# 6 Module 6: Sampled Data Systems

## 6.1 Overview

This module will look at ways of analyzing systems which have both continuous and discrete signals in the loop. The critical feature is that when a sampler if put in the loop, only the discrete signals behave in closed loop. The continuous signals can be treated in an open loop manner. The analysis then involves finding the zero-order hold equivalents for the pieces, calculating all of the discrete signals in the loop and feeding these discrete signals into the continuous parts to find the continuous signals. Study the example carefully to see how this is done.

We will then use this framework to look at the results of several designs. While the designs might look good from a discrete time point of view, when we look at the intersample behavior they may be quite bad.

The background material for this module is Chapter 3 of the text, although it will also require design concepts covered in Chapters 2, 4 and 5.

## 6.2 Notes

Here we will look at the effect of a deadbeat controller on a double integrator system. This will also illustrate how to use MATLAB to analyze interconnections of discrete and continuous systems.

The critical issue to note is that when a sampler is included in the loop, only the discrete system determines the closed loop behavior. For the loop to make sense, there must be a zero order hold (or other hold function) between the output of a discrete component and the input of a continuous component. Similarly, a sampler must be between the output of a continuous component and the input of a discrete component.

If these conditions are satisfied, the loop is meaningful, and the closed loop behavior can be studied. All continuous components are transformed to discrete time equivalents by taking into account the hold that preceeds them and the sampler that follows them.

The discrete loop can then be studied. This might involve the design of a discrete controller or a discrete controller might be calculated from a continuous one. In either case the closed loop properties, like stability, can be studied.

Figure 3 illustrates the configuration that we will look at here. As in the previous module, a zero order hold equivalent will be obtained for the sampler, $P(s)$, hold configuration. This will allow us to calculate the closed loop output, $y(k)$, from

a reference input, $r(k)$. We can also use these signals to calculate $u(k)$. To get $y(t)$ and study the intersample behavior we need only simulate P(s) and the zero order hold with input $u(k)$.

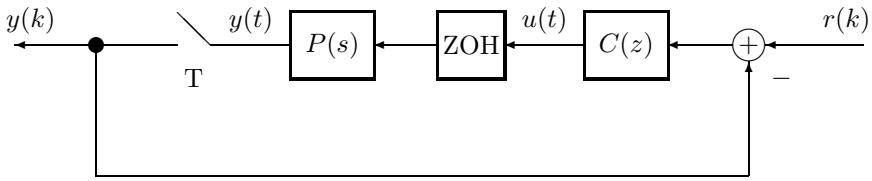



Figure 3: Closed loop sampled data system

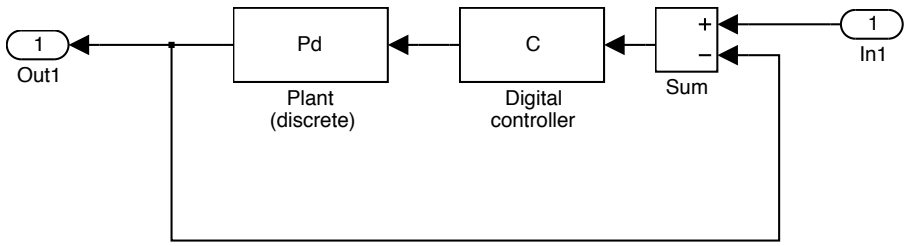
This module will make the point that, unless the sampling frequency is very fast compared to everything else, it is worthwhile going back and checking on the intersample behavior of the continuous components. This can be calculated by calculating the input to the hold preceeding the continuous part. The hold and continuous plant can then be simulated in continuous time.

The module also looks at the design of deadbeat controllers. These controllers are designed directly in the digital domain. In essence, we specify a desired closed loop transfer function, and then solve for the controller, $C(z)$. This can often be done, and as we will see, the controller does give the expected response in the discrete domain. Check it carefully in the continuous domain too; we get what we ask for, not necessarily what we want. In this case the example studied, a double integrator, is not open loop stable. Between samples, the continuous plant runs open loop which means that, in this case, it tends to go unstable. Other, more benign, systems can give a reasonable response under deadbeat control.

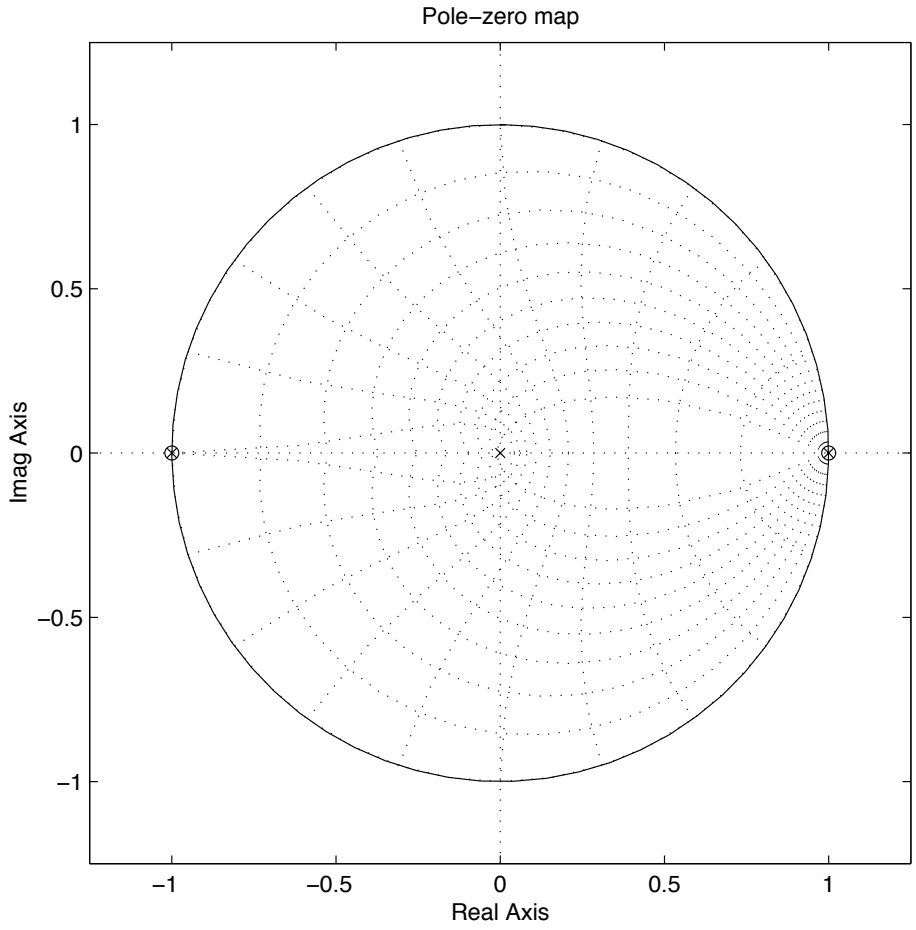
## 6.3 Module 6: Listing and Graphics

```
clf;axis([1,2,3,4]);axis;
echo on
%-------------------------------------------------------
%
%          Dept. of Electrical & Computer Eng.
%       University of California, Santa Barbara.
%
%               ECE 147 B:  Digital Control Systems
%
%               module 6:    Sampled Data Systems
%
%       ------------------------------------------------

% This module introduces the techniques required
% to study sampled data systems.  This material is
% introduced in Chapter 3 of the text.
%
% In sampled data systems we have both continuous
% and discrete time signals and would like to be able
% to do simulations to show both kinds of signal.

% Please adjust the Graph and Command
% windows for maximum size viewing.

pause   % press any key to continue

% We will study a dead beat control design for a double
% integrator plant (for example a satellite).  The plant
% is simply,
%
% P(s) = 1/s^2.

P = tf(1, [1,0,0]);

% Our control system will have a 1 second sample period.

T = 1.0;

% The input to the plant is preceeded by a zero-order
% hold and the output is sampled.  We can calculate
% the digital equivalent of this plant using sys2dsys.

Pd = c2d(P,T,'zoh');

pause % press any key to continue.
```

```
% Let's specify a desired closed loop transfer function,
% M(z).  Initially take M(z) to be a unit delay.

% It is a bit tricky to use Matlab to calculate the
% deadbeat controller.  The formula is,
%
%             M(z)
%       C(z) = ------------
%              (1-M(z))*P(z)
%
% The deadbead controller, C, required to give
% a tracking response of a unit delay is:
%
% 2(z-1)
% C  =  ----------
%              T^2(z+1)

C = tf([2,-2],[T^2,T^2],T);

pause % press any key to continue.

% Now we want to close the loop around  P(z)C(z),
% in the usual way.  Note that this gives a
% digital result which can be calculated in the
% usual way.

m6disc
```

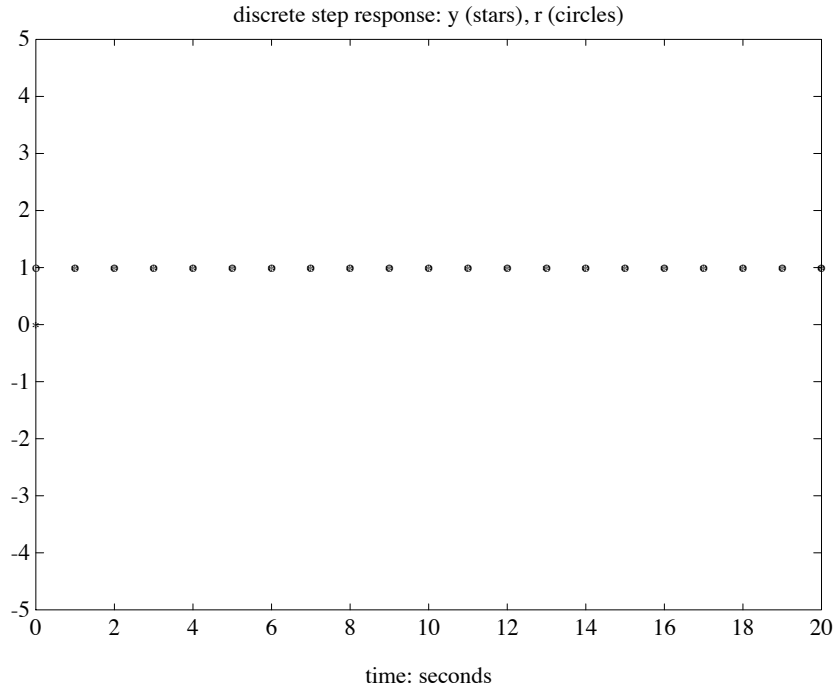



```
[a,b,c,d] = dlinmod('m6disc',T);
dclp = ss(a,b,c,d,T);

% Let's look at the poles of this.

clf
zgrid
axis('square')
axis([-1.25,1.25,-1.25,1.25])
pzmap(dclp)
```



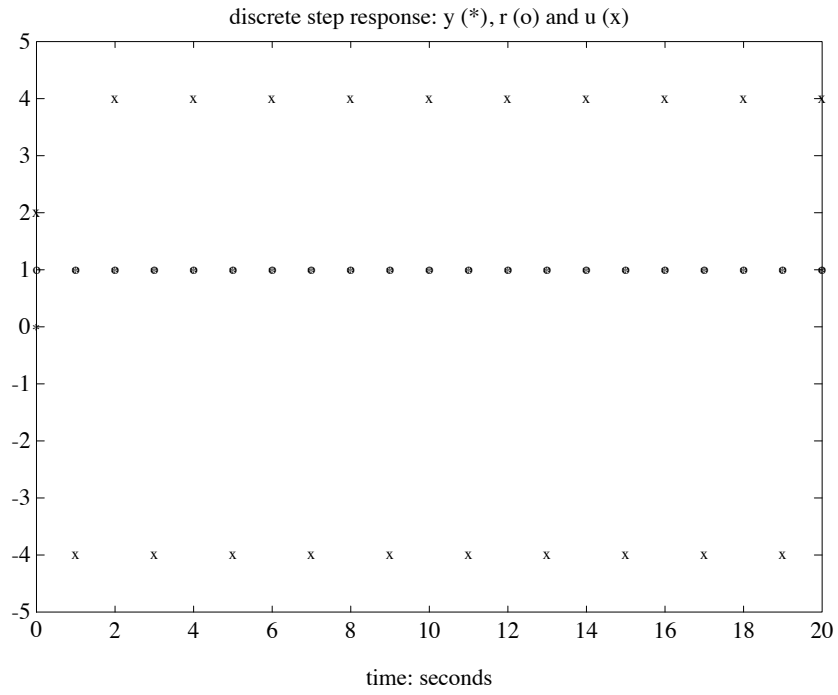

```
% Something funny is going on here - there are poles
% on the unit circle.

pole(dclp)

pause % press any key to continue.

% Now examine the step response, from input to output.

dtime = [0:T:20*T]';
r = ones(length(dtime),1);
yd = lsim(dclp,r,dtime);

clf
plot(dtime,yd,'*',dtime,r,'o')
```

```
axis([0,20*T,-5,5])
title('discrete step response')
legend('output: yd','reference: r')
xlabel('time: seconds')
```



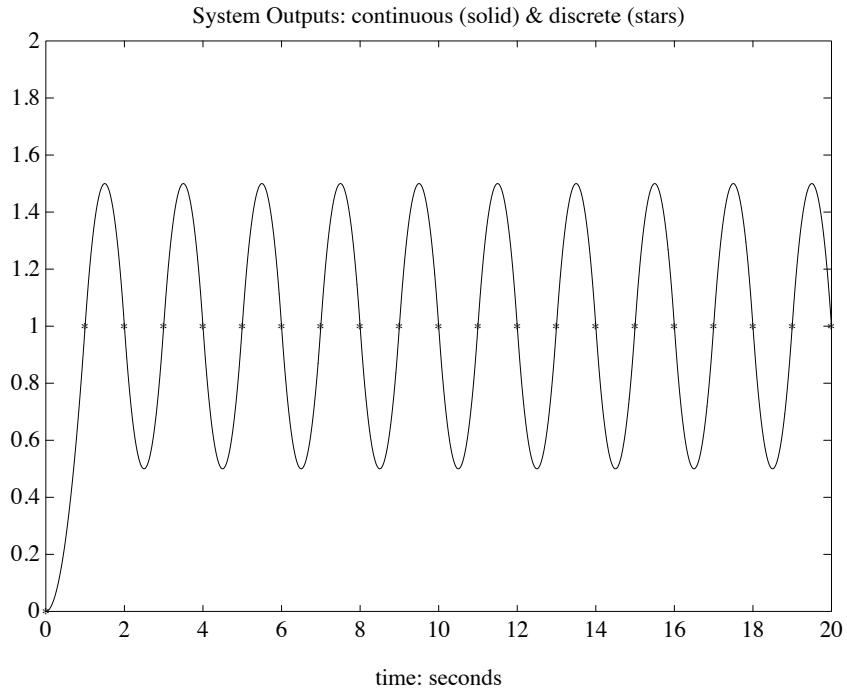

```
% This looks just fine.  The error goes to zero in one
% sample period.

pause % press any key to continue.

% Now we would like to see what the effect is on the
% plant itself.  To do this we have to be able to
% calculate the output of the controller (which is
% of course the input to the plant).

% It's easy to see the required relationship by looking at
% a block diagram,
%
% u(z) = C(z) * (r(z) - y(z)).
%
% Ok lets do this as a simulation to get u.

esig = r - yd;
```

```
u = lsim(C,esig,dtime);

% And include the controller output on the plot.

clf
plot(dtime,yd,'*',dtime,r,'o',dtime,u,'x')
axis([0,20*T,-5,5])
title('discrete step response')
legend('output: yd','reference: r','controller output: u')
xlabel('time: seconds')
```




```
% The controller output is a bit wild.  Here we see what
% the poles on the unit circle are doing - they create
% an oscillation at the controller output which doesn't
% die out.

pause % press any key to continue.

% Now do the cts time response by putting the controller
% output into a ZOH equivalent of the plant.  We can
% use the continuous time system for the simulation and
% recreate the effect of the zero-order hold on the
% input signal
```

```
tfine = [0:T/100:20*T]';      % make a fine time grid.
ufine = zeros(length(tfine),1);
echo off;
for i = 1:length(tfine),
  uptr = max(find(dtime <= tfine(i)));
  ufine(i) = u(uptr);
end; echo on

ycts = lsim(P,ufine,tfine);
plot(dtime,yd,'*',tfine,ycts,'-')
axis([0,20*T,0,2]);
title('System outputs')
legend('discrete output','continuous output')
xlabel('time: seconds')
```




```
% Here we see what happens to the continuous plant as a
% result of the input thrashing around.  Note that at each
% sample instant (after the one at zero) there is a no
% error.  However between samples the plant oscillates
% wildly.  Maybe we asked for too much here.  The problems
% will look at trying different closed loop specifications.
```

```
pause % press any key to continue.

% The critcal feature in this analysis was the fact that
% the loop was closed with a sampled signal.  So a discrete
% closed loop analysis could determine all of the signals
% at the sample times.  To get the intersample behavior,
% we just put the appropriate signals into the appropriate
% open-loop components.

%------------------------------------------------------------

echo off
```

## 6.4 Problems

1. Now let's try designing a continuous controller and converting it to a digital controller with a prewarped Tustin method.

   a) Design a continous controller which gives a stable response. If you use an aggressive design the closed loop will really get messed up when you make it digital. Go for something sedate.

   b) Simulate your design in the time domain.

   c) Choose a prewarping frequency and calculate a discrete time controller.

   d) Now simulate the discrete time time response using your discrete time controller. Adjust your original design and repeat if needed to get a reasonable response.

   e) Now simulate the intersample behavior of your design. Give a plot which shows the ideal continuous time response, the discrete time response and the actual (i.e. sampled data) continuous time response.

2. You will probably have noticed difficulties in doing all of the above problems. There is some feature which is always bad. Give a physical explanation why you cannot get around this difficulty.

3. Repeat all of the above problems for a different plant. In this case use,

   $$P(s) = \frac{1}{s^2 + 0.25s + 1},$$

   and $T = 0.2$. Does this plant exhibit the problems that the other one did?