Control Theory 5

Kamil Kamaliev, Var b
k.kamaliev@innopolis.university

April 2020

$$\begin{cases} (M+m)x'' - mlcos(\theta)\theta'' + mlsin(\theta){\theta'}^2 = F \\ -cos(\theta)x'' + l\theta'' - gsin(\theta) = 0 \end{cases}$$

The linearization of the system was performed using the tools from the previous homework with values $M = 3.4,\ m = 8.2,\ l = 0.89$:

$$\begin{cases} \delta z' = A\delta z + B\delta u \\ \delta y = C\delta z \end{cases}$$

where

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 23.66 & 0 & 0 \\ 0 & 37.6 & 0 & 0 \end{bmatrix},\ B = \begin{bmatrix} 0 \\ 0 \\ 0.29 \\ 0.33 \end{bmatrix},\ C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

A. Possibility to design state observer
The required check was made by constructing observability matrix with the following Matlab Code:

Listing 1: Checking the observability of the system

```
A = [0 0 1 0; 0 0 0 1; 0 23.66 0 0; 0 37.6 0 0];
B = [0; 0; 0.29; 0.33];
C = [1 0 0 0; 0 1 0 0];
D = [0; 0];
sys = ss(A,B,C,D);

Ob = obsv(A,C);
unob = length(A)-rank(Ob);
unob
```

unob shows the number of unobservable states.
In my case, it's equal to 0.

B. Stability of error dynamics

We have a system with dynamics $z' = Az + Bu$. Thus, the open loop state observer is $\overline{z}' = A\overline{z} + Bu$.

From this, we may derive the error: $e = z - \overline{z}$. And after subtracting z' from z, we obtain: $e' = Ae$.

To check the stability, one may find the eigenvalues of A. This was done by the following Matlab code:

Listing 2: Obtaining eigenvals of matrix A

```
1
2   A = [0 0 1 0; 0 0 0 1; 0 23.66 0 0; 0 37.6 0 0];
3   B = [0; 0; 0.29; 0.33];
4   C = [1 0 0 0; 0 1 0 0];
5   D = [0; 0];
6   sys = ss(A,B,C,D);
7
8   eig(A)
```

the result is : $\begin{bmatrix} 0 \\ 0 \\ 6.1319 \\ -6.1319 \end{bmatrix}$

As we can see, 1 eigenvalue is positive making the error dynamics unstable.

C. Luenberger observer

We have a system

$$\begin{cases} z' = Az + Bu \\ y = Cz \end{cases}$$

and the following observer $\overline{z}' = A\overline{z} + Bu + L(y - \overline{y})$. After deriving error $e = \overline{z} - z$ and subtracting z' from $\overline{z}'$ we obtain:

$$e' = (A - LC)e$$

We need to find such L that makes the matrix $A - LC$ negative definite. This is the same as finding such L that makes matrix $A^T - C^T L^T$

LQR Way For weight matrices we may take Identity matrices. The L is found by the following Matlab code:

Listing 3: LQR Method

```
1
2   A = [0 0 1 0; 0 0 0 1; 0 23.66 0 0; 0 37.6 0 0];
3   B = [0; 0; 0.29; 0.33];
4   C = [1 0 0 0; 0 1 0 0];
5   D = [0; 0];
6
7   N = [0 0; 0 0; 0 0; 0 0];
```

```
Q = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
R = [1];

[K,S,e] = lqr(transpose(A), transpose(C), Q, R, N);
L = transpose(K);
L
```

The result is L = $\begin{bmatrix} 4.1118 & 4.7048 \\ 4.7048 & 10.0914 \\ 19.0208 & 37.7669 \\ 29.0557 & 61.4856 \end{bmatrix}$

Pole Placement Way

For the error to be stable, we take negative numbers for poles. The L is found by the following Matlab code:

Listing 4: Pole Placement Method

```
A = [0 0 1 0; 0 0 0 1; 0 23.66 0 0; 0 37.6 0 0];
B = [0; 0; 0.29; 0.33];
C = [1 0 0 0; 0 1 0 0];
D = [0; 0];

p = [-1 -0.5 -0.2 -0.1]

L = place(A', C', p)';
L
```

The result is L = $\begin{bmatrix} 0.3 & 0 \\ 0 & 1.5 \\ 0.02 & 23.66 \\ 0 & 38.1 \end{bmatrix}$

D. State Feedback Controller

We have the following system:

$$\begin{cases} z' = Az + Bu \\ u = -Kz \end{cases}$$

From which we obtain: $z' = (A - BK)u$.

Now, we need to find such K that makes $A - BK$ negative definite. This may be done using the pole placement method just like in the previous exercise by the following code:

Listing 5: Finding K

```
A = [0 0 1 0; 0 0 0 1; 0 23.66 0 0; 0 37.6 0 0];
```

```
B = [0; 0; 0.29; 0.33];
C = [1 0 0 0; 0 1 0 0];
D = [0; 0];

p = [-1 -0.5 -0.2 -0.1]

K = place(A, B, p);
K
```

The result is $K = \begin{bmatrix} -0.0032 & 116.8816 & -0.0581 & 5.5056 \end{bmatrix}$

E. Simulation

After adding control and observer, we obtain: $z' = (A - BK - LC)z$.

Listing 6: Simulation

```
A = [0 0 1 0; 0 0 0 1; 0 23.66 0 0; 0 37.6 0 0];
B = [0; 0; 0.29; 0.33];
C = [1 0 0 0; 0 1 0 0];
D = [0; 0];

p = [-1 -0.5 -0.2 -0.1];

K = place(A, B, p);
L = place(A', C', p)';

A_ = A - B * K - L * C;

controlled_system = ss(A_, B, C, D);

t = 0:0.1:10;
u = sin(t);

figure(1);
subplot(121);
step(controlled_system);
title(Step response);
subplot(122);
y = lsim(controlled_system, u, t);
plot(t, y);
title(Sine response);
```

As we can see in Figure 1 (which for some reason went to task F), the system is stable, observer works quite good but the scaling is not quite right.

F. AWGN in the output

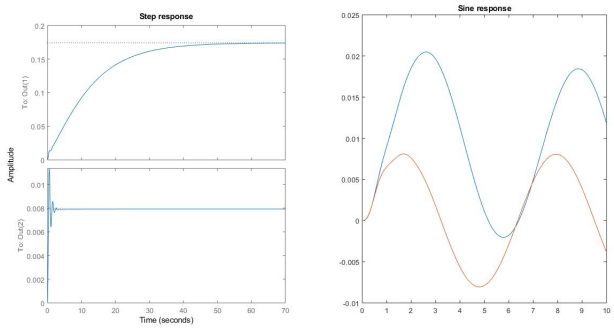

Figure 1: Simulation

After adding controller, observer to the system and noise to the output, we obtain the following system:

$$\begin{cases} \delta z' = A_{\_} \cdot \delta z + B_{\_} \cdot u \\ \delta y = C \cdot \delta z + u \end{cases}$$

where $A_{\_} = (A - BK - LC)$, $B_{\_} = -L$ and $u = v$
After these substitution, we can represent our dynamics in SS:

Listing 7: AWGN in output

```
A = [0 0 1 0; 0 0 0 1; 0 23.66 0 0; 0 37.6 0 0];
B = [0; 0; 0.29; 0.33];
C = [1 0 0 0; 0 1 0 0];

p = [-1 -0.5 -0.2 -0.1];

K = place(A, B, p);
L = place(A', C', p)';

A_ = A - B * K - L * C;
t = 0:0.01:10;
size(t)

controlled_system = ss(A_, -1 * L, C, [1 0; 0 1]);


u = zeros(2, 1001);
u_noise = awgn(u, 10);
z0 = [-0.2, 0, 0.1, 0.03];

figure(1);
subplot(121);
```

```
    [y, t, z] = lsim(controlled_system, u, t, z0);
    z
    plot(t, z(:, 1:2));
    title('Without noise');
    subplot(122);
    [y, t, z] = lsim(controlled_system, u_noise, t, z0);
    plot(t, z(:, 1:2));
    title('With AWGN');
```

Which gives the following output:

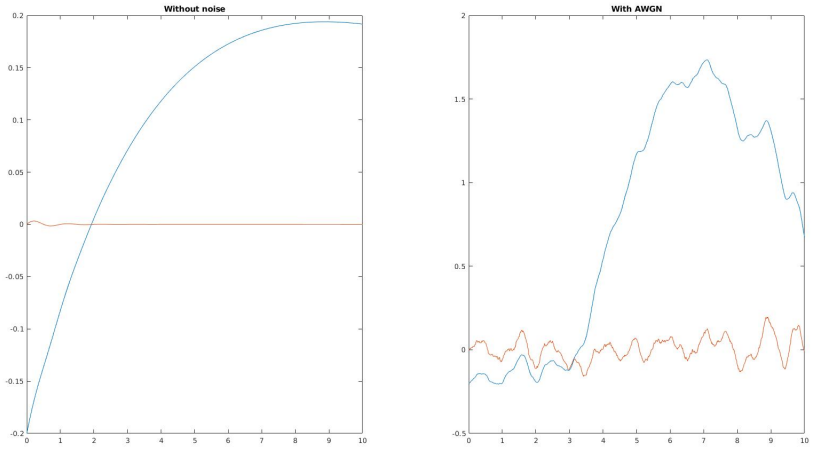

Figure 2: Graphs of x and θ

As Luenberger observer works with systems with no noise in the output, adding a noise (which is each time multiplied by L) to the output makes everything very terrible.

G. AWGN in the state

After adding noise to the state dynamics, we obtain the following system:

$$\begin{cases} \delta z' = A_{-} \cdot \delta z + \cdot u \\ \delta y = C \cdot \delta z \end{cases}$$

where $A_{-} = (A - BK - LC)$ and $u = w$
After these substitution, we can represent our dynamics in SS:

Listing 8: AWGN in output

```
    A = [0 0 1 0; 0 0 0 1; 0 23.66 0 0; 0 37.6 0 0];
    B = [0; 0; 0.29; 0.33];
    C = [1 0 0 0; 0 1 0 0];
```

```
4
5       p = [-1  -0.5  -0.2  -0.1];
6
7       K = place(A, B, p);
8       L = place(A', C', p)';
9
10      A_ = A - B * K - L * C;
11      t = 0:0.01:10;
12      size(t)
13
14      u = zeros(1, 1001);
15      u_noise = awgn(u, 10);
16      z0 = [-0.2, 0, 0.1, 0.03];
17
18      figure(1);
19      controlled_system = ss(A_, [1; 1; 1; 1], C, [0; 0]);
20      subplot(121);
21      [y, t, z] = lsim(controlled_system, u, t, z0);
22      plot(t, z(:, 1:2));
23      title('Without noise');
24      subplot(122);
25      [y, t, z] = lsim(controlled_system, u_noise, t, z0);
26      plot(t, z(:, 1:2));
27      title('With AWGN');
```

Which gives the following output:

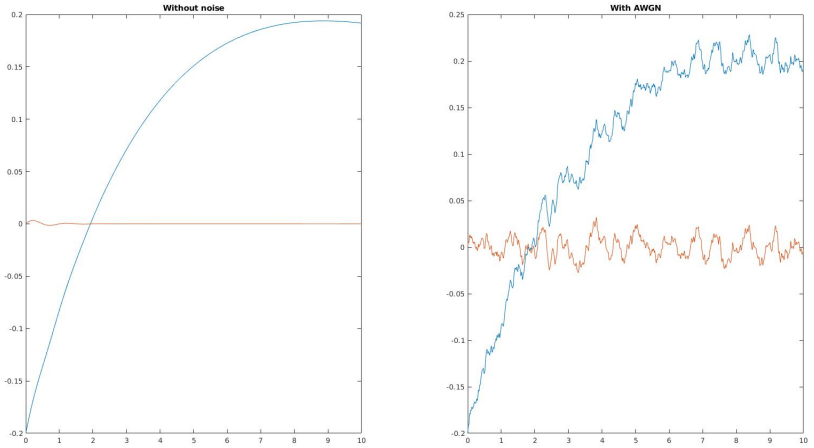

Figure 3: Graphs of x and θ

As we can see, adding noise to the state dynamics didn't make every very terrible as last time. This is because Gaussian noise is taken from a standard normal distribution centered at 0.
Due to the noise, absolute stability is lost. Nevertheless, seems like after some time the system will go into some kind of "borders", within which it will fluctuate.

H. Kalman Filter
I decided to use MatLab library

Listing 9: Kalman

```
A = [0 0 1 0; 0 0 0 1; 0 23.66 0 0; 0 37.6 0 0];
B = [0; 0; 0.29; 0.33];
C = [1 0 0 0; 0 1 0 0];
D = [0; 0];

p = [-1 -0.5 -0.2 -0.1];

K = place(A, B, p);
L = place(A', C', p)';

new_A = A - B*K - L*C;

B_ = [B [0; 0; 0; 0] [1; 1; 1; 1]];
D_ = [D [1; 1] [0; 0]];
sys = ss(new_A, B_, C, D_);

t = 0:0.01:10;
u = sin(t);
v = randn(1, length(t));
w = randn(1, length(t));
ua = [u; v; w];

Q = 1;
R = [1 0; 0 1];

[kest, L, P] = kalman(sys, Q, R);
kest = kest(1, :);

s = parallel(sys, kest, 1, 1, [], []);
SimModel = feedback(s, 1, 4, 2, 1);
SimModel = SimModel([1 3], [1 2 3]);
[out, x] = lsim(SimModel, ua, t);
y = out(:, 1);
yf = out(:, 2);

```

```
    figure(1);
    plot(t, y, 'r', t, yf, 'b');
    hold on
    [yn, x] = lsim(sys, ua, t);
    plot(t, yn(:, 1), '--g');
    title('Kalman Filter');
    legend('True Response', 'Kalman Filter', 'Noisy
        System');
```

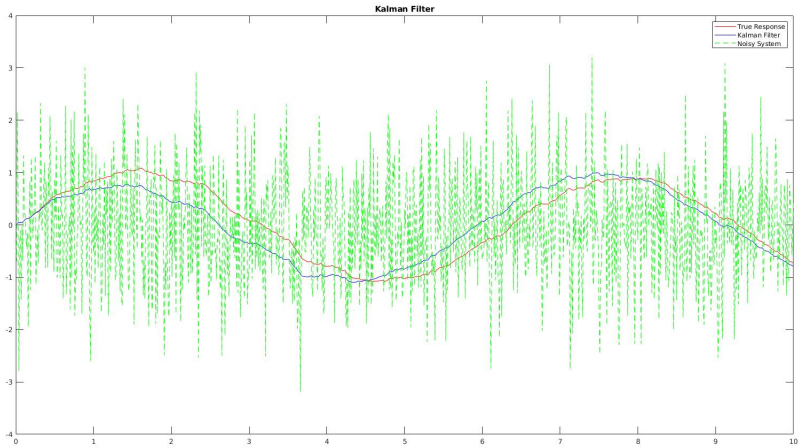

Figure 4: Kalman