# Computer exercises in Modeling of dynamical systems SYSTEM IDENTIFICATION

Urban Forssell and Fredrik Tjärnström, LiU
Jonas Sjöberg, Chalmers, Håkan Hjalmarsson, KTH
Christian Larsson, KTH, Othmane Mazhar, KTH

1997–06–12
Revisions: 2019–09–02, 2011–08–11, 2008–03–14

# Computer Exercise 1

The **preparation** part has to be done as a **homework**. You can show your results to the teaching assistant in the beginning of the session and proceed with the rest. These exercises are important since the approach that you will learn here will be useful for the final lab.

## 1.1 Introduction

These exercises treat system identification, i.e. how to compute mathematical models for dynamical systems using measured data. We will consider both non-parametric models, e.g. frequency domain models, as well as parametric models. We will use the System Identification Toolbox in Matlab.

## 1.2 Preparation

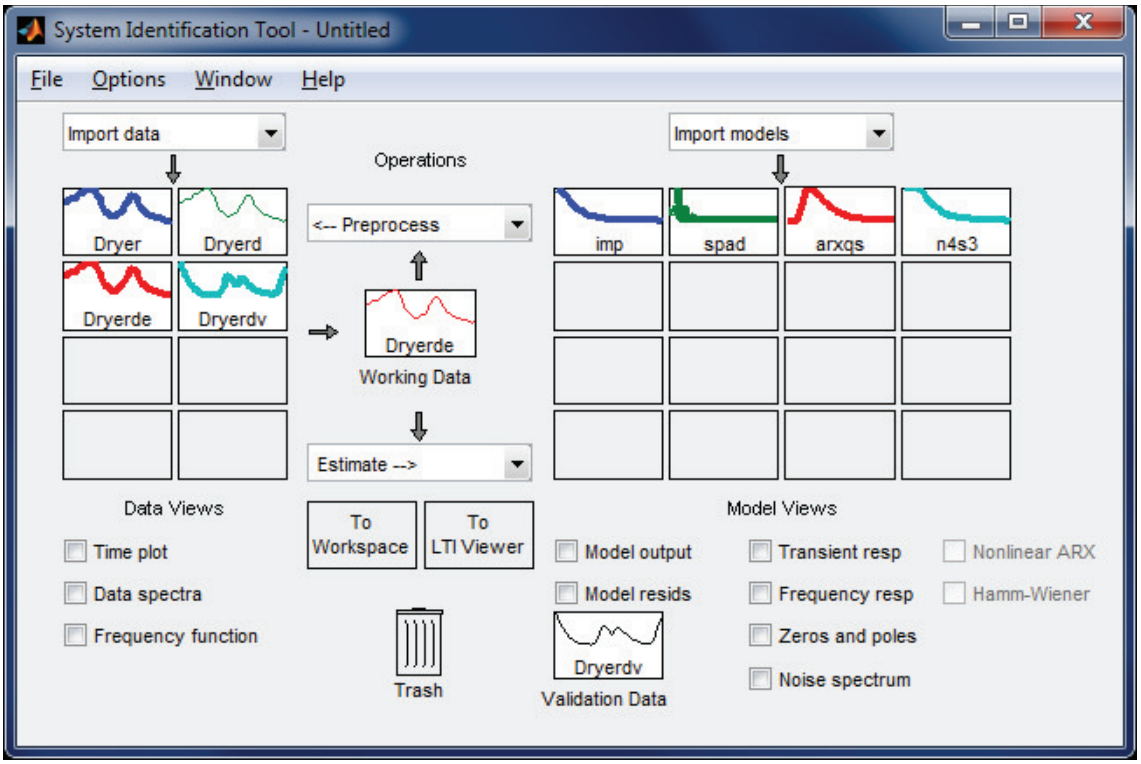
The aim of this part is to get familiar with the System Identification Toolbox in Matlab and to introduce some techniques of non-parametric identification. Make sure you do these preparation questions before attending the computer session.

### 1.2.1 Overview of the Matlab toolbox

Start the graphical user interface by entering the command

```
>> ident
```

in MATLAB's command window. A new window will open (`System Identification Tool - Untitled`) as shown below.




A quick introduction to the SYSTEM IDENTIFICATION TOOLBOX is obtained by using the built-in demo. It is launched by choosing **Interactive demo of the GUI** from the scroll down menu **Help**. A new window (`Interactive Demo of System identification tool`) will pop-up containing instructions. Click the **Continue**-button and simply follow the instructions. More information about the different functions and settings of the SYSTEM IDENTIFICATION TOOLBOX can be found under the **Help**-menu.

### 1.2.2 Pretreatment of data

We will use data from a DC-motor. The motor has been controlled by a computer, which means that the input signal has been kept constant over the sampling intervals and the output has been measured at discrete time

instances. Since the motor, considering the angular position as output, is not input-output stable, it has been stabilized with a feedback controller during the experiment. Our objective is to create a model of the open loop system. The sampling interval has been 0.05 s and 600 samples have been collected. These data have been stored in the file **dcservo.mat**, and can be retrieved by the command

```
>> load dcservo
```

in MATLAB's command window. The measurements are stored in two variables **u** (input signal, voltage) and **y** (output signal, angular position). Import these signals to `ident` by clicking on **Import data** and choosing **Time domain data**. A dialog box will open where input signal, output signal, starting time, sampling interval and name can be entered. Do this and then click the **Import**-button. Imported data are represented as an icon in the `ident`-window. Close the dialog box by clicking the **Close**-button.

The signals can be studied by clicking on **Time plot** under **Data Views**. Any signal can be selected by clicking on its icon and all selected signals are shown in the same plot. To indicate that a signal is selected, the icon is shown with heavy lines.

In order to obtain a good result in the identification one has to ensure that the mean values of the signals are zero. To remove constant levels, click on the **Preprocess**-menu and select **Remove means**. This creates a new data set which will appear as a new icon. The new data set will be plotted automatically but it might lie outside the plotted range. To solve this, click on the **Options**-menu and select **Autorange**.

Later on, when we have computed parametric models, it makes sense to test the model on other data than those used for the parameter estimation. For this purpose, we split the data set in two parts, one part will be used for parameter estimation and the second part for validation. From the **Preprocess**-menu, choose **Select range...**. This opens a new window where we can extract a segment of the data. Choose, e.g., the first 300 samples to be used for the parameter estimation and click on the **Import**-button. Repeat this for the second part of the data which will be used for validation. When you are done, click on the **Close**-button.

To select which data sets will be used for identification and validation, drag

the identification data set to the **Working Data**-block and the validation data to the **Validation Data**-block.

### 1.2.3 Non-parametric identification

One example of a non-parametric identification method is spectral analysis. This method is based on the relationship between the spectra of the input and the output signals. For a system given by

$$y(t) = H(q)u(t) + F(q)v(t), \tag{1.1}$$

it holds (provided $u(t)$ and $v(t)$ are independent) that

$$\Phi_{yu}(\omega) = H(e^{i\omega})\Phi_u(\omega), \tag{1.2}$$

where $\Phi_{yu}(\omega)$ and $\Phi_u(\omega)$ denote the cross-spectrum between the input and output signals and the spectrum of the input signal. By computing estimates for these two spectra, an estimate of the frequency function $H(e^{i\omega})$ can be obtained. It was mentioned above that the signals from the DC-motor have been collected in closed loop. This implies that $u(t)$ and $v(t)$ are dependent so the above result is not applicable. However, despite this it turns out that in this case the method works reasonably well. See Appendix A for further details.

A spectral analysis estimate can be obtained by clicking on the **Estimate**-menu and selecting **Spectral model...**. This shows a dialog with a few options, some of which will be studied further later. For now, leave the options at their default values and click on the **Estimate**-button. This results in a model that is added and shown as a new icon. In order to study the result, click on the box **Frequency Resp** (Frequency Response) below the **Model Views** in the `ident`-window. Make sure that the correct model is selected.

**Exercise 1.1:** Compute and plot an estimate of the frequency response for the DC-motor. Does the plot agree with what we know a priori concerning the motor?

. . . . . .

. . . . . .

Spectral analysis estimation is of limited applicability if the objective is to design feedback controllers for the system. However, it provides qualitative information about the system behavior. In addition it can be used for comparison with parametric estimates.

A "window" of size $\gamma$ is used in spectral analysis to smooth the estimate. Above, we used the default value. By entering a value in **Frequency resolution** in the dialog box `Spectral Model` another value can be chosen.

**Exercise 1.2:** Compute and plot the spectral estimate for a few different values of $\gamma$.
Which window size is suitable? . . . . . .
What happens when $\gamma$ is increased or decreased? . . . . . .
For which value of $\gamma$ does the estimate start to become noisy? . . . . . .

**Note!** You can select the frequency responses to be plotted by clicking on the appropriate icons.

## 1.3 Parametric identification

For parametric identification, we start from the model

$$y(t) = H(q)u(t) + F(q)e(t), \tag{1.3}$$

where $e(t)$ is white noise. First we must determine the structure of the transfer functions $H(q)$ and $F(q)$. The computationally simplest model is given by the recursive equation

$$\begin{aligned} &a_0 y(t) + a_1 y(t-1) + \cdots a_{n_a} y(t-n_a) = \\ &b_{n_k} u(t-n_k) + b_{n_k+1} u(t-n_k-1) + \cdots + b_{n_k+n_b} u(t-n_k-n_b) + e(t) \end{aligned} \tag{1.4}$$

where we will assume, without loss of generality, that

$$a_0 = 1, \tag{1.5}$$

i.e.

$$H(q) = \frac{B(q)}{A(q)}, \qquad F(q) = \frac{1}{A(q)}, \tag{1.6}$$

where

$$\begin{aligned} A(q) &= 1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a}, \\ B(q) &= b_{n_k} q^{-n_k} + \cdots + b_{n_k+n_b} q^{-n_k-n_b}. \end{aligned} \tag{1.7}$$

This model is known as an *ARX*-model (AutoRegression with eXogeneous input). We would now like to find values of the coefficients in (1.5) so that our model describes the true system as well as possible. One way of accomplishing this is to select the model that is able to predict the output signal one step ahead in time as well as possible. The model (1.5) gives that a reasonable prediction of $y$ at time $t$ given data to time $t-1$ is given by

$$\begin{aligned} \hat{y}(t) = -a_1 y(t-1) - \cdots - a_n y(t-n_a) + \\ b_{n_k} u(t-n_k) + \cdots b_{n_k+n_b} u(t-n_k-n_b). \end{aligned} \tag{1.8}$$

Since $e(t)$ is white noise its best prediction is its mean value 0.

A measure of how good a model performs is the mean-squared error

$$V_N = \sum_{t=1}^{N} (y(t) - \hat{y}(t))^2. \tag{1.9}$$

We can find the parameter values that minimize this criterion when the model (1.5) is used. Start by clicking on the **Estimate**-menu and selecting **Linear Parametric Models**. This opens a new dialog box, where you first choose model structure (**ARX** is the default) and then estimate the parameter values by clicking on the **Estimate**-button. A new icon will then be created representing the new model. The parameter **na** determines the number of estimated coefficients in the $A$ polynomial and $a_0 = 1$ holds (see above), which means that **na** is equal to the degree of $A$ in (1.5). **nb** determines the number of coefficients in $B$ while **nk** determines the number of leading zeros in $B$, i.e. the number of samples it takes before a change in $u(t)$ becomes visible in $y(t)$. A sampled data continuous time system normally contains one time delay but if the continuous system in itself has a delay it is necessary to select **nk** $> 1$. The parameters **na**, **nb** and **nk** can be chosen by clicking on the **Order editor...**-button or by entering their values in the **Orders**-field in the order **na**, **nb**, **nk**.

In order to see the values of the estimated parameters, right click on the corresponding icon. A `Data/model Info` window will open. **Present** results in a print-out of the estimated parameters together with some additional information. An example for a model having na=1, nb=1 and nk = 1 is shown below.

```
Discrete-time IDPOLY model: A(q)y(t) = B(q)u(t) + e(t)
A(q) = 1 - 0.9608 (+-0.01017) q^-1

B(q) = 0.04279 (+-0.005696) q^-1

Estimated using ARX from data set eDat
Loss function 0.0352262 and FPE 0.035508
Sampling interval: 0.08
Created:       17-Jan-2011 11:25:27
Last modified: 17-Jan-2011 11:25:31
```

The first row shows the model chosen model structure, ARX in this case. The second and third rows show the used polynomials with estimated parameter values and their standard deviation in parentheses. The fourth row tells which function and which data set was used in the estimation. The fifth row shows the loss function, which is the sum of the prediction errors where smaller numbers indicate better estimates. The FPE is a criterion that is modified to take the number of parameters into account in order to avoid using too many parameters. Finally, the last two rows tell when the model was created and when it was last modified.

Choosing a model structure and an appropriate number of parameters is a trial and error procedure. Later on we will study several methods to evaluate the quality of a model. Some indication is obtained from the standard deviation. If the standard deviation is of the same order of magnitude as the estimated parameter, then this parameter can be set to zero without significant change of the criterion.

**Exercise 1.3:** Which values of $na$, $nb$ and $nk$ are reasonable when it is known that the data originate from a DC-motor?

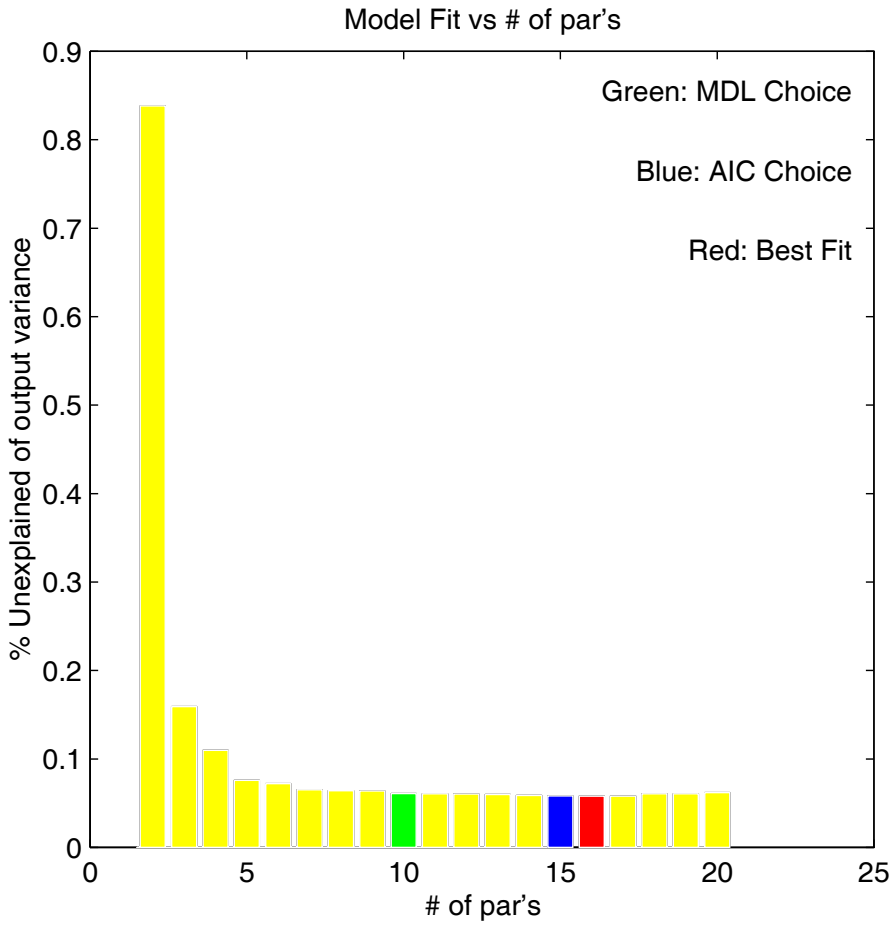
. . . . . .

**Exercise 1.4:** Compute a model for the DC-motor for some combinations of $na$, $nb$ and $nk$. Also write the value of the loss function.

| Model | $na$ | $nb$ | $nk$ | Loss function |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Theoretically, the loss function will be strictly decreasing when the number of parameters grows. For orders beyond the true values, the additional degrees of freedom offered are used to adapt to the specific noise realization. This is called over-fit. In order to avoid this, it is necessary to find a choice of na, nb and nk for which the decrease in the criterion is small when the number of parameters is increased.

Several ARX-models can be computed simultaneously by clicking on the **Order selection**-button and then on the **Estimate**-button in the dialog box `Parametric Models`. The estimation criteria for all ARX-models with one to ten a-parameters, one to ten b-parameters and one to ten time delays are then computed and a new window `ARX Model Structure Selection` is opened. The criterion is plotted as function of the number of estimated parameters, i.e. **na + nb**. The number of parameters and the criterion is shown when you click on one of the bars. Click on the **Insert**-button to choose this number of parameters.




The estimation criterion is plotted as function of the number of estimated parameters, i.e. **na**+**nb**. Each bar thus represents the criterion for the combination of **na**, **nb** and **nk** that results in the smallest criterion for a given number of parameter. We see that the criterion does not decrease significantly above 5 estimated parameters, and therefore we choose 5 parameters. The program gives that the best model for this number of parameters is **na**= 3, **nb**= 2 and **nk**= 1.

**Exercise 1.5:** Which values of **na**, **nb** and **nk** seem to be suitable from Exercises 1.3–1.4 and the result from `ARX Model Structure Selection`? Use this model in the exercises below.

. . . . . .

Besides the parameters, it may be of interest to study the Bode diagram of the model. The frequency response is shown in the window `Frequency Function`, if you have selected **Frequency resp** below **Model Views** in the `ident`-window. We can now compare the result from the parametric identification with the spectral analysis by plotting several frequency responses in the same diagram. Select the models you want to study by clicking on their icons.

**Exercise 1.6:** Plot the frequency responses from the spectral analysis and the parametric identification in the same diagram. How well do they fit in different frequency regions?

. . . . . .

. . . . . .

The estimated standard deviation of the frequency response estimate can be shown by clicking on the **Options**-menu and selecting, e.g. **Show 99% confidence intervals**. The confidence level can be changed via **Options > Set confidence level**.

To see the poles and zeros of the model, click on **Zeros and poles** under **Model views** in the `ident`-window. Also here confidence intervals can be shown from the **Options**-menu.

**Exercise 1.7:** Plot the poles and zeros of the model. Are they reasonable? Motivate!

. . . . . .

. . . . . .

## 1.4 Model validation

There are many ways to perform model validation, i.e. to evaluate the model, and we have already used some of these. By comparing the frequency response from spectral analysis with the one from the parametric model, an indication whether the parametric model is reasonable is obtained. One can also study if the pole and zero locations seem reasonable according to prior information. In addition, a pole and zero close to each other indicate over modeling and that the number of estimated parameters can be reduced. The standard deviations of the estimated parameters also provide information as already mentioned.

An additional method for testing the model quality is to simulate the model and then compare the model output with the output from the real system. This can be performed by clicking **Model output** under **Model Views**. This shows a plot of the output predicted by the estimated model as well as the measured output signal. A **model fit** in percent is also computed. This comparison should be made with a validation data set which has not been used for parameter estimation. By default, a pure simulation is used. This can be changed to prediction by going to the **Options**-menu and choosing **# step ahead predicted output**. The # is the prediction horizon, i.e. the number of samples ahead in time that we want to simulate. The default value is 5. The prediction horizon can be set from the **Options**-menu under **Set prediction horizon**.

A pure simulation sometimes does not work so well for the system that we are studying. The DC-motor contains a pure integrator and therefore the model may have a pole slightly outside the unit circle, i.e. the model is unstable. Simulation then results in that the output grows without limit. Therefore, a shorter prediction horizon should be chosen, e.g. 20 samples.

**Exercise 1.8:** Make a comparison between predicted and measured output signal. Try both pure simulation and shorter prediction horizons.

. . . . . .

. . . . . .

There are also statistical methods for model validation, such as computation of the standard deviations of the estimated parameters.

It is also possible to make a statistical analysis of the prediction errors (the residuals)

$$\varepsilon(t) = y(t) - \hat{y}(t), \tag{1.10}$$

which according to theory should be white noise if the model describes the system correctly. The autocovariance function

$$R_\varepsilon(\tau) = \mathrm{E}\{\varepsilon(t+\tau)\varepsilon(t)\} \tag{1.11}$$

then should be zero except for $\tau = 0$. By clicking **Model resids** (Model residuals) this autocovariance function is estimated and plotted together with a confidence interval which corresponds to white noise. The function also plots an estimate of the cross-correlation between the prediction errors and the input signal

$$R_{\varepsilon\ u}(\tau) = \mathrm{E}\{\varepsilon(t+\tau)u(t)\}. \tag{1.12}$$

If this function is non-zero for negative $\tau$, it means that the system is operating in closed loop. Correlation for positive $\tau$ indicates that the model can be improved. In certain cases cross-correlation for positive $\tau$ may arise due to nonlinearities in the system and these are difficult to model with linear models.

Validation data should be used in the statistical analysis.

**Exercise 1.9:** Compute and plot the autocovariance function for the identified model. Comment on your findings.

. . . . . .

. . . . . .

**Exercise 1.10:** Use two models to compare the different validation methods. Choose one model of suitable structure, i.e. a suitable order with not too many parameters. Then choose the other model of too high order. How is this visible in the different validation methods?

. . . . . .

. . . . . .

So far we have only been looking at ARX-models of different orders, sometimes it might be necessary to try more complex model structures in order to find a good model.

## 1.5 Different model structures

A general linear model according to (1.3) is given by the *Box-Jenkins (BJ)*-model

$$y(t) = \frac{B(q)}{F(q)}u(t - n_k) + \frac{C(q)}{D(q)}e(t), \tag{1.13}$$

where

$$\begin{aligned} B(q) &= b_1 + b_2 q^{-1} + \cdots + b_{nb} q^{-nb+1}, \\ C(q) &= 1 + c_1 q^{-1} + \cdots + c_{nc} q^{-nc}, \\ D(q) &= 1 + d_1 q^{-1} + \cdots + d_{nd} q^{-nd}, \\ F(q) &= 1 + f_1 q^{-1} + \cdots + f_{nf} q^{-nf}, \end{aligned}$$

and $e(t)$ is white noise.

Similar to when we estimated the ARX-model we want to minimize the quadratic criterion (1.9).

This model is also a parametric model. To use this model structure (1.13) choose **Linear parametric models...** from the menu **Estimate** in the `ident`-window. This opens a dialog where you can choose model structure and the number of parameters that is to be estimated in the polynomials. Choose **BJ** (Box-Jenkins) from the menu **Structure**. Then choose the number of parameters in a similar way as in the ARX case by using the button **Order editor...**. Here **nb**, **nc**, **nd** and **nf** define how many parameters should be estimated in each polynomial and **nk** is the number of samples the input is time delayed. Observe that the polynomials $C$, $D$ and $F$ all begin with a 1.

A special version of the BJ-model is generated if

$$F(q) = D(q) = A(q) = 1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a}. \tag{1.14}$$

This model is called *ARMAX* (Auto Regression Moving Average with eXogeneous input signal). This model can be written as

$$A(q)y(t) = B(q)u(t - n_k) + C(q)e(t), \tag{1.15}$$

where

$$\begin{aligned} A(q) &= 1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a}, & (1.16)\\ B(q) &= b_1 + b_2 q^{-1} + \cdots + b_{n_b} q^{-n_b}, & (1.17)\\ C(q) &= 1 + c_1 q^{-1} + \cdots + c_{n_c} q^{-n_c}, \end{aligned}$$

and $e(t)$ is white noise.

An ARMAX-model can be chosen in the same way as the ARX and BJ-models, by selecting **ARMAX** under **Structure**. The parameters **na**, **nb** and **nc** specify how many parameters should be estimated for each polynomial, and **nk** is the number of samples the input is time delayed. An ARX-model is a special version of the ARMAX-model with $C(q) = 1$.

Another special case of the BJ-model is generated if the modeling of the noise is not considered, i.e.

$$\frac{C(q)}{D(q)} = 1.$$

Such a model is called *Output-Error (OE)*-model, since $e(t)$ will be the difference between the true output signal and the model's output signal.

An OE-model can be selected by choosing **OE** under **Structure**. The parameters **nb** and **nf** specify how many parameters should be estimated for each polynomial, and again **nk** is the number of samples the input is time delayed.

How should you determine which model structure to choose? This is discussed in the lectures. Here we only make some short comments. If you do not have any knowledge about the system the principle for choosing model structure is to try simple things first. You start with an ARX-model. If you are not satisfied with the model after validating it or the model contains too many parameters, then try an ARMAX-model, and eventually also a BJ-model. An OE-model is used if you do not care about modeling the noise.

When validating the model the same tools as we used earlier can be considered, i.e.

- compare with spectral analysis
- study the standard deviations for the parameters estimations
- plot poles and zeros
- study the residuals
- simulate the model and compare with the true output signal

Furthermore, you need to determine if the model is reasonable. Normally, you need to do the complete modeling procedure several times until you are satisfied with your model.

We shall now try these different model structures on the data from the DC motor.

**Exercise 1.11:** Test the ARMAX-, BJ- and OE-model structures for different model orders. Use the "best" ARX-model as a starting point for selecting orders. Which model is to be preferred? .....

.....

.....

.....

.....

# Computer Exercise 2

The **preparation** part has to be done as a **homework**. You can show your results to the teaching assistant in the beginning of the session and proceed with the rest. These exercises are important since the approach that you will learn here will be useful for the final lab.

## 2.1 Introduction

In this exercise two different ways of estimating a model, black-box identification and identification of a parameters in a physical continuous time model, are compared. The importance of choosing a suitable sampling interval and input signal is also illustrated. How to validate the identified model is also shown. The exercises will also familiarize you with the command line functions of the System Identification Toolbox.

We will study in detail how to model a first order system, both through a discrete time model and a continuous time model.

The system to be identified is a first order system given in Figure 2.1.

## 2.2 Preparation

Experiment design is a part of the exercise. This means that your choice of sampling interval and input signal are critical for how good your model will become. It is therefore important to study the corresponding parts in the

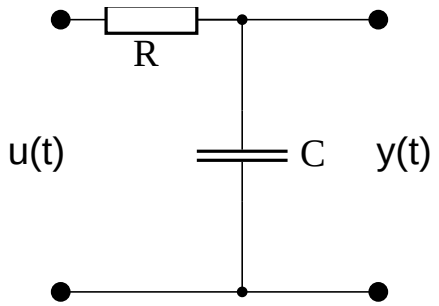



Figure 2.1: RC-circuit, a first order system

lecture material *before* this exercise.

1. Derive the differential equation that describes the relationship between $y(t)$ and $u(t)$ for the system in Figure 2.1. Which type of filter is this (low-pass, high-pass, *etc.*)?

2. Determine the recursive equation that is obtained if the system is sampled with sampling interval $T$ and $u(t)$ is kept constant during each sampling interval. Write the result in the form $y(t+T)+ay(t) = bu(t)$.

3. Read through Appendix B and write an m-file for the physically parameterized, continuous time model of the RC circuit.

## 2.3 Estimation of the parameters of a physical model

The system that will be modeled has been implemented in SIMULINK. By writing

```
>> rcident
```

in the MATLAB-command window, a SIMULINK-window that looks like the one in Figure 2.2 appears.

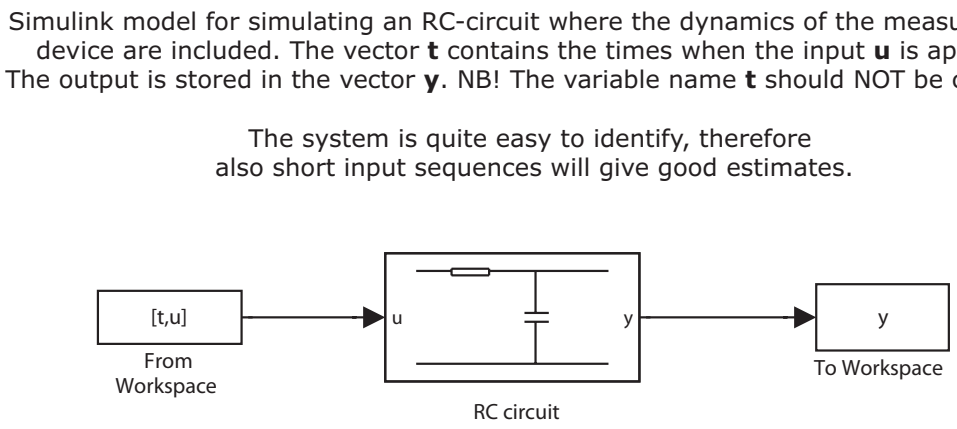



Figure 2.2: Simulink model of an RC-circuit.

## Simulation

In order to be able to simulate the system an input signal vector `u` has to be generated as well as a corresponding time vector `t`. The vector `t` should contain the sampling times for when the samples in the input vector `u` should be applied to the system. (Notice that it is important that you do not change the name of `t` since it is used in other places in the program.) When this is done, choose SIMULATION and then START from the menu. The system will now be simulated and an output signal vector `y` is created in the MATLAB-workspace. The vectors `u` and `y` can now be used for estimation. Notice that in many cases it is important to carefully choose the input and the sampling interval to ensure an accurate model of the system.

## Estimation

- Choose $R = 10\text{k}\Omega$ and $C = 10\mu\text{F}$ in the SIMULINK-model. Design the experiment, collect the data and estimate the parameters in the discrete time model that you computed in exercise 2 in the Preparations, using the command `arx`. What are the values of the estimated parameters?
- Estimate the parameter in a physically parameterized, continuous time model (c.f. exercise 3 in the Preparations and Appendix B). Try several different initial values in order to get as good result as possible. What is the resulting model?

- Re-do the above two steps one more time, but let the system parameters be $R = 1\text{M}\Omega$ and $C = 0.1\mu\text{F}$. What do you expect from the estimation this time? Try to explain the difference in the results.

  (Hint: In the SIMULINK-model, the dynamics of the measurement instrument has been included. This instrument has an impedance at the input of approximately $5\text{M}\Omega$.)

### Validation

- Compare the estimated parameter values for both the discrete time and the continuous time models with the theoretical values. Give the absolute as well as the relative errors.
- Plot Bode diagrams for the models in the same figure using the command `bodeplot`. Why do the curves not match?
- Can you see any advantages in using a continuous time model?

## 2.4 Nonlinear black-box identification

Here we will identify and compare the performance of linear and nonlinear black-box models. We are going to model the crane arm of a hydraulic log loader. The position of the crane arm is controlled by pumping oil into a cylinder which is coupled to the crane arm. The pressure in the cylinder is controlled by a valve. We know that the crane has a resonance somewhere between 8 and 20 rad/s.

We collect 20 s of experimental data at a sampling frequency of 50 Hz. The input signal $u$ is the valve opening and the output signal $p$ is the resulting cylinder pressure. The data is stored in the file **kran.mat**.

Start by loading the data into the Matlab workspace and importing it into the SYSTEM IDENTIFICATION TOOLBOX (`ident`). Detrend and divide the data set into estimation and validation data.

**Exercise 2.3:** Try to identify a linear ARX model from the data. Evaluate the performance of the model both in simulation and prediction. What are

suitable model orders?

. . . . . .

. . . . . .

Identification of nonlinear models is performed in much the same ways as for linear models in the System Identification Toolbox. Click on the **Estimate**-menu and select **Nonlinear models...**. On the **Configure**-panel you can select the model type you want to estimate; here we focus on **Nonlinear ARX**.

For nonlinear ARX models you need to select the orders of the regressors and the type of nonlinearities that you want to use. Start by selecting the regressors in the **Regressors**-tab. Use the insights from identifying the linear ARX model in the selection. The nonlinearities are configured in the **Model Properties**-tab. You can choose which type of nonlinearity and how many units you want to use. When you are satisfied with the model configuration, click on the **Estimate**-button. When the model has been estimated it can be found among the other models in the `ident-` window. More details on nonlinear identification in the System Identification Toolbox can be found by clicking on the **Help**-button.

**Exercise 2.4:** Identify some nonlinear ARX models. Try using different regressors, nonlinearities and number of units in the nonlinear block. Which model structures worked well?

. . . . . .

. . . . . .

# Appendix A

# Estimation of the Power Spectrum

Let $u$ be a weakly stationary stochastic process. The spectral content of $u$ is characterized by its *power spectrum*

$$\Phi_u^h(\omega) := \lim_{N\to\infty} \mathrm{E}\{\hat{\hat{\Phi}}_u^N(\omega)\}, \qquad -\frac{\pi}{h} < \omega \leq \frac{\pi}{h},$$

where

$$\hat{\hat{\Phi}}_u^N(\omega) := \frac{1}{Nh}|U_N(e^{i\omega h})|^2$$

is called the *periodogram* of $u$. Thus, the periodogram is in a sense the "natural" estimator of the power spectrum.

The function $U_N(e^{i\omega h})$ is the truncated discrete-time Fourier transform of $u$, *i.e.*,

$$U_N(e^{i\omega h}) := h\sum_{k=1}^{N} u[k]e^{-i\omega kh}.$$

The spectrum of a sampled signal $u$ can be computed from the expression for $h = 1$ ($\Phi_u(\omega)$) by evaluating $\Phi_u^h(\omega) = h\Phi_u(\omega h)$, thus we will assume without loss of generality that $h = 1$.

## A.1 Problems with the periodogram

Figure A.1 shows an example of the power spectrum of some signals, and their periodogram estimates. Notice that the periodogram exhibits wild fluctuations, but it appears to capture the main features of the theoretical spectra.

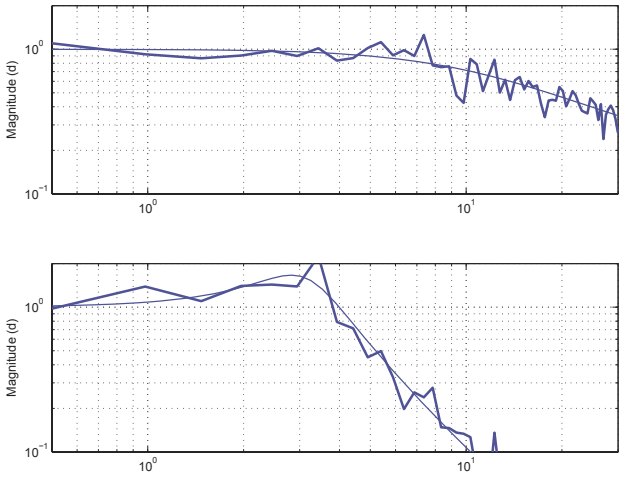



Figure A.1: Periodogram of two signals (thick, blue line), and their theoretical power spectra (thin, black line).

The periodogram is asymptotically unbiased, so it has good *frequency resolution*, *i.e.*, it can distinguish features in the spectrum that are separated by at least $2\pi/N$ rad. However, its variance does not go to zero with the sample size $N$, so it suffers from high variability.

An informal explanation for this problem is the lack of "data compression" in the periodogram: This estimator takes $N$ data samples, and transforms them into $N/2$ complex values (since $\hat{\hat{\Phi}}_u^N(-\omega) = \hat{\hat{\Phi}}_u^N(\omega)$, evaluated at multiples of $2\pi/N$), containing real and imaginary parts, that is, $N$ different numbers, which are asymptotically mutually uncorrelated. Therefore, each of these estimates does not condense more and more information as $N \to \infty$. Compare this with, for example, the average $\bar{x}$ of $N$ i.i.d. numbers as an estimator of their common mean: by condensing $N$ samples into 1, the average reduces the effect of their variability, and provides a estimate that becomes better and better as $N \to \infty$.

## A.2 Blackman-Tukey method

One way to reduce the high variance of the periodogram is to smooth it by averaging the spectral estimates at neighboring frequencies. This is called the *Blackman-Tukey method*.

The averaging can be performed using a *window function* $W_\gamma$:

$$\hat{\Phi}_u^N(\omega) := \frac{1}{2\pi} \int_{-\pi}^{\pi} W_\gamma(\omega - \zeta) \hat{\hat{\Phi}}_u^N(\zeta) d\zeta.$$

The subscript $\gamma$ will explained shortly later.

From a computational point of view, the averaging can be more conveniently implemented in the time domain, as

$$\hat{\Phi}_u^N(\omega) = h \sum_{k=-\gamma}^{\gamma} \{w_\gamma(k) \hat{R}_u^N(k)\} e^{-i\omega k},$$

where $w_\gamma$ is the inverse Fourier transform of $W_\gamma$, and

$$\hat{R}_u^N(k) := \frac{1}{N} \sum_{n=1}^{N} u[n+k]u[n]$$

is the autocovariance estimator of $u$ (where $u[k] = 0$ for $k \leq 0$ and $k > N$).

### A.2.1 Window functions

There are many different types of windows. One of the most common window types is the *Hamming window*, given by

$$w_\gamma(k) = \begin{cases} 0.54 + 0.46 \cos(\pi k/\gamma), & \text{if } |k| \leq \gamma \\ 0, & \text{otherwise.} \end{cases}$$

The subscript $\gamma$ in the definition of a window is called its *window size* (or *width*), and it is defined as the largest $k \in \mathbb{N}$ such that $w_\gamma(k) \neq 0$. The widths of the window in the time and frequency domains are linked via an "uncertainty principle": the wider $w_\gamma$ is in the time domain, the narrower $W_\gamma$ is in the frequency domain, and vice versa. See Figure A.2.

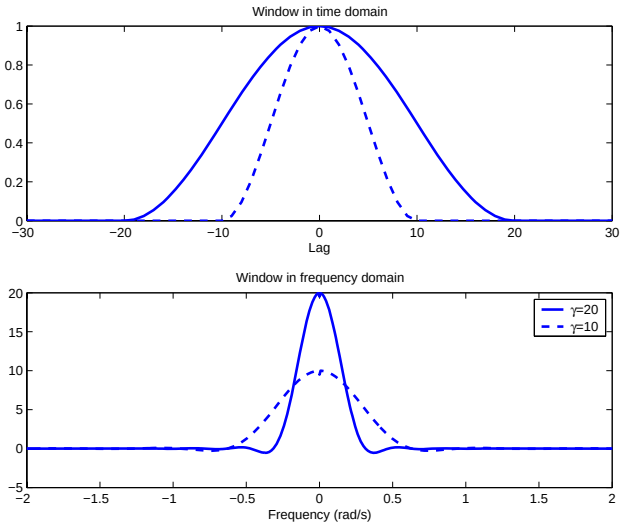



Figure A.2: Hamming window in the time- and frequency-domains.

Once the window type has been selected, one has to choose the window size. This choice is subject to the following trade-offs:

- The width of window in the frequency domain determines the frequency resolution: a wider $W_\gamma$ (*i.e.*, smaller $\gamma$) yields a lower resolution.
- The wider $W_\gamma$ is, the less variable the estimate becomes.

As a rule of thumb, start by choosing $\gamma \approx \min\{20 \sim 30, N/10\}$, and increase it to obtain a good balance between frequency resolution and variance. See the example in Figure A.3.

## A.3 Frequency response estimation

The power spectrum estimates can be used to estimate the frequency response of a linear system. To see this, recall the following relations between the input $u$ and output $y$ of a system, subject to an output disturbance $v$:

$$\Phi_{yu}(\omega) = G_d(e^{i\omega})\Phi_u(\omega)$$
$$\Phi_y(\omega) = |G_d(e^{i\omega})|^2\Phi_u(\omega) + \Phi_v(\omega).$$

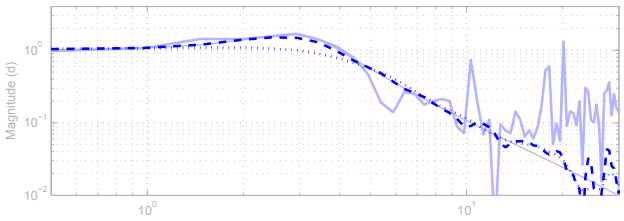



Figure A.3: True spectrum (thin, solid), periodogram (thick, solid), and Blackman-Tukey estimates for $\gamma = 40$ (dotted) and 100 (dashed).

By estimating $\Phi_y$, $\Phi_u$ and $\Phi_{yu}$ using the periodogram or Blackman-Tukey method, the frequency response $G_d(e^{i\omega})$ can be estimated as

$$\hat{G}_d(e^{i\omega}) := \frac{\hat{\Phi}_{yu}^N(\omega)}{\hat{\Phi}_u^N(\omega)}.$$

Here $\hat{\Phi}_{yu}^N$ is based on the cross-covariance of $y$ and $u$, instead of an autocovariance. See Figure A.4.

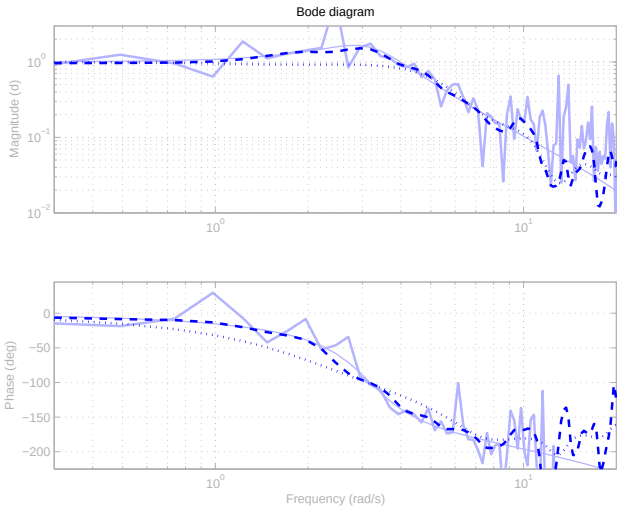



Figure A.4: Frequency response (thin, solid) of a system $G(s) = 10/(s^2+2s+10)$ sampled with $h = 0.1$, subject to output white noise of variance $0.1^2$, and their estimates based on the periodogram (thick, solid) and Blackman-Tukey method with $\gamma = 20$ (dotted) and 60 (dashed).

# Appendix B

# Identifying physically parameterized models

There are several ways of estimating physically parameterized models in the System Identification Toolbox. Here we present one option for identifying a state-space model where some parameters are known.

Consider the following parametric model

$$\ddot{y}(t) + \theta_1 \dot{y}(t) = \theta_2 u(t),\ y(0) = \theta_3,\ \dot{y}(0) = 0,$$

where also the initial state is taken into account. In state space form we get

$$\begin{aligned}
\dot{x}(t) &= \begin{bmatrix} 0 & 1 \\ 0 & \theta_1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ \theta_2 \end{bmatrix} u(t) \\
y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) + e(t) \\
x(0) &= \begin{bmatrix} \theta_3 \\ 0 \end{bmatrix}.
\end{aligned}$$

The state space model is of the form

$$\begin{aligned}
\dot{x}(t) &= A(\theta)x(t) + B(\theta)u(t) + K(\theta)e(t) \\
y(t) &= C(\theta)x(t) + D(\theta)u(t) + e(t) \\
x(0) &= x_0(\theta),
\end{aligned}$$

where

$$A(\theta) = \begin{bmatrix} 0 & 1 \\ 0 & \theta_1 \end{bmatrix}, \qquad B(\theta) = \begin{bmatrix} 0 \\ \theta_2 \end{bmatrix}, \qquad K(\theta) = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$
$$C(\theta) = \begin{bmatrix} 1 & 0 \end{bmatrix}, \qquad D(\theta) = 0, \qquad x_0(\theta) = \begin{bmatrix} \theta_3 \\ 0 \end{bmatrix}.$$

Suppose now that we have measured data and that we would like to estimate $\theta_1$, $\theta_2$ and $\theta_3$ in the above model.

First, we need to define the model structure. This is done using an m-file with the structure

```
[A,B,C,D,K,X0] = mFileName(theta,Ts,Arguments)
```

where `mFileName` is the name of the m-file where you specify the model, `theta` are the parameters of the model, `Ts` is the sampling time, and `Arguments` can be used to pass extra arguments to the m-file. The m-file should return the $A, B, C, D$ and $K$ matrices and the initial state $x_0$. For the model above we could use

```
function [A,B,C,D,K,X0] = myModel(theta,Ts,Arguments)
A = [0 1; 0 theta(1)];
B = [0; theta(2)];
C = [1, 0];
D = 0;
K = [0; 0];
X0 = [theta(3); 0];
```

Notice that `Ts` is not used since we have a continuous time model. We do not need to pass any extra arguments so `Arguments` is not used either.

Second, we need to create a grey-box model structure that the System Identification Toolbox can use. This is done using the command

```
>> M = idgrey(mFileName,initialParameters,CDmFile,Arguments)
```

Here `mFileName` is a string with the name of the m-file that defines the model, `initialParameters` is a vector with initial parameter guesses, `CDmFile` is either `'c'` for continuous time models, `'d'` for discrete time models or `'cd'` for models that can handle both discrete and continuous time. To create an `idgrey`-object from the m-file above, we use the command

```
>> M = idgrey('myModel',[-9 11 0],'c');
```

Here we guess that $\begin{bmatrix}\theta_1 & \theta_2 & \theta_3\end{bmatrix} = \begin{bmatrix}-9 & 11 & 0\end{bmatrix}$ and specify `'c'` for a continuous time model.

The last step before we can identify the model is to create an **identification data**-object. This is done with the command

```
>> data = iddata(y,u,Ts);
```

where `y` is the measured output, `u` the input and `Ts` the sampling time. You might want to remove trends from the data, which can be done with the `detrend` command.

Now that we have data and model objects, we can identify the unknown parameters with the command

```
>> M_est = pem(data,M);
```