Uppsala University, Systems and Control Group 1992 / AE, AM
Revised 1993 / AM, AE, 1998 /PSA, 1999/BC-PSA, 2001/SS, 2003/RA

System Identification - F4
Modelling and Control of Environmental Processes - W4

Instruction of the laboratory work:

IDENTIFICATION OF A FAN PROCESS

Preparation: Carefully read this instruction document and the appropriate chapters in the course literature, see also Section 3.

Name		Instructors comments
Class	Year of registration	
Carry-out date	Group	
Approval date	Sign	

Contents

1 Objectives

The goal of this laboratory assignment is to give some practical experience of system identification procedures. Different identification methods will be studied, and models will be validated. The lab consists of the following main parts:

- **Basic experiments**
  These experiments are done to decide some basic properties of the process such as the linear range of the process and a suitable sampling rate for measurements.
- **Non-parametric identification**
  By looking at the frequency domain properties of the system, further conclusions on sampling interval, system order etc can be drawn.
- **Influence of different inputs in ARX modelling**
  Different inputs may have different impact on the accuracy of the parametric models. The purpose with this task is to decide a suitable input signal for the process.
- **Parametric identification**
  The purpose with this task is to estimate the best possible parametric model of the process with respect to several different test criteria. In particular, we will study ARX and ARMAX models of different orders.
- **Control of the process**
  A common purpose with system identification is to use the model for controller design and hence we are satisfied with a model that is sufficiently good to be used for control design. A working controller can then be seen as the final validation of the estimated model. The last task in this lab includes the design and test of a pole placement controller using a model obtained from the system identification exercises.

As our tool, we have MATLAB including the System Identification Toolbox (and a number of special purpose MATLAB functions written for this laboratory exercise). It is important that you are familiar with the necessary background material and the MATLAB-functions used (The most important ones are listed in the appendix of this instruction document). Observe that you can also use the command `help` *function name* in the lab to get information on how the functions are used.

2 Examination

In order to pass the lab you need to answer the questions in this report. We recommend that you save the important plots and discuss the results with the lab supervisors. Take notes during the lab and write detailed comments after you have conducted all tasks.

3 Preparations

The necessary background for this laboratory assignment can be found in:

W4 : chapters 8-10 and 13 of *Modellbygge och Simulering.*
F4 : chapters 5, 6, 7, 11 and 12 of *System Identification [SI].* In particular, the general model structure described in section 6.2 of *SI* and the practical aspects described in chapter 12 should be studied.

It is also necessary to study the System Identification Toolbox of MATLAB in advance. You should *at least* know how all MATLAB functions described in Appendix A are used. Also, carefully read the instruction in advance, and try to determine (approximately) a suitable model order for the lab process.

4 Introduction

The process to be identified consists of a turnable plate which is affected by the air stream from an electric fan, see Figure 1. The input to the system is the voltage to the fan motor, and the output is a voltage proportional to the angular position of the turnable plate. The input voltage, $u(t)$, is in the range of 0 – 6 V (0 – 10 V if you are using the older system) and the output voltage $y(t)$ is somewhere in the range of -10 – +10 V (the offset is varying between different setups. Among many things, the following can be noted about the process:

- The transfer function from the input (voltage applied to the fan motor) to the angular velocity of the fan propeller can be approximated by a first order model.
- The transport time for the air-stream from the fan to the plate represents a time delay.
- The turnable plate can be viewed as a slightly damped pendulum.
- The turbulence in the air-stream gives the plate a tendency to oscillate even with a constant input voltage to the fan.

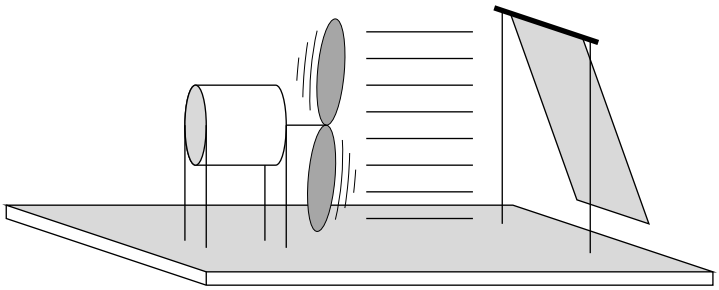

Figure 1: The laboratory process

These facts can be used in order to find an appropriate model structure. To describe a system, we can consider the general linear model

$$A(q^{-1})y(t) = \frac{B(q^{-1})}{F(q^{-1})}u(t) + \frac{C(q^{-1})}{D(q^{-1})}e(t), \quad \mathbf{E}e^2(t) = \lambda^2, \quad \mathbf{E}e(t) = 0. \qquad (1)$$

The model structures ARX and ARMAX can be seen as special cases of the model structure in (1). Choosing $C(q^{-1}) = D(q^{-1}) = F(q^{-1}) = 1$ results in an ARX structure, and choosing $D(q^{-1}) = F(q^{-1}) = 1$ results in an ARMAX structure.

To communicate with the process A/D and D/A converters are used. These can be reached from within MATLAB through the functions:
`proc_ad(channel, variable)`
`proc_da(channel, value)`
and
`stop`
which corresponds to `proc_da(0,0)` and stops the fan.
To collect data, we use the function
`z = coldata(Ts, u, showgraph,y_min,y_max)`
which samples the system every `Ts` seconds with the vector `u` as an input to the system. If the variable showgraph is set to 1, a real time plot appears on the screen. `y_min` and `y_max` are the vertical limits for this plot. Both the output and the input data are included as columns in `z`, *i.e.* `z = [y u]`. The interface is implemented so that the output is measured at A/D channel 0 and the input to the fan motor is taken from D/A channel 0. The input to the motor is also measured (to get better accuracy) at A/D channel 1. A more detailed description of these functions are given in Appendix A.

5 Tasks

1. **Basic experiments**
   First some elementary knowledge of the process is to be found.

   - Determine whether the process is linear or not. If it is not found to be linear, determine in what range it can be viewed as a linear process. One way of doing this is to use a stair-stepped function (a sequence of steps) as input to the process. To find the linear range you may take the mean values of the input and output for the last samples of each step, and then plot the mean values of the output versus the mean values of the input (i.e., the static gain plotted as a function of input level).

     To construct a stair-stepped input signal from 0 to 6 V in 0,5 V steps, each being 200 samples long, the MATLAB command
     `u=kron(0:0.5:6,ones(1,200))`
     can be used. Use `coldata` with e.g. `Ts=0.04` to collect the stair-stepped input-output data (use [-5,5 V] as a starting point for the limits of the vertical axis of the showgraph). To plot the static gain function, use the command `sgain(z,K,N)`, where `z` is the input-output matrix, `K` is the number of steps and `N` is the samples per step (in the example above `K`=12 and `N`=200).

     Discuss your findings below. For what input values can the process be considered linear? What are the corresponding steady state output levels?

   - Use `coldata` to determine a suitable sampling rate and an appropriate time delay from a step response, *i.e.* let `u` be a step in the linear range you have determined. As a rule of thumb, the rise-time of the step response should correspond to about ten samples.

     Use the sampling interval 0.01s as a starting point. Note that the real time plot can not be magnified. To be able to zoom, instead use `plot(z)`.

Which sampling interval did you select? What is the time delay expressed in the the number of sampling instants? What is the accuracy of your visual estimate of the time delay?

---

2. **Non-parametric identification**
   Next, frequency analysis will be used for a non-parametric identification of the process. A graphical representation of the sampled process is obtained by computing the gain $|G(i\omega)|$ and phase shift $\arg(G(i\omega))$ of the system for a set of (angular) frequencies ω. The MATLAB macro:
   `[w, mag, fi]=freqan(Ts)`
   is an iterative function that you can use to run the process for different frequencies. The `freqan(Ts)` function uses the `coldata` function and plots the measured gain and phase shift for each frequency in a Bode plot. Note that the frequencies that the program asks for, `w` are angular frequencies normalized to be in the range $[0, \pi]$, where π is the Nyquist frequency. That is, the given frequency `w` corresponds to a physical frequency $f = \frac{\omega}{2\pi} = \frac{\mathtt{w}}{2\pi \mathtt{Ts}}$ Hz.

   For our purposes it is probably sufficient if the frequency analysis is done for frequencies in the interval $\mathtt{w} \in [0, 1]$. Make sure that any interesting part of the Bode plot is described in detailed, *i.e.* try several frequencies around that area.

   Save the values in the vectors `w, mag,` and `fi` either by saving the variables to disk using the `save` command or just make sure you do not clear or overwrite these variables. These values will be used to compare the parametric models determined in the next task.

   What does the Bode plot say about the system? In particular, does the phase diagram say anything about the system order? Have you chosen a suitable sampling rate?

---

3. **Influence of input spectra in ARX modelling**
   In this task we will estimate a low order ARX model of the system by the least squares method and examine the influence of the input signal on the estimates of the model parameters. Use for instance an ARX(2,2,1) structure in this experiment. As input signals, we consider:

   - A pseudo random binary sequence (PRBS)
   - Filtered white noise

   Remember to use the `help` function in MATLAB.

   Generate the PRBS signal with the function
   `u=teleg(low, high, samples, p, duration)`.
   Here, `low` and `high` are the values the telegraph signal changes between and they should be chosen within the linear interval. `p` is the switch probability and `duration` is the minimum length for each level. Typical values of these parameters may be $p = 0.3$ and duration=2, respectively. A suitable experimental length may be about 30 seconds.

   A low pass filtered white noise input could be obtained by using the filter $(1 - 0.8q^{-1})u(t) = e(t)$, where $e(t)$ is a zero mean white noise with variance 1 (use `randn`). Use the MATLAB function `filter` to generate $u(t)$.

   Note that for a fair comparison of the different input signals the filtered white noise should be normalized such that it has the same average power as the telegraph signal. This can be done by scaling the filtered noise signal so as to match its standard deviation to that of the telegraph signal. Also note that you need to add a suitable offset to the input signal in order to be in the linear range of the process (see the task conducted in the basic experiments).

   Make sure that you use appropriate variable names for easy reference, e.g., add the suffix `_teleg` to variables based on using the telegraph signal input and `_filt` to variables based on using the filtered noise signal input. It is also recommended that you add a suffix to estimates explaining what model order used, e.g., `_221` for an ARX(2,2,1) model and `_3332` for a ARMAX(3,3,3,2) model.

   The output data is again collected using the function `coldata`.

   Remember to remove mean values of the input and output data before the identification. This can be done with the MATLAB command
   `zd = detrend(z, 0)`

   The frequency response of the obtained ARX model can be compared with the frequency analysis in Task 2 with the function
   `bodecomp(w, mag, fi, num, den, Ts)`

`num` and `den` are here the transfer function polynomials, these can be obtained from the theta format by using the function `th2poly`.

To plot the input signal spectrum, use the command
`bodeplot(spa(detrend(u)))`.
Observe that it is possible to find better input signals than those illustrated above. These choices are done just to illustrate that different inputs give different accuracies in the parameter estimates.

For the different input signals, compare the bode plots of the estimated transfer functions and the bode plots derived in the previous task. Summarize your findings below. How is the spectral content of the input signal affecting the accuracy of the model? Try to relate the shape of the input spectrum to the accuracy of the model.

---

4. **Parametric identification**
   For a suitable input signal, determined in the previous task, compare some different models of the system, e.g. a low order ARX, a high order ARX, and ARMAX models. Remember to remove mean values of the input and output data before the identification. Compare the performance of these models using

   - Values of the loss function and AIC.
   - Pole-zero plot.
   - Correlation tests based on the residuals.
   - Cross-validation. Compare the model output from a second data set, using the model obtained from the first data set, to the observed output.
   - Comparison with the result of the frequency analysis in Task 2.

   The above validations can be performed using the following functions from the SI toolbox: `present(th)`, `zpplot(th2zp(th)`, `sd)`,

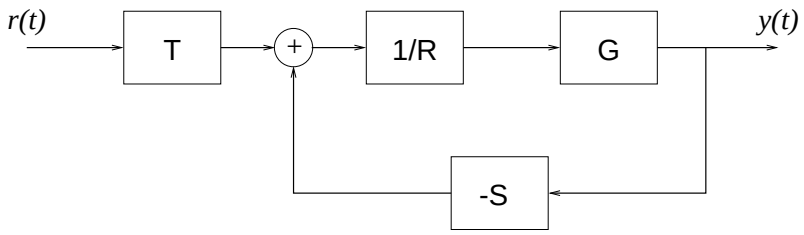

Figure 2: The RST controller structure.

`resid, compare, bodecomp`[1]

Discuss the results from the model validation below. Which models can be rejected? Why should the mean values of the input and output be removed?

5. **Control of the process**
   Finally, some of the models derived in the previous task will be used to design a polynomial pole placement controller, i.e. a RST-controller, having the structure given in Figure 2.

   The MATLAB macro `controller` computes the R, S and T polynomials which will give the desirable closed loop system. It then starts to control the system given a reference signal, see below. The estimated model used in the control is given by the A and B polynomials as well as the time delay. Note that the B polynomial must start with the first

[1] This function is actually not a part of the SI toolbox.

non-zero value, and that the parameters can be given as *variables*, *i.e.* do not write in numerical values, use symbolic names of your variables. The poles will be placed according to the scaling factor c, so that the poles of the closed loop system are a factor c closer to the origin than the poles of the estimated model. A typical value of c could be in the range [0.7, 0.9].

To get a reference signal to the system, connect a voltage generator to AD2. The reference signal can now be changed in the range 0 – 10 V. However, if the output signal in the linear range of the system is negative and a negative reference signal is needed an offset can be added to the reference signal virtually in the computer by giving it as an extra parameter, `refav`, to the `controller` function.

The controller is started by the MATLAB command
`controller(Ts, A, B, nk, c, ymin, ymax, refav)`
where `ymin`, `ymax` are scaling variables for the real time plot that will appear on the screen. To stop the controller press `ctrl+c` followed by the MATLAB command `stop`.

Discuss the following aspects below.

- What can you say about the performance of your controller for different c? How does the system respond to a step in the reference signal?
- In what ranges does the controller work in an acceptable manner?

A MATLAB functions

Useful functions in the System Identification Toolbox

```
ARMAX  Computes the prediction error estimate of an ARMAX model.

       TH = armax(Z,NN)

       TH: returned as  the estimated parameters of the ARMAX model
       A(q) y(t) = B(q) u(t-nk) + C(q) e(t)
       along with estimated covariances and structure information.
       For the exact format of TH, see HELP THETA.

       Z : The output-input data Z=[y u], with y and u being column vectors.
       For a time-series, Z=y only. The routine does not work for multi-
       input systems. Use PEM for that case.

       NN: Initial value and structure information. When no initial parameter
       estimates are available enter NN as
       NN=[na nb nc nk], the orders and delay of the above model,or as
       NN=[na nc] for the time-series case (ARMA-model). With an initial
       estimate available in THI, a theta matrix of standard format, enter
       NN=THI. Then the criterion minimization is initialized at THI.

       Some parameters associated with the algorithm are accessed by
       TH = armax(Z,NN,maxiter,tol,lim,maxsize,T)
       See HELP AUXVAR for an explanation of these and their default values.


ARX    Computes LS-estimates of ARX-models

       TH = arx(Z,NN)

       TH: returned as the estimated parameters of the ARX model
       A(q) y(t) = B(q) u(t-nk) + e(t)
       along with estimated covariances and structure information.
       For the exact structure of TH see HELP THETA.

       Z : the output-input data Z=[y u], with y and u as column vectors.
       For multivariable systems Z=[y1 y2 .. yp u1 u2 .. um]. For time series
       Z=y only.

       NN: NN = [na nb nk], the orders and delays of the above model.
       For multi-output systems, NN has as many rows as there are outputs
       na is then an ny|ny matrix whose i-j entry gives the order of the
       polynomial (in the delay operator) relating the j:th output to the
       i:th output. Similarly nb and nk are ny|nu matrices. (ny:# of outputs,
       nu:# of inputs). For a time series, NN=na only.
       Some parameters associated with the algorithm are accessed by
```

```
        TH = arx(Z,NN,maxsize,T)
        See HELP AUXVAR for an explanation of these and their default values.


 COMPARE    Compares the simulated/predicted output with the measured output.

        YH = COMPARE(Z,TH,M)

        Z : The output - input data for which the comparison is made (the
            validation data set.
        TH: The model in the THETA format (see also THETA).

        M : The prediction horizon. Old outputs up to time t-M are used to
            predict the output at time t. All relevant inputs are used.
            M = inf gives a pure simulation of the system.(Default M=inf).
        YH: The resulting simulated/predicted output.
        COMPARE also plots YH together with the measured output in Z, and
            displays the mean square fit between these two signals.
            (blue(cyan)/solid is YH. black(white)/dashed is measured output)

        [YH,FIT] = COMPARE(Z,TH,M,SAMPNR,LEVELADJUST)
        gives access to some options:
        FIT: The mean square fit.
        SAMPNR: The sample numbers from Z to be plotted and used for the
           computation of FIT. (Default: SAMPNR = all rows of Z)
        LEVELADJUST: 'yes' adjusts the first values of YH and Z(:,1) to
          zero before plot and computation of FIT. 'no' is default.


 DETREND    Remove a linear trend from a vector, usually for FFT processing.

       Y = DETREND(X) removes the best straight-line fit linear trend from
       the data in vector X and returns it in vector Y.  If X is a matrix,
       DETREND removes the trend from each column of the matrix.

       Y = DETREND(X,'constant') removes just the mean value from the vector X,
       or the mean value from each column, if X is a matrix.

       Y = DETREND(X,'linear',BP) removes a continuous, piecewise linear trend.
       Breakpoint indices for the linear trend are contained in the vector BP.
       The default is no breakpoints, such that one single straight line is
       removed from each column of X.


% IDSIM  simulates a given system
%
%        Y = idsim(Z,TH)
%
%        TH: contains the parameters of the model in the format described by
```

```
%        HELP THETA.
%
%        Z: the input-noise data Z=[u e]. For multi-variable systems
%        u=[u1 u2 ... un e1 e2 .. ep], with ui and ei being column vectors.
%        The number of noise sources should equal the number of outputs. If
%        the e-vector(matrix) is omitted, a noise-free simulation is obtained.
%        The noise contribution is scaled by the variance information con-
%        tained in TH.
%
%
 PRESENT    presents a parametric model on the screen.

       PRESENT(TH)

       This function displays the model TH together estimated standard
       deviations, innovations variance, loss function and Akaike's Final
       Prediction Error criterion (FPE).


 RESID  Computes and tests the residuals associated with a model

        E = resid(Z,TH)

        Z : The output-input data Z=[y u], with y and u being column vectors.
        For multi-variable systems Z=[y1 y2 .. yp u1 u2 ... un].
        For time-series Z=y only.
        TH: The model to be evaluated on the given data set. (Format as
        described by HELP THETA)
        E : The residuals associated with TH and Z. [resid(Z,TH); just performs
        and displays the tests, without returning any data.]

        The autocorrelation function of E and the cross correlation between
        E and the input(s) is computed and displayed. 3-standard deviation con-
        fidence limits for these values are also given (based on the hypothesis
        that the residuals are white and independent of the inputs). These
        functions are given up to lag 25, which can be changed to M by
        E = resid(Z,TH,M). The correlation information can be saved and re-
        plotted by
        [E,R] = resid(Z,TH).         resid(R);

        E = resid(Z,TH,M,MAXSIZE) changes the memory variable MAXSIZE from
        its default value. See HELP AUXVAR.


 TH2POLY Computes the polynomials associated with a given model.

        [A,B,C,D,F,LAM,T]=TH2POLY(TH)

        TH is the model with format described by (see also) THETA.
```

```
       A,B,C,D, and F are returned as the corresponding polynomials
       in the general input-output model. A, C and D are then row
       vectors, while B and F have as many rows as there are inputs.
       LAM is the variance of the noise source.
       T is the sampling interval.


TH2ZP  Computes zeros, poles, static gains and their standard deviations

       [ZEPO,K] = th2zp(TH)

       For a model defined by TH (in the format described by HELP THETA)
       ZEPO is returned as the zeros and poles and their standard deviations.
       Its first row contains integers with information about the column in
       question. See the manual.

       The rows of K contain in order: the input/output number, the static
       gain, and its standard deviation.

       Both discrete and continuous time models are handled by TH2ZP

       With  [ZEPO,K] = th2zp(TH,KU,KY) the zeros and poles associated with
       the input and output numbers given by the entries of the row vectors
       KU and KY, respectively, are computed.
       Default values: KU=[1:number of inputs], KY=[1:number of outputs].
       The noise e is then regarded as input # 0.
       The information is best displayed by ZPPLOT. zpplot(th2zp(TH),sd) is a
       possible construction.


ZPPLOT Plots zeros and poles.

       zpplot(ZEPO)   or   zpplot(ZEPO,SD)  or  zpplot(ZEPO,MODE)
                      or   zpplot(ZEPO,SD,MODE)

       The zeros and poles, specified by ZEPO (See ZP, ZPSD or ZPFORM
       for the format) are plotted, with 'o' denoting zeros and 'x' poles.
       Poles and zeros associated with the same input, but different models
       are always plotted in the same diagram, and 'ENTER' advances the plot
       from one model to the next (if any).

       When ZEPO contains information about several different inputs there
       are some options:
       MODE='sub' (The default value) splits the screen into several plots.
       MODE='same' gives all plots in the same diagram. Use 'ENTER' to advance
       MODE='sep' erases the previous plot before the next input is treated.

       If SD>0, confidence regions around the poles and zeros are plotted.
```

The region corresponding to SD standard deviations is marked. (This requires ZEPO to be generated by ZPSD.) SD=0 is default.

Other useful functions (most of the functions have been written locally and exists only in the lab computers))

```
PROC_AD     Read input from AD-channel, saves the measured input in value.

            proc_ad(channel, value)

            Note that the variable value must be defined when you use
            the function.

PROC_DA     Send output u to DA-channel

            da(channel, u)


COLDATA  Program for data acquisition.

       z=coldata(Ts, u, showgraph,y_min,y_max)

       z  : contains the output-input data vectors ( z = [y u] ).
       The output y is measured at AD-channel 0. The input u is measured
       at AD-channel 1.
       Ts : the sampling period.
       u  : data vector to be sent to DA-channel 0 (the input signal)
       showgraph : option for real time plot. If graph=1 a real time plot
       appears.
       ymin and ymax: specify the vertical axes of the showgraph
       (default is 0 to 10).

FREQAN  An interactive function for discrete-time frequency analysis.

       [w, mag, fi] = freqan(Ts, w, mag, fi)

       Ts  : the sampling period.
       w   : a vector of frequencies at which the magnification and phase shift
       of the system has been measured.
       mag : the magnification of the system.
       fi  : the phase shift of the system.

       If w, mag, fi are present as inputs, the new measurements are added to
       the old ones.


TELEG  Produces a telegraph signal which is switching between low
       and high for samples values

       u=teleg(low, high, samples, p, duration)

       low      : low value for telegraph signal
       high     : high value for telegraph signal
```

```
       samples  : number of data values to generate
       p        : switch probability
       duration : minimum length in samples for each level
```

```
BODECOMP Compare the bode plot of a parametric model and a non-parametric
       representation.

       bodecomp(w, mag, fi, num, den, Ts)

       w   : frequency at which the non-parametric model is specified.
       mag : magnification.
       fi  : phase shift.
       num : numerator of the parametric model.
       den : denominator of the parametric model.
       Ts  : sampling period
```

```
CONTROLLER  Function for controlling the fan process with model based
            polynomial pole placement design, i.e. RST-design

            controller(Ts,A,B,nk,c,ymin,ymax,refav)

            Ts is the sampling interval
            A and B are the polynomials of the model.
            B should not include the zeros given by the delay.
            nk is the time delay
            c is the scaling factor for the pole placement
            ymin, ymax are the scalingvariables of the y axis
            refav is the offset of y at u=0
            A real time plot is always shown when you run this function.
```