SFWR ENG Assignment 3: Digital Control Implementation

Due: 0830 Tuesday November 19, 2002

For this assignment you will hand in one copy of the assignment for each lab group.

Lab Work

1. Make sure that the PC is boots the RTAI patched version of Linux If you are logged into the system then you can check this with the command `uname -a` which will display something like:

   ```
   [lawford@rtlab-05 fifo]$ uname -a
   Linux rtlab-05 2.4.9-rthal5 #7 Wed Nov 14 11:30:23 EST 2001 i686 unknown
   [lawford@rtlab-05 fifo]$
   ```

   We have used the `-rthal5` extension to identify a kernel with RTAI's Real-Time Hardware Abstraction Layer patch compiled in.

2. Read through the RTAI Tutorials available on the `rtlab-xx` machines in HTML at:

   `file:///usr/local/src/lineo/html/day1/Details.htm`

3. Copy the ClassLab examples to your home directory and make then writable with the commands:

   ```
   cp -r /usr/local/src/lineo/ClassLabs/ .
   chmod -R u+w ./ClassLabs/
   ```

   You will edit and compile the files in this directory to work through the tutorials.

4. Work through the Lineo Real-Time Linux Tutorials. Complete the working parts of the tutorial up to at least the end of the Day 2 shared memory PWM example and first part of the debugging. If you have time, also look at the Day 3 examples.

   We do not have the R2D2 program set up and the kernel we are running does not support the LTT so you can skip those parts. If you have any problems with the examples, consult the TAs.

   NOTE: You do not have root access on the lab machines so the `su` command used in the tutorials will not work for you. Instead we have set up the `sudo` command to allow you to insert and remove modules. E.g., to insert the `rtai` module and `rtai_sched` module, you would type the commands:

   ```
   sudo /sbin/insmod rtai
   sudo /sbin/insmod rtai_sched
   ```

   To remove these commands you would type:

   ```
   sudo /sbin/rmmod rtai_sched
   sudo /sbin/rmmod rtai
   ```

   The core RTAI modules can be found in the `lib/modules/2.4.9-rthal5/rtai` directory.

   For security reasons it is not possible to insert a module that is not on a local file system. Therefore, to insert a module you have written and compiled yourself, you must first copy it to the `/tmp` directory. E.g., for a compiled module called `servofifo.o` in your current working directory you would do the following:

```
cp servofifo.o /tmp
sudo /sbin/insmod /tmp/servofifo.o
```

5. The servo has a range of motion of 180°. Use the Servo Fifo example to determine the pulse widths w_0 and w_{180} that correspond to angles 0° and 180° respectively. Try a couple of points in between (e.g., $\frac{w_0+w_{180}}{2}$) to determine if the output angle of the servo varies linearly as the pulse width. What did you find?

Lab Related Questions

1. Consider a motor that is modeled by the transfer function

$$G(s) = \frac{Y(s)}{U(s)} = \frac{3}{s(s+3)}$$

in the feedback loop:

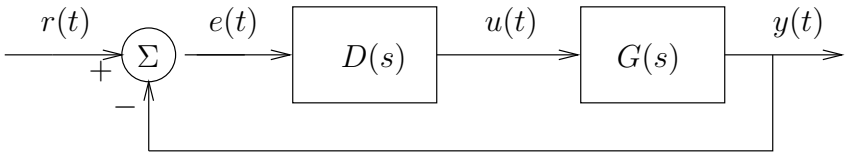

Figure 1: Continuous control loop

Unless stated otherwise, assume that the continuous controller is $D(s) = \frac{3}{4}$, a proportional controller with a gain of $\frac{3}{4}$.

a) Plot the root locus for $K > 0$ when $D(s) = K$. Identify the point where $K = \frac{3}{4}$ on the plot.

b) Do a Bode plot of the closed loop system $G_{cl}(s)$ when $D(s) = \frac{3}{4}$ and also plot the step response. What is the value of ω_{bw}, the bandwidth of this closed loop system?

c) The input reference signal $r(t)$ is to be generated by a digital system that only has a digital output that is either 0 Volts or 5 Volts. To do this you will use a pulse width modulated (PWM) signal on $r(t)$ to have the system output $y(t)$ track a desired reference signal y_{ref}. Experimentally using Matlab, Simulink, or Labview simulations, determine the minimum value, T_{min}, of the period of the pulses to be modulated, such that the closed loop system of Figure 1 with $D(s) = \frac{3}{4}$ .will "track" the signal: $y_{ref} = 4 \cdot 1(t)$without excessive ripple. How does ω_{bw} compare with T_{min}? Explain.

d) Let T_p be the basic period of the pulses being modulated for the PWM input $r(t)$. Plot the results of attempting to track the y_{ref} signal above using PWM signals with:

- **i)** $T_p = 10T_{min}$
- **ii)** $T_p = T_{min}$
- **iii)** $T_p = \frac{T_{min}}{10}$

e) What is the apparent frequency of the ripple for $T_p = \frac{T_{min}}{10}$? How does this relate to the Fourier series for a PWM signal of constant width pulse of 80%?

f) You have been asked to do a digital control design for the setting shown in Figure 2. Assuming a sampling rate of T, find the ZOH discrete equivalent $G_{ZOH}(z)$ system for $G(s)$. This represents the discrete transfer function from $u(k)$ to $y(k)$.

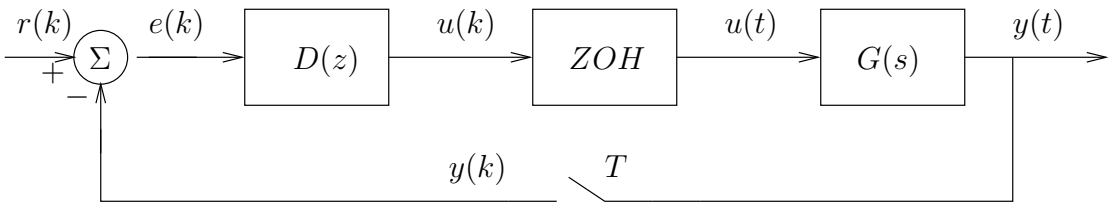

Figure 2: Block diagram for digital control loop

g) As we did before in the continuous case, it is decided to try to stabilize the system using proportional control (i.e., $D(z) = K$). Plot the discrete root locus for the system for $K > 0$ when $T = \frac{1}{30}$. Are there any significant differences between the root locus for the continuous control loop of Figure 1 and the digital control loop of Figure 2? Why?

2. The following lead network controller is designed to add approximately 60° of phase at $\omega_1 = 3$ rad/sec:

$$H(s) = \frac{s+1}{0.1s+1}$$

For each of the following design methods, compute the discrete equivalent controller and plot the z-plane poles and zeros. Compute the amount of phase lead given by the discrete controller at $z_1 = e^{j\omega_1 T}$ when $T = 0.25$ sec. Do the above for:

a) Forward rectangular rule

b) Backward rectangular rule

c) Trapezoid rule (aka bilinear)

d) Zero-pole mapping

e) Zero-order-hold equivalent

3. Use Matlab to create bode plots for each of the discrete equivalent controllers in the previous question.

4. Consider the difference equation:

$$y(k) + y(k-2) = u(k)$$

a) Find the transfer function $\frac{Y(z)}{U(z)}$.

b) Assuming zero initial conditions (i.e. $y(-1) = y(-2) = 0$), compute the system's step response.

c) The system is not BIBO stable. Find a bounded continuous input that, when sampled and used as input for $u(k)$, would cause the system's output to become unbounded.

NOTE: You do NOT have to compute what the output is, only suggest a bounded input that will produce an unbounded output.

5. Take a look at the Matlab Simulink Aerospace F14 Digital Control Demo example to see how integrator anti-wind up is done. Compare this to anti-windup mechanism used in the Labview PID block you used last lab. Explain the purpose of anti-windup.