Issued January 13, 2025
Due March 17, 2025 by 6pm

# Lab 1
# Discrete Equivalent Designs versus Direct Discrete Designs

## Objective

In this lab, you will first design discrete compensators by converting a classically designed continuous compensator into "equivalent" digital compensators. The performance of these will then be compared with digital compensators directly designed in the discrete domain using the root locus technique.

A servomechanism with voltage control of the motor has a linearized model described by the Laplace transform

$$G(s) = \frac{1}{s(s+0.4)}.$$

We want to design a digital controller for this system so that the closed-loop system has a unit step response rise time of less than 0.5 second and no more than 25% overshoot. (The rise time is the time for the response to go from 10% to 90% of its steady-state value.)

Remember that you should use the Lab or Project Report Format outlined in the *Lab and Project Guidelines* handout.

### Part 1

As a first design, compute the necessary bandwidth and phase margin or desired closed-loop pole locations and use a continuous lead network $D(s)$ to obtain a design that meets these specifications. Use the pole-zero mapping method of discrete equivalent design to map this continuous compensator design to the discrete domain. Use sampling rates of 4 and 10 Hz; the pole-zero mapping method should yield a different $D(z)$ design for each sampling rate. Can you predict the rise time and overshoot that would result using these discrete designs? The graph on the final page of this handout may be useful for this (as are the Matlab commands `pzmap` and `zgrid` (and `sgrid`)). How accurate do you think your predictions are?

One way to design the continuous lead compensator is to have the lead zero cancel the pole of the plant, and then choose the lead pole and gain of the compensator to yield the desired 2nd-order closed-loop response. What are the advantages and disadvantages of designing the lead network this way? What happens if there are modeling errors and the cancelation ends up not being exact?

### Part 2

As a second design, use the root locus in the $z$-plane to obtain discrete lead compensators that will meet the specifications at sampling rates of 4 and 10 Hz (one compensator for each sampling rate). You will need to compute the discrete plant transfer function using the zero-order-hold model of the D/A converter.

You will probably need to experiment with varying the lead zero and pole locations to obtain a design that will meet the specifications (`sisotool` may be helpful here). Again, can you predict the rise time and overshoot that would result using these discrete designs? How accurate do you think your predictions are?

**Part 3**

Now use Simulink to implement these compensators. Using Simulink, you can easily build a sampled-data system consisting of a discrete controller $D(z)$ and a continuous-time plant $G(s)$.

To create the Simulink simulation, follow the procedure outlined in the Simulink Primer (posted on course website) in the section on the last page entitled "Summary of Basic Steps". Your system should look like the one in Figure 1 on the next page. You can use either a `Zero-Pole` block or a `Transfer Fcn` block (both in the `Continuous` Library) for the plant. And you can use either the `Discrete Zero-Pole`, the `Discrete Transfer Fcn`, or the `Discrete Filter` block (all in the `Discrete` Library) for the discrete compensator. Use a `Zero-Order Hold` block (in the `Discrete` Library) to convert the discrete output of the controller to a continuous control signal for input into the plant. Make sure to click on the `Zero-Order Hold` block to set the sample time properly.

Use a `Sum` block (in the `Math Operations` Library) for the error summing junction (double-click on the block to change the signs), and a `Step` block (in the `Sources` Library) for the reference input. Tie `XY Graph` or `Scope` blocks (in the `Sinks` Library) to the controller output (plant input) and plant output (servo position). (You probably want to view/save more results than that indicated in the diagram in Figure 1.) Bring the controller output, plant output, and the step command into the MATLAB workspace with `To Workspace` blocks (in `Sinks`).

In this course, we will learn how to design discrete controllers for linear systems, but most real systems have some nonlinear characteristics as well. Examples of common nonlinear characteristics include saturation and friction. To make your simulations more realistic, include the effect of actuator saturation on the closed-loop response. All actuators have limits due to amplifier saturation, valves being opened all the way, etc. Simulink makes it easy to include nonlinear effects in your system model.

For the Simulink simulation, there is a `Saturation` block (in the `Discontinuities` Library (or the `Nonlinear` Library in older versions of Simulink)). This block generates a piecewise linear mapping described by the parameters you enter. Include actuator or D/A saturation (between the blocks for the controller and the plant) in your Simulink model. Let the saturation limits be $\pm 10$. (D/A converters often saturate at 10 volts). The gain is 1 in the linear region of the saturator. (If you want to examine the effect of saturation, you may want to define the saturation values as variables and then change their values in the MATLAB workspace. You can then vary the saturation (or turn it on and off) and compare the results. By using the MATLAB '`hold`' command you can overlay response plots with saturation and without saturation.)

Make the step occur at time = 1.0 sec to make it easier to visualize on the plots. This can be changed by double-clicking on the `Step` block. Don't forget to set the simulation stop time.

Be patient, it will take some time to get your system up and running if this is your first exposure to Simulink. Remember to save your system occasionally.

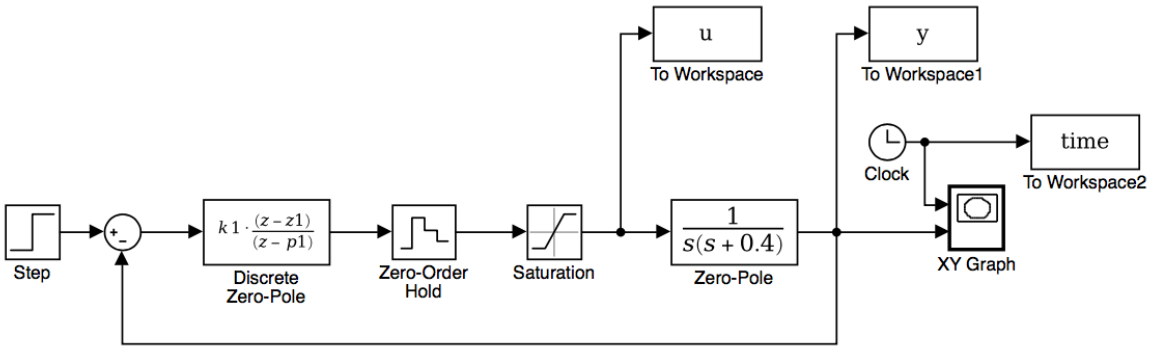



Figure 1: Sample block diagram of control system.

Once you have your control system implemented and running, record the step responses of your designs and compare them with your theoretical predictions of overshoot and rise time. Obtain plots of the **output position**, **step input signal**, and **controller output** (input to plant) (all on the same figure) to include in your lab report. You may want to scale some of these variables for better readability or to make better use of the axes in the plots. Also include your Simulink model. (If you run into errors plotting due to different vector lengths, you can always specify the sample time in each `To Workspace` block to the same value rather than using `-1` for inherited.)

**Part 4**

In your lab report, compare the two approaches to discrete design. Include considerations of sampling rate, ease of design computations, ability to predict the results of the designs, and performance achieved. If your discrete designs from Parts 1 and 2 above did not meet specifications, why not? Can you iterate and find a discrete design that does meet specifications at the sampling frequency of 4 Hz? Can you do the same for the sampling frequency at 10 Hz?

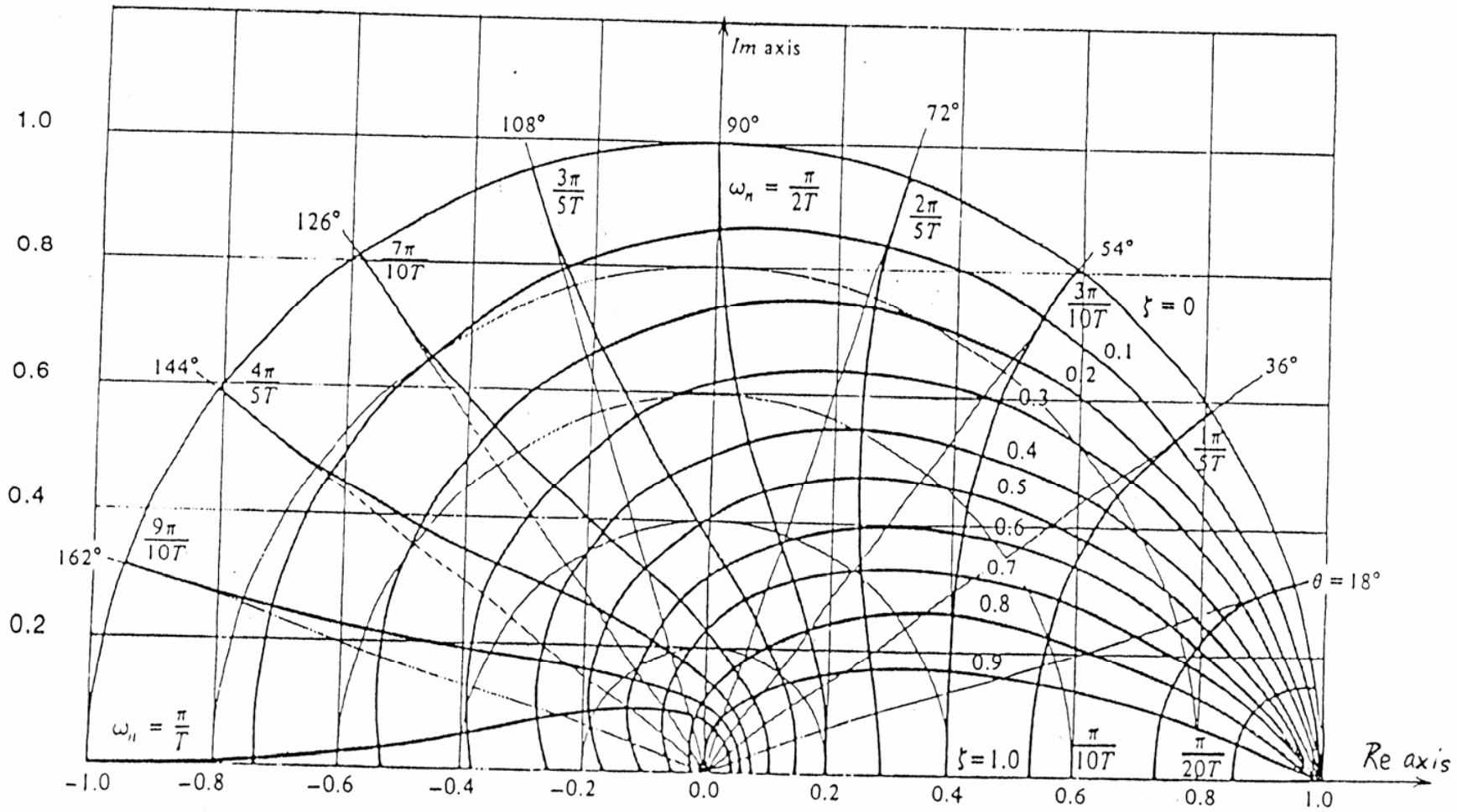
**Part 5** (Optional, worth up to 10 bonus points)

**(a)** (Up to 3 bonus points) Experiment with the sampling period in Part 1. Can you sample fast enough that your "discrete equivalent" design meets the specifications? In order to get a valid simulation, you may have to set the maximum simulation step size in your `Model Configuration Parameters` if your sampling rate is very fast.

**(b)** (Up to 4 bonus points) Determine the lowest sampling rate with which you can meet the specifications. Give the discrete compensator design that you used to achieve the specifications and obtain (overlaid) plots of the step input, controller output, and plant output after implementing this design in Simulink.

**(c)** (Up to 3 bonus points) Could an additional discrete-time feedforward filter or controller (e.g., placed between the step input and the summing junction) lead to better performance? If so, give the discrete feedforward filter design and obtain (overlaid) plots comparing the performance with and without the feedforward filter.

Explain carefully your analysis and process.


Im axis
Re axis
1.0
0.8
0.6
0.4
0.2
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
108°
90°
72°
126°
54°
144°
36°
162°
θ = 18°
3π/5T
2π/5T
7π/10T
3π/10T
4π/5T
1π/5T
9π/10T
ω_n = π/2T
ω_n = π/T
ζ = 0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
ζ = 1.0
π/10T
π/20T