ECE 463/663 - Homework #11

LQR Observers. Due Wednesday,. April 23rd

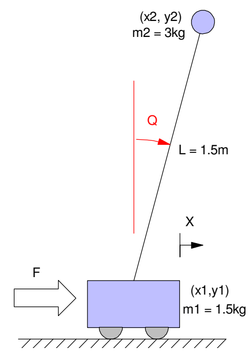
Kalman Filters

Cart and Pendulum (HW #4): The dynamics for a cart and pendulum system with sensor and input noise is as follows

$$s\begin{bmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -19.6 & 0 & 0 \\ 0 & 19.6 & 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0.667 \\ -0.444 \end{bmatrix}(F+\eta_u)$$

$$y_1 = x + n_x$$

$$y_2 = \theta + n_\theta$$



where there is Gaussian noise at the input and output

$n_u \sim N(0, 0.15^2)$ *mean zero, standard deviation 0.15*

$n_x \sim N(0, 0.02^2)$ *mean zero, standard deviation 0.02*

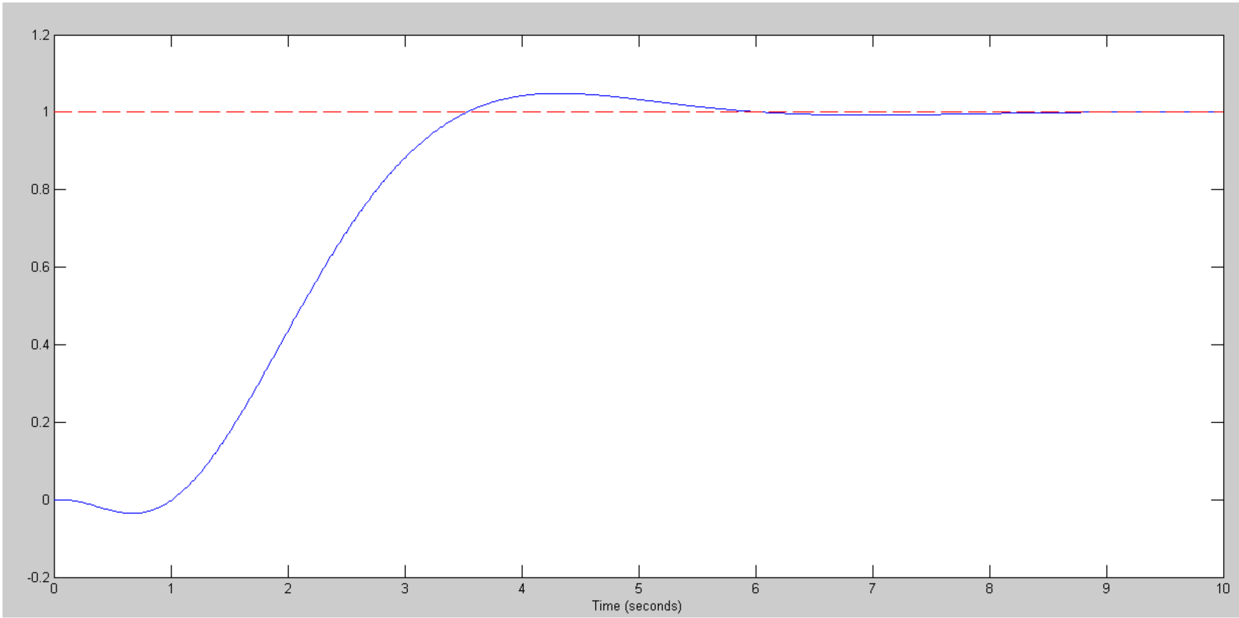
$n_\theta \sim N(0, 0.03^2)$ *mean zero, standard deviation 0.03*

1) Use a servo-compensator to force the DC gain to one (i.e. use the servo compensator from homework set #10.

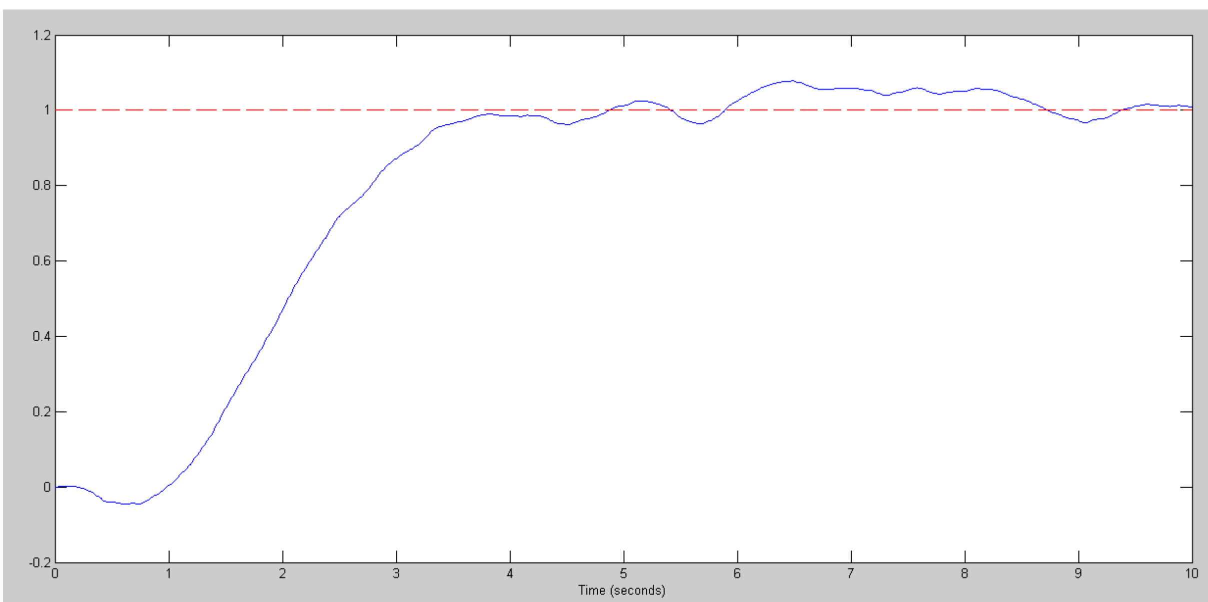
The dynamics become

$$\begin{bmatrix} sX \\ sZ \end{bmatrix} = \begin{bmatrix} A - BK_x & -BK_z \\ C & 0 \end{bmatrix}\begin{bmatrix} X \\ Z \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix}R + \begin{bmatrix} B \\ 0 \end{bmatrix}\eta_u + \begin{bmatrix} Bk_x \\ 1 \end{bmatrix}\eta_x + \begin{bmatrix} Bk_\theta \\ 0 \end{bmatrix}\eta_\theta$$

- The input noise (Nu) is fed directly into the plant (X) through the B matrix
- The sensor noise (Nx) is fed into U through the feedback gain Kx(1) and the servo
- The sensor noise (Nq) is fed into Y through the feedback gain Kx(2)



Step Response Without Noise



Step Response With Noise

Matlab Code

```
A = [0,0,1,0;0,0,0,1;0,-19.6,0,0;0,19.6,0,0];
B = [0;0;0.6667;-0.4444];
C = [1,0,0,0];

A5 = [A, zeros(4,1) ; C, 0];
B5 = [B; 0];
C5 = [C, 0];
B5r = [zeros(4,1); -1];

K5 = lqr(A5, B5, diag([10,0,0,0,100]), 1);

t = [0:0.01:10]';
npt = length(t);
Nu = 0.15*randn(npt,1);
Nx = 0.02*randn(npt,1);
Nq = 0.03*randn(npt,1);
R = 0*t+1;

B5u = [B;0];
B5x = [B*K5(1) ; 1];
B5q = [B*K5(2) ; 0];

D5 = [0,0,0,0];
X0 = zeros(5,1);

y = step3(A5-B5*K5, [B5r, B5u, B5x, B5q], C5, D5, t, X0, [R, 0*Nu, 0*Nx,
0*Nq]);

plot(t,y,'b', t, R, 'r--');
xlabel('Time (seconds)');
```

2) Design a full-order observer using pole-placement to place the observer poles at {-3, -3, -3, -3}

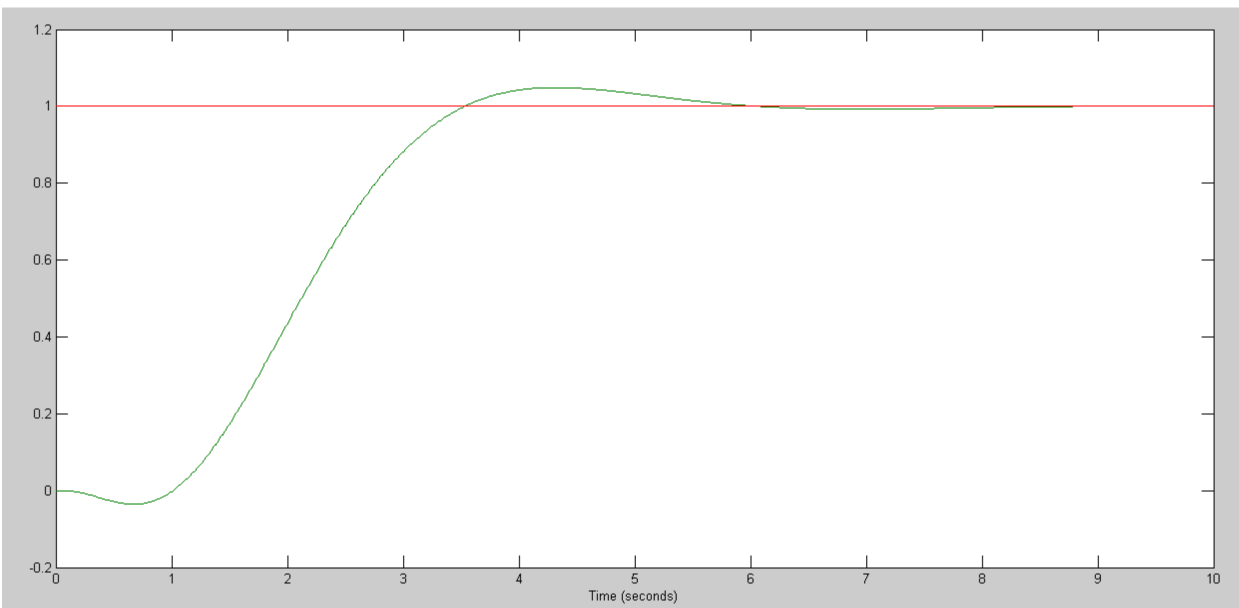
- Simulate the response of the cart with noise added at the input and output.
- Plot the states of the plant and the observer with noise,.

Plant + Observer + Servo

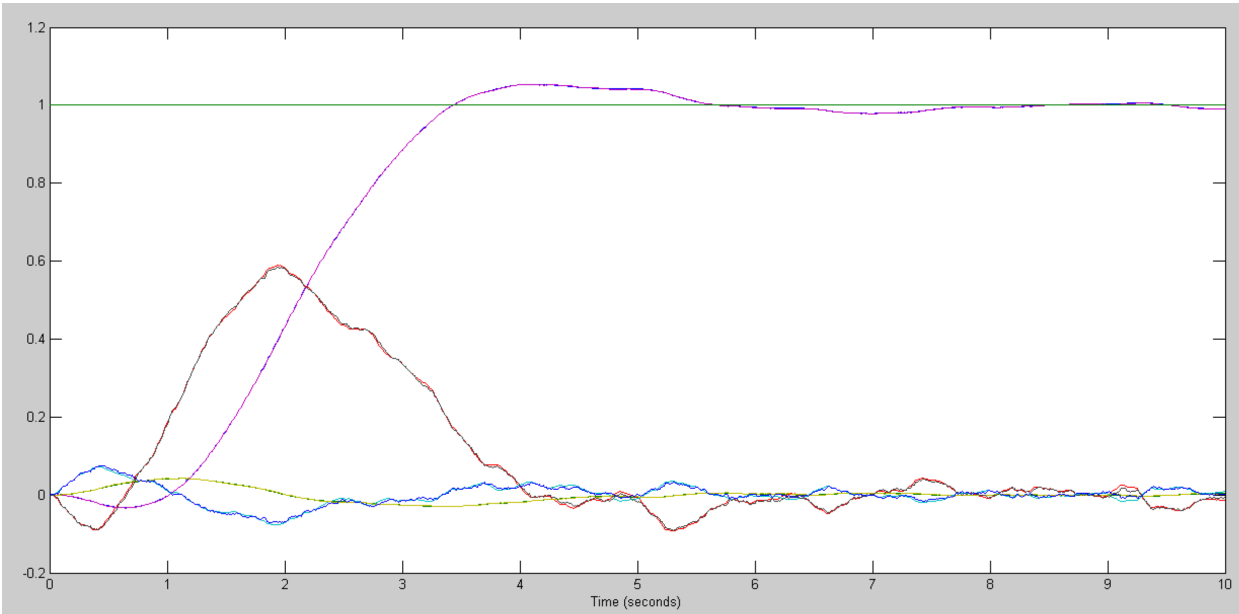
$$s\begin{bmatrix} X \\ Z \\ X_e \end{bmatrix} = \begin{bmatrix} A & -BK_z & -BK_x \\ C & 0 & 0 \\ HC & -BK_z & A-HC-BK_x \end{bmatrix}\begin{bmatrix} X \\ Z \\ X_e \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}R + \begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix}n_u + \begin{bmatrix} 0 \\ 1 \\ H_x \end{bmatrix}n_x + \begin{bmatrix} 0 \\ 0 \\ H_q \end{bmatrix}n_q$$

H in this case is:

```
     Hx            Hq
   12.0000          0
  -17.5102          0
   73.6000          0
  -77.7327          0
```



Step Response Without Noise



Step Response With Noise

Code

```
A = [0,0,1,0;0,0,0,1;0,-19.6,0,0;0,19.6,0,0];
B = [0;0;0.6667;-0.4444];
Cx = [1,0,0,0];
Cq = [0,1,0,0];

A5 = [A, zeros(4,1) ; Cx, 0];
B5 = [B; 0];
C5 = [Cx, 0];
B5r = [zeros(4,1); -1];

K5 = lqr(A5, B5, diag([10,0,0,0,100]), 1);
Kx = K5(1:4);
Kz = K5(5);

Hx = ppl(A', Cx', [-3,-3,-3,-3])';
Hq = zeros(4,1);
H = [Hx,Hq];
C = [Cx;Cq];

A9 = [A, -B*Kz, -B*Kx ; Cx,0,0,0,0,0 ; H*C, -B*Kz, A-H*C-B*Kx];
B9r = [0*B;-1;0*B];
B9u = [B;0;0*B];
B9x = [0*B;1;Hx];
B9q = [0*B;0;Hq];

C9 = [Cx,zeros(1,5) ; zeros(1,5),Cx];
D9 = zeros(2,4);

t = [0:0.01:10]';
npt = length(t);
Nu = 0.015*randn(npt,1);
Nx = 0.002*randn(npt,1);
Nq = 0.003*randn(npt,1);
R = 0*t+1;

X0 = zeros(9,1);

y = step3(A9, [B9r, B9u, B9x, B9q], C9, D9, t, X0, [R, Nu, Nx, Nq]);

plot(t,[y,R]);
xlabel('Time (seconds)');
```

3) Design a Kalman filter (i.e. a full-order observer with a specific Q and R)

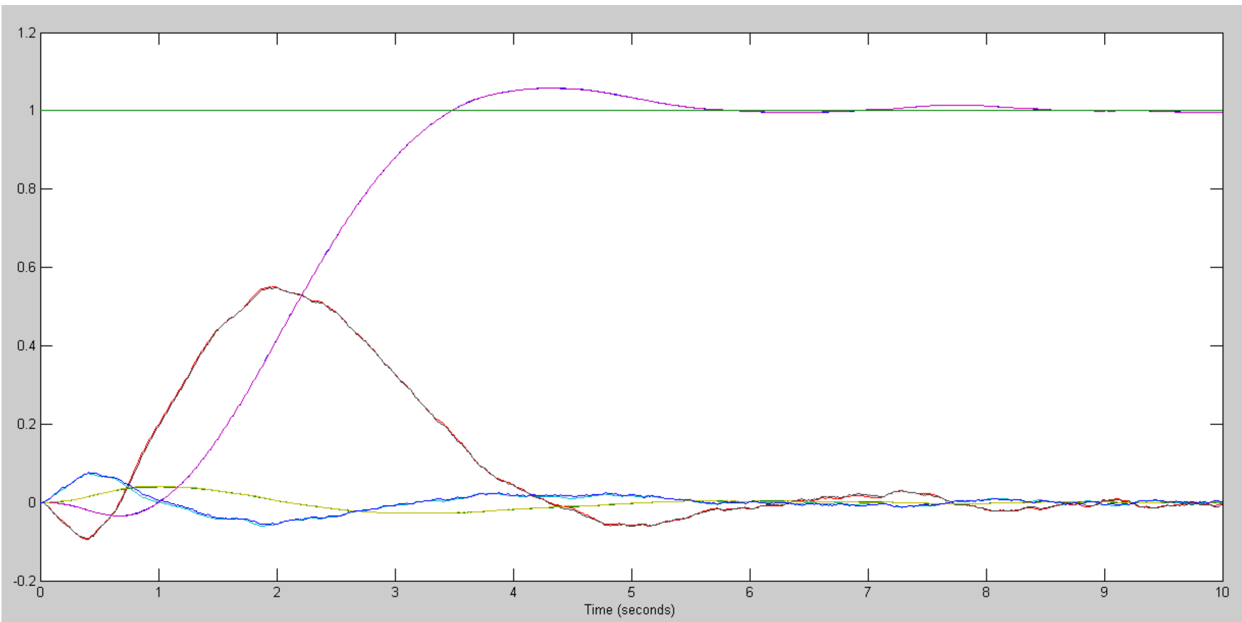
- Simulate the response of the cart with noise added at the input and output.
- Plot the states of the plant and the observer with noise,.

The obverver gains are:

```
      Hx        Hq
    6.3272   -2.9478
   -6.6325    4.4798
   29.7921  -18.7069
  -29.5866   19.8099
```

vs. before with pole placement

```
     Hx            Hq
   12.0000         0
  -17.5102         0
   73.6000         0
  -77.7327         0
```



The state estimates are better with a Kalman filter:

	std(x-xe)	std(q-qe)	std(dx-dxe)	std(dq-dqe)
pole placement	0.0070	0.0106	0.0441	0.0469
kalman filter	0.0049	0.0063	0.0262	0.0279

Code

```
A = [0,0,1,0;0,0,0,1;0,-19.6,0,0;0,19.6,0,0];
B = [0;0;0.6667;-0.4444];
Cx = [1,0,0,0];
Cq = [0,1,0,0];
C = [Cx;Cq];

A5 = [A, zeros(4,1) ; Cx, 0];
B5 = [B; 0];
C5 = [Cx, 0];
B5r = [zeros(4,1); -1];

K5 = lqr(A5, B5, diag([10,0,0,0,100]), 1);
Kx = K5(1:4);
Kz = K5(5);

F = B;
Q = (F*0.15)*(F*0.15)';
R = diag([0.02^2;0.03^2]);
H = lqr(A', C', Q, R)';
Hx = H(:,1);
Hq = H(:,2);

A9 = [A, -B*Kz, -B*Kx ; Cx,0,0*Cx ; H*C, -B*Kz, A-H*C-B*Kx];
B9r = [0*B;-1;0*B];
B9u = [B;0;0*B];
B9x = [0*B;1;Hx];
B9q = [0*B;0;Hq];

C9 = eye(9,9);
C9 = C9([1:4,6:9],:) ;
D9 = zeros(8,4);

t = [0:0.01:10]';
npt = length(t);
Nu = 0.015*randn(npt,1);
Nx = 0.002*randn(npt,1);
Nq = 0.003*randn(npt,1);
R = 0*t+1;

X0 = zeros(9,1);

y = step3(A9, [B9r, B9u, B9x, B9q], C9, D9, t, X0, [R, Nu, Nx, Nq]);

plot(t,[y,R]);
xlabel('Time (seconds)');
```