

DC Motor/Load Case Study

1 Introduction

We have considered the DC motor/load system in the Chapter 6 HW and in Chapter 7 HW 1 and 2. My intent is to continue with those designs, and here add the “final” piece of the controller in the form of a reference input. Hereafter the term “plant” refers to the motor + gear train + load.

As you remember, this system used a commercially-available DC motor to control the angular position of a cylindrical load. There is a gear train between motor and load, and there is a measurement of load position (deg) available. The single motor input is armature voltage.

2 Plant Model

We have modeled this plant in two previous homework assignments.

2.1 Continuous Model

We first obtained a continuous state-space model of the motor/load in Chapter 6—complete details are in the Chapter 6 HW Solution. The relevant terms here are the **A**, **B**, **C**, **D** matrices for the plant. The state vector is composed of motor armature current i_a , motor angular position θ_m (rad), and motor angular velocity ω_m (rad/s), hence

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} i_a \\ \theta_m \\ \omega_m \end{bmatrix} \quad (1)$$

2.2 Discrete-Time Model

In Chapter 7 HW 1 we obtained a discrete state-space model for the plant, resulting in matrices **Φ** , **Γ** , **\mathbf{C}** , **\mathbf{D}** . We assumed a sampling period of $T = 0.01$ sec, thus the sampling frequency is 100 Hz. The relevant equations are

$$\text{Plant state equation: } \mathbf{x}(k+1) = \Phi \mathbf{x}(k) + \Gamma u(k) \quad (2)$$

$$\text{Plant output equation: } y(k) = \mathbf{C} \mathbf{x}(k) \quad (3)$$

3 Controller

In the Chapter 7 HW we designed both a control law and prediction estimator for the plant.

3.1 Control Law

The state feedback control law is

$$u(k) = -\mathbf{K} \mathbf{x}(k), \quad (4)$$

and we placed the poles of the controlled plant to have good damping and 100 rad/s natural frequency. However, we cannot implement this control law since we cannot measure all the state variables.

3.2 State Estimator

We designed a prediction estimator for the plant to be about twice as fast as the control law (could have been faster). The measurement—which is the input to the estimator—was the load position θ_l (deg). This measurement is $y_m(k)$; hence for purposes of estimator design we selected a **\mathbf{C}_m** matrix which produced this measurement.

The estimator equation is

$$\hat{\mathbf{x}}(k+1) = \Phi \hat{\mathbf{x}}(k) + \Gamma u(k) + \mathbf{L}[y_m(k) - \mathbf{C}_m \hat{\mathbf{x}}(k)] \quad (5)$$

and now the control law is implemented using the estimated state:

$$u(k) = -\mathbf{K} \hat{\mathbf{x}}(k), \quad (6)$$

4 Reference Input

The final step is to add the reference input. Both a SISO and a MIMO design will be presented. I'm going to do this development assuming no state estimator, and at the end I'll make some comments about doing this with an estimator (which you will have to do).

4.1 SISO Design

The single reference input $r(k)$ is desired load position (deg), hence the “reference output” $y_r(k)$ must also be load position (deg). We must define an output matrix \mathbf{C}_r to produce this output, which is

$$\mathbf{C}_r = [0 \quad 180/(n\pi) \quad 0] \quad (7)$$

Following the development on pp. 162-163 in the notes, the reference input matrices were found to be

$$\mathbf{N}_x = \begin{bmatrix} 0 \\ 0.8727 \\ 0 \end{bmatrix} \quad (8)$$

$$\mathbf{N}_u = 0 \quad (9)$$

Here the number of plant inputs (the dimension of u) and the number of desired outputs (the dimension of y_r) are the same (both are 1×1) so the matrix to be inverted in equation (7.71) is square, and everything is fine.

Since the plant *is* Type 1, the zero value of \mathbf{N}_u makes sense.

4.1.1 Control Law.

With the reference input, the control law becomes

$$\begin{aligned} u(k) &= -\mathbf{K}[\mathbf{x}(k) - \mathbf{x}_r(k)] \\ &= -\mathbf{K}\mathbf{x}(k) + \mathbf{K}\mathbf{N}_x r(k) \end{aligned} \quad (10)$$

Applying this control law to the plant, we get controlled plant state and output equations

$$\begin{aligned} \mathbf{x}(k+1) &= \Phi \mathbf{x}(k) - \Gamma \mathbf{K} \mathbf{x}(k) + \Gamma \mathbf{K} \mathbf{N}_x r(k) \\ &= [\Phi - \Gamma \mathbf{K}] \mathbf{x}(k) + \Gamma \mathbf{K} \mathbf{N}_x r(k) \end{aligned} \quad (11)$$

and

$$y(k) = \mathbf{C}_r \mathbf{x}(k) \quad (12)$$

where I used output matrix \mathbf{C}_r to produce load position (deg) as the output. If I had wanted a different output (perhaps for simulation purposes) I could have used a different \mathbf{C} matrix without affecting the dynamics of the controlled system.

One could create a MATLAB LTI model of this system by

```
>> sys_siso = ss(Phi-Gamma*K,Gamma*K*Nx,Cr,0,T)
```

where we must include sampling period T to make this a discrete LTI system. You could then use function `lsim(sys_siso,r,t)` to find the response to input \mathbf{r} with time vector \mathbf{t} (the output would be the reference output, since we used matrix \mathbf{C}_r).

4.2 MIMO Design

Now we will use two inputs: desired load position $r(k)$ and desired load velocity $\dot{r}(k)$, we can express this in vector format as

$$\mathbf{r}(k) = \begin{bmatrix} r(k) \\ \dot{r}(k) \end{bmatrix} = \begin{bmatrix} \theta_l \text{ desired (deg)} \\ \omega_l \text{ desired (deg/s)} \end{bmatrix} \quad (13)$$

Hence the “reference output” must be actual load position and actual load velocity (both in degrees). So the output matrix \mathbf{C}_r must now be

$$\mathbf{C}_r = \begin{bmatrix} 0 & 180/(n\pi) & 0 \\ 0 & 0 & 180/(n\pi) \end{bmatrix} \quad (14)$$

In Section 7.9 the issues of MIMO systems are covered—note particularly that quantities n, m, p are the number of states, plant inputs, and system outputs, respectively. So now we have

$$\begin{aligned} n &= 3 \text{ (there are 3 state variables)} \\ m &= 1 \text{ (there is 1 plant input)} \\ p &= 2 \text{ (there are 2 system outputs)} \end{aligned}$$

Again following the development on pp. 162-163 in the notes, the reference input matrices were found to be...well, let's first try to invert the matrix in (7.71)...

```
>> inv([Phi-eye(3) Gamma; Cr zeros(2,1)])
??? Error using ==> inv
Matrix must be square.
```

Referring to equation (7.73), the dimension of this matrix is $(n+p) \times (n+m)$, or 5×4 , and we can't invert it!

So how do we solve for \mathbf{N}_x and \mathbf{N}_u ? Use the **pseudoinverse**—this is discussed in Section 7.9.

If we call this 5×4 matrix Ψ , then the pseudoinverse (sometimes called the generalized inverse and denoted by Ψ^+) is

$$\Psi^+ = [\Psi^T \Psi]^{-1} \Psi^T \quad (15)$$

This is illustrated in equation (7.74). The dimensions of all quantities are given in equation (7.73).

Anyway, using the pseudoinverse, we find that

$$\mathbf{N}_x = \begin{bmatrix} 0 & 0.0001 \\ 0.8727 & 0 \\ 0 & 0.8726 \end{bmatrix} \quad (16)$$

and

$$\mathbf{N}_u = [0 \quad 0.0197] \quad (17)$$

4.2.1 Control Law.

With the reference input, the control law becomes

$$\begin{aligned} u(k) &= -\mathbf{K}[\mathbf{x}(k) - \mathbf{x}_r(k)] + \mathbf{N}_u \mathbf{r}(k) \\ &= -\mathbf{K}\mathbf{x}(k) + \mathbf{K}\mathbf{N}_x \mathbf{r}(k) + \mathbf{N}_u \mathbf{r}(k) \\ &= -\mathbf{K}\mathbf{x}(k) + [\mathbf{K}\mathbf{N}_x + \mathbf{N}_u] \mathbf{r}(k) \end{aligned} \quad (18)$$

Applying *this* control law to the plant, we get controlled plant state and output equations

$$\begin{aligned} \mathbf{x}(k+1) &= \Phi \mathbf{x}(k) - \Gamma \mathbf{K} \mathbf{x}(k) + \Gamma [\mathbf{K} \mathbf{N}_x + \mathbf{N}_u] \mathbf{r}(k) \\ &= [\Phi - \Gamma \mathbf{K}] \mathbf{x}(k) + \Gamma [\mathbf{K} \mathbf{N}_x + \mathbf{N}_u] \mathbf{r}(k) \end{aligned} \quad (19)$$

and

$$y_r(k) = \mathbf{C}_r \mathbf{x}(k) \quad (20)$$

where we used reference output matrix \mathbf{C}_r to produce both load position (deg) and load velocity (deg/s) as the output(s).

As before, one could create a MATLAB LTI state-space model for the MIMO controlled system:

```
>> sys_mimo = ss(Phi-Gamma*K,Gamma*(K*Nx+Nu),Cr,0,T)
```

and we could use `lsim` to find the response to an input (now the input would need to incorporate both position and velocity).

Intuition would suggest that the MIMO system should perform better than the SISO system because it is receiving more information.

5 Controller Model

In Section 4 we added both a single and multiple reference inputs and found the system model, but there was no estimator—the full state was assumed to be accessible. In this final section we examine the effect of using an estimator. As indicated during class, the combination of (1) control law and (2) estimator yields (3) the *controller*. The reference input is introduced as part of the control law.

5.1 Adding the Estimator

There are really only two issues regarding the addition of the estimator with a reference input:

1. In both control laws of (10) and (18) the actual state $\mathbf{x}(k)$ must be replaced by the estimated state $\hat{\mathbf{x}}(k)$...
2. In the estimator state equation—not shown here but equation (7.36) in the notes—one must substitute for $u(k)$ generated by the control law of (10) or (18)—with the estimated state as shown above.

With these items in mind one can then write a state equation and an output equation for the *controller*—this is what will be needed in your `Simulink` block diagrams and in the laboratory evaluation.

5.2 Controller Modeling

The controller will have a *state equation* and an *output equation*—just like any state-variable system. Let's use the following choices:

- Controller state: state estimate $\hat{\mathbf{x}}(k)$
- Controller inputs: there are really *two* classes of inputs:
 - measurement $y(k)$ (could be a vector, but here a scalar)
 - reference input $\mathbf{r}(k)$ (a scalar for SISO, but a vector for MIMO)
- Controller output: plant input $u(k)$ (here a scalar, but could be vector)—sometimes called the *control force*

With these selections, the *structure* of the controller equations will be

$$\hat{\mathbf{x}}(k+1) = [\mathbf{A}] \hat{\mathbf{x}}(k) + [\mathbf{B}] \begin{bmatrix} y(k) \\ \mathbf{r}(k) \end{bmatrix} \quad (21)$$

$$u(k) = [\mathbf{C}] \hat{\mathbf{x}}(k) + [\mathbf{D}] \begin{bmatrix} y(k) \\ \mathbf{r}(k) \end{bmatrix} \quad (22)$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ are functions of Φ, Γ, \dots etc. This is also described in Section 3.5.1 of the “Project” assignment. Note that the “arrangement” of $y(k)$ and $\mathbf{r}(k)$ in (21) and (22) are important. Furthermore, for the MIMO input $\mathbf{r}(k)$ the position is the first element and velocity is second, *e.g.*

$$\mathbf{r}(k) = \begin{bmatrix} r(k) \text{ (desired load position in deg)} \\ \dot{r}(k) \text{ (desired load velocity in deg/s)} \end{bmatrix} \quad (23)$$

This arrangement is consistent with the `Simulink` block diagrams shown in the project handout.

It is the $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ matrices that we need to run your programs. MATLAB M-files will be available on my website that are “templates” for your files. They are exactly the same as given in the project handout.

6 System Model

I wrote the preceding sections of this document in April 2007. Today (April 30, 2008) I had a question after class about modeling the complete controlled system; in particular some confusion about two ways of modeling/simulating the complete system.

So I’ve added Section 6 on this topic. Basically, the two ways one can model and simulate sampled-data LTI systems like ours are:

- Write the state and output equations for the complete system, and simulate in MATLAB:
 - Unified discrete-time model of plant, control law, estimator, reference input
 - Unable to easily add measurement noise
 - Unable to easily add measurement quantization
- Write the state and output equations for the controller only, and simulate in Simulink:
 - Separate modeling of plant, controller (control law + estimator)
 - Not as computationally efficient
 - Can access internal signals for adding measurement/quantization noise
 - Can easily add nonlinear effects (*e.g.* saturation)
 - This is what Simulink was really designed for

NOTE: the estimator will now be included.

6.1 State and Output Equations for Complete System—Simulate in MATLAB

Without going into detail, the following three equations describe the plant, estimator, and control law:

$$\mathbf{x}(k+1) = \Phi \mathbf{x}(k) + \Gamma \mathbf{u}(k) \quad (24)$$

$$\hat{\mathbf{x}}(k+1) = \Phi \hat{\mathbf{x}}(k) + \Gamma \mathbf{u}(k) + \mathbf{L} [\mathbf{y}_m(k) - \mathbf{C}_m \hat{\mathbf{x}}(k)] \quad (25)$$

$$\mathbf{u}(k) = -\mathbf{K} [\hat{\mathbf{x}}(k) - \mathbf{N}_x \mathbf{r}(k)] + \mathbf{N}_u \mathbf{r}(k) \quad (26)$$

Using combined state $[\mathbf{x}^T \quad \hat{\mathbf{x}}^T]^T$ for the full system, and substituting for $\mathbf{u}(k)$ from (26) in (24)–(25), and $\mathbf{y}_m = \mathbf{C}_m \mathbf{x}$ in (25), we get system state equation

$$\begin{bmatrix} \mathbf{x}(k+1) \\ \hat{\mathbf{x}}(k+1) \end{bmatrix} = \begin{bmatrix} \Phi & -\Gamma \mathbf{K} \\ \mathbf{L} \mathbf{C}_m & \Phi - \Gamma \mathbf{K} - \mathbf{L} \mathbf{C}_m \end{bmatrix} \begin{bmatrix} \mathbf{x}(k) \\ \hat{\mathbf{x}}(k) \end{bmatrix} + \begin{bmatrix} \Gamma (\mathbf{K} \mathbf{N}_x + \mathbf{N}_u) \\ \Gamma (\mathbf{K} \mathbf{N}_x + \mathbf{N}_u) \end{bmatrix} \mathbf{r}(k) \quad (27)$$

and (keeping the same output as for the plant model) output equation

$$\mathbf{y}(k) = [\mathbf{C} \quad \mathbf{0}] \begin{bmatrix} \mathbf{x}(k) \\ \hat{\mathbf{x}}(k) \end{bmatrix} + \mathbf{0} \mathbf{r}(k) \quad (28)$$

From (27)–(28) one can extract **ABCD** matrices for the system model, form a MATLAB `ss` system, and simulate. I will demo this in class using the control law and estimator from the homework, and a cubic polynomial input for $\mathbf{r}(k)$.

6.2 State and Output Equations for Controller, Simulate in Simulink

In this case we are separating the plant and the controller; there will be separate Simulink models for the plant (or the actual plant, like in the lab) and the controller.

6.2.1 Plant Model

The plant model can be any of the following:

1. Continuous transfer function plant $G(s)$
2. Discrete transfer function plant $G(z)$
3. Continuous state space plant $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$
4. Discrete state space plant $\Phi, \Gamma, \mathbf{C}, \mathbf{D}$

Choices 2 and 4 are preferable, of course, if the plant has multiple inputs and/or outputs, but if the plant is SISO then choices 1 or 3 are fine. If you model the plant as a continuous system (like I'm going to do here) then you have to put a sample (MATLAB ZOH) block in the loop in a couple of places. If you model the plant as a discrete system it's inherently sampled.

NOTE: In the SS project handout I gave you Simulink examples with the plant modeled as the same discrete transfer function $G(z)$ that we used in the transform-based project. That is what you should use for the SS project.

Plant Transfer Function. I'll use a continuous transfer for the plant model—the output of interest will be load position θ_l in degrees. One can use MATLAB `ss2tf` to find the motor transfer function from the **ABCD** matrices, although one must use the **C** matrix that produces the desired output. Anyway, for this voltage-controlled DC motor plus load you get

$$\frac{\theta_l(s)}{E_a(s)} = \frac{1.1443 \times 10^7}{s(s+107)(s+2112)} \frac{\text{degrees}}{\text{V}} \quad (29)$$

The pole at -107 is from the mechanical time constant J/B while the pole at -2112 is from the electrical time constant L/R . The mechanical stuff is always slower.

6.2.2 Controller Model

The controller model is simply the state-variable estimator update equation of (25),

$$\hat{\mathbf{x}}(k+1) = \Phi \hat{\mathbf{x}}(k) + \Gamma u(k) + \mathbf{L}[y_m(k) - \mathbf{C}_m \hat{\mathbf{x}}(k)] \quad (30)$$

and the control law of (26),

$$\mathbf{u}(k) = -\mathbf{K} [\hat{\mathbf{x}}(k) - \mathbf{N}_x \mathbf{r}(k)] + \mathbf{N}_u \mathbf{r}(k) \quad (31)$$

These equations must be put into the form of (21) and (22) in Section 5.2. The resulting discrete state-space block will be the controller in the Simulink model.

I will have such a Simulink model constructed and I will demonstrate it in class—the response should be the same as the MATLAB model of Section 6.1.