

# Digital Control Semester Project

## Part II: State-Space Designs

### 1 Introduction

In Part II you will be designing two state-space controllers, (1) a single-input single-output (SISO) controller, and (2) a multiple-input multiple-output (MIMO) controller for the same  $G(z)$  plant as with the transform-based controllers (note that you may want to identify a new model for the plant as discussed in Section 2 below). The steps in executing this design follow Chapter 7 in the notes quite closely, and you will have to consult the notes as needed.

*NOTE:* Some of the MATLAB functions require LTI models, while some require [num,den] polynomials, and some require the [A,B,C,D] matrices. I use “A,B,C,D” to mean: “system, input, output, and feedforward” matrices, respectively (continuous or discrete).

### 2 Plant ID

We will be using exactly the same system for the State-Space project, so the “plant” will be the same. However, it’s my intent to adjust the “bias” in the servo amplifier to try to eliminate some of the drift that we saw in the experimental ID data. I have not done that yet. When I do, I will put up a new dataset (probably just using the “binary” input) on the class website, and you should identify a new **second-order** model (a second-order model is adequate to represent the plant dynamics).

However, in the meantime, you can begin the SS design using the model that I used for the TF project:

$$G(z) = \frac{4.0925z + 3.0697}{z^2 - 1.9578z + 0.9576} \frac{\text{counts}}{\text{V}}$$

### 3 SISO Controller Design

#### 3.1 State-Space Model Determination

Given the  $G(z)$  for the plant; you should create a discrete-time state-space model using MATLAB `tf2ss`. Note that in creating the state-space model MATLAB will choose its own canonical state vector  $\mathbf{x}(k)$ , which will *not* be position and velocity. The input will still be  $u(k)$  in V, and the output  $y(k)$  will still be angular position in encoder counts. This SISO model is represented by

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

$$y(k) = \mathbf{C}\mathbf{x}(k) + Du(k) \quad (2)$$

#### 3.2 Control Law Design

Select desired closed-loop pole locations (hence the term “pole placement”) and find the row vector  $\mathbf{K}$  of state feedback gains. This is covered in Section 7.4 in the notes. You may wish to check the plant for controllability. The control law

$$u(k) = -\mathbf{K}\mathbf{x}(k) \quad (3)$$

may be substituted in the plant model of (1) to yield the closed-loop system. You may wish to simulate the closed-loop system from a given initial condition (there is no input yet). I suggest that you be realistic in your choice of desired natural frequency (reflect on your experience with the transform-based controllers; most of you were very conservative).

### 3.3 Prediction Estimator Design

Since you cannot measure *either* state variable (due to the canonical form the state variables in (1) don't have a physical meaning) you cannot really implement the control law. Therefore you must construct a state estimator; use a *prediction estimator* for this project, which is discussed in Section 7.5 of the notes. Select estimator poles to be at most twice as fast as your “control” poles (faster estimator might have noise issues) and find the column vector  $\mathbf{L}$  of estimator gains. Note that the estimator *input* is the plant *output*  $y(k)$  (the encoder position). In this case the measurement and the output of interest are the same ( $y_m = y$ ). The estimator plus control law (together those make up the “controller”) generates control  $u(k)$  which is sent to the amplifier. Note that the controller input-output relationship is exactly opposite to the plant. The prediction estimator equation is

$$\hat{\mathbf{x}}(k+1) = \Phi\hat{\mathbf{x}}(k) + \Gamma u(k) + \mathbf{L}[y(k) - \mathbf{C}\hat{\mathbf{x}}(k)] \quad (4)$$

where  $\hat{\mathbf{x}}$  is the state vector estimate. Again, the estimator combined with the state feedback law (using estimated state  $\hat{\mathbf{x}}$ ) is the controller (Section 7.6), but there is still no input; that case is commonly termed a “regulator.”

### 3.4 Reference Input

Add the reference input  $r(k)$  using the methods of Section 7.8 in the notes. In this SISO design the input will be the cubic position trajectory used previously, and will be exactly 126 samples (5.00 seconds) in duration.

We use the input  $r(k)$  to provide a “reference state”  $\mathbf{x}_r(k)$  *via* matrix  $\mathbf{N}_x$ , and a “feedforward” term through scalar  $N_u$ . The approach of Section 7.8 is to select  $\mathbf{N}_x$  and  $N_u$  to provide the desired steady-state behavior. If the plant is Type 1 or higher,  $N_u = 0$ ; if your model doesn't have a pole exactly at 1.00 then  $N_u$  will not be exactly zero, but should be quite small.

The discussion of  $\mathbf{C}$  (plant output matrix) *vs*  $\mathbf{C}_r$  (reference output matrix) is applicable to MIMO systems—for this design you should assume  $\mathbf{C}_r = \mathbf{C}$ . The reference input equations are

$$u(k) = -\mathbf{K}[\hat{\mathbf{x}}(k) - \mathbf{x}_r(k)] + N_u r(k) \quad (5)$$

$$\mathbf{x}_r(k) = \mathbf{N}_x r(k) \quad (6)$$

### 3.5 System Integration

The complete system is diagrammed in Figure 7.13 in the notes. However, our Simulink model (and lab setup) cannot quite model it that way. We must consider the control law, estimator, and reference input as a single system—a *controller* that accepts plant output measurement  $y(k)$  and reference input  $r(k)$  and produces output  $u(k)$  which is sent to the plant. Hence this “block” has two inputs and one output.

#### 3.5.1 Controller

For the purposes of both simulation and experimentation, you should combine these three functions (control law, estimator, reference input) into a single controller state and output equation, of the form

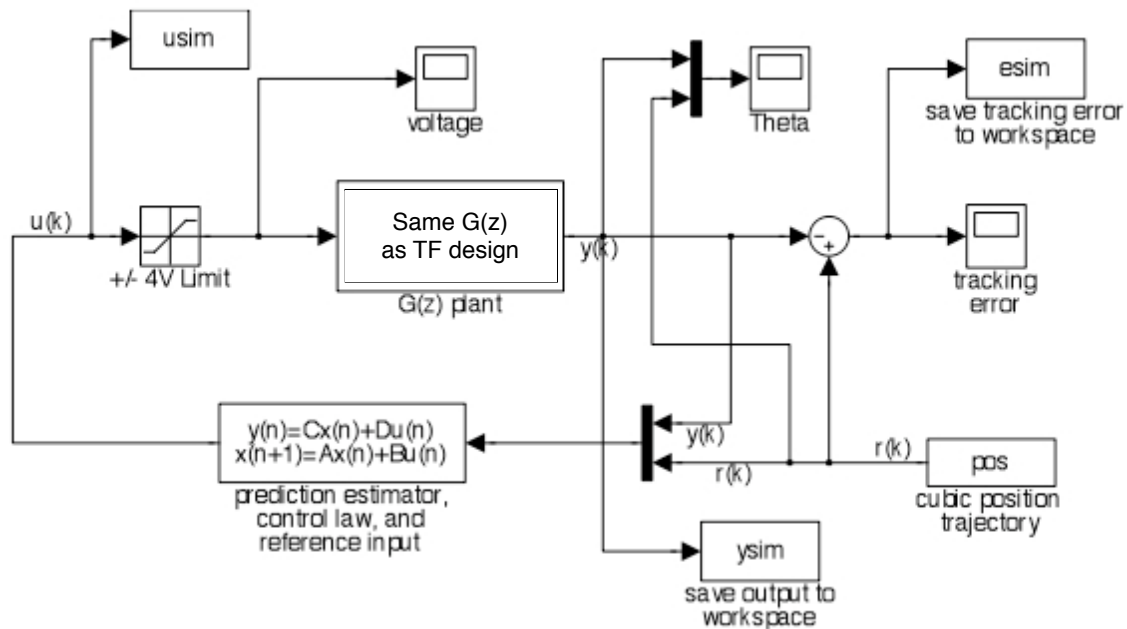
$$\hat{\mathbf{x}}(k+1) = \mathbf{A}\hat{\mathbf{x}}(k) + \mathbf{B} \begin{bmatrix} y(k) \\ r(k) \end{bmatrix} \quad (\text{controller state equation}) \quad (7)$$

$$u(k) = \mathbf{C}\hat{\mathbf{x}}(k) + \mathbf{D} \begin{bmatrix} y(k) \\ r(k) \end{bmatrix} \quad (\text{controller output equation}) \quad (8)$$

Controller matrices  $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$  will be functions of  $\Phi, \Gamma, \mathbf{K}, \mathbf{L}, \mathbf{C}, \mathbf{N}_x, N_u$ . Don't confuse the *plant output*  $\mathbf{C}$  matrix of (2) with the *controller*  $\mathbf{C}$  matrix of (8). Again, the controller has **two inputs**:  $y(k)$  and  $r(k)$  and one output:  $u(k)$ .

### 3.6 Simulink Model

The Simulink model I used for system simulation is shown below.



### 3.7 SISO System Evaluation

While the state-space method of system modeling and controller design is more powerful than the transform-based method (and may seem more elegant), you may be somewhat surprised at the performance of the SISO controller. As before, you should use the “rms error” metric to quantify the tracking performance of your controller.

### 3.8 Submission of Controller Parameters

Email me a textfile (as an attachment, please, not as “inline” text) containing controller matrices **A**, **B**, **C**, **D** from equations (7)–(8). This file should have a name consisting of your three initials prepended to “siso” with filetype .m (for example, `gpssiso.m`); the file should contain

```
% SISO rms error
disp('gps SISO rms error = XX.X counts');

% SISO controller matrices
A = [1 3; 5 7];
B = [1 2; 4 5];
C = [8 7 ];
D = [1 2];
```

Just in case there’s any problem, bring your numerical values to the lab.

## 4 MIMO Controller Design

The design of the MIMO controller is very similar to the SISO design. The main difference lies in the introduction of the reference input. In addition to the *position* input, we will add a *velocity* input; hence the input is now a  $2 \times 1$  vector  $\mathbf{r}(k)$ . The manner in which we introduced the reference input requires there to be a “reference output” of the same form; this is easy if the state vector is position and velocity. Hence a different state-space model of the plant is necessary.

### 4.1 Plant Model

The reference input will now consist of both desired plant position  $r(k)$  and velocity  $\dot{r}(k)$ . We must have the plant *reference output* be the same, that is,

$$\mathbf{y}_r(k) = [y(k) \quad \dot{y}(k)]^T$$

Thus a different plant model is necessary—one from which we can extract both position and velocity as outputs. The simplest way to do this is to find a plant model whose state vector *is* position and velocity.

#### 4.1.1 Revised Transfer Function

Start with the identified discrete transfer function  $G(z)$  (the numbers below are contrived but representative):

$$G(z) = \frac{Y(z)}{U(z)} = \frac{4z + 3}{z^2 - 1.5z + 0.5} \quad (9)$$

Convert this  $G(z)$  to an equivalent  $G(s)$  using the MATLAB `d2c` function. You will get something like

$$G(s) = \frac{3.426s + 6065}{s^2 + 17.33s + 1.23e-13} = \frac{3.426(s + 1770)}{s^2 + 17.33s} \quad (10)$$

Examine the numerator of  $G(s)$  and note that the zero is *much, much* farther to the left than the poles. Therefore, it can be neglected with very little error, and the  $G(s)$  can be written as

$$G(s) = \frac{3.426s + 6065}{s^2 + 17.33s} \approx \frac{6065}{s^2 + 17.33s} = \frac{Y(s)}{U(s)} \quad (11)$$

The differential equation which corresponds to the transfer function of (11) is

$$\ddot{y} + 17.33\dot{y} = 6065u \quad (12)$$

#### 4.1.2 Revised State-Space Model

Now pick the desired state vector:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y \\ \dot{y} \end{bmatrix} \quad (13)$$

and write the corresponding state equations using (12); you will get the continuous plant  $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$  matrices. This continuous state-space model has the state vector of  $[y \quad \dot{y}]^T$  that we need (position and velocity).

Finally, this continuous state-space model can be discretized using MATLAB `c2d` to yield  $\Phi, \Gamma, \mathbf{C}, \mathbf{D}$  matrices.

## 4.2 Control Law and Prediction Estimator

The design of the control law and prediction state estimator are exactly the same as for the SISO system. I suggest using the same pole locations for both control law and estimator as for the SISO system.

### 4.3 Reference Input

As stated before, for the MIMO system the reference input is now

$$\mathbf{r}(k) = \begin{bmatrix} r(k) \\ \dot{r}(k) \end{bmatrix} \quad (14)$$

where velocity input  $\dot{r}(k)$  must be computed using the derivative of the cubic polynomials (velocity will be a quadratic function of time). Perform this derivative analytically (*i.e.* differentiate the cubic polynomial which describes position *vs* time), not numerically!

#### 4.3.1 Output Measurement and “Reference Output”

The output (measurement) of our system is sensed position  $y(k)$  in encoder counts. However, we want the steady-state output to correspond to the input, so if we have two inputs (position and velocity) we must two corresponding outputs (position and velocity). Thus we define a “reference output” matrix  $\mathbf{C}_r$  which produces these two outputs. You should *read Section 7.8 very carefully!!*

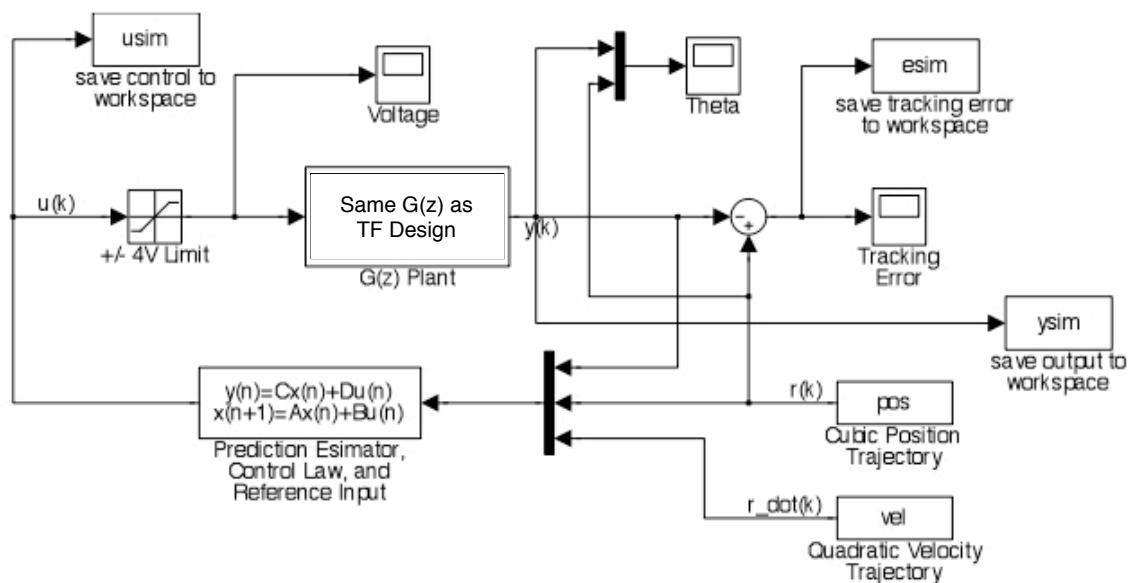
Otherwise the procedure is similar to the SISO controller. However, there is one critical issue. You will again be solving for the  $\mathbf{N}_x$  and  $\mathbf{N}_u$  matrices which produce the reference state vector and feedforward term. Again, you will probably find  $\mathbf{N}_u$  to be very nearly (or exactly) zero. Anyway, in solving for these matrices you will be using equation (7.71), which involves a matrix inverse. You may find this matrix to be non-square, therefore the “pseudoinverse” must be used—this is the topic of Section 7.9. Again, *read this section very carefully!!*

### 4.4 System Integration

You should assemble the control law, estimator, and reference input into a single state equation in the same manner as in Section 3.5 for the SISO controller. Controller matrices  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{C}$ ,  $\mathbf{D}$  will again be functions of the other matrices, and will be what is needed for controller evaluation.

### 4.5 Simulink Model

Below is the Simulink model used for simulation; also for experimental evaluation.



## 4.6 MIMO System Evaluation

You should find that the MIMO system performs *much* better than the SISO system. As usual, “rms error” is the measure.

## 4.7 Submission of Controller Parameters

As before, email me a textfile containing controller matrices **A**, **B**, **C**, **D** from equations (7)-(8). This file should have a name consisting of your initials prepended to “mimo” with filetype `.m` (for example, `gpsmimo.m`); the file should contain

```
% MIMO rms error
disp('gps MIMO rms error = X.XX counts');

% MIMO controller matrices
A = [1 3; 5 7];
B = [1 2 3; 4 5 6];
C = [8 7];
D = [1 2 3];
```

Just in case there's any problem, bring your numerical values to the lab.

## End of Semester Schedule

- Controller parameters due *via* email by 5:00 p.m. Tuesday, May 5
- Experimental evaluation in MTTC room 156 at 2:00 p.m., Wednesday, May 6
- Semester project reports due Friday, May 8, at 2:00 p.m. in class