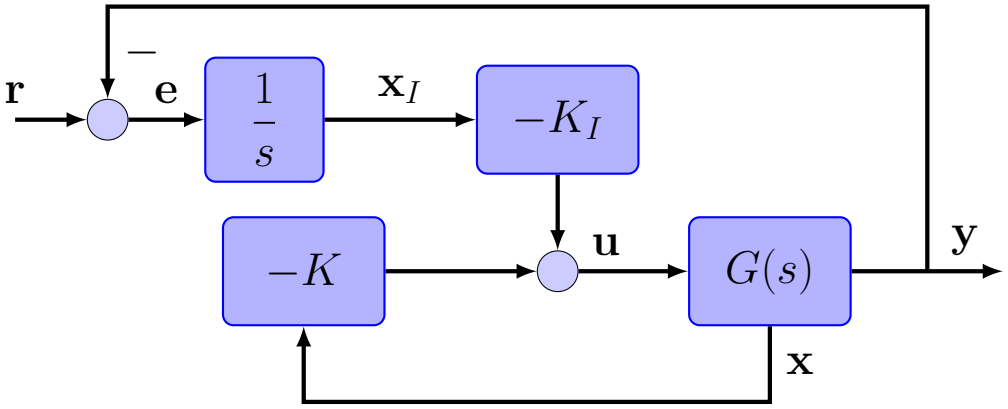
$$J = \int_0^\infty \left[\overline{\mathbf{x}}^T R_{\mathrm{xx}} \overline{\mathbf{x}} + \mathbf{u}^T R_{\mathrm{uu}} \mathbf{u}\right] dt$$

  is of the form:

$$\mathbf{u} = -\begin{bmatrix} K & K_I \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_I \end{bmatrix} = -\overline{K}\overline{\mathbf{x}}$$

- Once have used LQR to design control gains K and K_I, we have the freedom to choose how we implement it the control system.
  - Provided that we don't change the feedback path and thus modify the closed loop pole locations

- The first controller architecture on (13-??) was of the form:



- And the suggestion is that we modify it to this revised form:

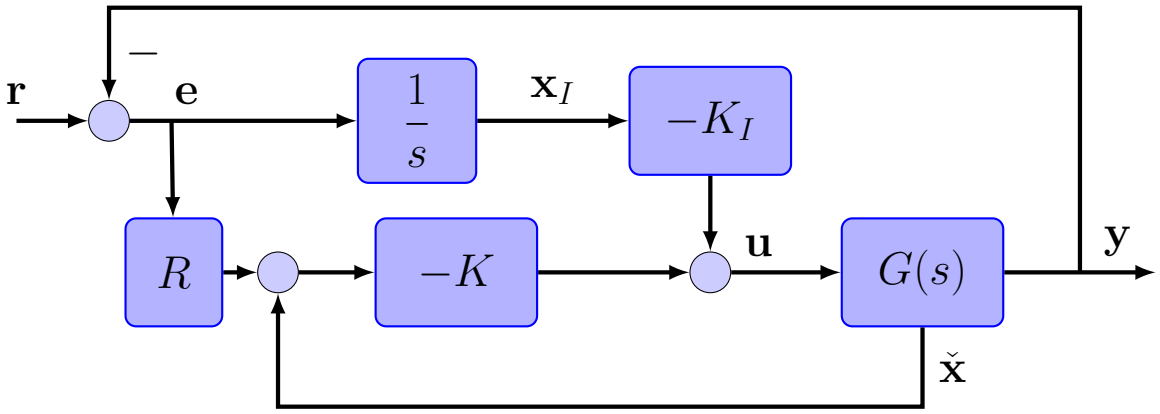


Fig. 1: Revised implementation of the LQ servo

- Note key difference from above architecture is the feedforward term of the $\mathbf{e}$ through R.
  - This actually adds another feedback loop, so we need to clarify how this is done.

Design Approach

- Control law for the revised implementation can be written as:

$$\mathbf{u} = -K\left(\check{\mathbf{x}} + R\mathbf{e}\right) - K_I\mathbf{x}_I$$

  - Note that the tracking error is in the output of the integrator and the state feedback components.
  - So this approach has the potential to correct the "slow response" problems identified in the original implementation.

- But how do we determine R?

- Assume that the state can be partitioned into parts we care about for tracking $(T\mathbf{x})$ that we assume are directly available from $\mathbf{y}$ and parts we don't $(\check{\mathbf{x}} = \overline{T}\mathbf{x})$
  - Can think of T and $\overline{T}$ as "selector" matrices with a diagonal of 1's and 0's - but not always this form
  - But must have $T + \overline{T} = I$.

- **Example**: Consider position control case, where location x is part of state vector $\mathbf{x} = [x, v, \ldots]^T$, and $y = x = C\mathbf{x}$
  - Scalar position reference r, and $e = r - y$, $\dot{x}_I = e$
  - In this example it is clear that we have

$$\begin{bmatrix} x \\ 0 \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 & & \\ & 0 & \\ & & \ddots \end{bmatrix} \begin{bmatrix} x \\ v \\ \vdots \end{bmatrix}$$

$$\Rightarrow T = \begin{bmatrix} 1 & & \\ & 0 & \\ & & \ddots \end{bmatrix} \text{ and } \overline{T} = \begin{bmatrix} 0 & & \\ & 1 & \\ & & \ddots \end{bmatrix}$$

- So the control is

$$\begin{aligned}\mathbf{u} &= -K(\check{\mathbf{x}} + Re) - K_I x_I \\ &= -K(\check{\mathbf{x}} + R[r - y]) - K_I x_I \\ &= -K(\underbrace{\check{\mathbf{x}} - RC\mathbf{x}}_{\text{key}} + Rr) - K_I x_I\end{aligned}$$

- To proceed, note what happens if we define $RC = -T$

$$\Rightarrow \check{\mathbf{x}} - RC\mathbf{x} = \overline{T}\mathbf{x} - (-T)\mathbf{x} = \mathbf{x}$$

  - Then the control signal becomes:

$$\mathbf{u} = -K(\mathbf{x} + Rr) - K_I x_I$$

  - This choice of $T = -RC$ ensures that we avoid double counting in the feedback $\Rightarrow$ the elements of $\mathbf{x}$ are not fedback more that once
  - Appears to be useful to choose $R = -\alpha C^T$, with $\alpha < 1$

- Closed-loop dynamics
  - Two control laws being compared are:

$$\mathbf{u}_1 = -K(\mathbf{x}) - K_I x_I \quad (1)$$
$$\mathbf{u}_2 = -K(\mathbf{x} + Rr) - K_I x_I \quad (2)$$

    with dynamics

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}, \quad y = C\mathbf{x} \quad (3)$$

    augmented with an integrator $\dot{x}_I = r - y$

- Closed-loop:

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{x}_I \end{bmatrix} = \begin{bmatrix} A - BK & -BK_I \\ -C & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ x_I \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} r \tag{4}$$

and

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{x}_I \end{bmatrix} = \begin{bmatrix} A - BK & -BK_I \\ -C & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ x_I \end{bmatrix} + \begin{bmatrix} -BKR \\ I \end{bmatrix} r \tag{5}$$

- Example $G(s) = \frac{(s+2)}{(s-1)(s-s+2)}$

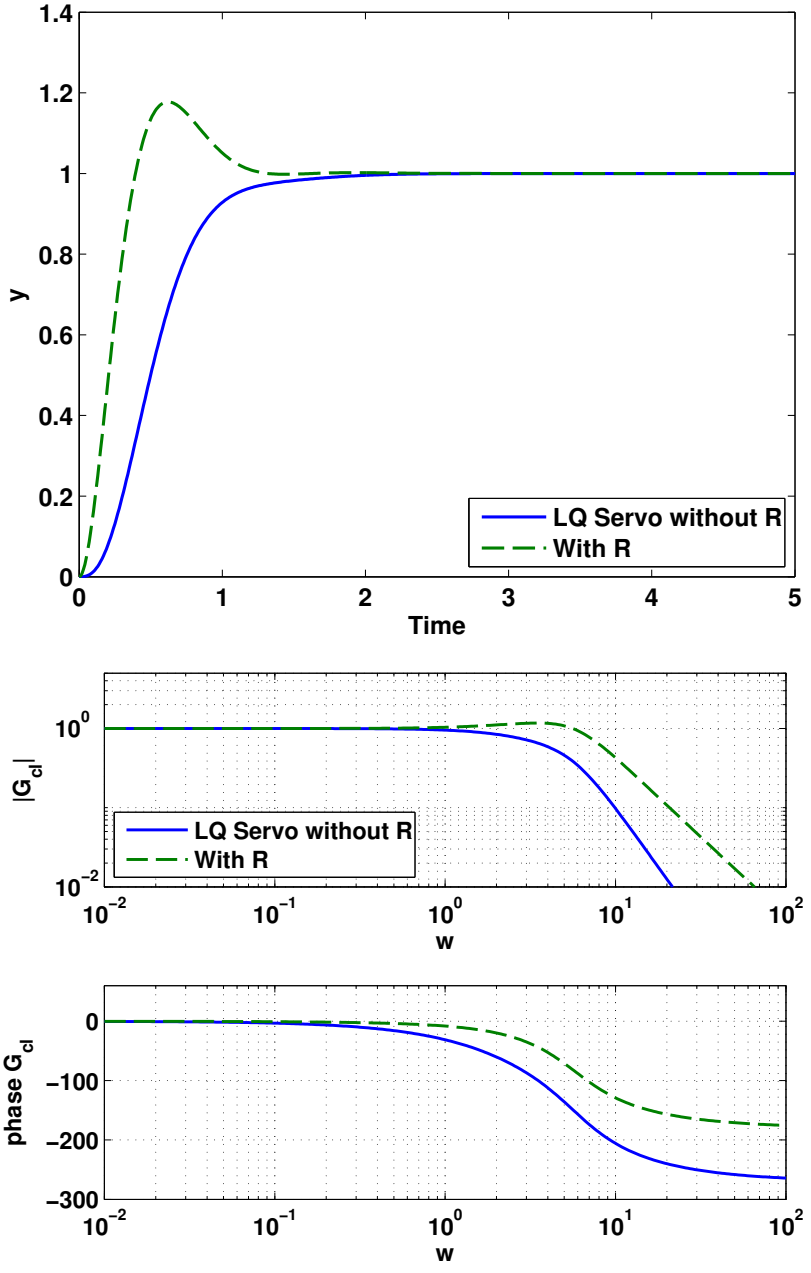


Fig. 2: LQ servo implementation comparison

Code: LQservo Comparison

```
% LQR servo
% Fall 2010
% Jonathan How, MIT
close all;clear all;j=sqrt(-1);

G=tf([1 2],conv([1 -1],[1 -1 2]));
[a,b,c,d]=ssdata(G);
% consider possibility that c is wrong in the design
% use c in design of K's and N's, but simulate with cpert
ns=size(a,1);
z=zeros(ns,1);
Rxx=c'*c;
Ruu=.001;
Rxx1=[Rxx z;z' 10];
R=-c'/1.75;

abar=[a z;-c 0];
bbar=[b;0];
% for Nbar calc
kbar1=lqr(abar,bbar,Rxx1,Ruu);
K1=kbar1(1,1:ns);Ki1=kbar1(ns+1);
% for R calc
kbar2=lqr(abar,bbar,Rxx1,1*Ruu);
K2=kbar2(1,1:ns);Ki2=kbar2(ns+1);

acl1=abar-bbar*kbar1;
acl2=abar-bbar*kbar2;
% implementation as on 17-5
bcl1=[zeros(ns,1);1];
bcl2=[-b*K2*R;1];
ccl=[c 0];
dcl=0;

Gcl1=ss(acl1,bcl1,ccl,dcl);
Gcl2=ss(acl2,bcl2,ccl,dcl);

figure(1);
t=[0:.01:5]';
[y1,t1]=step(Gcl1,t);
[y2,t2]=step(Gcl2,t);
plot(t,y1,t,y2,'--')
xlabel('Time')
ylabel('y')
legend('LQ Servo without R','With R','Location','SouthEast')
export_fig lqrservo2_1 -pdf

w=logspace(-2,2,300);
figure(2)
[mag1,ph1]=bode(Gcl1,w);mag1=squeeze(mag1);ph1=squeeze(ph1);
[mag2,ph2]=bode(Gcl2,w);mag2=squeeze(mag2);ph2=squeeze(ph2);
subplot(211);
loglog(w,mag1,w,mag2,'--');grid on
legend('LQ Servo without R','With R','Location','SouthWest')
ylabel('|G_{cl}|');xlabel('w');axis([1e-2 1e2 1e-2 5])
subplot(212);
semilogx(w,ph1,w,ph2,'--');grid on
axis([1e-2 1e2 -300 60]);ylabel('phase G_{cl}');xlabel('w')
export_fig lqrservo2_2 -pdf
```

DOFB Servo

- Now back up a bit add address how to add an integrator into the DOFB compensator to obtain robust tracking performance. For the DOFB problem,

$$\begin{aligned}\dot{\hat{\mathbf{x}}} &= A\hat{\mathbf{x}} + B\mathbf{u} + L(\mathbf{y} - \hat{\mathbf{y}}) \\ &= (A - LC)\hat{\mathbf{x}} + B\mathbf{u} + L\mathbf{y} \qquad (6)\end{aligned}$$

$$\dot{\mathbf{x}}_I = \mathbf{r} - \mathbf{y}$$

$$\mathbf{u} = -\begin{bmatrix} K & K_I \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}} \\ \mathbf{x}_I \end{bmatrix}$$

  - Augment integrator state used in LQR design, since it must be part of the compensator (does not need to be estimated)

$$\breve{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{x}}^T & \mathbf{x}_I^T \end{bmatrix}^T$$

  - Which gives the **DOFB servo compensator** as:

$$\begin{aligned}\dot{\breve{\mathbf{x}}} &= \left[\begin{array}{c|c} A - LC - BK & -BK_I \\ \hline 0 & 0 \end{array}\right] \breve{\mathbf{x}} + \begin{bmatrix} L \\ -I \end{bmatrix} \mathbf{y} + \begin{bmatrix} 0 \\ I \end{bmatrix} \mathbf{r} \\ \mathbf{u} &= -\overline{K}\breve{\mathbf{x}} \qquad (7)\end{aligned}$$

- Resulting closed-loop system (apply (7) to (3))

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\breve{\mathbf{x}}} \end{bmatrix} = \left[\begin{array}{cc} A & -B\overline{K} \\ \begin{bmatrix} LC \\ -C \end{bmatrix} & \left[\begin{array}{c|c} A - LC - BK & -BK_I \\ \hline 0 & 0 \end{array}\right] \end{array}\right] \begin{bmatrix} \mathbf{x} \\ \breve{\mathbf{x}} \end{bmatrix} + \left[\begin{array}{c} 0 \\ \begin{bmatrix} 0 \\ I \end{bmatrix} \end{array}\right] \mathbf{r}$$

$$\mathbf{y} = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \breve{\mathbf{x}} \end{bmatrix}$$

DOFB Servo Example

- System

$$G(s) = \frac{(s+2)}{(s-1)(s-s+2)}$$

  - Use LQR and its dual to find K and L using the dynamics augmented with an integrator.
  - Implement as in Eq. 7 to get the following results
  - Note root locus as a function of scalar loop gain multiplier α, where nominally $\alpha = 1$

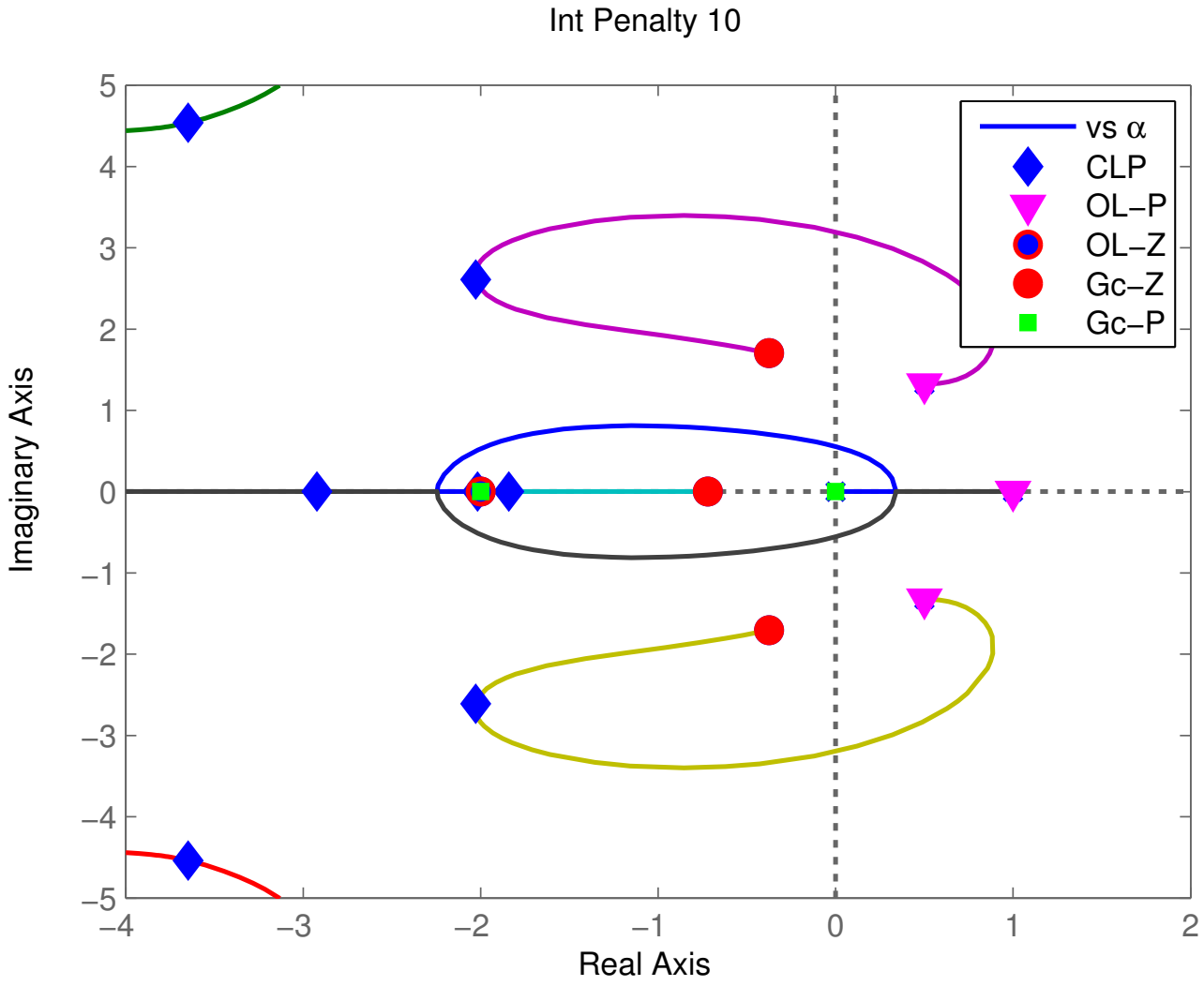


Fig. 3: DOFB servo root locus

- Obviously a pretty complex structure for the controller – often see that the compensator zeros are close to the plant poles.

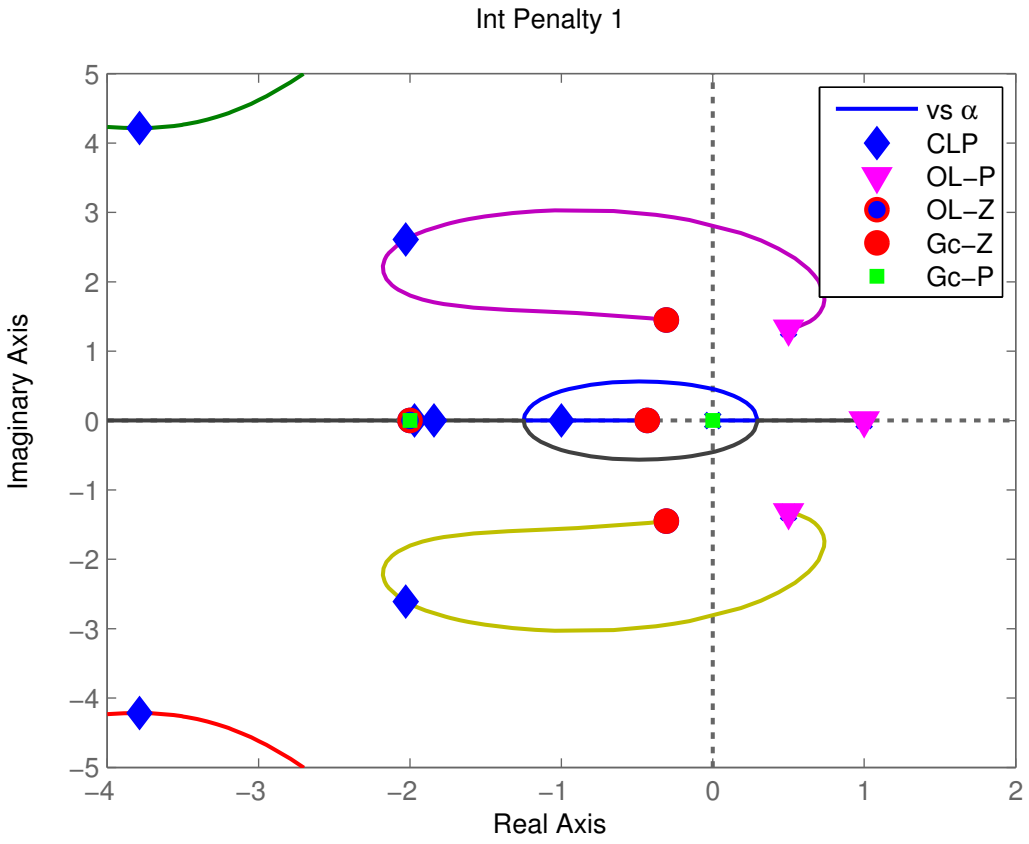


Fig. 4: DOFB servo root locus

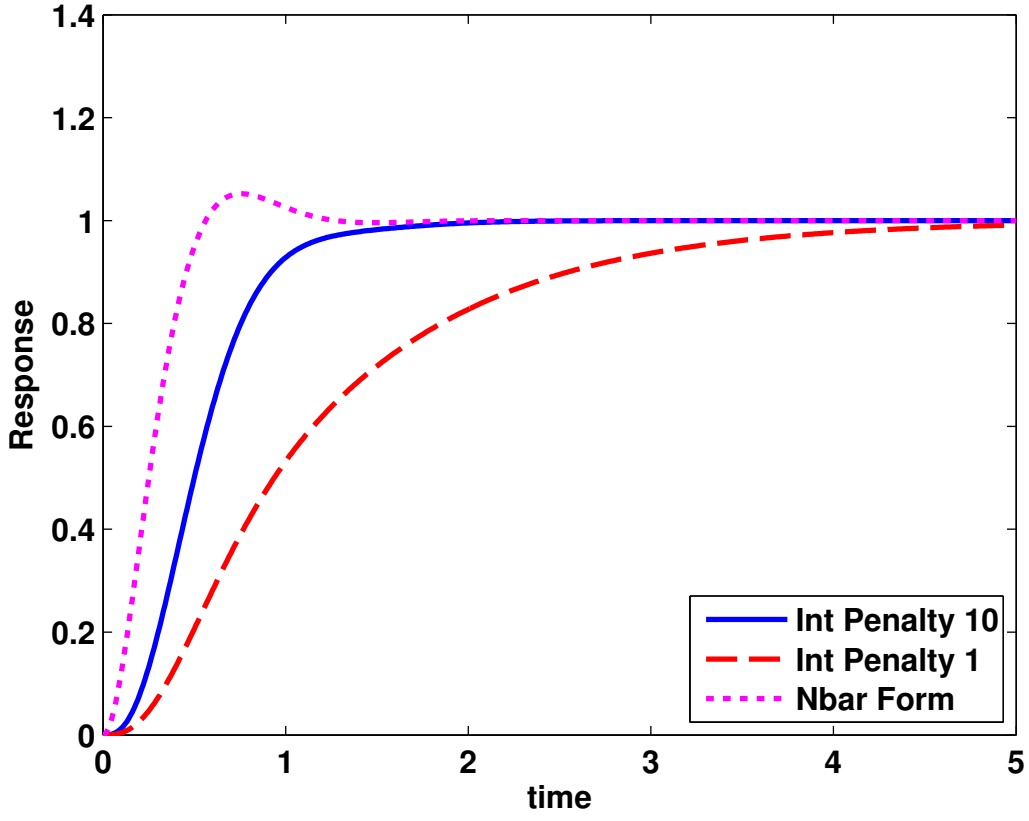


Fig. 5: Step response of the DOFB servo – interplay now between the various weightings - all give zero SS error, but transient response quite different.

DOFB Servo II

- Now, as before, consider changing the implementation to improve the transient response.
  - Consider same modification as on 17-2 and add a more direct feedback of parts of $\mathbf{e}$ other than through integrator state.

- As before, assume that we can partition the state into parts we care about for tracking $T\mathbf{x}$ and parts we don't $\overline{T}\mathbf{x}$
  - Require that $T + \overline{T} = I$

- For the DOFB problem, basic form of the estimator is

$$\begin{aligned} \dot{\hat{\mathbf{x}}} &= (A - LC)\hat{\mathbf{x}} + B\mathbf{u} + L\mathbf{y} \\ \dot{\mathbf{x}}_I &= \mathbf{e} = \mathbf{r} - \mathbf{y} \end{aligned}$$

  - Note: can also define an **error based on the state estimate**

$$\hat{\mathbf{e}} = \mathbf{r} - \hat{\mathbf{y}} = \mathbf{r} - C\hat{\mathbf{x}}$$

- Let us define the control signal to be (note similarities to $\mathbf{u}$ equation on 17-4):

$$\begin{aligned} \mathbf{u} &= -K(\overline{T}\hat{\mathbf{x}} + R\hat{\mathbf{e}}) - K_I\mathbf{x}_I \\ &= -K(\overline{T}\hat{\mathbf{x}} + R(\mathbf{r} - C\hat{\mathbf{x}})) - K_I\mathbf{x}_I \\ &= -K((\overline{T} - RC)\hat{\mathbf{x}} + R\mathbf{r}) - K_I\mathbf{x}_I \end{aligned}$$

  Again set $RC = -T$, so $\overline{T} - RC = I$ and

$$\begin{aligned} \mathbf{u} &= -K(\hat{\mathbf{x}} + R\mathbf{r}) - K_I\mathbf{x}_I \\ &= -\overline{K}\breve{\mathbf{x}} - KR\mathbf{r} \end{aligned} \tag{8}$$

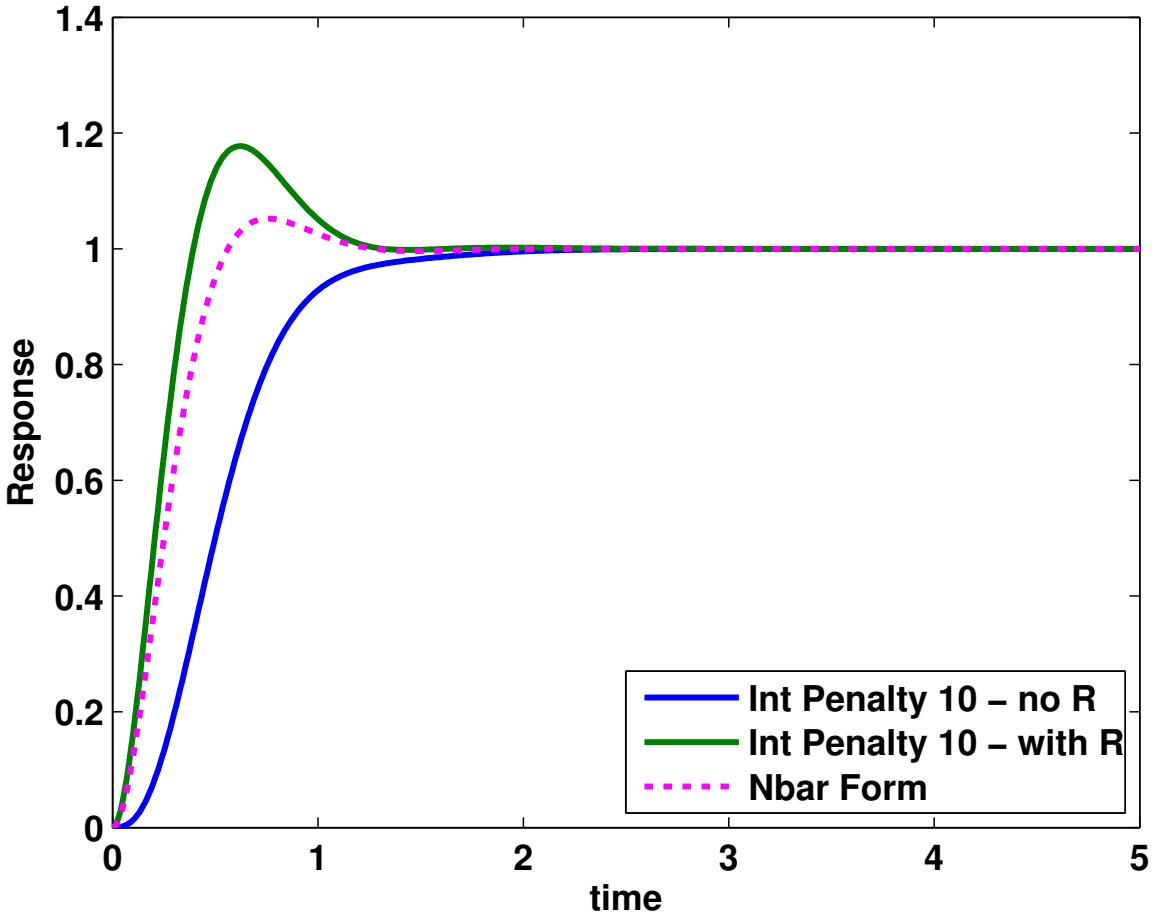


Fig. 6: DOFB servo step response – with and without R formulations compared to $\overline{N}$ formulation (see Figure 5).

- Closed-loop system (apply (8) and first line of (7) to (3))

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \breve{\mathbf{x}} \end{bmatrix} = \left[\begin{array}{cc} A & -B\overline{K} \\ \begin{bmatrix} LC \\ -C \end{bmatrix} & \left[\begin{array}{c|c} A-LC-BK & -BK_I \\ \hline 0 & 0 \end{array}\right] \end{array}\right] \begin{bmatrix} \mathbf{x} \\ \breve{\mathbf{x}} \end{bmatrix} + \left[\begin{array}{c} -BKR \\ \begin{bmatrix} -BKR \\ I \end{bmatrix} \end{array}\right] \mathbf{r}$$

$$\mathbf{y} = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \breve{\mathbf{x}} \end{bmatrix}$$

- Example for same system as 17-5: $G(s) = \frac{(s+2)}{(s-1)(s-s+2)}$
- Adding extra reference input like this increases the bandwidth of the controller and speeds up the response.
  - But perhaps too much - so further tuning required.

DOFB Implementation with Saturations

- Closed-loop implementation of DOFB assumes that the system and compensator both see $\mathbf{u} = -K\hat{\mathbf{x}}$.
  - If actuator saturates, then possible that the system sees

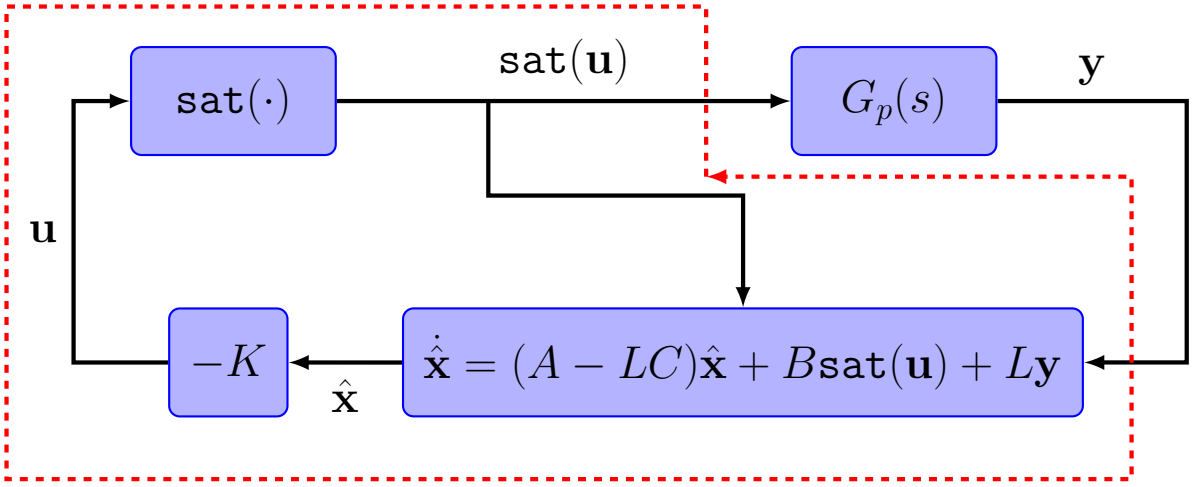
$$\mathtt{sat}(u) = \begin{cases} \alpha & \text{if } u > \alpha \\ u & \text{if } -\alpha < u < \alpha \\ -\alpha & \text{if } u < -\alpha \end{cases}$$

  whereas the input to the estimator is u.

- Can improve the response to some extent if the compensator is modified so that it also sees $\mathtt{sat}(u)$ rather than u.

- Compensator for the saturated DOFB problem

$$\begin{aligned} \dot{\hat{\mathbf{x}}} &= (A - LC)\hat{\mathbf{x}} + B\mathtt{sat}(\mathbf{u}) + L\mathbf{y} \\ \mathbf{u} &= -K\hat{\mathbf{x}} \end{aligned}$$



Code: DOFB servo (lqservo.m)

```
% DOFB servo
% Jonathan How, MIT, Fall 10
clear all;close all;j=sqrt(-1);
%
G=tf([1 2],conv([1 -1],[1 -1 2]));
[N,D]=tfdata(G,'v');roots(N);
[a,b,c,d]=ssdata(G);ns=size(a,1);
Ruu=.001;
Rxx=c'*c;
[klqr,P,E]=lqr(a,b,Rxx,Ruu);
z=zeros(ns,1);
Rxx1=[Rxx z;z' 10];
Rxx2=[Rxx z;z' 1];

abar=[a z;-c 0];bbar=[b;0];
kbar1=lqr(abar,bbar,Rxx1,Ruu);
kbar2=lqr(abar,bbar,Rxx2,Ruu);

% use dual of LQR to design estimator
[l,QQ,EE]=lqr(a',c',b*b',.01);l=l';

% form compensators
ac1=[a-l*c-b*kbar1(1:ns) -b*kbar1(end);z' 0];
ac2=[a-l*c-b*kbar2(1:ns) -b*kbar2(end);z' 0];
by=[l;-1];br=[l*0;1];
cc1=-[kbar1];cc2=-[kbar2];

% form closed-loop dynamics
acl1=[a -b*kbar1;[l;-1]*c ac1];
acl2=[a -b*kbar2;[l;-1]*c ac2];
bcl=[b*0;br]; % r input matrix for basic form - changed below
ccl=[c c*0 0];  % not changed below

% using the Nbar form of closed loop
acl3=[a -b*klqr;l*c a-b*klqr-l*c];
bcl3=[b;b];
ccl3=[c c*0];
Nbar=-inv(ccl3*inv(acl3)*bcl3);
bcl3=bcl3*Nbar;

figure(1);t=[0:.01:5]';
[y2,t2]=step(ss(acl2,bcl,ccl,0),t);
[y1,t1]=step(ss(acl1,bcl,ccl,0),t);
[y3,t3]=step(ss(acl3,bcl3,ccl3,0),t);
plot(t1,y1,t2,y2,'r--',t3,y3,'m:','LineWidth',2)
xlabel('time');ylabel('Response')
legend('Int Penalty 10','Int Penalty 1','Nbar Form','Location','SouthEast')
export_fig lqservo1 -pdf

Gc_uy=tf(ss(ac1,by,cc1,0));
[nn,dd]=tfdata(Gc_uy,'v');
figure(2);
rlocus(-G*Gc_uy) % is a positive feedback loop now
hold on;plot(eig(acl1)+j*eps,'bd','MarkerFaceColor','b');hold off
hold on;plot(eig(a)+j*eps,'mv','MarkerFaceColor','m');hold off
hold on;plot(roots(N)+j*eps,'ro');hold off
hold on;plot(roots(nn)+j*eps,'ro','MarkerFaceColor','r');hold off
hold on;plot(roots(dd)+j*eps,'gs','MarkerFaceColor','g','MarkerSize',5);hold off
legend('vs \alpha','CLP','OL-P','OL-Z','Gc-Z','Gc-P','Location','NorthEast')
title('Int Penalty 10')
axis([-4 2 -5 5])
export_fig lqservo2 -pdf

Gc_uy=tf(ss(ac2,by,cc2,0));
[nn,dd]=tfdata(Gc_uy,'v');
figure(3);
rlocus(-G*Gc_uy) % is a positive feedback loop now
hold on;plot(eig(acl2)+j*eps,'bd','MarkerFaceColor','b');hold off
hold on;plot(eig(a)+j*eps,'mv','MarkerFaceColor','m');hold off
hold on;plot(roots(N)+j*eps,'ro');hold off
hold on;plot(roots(nn)+j*eps,'ro','MarkerFaceColor','r');hold off
hold on;plot(roots(dd)+j*eps,'gs','MarkerFaceColor','g','MarkerSize',5);hold off
legend('vs \alpha','CLP','OL-P','OL-Z','Gc-Z','Gc-P','Location','NorthEast')
title('Int Penalty 1')
```

```
75 axis([-4 2 -5 5])
76 export_fig  lqservo3 -pdf
77
78 % Use R=/= formulation
79 R=-c'/1.75;
80 figure(4);t=[0:.01:5]';
81 [y4,t4]=step(ss(acl1,[-b*kbar1(1:ns)*R;-b*kbar1(1:ns)*R;1],ccl,0),t);
82 %[y4,t4]=step(ss(acl2,[-b*kbar2(1:ns)*R;-b*kbar2(1:ns)*R;1],ccl,0),t);
83 plot(t1,y1,t4,y4,t3,y3,'m:','LineWidth',2)
84 %plot(t2,y2,t4,y4,t3,y3,'m:','LineWidth',2)
85 xlabel('time');ylabel('Response')
86 legend('Int Penalty 10 - no R','Int Penalty 10 - with R','Nbar Form','Location','SouthEast')
87 export_fig  lqservo4 -pdf
```

16.30 / 16.31 Feedback Control Systems

Fall 2010