Anders Helmersson, ISY, 2017-08-23

# 1 Getting Started

Regard the following plant

$$\ddot{y} = y + u,$$

which models an inverted pendulum that is controlled by a torque at the pivot point of the pendulum. The plant is unstable and has its open-loop poles in $\pm 1$. Try to find a controller that stabilizes the plant. Try to obtain reasonable phase and gain margins (e.g. 35 deg and 6 dB). Try to reduce the bandwidth. What happens with the stability margins?

Next introduce a delay in the loop. The delay is unknown and may take any value from zero up to $\tau$ seconds. Try to find controllers for some values of $\tau$, say 0.1, 0.2, 0.5 and 1 seconds. What happens with the phase and gain margins as the delay is increased? What is the maximum allowed delay? Remember that the controller should be stable for any delay between 0 and $\tau$.

# 2 Commands

During this course, we will use commands in the Robust Control Toolbox for use with Matlab. I have summarized a number of commands that you may find useful.

## 2.1 System representation

In the Robust Control Toolbox (RCT), systems can represented by their $A$, $B$, $C$ and $D$ matrices.

```
>> A = [-1 2;0 -2];
>> B = [0; 1];
>> C = [1 1];
>> D = 0;
>> sys = ss (A, B, C, D);
```

or

```
>> sys1 = tf ([1 3], [1 3 2]);
```

To show the system's representation just type its name `sys`:

```
sys

a =
       x1  x2
   x1  -1   2
   x2   0  -2

b =
       u1
   x1   0
   x2   1
```

```
c =
       x1  x2
   y1   1   1

d =
       u1
   y1   0

Continuous-time state-space model.
```

or

```
>> sys1

Transfer function:
    s + 3
-------------
s^2 + 3 s + 2
```

If you just want to know the size of the system, use

```
>> size(sys)
State-space model with 1 outputs, 1 inputs, and 2 states.
```

or

```
>> size(sys1)
Transfer function with 1 outputs and 1 inputs.
```

## 2.2 Manipulating Systems

You can connect systems in series using multiplication * or in parallel using addition or subtraction, + or -.

```
>> GK = G * K
>> GK = G + K
>> GK = G - K
```

The inverse of a (square) system:

```
>> Ginv = inv (G); % or Ginv = eye(size(G,1))/G;
```

You can stack and augment systems

```
>> GK = [G, K];
>> GK = [G; K];
>> GK = append (G, K)                   %% [g 0; 0 k]
>> GK = blkdiag (G, K)                  %% [g 0; 0 k]
```

Feedback using linear fractional transformation (LFT):

```
>> Cl = lft (G, K); % or Cl = star (G, K);
```

In some cases the system $G$ may appear at several positions:

```
>> GG = [G; G];
```

This means that $G$ is repeated twice:

```
>> size ([sys; sys])
State-space model with 2 outputs, 1 inputs, and 4 states.
```

Note that here we have four states, twice as may as in the original `sys`. (We have introduced unobservable modes). If we instead write

```
>> size ([1; 1]*sys)
State-space model with 2 outputs, 1 inputs, and 2 states.
```

You can select certain rows and columns from a system

```
>> Gsel = G(1:2, [1 3]);
```

## 2.3 Analysis

Poles and zeros of a system:

```
>> disp (pole (sys)')
    -1    -2
>> disp (zero (sys)')
    -3
```

Instead of using `pole`, you can also use `eig`. If the argument is an LTI (linear time invariant) system, `eig` returns the poles, otherwise the eigenvalues. Note that a static matrix `D` and `ss (D)` are different objects, the first one is an ordinary matrix, while the second one is a static LTI system. For instance `eig (D)` returns the eigenvalues of `D`, while `eig (ss (D))` returns an empty matrix.