

Chapter 7 HW 2 Solution

Problem 1. (a) To find the combined state equation for the plant+estimator+control law, the necessary equations are:

$$\text{Plant state equation: } \mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}u(k) \quad (1)$$

$$\text{Plant measurement equation: } y_m(k) = \mathbf{C}_m\mathbf{x}(k) \quad (2)$$

$$\text{Estimator equation: } \hat{\mathbf{x}}(k+1) = \mathbf{\Phi}\hat{\mathbf{x}}(k) + \mathbf{\Gamma}u(k) + \mathbf{L}[y_m(k) - \mathbf{C}_m\hat{\mathbf{x}}(k)] \quad (3)$$

$$\text{Control law using state estimate: } u(k) = -\mathbf{K}\hat{\mathbf{x}}(k) \quad (4)$$

NOTE: “Measurement” output matrix \mathbf{C}_m in equation (2) must produce the measurement (the particular output variable that is sensed and fed to the controller).

Using the composite state vector $[\mathbf{x}(k) \quad \hat{\mathbf{x}}(k)]^T$, the state equation for the plant + controller is:

$$\begin{bmatrix} \mathbf{x}(k+1) \\ \hat{\mathbf{x}}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi} & -\mathbf{\Gamma}\mathbf{K} \\ \mathbf{L}\mathbf{C}_m & \mathbf{\Phi} - \mathbf{\Gamma}\mathbf{K} - \mathbf{L}\mathbf{C}_m \end{bmatrix} \begin{bmatrix} \mathbf{x}(k) \\ \hat{\mathbf{x}}(k) \end{bmatrix} \quad (5)$$

There is not really a standard “output” equation; you can define whatever output equation you need to get the output of interest.

(b) To get a state-variable model of just the controller, use (3) and (4) above, and you will get

$$\hat{\mathbf{x}}(k+1) = [\mathbf{\Phi} - \mathbf{\Gamma}\mathbf{K} - \mathbf{L}\mathbf{C}_m] \hat{\mathbf{x}}(k) + \mathbf{L}y_m(k) \quad (6)$$

$$u(k) = -\mathbf{K}\hat{\mathbf{x}}(k) + 0y_m(k) \quad (7)$$

These would be used in a discrete state-space block for the controller in a Simulink model (or in the actual controller).

Problem 2. We are told that the measurement is load angle θ_l in degrees, hence the plant output matrix which produces this measurement (for estimation purposes) will be called \mathbf{C}_e , and is

$$\mathbf{C}_e = [0 \quad 180/(n\pi) \quad 0] = [0 \quad 1.1459 \quad 0]. \quad (8)$$

Estimator Design. (a) I had placed the three poles for the control law at $s = -100, -70 \pm j70$ (each with $\omega_n \approx 100$ rad/s), so I placed the estimator error poles to be about twice as fast, or

$$s = -200, -140 \pm j140 \quad \implies \quad z = 0.1353, 0.0419 \pm j0.2430 \quad (9)$$

Using the `place()` function to solve for estimator gains yielded

```
>> L = place(Phi', Ce', e_poles_z)
```

```
L = -13.7204    0.9812    57.0629
```

where \mathbf{L} must be transposed to yield

$$\mathbf{L} = \begin{bmatrix} -13.7204 \\ 0.9812 \\ 57.0629 \end{bmatrix} \quad (10)$$

This estimator is generating current, position, and velocity from a measurement of position.

You can then put the controlled system into the form of (5) in Problem 1 and simulate from an initial condition; I chose $[\mathbf{x}(0) \quad \hat{\mathbf{x}}(0)]^T = [0 \quad 2\pi \quad 0 \quad 0 \quad 0 \quad 0]^T$ which is the same plant initial condition as the previous assignment (one motor revolution), and the estimator state still zero. Responses on next page.

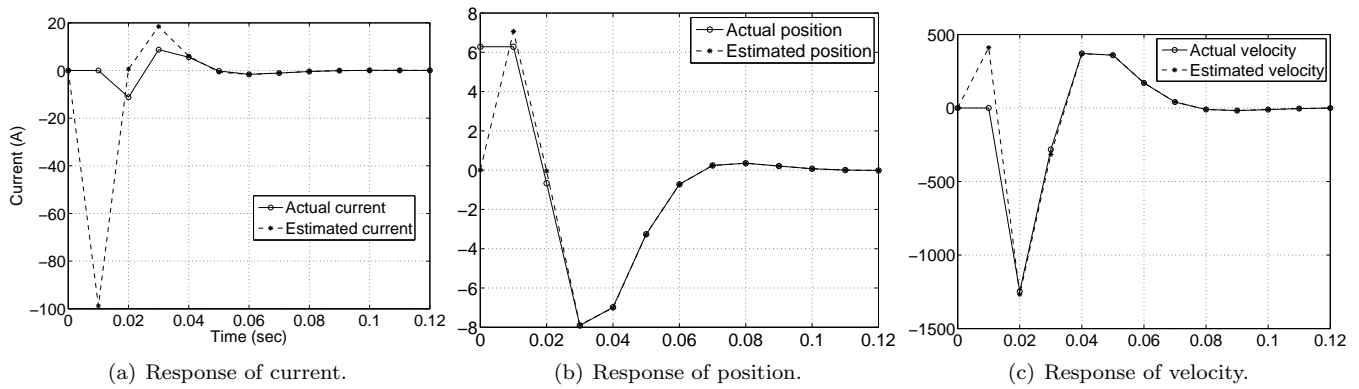


Figure 1: Actual and estimated state responses.

In practice, you would have the controller running when you displaced the system, so this level of estimation error would not develop. But all state variables converge pretty quickly.

(b) Can you design an estimator using a measurement of load velocity or motor current? Try load velocity first...in this case the measured output matrix \mathbf{C}_m is

$$\mathbf{C}_m = [0 \quad 0 \quad 60/(2\pi n)] \quad (11)$$

and the first thing to do is check for *observability*. This can be done by finding the rank of the observability matrix

$$\mathcal{O} = [\mathbf{C} \quad \mathbf{C}\Phi \quad \mathbf{C}\Phi^2 \quad \dots \quad \mathbf{C}\Phi^{n-1}] \quad (12)$$

The observability matrix may be found using the MATLAB `ctrb` function, with transposed arguments, as:

```
>> Cm = [0 0 60/(2*pi*n)]; % Output matrix that produces load velocity (rpm)
```

```
>> O = ctrb(Phi',Cm') % Compute the observability matrix O
```

```
O =
     0     0.2255    0.0775
     0         0         0
    0.1910    0.0691    0.0237
```

By inspection matrix \mathcal{O} does not have full rank (3); this may be verified by

```
>> rank(O)
```

```
ans = 2
```

Of course this could have been done in one step:

```
>> rank(ctrb(Phi',Cm'))
```

```
ans = 2
```

So one *cannot* construct a state estimator using a measurement of only velocity.

Same thing is true of motor current; use $\mathbf{C}_m = [1 \quad 0 \quad 0]$ and you will find the rank of the observability matrix \mathcal{O} is 2.

(c) Here's the Simulink model I showed in class. I used a continuous-domain state-space model for the motor/load, so I was able to impose an initial condition on it. Used a discrete state-space block for the controller, with the equations of (6)–(7). It looks like the line connecting the $u(k)$ output from the controller going up to the ZOH at upper left got cut out somehow. Anyway, we saw that the measurement noise had a substantial effect on the performance of the system. The quantization effect was almost negligible—the closed loop mitigates the effect of quantization “noise” at that point.

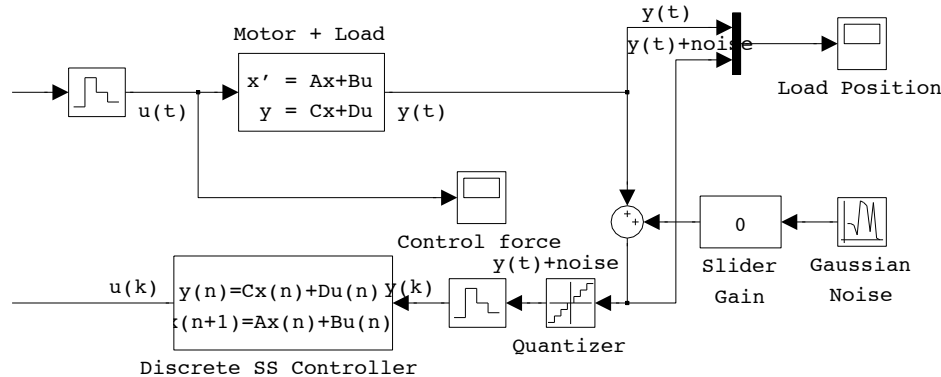


Figure 2: Simulink model of motor/load system.

Controller Transfer Function (not required, just FYI). Interestingly, it is not difficult to find the *transfer function* of the controller (and the plant) and examine the *root locus* of the system. Here's how you do it.

Use the MATLAB `ss2tf` function to produce a transfer function from the state-variable model. For this we need the controller (A, B, C, D) matrices—from (2)–(4) of Problem 1 we get

$$\hat{\mathbf{x}}(k+1) = [\Phi - \Gamma\mathbf{K} - \mathbf{L}C_m]\hat{\mathbf{x}}(k) + \mathbf{L}y(k) \quad (13)$$

$$u(k) = -\mathbf{K}\hat{\mathbf{x}}(k) + 0y(k) \quad (14)$$

so that

$$\mathbf{A} = \Phi - \Gamma\mathbf{K} - \mathbf{L}C_m, \quad \mathbf{B} = \mathbf{L}, \quad \mathbf{C} = -\mathbf{K}, \quad \mathbf{D} = 0$$

Substituting numerical values for all the terms, we obtain the following from MATLAB:

```
>> [numc,denc] = ss2tf(Phi-Gamma*K-L*Cm,L,-K,0)
```

```
numc = 0 -6.1377 7.0548 -1.6982
```

```
denc = 1.0000 -0.0031 -0.6086 -0.7752
```

Hence the controller transfer function is

$$D(z) = \frac{U(z)}{Y(z)} = \frac{0z^3 - 6.1377z^2 + 7.0548z - 1.6982}{z^3 - 0.0031z^2 - 0.6086z - 0.7752} = \frac{-6.1377(z - 0.8062)(z - 0.3432)}{(z - 1.137)(z + 0.5670 \pm j0.6001)} \quad (15)$$

where I obtained the factored form using the MATLAB `zpk` and `roots` functions. Note that the controller by itself is unstable (pole at $z = 1.137$, outside the unit circle!) Also, note that I factored out the negative constant -6.1377 from the numerator since negative feedback is already implied in the control law (recall $u(k) = -\mathbf{K}\hat{\mathbf{x}}(k)$).

System Root Locus. It is actually not hard to verify the state-space design using the root locus. For this we need also the plant transfer function; this can be found using the `ss2tf` function to be

$$G(z) = \frac{Y(z)}{U(z)} = \frac{0.18022(z + 0.8419)(z + 0.002897)}{z(z - 1)(z - 0.3436)} \quad (16)$$

We can draw the root locus using MATLAB `rlocus(D*G)`, but remember that the `rlocus` function presumes negative feedback, so we must remove the “-” sign in the $D(z)$ of (13). So we draw the locus using

```
>> rlocus(-Dc*Gp); % Dc is the controller and Gp is the plant transfer function
```

Figure 1 shows the root locus, and I clicked to place the poles where I had desired ($\zeta = 0.707$, $\omega_n = 100$ rad/s); note that this frequency corresponds to $0.315\pi/T$. The “red crosses” show the pole locations—the three poles to the right correspond to the control law; the three to the left are the estimator (faster).

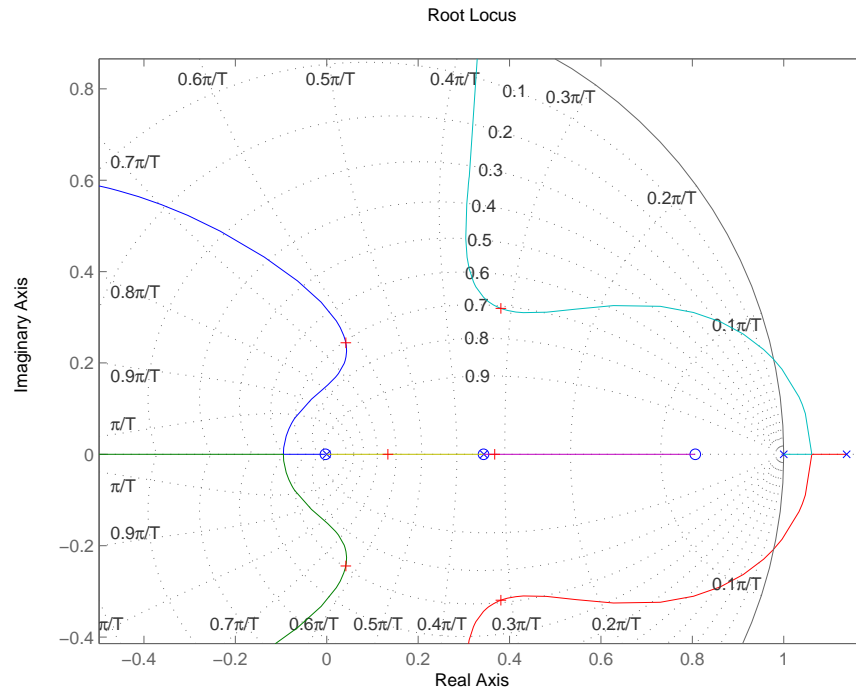


Figure 3: Root-locus of plant + controller.

Using the `rlocfind` function, I clicked on the dominant (rightmost) RL branch exactly at the $\zeta = 0.7$ line, and got the following:

```
>> Kgain = rlocfind(-Dc*Gp)
```

```
Select a point in the graphics window
```

```
selected_point = 0.3816 + 0.3210i % I had clicked exactly at zeta = 0.7
```

```
Kgain = 0.9998
```

Getting a gain of 0.9998 (almost exactly 1.0000) illustrates that the poles are located as shown in Figure 1 with a gain of 1—that is, the given controller places them there. So the root locus confirms the state-variable design. Science is wonderful!