Computerized control - RST design and implementation of a velocity servo for a DC motor

Kjartan Halvorsen

Physical setup demo due Tue 2018-05-01 17:30. Report due Tue 2017-05-08 by end of day.

Practical information

Objective To design and implement a two-degree-of-freedom RST controller for controlling the angular velocity of a DC motor. The controller should conform to given requirements on the properties of the closed-loop system.

Tasks The exercise is divided into three parts.

1. Hardware setup. The DC-motor is controlled using an input between -5V and +5V. It is thus not possible to control the motor velocity and direction of rotation using an H-bridge. Instead you will need to design and implement a circuit that can deliver $\pm$ 5V. You will also need to design and implement an analog anti-aliasing filter. Use an arduino microcontroller (or other of your choosing) as the controller in your system.
2. System identification. Send a suitable input signal to the DC-motor and record the response. Use the input/output data to identify a suitable, low-order model of the plant.
3. Design an RST controller based on your identified model. Perform some step-response tests to verify the performance of the closed-loop system.

Physical setup We will use a laboratory setup from TecQuipment consisting of a DC motor with sensors for angular velocity (tachometer) and angular position (optical encoder). The setup is contained in a black suitcase and can be checked out at the counter in the laboratory. See figure 1

Lab notes Write down on paper what you do and observe during the different experiments. Write the group name and date on each page. These notes should be scanned and added as an appendix to your report.

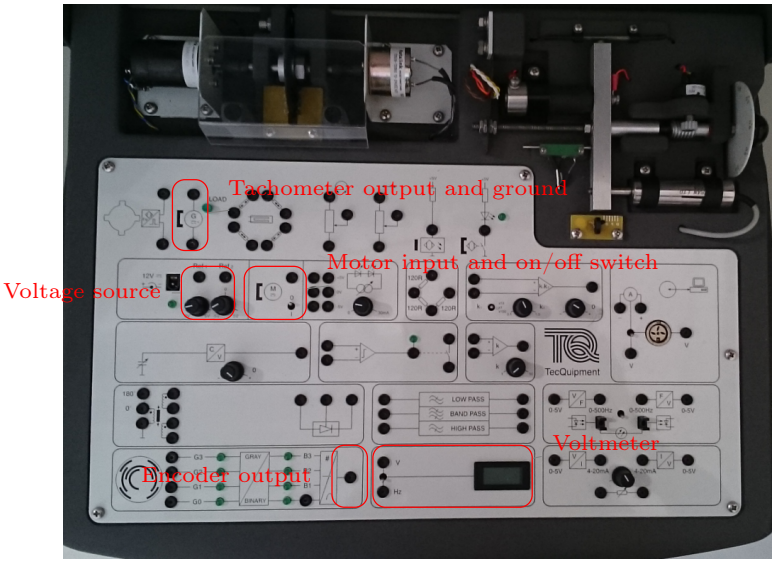

Figure 1: Relevant parts of the experimental device.

Specifications

- Settling time ($\pm 5\%$) of 1 second
- Overshoot of maximum 15%

In order to determine the desired **continuous-time** poles of your closed-loop system, you may use the following relations for a second-order dominant system

Settling time (5%) $t_s = \frac{3}{\zeta \omega_n}$

Overshoot and damping ratio

$$\zeta = \sqrt{\frac{(\ln \frac{PO}{100})^2}{\pi^2 + (\ln \frac{PO}{100})^2}},$$

where PO is the percent overshoot.

Observer poles (if needed) should be twice as fast as the closed-loop poles. Design a regular RST controller (not incremental).

Part 1 - Physical setup

Anti-aliasing filter

You will need to determine the sampling period h first, and then design a low-pass analog filter that will give good attenuation of signals above the Nyquist frequency. To keep

things simple, it is suggested that you use a second order filter. Implement the filter in hardware, using op-amps, resistors and capacitors.

Motor driver

You will need to design and implement a circuit that can deliver $\pm$ 5V, using an external power source, and a control signal (pwm) from the microcontroller.

Controller

The controller will run on your arduino. Send the control signal $u(kh)$ and the measured angular velocity (tachometer signal) $y(kh)$ to your computer so you can do system identification and plot results.

Part 2 - System identification

Test setup

The idea is to send a suitable input signal to the DC motor and record the response. A pseudo-random binary sequence (PRBS) is often a good choice for the input signal. The amplitude should be sufficient to get good response from the motor. Since you are using a circuit to generate the input voltage to the DC motor, it makes sense to include this in the model of the plant, since it may have some non-neglectble dynamics. Thus, if your output signal from the arduino is a pwm analog signal in the range 0-5V, where 2.5V will give 0V to the DC motor, it makes sense to let your control signal be the deviation from 2.5V. Hence you can write the signal as

$$\bar{u}(t) = 2.5 + u(t)$$

where $u(t)$ is your control signal.

The output from the the plant $y(t)$ is the voltage from the tachometer sensor, filtered through the anti-aliasing filter. Thus, also the anti-aliasing filter will be part of the plant model.

Model selection and identification

In order to make the design of the controller simpler, assume a low-order model for the plant, for instance a first- or second-order model with delay (the delay is typically mostly due to the anti-aliasing filter). Do a long test (about one minute), so that you can divide the data set in modelling data and validation data.

Figure 2: Closed-loop system with two-degree-of-freedom controller

Make a simple arduino program that will perform the test on the system. Recording the input/output data can be done on a regular computer if you send the data over serial link. Use the system identification toolbox in matlab to identify your discrete-time model. Verify your model using the validation data. Include the plot of the model-output $\hat{y}(kh)$ and the plant output $y(kh)$ that shows how good the model is.

Part 3 - Controller design and implementation

Determine the desired closed-loop poles

From the specifications, determine the desired continuous-time poles and transform these to the discrete-time.

Design a 2-DoF controller

Assume a structure of the controller as given in figure 2. The controller is given by

$$R(q)u = -S(q)y + T(q)u_c.$$

With the plant-model

$$A(q)y = B(q)u$$

we get the following difference equation for the closed-loop system

$$\big(A(q)R(q) + B(q)S(q)\big)y = B(q)T(q)u_c.$$

Determine the order (as low as possible) of the controller polynomials $R(q)$ and $S(q)$ and solve the diophantine equation

$$A(q)R(q) + B(q)S(q) = Ac(q)$$

for R and S.

Deadzone compensation

The DC-motor has deadzone, meaning that the motor will not move unless the voltage is above a certain value. Determine this deadzone by connecting the reference voltage to the motor as seen in figure 3. You can read off the voltage from the LCD-display.

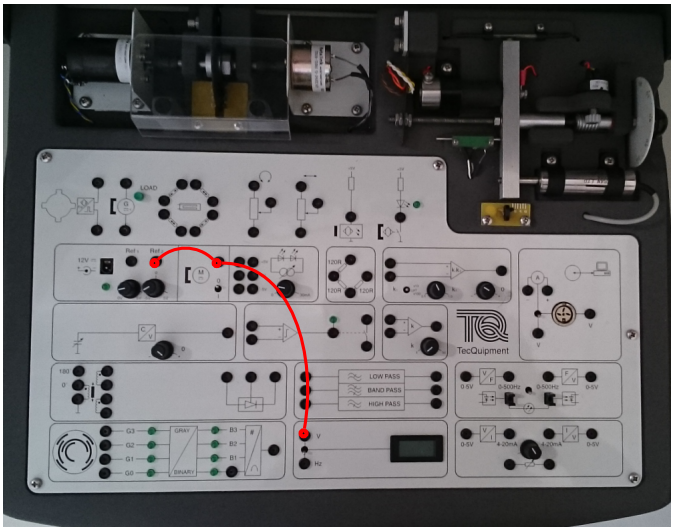

Figure 3: Connections for checking the deadzone of the DC motor.

The deadzone can be compensated by inverting the deadzone function. In practice this means adding (or subtracting) to the control signal, the offset corresponding to the deadzone. Your controller algorithm should do this compensation just before writing the value to the analog output of the arduino. The code should do something like this

```
const float deadzonePos = 0.8; # Or what you determine
const float deadzoneNeg = 0.9;

float y = current_position(); # Read the angular position signal y
float uc = current_command(); # Read the command input

float u = next_control_signal(y, uc); # Calculate the control signal u

if (u>0) {
   u = u + deadzonePos;
else {
   u = u - deadzoneNeg;
}

write_control_signal(u); # Write control signal to output channel
```

Implementation and tests

Implement your RST controller on the microcontroller. Run some step responses on the closed-loop system. Plot the results from these, and verify that the closed-loop system

satisfies the specifications.

Report

Document your work in a laboratory report. Write briefly how you solved the task. Include graphs to illustrate your results. The report should have the following sections

1. Introduction
2. Circuit design and physical setup
3. Plant model and identification
4. Controller design
5. Implementation and results
6. Conclusion
7. Appendix: Your lab notes.