Part III

Dynamics and motion control
Control exercises 2016

Control design for the simulated DC-motor

The result of the exercises should be executable M-scripts with Simulink files. Simulation results should always at least show the controlled state, (**velocity or position**), the **reference signal**, the **error** and the **control signal** (voltage). Do not forget to plot the control signal it is very important since it gives a lot of information about the closed loop control performance..

Velocity Control:

1. Continuous time velocity control

For design and analysis you should use the linear model from exercise II.2 and for closed loop simulation in **Simulink** you should use a model which includes as much details as possible, i.e., the inductance from exercise II.1 and the static friction from exercise II.3.

P-control

Design a P-controller by using root locus, select a pole that gives a faster but not too fast closed loop response compared with the response of the motor without control.

- Check the amplitude and phase margins.
- Simulate the closed loop controller using a reference signal that i) is a step (What is a reasonable amplitude?) and ii) a sin wave (try different frequencies and observe the gain and phase between reference signal and motor speed). Try for example

$$v_{ref}(t) = 20\sin(0.2\mathrm{t})f \text{ and } v_{ref}(t) = 200\sin(10t)$$

as reference velocity.

PI-control

To get rid of the static error in the P controller design a PI-controller by pole placement, start by choosing both closed loop poles near to where the single closed loop pole of the P-controller was.

This is an excellent time to test the difference between error and output feedback. The error feedback will probably give an overshoot (depends al little on the location of the poles) for a reference step and the output feedback will give a step response without overshoot. Why?

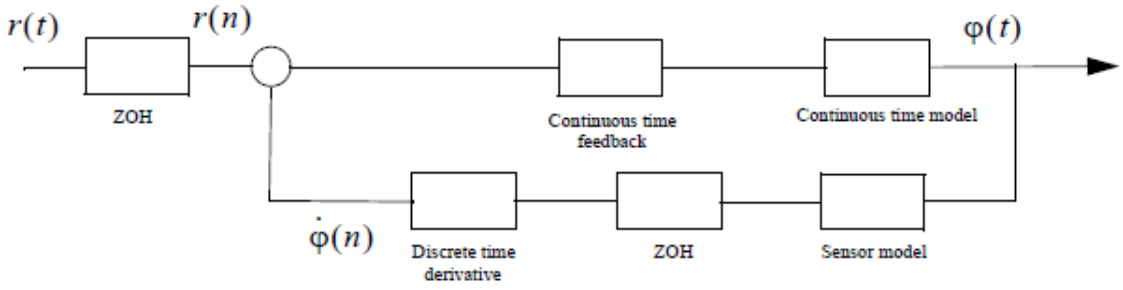
- Simulate and make an analysis as you did for the P-controller

Deliver the following for your final design:

1. Closed loop poles and zeros
2. Time response to reference step and sinewave reference

2. Discrete time controller

Before you actually design a discrete time controller you can analyze your continuous time design from exercise 1 by simulating it as a sampled system micro-processor implementation as shown below for the case with error feedback. This means that you also should include the sensor model (exercise II.4) with discrete time derivation of the sensor position. Use the Simulink option Sample time colors to see which blocks simulates continuous and which that simulate discrete.



- Use the designed P and PI-controllers from exercise 1 and simulate with sampling period $T_{s1} = 0.02$ and $T_{s2} = 0.002$. The encoder has a resolution of 1000 lines per revolution.

Does it work?

To understand the consequences of the discrete implementation of the controllers they have to be converted into discrete time models so that we can check the location of the discrete time closed loop poles.

- Convert the DC-motor model using zero order hold.
- Convert the PI controller using Tustins approximation. The P-controller does not need to be converted.

- Calculate and plot the closed loop poles. Are they within the unit circle? Are they on the positive or negative real axis?
  Tip, use the command, damp(), in Matlab.
- Do it for both sampling periods.

To make a good controller independent of sampling period must the design be made in discrete time instead of continuous time or at least must the desired closed loop poles be calculated with the rule of thumb that is, they should be 10 to 30 times slower than the sampling frequency.

Try that!

Position Control

3. Continuous time controller

Design P, PD and PID controllers using the same technics as with the velocity control design. That is, root locus and pole placement.

4. Discrete time controller

As above the controller should run on to different micro controllers where one has period $T_{s1} = 0.01$ and the other $T_{s2} = 0.01$.

5. Control of the hydraulic cylinder

Design a velocity controller using poleplacement for the cylinder, exercise II.6.

Linearize the non-linear model. Depending on the type of approximations done can you derive either a second or third order linear model. If you use a nonsymmetrical piston model with different cross sectional areas you will get a third order model and if you use a symmetrical with equal cross sectional areas a second order model.

Test if it works for both positive and negative piston velocities.

When you are satisfied, design a position controller in an outer loop round the velocity controller so that you get a cascaded control structure where the output of the position controller is the reference velocity for the velocity loop.

If the closed loop poles for the position loop are sufficiently slower compared to the closed loop poles of the velocity controller then it is possible to use the simple model

$$y = \frac{1}{s} v$$

If where *v* is the velocity and *p* the position. The standard rule of thumb is that loops should be separated with a factor of 10 in frequency.

A potential problem is that the position controller may generate very large velocity references. Instead of just generating a position reference as a step could you try with something “smoother” for example a ramp or a low pass filter on the position reference.

Make a discrete time approximation of the controllers. What is the longest sample period that you can select without losing to much performance.