

## EE3304 Digital Control Systems

### EXPERIMENT

#### An Introduction to Computer Aided Digital Controller Design

(Via Matlab and Simulink, R2014a)

## 1 Introduction

Engineering workstations are rapidly becoming an essential part of the engineer's toolbox. Just as first the slide rule and then the calculator served to remove some of the drudgery from design work, design software package running on workstations allows engineers to work much more efficiently by eliminating the tedious and often repetitive time consuming tasks which are part of their routine work. There is now a wide variety of package available for control system design, many of which even run on personal computers.

One package, which has become popular, is Matlab. Matlab is a highly interactive software package. Its use of “M-files” allows one to write programs. Many algorithms used in control design work have been compiled and marketed as toolboxes. One such toolbox is the Control Systems Toolbox.

Simulink is a simulation package, which runs under Matlab. It contains a number of routines to integrate differential equations. These routines allow it to simulate continuous time and nonlinear systems. Simulink also allows one to build systems graphically. The graphical construction allows simulations to be easily constructed and complicated systems could be built easily.

## 2 Simulation with Matlab and Simulink

To run Matlab, double click the Matlab icon with your mouse. The Matlab command window appears with “>>”. You can either open an existing M-file or a new M-file. To run Simulink, click the “Simulink Library” button shown in Figure 1, or type command **simulink**.

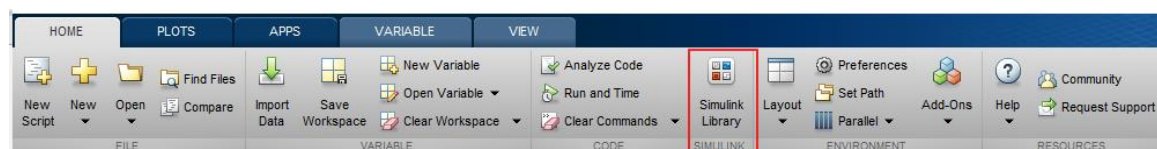


Figure 1 Run Simulink

### 2.1 Revision on Matlab

A list of all the available commands is available by typing **help**. To get help on a particular command, you can type **help** followed by the particular command name. For example, **help rank** gives the documentation for the matrix rank function **rank**.

Matrices are defined in Matlab using square brackets. For example,

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad (1)$$

is defined in Matlab as  $A = [1 \ 2; 3 \ 4]$

In the Control Systems Toolbox, polynomials are defined as vectors of the coefficients in descending order of powers. For example,  $n(s) = s^4 + 2s^3 + 6s^2 + 7s + 1$  is defined in Matlab as  $n = [1 \ 2 \ 6 \ 7 \ 1]$ ,  $d(s) = 2s^3 - 6s^2 + 7s$  is defined as  $n = [2 \ -6 \ 7 \ 0]$  (note that there is a zero at last).

Some commonly used built-in functions for control are **rlocus**, **nyquist** and **bode**. Check out the usage of these commands by using help.

## 2.2 Use of Simulink

To run Simulink, click the “Simulink Library” button shown in Figure 1, or type command **simulink**. Then you can see the Simulink Library Browser (Simulink window) as below.

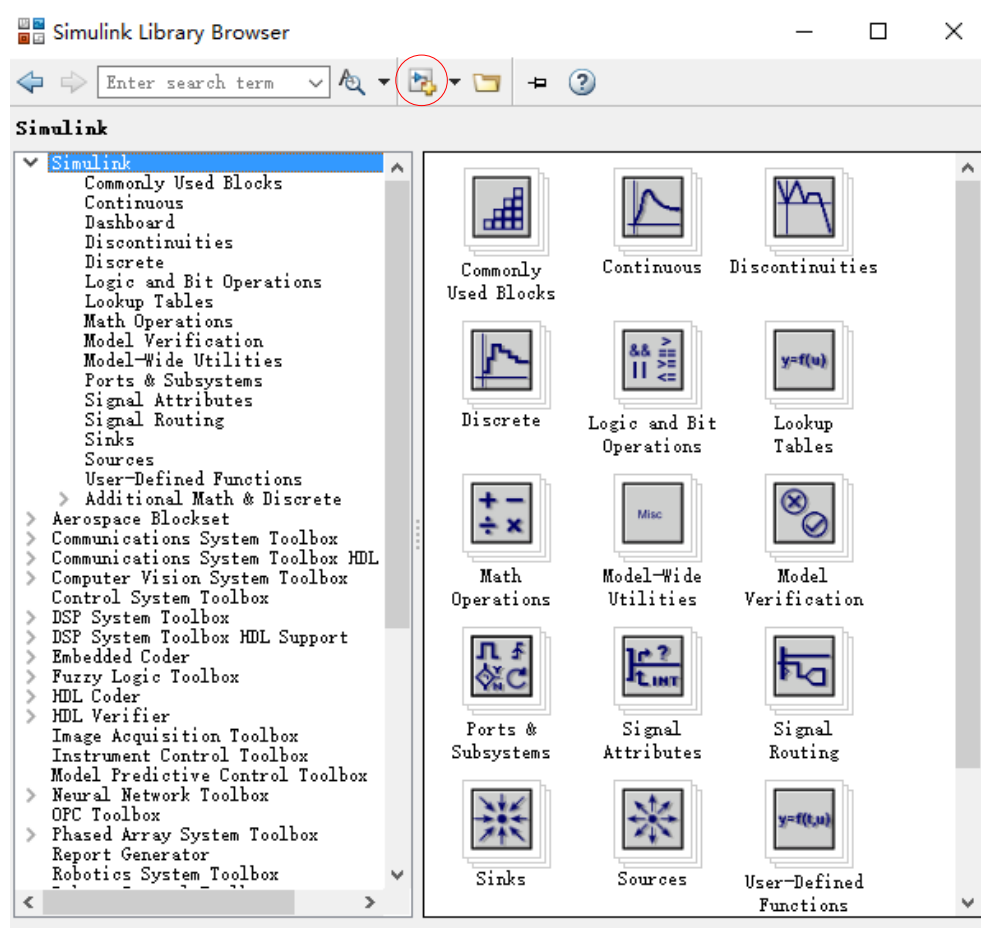



Figure 2 Simulink Library Browser

In this section, we will learn how to use **Simulink** by going through an example.

To create a new simulation file, go to the **Simulink** window. Click the  button shown in Figure 2. Then a new window with the title, **Untitled**, appears.

To add blocks into the new file, just click any of the blocks in the Simulink Library Browser window and drag it into the **Untitled** window. Find a **Transfer Fcn** block from **Continuous** pool and place it in the new file. Double click any block in **Untitled**, then you can change the parameter setting

of the block.

Now let's construct the following TF in Simulink, which is a simplified DC servo dynamics

$$G(s) = \frac{9}{s(s+1)} \quad (2)$$

Go to the Simulink Library Browser window, and choose appropriate items from **Sources** block, **Sinks** block, and **Math Operations** block, to generate the block diagram of the following closed-loop, which includes a "Step" block, a "Scope" block, and a "Sum" block. Click the mouse and drag the mouse between arrows of two blocks will build up the connection, as shown in Figure 3.

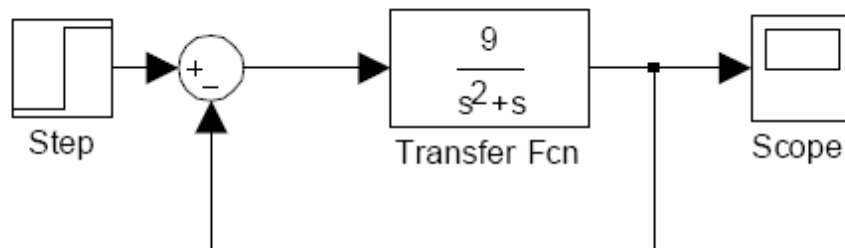


Figure 3 DC in Simulink

Next double click the "Transfer Fcn" block to set the parameters for the specified transfer function in (2). The result is shown in Figure 4. Then set the step time for "Step" block to be 0, which is shown in Figure 5.

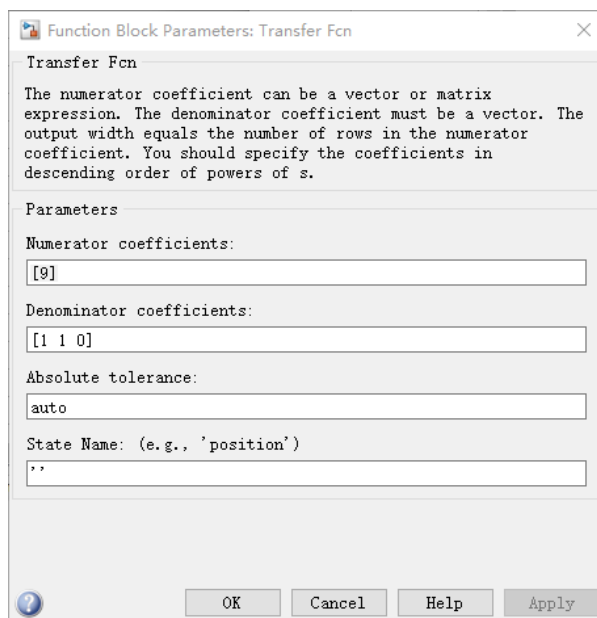


Figure 4 Parameters in "Transfer Fcn" block

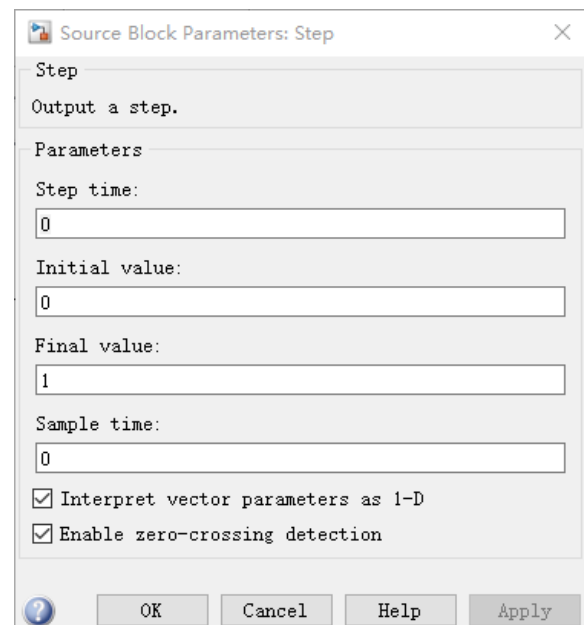


Figure 5 Parameters for "Step" block

Help on any particular block can be obtained by right clicking the block and then selecting **help** from the context menu.

Run a simulation by clicking the Run button. To stop simulation, click **Stop** button. To specify the simulation time, just input the new time. These functions are shown in Figure 6.

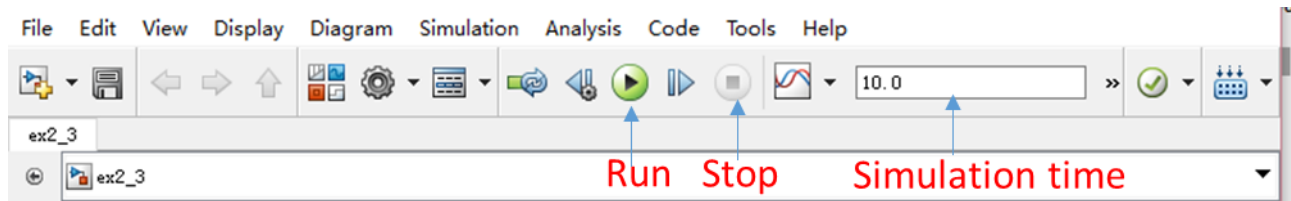


Figure 6 Run or stop a simulation

Get yourself be familiar with Simulink functions, such as **ZOH**, **clock**, **Transfer Fcn**, etc. from **Continuous**, **Discrete**, **Sink**, **Source**, and **Math Operation** pools.

## 2.3 Using Matlab with Simulink

Note that variables defined in Matlab workspace could be used. For example if the variable **num** is defined in Matlab, the numerator parameter in **Transfer Fcn** could be entered as **num**.

Simulation result connected to **Scope** could not be stored. To save the simulation results, you have to pass it back to Matlab by using the **To Workspace** block in the **Sinks** category in the Simulink Library Browser. For example, shows the **To Workspace** block is added in Figure 7.

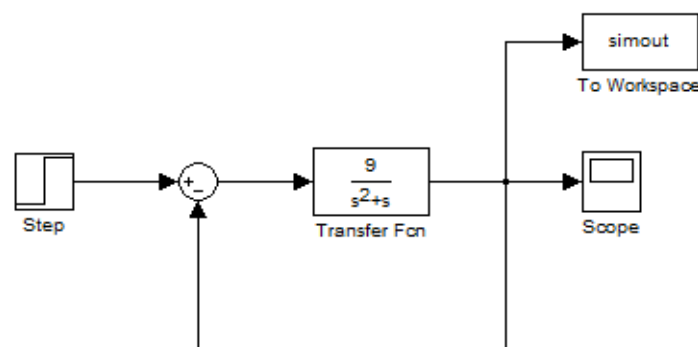


Figure 7 The block “To Workspace” is added

The parameter setting of block **To Workspace** is shown as Figure 8. Here we use the default variable name “simout” for the **To Workspace** block, and you can change it as you like. Make sure the “save format” should be “Array”.

After running this simulation, you can find two variables *simout* and *tout* in the MATALB workspace. Now you can obtain the same plot as shown in **Scope** by typing the command “plot (tout, simout)” in the command window.

Get yourself be familiar with **Matlab** commands, such as **c2d**, **bandwidth**, **plot**, **tf**, etc.

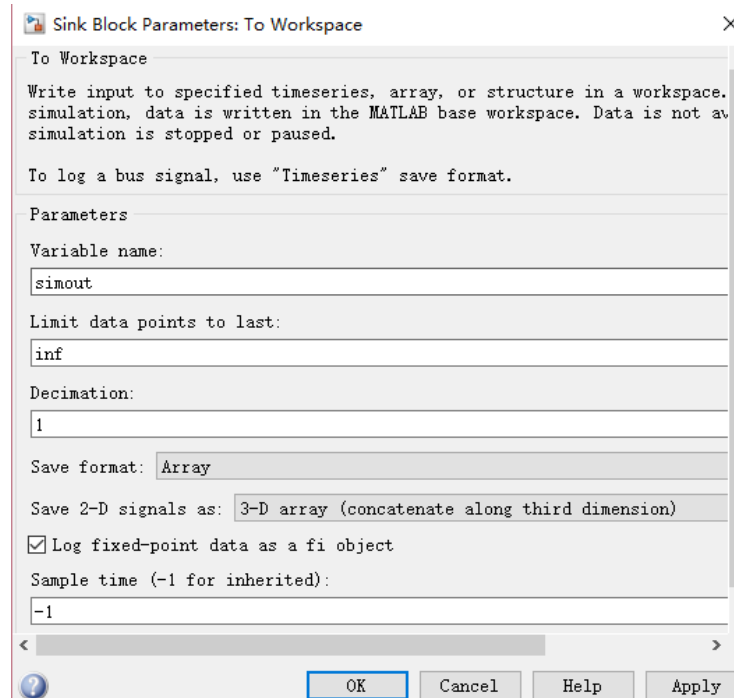


Figure 8 Parameter setting of “To Workspace”

### 3 Design and Analysis of a Continuous-time System

Consider the simplified DC servo dynamics in equation (2).

**Step 1:** Run the simulation when the controller is a unit P ( $P=1$ ) controller, as shown in Figure 9. Comment on the step response.

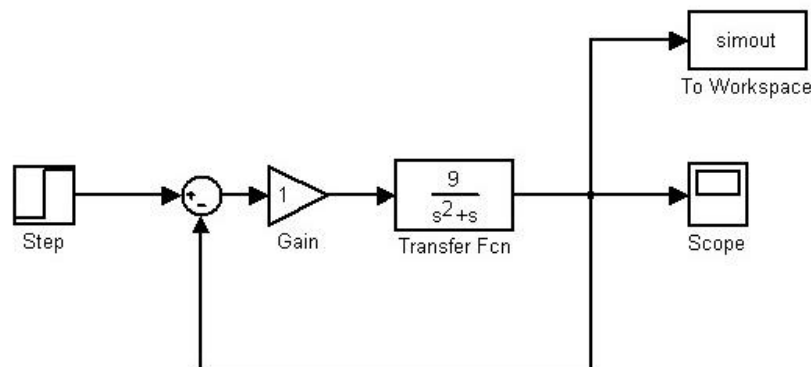


Figure 9 Block Diagram of P control

**Step 2:** Close the loop with a PD control

$$C(s) = l_p + l_d s \quad (3)$$

with unity negative feedback. Choose the P control gain  $l_p=1$  and the differential gain  $l_d=0.1, 0.6, 2$  respectively. The plant is shown in Figure 10. Comment on the responses.

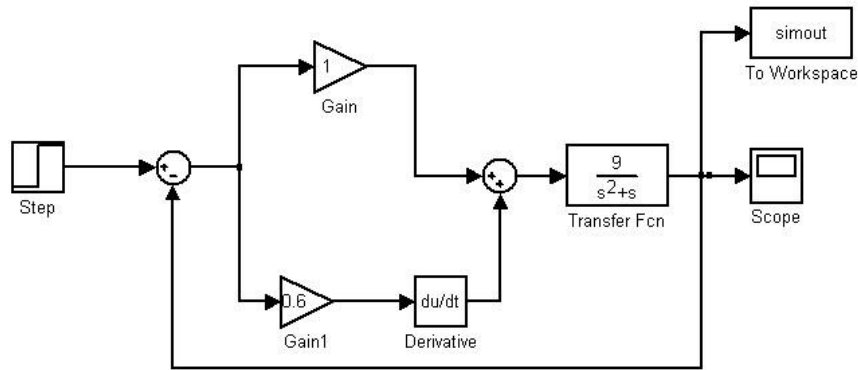


Figure 10 Close loop with PD controller

When  $l_d = 0.6$ , is the closed-loop performance analogous to a critical damped system (damping ratio  $\zeta = 1$ )? What is the bandwidth of the closed-loop system?

Tips: If you want to calculate the bandwidth of system  $\text{sys} = \frac{s^2 + 1}{s^4 + 2s^3 + 6s^2 + 7s + 1}$ , use the commands: “`sys=tf([1,0,1],[1,2,6,7,1])`” and “`bandwidth(sys)`”.

## 4 Design and Analysis of a Discrete-time System with Digital PD

In this part of the experiment, a discrete-time system with digital PD controller will be simulated and evaluated.

An important class of digital control algorithm is obtained by direct discretization of well-known analog controllers (emulation). Several methods of discrete equivalence, such as backward difference and Tustin approximation are available and the common assumption is that the sampling interval  $T$  is sufficiently small. Their performance will thus deteriorate rapidly when  $T$  exceeds certain limits.

**Step 3:** Discretize the analog PD controller  $l_p + l_d s = 1 + 0.6s$  using backward rule ( $C(z) = C(s) \big|_{s=\frac{z-1}{Tz}}$ ) with the sampling period  $T=0.03$  and  $0.1$  seconds respectively. **Note that** the “Sampling time” parameter in the **Discrete Transfer Fcn** block and **Zero-Order Hold** block should be set according to  $T$ . (These two blocks can be found from the *Discrete* panel in Simulink Library Browser.) Plant in simulink is shown as Figure 11. Simulate the closed-loop step responses, compare them with the continuous case in **Step 2**, and comment on the results.

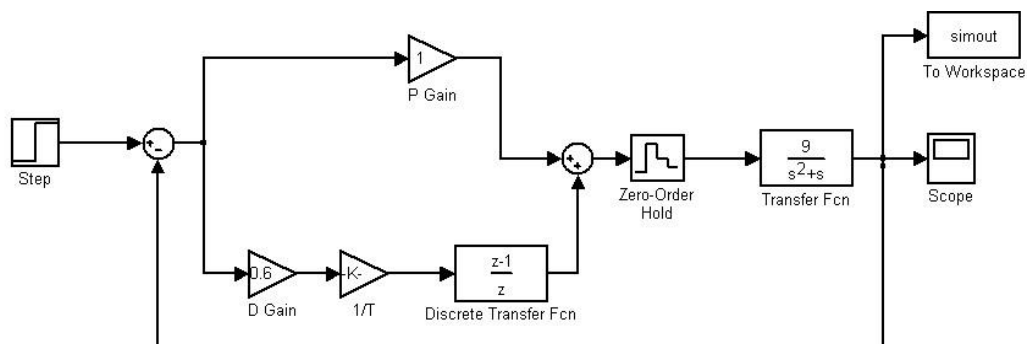


Figure 11 Plant with PD controller discretized by backward rule

**Step 4:** Discretize the PD controller  $l_p + l_d s = 1 + 0.6s$  using bilinear rule ( $C(z) = C(s) \big|_{s=\frac{2}{T} \frac{z-1}{z+1}}$ ) with

the sampling period  $T=0.03$  and  $0.1$  seconds respectively. Simulate the closed loop step responses, compare them with the discrete cases in **Step 3**, and comment on the results. Note that the “Sampling time” parameter in the **Discrete TF** block and **ZOH** block should be set according to  $T$ . Plant in simulink is shown as Figure 12.

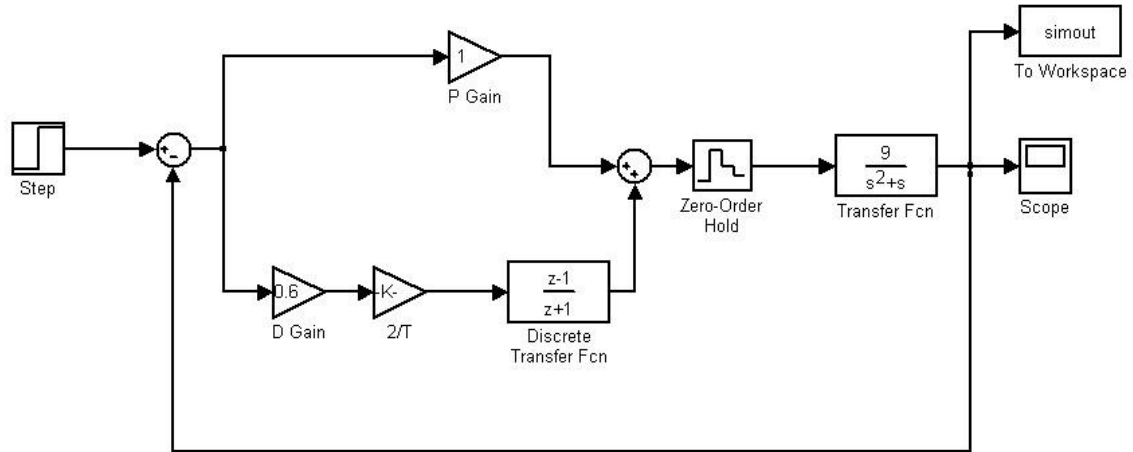


Figure 12 Plant with PD controller discretized by bilinear rule

## 5 Design and Analysis of a Discrete-time System with Compensator

In this part of the experiment, a digital control system with a phase lead compensator is designed, simulated and evaluated.

**Step 5:** For the servo model,

$$G(s) = \frac{9}{s(s+1)}, \quad (4)$$

use Matlab command “margin(sys)” finding its gain margin and phase margin.

**Step 6:** The lead compensator is chosen to be

$$C(s) = \frac{\tau s + 1}{\alpha \tau s + 1}. \quad (5)$$

By comparing it to the one in the lecture slides, you may know that the  $k=1$  in (5). From the **Step 5**, we know the phase margin of (4) is  $\phi$ , let the desired  $\phi_{desire} = 40^\circ$ . Following the **Lead Compensation Design Procedure** (Chapter Four, section 4.9.3) provided in the lecture notes, find the gain crossover frequency  $\omega_g$  by checking the Bode plot. Evaluate the phase margin  $\phi$  of  $kG(j\omega)$  at the gain crossover  $\omega_g$ . Then decide the compensator parameters by using:

$$\phi_m = \phi_{desire} - \phi + (5^\circ \rightarrow 12^\circ) \quad (6)$$

$$\alpha = \frac{1 - \sin \phi_m}{1 + \sin \phi_m} \quad (7)$$

Find out the frequency  $\omega_m$ , at which the magnitude of the uncompensated system  $kG(j\omega)$  is equal to  $-20\log_{10}(1/\sqrt{\alpha})$  dB, and then compute

$$\tau = \frac{1}{\omega_m \sqrt{\alpha}}. \quad (8)$$

**Step 7:** Evaluate the compensated system via the Bode plots (use command “margin(C(s)\*G(s))”). Check whether the frequency domain specification is satisfied. Redo if necessary. (Tips: you can use the MATLAB function **series(sys1, sys2)** to compute the series connection of two models sys1 and sys2.)

**Step 8:** Simulate the unit step response of the closed-loop system with the lead compensator C(s). Compare it with the uncompensated system, i.e. C(s)=1, and give your comments.

**Step 9:** Discretize the lead compensator C(s) to be C(z) by using bilinear transform ( $s = \frac{2}{T} \cdot \frac{z-1}{z+1}$ ), with sampling period T=0.03s, and T=0.2s respectively. (Tips: you can compute it manually or use the “c2d” function in MATLAB.)

**Step 10:** Simulate the unit step response of the closed-loop system with the digital lead compensator with T=0.03s and T=0.2s. Compare it with the performance of the analog controller C(s), and give your conclusions.

Tips: Command “step(sys)” for step response of system “sys”. Command “sys=feedback(C(s)\*G(s),1)” returns an LTI model sys for the negative feedback connected as Figure 9. For more information please check **help**.

## 6 References

1. Lecture Notes: EE3304 Digital Control Systems, Part II, 2015.
2. Simulink User's Guide
3. Franklin, Powell and Workman, Digital Control of Dynamic Systems, Addison Wesley, 3rd Edition, 1998.
4. Ogata, Modern Control Engineering, 3rd Edition, Prentice-Hall, 1997.