Contents

- System identification of DC motor with measurements from Arduino
- Import file into Matlab. File should be in txt-format.
- We need to rearrange the data on the form iddata(output,input,sampling time)
- First, we check the sampling time
- Now, lets structure our data as iddata
- Visualize the data before starting system identification
- Going into system identification
- Now, let's try to tune a PID using our estimated system as a digital twin.
- We then define our PID

System identification of DC motor with measurements from Arduino

By Øystein Bjelland, IIR, NTNU

```
clear;
clc;
close all;
```

Import file into Matlab. File should be in txt-format.

```
filename = 'Data_DC_motor_2.txt'; %File must be located in same folder
A = importdata(filename);
```

We need to rearrange the data on the form iddata(output,input,sampling time)

```
inputVoltage_raw = A(:,1);  %Input voltage [mV]
outputShaftAngle_raw = A(:,2);   %Output shaft angle [deg]
time_raw = A(:,3);  %Raw time from millis() in Arduino [milliseconds]

inputVoltage = inputVoltage_raw / 1000; % Converting to volt
outputShaftAngle = abs(outputShaftAngle_raw); %Taking absolute value
time = zeros(length(time_raw),1);
Ts_vect = [];
```

First, we check the sampling time

```
for i = 2:length(time_raw)

   Ts = time_raw(i) - time_raw(i-1);
   Ts_vect = [Ts_vect, Ts];

   time(i) = (time_raw(i) - time_raw(1))/1000; %Starting the time vector from zero and converting from ms to s.

end

maxTs = max(Ts_vect);
disp('Maximum sample time [ms]: ')
disp(maxTs)

minTs = min(Ts_vect);
disp('Minimum sample time [ms]: ')
disp(minTs)

Ts_average = mean(Ts_vect);
disp('The average sample time is [ms]')
disp(Ts_average)
```

```
disp('Our sampling time is, Ts [sec]')
Ts = round(Ts_average)*10^-3;
disp(Ts)
```

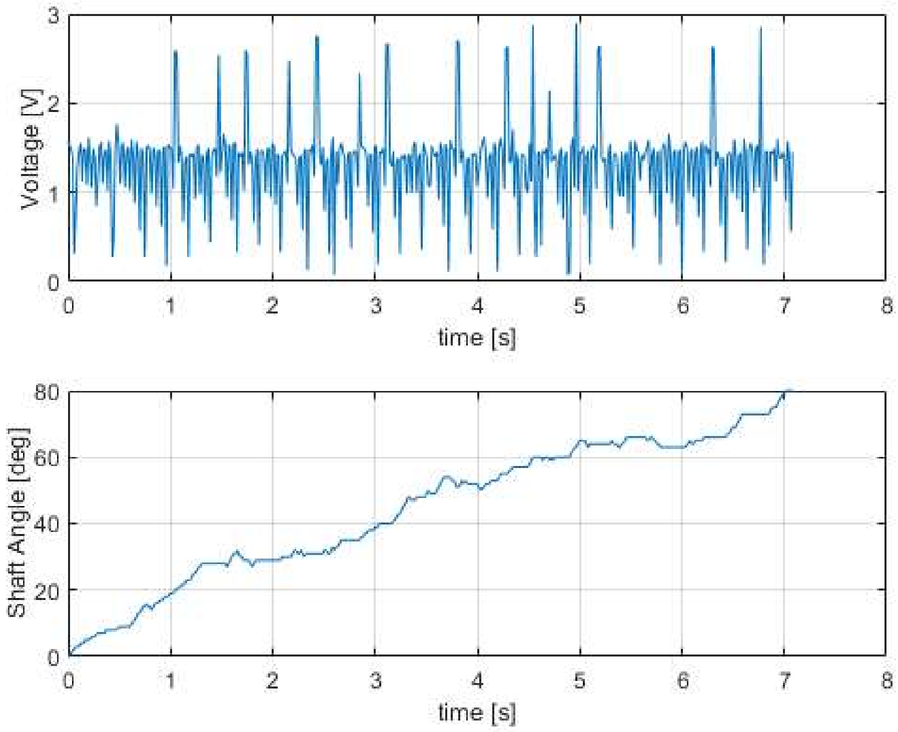
```
Maximum sample time [ms]:
    18

Minimum sample time [ms]:
    14

The average sample time is [ms]
   16.3180

Our sampling time is, Ts [sec]
    0.0160
```

Now, lets structure our data as iddata

```
DC_motor_DATA = iddata(outputShaftAngle, inputVoltage, Ts);
```

Visualize the data before starting system identification

```
figure(1)

subplot(2,1,1)
plot(time, inputVoltage)
grid on
xlabel('time [s]')
ylabel('Voltage [V]')

subplot(2,1,2)
plot(time, outputShaftAngle)
grid on
xlabel('time [s]')
ylabel('Shaft Angle [deg]')
```



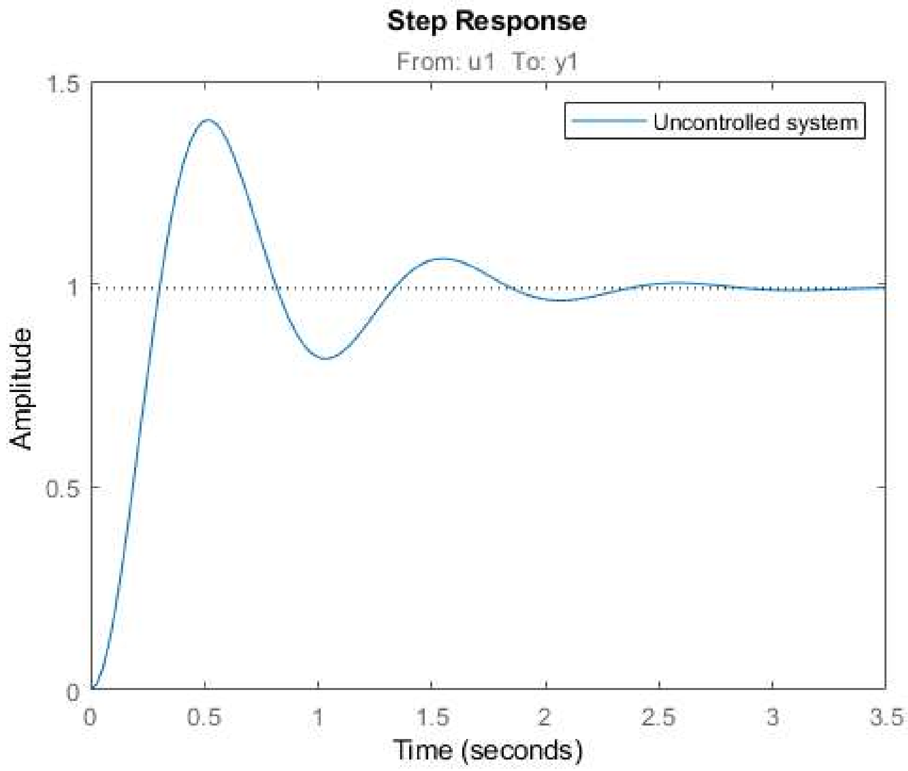
Going into system identification

```
Gp = tfest(DC_motor_DATA, 2, 0) %Estimating our system as a transfer function with no zeros and two poles.

% Alternatively go into System Identification app and try other model structures:
%systemIdentification
```

```
Gp =

  From input "u1" to output "y1":
          39.46
  ----------------------
  s^2 + 3.366 s + 0.4107

Continuous-time identified transfer function.

Parameterization:
   Number of poles: 2   Number of zeros: 0
   Number of free coefficients: 3
   Use "tfdata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using TFEST on time domain data "DC_motor_DATA".
Fit to estimation data: 86.47%
FPE: 8.26, MSE: 8.072
```

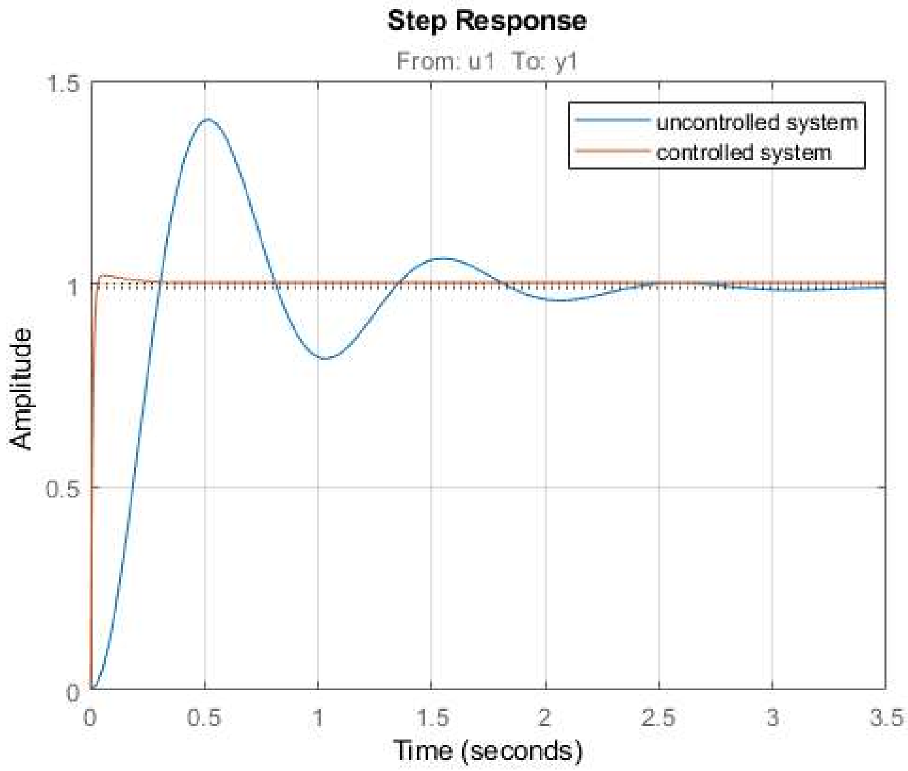
Now, let's try to tune a PID using our estimated system as a digital twin.

```
% First, we define our feedback transfer function, H
H = [1];

% For reference, we define an open loop system (uncontrolled) and have a look at the
% response
figure(2)
M = feedback(Gp, H);
step(M)
hold on
legend('Uncontrolled system')
```



We then define our PID

```
Kp = 20;     %Proportional
Ki = 5;     %Integral
Kd = 3;     %Differential

% Our control transfer function is
Gc = pid(Kp, Ki, Kd);

% And we can define our controlled system
Mc = feedback(Gc*Gp, H);

step(Mc)
grid on
legend('uncontrolled system' ,'controlled system')

%After running, try to change the three parameters Kp, Ki and Kd by marking the value -->
%right clicking --> select 'increment value and run section'

% Finally, try to run the PID controller on the Arduino :)
```



Published with MATLAB® R2020a