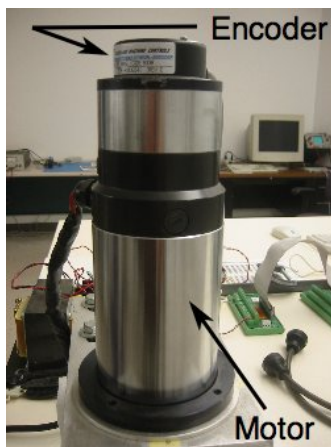


Digital Control Semester Project

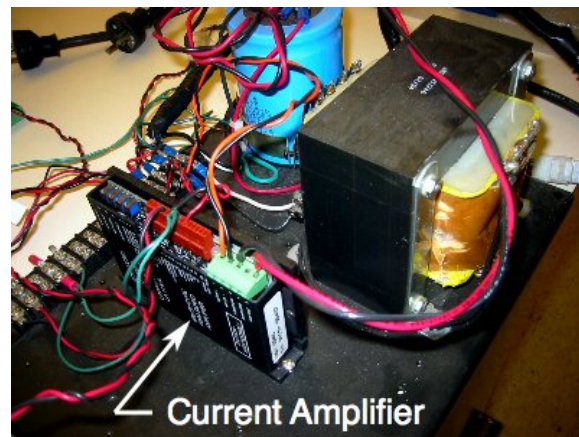
Part I: Transform-Based Design

1 Introduction

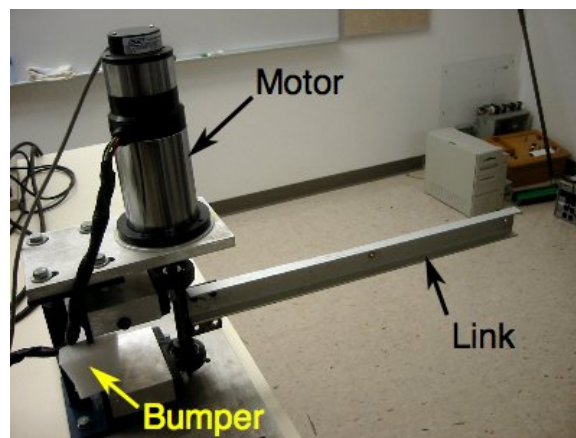
For this project you will be designing a digital controller for a system which consists of a DC motor driving a shaft with a rigid beam mounted to it. A pulse-width modulation (PWM) current amplifier is used to drive the motor, and the angular position of the motor is measured by an incremental encoder. This is representative of an angular position control system common in industrial applications. Figure 1 shows a photograph of the system with the major components.



(a) Motor and encoder.



(b) Current amplifier and power supply.



(c) Link, motor, and bumpers.

Figure 1: Single-link system and components.

1.1 Motor

The permanent-magnet DC motor shown in Figure 1(a) can generate 155 oz-in stall torque, and has some internal viscous friction and brush friction. The brush friction is Coulomb friction, and as such is nonlinear. The motor has a maximum current rating of 9A continuous and 12A peak.

1.2 PWM Amplifier

This amplifier, shown in Figure 1(b), accepts a voltage input and generates a current output in pulse-width-modulated form. The PWM waveform is more effective than a DC waveform for slow-speed control, and also more efficient in terms of power dissipation. PWM drive is very common in modern motor control. The amplifier accepts inputs in the range $\pm 5V$. The amplifier has a nominal gain of 2.5 A/V. Your controllers should not exceed an amplifier command of 4V (motor current 10A).

1.3 Beam

The beam is an aluminum channel section 22 inches in length, and is assumed to be rigid. The beam is shown in Figure 1(c) along with the styrofoam “bumpers” to prevent the beam from colliding with the support bracket.

1.4 Encoder

The incremental encoder, also shown in Figure 1(a), provides a measure of motor shaft (and therefore link) angular position. Using “quadrature” decoding, it has an effective resolution of 4000 counts per revolution (almost exactly equal to a 12-bit quantization of one revolution). All of our analysis and design will use angular position measurement in encoder counts (not radians or degrees). It is simple to convert to degrees or radians if that were needed.

1.5 Required Project Tasks

Successful completion of the Semester Project (Part I) will require completion of the following tasks, each of which will be discussed in greater detail later in this document:

1.5.1 Plant Model Identification

Using the techniques of Chapter 8 you will be required to identify a transfer-function model $G(z)$ of the plant.

1.5.2 Reference Trajectory

The control system will be required to move the beam through the following trajectory:

- $0^\circ \rightarrow 90^\circ$ in 1 second
- $90^\circ \rightarrow -90^\circ$ in 2 seconds
- $-90^\circ \rightarrow 0^\circ$ in 1 second
- a “settling” period at 0° of 1 second

Each segment will be a cubic polynomial, resulting in “smooth” motion.

1.5.3 Transform-Based Controller Design

You will design a controller $D(z)$ such that the controlled system has acceptable performance. There is no particular specification you must meet, but we expect good tracking of the input. You may wish to explore the following controller designs (*e.g.* you are encouraged to investigate and submit multiple designs):

- Lead compensator
- Proportional + derivative (PD) control
- PID control

Some controllers have been omitted (P and PI) because they do not yield good performance and are a waste of our valuable time. You may wish to include design and analysis of these two controller to illustrate the poor performance. The lab equipment will be set up to implement the three controllers in the list.

1.5.4 Simulation and Evaluation

You must simulate the performance of all of your systems using the reference trajectory of Section 1.5.2, and predict the *RMS tracking error* in units of *encoder counts*. If everything is done properly, there should be good agreement between the results of your simulation and the experimental evaluation in the laboratory.

2 Plant Model Identification

Figure 2 below shows a block diagram of the lab setup. Variables r , e , and y are in units of encoder counts, while u is in units of V (volts), while θ is in units of radians. The components within the *red* dashed rectangle are inside the

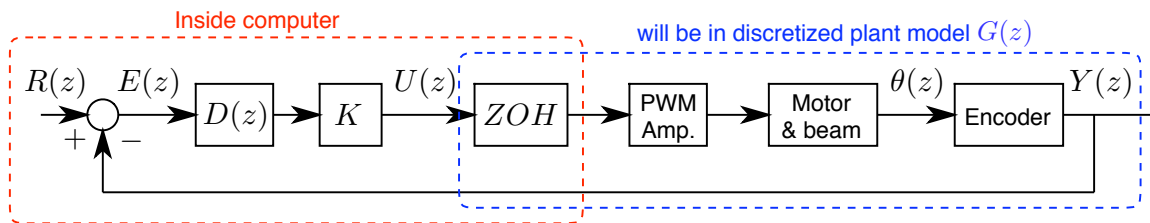


Figure 2: Block diagram of angular positioning system.

computer, while the components within the *blue* dashed rectangle will be included in the $G(z)$ plant model you will be identifying, that is:

$$G(z) = \frac{Y(z)}{U(z)} \frac{\text{counts}}{\text{V}} \quad (1)$$

The sampling frequency will be $f_s = 25$ Hz, thus the sampling period is $T = 0.04$ second.

2.1 Model Order

A system with inertia—from basic principles—will be at least second order. However, additional dynamic effects may cause a higher-order model to be a better fit. Part of the challenge in system identification is to find the proper order model. Generally speaking, the lowest-order model that is sufficiently accurate is best.

The methodology of Chapter 8 requires you to use models of the form

$$G(z) = \frac{b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}}{1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3}} = \frac{b_1 z^2 + b_2 z + b_3}{z^3 - a_1 z^2 - a_2 z - a_3}, \quad (2)$$

where (2) is the model structure if you assumed a third order model.

2.2 Input-Output Data

On my website there will be a dataset in the form of a MATLAB binary file (filename `data.mat`). Once downloaded and placed in your MATLAB working directory, this file can be loaded into the MATLAB workspace by executing the following command: `>> load data;` You should then see six variables in your workspace. In the example below I loaded the data, then typed `who` to show the variables:

```
>> load data
>> who
```

Your variables are:

```
ub un us yb yn ys
```

Vectors `un` and `ub` are the random “normal” and random “binary” inputs (units of V), while vectors `yn` and `yb` are the corresponding outputs (units of encoder counts). These inputs were scaled and biased to use as much range as possible of the quantization (The D/A converter is 12-bit). Finally, vectors `us` and `ys` are a step input and the response to that step input. The step input is 1.5V in magnitude (*not* a unit step, but a 1.5V step). Plots of the input and output data are shown in Figure 3. *NOTE: The data shown are an example, and are not the actual data you will receive!*

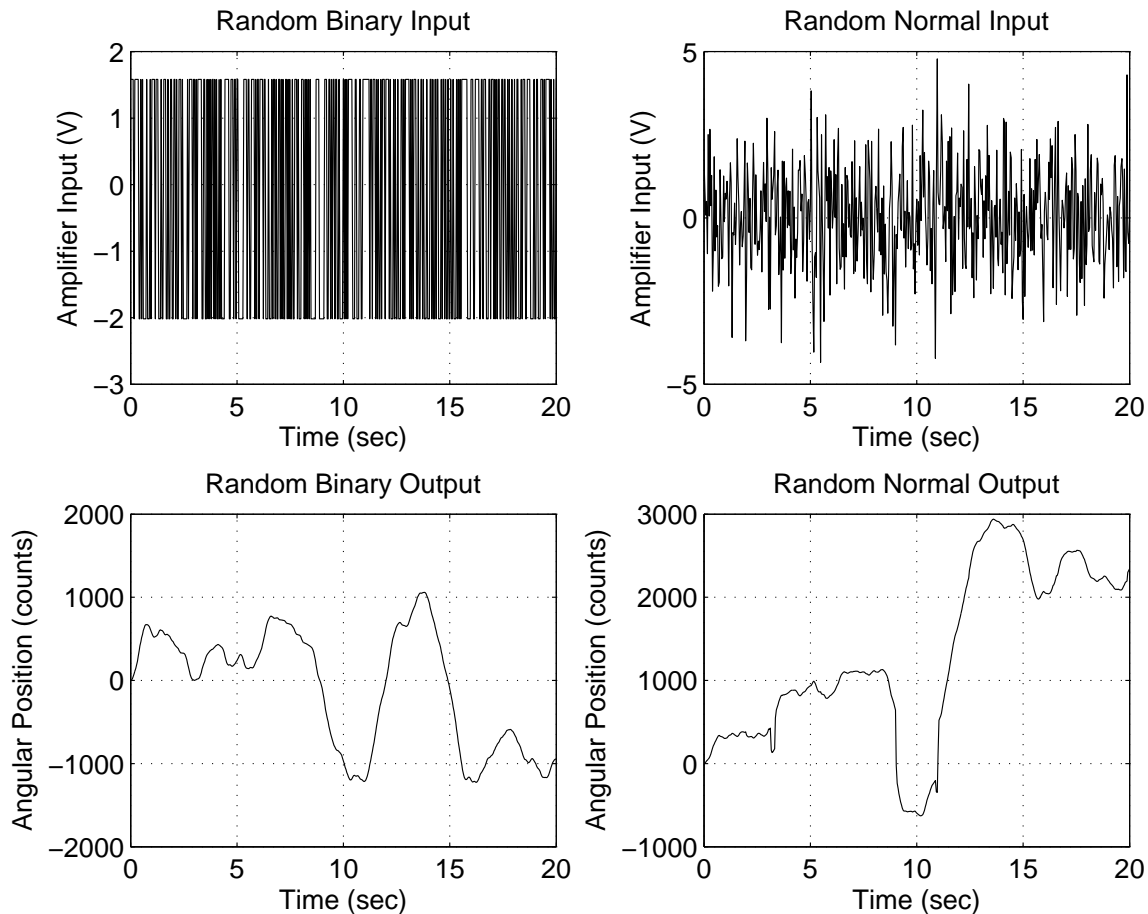


Figure 3: Input-output data for identification.

There will be 500 samples in each ID dataset, thus $k = 0 \dots 499$. You should be able to fit an adequate model with a subset of these samples. You may wish to compare the step response of your model to the experimental step response (remember it’s from a 1.5V step input). You should select *one* model $G(z)$ for subsequent use in controller design.

Since you are commanding a current (torque) input to an inertia J which has some viscous friction B and Coulomb friction (“stiction”—which is nonlinear and very difficult to model) I would expect the transfer function from current to angular *velocity* to have the form $K/(Js+B)$, with an integration added when you go from angular velocity to angular *position*. However, there are almost certainly *unmodeled dynamics*—perhaps the beam has a structural resonance—so the order of your assumed model should be explored thoroughly.

2.3 Evaluating your Identified Models

There is a potential problem with evaluating your ID’ed models. The actual system contains a pure integration (pole at $z = 1$). Your ID’ed models will likely have a pole at $z = 1.0036$, or $z = 0.9996$, or somewhere very close to 1.0000, *but not exactly at 1.000!!*. If you then take some of the random input and use it on your model, you may not get the same “drift” as the actual system. This can cause an extremely large cumulative error, which is *not* indicative of your model performance. Here are a couple ways around this:

- Reformulate the ID process to allow you to *specify* that the resulting model has a pole exactly at $z = 1$. This is actually not that hard, and I encourage you to consider it.
- Differentiate your $G(z)$, apply the random input; then compare the response to the numerically differentiated experimental random output. This effectively removes the integration.

In the above discussion, “random” refers to either the normal or binary random data.

3 Reference Trajectory

The trajectory that the system will be expected to follow is composed of three cubic polynomials—one for each segment. The nature of the trajectory was described in Section 1.5.2. The units of the trajectory will be encoder counts, and a plot of the trajectory will be r (encoder counts) *vs* t (sec).

A cubic polynomial allows you to specify endpoint constraints of both position and velocity, and is of the form

$$r(t) = a_0 + a_1t + a_2t^2 + a_3t^3 \quad (3)$$

It can be shown that for initial position r_i , final position r_f , initial velocity \dot{r}_i , final velocity \dot{r}_f , and duration t_f , the cubic coefficients are

$$\begin{aligned} a_0 &= r_i \\ a_1 &= 0 \\ a_2 &= \frac{3}{t_f^2}(r_f - r_i) \\ a_3 &= -\frac{2}{t_f^3}(r_f - r_i) \end{aligned}$$

You will be fitting three cubics of the form of (3) together to make up the complete trajectory. A plot of the resulting position and velocity is shown in Figure 4(a). *NOTE:* at the junction of two trajectories you will have one “duplicate” sample which should be deleted. The final reference trajectory should contain exactly 126 points (5.00 seconds in duration; from $t = 0.00$ to $t = 5.00$). The final one second is a “runoff” area for overshoot and settling.

If you are unable to create this reference input, I will give it to you (with a corresponding deduction—not severe).

4 Transform-Based Controller Design

You should design as many controllers as you wish (see the list on page 2) but we will only be using those three $D(z)$ which give good performance. You should document *all* of your designs, including those which don’t work too well (and your speculation on why they didn’t work). Design your $D(z)$ using MATLAB and the root locus for pole placement.

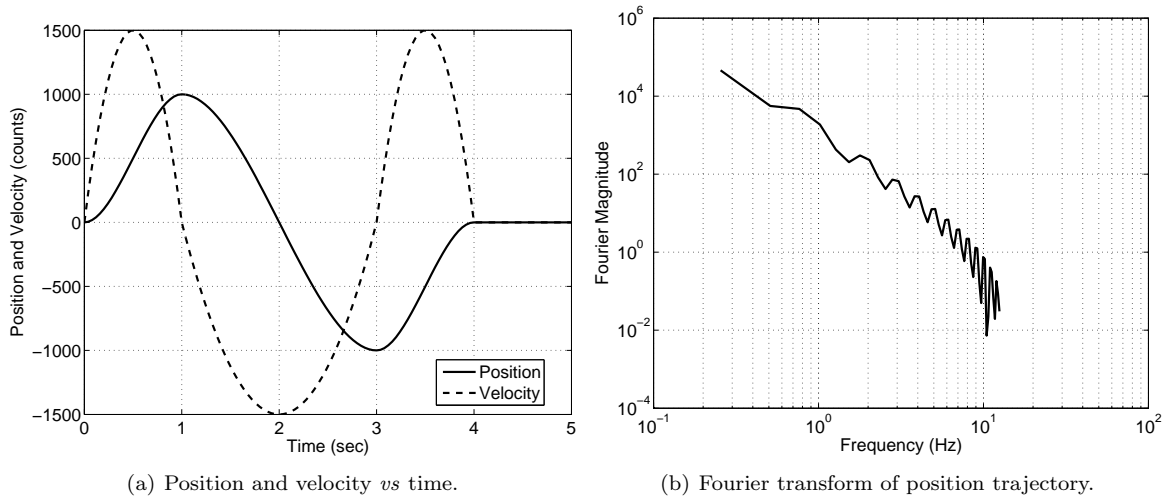


Figure 4: Reference trajectory plots.

4.1 Some Guidelines

So far your only design guideline is to achieve “good” tracking performance while keeping the voltage command to the amplifier $< \pm 4V$. To give you a little more insight, consider the following observations on damping ratio ζ and natural frequency ω_n .

Selecting a desired damping ratio ζ should be simple—you want a reasonable amount of damping. However, selecting a desired natural frequency ω_n is more difficult—since you don’t want ω_n higher than necessary—but it must be high enough for adequate performance. Excessive ω_n is “expensive”—it requires larger actuators, higher currents, *etc.*

One strategy for selecting ω_n is to examine the frequency content of the input; this may be done by finding the Fourier Transform of the cubic polynomial position trajectory of Figure 4(a). The MATLAB `fft` function (Fast Fourier Transform) is ideally suited for this operation.

The resulting plot of Fourier magnitude *vs* frequency is shown in Figure 4(b). You should probably not try to place the natural frequency of your dominant closed-loop poles above *all* the harmonic content, but you should try to respond to *most* of it. Note that the Fourier Magnitudes of Figure 4(b) are plotted on a logarithmic scale. You may wish to use the MATLAB `bode` function to check the frequency responses of your open- and closed-loop transfer functions during the design.

5 Simulation and Evaluation

Simulation of the behavior of your system is part of the design process of your controllers, so Sections 4 and 5 are related. You should simulate your system using the cubic trajectory input. You may use either MATLAB or Simulink. Since you will need to examine the peak level of voltage (to check for saturation), Simulink may be attractive since it’s easy to “pick off” intermediate variables. An example Simulink model for a “lead compensator” $D(z)$ is shown below. In your $G(z)$ plant model there will be numerical coefficients for the a_i and b_i terms (from your ID runs). The terms “From/To Workspace” and “Discrete Transfer Fcn” are the Simulink block names for those elements.

5.1 From and To Workspace Blocks

These are convenient ways to import/export data to/from your workspace with Simulink models.

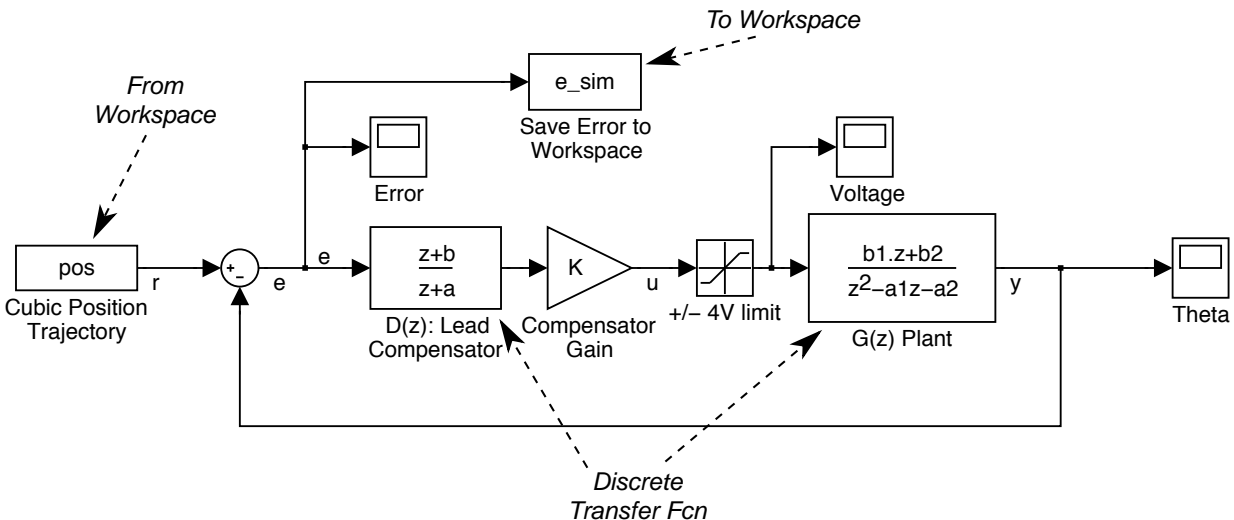


Figure 5: Simulink diagram for simulation.

5.1.1 From Workspace

This block should contain the cubic polynomial reference trajectory. When you open this block there is a dialog box with a “Parameters” section, which contains fields for “Data” and “Sample Time.” The “Data” field should contain the name of a two-column array in your workspace—the two columns will be t (time) and r (reference position), respectively. The size of this array will be 126×2 . The “Sample time” field should be set to 0.04.

5.1.2 To Workspace

This block is used to archive the tracking error to your workspace for **RMS error** determination. This dialog box contains only one field you need to change, that’s the “Variable name” field. Whatever name you enter here will be the name of the error stored subsequently in your workspace. The other thing you should check is the “Save format” menu—it should be specified as “Array.”

5.2 Tracking Performance

The single “figure of merit” for your designs will be the *RMS tracking error* in encoder counts. This is kind of an “average” tracking error. If vector \mathbf{e} contains the time history of tracking error, and there are N samples, then

$$e_{RMS} = \sqrt{\frac{\sum_{k=1}^N (e(k))^2}{N}} = \sqrt{\frac{\mathbf{e}^T \mathbf{e}}{N}} \quad (4)$$

5.3 A Word to the Wise

Don’t try to get too “aggressive” in trying to reduce the RMS error. There are *always* unmodeled dynamics in a system that will appear if you try to push the system too hard. One sign of trouble is having lots of “chatter” in the control force u sent to the plant.

5.4 Submission of Controller Parameters

To simplify the evaluation of all your designs, I want you to email me textfiles for each $D(z)$ you design that contain the parameters for that controller, as well as the predicted RMS error of that controller. The file will be used as a MATLAB script file, so its name should consist of a concatenation of: (1) your initials (lower case, please), (2) the

controller type (PD, lead, PID), and (3) the suffix “.m”—hence my file for the lead network $D(z)$ would be a file called `gpslead.m` containing the following:

```
% lead compensator rms error
disp('rms error = xxx counts');
```

```
% lead compensator coefficients:
a = xx.xx; b= xx.xx; K=xx.xx;
```

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

Templates will be available online for the controller parameter files for all three cases.

6 Semester Project Report

You will eventually prepare a full report on both Part I and Part II (state-space) of the Semester Project. This report will be due on Wednesday May 14, at 12:30 p.m. (this is the date of the scheduled Final Exam). We will evaluate the state-space controllers on Friday, May 9.

6.1 Performance of your Controllers

The controllers will be evaluated down in the MTTC Laboratory on Wednesday, April 8, at 2:00. Therefore you *MUST* submit your controllers parameters by Monday, April 6 (before midnight).

The performance of your controllers will be acquired during the actual runs, and will be emailed back to you. Your semester report should comment on the agreement (or lack thereof) between your simulations and the actual evaluations.

Your reports will be graded on just how much you have done, the thoroughness and validity of your analysis, and your discussion of what you did. I'm actually as much concerned about the words you use to describe what you did (to see if you understood what you did) as the actual quantitative performance of your controllers. I always appreciate reports that are in a clear format which is easy to read.

6.2 A Threat

NOTE: The experimental data from the validation runs is **extremely important!** Your project report *MUST* contain a discussion of your simulation's performance *vs* the actual experimental data, or you will not pass this course.