

Adaptive Control

Laboratory experiment 2

Indirect Self Tuning Control

Department of Automatic Control
Lund Institute of Technology

1. Introduction

The goal of this experiment is to give some empirical experience of indirect adaptive control. An adaptive control algorithm based on recursive least-squares estimation and pole placement design will be explored. The process is a DC-servo with a flywheel that will be controlled to follow a desired angular position. Adaptive control of the process will demonstrate the role of different choices of some design variables for an indirect self tuning regulator.

A PC equipped with AD- and DA-converters will be used to perform design calculations and to implement the adaptive controller. The controller is implemented in the Matlab/Simulink environment and Matlab is also used for design calculations.

Preparation

Read Chapter 2 in Adaptive Control (in particular Section 2.2 and 2.3) on recursive identification, Chapter 3 on self tuning control and Chapter 11.8 on implementation where the design method is described. Work through the three exercises marked *Preparation* in this manual.

One preparation involves writing Matlab code to complete the Matlab function used in the RLS-algorithm. The partially implemented Matlab function is given in Appendix B. It can be downloaded from the webpage of the course, <http://www.control.lth.se/~FRT050>. At the lab session the completed Matlab function should be available for evaluation.

Pole Placement Design

The controller is given by

$$R(q)u(k) = T(q)u_c(k) - S(q)y(k). \quad (1)$$

The polynomial coefficients are calculated using the design procedure in Adaptive Control section 3.2 (or Computer Controlled Systems chapter 5). No process zeros are canceled, therefore we choose $B^+(q) = 1$ and $B^-(q) = B(q)$. An integrating controller is also required. The closed loop transfer function from reference $u_c(k)$ to output $y(k)$ is

$$H_c(q) = \frac{A_m(1)B^-(q)}{B^-(1)A_m(q)} \quad (2)$$

where $A_m(q)$ is specified. The observer polynomial $A_o(q)$ is also specified but is canceled in (2).

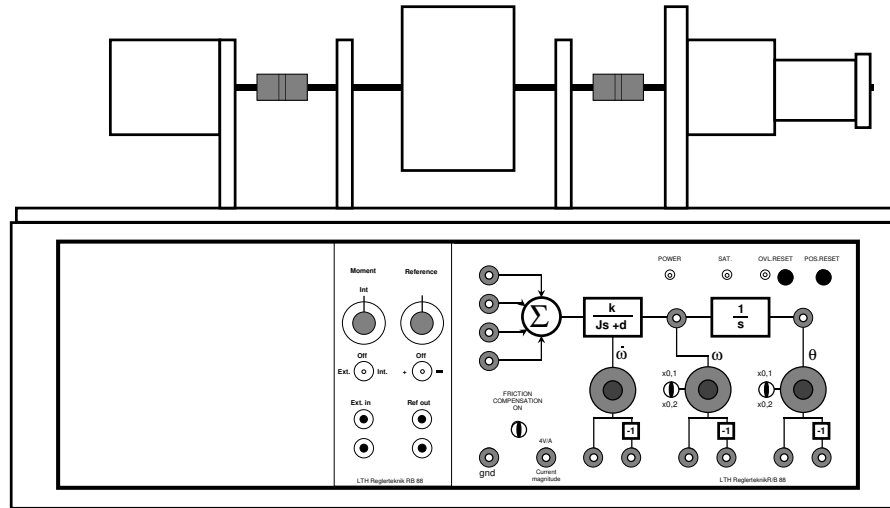


Figure 1 Front panel of the DC-servo.

Implementation

The adaptive controller is implemented in Matlab/Simulink. Each sampling interval the following steps are performed:

- Control: Read $y(k)$ from ADC. Calculate and send $u(k)$ to DAC.
- Estimation: Update $\theta(k)$ using $\varphi(k)$ in the RLS-algorithm.
- Design: Solve the Diophantine equation to calculate new polynomials R, S and T in the controller.

To avoid large transients when the adaptive controller is started the controller output u is limited for the 20 first samples.

2. Position Control of DC-Servo

The process in Figure 1 is a DC-servo that should be controlled to follow a desired angular position. The process interface is customized to fit the AD- and DA-converters on the computer. All signals are in the interval ± 10 V. The position y is given by

$$J\ddot{y} = -d\dot{y} + ku + v \quad (3)$$

where u is the control signal and v is an optional disturbance that can affect the system. The disturbance is activated by a switch on the process front panel. Process variations on the damping d and the moment of inertia J may be introduced using the potentiometers on the front panel together with local feedback.

The nominal continuous time transfer function from u to y is

$$G(s) = \frac{11.2}{s(s + 0.12)} \quad (4)$$

The corresponding pulse transfer operator is on the form

$$H(q) = \frac{B(q)}{A(q)} = \frac{b_1q + b_2}{q^2 + a_1q + a_2} \quad (5)$$

(Hint: Use Matlab to obtain the numerical values in the pulse transfer function.)

Preparation 1: Write down the algorithm for recursive least square estimation. Specify the elements of $\theta(k)$ and of $\varphi(k)$ for the model (5).

Preparation 2: Complete the Matlab function RLS.m by filling in the necessary Matlab code in Appendix B at the indicated positions.

The closed-loop characteristic polynomials $A_m(q)$ and $A_o(q)$ are conveniently defined as the discrete-time counterparts to the continuous time polynomials

$$A_{mc}(s) = s^2 + 2\zeta_m\omega_m s + \omega_m^2 \quad (6)$$

$$A_{oc}(s) = s^2 + 2\zeta_o\omega_o s + \omega_o^2 \quad (7)$$

We will use the sampling interval $h = 0.1$ s here. With fixed relative damping, e.g, $\zeta_m = 0.7$ and $\zeta_o = 0.7$, the closed looped system is parameterized by ω_m and ω_o . As a *nominal design* we choose $\omega_m = 5$ and $\omega_o = 7$.

Preparation 3: Use Matlab to determine the fixed discrete time-integrating controller (1) for the process (4) when $\omega_m = 5$, $\omega_o = 7$, and $h = 0.1$. (The specification are conveniently transformed to discrete-time equivalents by using the command `polyc2d`). Determine the Bode diagram and the Nyquist diagram for the loop gain $L = BS/AR$.

Matlab hints for the preparation exercises In the preparation exercises the following Matlab routines may be useful:

`bode`, `c2d`, `grid`, `margin`, `nyquist`, `polyc2d`, `rstd`, `tf`.

Also the following remarks can be of help:

- use the command `help <routine>` in Matlab for help,
- a polynomial $A(q) = a_0q^2 + a_1q + a_2$ is represented in Matlab as a vector: `A = [a0 a1 a2]`,
- the numerator of a pulse transfer function $H(q)$ can be accessed as `H.num{1}`,
- the denominator of a pulse transfer function $H(q)$ can be accessed as `H.den{1}`.

Experiment 1: Fixed Control

Use a fixed controller with $R(q)$, $S(q)$, and $T(q)$ polynomials obtained from Preparation 3. Make sure that the polynomials R, S and T are available in Matlab's workspace. How does the system behave for reference steps and load disturbances? How does process variations affect? Compare with controllers designed with $\omega_m = 10$ and $\omega_o = 14$, and with $\omega_m = 1$ and $\omega_o = 1.4$.

Experiment 2: Adaptive Control

Find out what the default choices in the program are for

- the forgetting factor λ
- the initial covariance-matrix

Investigate the influence of these variables on

- the initial transient,
- the convergence of estimated parameters,
- the closed loop performance for reference steps and load disturbances,
- the P -matrix.

Start with fixed $P_0 = 100$ and vary $\lambda = (0.8, 0.9, 0.99, 0.999)$. Then use $\lambda = 0.99$ and change $P_0 = (1, 10, 100, 1000)$.

Use the nominal closed-loop specifications and let the reference input be a square wave with the period 10 s and the amplitude 1.5. Choose an initial process model

$$H(q) = \frac{2q + 2}{q^2 - 1.5q + 0.7}$$

in the Model-Design menu. This gives a model that is not too close to the expected. Notice that you have to stop the algorithm when changing parameter values. By pressing the position reset button on the servo each time the adaptive algorithm is started, the initial transient due to the control error is reduced.

Experiment 3: Process Variations

The forgetting factor λ determines the ability to track process variations. Run the adaptive controller until the parameters of the process model have converged. Then change the process through a local feedback on the process panel. Investigate

- the closed-loop performance,
- the estimated parameters,
- the P -matrix.

Try e.g., $\lambda = (0.8, 0.9, 0.99, 0.999, 1.00)$.

Experiment 4: Choice of Regression Filter

The regression filter $B_f(q)/A_f(q)$ determines the frequency range where the frequency responses for the estimated model and the real process should be close.

A second order regression filter is implemented in the program. Its denominator $A_f(q)$ is the discrete-time counterpart to the continuous-time characteristic polynomial

$$A_{fc}(s) = s^2 + 2\zeta_f \omega_f s + \omega_f^2$$

What filter has been used so far?

Investigate the closed loop behavior for band-pass filters with different $\omega_f = (0.1, 1, 5, 20)$. Study the Bode diagram of the estimated model. Try also a low-pass filter with $\omega_f = 5$. How does a step load disturbance affect?

Experiment 5:

Use a non-integrating controller together with non-differentiating regression filter. Is it possible to make the control error zero in stationarity? What drawbacks are there with this approach?

- Double click the grey `Signal Generator` box to choose amplitude, frequency and wave form of the reference signal.

- Double click the yellow RST regulator block to switch between a fixed RST controller and an adaptive RST controller. After an execution the current $R(q)$, $S(q)$ and $T(q)$ polynomials are available in Matlab's workspace (in the variables R, S and T).

Start/Stop controller Choose Simulation | Start (or press ctrl-t) to start and stop the execution. By pressing the position reset button on the servo each time the controller is started, the initial transient due to the control error is reduced. Some signals can be viewed in real time during execution. Double click the grey box with the corresponding signal label.

Plot results

- Double-click the grey `Plot Result` box. A figure window with the results of the last controller execution is opened. Plot legends can be turned on and off with the View | Legends menu command. It is a good idea to keep old plot windows for later comparisons.
- Double-click the grey `Bode B/A` box to display the Bode-diagram for the estimated model B/A .
- Double-click the grey `Bode Bf/Af` box to display the Bode-diagram for the regression filter B_f/A_f .

A note on real-time performance

The DC servo is sampled periodically and the control signal is applied to the process immediately with no time delay after each sampling instant. These are ideal *timing constraints*. In practice this is hard to achieve. Failure to meet the real-time constraints may lead to severe performance deterioration. In the Matlab/Simulink environment the Graphical User Interface (GUI) under some circumstances consumes so much computational power that the timing constraints are violated. This often is the case the first few seconds of a real-time simulation, or if the user invokes operations such as moving, opening or closing windows in the GUI. The Simulink models includes a measure of the timing constraints violations in the *jitter* signal. When the jitter is close to zero the timing is close to ideal. When the jitter is equal to one there is a delay of one sampling period at the sample instants. Be careful with working with the GUI when a simulation is running. Use keyboard shortcuts instead of menu commands whenever possible.

B. Matlab function

```
function [new_theta,new_P] = RLS(P,phi,theta,y,lambda,n)

% [K,new_theta,new_P] = RLS(P,phi,theta,y,lambda,n)
%
% A M-function performing least-squares estimation with
% exponential forgetting factor.
```

```

%
% Inputs:
% P      -- covariance matrix
% phi    -- regression vector
% theta  -- parameter vector
% y      -- process output
% lambda -- forgetting factor
% n      -- total number of parameters to be estimated
%
% Outputs:
% new_theta -- new parameter vector
% new_P     -- new covariance matrix
%
% lastedit 990802

% compute new estimate and update covariance matrix

new_theta =< insert code here > ;
new_P = < insert code here > ;

```

C. Experimental setup

The wiring diagram is shown in Figure 3. The wiring is carried out prior to the laboratory experiment because of the restricted time available.

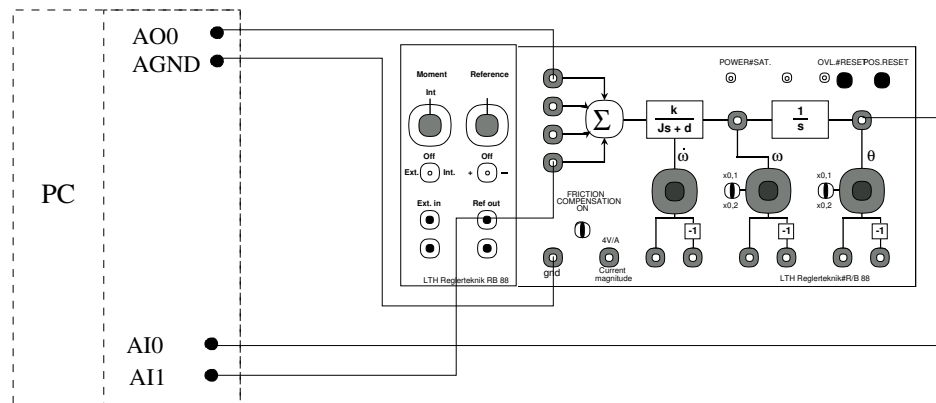


Figure 3 Wiring diagram.

Document history:

Created by Michael Lundh

Revised August 1999 by Johan Alfvén and Mats Åkesson

Revised September 2003 by Henrik Sandberg

Revised June 2004 by Stefan Solyom