ABDULLAH ALMOSALAMI

ECE 463

FINAL EXAM

05/08/2021

SATURDAY

You've got a Ball & Beam system w/ $m_{ball} = 0.5\,kg$, $J_{Beam} = 2\,kg\cdot m^2$. You are utilizing sensors for measuring position $x(t)$ of the ball and $\theta(t)$ of the beam. There is input noise $n_u \sim N(0, 0.02^2)$ and noise on the sensors $n_r \sim N(0, 0.01^2)$ and $n_\theta \sim N(0, 0.03^2)$, as well as an offset in the angle $d = 0.1\,rad$ ($\sim 5.7°$). The input noise only comes through in the angular velocity state $\rightarrow F = [0;0;0;0.4]$.

$$s\begin{bmatrix} r \\ \theta \\ \dot{r} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -7 & 0 & 0 \\ -1.96 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} r \\ \theta \\ \dot{r} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.4 \end{bmatrix} T + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.4 \end{bmatrix} n_u$$

$$Y_r = r + n_r, \quad Y_\theta = \theta + n_\theta + d$$

1) Design a feedback control law s.t. the position (x) has no error for step input, 2% $T_s = 8s$, $OS \leq 5\%$.

Well, utilizing servo-compensation: I'll use LQR techniques to get K_x and K_z.

Desired Poles: 2% T_s of 8s $\rightarrow \sigma = -0.5$ Hz, 5% OS $\rightarrow \omega_d \leq 0.5243$ rad/s (i.e. $\zeta \geq 0.6901$)

Dom. Poles $\boxed{-0.5 \pm j0.5243}$

(attached at end)

I will use

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

, which means Z is 2x1 and so is Ref

$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$

This means K_z needs to be 1x2.

I will opt to use LQR techniques to find K_x and K_z.

Let me recall first...

$$\left.\begin{aligned} sX &= AX + BU \\ sZ &= CX - Ref \end{aligned}\right\} \rightarrow \underbrace{s\begin{bmatrix} X \\ Z \end{bmatrix} = \begin{bmatrix} A & 0 \\ C & 0 \end{bmatrix}\begin{bmatrix} X \\ Z \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u}_{} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} Ref$$

* Ok, so system is NOT controllable if I have Z be 2x1 and C be [1,0,0,0; 0,1,0,0] ...

So hum. I'll go for $C_z = [1, 0, 0, 0]$ for the servo-compensator $\Rightarrow$ For the observer, I'll use C = [1, 0, 0, 0; 0, 1, 0, 0]...

So using that, I go to my lqr_test_script and err_from_Q_a_b scripts to obtain:

(Attached at end)

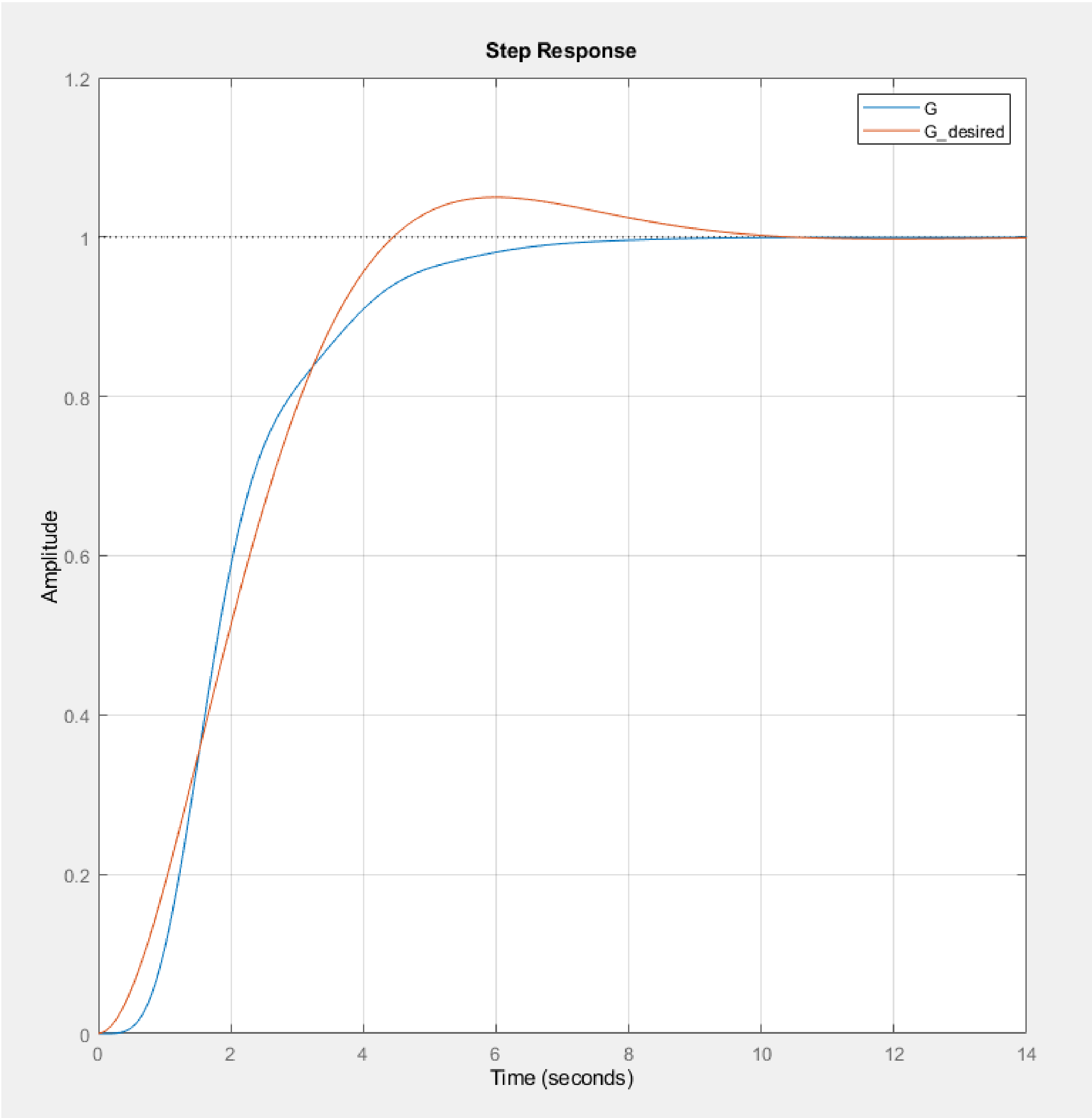
$$Q = \begin{bmatrix} 677.5260 & \cdots & 0 \\ \vdots & 0 & \cdots \\ & \cdots & 422.9956 \end{bmatrix}, \quad R = 1 \Rightarrow \|Y - Y_d\|^2 = 1.7058 \text{ over } 15 \text{ sec}$$

$$\boxed{\begin{aligned} K_x &= -49.2998,\ 84.6513,\ -30.8701,\ 20.5731 \\ K_z &= -20.5669 \end{aligned}}$$

```
>> clear all
>> [X,fval] = fminsearch(@(X) err_from_Q_a_b(X),[1,1])

X =

  677.5260  422.9956


fval =

    1.7058

>> lqr_test_script

Q =

  677.5260         0         0         0         0
         0         0         0         0         0
         0         0         0         0         0
         0         0         0         0         0
         0         0         0         0  422.9956


Kx =

  -49.2998   84.6513  -30.8701   20.5731


Kz =

  -20.5669

>>
```

```
function [err] = err_from_Q_a_b(X)
a = X(1); b = X(2);

if a<0 || b<0
    err = 1e3;
    return
end

% mc = 1; ml = 4; L = 1;
% [A,B] = linearizedCartPend(mc,ml,L);
mball = 0.5; Rball=1; mbeam = 4.8990; L = 2.4746;
[A,B] = linearizedBeamBall(mball,Rball,mbeam,L);
C = [1 0 0 0];
Aaug = [A, zeros(4,1); C, 0]; Baug = [B;0]; Caug = [C,0];
Cz = [0 0 0 0 1];
Qx = Caug'*Caug;
Qz = Cz'*Cz;

Q = a*Qx + b*Qz;
R = 1;

[K,S,CLP] = lqr(Aaug,Baug,Q,R);

Kx = K(1:4); Kz = K(5);
Acl = [A-B*Kx, -B*Kz; C, 0]; Bcl = [zeros(4,1); -1]; Ccl = Caug; Dcl = 0;
G = ss(Acl, Bcl, Ccl, Dcl);

des_dom_poles = [-0.5+0.5243j, -0.5-0.5243j];
num = abs(des_dom_poles(1))^2;
den = poly(des_dom_poles);
Gd = tf(num,den);

Tend = 15;
err = resp_err(G,Gd,Tend);

end
```

```matlab
%% Desired response
des_dom_poles = [-0.5+0.5243j, -0.5-0.5243j];
num = abs(des_dom_poles(1))^2;
den = poly(des_dom_poles);
Gd = tf(num,den);

%% Current system
mball = 0.5; Rball=1; mbeam = 4.8990; L = 2.4746;
[A,B] = linearizedBeamBall(mball,Rball,mbeam,L);
C = [1 0 0 0];
Aaug = [A, zeros(4,1); C, 0]; Baug = [B;0]; Caug = [C,0];
Cz = [0 0 0 0 1];

%% Check controllability
% OO = [Baug, Aaug*Baug, Aaug^2*Baug, Aaug^3*Baug, Aaug^4*Baug, Aaug^5*Baug];
% rankOO = rank(OO)
% return

%% Observer
Qx = Caug'*Caug;
Qz = Cz'*Cz;
a = 677.5260; b = 422.9956;
Q = a*Qx + b*Qz
R = 1;

[K,S,CLP] = lqr(Aaug,Baug,Q,R);

Kx = K(1:4)
Kz = K(5)
% Acl = [A-B*Kx, -B*Kz; C, 0]; Bcl = [zeros(4,1); -1]; Ccl = Caug; Dcl = 0;


%% Test
G = ss(Acl, Bcl, Ccl, Dcl);
step(G)
hold on;
step(Gd)
legend('G','G_{desired}');
grid on;

resp_err(G,Gd,15)
```


Step Response
G
G_desired
Amplitude
Time (seconds)
0
0.2
0.4
0.6
0.8
1
1.2
2
4
6
8
10
12
14

② Design a Kalman Filter to estimate the states and the constant disturbance using both position and angle.

Firstly, since we have an output disturbance offset, I need a state to account for this in my observer.

$$\Longrightarrow \quad A_s = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}, \quad B_s = \begin{bmatrix} B \\ 0 \end{bmatrix}, \quad C_s = \begin{bmatrix} C_r & 0 \\ C_\theta & 1 \end{bmatrix} \begin{matrix} \leftarrow r \\ \leftarrow \theta \end{matrix}$$

Now for this system, I make a Kalman Filter to observe the states + disturbance.

$$Q = B(0.02^2)B^T \qquad R = \mathrm{diag}([0.01^2, 0.03^2])$$

$\downarrow$

I had to add 1e-5 * eye(5) for LQR to work

$\downarrow$

$$Q = 1e{-4} * \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0.74 & 0 \\ 0 & 0 & 0 & 0 & 0.1 \end{bmatrix}, \quad R = 1e{-3} * \begin{bmatrix} 0.1 & 0 \\ 0 & 0.9 \end{bmatrix}$$

$$H: \begin{array}{cc} r & \theta \\ 4.2513 & -0.2425 \\ -2.3155 & 0.2082 \\ 9.2513 & -0.6711 \\ -3.9025 & 0.2997 \\ 0.1328 & 0.0957 \end{array}$$

$\Rightarrow$ Observer poles : -0.1054, $-1.9562 \pm j0.1278$, $-0.2687 \pm j1.9563$

$\Downarrow$

2% T_s = 40 seconds ...

Yikes !

③ Do the linear simulation /w $\alpha = 0.1$ rad ($\sim 5.7°$)

Ok there's a few things to watch for...

Includes disturbance state

$$sX = AX + BU + B_{nu} n_u$$

$$s\hat{X} = A\hat{X} + BU + H(Y - \hat{Y})$$

$+\begin{bmatrix} n_r \\ n_e \end{bmatrix} = n_y$

$$H(Y - \hat{Y}) = H(CX - C\hat{X}) = HC(X - \hat{X}) + Hn_y$$

$$= (A - HC)\hat{X} + BU + (HC)X + Hn_y$$

$$sZ = \cancel{Y}\,\hat{x} - Ref = C_z\hat{X} - Ref \leftarrow = 1$$

$\cancel{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}$

So,

$$s\begin{bmatrix} X \\ \hat{X} \\ Z \end{bmatrix} = \begin{bmatrix} A & 0 & 0 \\ HC & A-HC & 0 \\ 0 & C_z & 0 \end{bmatrix}\begin{bmatrix} X \\ \hat{X} \\ Z \end{bmatrix} + \begin{bmatrix} B_{nu} \\ 0 \\ 0 \end{bmatrix} n_u + \begin{bmatrix} 0 \\ H \\ 0 \end{bmatrix} n_y + \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} Ref + \begin{bmatrix} B \\ B \\ 0 \end{bmatrix} U$$

5×1, 5×1, 1×1; 5×1, 5×1, 1×1, 1×1; 5×2, 5×2, 2×1, 1×2; 5×2, 5×2, 1×2, $\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$; 5×1, 5×1, 1×1

Open Loop

$$s\begin{bmatrix} X \\ \hat{X} \\ Z \end{bmatrix} = \begin{bmatrix} A & 0 & 0 \\ HC & A-HC & 0 \\ 0 & C_z & 0 \end{bmatrix}\begin{bmatrix} X \\ \hat{X} \\ Z \end{bmatrix} + \begin{array}{c} \begin{array}{cccccc} u & n_u & n_r & n_e & Ref \end{array} \\ \begin{bmatrix} B & B_{nu} & 0 & 0 & 0 \\ B & 0 & H_r & H_e & 0 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix} \end{array} \begin{bmatrix} u \\ n_u \\ n_r \\ n_e \\ Ref \end{bmatrix}$$

12×5   5×1

Closed Loop:

$$u = -K_z Z - K_x \hat{X}$$ not including disturbance state

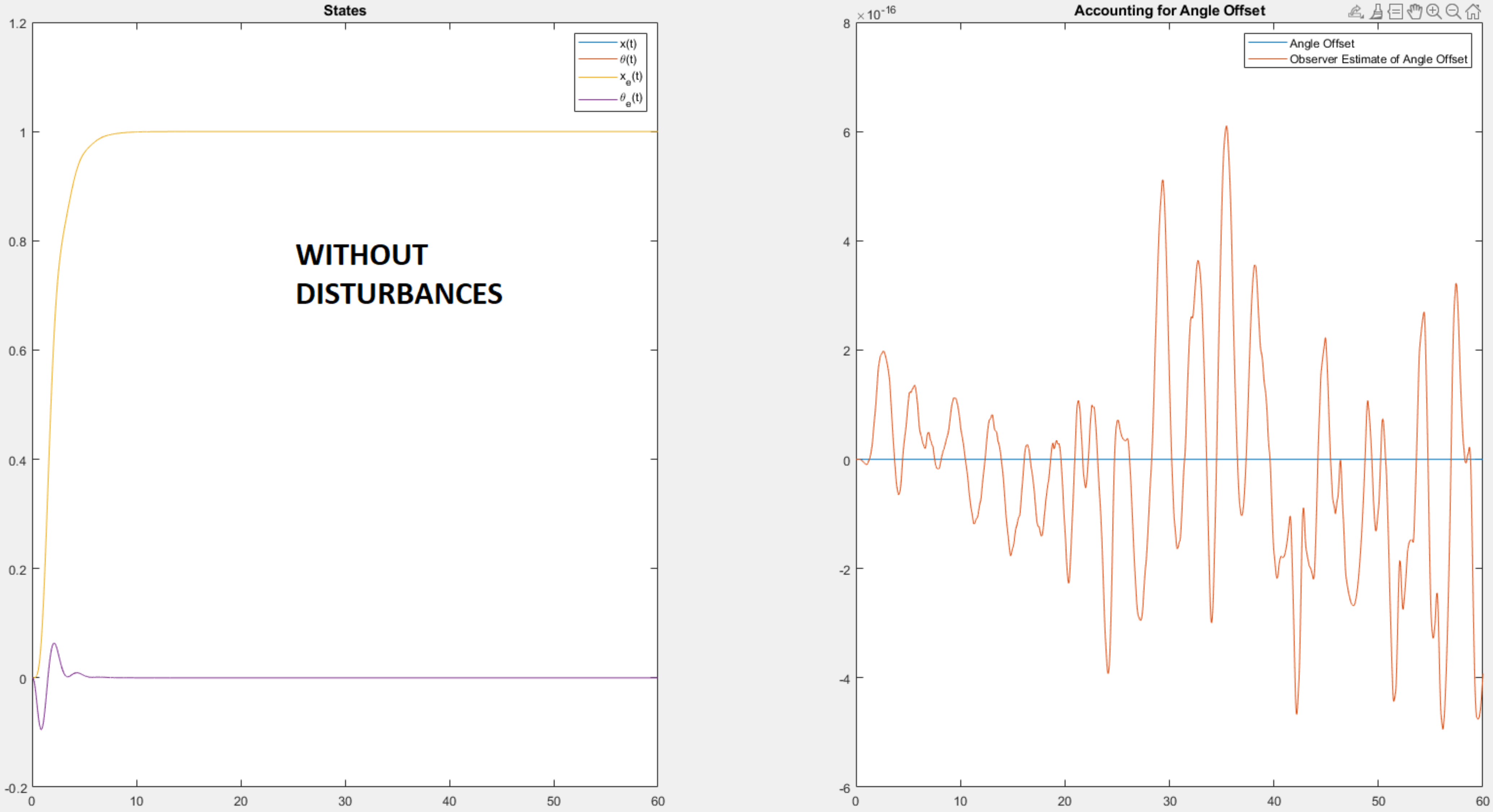
$$sX = AX - BK_zZ - BK_x\hat{X} + B_{nu}n_u = (A)X + (-BK_x)\hat{X} + (-BK_z)Z + B_{nu}n_u$$

$$s\hat{X} = A\hat{X} - BK_zZ - BK_x\hat{X} + HCX + Hn_y = (HC)X + (A - BK_x - HC)\hat{X} + (-BK_z)Z + Hn_y$$

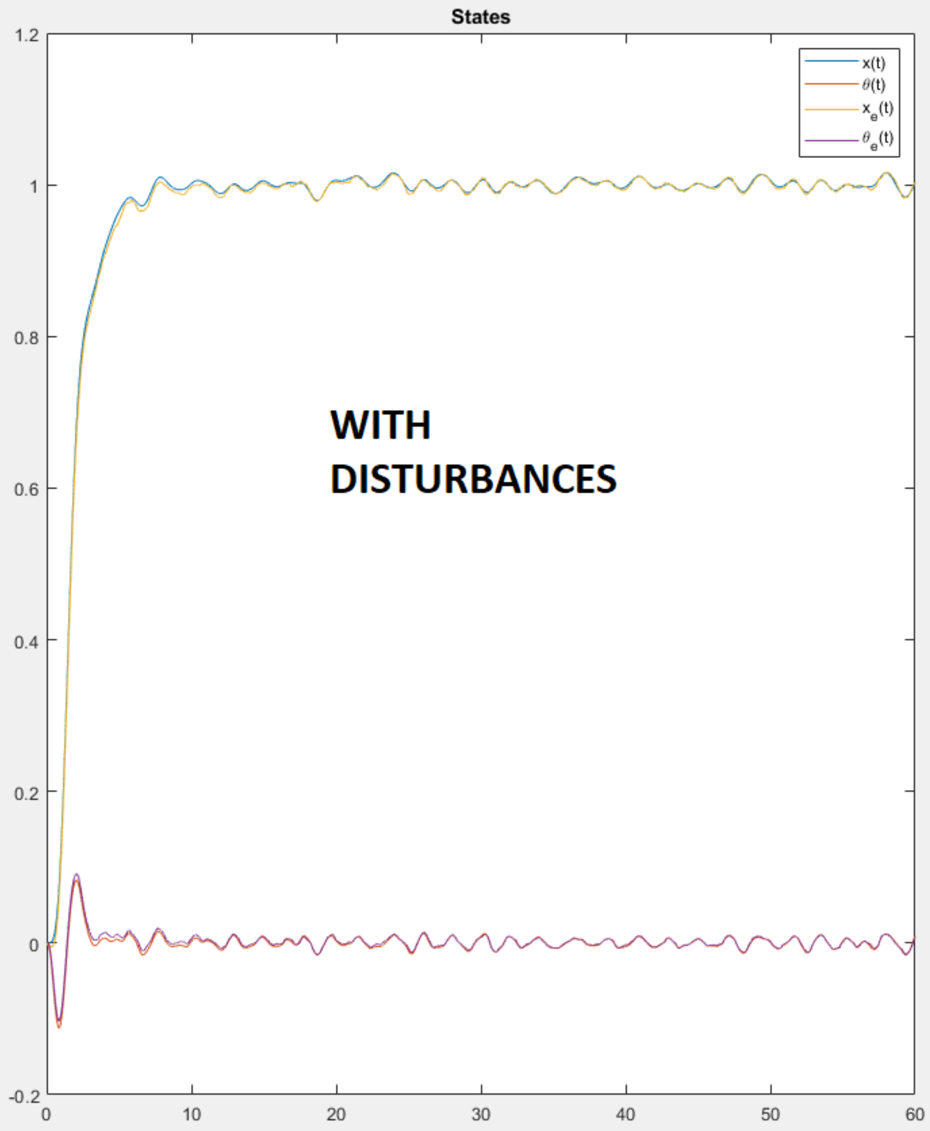
$$sZ = C_z\hat{X} - Ref$$

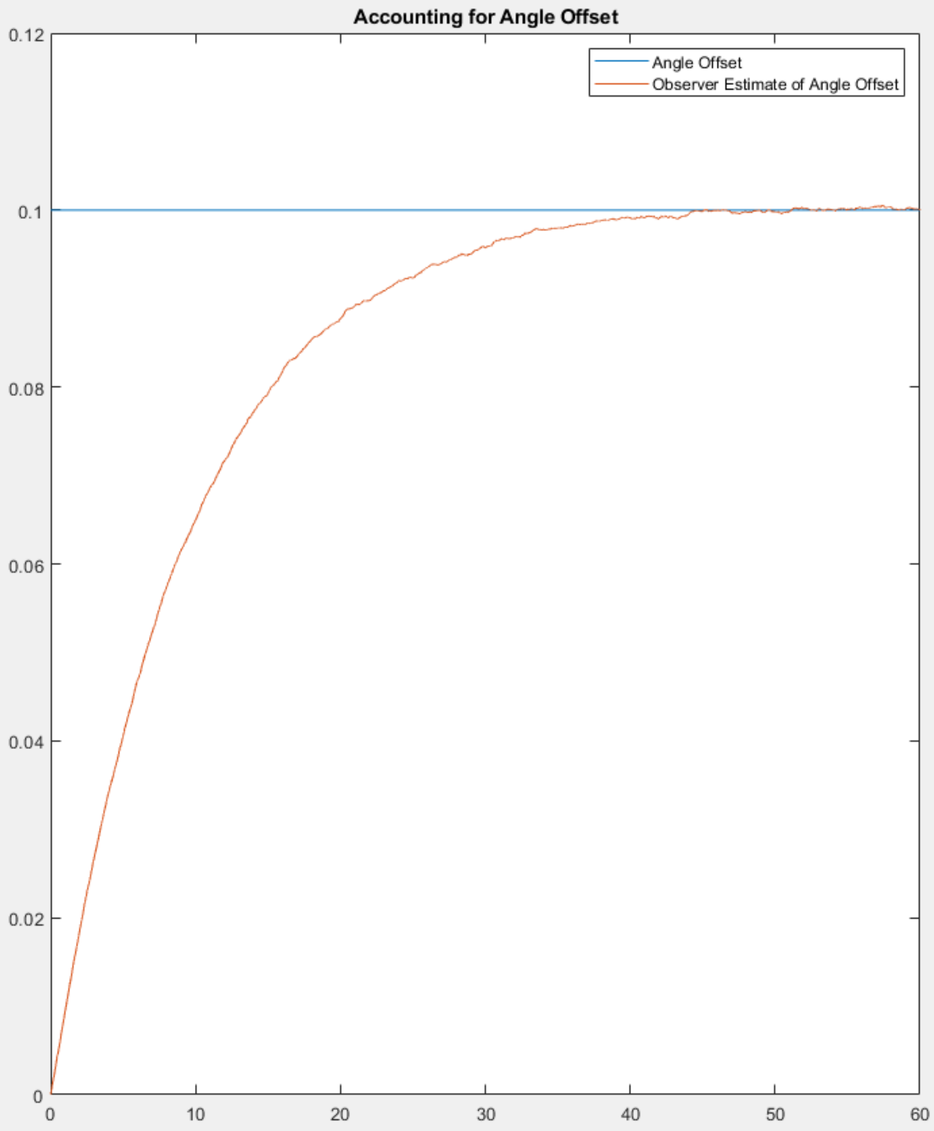
Closed Loop

$$s\begin{bmatrix} X \\ \hat{X} \\ Z \end{bmatrix} = \begin{bmatrix} A & -BK_x & -BK_z \\ HC & A-BK_x-HC & -BK_z \\ 0 & C_z & 0 \end{bmatrix}\begin{bmatrix} X \\ \hat{X} \\ Z \end{bmatrix} + \begin{bmatrix} B_{nu} & 0 & 0 & 0 \\ 0 & H_r & H_e & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}\begin{bmatrix} n_u \\ n_r \\ n_e \\ Ref \end{bmatrix}$$



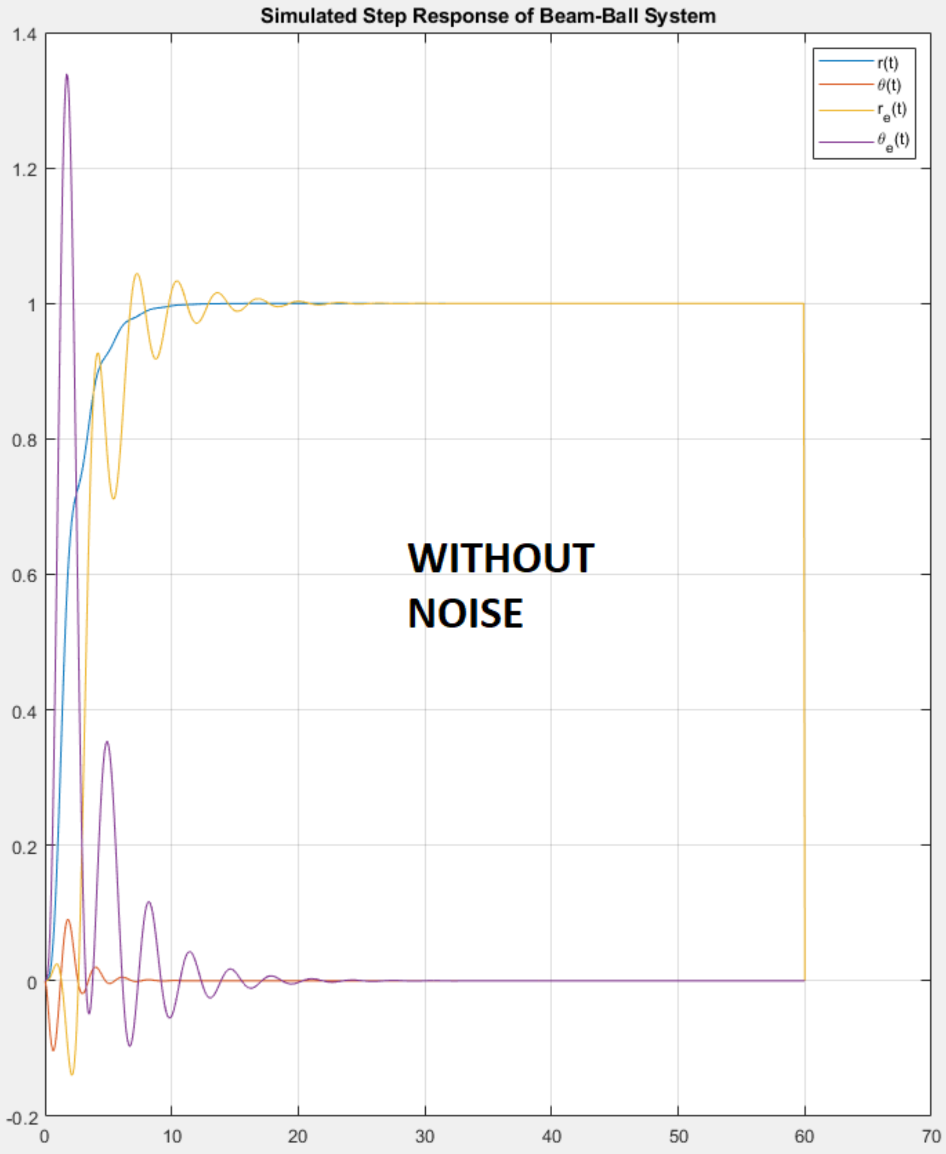
States
x(t)
θ(t)
x_e(t)
θ_e(t)
WITHOUT
DISTURBANCES
Accounting for Angle Offset
× 10^-16
Angle Offset
Observer Estimate of Angle Offset

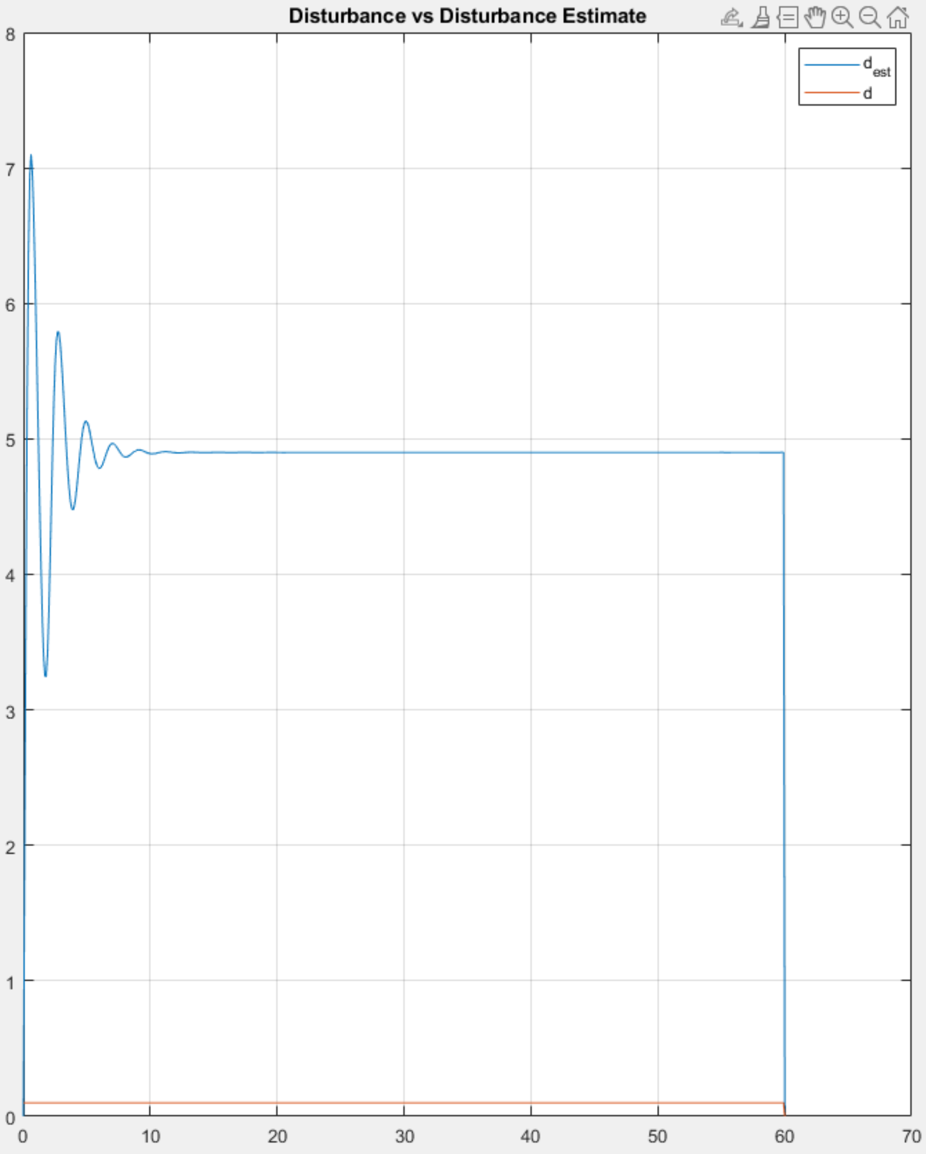

States
x(t)
θ(t)
x_e(t)
θ_e(t)
WITH
DISTURBANCES

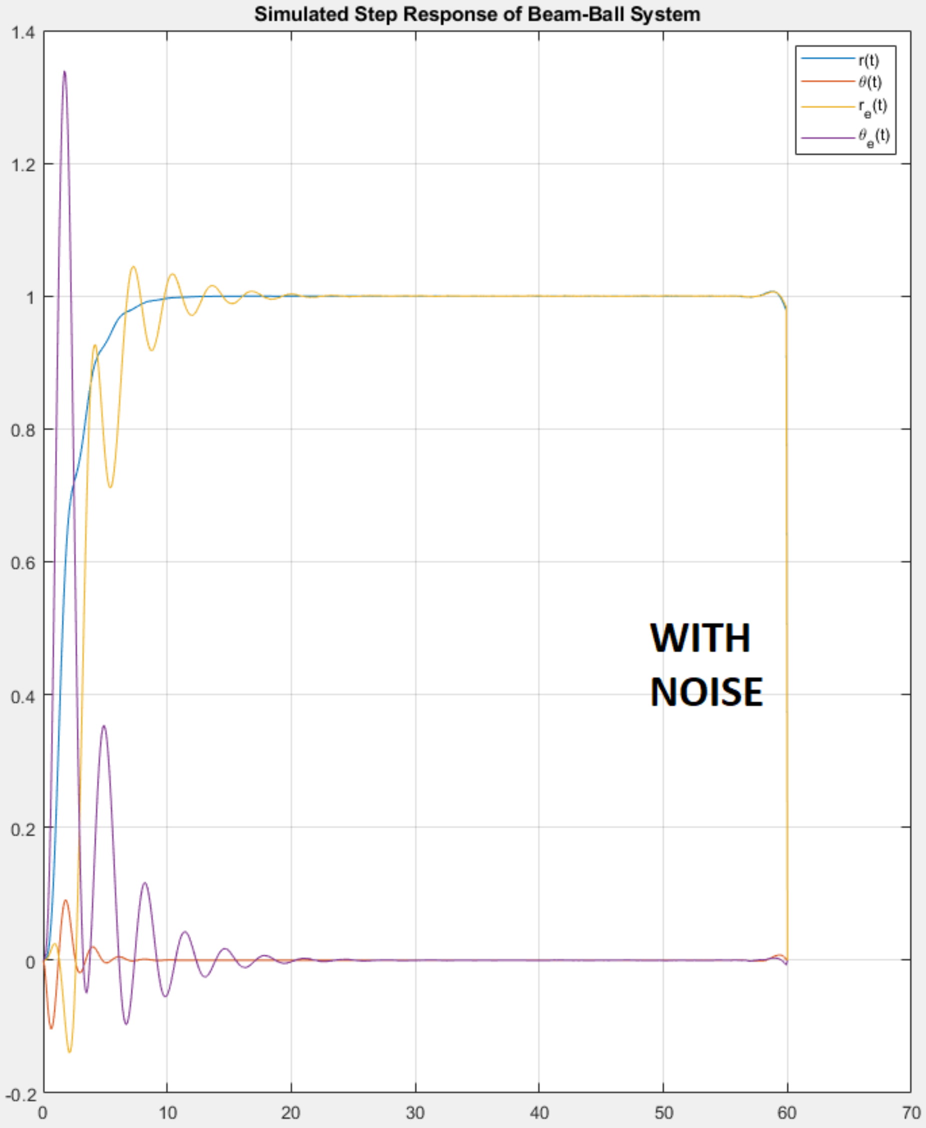

Accounting for Angle Offset
Angle Offset
Observer Estimate of Angle Offset

(4) Now nonlinear sim.

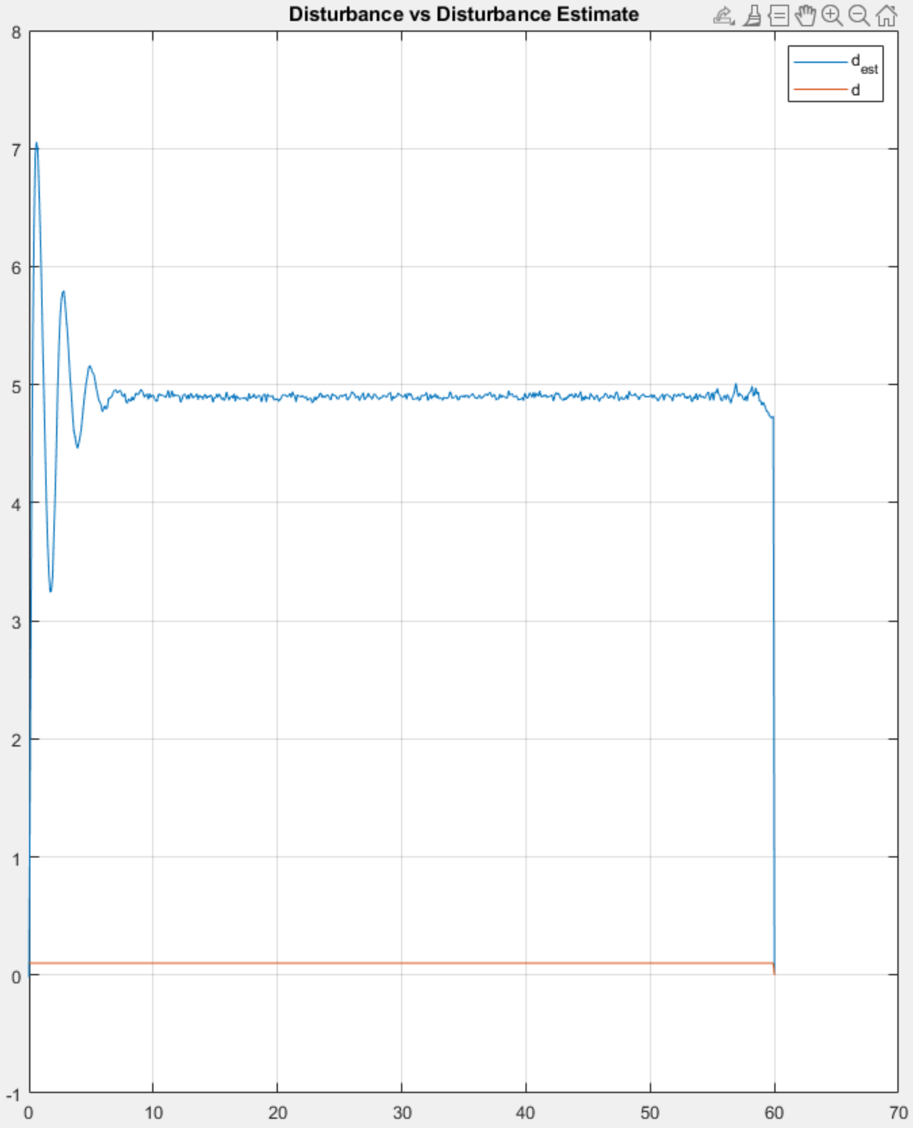
Because the Kalman Filter is so slow, I waited 60 sec before using it in the Control Law...


Simulated Step Response of Beam-Ball System
r(t)
θ(t)
r_e(t)
θ_e(t)
WITHOUT
NOISE
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
10
20
30
40
50
60
70


Disturbance vs Disturbance Estimate
d_{est}
d
8
7
6
5
4
3
2
1
0
0
10
20
30
40
50
60
70



Simulated Step Response of Beam-Ball System
r(t)
θ(t)
r_e(t)
θ_e(t)
WITH
NOISE
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
10
20
30
40
50
60
70



Disturbance vs Disturbance Estimate
d_est
d
8
7
6
5
4
3
2
1
0
-1
0
10
20
30
40
50
60
70

LINEAR SIMULATION CODE

```matlab
%% System description

% Ball & Beam
mball = 0.5; Rball = 1; mbeam = 4.8990; L = 2.4746;
[A,B] = linearizedBeamBall(mball,Rball,mbeam,L);
C = [1 0 0 0; 0 1 0 0]; D = [0;0];
Cz = [1 0 0 0 0]; % Servo-comp will only use x(t), not q(t)

% Disturbance
Aaug = [A, zeros(4,1); zeros(1,4), 0]; Baug = [B;0];
Caug = [C, [0;1]];
% return

%% Noise character
Bnu = [0;0;0;0.4;0];
nu_mu = 0; nu_sig = 0.02;
nr_mu = 0; nr_sig = 0.01;
nq_mu = 0; nq_sig = 0.03;


%% Observer
% Check observability
% OO = [Caug; Caug*Aaug; Caug*Aaug^2; Caug*Aaug^3; Caug*Aaug^4];
% rankOO = rank(OO)
% return

% Kalman filter
F = Bnu; V = nu_sig;
Q = F*V^2*F'; R = diag([nr_sig^2, nq_sig^2]);
Q = Q + 1e-5*eye(5);    % Because Q is too small
H = transpose(lqr(Aaug',Caug',Q,R));

% Check stability
eig(Aaug - H*Caug);

% return


%% Feedback Control Law
Kx = [-49.2998, 84.6513, -30.8701, 20.5731, 0]; % 0 for the dummy disturbance
state
Kz = -20.5669;

%% Linear simulation
% Afull = [Aaug, -Baug*Kx, zeros(5,2); H*Caug, Aaug-H*Caug, zeros(5,2);
zeros(2,5), Caug, zeros(2,2)];
% Bfull = [Baug, Bnu, zeros(5,4); Baug, zeros(5,1), H, zeros(5,2);
zeros(2,4), diag([-1,-1])];
Afull = [Aaug, -Baug*Kx, -Baug*Kz; H*Caug, Aaug-Baug*Kx-H*Caug, -Baug*Kz;
zeros(1,5), Cz 0];
Bfull = [Bnu, zeros(5,3); zeros(5,1), H, zeros(5,1); zeros(1,3), -1];
Cfull = [1, zeros(1,10); ...
         0,1,zeros(1,9); ...
         zeros(1,5),1,0,0,0,0,0; ...
         zeros(1,6),1,0,0,0,0; ...
```

```
         zeros(1,4),1,zeros(1,6); ...
         zeros(1,9),1,0];
Dfull = zeros(6,4);
% return
% Dfull(1,3) = 1; Dfull(2,4) = 1; % To see measured outputs instead...

d = 0.1;    % 0.1 rad offset for angle measurement
% X0 = zeros(11,1);
X0 = zeros(11,1); X0(5) = d;
Tend = 60; t = transpose(linspace(0,Tend,10001)); N = size(t);
% U = [zeros(N), zeros(N), zeros(N), ones(N)];
U = [normrnd(nu_mu,nu_sig,N), normrnd(nr_mu,nr_sig,N),
normrnd(nq_mu,nq_sig,N), ones(N)];
Y = step3(Afull, Bfull, Cfull, Dfull, t, X0, U);

subplot(1,2,1);
plot(t,Y(:,[1:4]));
legend('x(t)','\theta(t)','x_e(t)','\theta_e(t)');
title('States');

subplot(1,2,2);
plot(t,Y(:,[5:6]));
legend('Angle Offset','Observer Estimate of Angle Offset');
title('Accounting for Angle Offset');
```

NONLINEAR SIMULATION CODE

```
% Ball & Beam System

% System
mball = 0.5; Rball = 1; mbeam = 4.8990; L = 2.4746;
[A,B] = linearizedBeamBall(mball,Rball,mbeam,L);
C = [1 0 0 0; 0 1 0 0];
Aaug = [A, zeros(4,1); zeros(1,4), 0]; Baug = [B;0];
Caug = [C, zeros(2,1)];

% Feedback gains
Kx = [-49.2998   84.6513  -30.8701   20.5731];
Kz = -20.5669;

% Observer gains
H = [4.2513,-0.2425; -2.3155,0.2082; 9.2513,-0.6711; -3.9025,0.2997;
0.1328,0.0957];

% Noise
nu_mu = 0; nu_sig = 0.02;
nr_mu = 0; nr_sig = 0.01;
nq_mu = 0; nq_sig = 0.03;

% Initial conditions
X = zeros(5,1); Xe = zeros(5,1); Z = 0;
d = 0.1;
dX = zeros(5,1); dXe = zeros(5,1);
X(5) = d; % Disturbance

% Simulation setup
Ref = 1;
t = 0; dt = 100e-6; Tend = 60;
N = (Tend / dt) + 1;
DATA = zeros(N,7); % x, th, xe, the, T, d, de

i=1;
tic
while(t < Tend)
    if t < 55
%         U = -Kz*Z - Kx*X([1:4]);
        U = -Kz*Z - Kx*X([1:4]) + normrnd(nu_mu,nu_sig);
        dZ = X(1) + normrnd(nr_mu,nr_sig) - Ref;
    else
%         U = -Kz*Z - Kx*Xe([1:4]);
        U = -Kz*Z - Kx*Xe([1:4]) + normrnd(nu_mu,nu_sig);
%         dZ = Xe(1) - Ref;
        dZ = Xe(1) + normrnd(nr_mu,nr_sig) - Ref;
    end

    dX([1:4]) = BeamDynamics(X([1:4]), U, mball, Rball, mbeam, L);
    dXe = Aaug*Xe + Baug*U + H*Caug*(X +
[normrnd(nr_mu,nr_sig);normrnd(nq_mu,nq_sig);0;0;0] - Xe);

    X = X + dX * dt;
    Xe = Xe + dXe * dt;
    Z = Z + dZ * dt;
```

```
    t = t + dt;

    DATA(i,:) = [X(1), X(2), Xe(1), Xe(2), U, X(5), Xe(5)];
    i = i+1;

end
toc

kk = 1e3;
t = [1:length(DATA)]' * dt;
DATAds = downsample(DATA,kk);
tds = downsample(t,kk);

subplot(1,2,1);
plot(tds,DATAds(:,[1:4]));
grid on;
legend('r(t)','\theta(t)','r_e(t)','\theta_e(t)');
title('Simulated Step Response of Beam-Ball System');

subplot(1,2,2);
plot(tds,DATAds(:,5), tds,DATAds(:,6));
grid on;
legend('d_{est}','d');
title('Disturbance vs Disturbance Estimate');
```