FRT 041 System Identification
Laboratory Exercise 3

Ulf Holmberg
Revised: Kjell Gustafsson
Karl Henrik Johansson
Anders Wallén
Johan Nilsson
Rolf Johansson
Johan Bengtsson
Maria Henningsson

Department of Automatic Control
Lund Institute of Technology
Lund University
October 1987

Revised: September 1989, March 1994,
March 1995, February 1996,
February 1997, February 1999,
March 2005, February 2011

Identification and Controller Design

The purpose of identification is often to get a model that can be used for controller design. What is aimed at is a low order model that captures the main properties of the process, and which can be used for analysis and synthesis. It is important that the model is accurate for a frequency band around the crossover frequency, while higher order dynamics and weak nonlinearities often can be discarded. Also the low frequency properties are often less important, since the closed loop behavior at these frequencies is determined by having integration (high gain) in the controller. Still, the sign of the static gain has to be known.

The model is valid only for a limited frequency range, and probably uncertain also within this range. Therefore when doing the design of the controller a synthesis method resulting in a robust controller should be used, i.e., the performance of the controller should not be affected by small changes in the model.

In this laboratory exercise an identification experiment will be done. Using MATLAB a model will be estimated and used to design a controller. The controller is then tested on the real process.

Throughout the laboratory exercise MATLAB is used. This manual contains many references to MATLAB functions. Some of these are standard MATLAB functions while others were written to facilitate the exercise. Whenever you are in doubt about what a certain function does, use `help`.

Preparation
Read through "A Manual For System Identification"[Andersson *et al.* (1998)] which was used in lab 2 and bring it to the laboratory exercise.

1. The process

The process to be designed is a little tricky to control. It consists of a rectangular plate hanging in one of its edges, see Figure 1. The plate is mounted such that it can swing back and forth. A weight is mounted on one side of the plate to make it deflect from the vertical plane. A fan is positioned a short distance from the plate and used to blow an air stream on the plate, thus affecting its position. The angle between the plate and the vertical plane is measured and act as output from the process. It is possible to control the plate angle using the voltage to the fan motor as control variable. The process has the following properties

- There is a time constant in the fan motor and hence the air stream is not immediately affected when varying the motor voltage.
- The process contains a time delay since it takes some time for the air stream to reach the plate.
- The plate acts like a pendulum resulting in a lightly damped resonance.
- Turbulence causes noise which affects the process.

Try to control the process manually. Use a potentiometer to vary the voltage to the fan motor. Try to estimate the time constant of the process and the

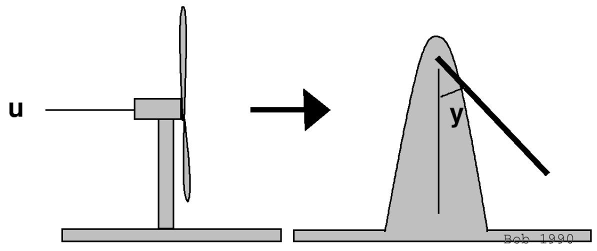


Figure 1 The process.

frequency of the resonance so that you later may judge if the identified model is reasonable.

2. The identification experiment

You are to perform an open loop identification experiment on the process. Start MATLAB and type `initlab3` at the command prompt. This sets up MATLAB and opens the SIMULINK model `logger`, which will be used to log data, see Figure 2. The model generates a PRBS (pseudo random binary sequence) signal which is used as input to the process. The excitation signal and the process output are logged and saved in the workspace as `u` and `y` for later processing in MATLAB.

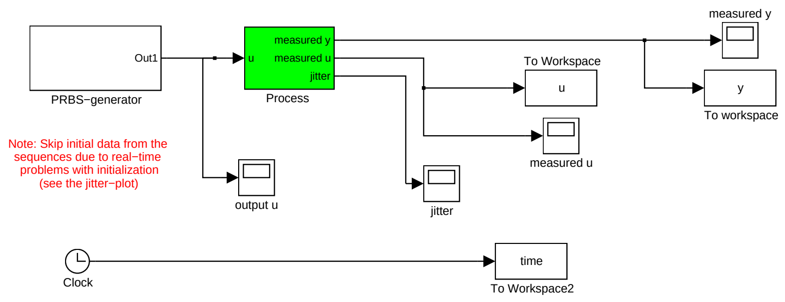
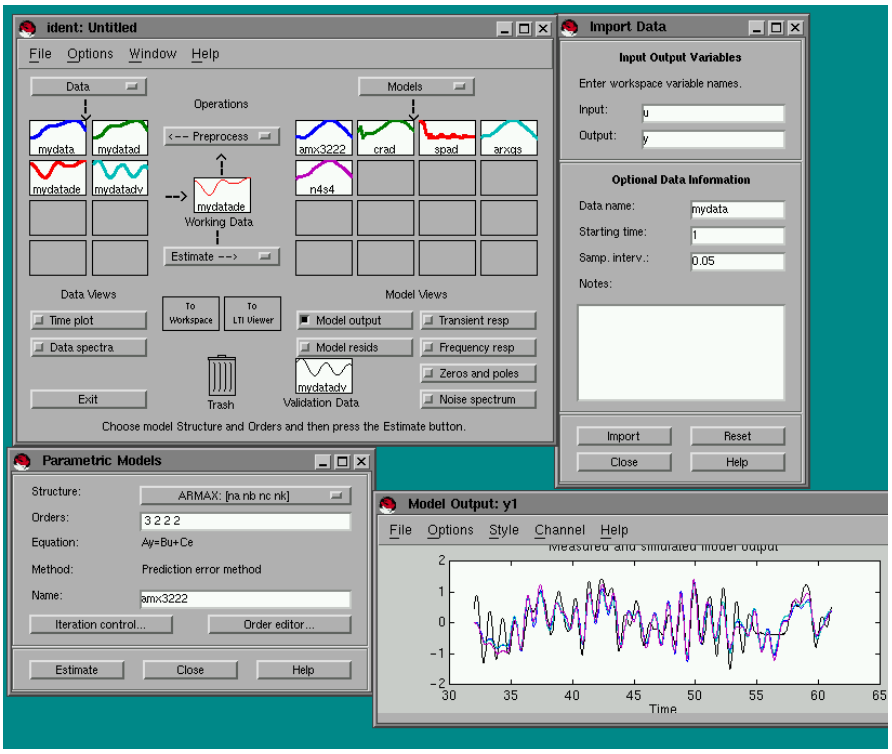


Figure 2 Simulink model for logging data.

The plate is to be controlled approximately around its downright position, i.e. a plate angle slightly greater than zero. This corresponds to a certain voltage fed to the fan motor since the weight on the plate causes it to deflect. Adjust the mean value of the excitation signal to get a small positive plate angle. One easy way to achieve this is to use an external adjustable voltage added to the signal to the fan motor.

When the mean value is adjusted, set the amplitude of the excitation signal to 0.5 V. According to Åström and Wittenmark (1997) it is reasonable to choose the sample period h such that $\omega h \approx 0.2 - 0.6$, where ω is the resonance frequency of the plate estimated above. Thus we may choose $h = 50$ ms. Further we decide to collect 1200 data points, and to set the PRBS period to 10. Start the logging and consider the logged data in the scopes.



Choose model Structure and Orders and then press the Estimate button.

Enter workspace variable names.

Input: u

Output: y

Data name: mydata

Starting time: 1

Samp. interv.: 0.05

Structure: ARMAX: [na nb nc nk]

Orders: 3 2 2 2

Equation: Ay=Bu+Ce

Method: Prediction error method

Name: amx3222

Figure 3 GUI for System Identification Toolbox.

3. The identification

The System Identification Toolbox in MATLAB will be used to estimate a model of the process. The identification may be done *either* by writing the commands below *or* by using the graphical user interface of the System Identification Toolbox, see Figure 3 The graphical user interface is started by writing at the command prompt

```
ident
```

The rest of this section describes command-line identification. You can follow the same procedure in the graphical user interface.

Begin by looking at the data by running

```
plot([y u])
```

Do the signals look all right? What transients should be neglected? Pick out the process output and the excitation signal and remove the bias using

```
z = [y u];
z = detrend(z,'constant');
```

which gives a matrix with two columns y and u. We check in what frequency interval a good model might be estimated by plotting the coherence function

$$\Gamma_{yu}(\omega) = \frac{|S_{yu}(\omega)|}{\sqrt{S_u(\omega)S_y(\omega)}}$$

using the command

```
mscohere(u,y,[],[],[],1/h);
```

In a previous laboratory exercise we did frequency response analysis. An alternative is to do spectral analysis in order to get an estimated frequency response. The quality of this estimate will highly depend on the length of the Hamming window used. This makes spectral analysis sometimes hard to use in practice. It has been shown that for our data a window length of about 100 is good. Try

```
g = spa(z,100,[],[],h);
plot(g);
```

It is often useful to split the data into two sequences; one for identification and one for verification:

```
nz = size(y,1);
z1 = iddata(y(1:nz/2),u(1:nz/2),h);
z2 = iddata(y(nz/2+1:nz),u(nz/2+1:nz),h);
z1 = detrend(z1,0);
z2 = detrend(z2,0);
```

Use the function `armax(z,[na,nb,nc,k])` to estimate an ARMAX model in the backward shift operator (q^{-1}) according to

$$\mathcal{M}: \quad \begin{aligned} (1+a_1q^{-1}+\cdots a_{na}q^{-na})y(t) &= (b_1q^{-k}+\cdots b_{nb}q^{-k-nb+1})u(t) \\ &\quad +(1+c_1q^{-1}+\cdots c_{nc}q^{-nc})e(t) \end{aligned}$$

Note that `nb` corresponds to the number of b-parameters and not the degree of the B polynomial. The parameter `k` denotes the time delay in the system. Remember to have `k` > 0 to get a causal model without any direct term. Do you have any ideas about initial values for `[na,nb,nc,k]`?

The result of the command `armax` is represented on a special form, the `theta`-form. The command `present` will list the parameter values, their variance, the value of the loss function, and the Akaike FPE (final prediction error) value. An example is

```
th3222 = armax(z1,[3,2,2,2]);
present(th3222)
```

Use the FPE and the variance of the parameter values to determine a suitable time delay (`k`) and model order (`na`, `nb` and `nc`). Try to find a good model with an order as low as possible (the order equals max(`na`,`nb+k-1`)). A good way to verify the model is to compare its output signal with the process output. This can be done as follows

```
ym = idsim(z2.InputData,th3222);
t = h*[1:1:length(z2.OutputData)];
plot(t,z2.OutputData,t,ym);
```

Note that we use `z2` and not the data that were used for model estimation when we simulate the model.

You should also take a look at the pole-zero configuration of the model. A too large model order may show up as poles and zeros that almost cancel. Use

```
pzmap(th3222)
```

The final model, i.e. the one that is most satisfactory, should be used to design a controller. This is also done in MATLAB. The macros that do the design need a process model on the form $B(q)/A(q)$, with $B(q)$ and $A(q)$ being polynomials in the forward shift operator. The two polynomials have to be extracted from the `theta`-form and converted to forward shift operator form. `A` and `B` in backward shift operator are derived by

```
A = th3222.a; B = th3222.b;
```

Rewrite the transfer function into forward shift representation on a piece of paper. MATLAB commands for what you have done are

```
A = [A zeros(1,k+nb-1-na)]
B(1:k) = [],     B = [B zeros(1,na-k-nb+1)]
```

Make sure you understand that these commands convert from backward to forward shift. Compare the zeros of `A` with the expected closed-loop poles from the introduction.

4. The controller design

The design method that is going to be used is pole placement. It may be hard to decide where to place all the poles, so for simplicity we will choose a pole pattern and only vary its distance from the origin. This is equivalent to restricting the time response of the closed loop system to a certain form, and then only vary its "speed" (why?). We will vary the desired "speed" and try to evaluate the robustness of the resulting closed loop system. Finally, some promising designs will be stored for future tests on the real process.

Here is a step by step description of the design method.

1. First look at the frequency response of the model to try to get an idea of how much it is possible to demand from the closed loop system. Plot the Bode diagram using

   ```
   bode(th3222)
   ```

   As one expects there is a large resonance in the Bode diagram, corresponding to the plate acting as a pendulum. The frequency of this resonance tells about the natural frequency of the open loop system and gives a hint about what to expect from the closed loop system. The controller should take care of the resonance and damp it. Trying to get a closed loop bandwidth differing much from the resonance frequency will require a large control effort and a very accurate model. (Compare with how the closed loop poles are moved in a root locus plot when a proportional controller is used.) What do you regard as a reasonable range for the closed loop bandwidth?

   Plot also the Nyquist curve of the model using

   ```
   nyquist(th3222)
   ```

   Is proportional feedback sufficient to get a stable closed loop system with reasonable performance?

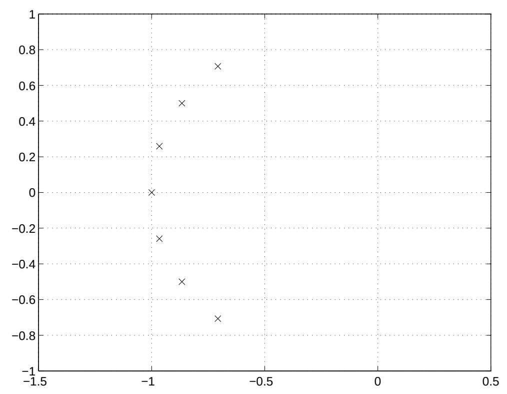


Figure 4 The continuous time pole pattern used in the design.

2. Pole placement will be used as design method, see Åström and Wittenmark (1997). The desired characteristic polynomial A_m and the observer polynomial A_o have to be chosen. To get a causal controller it must hold that

$$\deg A_o \geq 2\deg A - \deg A_m - \deg B^+ - 1$$

where B^+ is the canceled (stable) part of B. For simplicity lets decide not to cancel any zeros, i.e. $B^+ = 1, \quad B^- = B$, and choose $\deg A_m = \deg A \equiv n$. Then

$$\deg A_o \geq n - 1$$

Choosing $\deg A_o = n - 1$ gives a controller with direct term. To get an integrator in the controller, the degree of the observer has to be increased (why?). Thus

$$\deg A_o \geq n$$

It may be hard to relate discrete time poles to the properties of the closed loop system (at least the author believes this is hard). We will therefore relate the choice of A_m and A_o to continuous time poles. For simplicity choose the poles equally spaced on a segment of a circle in the left half plane of the s-plane. The radius of the circle is ω_m and half the opening angle is 45° (a kind of Butterworth configuration, see Figure 4). The observer poles are chosen in the same way but at twice the distance from the origin, i.e. $\omega_o = 2\omega_m$. The continuous time poles are translated to discrete time through $z = e^{sh}$. For a non-integrating controller the commands are

```
Amc = polybutt(n,wm,45);

Aoc = polybutt(n-1,2*wm,45);

Am = real(poly(exp(roots(Amc)*h)));

Ao = real(poly(exp(roots(Aoc)*h)));
```

There is only one parameter, ω_m, to vary in the design. It corresponds to the closed loop bandwidth, and we will choose it in relation to our observations in step 1.

The choice of pole pattern is naturally just a suggestion. It is probably possible to make a better design by choosing a different pattern. If you have time, try other configurations, e.g. multiple poles on the real axis.

3. The controller is calculated by solving the DAB-equation (Diophantine-Aryabhatta-Bezout)

$$AR_1 + B^- S = A_m A_o$$

with $B = B^+ B^-$ and $B_m = B'_m B^-$. The controller polynomials are given by

$$\begin{aligned} R &= R_1 B^+ \\ S &= S \\ T &= t_0 A_o B'_m \end{aligned}$$

where t_0 is used to adjust the static gain of the closed loop system to one. In MATLAB the calculations are done using (do `help rstd`):

```
[R,S,T] = rstd(1,B,A,1,Am,Ao,Ar)
```

The polynomial A_r is forced into R when solving the DAB-equation. This makes it possible to include integral action in the controller, i.e. $A_r = q - 1$ or in MATLAB notation `Ar = [1 -1]`. If no integrator is wanted just put `Ar = 1`.

4. When the controller has been calculated it needs to be evaluated. This can be done by plotting the Nyquist curve of the loop transfer function $G_o(q) = B(q)S(q)/A(q)R(q)$. If the Nyquist curve passes close to -1 our design is probably not very good. Naturally, the controller stabilizes the model, but since the model does not exactly describe the true process it may very well be that the real system will be unstable. Therefore try to find an ω_m that gives a reasonable closed loop bandwidth, but without having a Nyquist curve passing too close to -1. If this can be satisfied it is likely that the controller will perform well even if the true process should differ slightly from our model. When interpreting closeness to -1, think in terms of gain and phase margin. Try to get a loop with gain margin approximately equal to 2 and phase margin approximately equal to 60°.

   Plot the Nyquist curve of $G_o(q)$

```
nyquist(series(tf(S,R,h),tf(B,A,h)));
```

Evaluate the controller first by simulation. To save a controller for later use, do

```
save regname R S T
```

This command saves your controller in a file `regname.mat`. Of course, the controller should be causal. Check the polynomial degrees so that this requirement is fulfilled.

Iterate the design steps until a couple of good controllers are found. At least, try to find two with integration and two without integration.

5. Testing the controller

We are now ready to test the controllers designed above. If not already closed, close the logger-model and then type `lab3_controller` to bring up the SIMULINK model which will be used for control, see figure 5. The model consists of a reference generator, a controller on RST-form, the I/O connection to the process, and some scopes for displaying the signals of the system.

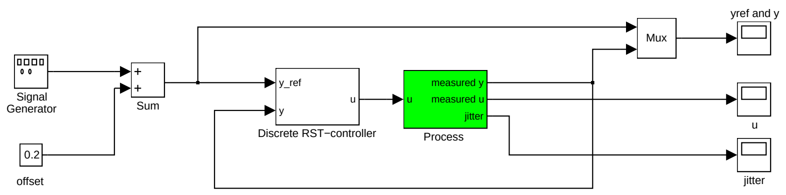


Figure 5 Simulink model for real-time control.

Start by choosing a square wave with mean 0.1 V, amplitude 0.05 V, and period 20 s as reference signal. Load a controller in workspace by executing

```
load regname
```

Finally, test the performance of the controller by varying the process parameters: change the distance between the fan and the plate, or change the weight on the plate. Compare your different controllers. Do they behave as you expected from the design phase above? Which controller would you prefer?

6. References

Andersson, L., U. Jönsson, and K. H. Johansson (1998): "A manual for system identification." In *Laboratory Exercises in System Identification*. KF Sigma i Lund AB. Department of Automatic Control, Lund Institute of Technology, Box 118, S-221 00 Lund, Sweden.

Åström, K. J. and B. Wittenmark (1997): *Computer-Controlled Systems*. Prentice Hall.