1 Objectives

The objectives of this lab is to perform the inter-sample response analysis and the computer simulation of a sampled-data system using MATLAB.

2 Introduction

A typical sampled-data system is shown in Figure 1, where $S$ and $H$ represent the Sampler (A/D converter) and the Holder (D/A converter) respectively.

![Sampled-Data System](figure1.jpg)

Unlike a pure analog system (that has an analog controller and the continuous-time plant), or a pure discrete-time system (with a digital controller and the discretized model of the plant), the sampled-data system has both the digital controller and the continuous-time plant. The signals passing through the entire system is a mixture of sampled data and continuous-time signals. Most control system simulation software packages such as MATLAB only have functions for continuous-time and discrete-time simulations, e.g., lsim, step, and dstep, dlsim, etc., but none for sampled data simulation. This lab is to write a general MATLAB program (function) to simulate the step response of a sampled-data (digital) control system.
3 Preparation

Before the experiment, the students should understand the procedure of the sampled-data simulation, and the procedure for calculating the inter-sample response.

4 Experiment Procedure

The analog plant $P(s)$ that we considered in this lab is a flexible beam system:

$$P(s) = \frac{1.6188s^2 - 0.1575s - 43.9425}{s^3 + 0.1736s^5 + 27.9001s^2 + 0.0186s}$$

(1)

Assume that you have designed an advanced digital controller for this complex system (at sampling period $T = 0.5$ sec), i.e. as in Figure 1:

$$C_d(z) = \frac{-0.1084z^5 - 0.01202z^4 + 0.1708z^3 + 0.08469z^2 - 0.09198z - 0.04313}{z^6 - 0.6528z^5 - 0.8377z^4 + 0.4495z^3 + 0.4709z^2 - 0.5820z + 0.1521}$$

(2)

Before we can implement this digital controller on-line and perform real-time control on the real flexible beam system, first of all, we need to test the control system performance in a computer environment. This can be achieved by sampled-data (digital) simulation.

The following steps describe the procedure for simulating the step response of the sampled-data system using existing MATLAB functions:

1. Define the models of the digital controller and the analog plant, e.g.

   ```matlab
   sysC_d = tf(numC_d, denC_d, T)  % Create a discrete-time transfer function with sampling period T (T = 0.5)
   sysP = tf(numP, denP)
   ```

2. Discretize the continuous-time plant $P(s)$ to obtain the ZOH equivalent $P_d(z) (P_d(z) = SP(s)H)$ via the MATLAB function `c2d`. 


The discretized system is shown in Figure 2:

![Discretized System Diagram]

Here \( r_d \) is the sampled input sequence, which is a discrete-time unit step. Figure 2 is the exact model of Figure 1 at sampling instants.

3. Obtain the control signal sequence \( u_d \). First of all, compute the transfer function from \( r_d \) to \( u_d \) in Figure 2,

\[
G_1(z) = \frac{U_d(z)}{R_d(z)} = \frac{C_d(z)}{1 + C_d(z)P_d(z)}
\]  

\( G_1 = \text{feedback(sysC.d, sysP.d)} \)

and then compute the discrete-time control sequence \( u_d \) via \text{dstep}, e.g.

\[
\text{[numG1,denG1]} = \text{tfdata}(G1, \text{'v'})
\]

\[
u_d = \text{dstep(numG1,denG1,N)}
\]

\textbf{Note:} \( N \) is the number of samples to be simulated

4. Simulate zero-order-hold, i.e., compute the continuous-time control input \( u \) after the ZOH in Figure 1 (\( u = Hu_d \)). Suppose we would like to compute \( n \) inter-sample points for every sampling period (of length \( T \)), the time increment (step size) in \( u \) is then \( T/n \). Thus the vector \( u \) is obtained by holding each value of \( u_d \) for \( n \) time, i.e.

\[
u = [u_d(1)u_d(1)\ldots u_d(1)u_d(2)\ldots u_d(2)\ldots \ldots u_d(N)\ldots u_d(N)]
\]  

\text{the for-end loops can be used to obtain } u, \text{ and the length of } u \text{ is } N \times n

5. With input \( u \) to \( P(s) \) obtained from above, the output \( y(t) \) of \( P(s) \) in Figure 1 can finally be computed by continuous-time simulation, e.g. \text{lsim}.  

3
(The obtained \( y(t) \) is in fact the computer approximation of the actual continuous-time output in the real system. It should be a good approximation because 10 points of the inter-sample response are calculated during each sampling interval \( T \).)

Finally, the above steps can be integrated into a general function named `sdstep` (or choose your own preferred name) in MATLAB:

\[
[y,u,t] = \text{sdstep}(\text{numC}_d,\text{denC}_d,\text{numP},\text{denP},T,N,n)
\]

For this function, the user can define the model of the digital controller in \( \text{numC}_d \) and \( \text{denC}_d \), and the model of the continuous-time plant in \( \text{numP} \) and \( \text{denP} \). \( T \) and \( N \) are sampling period and number of samples to be simulated, respectively; and the integer \( n \) is the ratio of sampling period \( T \) to the time increment \( T_1 \) used when calculating the inter-sample response. The returned variables of the above function are the output from continuous-time plant \( P(s) \), \( y(t) \) (see Figure 1, and the corresponding input \( u(t) \) and the time vector \( t \). (Actually, during debugging and the experiment, you can return as many inter-variables as you prefer.)

Once you have programmed the function `sdstep`, perform the following for this experiment:

- Debug your program(`sdstep.m`) and simulate the step responses of the sampled-data system with \( C_d(z) \), \( P(s) \) given in Eqs. (1) and (2), with the sampling period \( T = 0.5 \), number of samples \( N = 40 \), and the intersample ratio \( n = 10 \).
- Calculate the discrete-time output \( y_d \) based on the control sequence \( u_d \) and the discrete-time model using `dlsim`, the time sequence \( t_d \) can be chosen as

\[
t_d = [0:T:(N-1)*T]
\]

- On the same figure, plot the discrete-time control sequence \( u_d \) vs. \( t_d \), and the continuous-time control sequence \( u \) vs. \( t \), for example, use:

\[
\text{plot}(t_d,u_d,\text{`*'},t,u,\text{`r'})
\]

Can you see the effect of the zero-order-hold?

- On the same figure, plot the continuous-time output \( y \) vs. \( t \), and the discrete-time output (\( t_d \) vs. \( y_d \)).
plot(t_d,y_d,‘*’,t,y,’r’)

Does the discrete-time output samples match the continuous-time output at the sampling instants?

5 Lab Report

Your report should contain:

- A brief introduction.
- Your program in MATLAB for the sampled-data simulation function: sdstep.m
- All the plots.
- Answering the questions given at the end of Section 4.
- Any conclusions you would like to draw.

The lab report is due by 4pm on Mar. 7, 2011 for students in section H3 and 4pm on Mar. 14, 2011 for students in section H4. Please drop your report into the box assigned to EE461 LAB outside the main office at 2nd floor ECERF.