John Sigmon
Js85773

HW 5

1.)

This system is reachable/controllable by inspection. It is already in reachable canonical form. We know that a system in reachable canonical form is always reachable, by definition.

The controllability matrix is given by a simple modification to the code below, change line 5 to Wo = ctrb(A, B);

Replace “observable” with controllable…

```
>> problem1
System is observable
     1    -4     4
     0     1    -4
     0     0     1
```

2.)

```
>> problem1
System is observable
     0     1     3
     1     3     0
    -1   -12    -7
```

```
1   function problem1
2
3   A = [-4 -12 -7; 1 0 0; 0 1 0];
4   C = [0 1 3];
5   Wo = obsv(A,C);
6   observable = 0;
7   if (rank(Wo) == size(A))
8       observable = 1;
9       disp("System is observable");
10      disp(Wo);
11  else
12      disp("System is not observable");
13  end
```

The system is observable by the MATLAB code above, i.e. Wo is full rank/invertible.

John Sigmon
Js85773

3.)

To test for reachability, we are looking for a change in the state variables induced by a specific output. This can be accomplished in a few ways, including a coupling between states, or a specific input. Some systems are not reachable. Mathematically, reachability depends on the A and B, via the reachability matrix. If the reachability matrix is invertible, then the system is reachable.

To test for observability, we are looking to see if all states can be measured, i.e. does C=Az have a solution. We can test for observability by taking successive derivatives of C, and seeing if it allows us to solve the problem. Mathematically, observability depends on A and C, via the observability matrix. If the observability matrix is invertible, then the system is observable.

Another way to look at this question is: reachability is asking if an initial condition can be taken back to the origin, and observability is asking if you can recreate the initial condition if you start at the origin. There are certainly cases where one can be true and not the other. Using the example from class, consider the system:

$$A = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad C = [1 \quad 1] \qquad and\ z_1(0) = 0\ ;\ z_2(0) = 2$$

Then if you require a desired final state of:

$$z_1(< T) = 1\ ;\ z_2(< T) = 2$$

Then the system is not reachable. There are no inputs that would drive those initial conditions to the desired final state. However, the system is completely observable, as C allows measurements of z1 and z2.

John Sigmon
Js85773

4.) a.)

```
>> problem4a
The C1
system is not observable
The C2
system is observable
The C3
system is not observable
Wo,1 is
     1     0     1
     4     0     1
    13     0     4

Wo,2 is
     0     1     0
    -3     1     0
   -12     1    -3

Wo,3 is
     0     0     1
     1     0     0
     3     0     1
```

```
1   function problem4a
2
3   A = [3 0 1; -3 1 0; 1 0 0];
4   C1 = [1 0 1];
5   C2 = [0 1 0];
6   C3 = [0 0 1];
7   W1 = obsv(A,C1);
8   W2 = obsv(A,C2);
9   W3 = obsv(A,C3);
10
11  disp('The C1');
12  observable(W1,A);
13  disp('The C2');
14  observable(W2,A);
15  disp('The C3');
16  observable(W3,A);
17
18  disp('Wo,1 is');
19  disp(W1);
20  disp('Wo,2 is');
21  disp(W2);
22  disp('Wo,3 is');
23  disp(W3);
24
25
26  end
27
28  function observable(W, A)
29  if (rank(W) == size(A))
30      disp("system is observable");
31  else
32      disp("system is not observable");
33  end
34  end
```

By definition, rank(Wo) must be full (or equal to rank(A)) for the system to be observable. C1 and C3 do not satisfy this condition, by the above code. C2 does.

John Sigmon
Js85773

b.)

$$A = \begin{bmatrix} 3 & 0 & 1 \\ -3 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \rightarrow \det(A - \lambda I) = 0$$

$$\begin{bmatrix} 3-\lambda & 0 & 1 \\ -3 & 1-\lambda & 0 \\ 1 & 0 & -\lambda \end{bmatrix} \rightarrow (3-\lambda)(1-\lambda)(-\lambda) + (-1+\lambda) = 0$$

$$-\lambda^3 + 4\lambda^2 - 3\lambda + \lambda - 1 = 0$$

$$p(\lambda) = \lambda^3 - 4\lambda^2 + 2\lambda + 1 \qquad a_1 = -4,\ a_2 = 2,\ a_3 = 1$$

```
function problem4b

A = [3 0 1; -3 1 0; 1 0 0];
C = [0 1 0];
Atilda = [4, 1, 0; -2, 0 1; -1, 0, 0];
Ctilda = [1 0 0];

wo = obsv(A,C);
wotilda = obsv(Atilda, Ctilda);

T = wo\wotilda;
disp('The transformation T is')
disp(T)
```

```
>> problem4b
The transformation T is
   -1.0000   -0.3333         0
    1.0000         0         0
   -0.3333         0   -0.3333
```

c.)

The eigenvalues for A are -0.3208, 1, and 3.3028. Therefore L = 0 will not produce a functional observer. The error will not go to zero because the eigenvalues are not "in the left-hand side of the plane". In other words, $Re\{\lambda_i\} \leq 0 \ \forall \ \lambda$. Additionally, by

$$\frac{d\bar{e}}{dt} = (A - LC)\bar{e}$$

If L = 0, then we have:

$$\frac{d\bar{e}}{dt} = A\bar{e}$$

And so we see that the error goes to zero iff A is stable. Because A is not stable, this is not the case for us.

John Sigmon
Js85773

d.)

The estimator was found with standard MATLAB methods. Zerror was interpolated and used to calculate Zhat. Note the different graph scales used to show convergence more clearly.

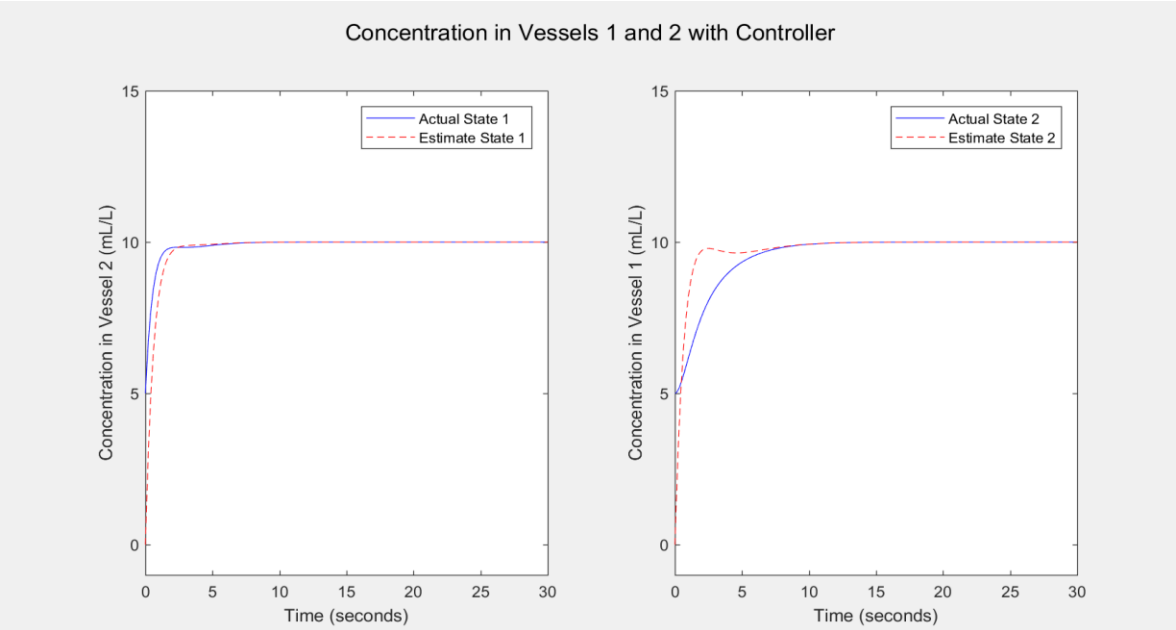
```
>> redo
rank Wo is:
     3

L is:
  -30.0000
   16.0000
  -21.0000

Eigenvalues of A-LC are:
   -6.0000
   -4.0000
   -2.0000
```

```
function redo
close all;

A = [3 0 1; -3 1 0; 1 0 0 ];
B = [1;0;0];
C = [0 1 0];

%check observability
Wo = obsv(A, C);
disp('rank Wo is: ');
disp(rank(Wo));

%pole placement
L = (place(A', C', [-2 -4 -6]))';
disp('L is: ');
disp(L);

%check eigenvalues
disp('Eigenvalues of A-LC are: ');
disp(eig(A-L*C));


%%%%%%%%%%%% Plotting Zero State Response %%%%%%%%%%%%
[t1,z1] = ode45(@ss1, [0 2], [0 0 0]);
[t2,z2] = ode45(@sserror, [0 2], [0 0 0]);
zerror = interp1(t2, z2, t1);
zhat = z1-zerror;
figure;
subplot(2,3,1); semilogy(t1, z1(:,1), 'b'); hold on;
subplot(2,3,1); semilogy(t1, zhat(:,1), '--r');
xlabel('Time');
ylabel('State 1');
ylim([10.^(-10) 10.^2]);
legend('Actual','Estimated', 'Location', 'southeast');

subplot(2,3,2); plot(t1, z1(:,2), 'b'); hold on;
subplot(2,3,2); plot(t1, zhat(:,2), '--r');
xlabel('Time');
ylabel('State 2');
xlim([0 1]);
legend('Actual','Estimated', 'Location', 'southwest');

subplot(2,3,3); plot(t1, z1(:,3), 'b'); hold on;
subplot(2,3,3); plot(t1, zhat(:,3), '--r');
xlabel('Time');
ylabel('State 3');
xlim([0 2]);
legend('Actual','Estimated', 'Location', 'northwest');


%%%%%%%%%%%%%%%%%% Plotting IC response %%%%%%%%%%%%%%
clear all;
[t1,z1] = ode45(@ss1, [0 2], [1 1 1]);
[t2,z2] = ode45(@sserror, [0 2], [1 1 1]);
zerror = interp1(t2, z2, t1);
zhat = z1-zerror;
subplot(2,3,4); plot(t1, z1(:,1), 'b'); hold on;
subplot(2,3,4); plot(t1, zhat(:,1), '--r');
xlabel('Time');
ylabel('State 1');
xlim([0 1]);
legend('Actual','Estimated', 'Location', 'northwest');

subplot(2,3,5); plot(t1, z1(:,2), 'b'); hold on;
subplot(2,3,5); plot(t1, zhat(:,2), '--r');
xlabel('Time');
ylabel('State 2');
xlim([0 1]);
legend('Actual','Estimated', 'Location', 'southwest');

subplot(2,3,6); plot(t1, z1(:,3), 'b'); hold on;
subplot(2,3,6); plot(t1, zhat(:,3), '--r');
xlabel('Time');
ylabel('State 3');
xlim([0 1.5]);
legend('Actual','Estimated', 'Location', 'northwest');
suptitle({'Plots of All System States;','Actual vs. Estimator','Zero State and IC Responses'})


%%%%%%%%%%%%%%%% Plotting Error %%%%%%%%%%%%%%
[t2,z2] = ode45(@sserror, [0 30], [0 0 0]);
figure;
subplot(1,3,1); plot(t2,z2(:,1));
xlabel('Time')
ylabel('State 1');
subplot(1,3,2); plot(t2,z2(:,2));
xlabel('Time')
ylabel('State 2');
subplot(1,3,3); plot(t2,z2(:,3));
xlabel('Time')
ylabel('State 3');
suptitle('Error in Estimator')

end

function zprime = ss1(t,z)
    A = [3 0 1; -3 1 0; 1 0 0 ];
    B = [1;0;0];
    zprime = A*z+B;
end
function error = sserror(t,z)
    A = [3 0 1; -3 1 0; 1 0 0 ];
    B = [1;0;0];
    C = [0 1 0];
    L = (place(A', C', [-2 -4 -6]))';
    Ao = (A-L*C);
    error = Ao*z +B;
end
```

John Sigmon
Js85773

5.)

```
function compcontroller
close all;

t = 0:0.1: 30;
%system parameters
ko = 0.1;
k1 = 0.1;
k2 = 0.5;
bo = 1.5;

%system matrices
A = [-ko-k1, k1; k2, -k2];
B = [bo; 0];
C = [1 0];

%desired output
yr = 10*ones(length(t), 1);

Wo = obsv(A, C);
Wr = ctrb(A, B);
disp('Wo is: ');
disp(Wo);
disp(['rank Wo is ', num2str(rank(Wo))]);
disp('Wr is: ');
disp(Wr);
disp(['rank Wr is ', num2str(rank(Wr))]);

Lt = place(A', C', [-1 + 0.1*j; -1 - 0.1*j]);
L = Lt';
K = place(A, B, [-0.5 -1]);
%disp(['cond no of L is: ', num2str(cond(L))]);
%disp(['cond no of K is: ', num2str(cond(K))]);
disp('Eigenvalues of A-BK are: ');
disp(eig(A-B*K));
disp('Eigenvalues of A-LC are: ');
disp(eig(A-L*C));


%kr derivation
kr = -1/(C*inv(A-B*K)*B);

% augmented system
Aa = [A-B*K B*K; zeros(2,2) A-L*C];
Ba = [B*kr; zeros(2,1)];
Ca = [C zeros(size(C)); zeros(size(C)) C];

%Aa = [A-B*k];
%Ba = B*kr;
sys1 = ss(Aa, Ba, Ca, 0);
%{
Ae = (A-L*C)*e;
Be = B*K;
Ce = [1 1];
%}
[z1, t1, x1] = lsim(sys1, yr, t, [5 5 5 5]);
observer1 = x1(:,1)- x1(:,3);
actual1 = x1(:,1);
observer2 = x1(:,2)- x1(:,4);
actual2 = x1(:,2);
subplot(1,2,1);
plot(t1, actual1, 'b', t1, observer1, '--r');
xlabel('Time (seconds)');
ylabel('Concentration in Vessel 2 (mL/L)');
legend('Actual State 1', 'Estimate State 1');
ylim([-1 15]);
subplot(1,2,2);
plot(t1, actual2, 'b', t1, observer2, '--r');
xlabel('Time (seconds)');
ylabel('Concentration in Vessel 1 (mL/L)');
ylim([-1 15]);
legend('Actual State 2', 'Estimate State 2');
suptitle('Concentration in Vessels 1 and 2 with Controller');
```




```
>> compcontroller
Wo is:
    1.0000         0
   -0.2000    0.1000

rank Wo is 2
Wr is:
    1.5000   -0.3000
         0    0.7500

rank Wr is 2
Eigenvalues of A-BK are:
   -0.5000
   -1.0000

Eigenvalues of A-LC are:
  -1.0000 + 0.1000i
  -1.0000 - 0.1000i
```

The code shows all steps taken to identify A, B, C, D. Eigenvalues were chosen by trial and error. The plot also shows all states.

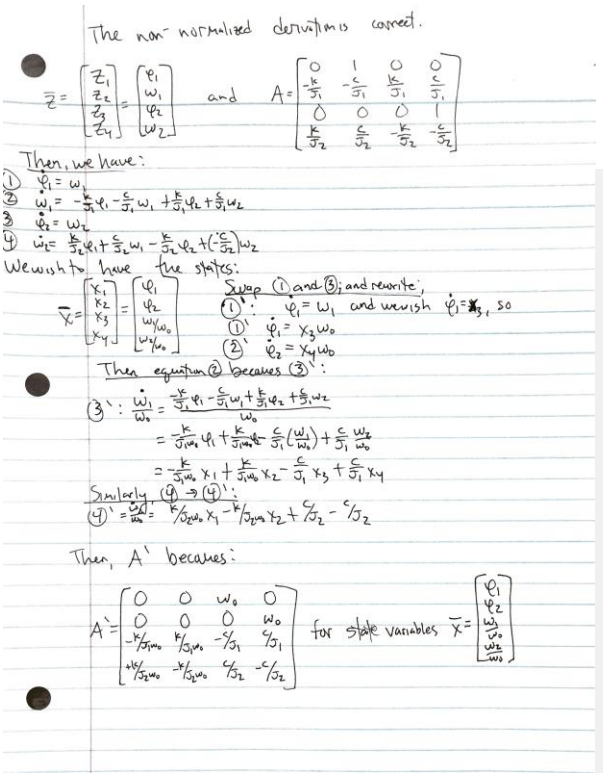
John Sigmon
Js85773

6.)

I rederived the model, ran the code, and got errors because the matrix is so poorly conditioned MATLAB could not place the eigenvalues. I gave up for the day and was a dummy and did not realize the homework was due on Tuesday as opposed to Thursday. Here is my code, set up and ready to go, just waiting for L and K to be calculated by hand (if even possible). Also included is the derivation, and my MATLAB error.

```
%%this assumes ic = 0, yr is step, and output of interest is theta, disk 2
[z, t, x] = lsim(sysl, yr, t, ic);
observer = x(:,3) - x(:,7);
actual = x(:,3);
plot(t,actual, 'xb', t, observer, 'r');




end

```

John Sigmon
Js85773




```
>> problem6
Wo is:
         0         0    1.0000         0
   -0.9000    0.9000   -0.0900    0.0900
    0.0900   -0.0900   -0.8910    0.8910
    0.8910   -0.8910    0.1791   -0.1791

rank Wo is 3
Wr is:
         0         0    0.8100   -0.0810
    0.9000         0   -0.0900    0.0090
         0    0.8100   -0.0810   -0.8019
         0   -0.0900    0.0090    0.0891

rank Wr is 3
Error using place (line 171)
The "place" command could not place the poles
at the specified locations. Probable causes
include:
* (A,B) is nearly uncontrollable
* The specified locations are too close to each
other.

Error in problem6 (line 58)
Lt = place(A',C',p2);
```

```
function problem6
%{
 Given motor drive of two disks with spring between:
 Open Loop eigenvalues: 0, 0, -0.05 +- j
 Desired state feedback controller eigenvalues:
    -2, -1, -1 +- j
 Desired observer eigenvalues:
    -4, -2, -2 +- 2j
%}


%derivation of system equations
J1 = 10/9;          %inertia of disk 1
J2 = 10;            %inertia of disk 2
c = 0.1;            %spring constant
k = 1;              %friction
k1 = 1;             %motor constant
wo = sqrt(k*(J1 + J2)/(J1*J2));

%{
Let    z1 = disk1 position
        z2 = disk2 position
        z3 = disk1 normalized ang. velocity
        z4 = disk2 normalized ang. velocity
%}
% define SS model
A = [0 0 wo 0;  0 0 0 wo;
    -k/(wo*J1) k/(J1*wo) -c/J1 c/J1;
     k/(wo*J2) -k/(wo*J2) c/J2 -c/J2];
B = [0; k1/(wo*J1); 0; 0];
C = [0 0 1 0];

%desired output and time
t = 0:0.1:30;
yr = ones(1, length(t));
```

```
%check observability, reachability
Wo = obsv(A, C);
Wr = ctrb(A, B);
disp('Wo is: ');
disp(Wo);
disp(['rank Wo is ', num2str(rank(Wo))]);
disp('Wr is: ');
disp(Wr);
disp(['rank Wr is ', num2str(rank(Wr))]);

%disp(cond(A));
%disp(A);

%desired controller eigenvalues
p1 = [-2 -1 -1+j -1-j];

%desired observer eigenvalues
p2 = [ -4 -2 -2+2*j -2-2*j];

%assign poles for K, L
%K = place(A,B,p1);
Lt = place(A',C',p2);
L = Lt';

%kr term
kr = -1/(C*inv(A-B*K)*B);

% augmented system
Aa = [A-B*K B*K; zeros(4,4) A-L*C];
Ba = [B*kr; zeros(4,1)];
Ca = [C zeros(size(C)); zeros(size(C)) C];

sys1 = ss(Aa, Ba, Ca, 0);

ic = zeros(1, size(Aa,1));
```