NATIONAL UNIVERSITY OF SINGAPORE

EE3331C – FEEDBACK CONTROL SYSTEMS

Instruction Manual for Experiment 1

Design and Simulation of Feedback Control Systems

Objectives

1. To learn to use MATLAB
2. To design P-controller and PI-controller for a 1^{st} order plant
3. To study through simulation the effects of controller gains on closed loop response

1 Background

1.1 Introduction

MATLAB is a high-level programming language and interactive environment that allows the user to perform computationally intensive tasks with less effort compared to traditional programming languages such as C and C++. It offers the facility to obtain numerical solutions of differential equations and matrix manipulation, to visualize data through 2-D and 3-D graphics, to implement an algorithm, to create suitable user interface, to interface with other programming languages etc. Add-on toolboxes extend the MATLAB environment to solve particular classes of problems in specific application areas, such as, signal processing, image processing, control design, communications, financial modeling and analysis, etc.

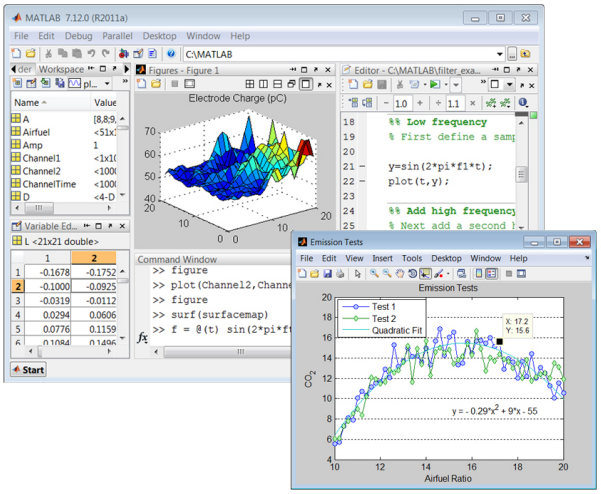

Figure 1: Screenshot of MATLAB.

In this simulation exercise, you will use functions that come with Control Systems Toolbox of MATLAB. A list of useful functions is provided in this manual. You will need them at different steps of control design and simulation. You may also use other function not listed in this manual. Exactly which function is needed depends on what steps you follow to design the controller. General guideline for design of the controller is given in the section titled 'Design and Simulation'.

1.2 Familiarization with MATLAB

Commonly used commands and functions are given in this Section. Brief explanation is also given on how to use each of these commands and functions. You don't have to practice all commands and functions given in this section. This section is for reference only. In the next section and in subsequent sections, you will use functions required to simulate response of dynamics systems and to design feedback system.

a. Command Prompt

- MATLAB Comman prompt is >>.
- Type a=5 at the command prompt and press ENTER. What do you see? MATLAB prompts back the value of the variable a.
- Type b=3; at the command prompt and then press ENTER. This time MATLAB doesn't prompt back. However, the variable 'b' is still there in MATLAB workspace.
- Type who at the command prompt and then press ENTER. MATLAB will list all variable in the workspace. For the above exercise, they are a and b.

b. Entering Matrices: You can enter matrices by using square brackets. For example, at the command prompt, type $x = [1 \;\; 10; \;\; 2 \;\; 5]$ and then press ENTER. This will create variable x in the workspace with the following value

$$x = \begin{bmatrix} 1 & 10 \\ 2 & 5 \end{bmatrix}$$

Rows of the matrix are separated by ;. You must give space between any two elements of a row.

A row vector can be defined as $y = [1\ 10\ 15]$ and, similarly, a column vector as $z = [1; 10; 15]$.

The command $transpose(m)$ returns the transpose of matrix m. So $z = [1; 10; 15]$ is identical to $z = transpose([1\ 10\ 15])$.

Entering $v = [2 : 1 : 10]$ creates a vector with elements of increasing value. The first element is 2 and the last element is 10; increment between adjacent elements is 1. So $v = [1 : 1 : 10]$ generates a row vector $v = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10]$. Vector with elements having increasing value can also be generated using the function $linspace$. If you type $v = linspace(1, 10, 100)$, MATLAB returns a vector v with 100 elements whose first element is 1 and the last element is 10.

$v = [10 : -2 : 2]$ generates a vector with elements decreasing by 2, i.e., the vector $v = [10\ 8\ 6\ 4\ 2]$.

c. Basic Operations

- Add or Subtract: x = A+B, y = A-B. Make sure that dimensions of A conforms to the dimensions of B.

- Product: x = A*B; [dimensions must agree].
- Division: x = A/B; [dimensions must agree].
- Inverse: x = inv(A); [A must be a square matrix]
- Element by element Product: x = p.*q;

d. M-files: MATLAB can execute a sequence of statements saved in a file. Such files are called "M-files" and must have the file type ".m" as the last part of their filename. There are two types of M-files, $scriptfiles$ and $functionfiles$. For this lab, you will use script file only. You can create a script file using MATLAB editor.

- To open the editor, go to File → New → Script. The editor window appears with a blank page.
- In the editor window, type in the commands that you would like to execute.
- Save the file with a name given, i.e., abc.m
- You can later execute all commands or function written in the file by typing the file's name (abc) at MATLAB command prompt (>>) and pressing the ENTER key.

Exercise:

i. Open editor window

ii. Type the following commands,
$a = [1\ 10; 2\ 1];$
$b = inv(a);$

iii. Save the file as abc.m

iv. Go to MATLAB command window, type *abc* at the command prompt and press ENTER

e. Graphics:

- the command $plot(x)$ opens a figure window that shows the plot of the vector x.
- the command $plot(x1, x2)$ shows the plot with the variable $x1$ along the horizontal-axis and $x2$ anong the vertical-axis.

- the command *grid* creates grid lines on the plot.
Type the following at the command prompt,
$t = [-1 : 0.1 : 1];$
$y = 2 + 3 * t;$
$plot(t, y), grid$

f. Control related functions:

- Define a transfer function-
$Num = [1\ 10];$
$Den = [1\ 3\ 5];$
$G = tf(Num, Den);$
the function tf generates the transfer function with the given numerator

(Num) and denominator (Den). If you type G at the command prompt and press ENTER, matlab will show the following

$$G = \frac{s+10}{s^2+3s+5}$$

- Define a transfer function with transportation delay-
  $Num = [1\ 10];$
  $Den = [1\ 3\ 5];$
  $td = 0.1;$
  $G = tf(Num, Den, 'InputDelay', td);$
  If you type G and press ENTER, matlab will show the following

$$G = \frac{s+10}{s^2+3s+5}e^{-0.1s}$$

- Simulate step response -
  - The command $step(G)$ generates step response of the system described by G and show it in a figure window.
  - You can specify the time vector before generating the step response.
    $t = [0 : 0.01 : 10];$
    $step(G, t);$
    where the transfer function G has already been defined.
  - These two examples of *step* do not create a output vector in the workspace. It simply shows the response in a figure window. If you want to make the output vector available in the workspace, use the following:
    $t = [0 : 0.01 : 10];$
    $y = step(G, t);$
  - If you type $[y, t] = step(G)$, MATLAB generates step response of the dynamic system defined by G using automatically generated time vector and return both t and y.
  - You can specify the duration of step response using $[y, t] = step(G, Tend)$, where $Tend$ is the end time of simulation. MATLAB generate the time vector t automatically.
- Root Locus - The command $rlocus(G)$ creates the root locus of the transfer function G already defined. Note that for system with delay, you need to approximate the delay using Pade Approximation before you can use the command.
- Generate Bode plot
  - The command $bode(G)$, where G is a dynamic system already defined, shows the Bode plot in a figure window.
  - You can specify the frequency vector ω before generating the Bode plot and use $bode(G, \omega)$ instead.
  - If you want the results to be available in the workspace for further processing, use

$[mag, ph] = bode(G, \omega)$.
It creates two vectors - mag for the magnitude of the Bode plot and ph for the phase.

- Instead of user defined frequency vector, the Bode plot can be generated for MATLAB-generated frequency vector using $[mag, ph, \omega] = bode(G)$.

- Obtain closed loop system from the open loop - If G and H are the forward path transfer function and feedback path transfer function, respectively, you can determine the transfer function system of the closed loop using the following command
$Gc = feedback(G, H)$;
The command $Gc = feedback(G, 1)$ returns the closed loop transfer function under unity feedback.

1.3 Selecting Gains of the PID Controller

The PID controller generates control input according to the following,

$$u(t) = K_p e + K_i \int e(\tau) d\tau + K_d \frac{de}{dt}$$

Taking Laplace transform,

$$\begin{aligned} U(s) &= K_p E(s) + K_i \frac{E(s)}{s} + K_d s E(s) \\ &= (K_p + \frac{K_i}{s} + K_d) E(s) \\ &= K_p (1 + \frac{1}{T_i s} + T_d s) E(s) \end{aligned}$$

where, $T_i = \frac{K_p}{K_i}$ and $T_d = \frac{K_d}{K_i}$. So the transfer function of the PID controller is,

$$C(s) = \frac{U(s)}{E(s)} = K_p \frac{T_d T_i s^2 + T_i s + 1}{T_i s}$$

For a first-order plant transfer function, the denominator of the resulting closed loop transfer function is quadratic function of s. One can find the gains such that closed loop poles have desired narural frequency and damping factor. However, the closed loop will also have zero which will make the response deviate from the expected response of the designed closed loop poles. For plant with higher order transfer function, finding the PID gains become more difficult using this approach. Ziegler-Nichols tuning of the PID controller is a widely used method for tuning of the PID gains. There are two different ways of finding the gains using Ziegler-Nichols tuning. In this experiment, you will learn one of them, namely, the method based on the process reaction curve.

1.4 Ziegler-Nichols Tuning

Step responses of a large number of process control systems exhibit a process reaction curve which can be generated from experimental step response data. If a tangent is drawn at the inflection point of the s-shaped reaction curve (shown in Figure 2), the slope of the line is $R = \frac{A}{\tau}$ and the intersection of the tangent line with the time-axis identifies

the time delay $L = t_d$. The controller parameters are designed to result in a closed-loop step response with a decay ratio of approximately 0.25. This means that the transient decays to a quarter of its value after one period of oscillation.

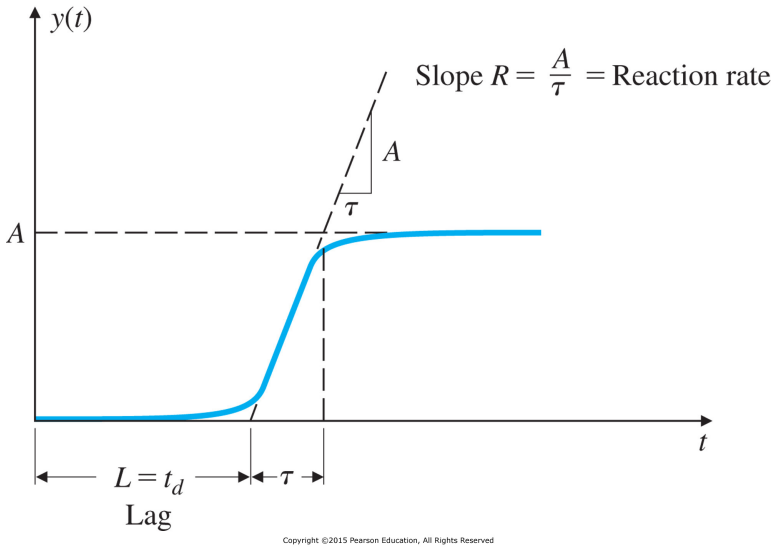

Figure 2: Reaction curve.

Ziegler-Nichols tuning formulae for different controllers are given in Table 1.

Table 1: Ziegler-Nichols Tuning Formulae for Decay Ratio of 0.25

Type of Controller	Optimum Gain
P	$K_p = \frac{1}{RL}$
PI	$K_p = \frac{0.9}{RL}, T_i = L/0.3$
PID	$K_p = \frac{1.2}{RL}, T_i = 2L, T_d = 0.5L$

2 Hands-On Exercise

You will use MATLAB to simulate responses of a plant and also under closed loop control using P-control and PI-control. All observations must be noted down and responses are to be printed. Printouts must be attached with the report to be submitted at the end of the session. You are also required to submit the list of all commands and functions used. Write these commands and functions in a script file and take printout at the end of the session.

2.1 Open Loop Response of a First-Order Transfer Function

1. You will be given the actual plant model during the laboratory session.

2. Create a new script file. Open MATLAB editor and enter the following lines:
   $A = xx$;
   $tau = xx$;
   $td = xx$;

   The plant can then be generated using the following command:
   $Gp = tf(A, [tau\ 1], 'InputDelay', td)$;

   These lines define the following 1^{st}-order plus delay transfer function,

   $$G_p = \frac{A}{\tau s + 1} e^{-t_d s}.$$

   Add the command
   $[y, t] = step(Gp, Tend)$;
   to simulate step response, and the following line to plot the step response.
   $figure(1), plot(t, y), grid, title('Step\ response\ of\ plant')$;

   $Tend$ is the end time of simulation duration. Choose the value for $Tend$ such that relevant features of step response, e.g., delay, transient and steady state are clealry visible in the plot.

   The *plot* command draws y as a function of t, the command *grid* creates grid lines on the plot, *title* adds a caption to the plot. The function $figure(1)$ opens a figure window and labels it as number 1.
   Save the lines you wrote in the editor as a script file. Let the name of the file be abc.m

3. Type abc at the command prompt. It will execute all commands included in the m-file abc.m.

4. Take a printout of the response, which is the step response of the plant G_p. Estimate the first-order plus delay plant model from the printout.
   From next section onwards, you will use this estimated plant model.

2.2 Study the Effect of K_p of the P-Controller

In this section, you will simulate closed loop response when the plant is put under feedback control using a proportional controller.

- The controller produces a signal proportional to the error, *i.e.*, $u(t) = K_p e(t)$, where $e(t)$ is the error and K_p is the proportional control gain,

$$\begin{aligned} u(t) &= K_p e(t), \\ U(s) &= K_p E(s). \end{aligned}$$

The transfer function of the proportional controller is,

$$C(s) = \frac{U(s)}{E(s)} = K_p. \tag{1}$$

The closed loop transfer function is,

$$G_{cl} = \frac{K_p G_p}{1 + K_p G_p}$$

2.2.1 Closed loop step response

Use the step response obtained earlier to determine the proportional control gain K_p using the Ziegler-Nichols tuning formula for P-controller. Let this value be x. You can use the following functions to simulate closed loop response
$Kp = x;$
$Gol = series(Kp, Gp);$
$Gcl = feedback(Gol, 1);$
$[y1, t1] = step(Gcl, Tend);$

The first one of these four lines ($K_p = x$) defines the controller transfer function. The second line defines the open loop transfer function formed by series connection of controller (K_p) and plant (G_p). The third line finds the closed loop function for unity feedback, and the fourth line simulates the step response of the closed loop.

Simulate closed loop response with two other gains, one greater than x and one smaller than x. For example, you may choose $xh = 1.25x$ and $xl = 0.75x$, respectively. Let the outputs be $y2$ and $y3$, respectively for these two gains and the corresponding time arrays are $t2$ and $t3$.

- Show the closed loop step responses for all three gains on a single plot.
  $figure(2), plot(t1, y1, t2, y2, t3, y3), grid;$

  The function $figure(2)$ creates a new figure window and plots step responses in this new window. Add a title to the figure.
  $title('Closed\ loop\ step\ response\ with\ P - control');$

  Take printout of this figure. From the plots of the closed loop step responses, determine steady-state error and rise time. How does the steady-state error vary with changing gain? Does it increase or decrease? What about response time? You may change the gain (K_p) to few other values to verify your observations. Include these observations in your report.

2.2.2 Bode plot

- Generate Bode plot data for the plant:
  Use the following commands to generate Bode plot of the plant.
  $[mag, php, w] = bode(Gp);$
  $dbp = 20 * log10(mag);$
  The first function generates Bode plot data for the plant described by Gp using MATLAB-generated frequency vector. The function also returns the frequency vector w. The second line gets the dB magnitude.

- Generate Bode plot data for open loop transfer function:
  $Kp = x;$
  $[mag, phol1] = bode(series(Kp, Gp), w);$
  $dbol1 = 20 * log10(mag);$
  The function $series(Kp, Gp)$ inside the function *bode* creates the open loop transfer function $G_{ol}(s) = C(s)G_p(s) = KpG_p(s)$.
  Repeat this for another value of $Kp > x$, say, xh.

  $Kp = xh;$
  $[mag, phol2] = bode(series(Kp, Gp), w);$
  $dbol2 = 20 * log10(mag);$

  Use the following functions to open a new figure window and show the Bode plots there.
  $figure(3);$
  $subplot(211), semilogx(w, dbp(:), w, dbol1(:), w, dbol2(:)), grid;$
  $title('Open\ loop\ Bode\ (mag)\ plot\ with\ P - Control');$
  $legend('Plant', 'gain = Kp', 'gain > Kp');$
  $subplot(212), semilogx(w, php(:), w, phol1(:), w, phol2(:)), grid;$
  $title('Open\ loop\ Bode\ (phase)\ plot\ with\ P - Control');$
  $legend('Plant', 'gain = Kp', 'gain > Kp');$

The function $subplot(mnk)$ divides the figure window into $m \times n$ sub-windows and show the plot in the k^{th} sub-window. For example, $subplot(211)$ divides the window into two rows and one column; draws the plot in the 1^{st} sub-window.

Compare bode plots of the plant and of the two open loop transfer functions. Your report must include your observations about the effect of gain K_p on the open loop Bode plots.

- Generate Bode plot data for closed loop transfer function:
  $Kp = x;$
  $[mag, phcl1] = bode(feedback(series(Kp, Gp), 1), w);$
  $dbcl1 = 20 * log10(mag);$
  The function $feedback(series(Kp, Gp), 1)$ inside the function *bode* creates the closed loop transfer function for $KpG_p(s)$ in the forward path and unity gain in the feedback path.

  Repeat this for $Kp = xh$ by using the following
  $Kp = xh;$
  $[mag, phcl2] = bode(feedback(series(Kp, Gp), 1), w);$
  $dbcl2 = 20 * log10(mag);$

  Then plot them,
  $figure(4);$
  $subplot(211), semilogx(w, dbcl1(:), w, dbcl2(:)), grid;$
  $title('Closed\ loop\ Bode\ (mag)\ plot\ with\ P - Control');$
  $legend('gain = Kp', 'gain > Kp');$
  $subplot(212), semilogx(w, phcl1(:), w, phcl2(:)), grid;$
  $title('Closed\ loop\ Bode\ (phase)\ plot\ with\ P - Control');$
  $legend('gain = Kp', 'gain > Kp');$

2.3 PI Controller

- Proportional-plus-Integral (PI) Control: The PI control consists of a proportional gain and an integral gain. The integral part produces a correcting signal proportional to the integral of the error signal,

$$\begin{aligned} u(t) &= K_p e(t) + K_i \int e(t)dt, \\ U(s) &= K_p(1 + \frac{1}{T_i s})E(s). \end{aligned}$$

  The transfer function of the PI controller is,

$$C(s) = \frac{U(s)}{E(s)} = K_p \frac{T_i s + 1}{T_i s}.$$

- Next you will simulate the closed loop response with PI control for different values of K_p and T_i. Choose the initial gains using the Ziegler-Nichols tuning method for PI-control (given in Table 1). Let these values be $K_p = \alpha$ and $T_i = \beta$. The following lines simulate the step response with PI-control.
  $Kp = \alpha$;
  $Ti = \beta$;
  $C = Kp * tf([Ti \;\; 1], [Ti \;\; 0])$;
  $Gol = series(C, Gp)$;
  $Gcl = feedback(Gol, 1)$;
  $[y4, t4] = step(Gcl, Tend)$;

- Keeping T_i fixed, increase K_p to a higher value and repeat the step response simulation. Let the output be $y5$. Plot step responses in a single figure and take printout, $figure(5), plot(t4, y4, t5, y5), grid$;

  Observe the changes in response. Your report should include these observations and your comments on them.

- Generate open loop Bode plot for one set of $K_p = \alpha$ and $T_i = \beta$. How does the PI-controller modify the magnitude and phase plot of the open loop transfer function compared to those with P-controller? These observations are to be included in the report.

3 Report

We suggest you write your report in Word, copy your plots and Matlab codes and upload to the LumiNUS 'Lab1 report upload' folder. Your report must include the following key issues.

1. For P-control, the effect of gain K_p on step response and Bode plots
2. For PI-control,
   - effect of integral control on steady-state error
   - effect of increasing K_p on the closed loop step response
   - effect of integral control on the open loop Bode plot

Print/Save your file in pdf format. Name your report using your matric number, i.e. A1234567E.pdf and upload to LumiNUS.

END OF MANUAL