Computing loop gains and closed-loop transfer functions

Joao Pedro Gomes and Joao Miranda Lemos, December 2016

These notes discuss how to compute loop transfer functions and closed-loop transfer functions for the flexible bar lab project.

Contents

State-space representation

Using the notation of Franklin, Powell & Workman ("Digital Control of Dynamic Systems", 3rd ed., Addison-Wesley, 1997), the system is represented by the state-space model

$$x(k+1) = \Phi x(k) + \Gamma u(k)$$

$$y(k) = Hx(k).$$

The Matlab variables, however, are named as in other lab resources: $\Phi \rightarrow$ `A`, $\Gamma \rightarrow$ `B`, $H \rightarrow$ `C`, $0 \rightarrow$ `D`.

```
bdclose('all')
close all

load('modelo_50hz.mat')
whos A B C D
G = ss(A,B,C,0,Ts);
```

```
  Name      Size            Bytes  Class     Attributes

  A         5x5               200  double
  B         5x1                40  double
  C         1x5                40  double
  D         1x1                 8  double
```

The regulator and observer are designed as described in `controltuning.m` and repeated here for convenience.

```
% Regulator design
Q = C'*C;
R = 100;
[K,S,E] = dlqr(A,B,Q,R);

% Current estimator design
```

```
w2 = 100;
QE = eye(size(A))*w2;
RE = 1;
G1 = eye(size(A));
[M,P,Z,EE] = dlqe(A,G1,C,QE,RE);
```

The symbols for the regulator and current estimator are related to the Matlab variables as follows: $K \rightarrow$ K, $L_c \rightarrow M$.

```
whos K M
```

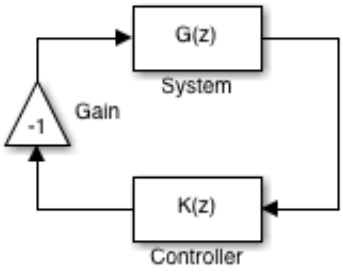
```
  Name      Size             Bytes  Class     Attributes

  K         1x5                 40  double
  M         5x1                 40  double
```

Loop transfer function overview

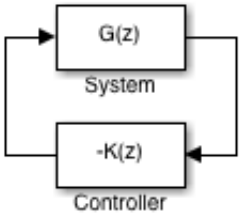
The plant is represented by the transfer function $G(z)$ and is used in a feedback configuration with a controller. In the absence of an external reference input this may be represented as shown in the figure, where $K(z)$ denotes the controller transfer function.

```
open_system('loop1')
```



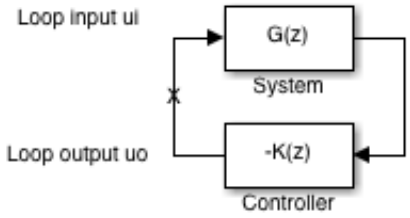
The loop transfer function is $T(z) = K(z)G(z)$. For analysis it may be convenient to lump together $K(z)$ and the minus sign.

```
open_system('loop2')
```



Then, breaking the loop as shown in the figure below and computing the transfer function from the loop input u_i to the loop output u_o will yield $-T(z)$.

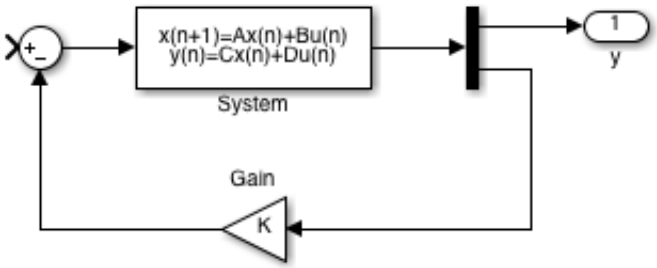
```
open_system('loop3')
```



Linear state feedback

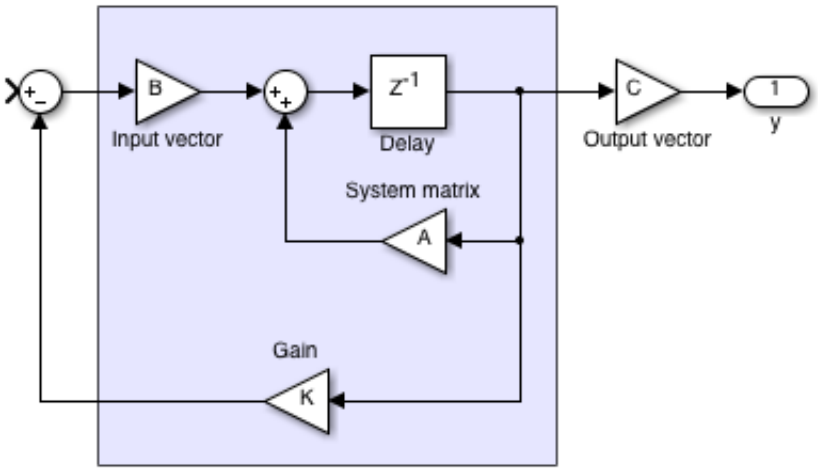
For simulation purposes, the linear quadratic regulated system may be implemented with a discrete state-space block.

```
open_system('lqr1')
```



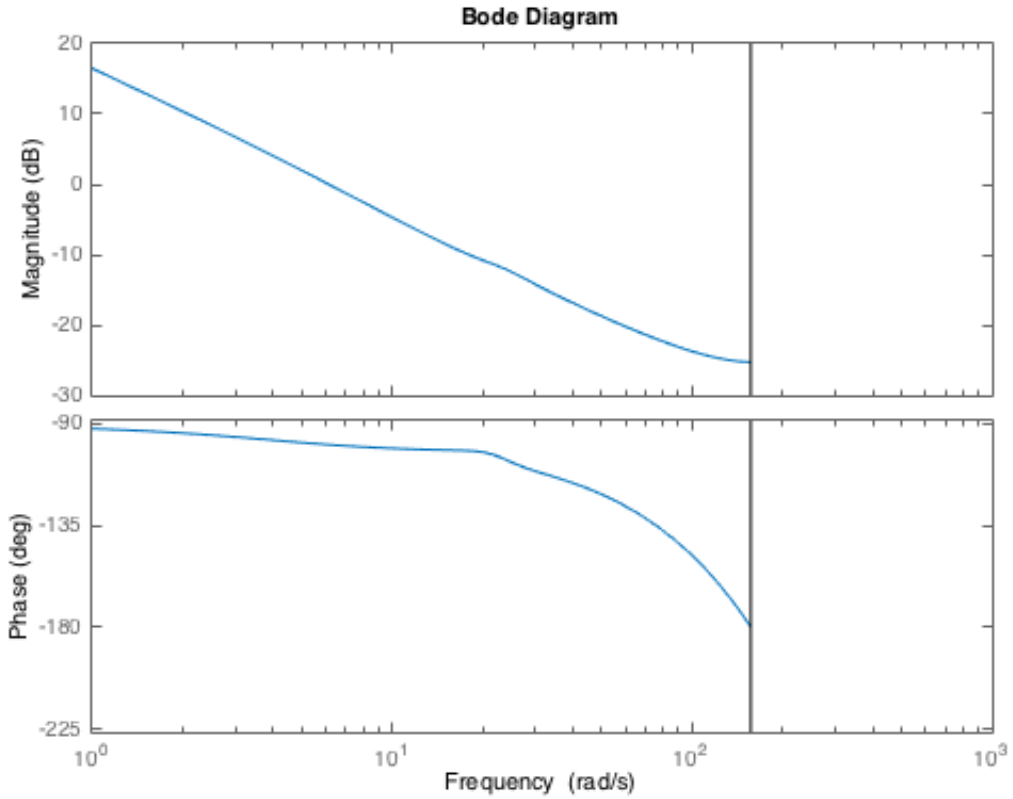
The following equivalent figure shows the internal structure.

```
open_system('lqr2')
```



The shaded area makes it clear that the loop transfer function may be described by a discrete state-space model with input Γ, dynamic matrix Φ and output vector K. Then, the frequency response may be readily obtained.

```
T_lqr = ss(A,B,K,0,Ts);
bode(T_lqr)
```

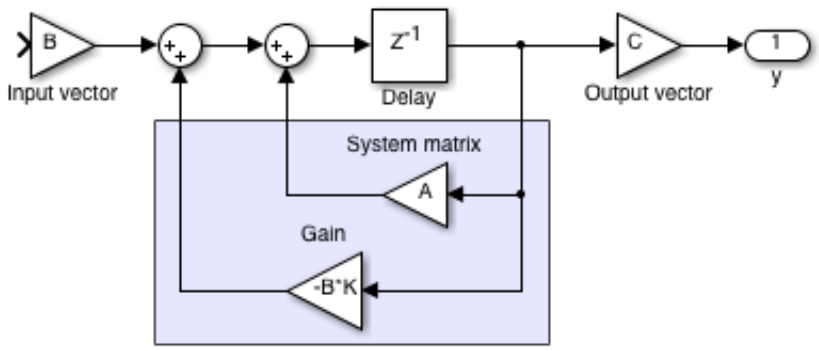


For the closed-loop frequency response we first compute the gain for the external input

```
N = inv([A-eye(size(A)), B; C,0])*[zeros(size(A,1),1);1];
Nx = N(1:end-1,:);
Nu = N(end,:);
Nbar = Nu+K*Nx;
```

Pushing back Γ across the outer sum in the previous block diagram and rearranging we get

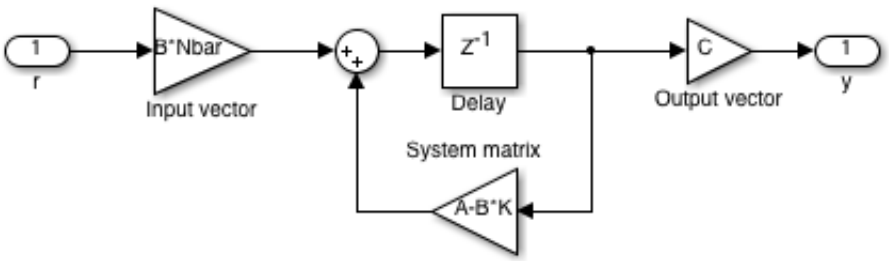
```
open_system('lqr3')
```

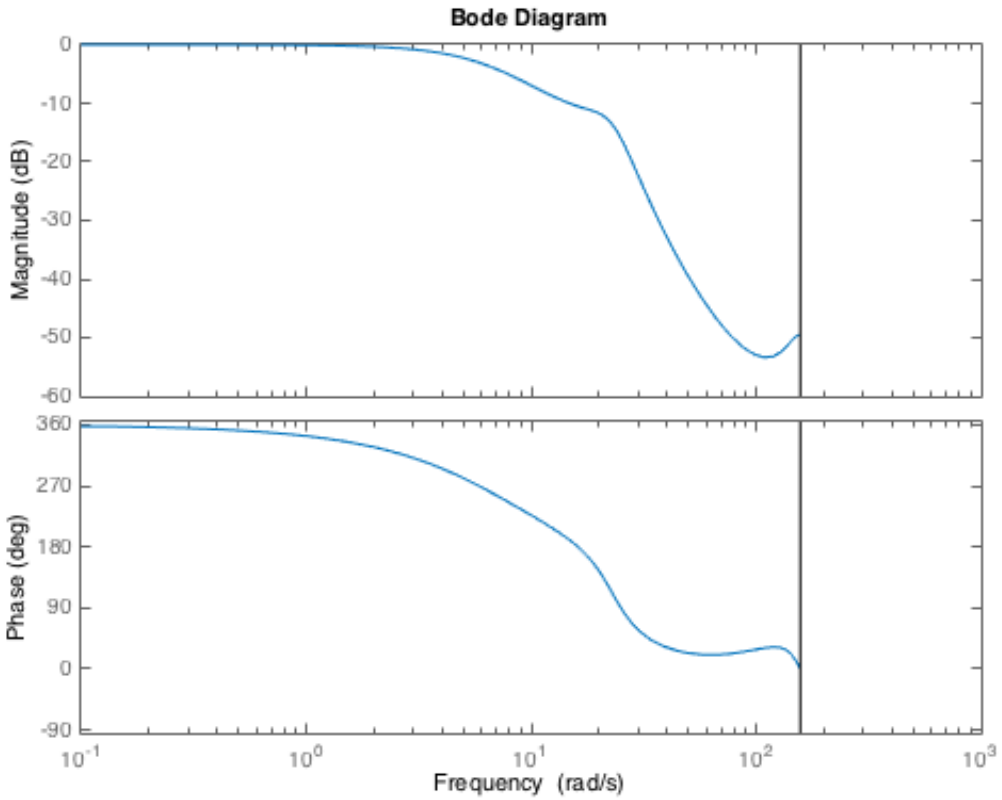


Adding the external input and combining the two shaded feedback connections directly yields a state-space realization for the closed-loop system.

```
open_system('lqr4')
```

```
C_lqr = ss(A-B*K,B*Nbar,C,0,Ts);
bode(C_lqr)
```

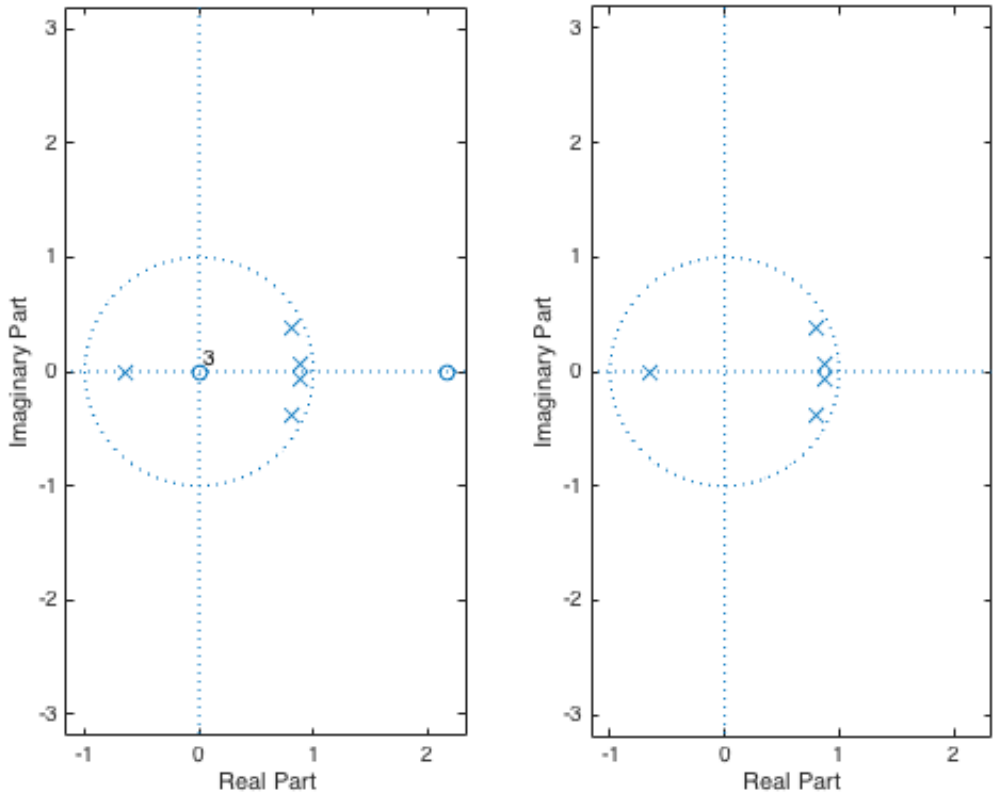




Note that the closed-loop poles should match, up to a permutation, those obtained from the call to `dlqr`.

```
[NC_lqr,DC_lqr] = tfdata(C_lqr,'v');

subplot(1,2,1)
zplane(NC_lqr,DC_lqr)
ax = axis;
subplot(1,2,2)
zplane([],E)
axis(ax)
```



Estimator/observer

In a *predictive* estimator the state estimate $\bar{x}$ is updated as

$$\bar{x}(k+1) = \Phi\bar{x}(k) + \Gamma u(k) + L_p(y(k) - H\bar{x}(k)),$$

where L_p denotes the vector of predictive observer gains. Our goal is to design a *current* estimator, whose estimate $\hat{x}$ is related to the predictive one as

$$\bar{x}(k+1) = \Phi\hat{x}(k) + \Gamma u(k).$$

Plugging this into the predictive update equation, rearranging and defining the vector of current observer gains $L_c = \Phi^{-1}L_p$, we get the recursive expression

$$\hat{x}(k) = \underbrace{(\Phi - L_c H\Phi)}_{\Phi_E}\hat{x}(k-1) + \underbrace{(\Gamma - L_c H\Gamma)}_{\Gamma_E}u(k-1) + L_c y(k).$$

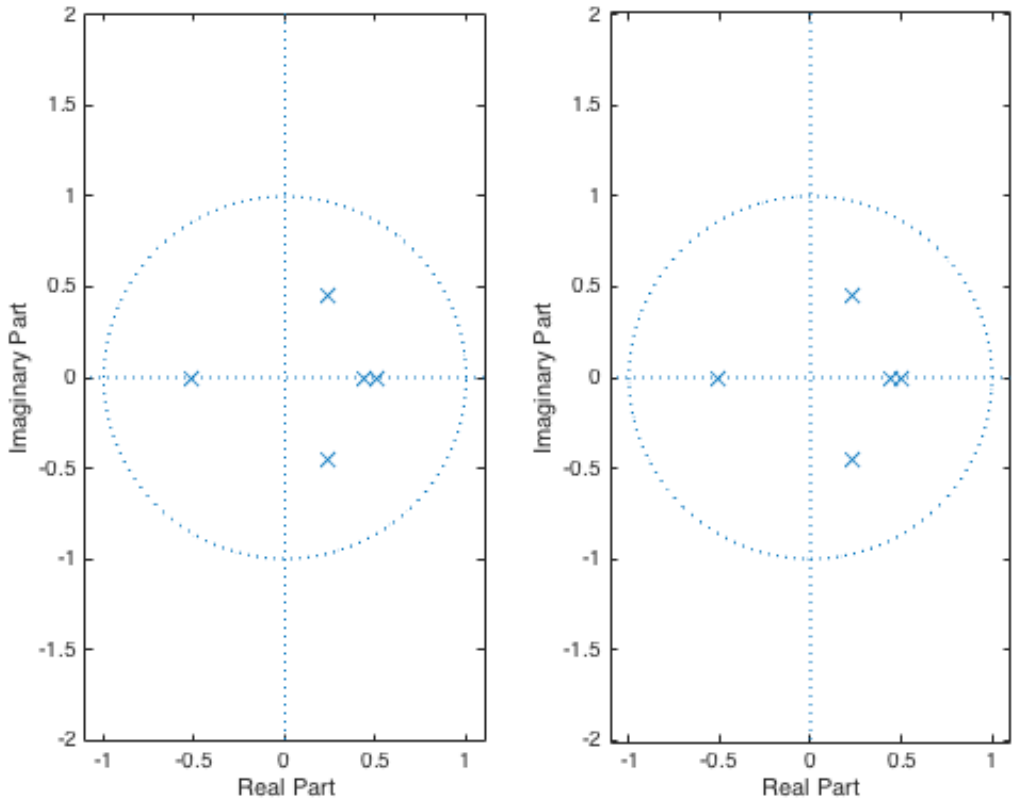
```
PHIE = A-M*C*A;
GAMMAE = B-M*C*B;
```

The update recursion above readily provides a state-space-like architecture (but not exactly state-space, as discussed below) for the current estimator that is equivalent to the one given on the course notes on linear state feedback. The latter emphasizes the close coupling between predictive and current estimates as two outputs taken from a single block diagram (hence having the same dynamics).

Check the consistency of the closed-loop poles with the call to `dlqe`.

```
subplot(1,2,1)
zplane([],eig(PHIE))
subplot(1,2,2)
```

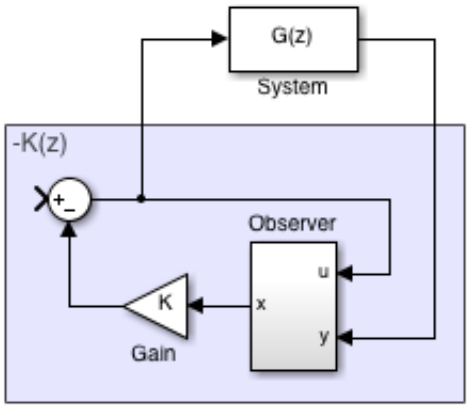
```
zplane([],EE)
```



Coupled observer and state feedback

Regardless of how the current estimator update equations are implemented, the coupled control system (no external reference) is as follows.

```
open_system('lqg1')
```



The shaded region is recognized as (minus) the controller transfer function discussed in the general overview above. Two alternatives are described to determine the loop transfer function.

Method 1: Explicitly determine the controller response

Let us explicitly determine $K(z)$. Due to the internal feedback in the controller we have $u(k) = -K\hat{x}(k)$, hence the estimator update equation becomes

$$\hat{x}(k) = (\Phi_E - \Gamma_E K)\hat{x}(k-1) + L_c y(k).$$

The output of this block is $u_o(k) = -K\hat{x}(k)$. If we advance time $k \to k+1$ in the update equation we *almost* get a state-space description

$$\hat{x}(k+1) = (\Phi_E - \Gamma_E K)\hat{x}(k) + L_c y(k+1)$$

$$u_o(k) = -K\hat{x}(k).$$

The problem here is that the state update at time k depends on the *future* plant output $y(k+1)$. Setting this difficulty aside for the time being, we define a modified input $y_2(k) = y(k+1)$ and obtain a state-space model for the modified controller, $K_2(z)$

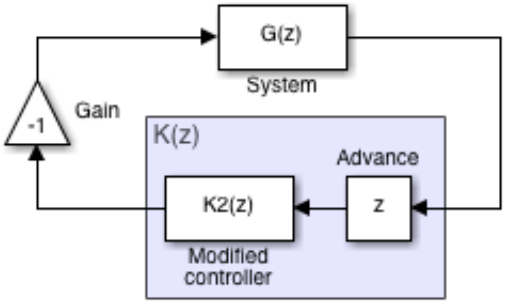
$$\hat{x}(k+1) = (\Phi_E - \Gamma_E K)\hat{x}(k) + L_c y_2(k)$$

$$-u_o(k) = K\hat{x}(k).$$

```
K2_lqg = ss(PHIE-GAMMAE*K,M,K,0,Ts);
```

The actual controller is obtained by cascading K_2 with a unit *advance*, which is obviously not causal, but the relevant issue here is whether its cascade with K_2 is causally realizable.

```
open_system('lqg2')
```



Fortunately this is the case here, as the numerator polynomial of $K_2(z)$ has a zero coefficient for the term z^0, and hence corresponds to *strictly positive* delays, which can be advanced by one sample without jeopardizing causality (for computing the loop transfer function alone the causality condition is even less stringent, as only the cascade $K_2(z)G(z)$ needs to be "advanceable").

```
[NK2_lqg,DK_lqg] = tfdata(K2_lqg,'v');
NK2_lqg
```

```
NK2_lqg =

         0    0.7287   -1.3287    0.4232    0.5276   -0.3190
```

Now obtain the final numerator coefficients by eliminating the initial zero (actually shift it to the end, to preserve the length of this vector as needed by some subsequent function calls).

```
NK_lqg = wshift('1D',NK2_lqg,1);
K_lqg = tf(NK_lqg,DK_lqg,Ts);
```

Finally, build the loop transfer function.

```
T_lqg = series(G,K_lqg);
```

Adding an external reference, it would be tempting to simply compute the closed-loop transfer function as

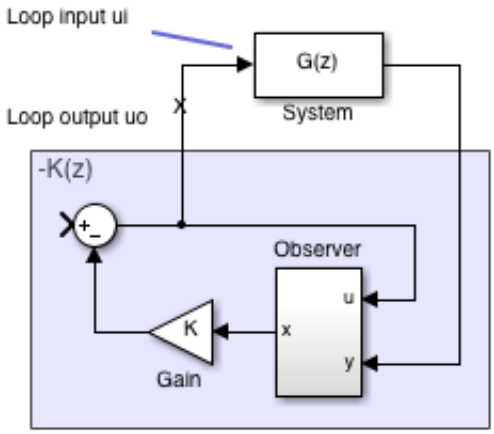
```
C_lqg = feedback(G,K_lqg)*Nbar;
```

However, this is incorrect because it does not properly account for the direct connection of r to the observer input (through a +/- sum block). The resulting transfer function has wrong numerator coefficients.

Method 2: Compute the joint system/controller dynamics

We temporarily break the loop as shown in the figure below and express the joint dynamics of the system and observer in terms of the state variables $x(k)$, $\hat{x}(k)$ and the loop input $u_i(k)$.

```
open_system('lqg3')
```



For the system, this is trivially given by

$$x(k+1) = \Phi x(k) + \Gamma u_i(k).$$

For the current estimator, we must express the variables $y(k+1)$, $u(k)$ in the update expression

$$\hat{x}(k+1) = \Phi_E \hat{x}(k) + \Gamma_E u(k) + L_c y(k+1)$$

in terms of $u_i(k)$ and the state variables. But this is easily done as follows

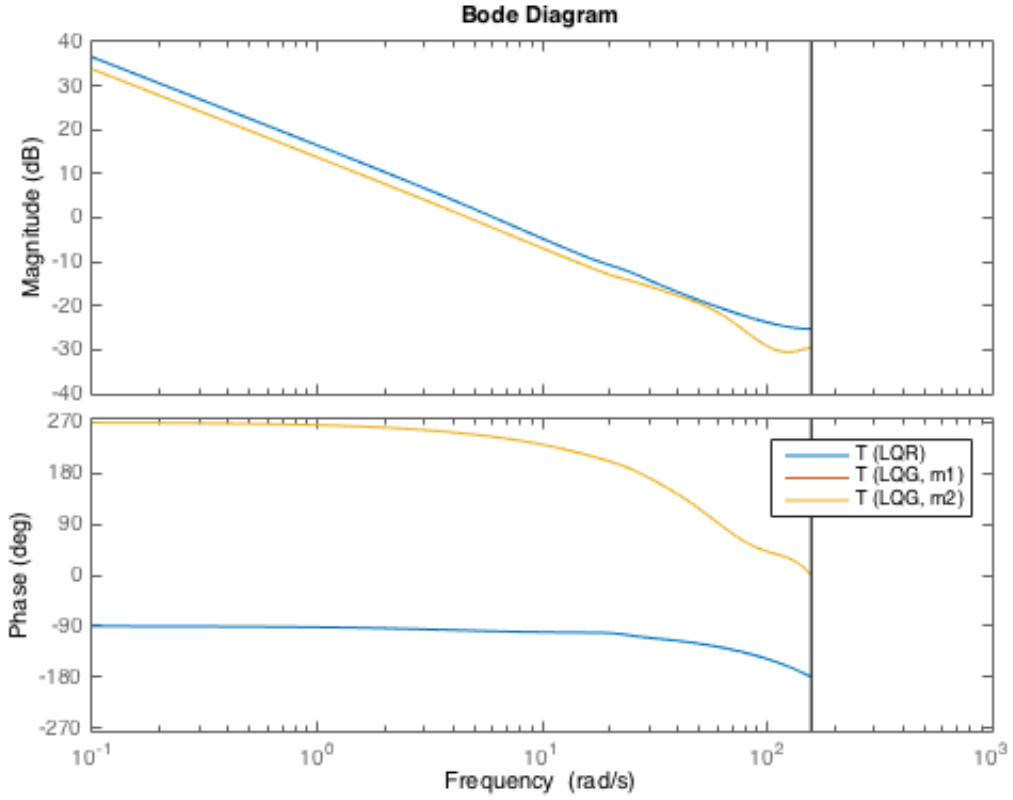
$$\hat{x}(k+1) = \Phi_E \hat{x}(k) + \Gamma_E \underbrace{u(k)}_{-K\hat{x}(k)} + L_c \underbrace{y(k+1).}_{H(\Phi x(k)+\Gamma u_i(k))}$$

These expressions and the output equation $u_o(k) = -K\hat{x}(k)$ lead to the following state-space model to compute the loop transfer function

$$\begin{bmatrix} x(k+1) \\ \hat{x}(k+1) \end{bmatrix} = \begin{bmatrix} \Phi & 0 \\ L_c H\Phi & \Phi_E - \Gamma_E K \end{bmatrix} \begin{bmatrix} x(k) \\ \hat{x}(k) \end{bmatrix} + \begin{bmatrix} \Gamma \\ L_c H\Gamma \end{bmatrix} u_i(k)$$

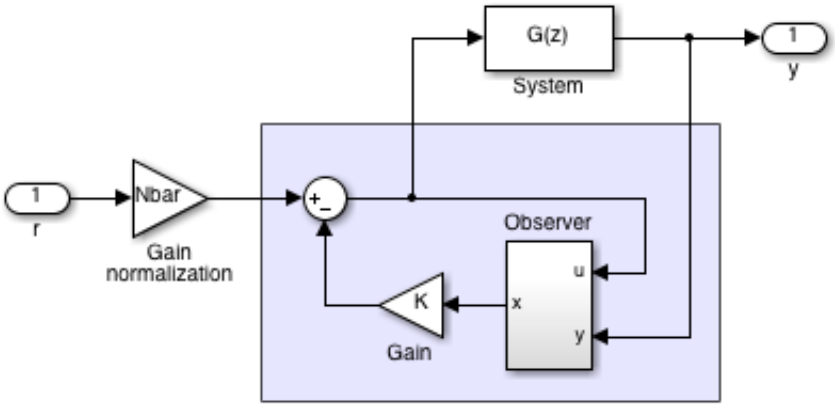
$$-u_o(k) = \begin{bmatrix} 0 & K \end{bmatrix} \begin{bmatrix} x(k) \\ \hat{x}(k) \end{bmatrix}.$$

```
T2_lqg = ss([A zeros(size(A)); M*C*A PHIE-GAMMAE*K], ...
            [B; M*C*B],[zeros(size(K)) K],0,Ts);
clf
bode(T_lqr,T_lqg,T2_lqg)
legend('T (LQR)','T (LQG, m1)','T (LQG, m2)')
```



To get the closed-loop transfer function we now close the loop ($u_i \to u$) and add the external reference input.

```
open_system('lqg4')
```



We have $u(k) = \bar{N}r(k) - K\hat{x}(k)$, and the state update expressions read

$$x(k+1) = \Phi x(k) + \Gamma \underbrace{u(k)}_{\bar{N}r(k) - K\hat{x}(k)}$$

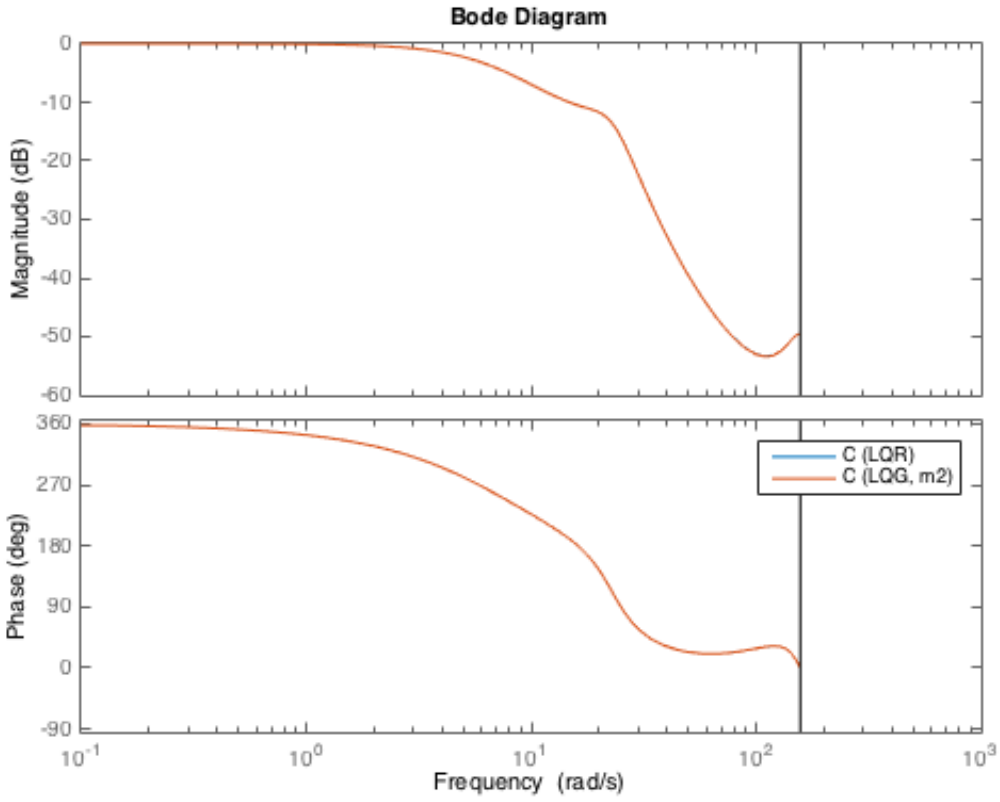
$$\hat{x}(k+1) = \Phi_E \hat{x}(k) + \Gamma_E \underbrace{u(k)}_{\bar{N}r(k) - K\hat{x}(k)} + L_c \underbrace{y(k+1).}_{H(\Phi x(k) + \Gamma(\bar{N}r(k) - K\hat{x}(k)))}$$

The state-space model for the closed-loop system is

$$\begin{bmatrix} x(k+1) \\ \hat{x}(k+1) \end{bmatrix} = \begin{bmatrix} \Phi & -\Gamma K \\ L_c H\Phi & \Phi_E - \Gamma_E K - L_c H\Gamma K \end{bmatrix} \begin{bmatrix} x(k) \\ \hat{x}(k) \end{bmatrix} + \begin{bmatrix} \Gamma \\ L_c H\Gamma + \Gamma_E \end{bmatrix} \bar{N}r(k)$$

$$y(k) = \begin{bmatrix} H & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ \hat{x}(k) \end{bmatrix}.$$

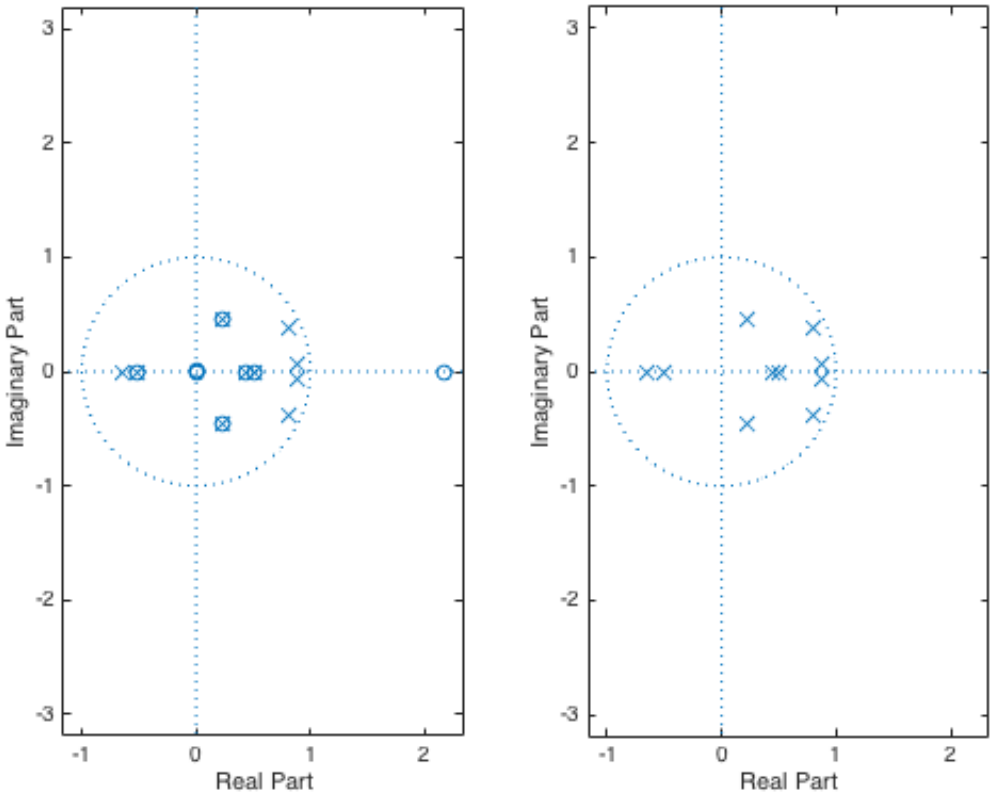
```
C2_lqg = ss([A -B*K; M*C*A PHIE-GAMMAE*K-M*C*B*K], ...
            [B; M*C*B+GAMMAE]*Nbar,[C zeros(size(C))],0,Ts);
bode(C_lqr,C2_lqg)
legend('C (LQR)','C (LQG, m2)')
```



Finally, check the separation principle for pole locations.

```
[NC2_lqg,DC2_lqg] = tfdata(C2_lqg,'v');

subplot(1,2,1)
zplane(NC2_lqg,DC2_lqg)
ax = axis;
subplot(1,2,2)
zplane([],[E; EE])
axis(ax)
```



Compare the left pole-zero plot with the one obtained earlier for direct state feedback using `dlqr` , and see how it approximates it through selective pole-zero cancellations.

Published with MATLAB® R2015a