Chapter 1

Computer Laboratories

1.1 Computer lab 1: Least Squares

1.2 Computer lab 2: Time Series

1.3 Computer lab 3: System Identification Toolbox

This computer lab demonstrates the use of the MATLAB System Identification toolbox. We will omit state-space models from our experiments as yet.

1.3.1 Setting up the Experiment

In order to illustrate the use we will use the data available in `inout.dat`. The data is loaded in the MATLAB workspace as an `iddata`,

```
 >> load mydata.dat;  data=iddata(mydata(:,2),mydata(:,1),1)
```

The properties of the *object* `iddata` are displayed using the command

```
 >> get(data)
```

The first step, however trivial, is to visualize the data and to look for some regularities here.

```
 >> plot(data)
```

At this early point in the analysis, it is good practice to set aside a portion of the data for validation lateron. As such we make sure this validation data does not influence parameter estimation.

```
>> data1 = data(1:400);
>> data2 = data(401:end);
```

The next step concerns typically preprocessing. Here we want to subtract the mean trend of the signals.

```
 >> data1 = detrend(data1);
 >> data2 = detrend(data2);
```

Again, look at the data. Does the signals look 'normal', or are there outlying measurements in the data which may easily disrupt the coming results? If so, set such value to a reasonable value. In the present data there is no such effects.

1.3.2 Nonparametric Estimation

When the data is loaded in the toolbox, the first step is to perform an nonparametric analysis. Let us review a number of those. The first one is to compute the impulse response using a rough approximation (a large order FIR model). This is performed using

```
>> impulse(data)
```

The dashed dotted lines mark a 99% confidence interval. Meaning that if the experiment were repeated m times, only 1/100 of those cases would not fit the confidence band on the average. There is a time delay (dead-time) of 1 sample before the output responds to the input (significant output outside the confidence interval). Other time-constants can as half-life time and damping are suggested in this plot. A similar analysis is performed using a correlation analysis.

```
>> cra(data,100)
```

The result of an elementary spectral analysis is displayed using the command.

```
>> plot(spa(data))
```

Those tools give you a basic insight of the concerned signals.

1.3.3 Parametric Estimation

Now that we have gained some insight into the experiment, we are well prepared to fit a parametric model on the data. Let us start with a simple example, fitting a FIR(10) model to the data. This can be done using the following command

```
>> m1 = arx(data,[0 2 1])
```

This command not only fits the parameters of such a FIR model, but computes as well the variance of the estimated parameters by taking into account the Fisher information matrix. All information is presented in the command line as

```
>> present(m1)
```

A visual representation of the model is given as

```
>> plot(m1)
```

Look at the pole-zero diagram of this model in the GUI coming up. Is the model estimate MIMO stable? Is it minimal phase? Let us now consider another model

```
>> m2 = arx(data,[10 10 1])
```

Thirdly, consider a ARMAX model os small orders

```
>> m3 = armax(data,[2,2,2,1])
```

Fourthly, consider the estimate using a different model structure.

```
>> m4 = oe(data,[1 2 1])
```

Remember that for OE models a PEM approach does not coincide with a least squares approach, and the parameter estimation procedure internally implemented may be stuck in a local suboptimal estimate.

The next step is to look at the characteristics of the fitted model. For example, one can plot the impulse response corresponding to the fitted model.

```
>> impulse(m1)
```

It is instructive to plot the pole-zero diagram of the estimated model. We do this for models `m1` and `m2` as follows

```
>> pzmap(m1,'b', m2,'r','sd',3)
```

Here the SI toolbox also displays on the plot the variance of the estimated poles and zeros as the small circles. This represents the uncertainty corresponding to the fact that n is only finite, and the estimate might slightly differ if looking at a different realization of the data. The incorporation of this information is a key feature of the SI toolbox, and is quite useful in practice. The bode diagram of the various models are given as

```
>> bode(m1,m2,m3,m4)
```

Other plots of model properties can be brought up using `plot(m1,m2,m3)`.

1.3.4 Model Validation

Now that we have the estimated model and the nonparametric estimates, we concern the question wether this one is actually capturing the real dynamics in the data. Note that in the SI toolbox (help/labels) one often uses the term 'fit' to denote how well the estimated model actually correspond on the observed signals used for estimating. It is paramount to make the distinction between *'fit'*, also called *'training error'*, and the performance on the validation data kept aside. Saying that your model *'fits' the data quite well* merely implies that your estimation procedures work properly, it does **not** say that the estimated model will perform well on new cases, or that you actually captured the dynamics underlying the data. Lets consider the following example

```
>> m5 = arx(data, [20,20,1]);
>> pzmap(m5,'sd',3);
>> compare(data1, m5)
>> compare(data2, m5)
```

This illustrates that things may easily go wrong if selecting too high a model orders. Note that the fitted model suffers from many almost canceling pole-zeros, often an indication of a bad choice of model. Moreover, the estimated model has a training performance which differs quite a lot from the performance on the validation set, suggesting as well that the estimate does not capture the dynamics underlying the data.

If you have to make decide between different model structures/model orders which may be used for parameter estimation, a way is to use an appropriate information criterion as Akaike's:

```
>> aic(m1,m2,m3,m4)
```

or the Finite Prediction Error (FPE) criterion as

```
>> fpe(m1,m2,m3,m4)
```

Yet another test to help you decide wether the estimate is sufficient for your needs is based on the autocorrelations of the residuals. For example, to visualize the autocorrelation of the residuals and the cross-correlations of the given data using the estimated model, look at

```
>> resid(data,m2)
>> resid(data,m4)
```

Based on those plots a clear preference of model `m2` over model `m4` is suggested. The command `advice` suggests useful actions of you for analysis and refining the model. The procedure of selecting the best orders of an ARX model is automized further using the following commands

```
>> V = arxstruc(data1,data2, struc(1:10,1:10,1:10))
>> nn = selstruc(V,'mdl');
>> mopt = arx(data,nn)
```

This gives you some elementary tools to perform a system identification experiment.

1.3.5 The GUI interface

Now the previous steps can be performed using a Graphical User Interface (GUI). Therefor, type

```
>> ident
```

Try the various options in the GUI to redo the previous steps in the analysis. This concludes our tour of the SI toolbox. As a conclusion, which model structure works well for this data? Now you are all ready to construct a good model.

1.3.6 The Question

The specific that illustrates that you have understood how this works goes as follows. Load the data `inout.dat` in the MATLAB workspace. Then decide (i) which model structure to choose, (ii) which model orders are appropriate, (iii) argue for your favorite model using the plots. Which model would you propose to use if we intend to use the data for control purposes? Happy clicking!