8 Module 8: Observer Based Design

8.1 Overview

This module continues the state space approaches initially covered in module 7. The satellite antenna dish problem, illustrated in Figure 8, is again used as an example.

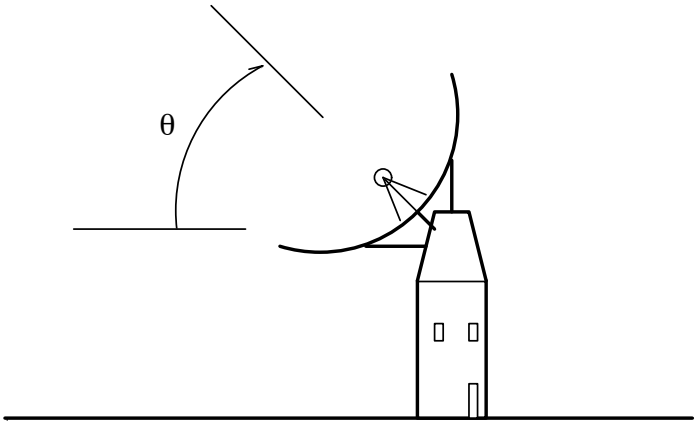


Figure 8: Satellite antenna dish

This module will use this example to study full and reduced order observer designs. The problems at the end will extend this to reference tracking and integral control designs.

8.2 Notes

In this case, only the dish angle, $\theta(t)$, is available for measurement. Again a discrete time implementation will be used so it is actually $\theta(k)$ that will be measured. The first objective will be to design a full order estimator to estimate $\theta(k)$ and $\dot{\theta}(k)$.

The new open loop plant, $P_\theta(s)$, is illustrated in Figure 9. Again a continuous time wind input ($Td(t)$) simulation is performed with $Tc(t) = 0$.

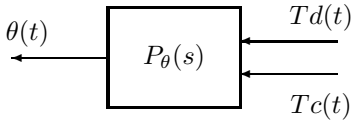


Figure 9: Open loop plant: Satellite antenna dish

An full order estimator is designed. The design configuration is illustrated if Figure 10. The estimated discrete states are $\theta_e(k)$ and $\dot{\theta}_e(k)$. In the simulation of this estimator, $\dot{\theta}(k)$ is also calculated and displayed.

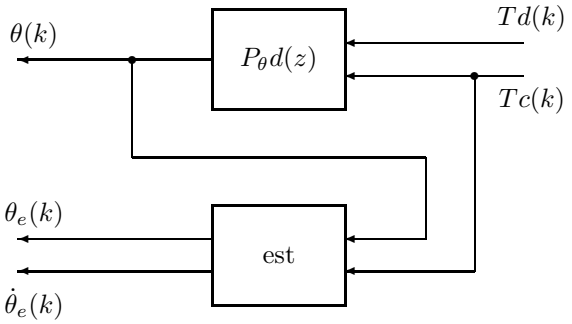


Figure 10: Estimator design configuration

The estimator, `est`, is tested and found to work well. A state feedback design is done and this is connected up to the estimator outputs. The resulting system is illustrated in Figure 11.

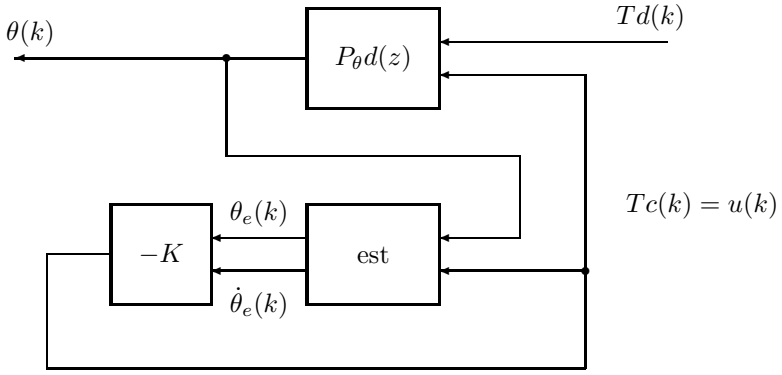


Figure 11: Full order estimator based control design

This discrete time control system is now studied using SIMULINK. The full set up is shown in Figure 12. The continuous response, $\theta(t)$, was found to be very close to $\theta(k)$.

The module next considers the problem of designing a reduced order estimator. The reduced order problem is set up such that the reduced order estimator, `estr`, is a plug in replacement for `est` in Figures 11 and 12. However only, $\dot{\theta}(k)$ is estimated. The measurement of $\theta(k)$ is used directly into $-K$. Figure 13 shows the construction of `estr` in order to achieve this. The notation in the module and the lecture notes are very similar to make referring from one to the other easy.

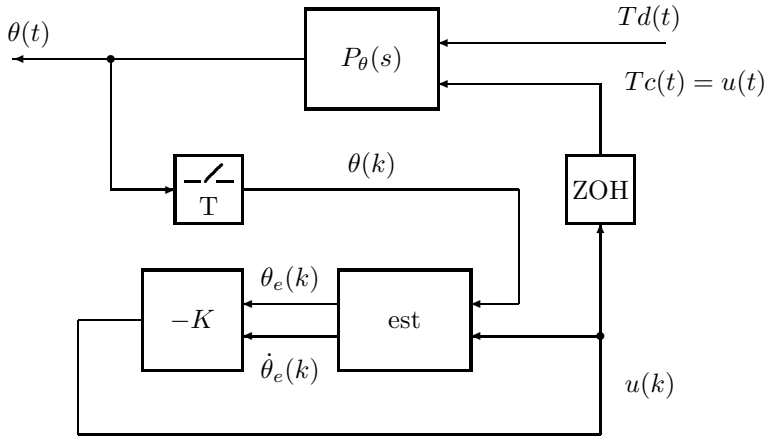


Figure 12: Sampled data system with full order estimator based control design

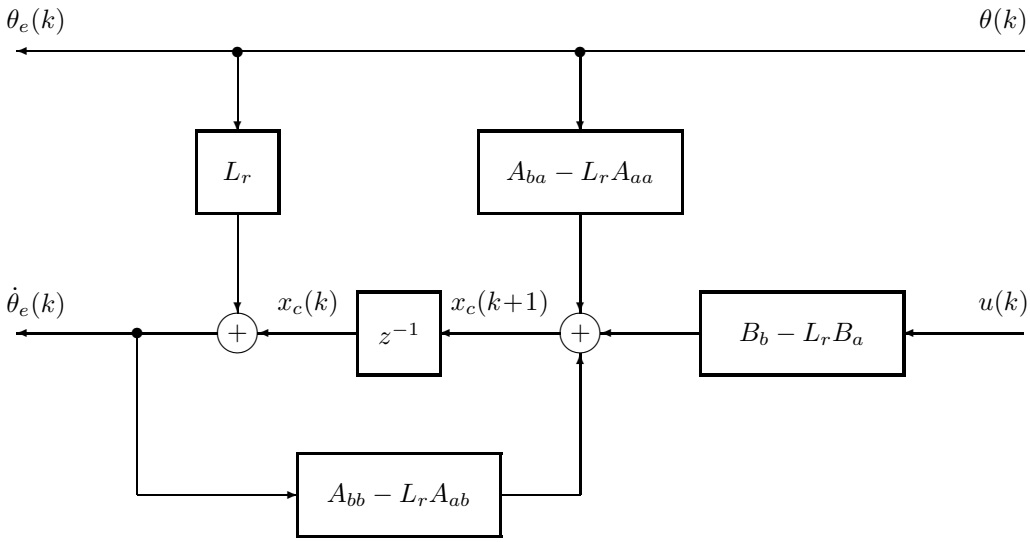
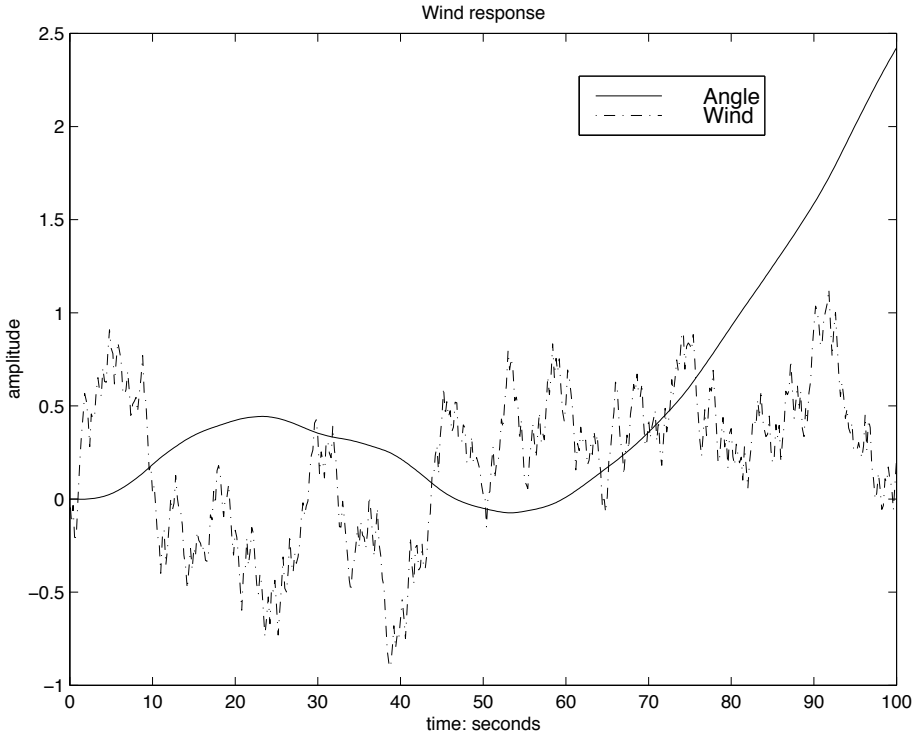


Figure 13: Structure of the reduced order estimator: `estr`

8.3 Module 8: 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 8:       Observer based design.
%
%       ------------------------------------------------

% This module will expand on module 7 to cover observers
% and reduced order observers.  Refer to Chapter 8 of the
% text for details of these methods.

% Adjust the Command and Graphics windows to your preferred
% sizes.

pause   % press any key to continue.

% We will start with the second order system studied
% in module 7.  The state was defined as,
%
%        x = [theta ; theta'].
%
% The state equations were,
%
%              | 0     1 |        |  0  |         |  0  |
%       x'  =  |         | x   +  |     | Td  +   |     | Tc
%              | 0  -B/J |        | 1/J |         | 1/J |
%
% with B = 5 and J = 100.

J = 100;
B = 5;

A = [0, 1; 0, -B/J];
B = [0, 0; 1/J, 1/J];

pause   % press any key to continue.

% In this case we will assume that we can measure only
% the angle theta.  Therefore C and D are,
```

```
C = [1, 0];
D = [0, 0];

% This system will be referred to as sat_dish.

sat_dish = ss(A,B,C,D);
set(sat_dish,'Notes','Satellite dish');
set(sat_dish,'InputName',{'Wind';'Control actuation'});
set(sat_dish,'OutputName','Angle');

pause   % press any key to continue.

% We will again create a wind disturbance signal for
% comparison purposes later.  Refer to module 7 for
% the discussion about how this signal is formed.

t = [0:0.2:100]';
w = rand(length(t),1) - 0.5;
lpfilter = tf(20,[10,1]);
wind = lsim(lpfilter,w,t);
clear w lpfilter

dist = [wind,0*wind];     % this is the full input.
wind_resp = lsim(sat_dish,dist,t);
theta = wind_resp(:,1);

plot(t,theta,'-',t,wind,'-.')
title('Wind response')
legend(char(get(sat_dish,'OutputName')),'Wind')
xlabel('time: seconds')
ylabel('amplitude')
```

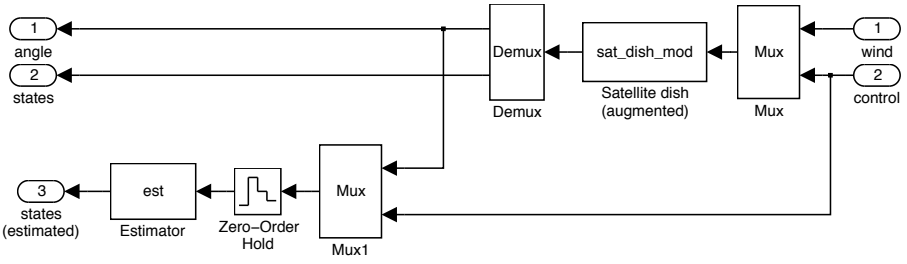



```
% The zero-order hold system is formed.
T = 1;                  % 1 second sample period.
sat_dish_Z = c2d(sat_dish,T,'zoh');
set(sat_dish_Z,'Notes','ZOH equivalent of the satellite dish');

% We already know that this system is controllable from
% the Tc input.  Now let's check and see if it is
% observable from the output: theta.

[Ad,Bd,Cd,Dd] = ssdata(sat_dish_Z)

Ad =

    1.0000    0.9754
         0    0.9512


Bd =

    0.0049    0.0049
    0.0098    0.0098
```

```
Cd =

     1     0


Dd =

     0     0

Omat = [Cd; Cd*Ad];

size(Omat)

ans =

     2     2


rank(Omat)

ans =

     2


% The observability matrix is full rank.  We can therefore
% proceed with an observer design.  Here we will do a
% predictor estimate (as opposed to a current estimator).

pause   % press any key to continue.

% Now to choose some pole locations.  Lets try z = 0.1 and
% z = 0.5.  Again Ackermann's formula can be used.
% The polynomial that gives these root locations is worked
% out as,
%
%       (z - 0.1)(z - 0.5) = 0
%
%       z^2 - 0.6z + 0.05 = 0.
%
% So, using the notation in the book,
%
%        alphae(Ad) = Ad^2 - 0.6*Ad + 0.05*I

alphae = Ad^2 - 0.6*Ad + 0.05*eye(2);

% and L can now be calculated:
```

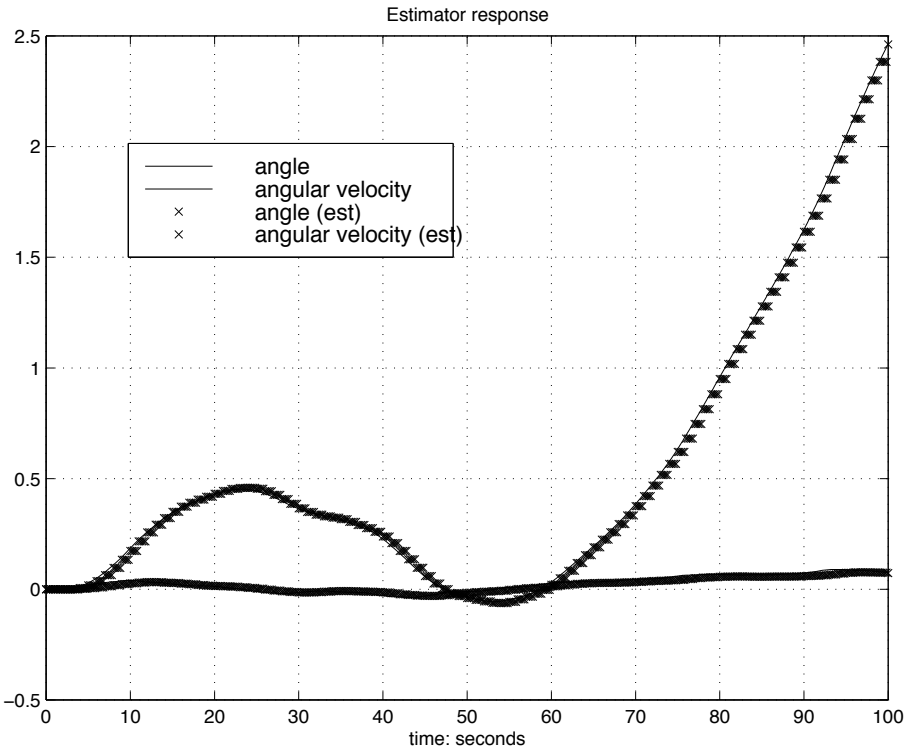
```
L = alphae*inv(Omat)*[0;1]

L =

    1.3512
    0.3938


pause   % press any key to continue

% We will check where the poles of the error system are.

eig(Ad - L*Cd)

ans =

    0.1000
    0.5000


% Good.  Exactly where we want them to be.

pause   % press any key to continue.

% Let's build the estimator and compare it the
% actual system for our wind disturbance.  There are
% several things to note here.  The estimator has
% two inputs - the control input to the system (Tc)
% and the system output (theta).  Initially we will
% simulate this with no control input.  I.e. the wind
% will drive the state of the system causing the error
% between the actual state and the estimated state.

pause   % press any key to continue.

% The estimator is given by,
%
%      xe(k+1) = (Ad - L*Cd) xe(k) + L*sat_dish_Z(k) + BdTc*Tc(k),
%
% where xe is the estimated state and sat_dish_Z(k) is the
% measured output of sat_dish_Z.  The outputs of interest are
% the estimated state so Ce is the identity.
% Ok - form this system.

Ae = Ad - L*Cd;
BdTc = Bd(:,2);
Be = [L,BdTc];
Ce = eye(2);
```

```
De = zeros(2,2);

est = ss(Ae,Be,Ce,De,T);
set(est,'Notes','Estimator');
set(est,'InputName',{'Angle measurement';'Control actuation'});
set(est,'OutputName',{'Angle (estimate)';'Angular velocity (estimate)'});

pause   % press any key to continue.

% OK now to simulate this.  This is again tricky - we
% have to think very carefully about what it is we want
% to look at.

% First, consider the plant sat_dish.  It has a single
% output.  For the purposes of this simulation we would
% also like to look at the states of sat_dish.  To do this
% define the two states as two new outputs.  Attaching
% an identity to C will do this.

Cmod = [C; eye(2)];
Dmod = [D; zeros(2,2)];
sat_dish_mod = ss(A,B,Cmod,Dmod);
set(sat_dish_mod,'Notes','Augmented satellite dish');
set(sat_dish_mod,'OutputName',{'Angle';'Angle';'Angular velocity'});
set(sat_dish_mod,'InputName',{'Wind';'Control actuation'});

pause   % press any key to continue.

% Check out the performance of this estimator via a simulation.

m8est
```



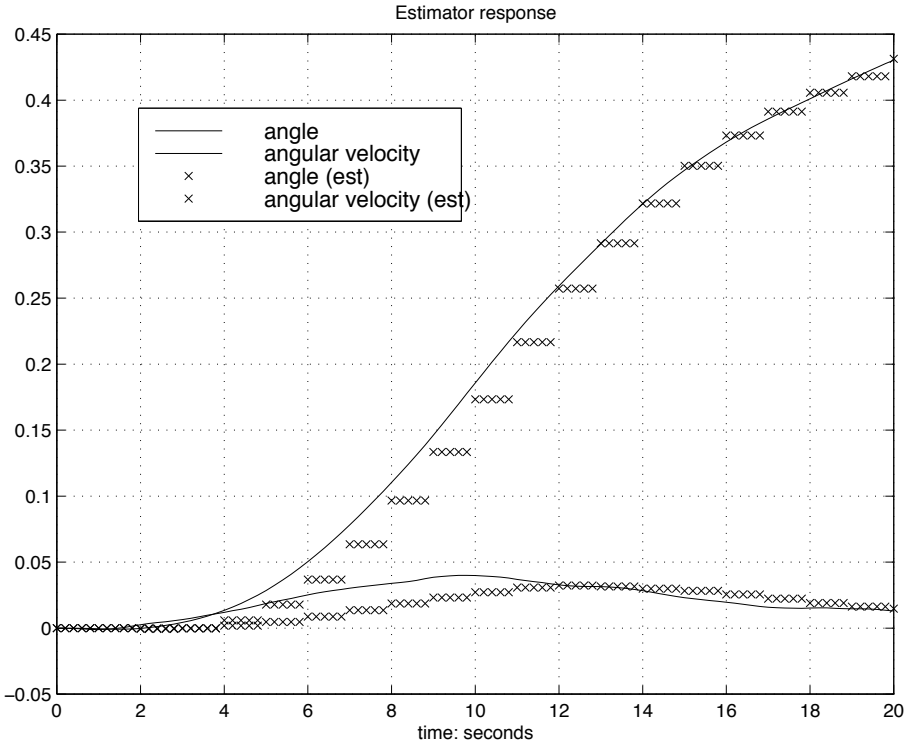

```
% The two inputs are wind and control.  The first two outputs
% are the state of sat_dish_Z.  The second two are the estimated
% state.
```

```
pause   % Press any key to continue.

% Now do the simulation.  Note that the control input is
% zero because we are doing no feedback here.

options = [];
[tout,x,yout] = sim('m8est',t,options,[t,dist]);

x = yout(:,[2:3]);
xe = yout(:,[4:5]);

plot(tout,x,'-',tout,xe,'x')
title('Estimator response')
legend('angle','angular velocity','angle (est)','angular velocity (est)')
xlabel('time: seconds')
grid
```



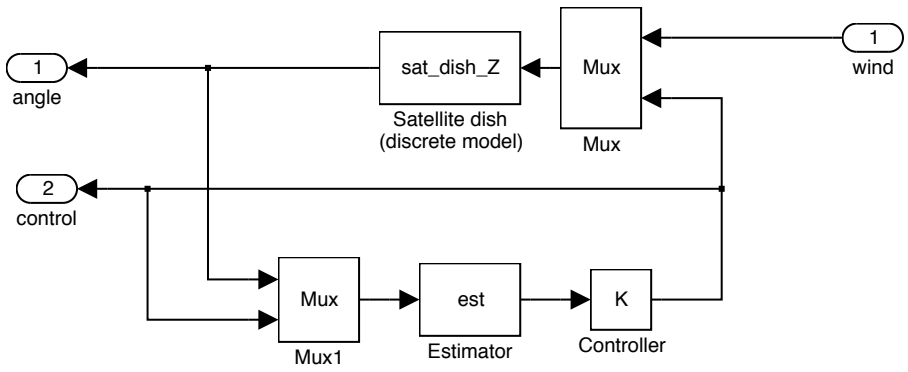

```
% This looks like a very good estimate.  Let's study the
% first twenty seconds to get a better idea of what is
% going on.  Note which of the two responses is theta.
```

```
pause   % press any key to continue.

tsidx = find(tout <= 20);
ts = tout(tsidx);
xs = x(tsidx,:);
xes = xe(tsidx,:);

plot(ts,xs,'-',ts,xes,'x')
title('Estimator response')
legend('angle','angular velocity','angle (est)','angular velocity (est)')
xlabel('time: seconds')
grid
```



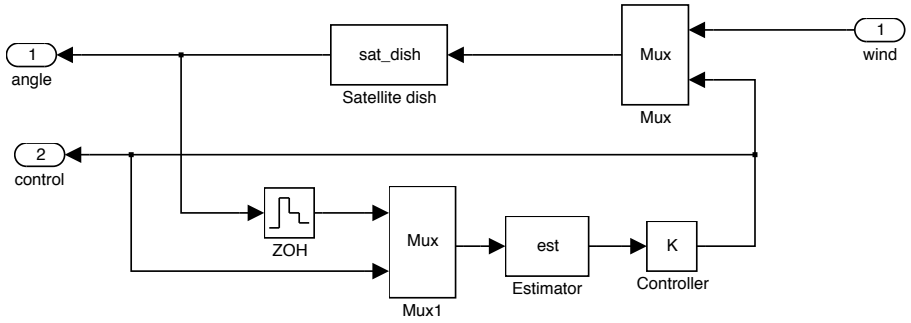

```
% The estimate of the angle is probably better than that
% of the angular velocity.  The discrete-time nature of the
% estimator is also obvious.

pause   % press any key to continue.

% We will now do a state-feedback design exactly as
% it was done in module 7.  The closed loop root locations
```

```
% will be 0.8 and 0.7.

BdTc = Bd(:,2);
Cmat = [BdTc, Ad*BdTc];
alphac = Ad^2 - 1.5*Ad + 0.56*eye(2);
K = [0, 1]*inv(Cmat)*alphac ;

% check this

eig(Ad - BdTc*K)

ans =

    0.8000
    0.7000


pause   % press any key to continue.

% Now to connect it all up.  As we won't be comparing
% the states, we will go back to using sat_dish as the plant.
% To connect it up we use -K to close the loop between the
% estimated states (xe) and the control input (Tc).

negK = -1*K;

%  We will first look at a completely discrete-time
%  version and check out the separation theorem.

m8disc
```

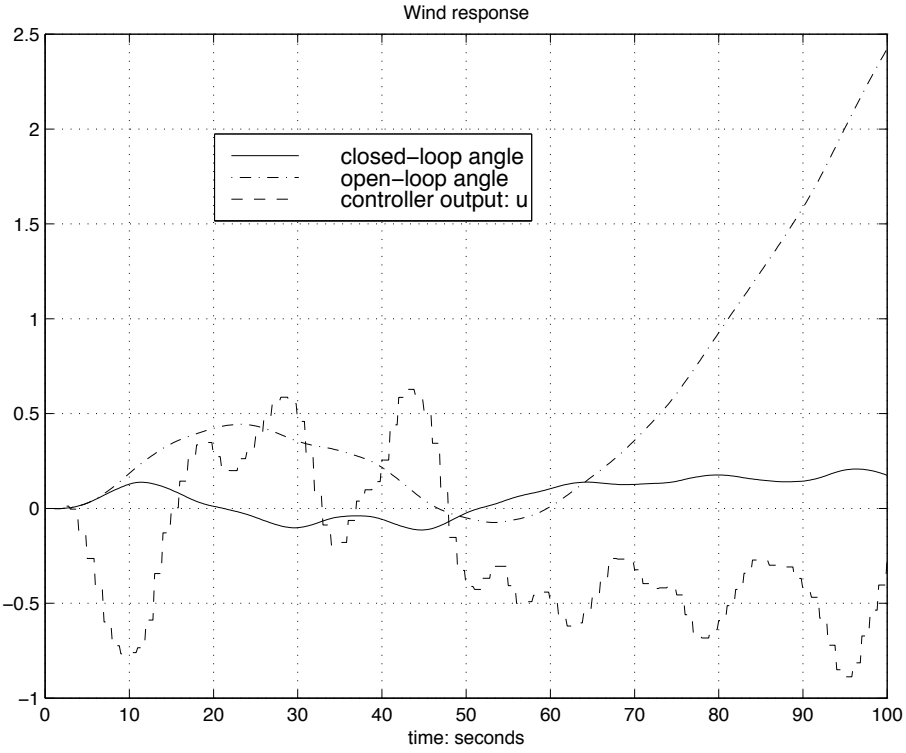



```
% Check the separation theorem.

[a1clp,b1clp,c1clp,d1clp] = dlinmod('m8disc',T);
Using variable-step discrete time solver for model 'm8disc'.
```

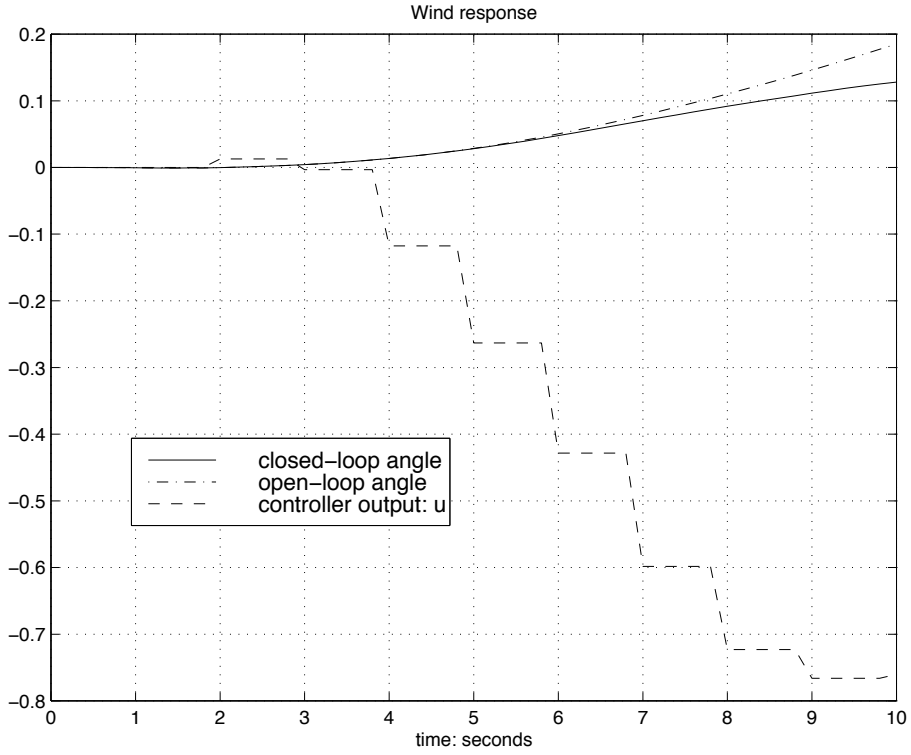
```
Using variable-step discrete time solver for model 'm8disc'.
Using variable-step discrete time solver for model 'm8disc'.
clp1 = ss(a1clp,b1clp,c1clp,d1clp);
clear a1clp b1clp c1clp d1clp

pole(clp1)

ans =

    0.1000
    0.7000
    0.8000
    0.5000


% It works - these are the design closed loop poles and
% the estimator error poles.

pause    % press any key to continue.

% At this point we could use m8disc or clp1 to do a discrete
% time simulation.  However, we might as well go straight
% to a sampled data simulation (continuous plant and discrete
% controller).

m8sd
```




```
[tclp,x,yclp] = sim('m8sd',t,options,[t,wind]);

plot(tclp,yclp(:,1),'-',t,theta,'-.',tclp,yclp(:,2),'--')
title('Wind response')
legend('closed-loop angle','open-loop angle','controller output: u')
xlabel('time: seconds')
grid
```



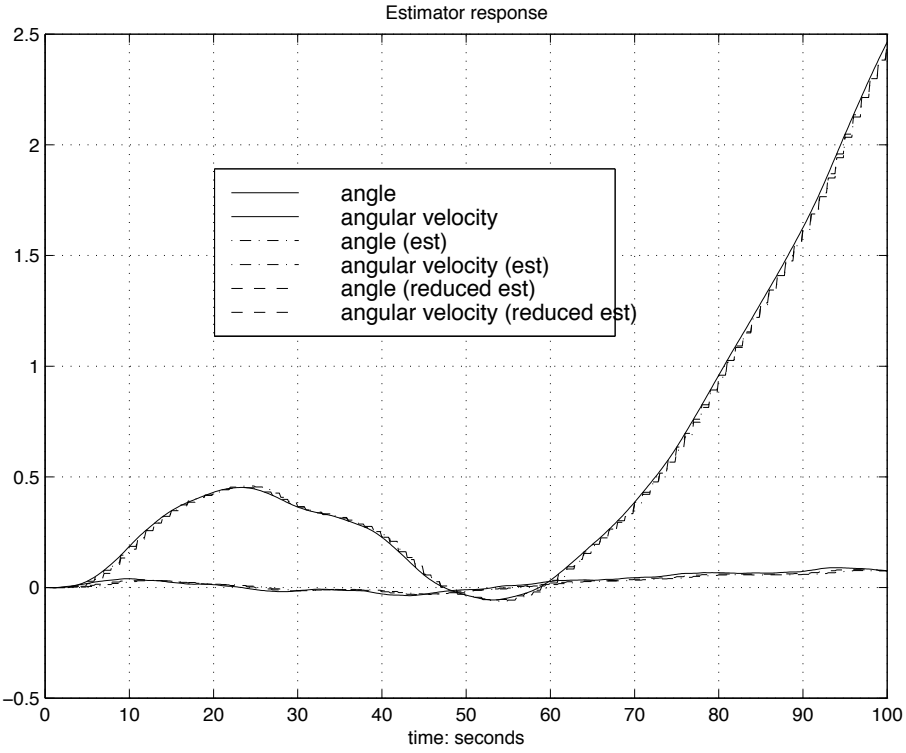

```
% Looks good - probably not quite as good as the result of
% module 7.  Recall though that we used a measurement of the
% angular velocity in that case.  Here we are estimating the
% angular velocity and estimating (filtering really) the
% angle.  In light of this we have also specified a less
% aggressive design than in module 7.

pause   % press any key to continue.

% For more detail let's look at the first ten seconds

toidx = find(tclp <= 10);
tidx = find(t <= 10);

plot(tclp(toidx),yclp(toidx,1),'-',t(tidx),theta(tidx),'-.',...
title('Wind response')
legend('closed-loop angle','open-loop angle','controller output: u')
xlabel('time: seconds')
grid
```



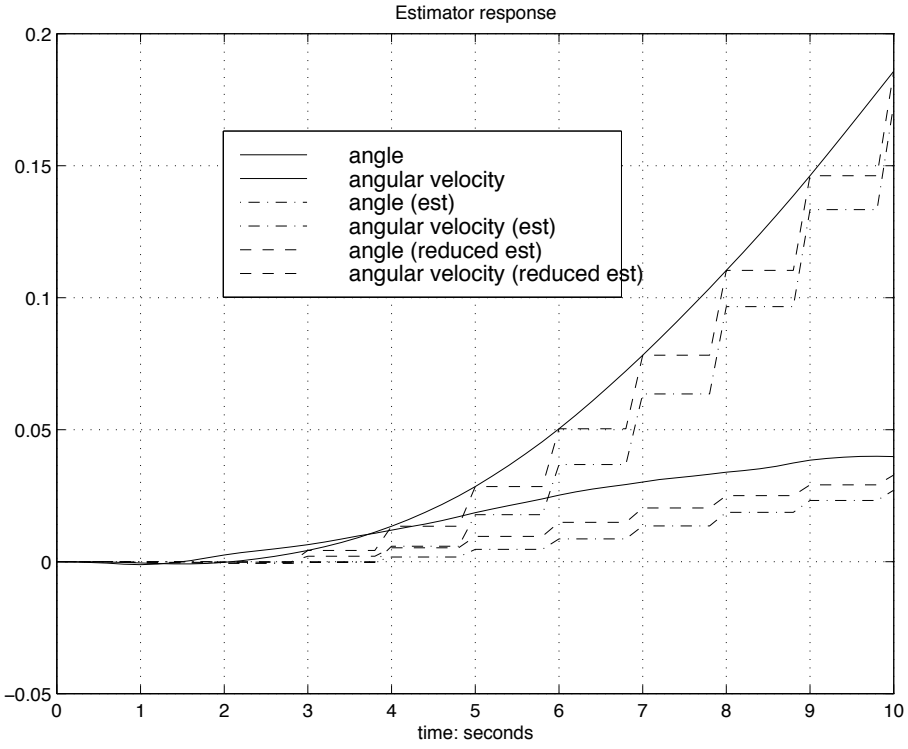

```
% Now we will look at a reduced order estimator for this
% problem.  We have a direct measurement of one of the
% states (angle), and we will use this to reconstruct
% the other state (angular velocity).

% We will maintain the notation in the notes so that
% the procedure can be followed.  It is possible to
% simplify this somewhat.

A_aa = Ad(1,1);
A_ab = Ad(1,2);
A_ba = Ad(2,1);
A_bb = Ad(2,2);

pause   % press any key to continue.

% Now we design error dynamics as though A_bb were the
% A matrix and A_ab were the output matrix.  To give
% a comparison with the above, choose z = 0.5 as the
% error dynamic pole.

alphae = A_bb - 0.5;
Omat = A_ab;
```

```
Lr = alphae*inv(Omat)

Lr =

    0.4626


% check this

eig(A_bb - Lr*A_ab)

ans =

    0.5000


pause   % press any key to continue.

% Now we can build the estimator.  To correspond to the
% notes x_a is the angle and x_b is the angular velocity.
% x_be will be the estimated version of x_b (angular velocity).
% B_a and B_b are obtained from the Tc -> angle and Tc -> angular
% velocity parts of the original Bd matrix.

B_a = BdTc(1);
B_b = BdTc(2);

% We also define x_c as an intermediate state in the system.
% The reduced order observer equations are:
%
% x_c(k+1) = (A_bb - Lr*A_ab)x_c(k)
%          +   ((A_bb - Lr*A_ab)*Lr + (A_ba - Lr*A_aa))*x_a(k)
%          +   (B_b - Lr*B_a) u(k)
%
% The estimated state (x_be(k)) is given by,
%
% x_be(k) = x_c(k) + Lr*x_a(k).

pause   % press any key to continue

% This is in state space form.  Create the reduced order
% estimator.  The first input is x_a(k) (theta) and the
% second is u(k).  The two outputs are x_a(k) and x_be(k)
% so that this estimator can be directly compared to the
% last.

Aer = A_bb - Lr*A_ab;
Ber = [((A_bb - Lr*A_ab)*Lr + A_ba - Lr*A_aa),...
```

```
Cer = [0; 1];
Der = [ 1, 0; ...
       Lr, 0];

estr = ss(Aer,Ber,Cer,Der,T);
pole(estr)

ans =

    0.5000


% Note that our estimator is only one state (reduced order) and
% it is stable.  It is a plug in replacement for our previous
% (full state) estimator except that it has only one state instead
% of two.

pause   % press any key to continue.

% OK.  Now let's connect it up and see what happens.
% To do this we will set

est = estr;

% and reuse the prior simulations.  We begin by comparing
% the reduced order and full order estimators.

m8est
[toutr,x,youtr] = sim('m8est',t,options,[t,dist]);
xr = youtr(:,[2:3]);
xer = youtr(:,[4:5]);

plot(toutr,xr,'-',tout,xe,'-.',toutr,xer,'--')
title('Estimator response')
legend('angle','angular velocity','angle (est)',...
xlabel('time: seconds')
grid
```

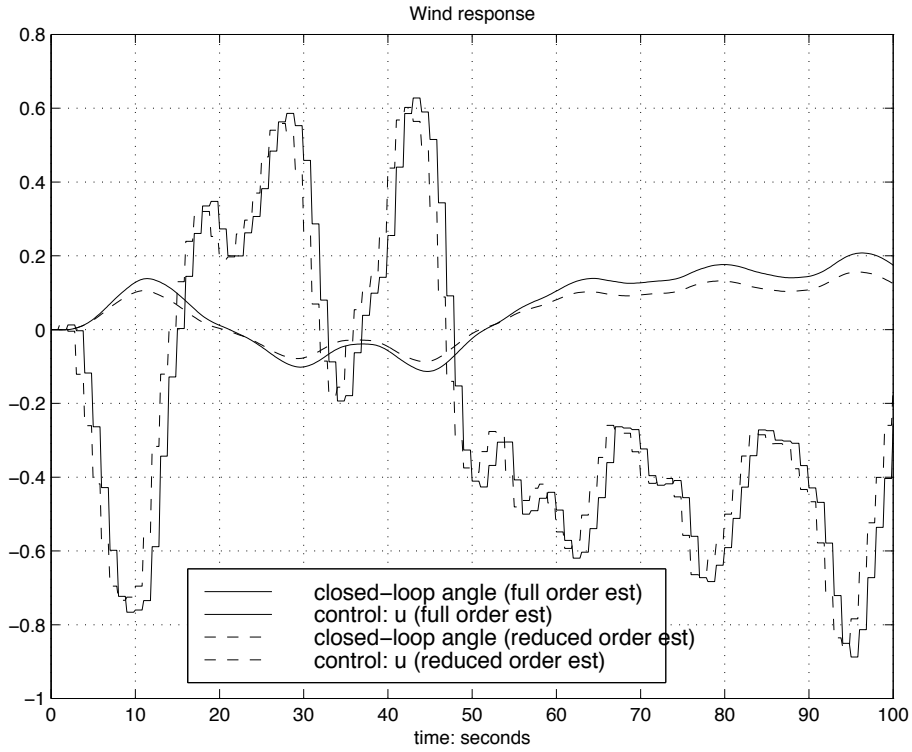



```
% This also looks like a very good estimate.  Let's study the
% first ten seconds to get a better idea of what is
% going on.  Note which of the two responses is theta.

pause   % press any key to continue.

tsidx = find(tout <= 10);
ts = tout(tsidx);
xes = xe(tsidx,:);
tsidxr = find(toutr <= 10);
trs = toutr(tsidxr);
xrs = xr(tsidxr,:);
xers = xer(tsidxr,:);

plot(trs,xrs,'-',ts,xes,'-.',trs,xers,'--')
title('Estimator response')
legend('angle','angular velocity','angle (est)',...
xlabel('time: seconds')
grid
```



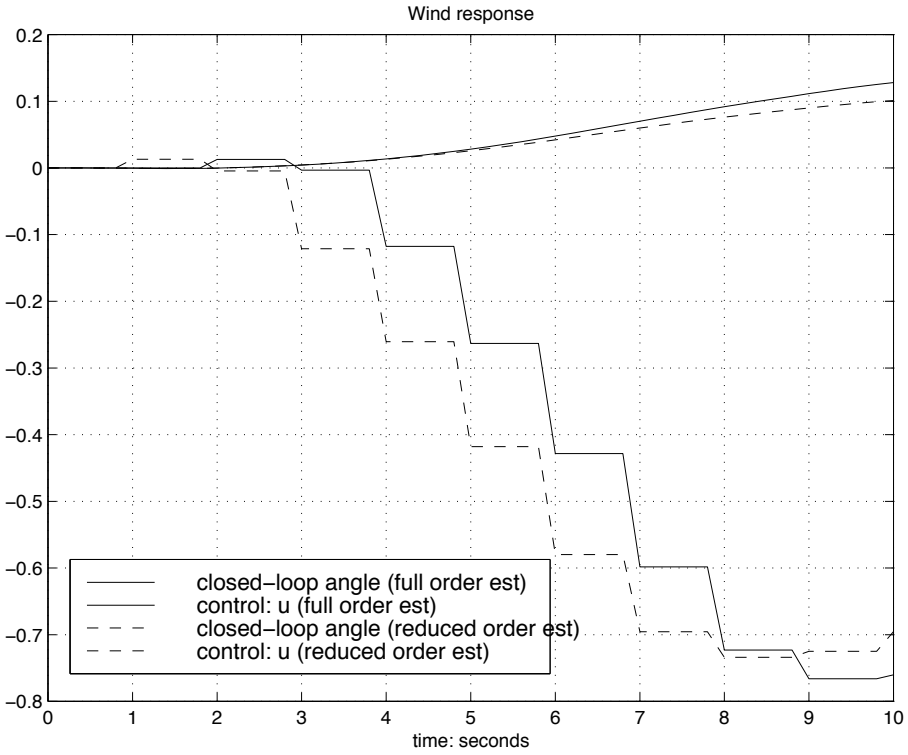

```
% For the angle state, the reduced order estimate is simply
% a sample and hold version of the actual state.  This is
% expected because we are measuring that state directly.
% The reduced order estimator may even be better than the full
% order one here although this is unlikely to be true if there
% is noise.

pause   % press any key to continue.
% Now try the closed-loop system with the reduced order
% estimator.  We will again check out the separation
% theorem with the discrete-time equivalent.

m8disc
clp2 = ss(a2clp,b2clp,c2clp,d2clp);
clear a2clp b2clp c2clp d2clp

pole(clp2)

ans =

    0.5000
    0.7000
    0.8000
```

```
% It works - these are the design closed loop poles and
% the reduced order estimator error pole.

pause   % press any key to continue.

% Now for the closed-loop sampled data system with the
% reduced order estimator.

m8sd
[tclp2,x,yclp2] = sim('m8sd',t,options,[t,wind]);

% We compare this with the full order estimator version.

plot(tclp,yclp,'-',tclp2,yclp2,'--')
title('Wind response')
legend('closed-loop angle (full order est)','control: u (full order est)',...
xlabel('time: seconds')
grid
```




```
% The reduced order estimator may be better here.  Again look
% at the first ten seconds to see what's happening in more
```

```
% detail.

pause   % press any key to continue.

tsidx1 = find(tclp <= 10);
ts1 = tclp(tsidx1);
yclp1s = yclp(tsidx1,:);
tsidx2 = find(tclp2 <= 10);
ts2 = tclp2(tsidx2);
yclp2s = yclp2(tsidx2,:);

plot(ts1,yclp1s,'-',ts2,yclp2s,'--')
title('Wind response')
legend('closed-loop angle (full order est)','control: u (full order est)',...
xlabel('time: seconds')
grid
```




```
% The reduced order estimator based design does give a better
% result in this case.  Note that the reduced order estimator
% seems to implement the control one time step ahead of the
% full order estimator.  The actual control implemented is
% similar.  Why is this?
```

```
%-------------------------------------------------------------
echo off
```

8.4 Problems

1. Modify the both the full state observer and full state control pole positions to improve the design. Try not to use significantly more control effort than used in the module 7 state feedback design.

2. Modify the sampled-data simulation to include noise on the measurement. The noise should be a random (zero mean) signal, about 10% of the size of the wind. Don't put it through a low pass filter, as I did for the wind signal.

   Modify both the full order and reduced order designs to get the best performance in the presence of this measurement noise.

3. Evaluate your best full order and reduced order designs for several values of T. Include noise in the simulations. As T is increased which controller performs better?

Bonus Question: This question is optional

4. Design, and tune, a full order estimator based controller which will also handle the reference tracking problem. Do a simulation for 20 degree step change in satellite dish angle. This simulation should also have noise and wind gusts as well.

Bonus Bonus Question: This question is also optional

5. Now use a reduced order estimator as the basis of a controller which handles reference tracking. Include integral control in this controller and simulate the result with a 20 degree step input, measurement noise and wind gusts.