Time Series Analysis 2022
Lund University

Computer Exercise 1
Estimation and Model Validation

This computer exercise treats identification, estimation, and model validation in ARMA- and SARIMA-processes. Your goal is to be proficient in simulating different kinds of stationary and non-stationary time series, and to master the 'craft' of identification and modeling of such. You will use the autocorrelation function (ACF), the partial autocorrelation function (PACF), probability plots, and different parameter estimators as your basic toolkit. In this exercise, you will use these on both simulated and measured time series data.

1 Preparations before the exercise

Read Chapter 3 and 4 in the course textbook as well as this guide to the computer exercise. Solve the problem exercises 3.6, 3.10, 4.3, 4.4, and 5.2 in the book. Answers to some of the computer exercise will be graded using the course's *Mozquizto* page. Ensure that you can access the system before the exercise and answer the preparatory questions as well as (at least) three of numbered exercise questions below *before the exercise*.

You can find the *Mozquizto* system at

`https://quizms.maths.lth.se`

It should be stressed that a thorough understanding of the material in this exercise is important to be able to complete the course project, and we encourage you to discuss any questions you might have on the exercises with the teaching staff. This will save you a lot of time when you start working with the project!

You are allowed to solve the exercise in groups of two, but not more. Please respect this.

2 Tasks

The computer program Matlab and the functions that belong to its System Identification Toolbox (SIT) will be used. In addition, some extra functions will also be used in this exercise. Make sure to download these functions and the required data files from the course homepage. You are free to use other programs, such as R or Python, but then need to find the appropriate functions to use on your own.

Important: In the following, we will make use of various Matlab functions. Use the command `help` to learn more about these functions and their expected inputs and outputs. For example, type `help pzmap` in the command window to learn about the function `pzmap`.

Hint: Use a new empty script in matlab for inputing all your code. Save this in the same folder as the toolkit or use the `addpath()` command. You can easily divide your script into cells using *cell mode* in Matlab and then run these separately. Simply switch *cell mode* on and use `%%` on an empty line to divide your code is separate parts. To run a cell, press *Run Section* in the toolbar. Create separate functions for different subtasks, you will likely need to use the same code several times, and this both saves time and helps you to debug your code efficiently.

2.1 Working with time series in Matlab

In this first section, we will work with two sets of A- and C-polynomials, the first set being:

```
A1 = [ 1 -1.79 0.84 ];
C1 = [ 1 -0.18 -0.11 ];
```

and the second set:

```
A2 = [ 1 -1.79 ];
C2 = [ 1 -0.18 -0.11 ];
```

These polynomials builds an ARMA-process of the form

$$A(z)y_t = C(z)e_t.$$

Create the model structure objects for the two sets of polynomials using the function `idpoly` with the command

```
ARMA_poly = idpoly( A, [], C );
```

The resulting object `ARMA_poly` will contain the specified polynomials, making it easier to pass all the information about the process to the different relevant Matlab functions (remember to use the `help` command to learn more about the function `idpoly`). Of course, you may name your objects as you see fit, for example `arma1` and `arma2`.

You may directly access the contained polynomials by access the member variables of the object. For example, to access the $A(z)$-polynomial, type

```
ARMA_poly.a
```

If you do not know the member variables of an object, you may view these by typing

```
ARMA_poly.
```

and then press tab. To view the poles and zeros of the ARMA-process, you can use the function `pzmap` by writing

```
pzmap( ARMA_poly )
```

An easy way to simulate an ARMA-process

$$y_t = \frac{C(z)}{A(z)} e_t$$

in Matlab, is to use the function `filter`. The function needs the $A(z)$ and $C(z)$ polynomials as well as the driving noise process to generate a simulation. To allow for easy comparison with the values used in the *Mozquizto* system, set the random seed using

```
rng(0)
```

Then, generate some normal distributed noise with the command

```
e = sqrt(sigma2) * randn( N, 1 );
```

where `sigma2` is the variance of the noise process `e`, and `N` is the length of the resulting vector. After creating the noise vector, you simulate the ARMA process with the command

```
y = filter( ARMA_poly.c, ARMA_poly.a, e );
```

Important: When *simulating* a process containing an AR part, i.e., an AR or ARMA process, the initial values in the simulated signals will behave differently to the latter values as these are not initiated by earlier values being a part of the process, but with zeros (as, by default in Matlab, $y_0 = y_{-1} = y_{-2} = \ldots = 0$). These will in turn affect the following samples, and so on, with diminishing influence the further you go. As a result, the initial simulated samples will behave somewhat differently from what one expects from the simulated process.

A simple way to handle this is to just simulate a longer process than needed, and then omitting the initial samples. Often, one use a significant "buffer" in this way to avoid any unwanted initial effects to affect the following samples, preferring to exaggerate the number of omitted samples (obviously, this is only

done when simulating signals). For almost all simulated processes, one may safely assume that any initial effects will be negligible after, say, 100 samples (and, typically, well before this).

Hint: It is always a very good idea to test your code using simulated data before actually testing it on real data. In this way, you know the answer and can easily test if your code is working as it should. It is recommended that you create your own function for simulating data, naming it, `simulateMyARMA()`, that takes the specified parameters as input, as well as the desired data length, and then returns the simulated data. Internally, the function then simulates a longer signal, say 100 samples longer, and then just returns the signal from the end of the simulated signal. By doing this, you will ensure that it is easy for you to simulate new data, and that you do not forget to handle the initial data effects when doing do (this is very easy to do!).

Now simulate `300` samples of the ARMA processes and omit the first 100 samples of each; you can do this easily using the command

```
y = y(101:end);
```

Proceed to simulate the `y1` and `y2` processes, using `sigma2=1.5`. Plot the two process using the subplot command.

```
subplot(211)
plot(y1)
subplot(212)
plot(y2)
```

As you can see, one of the processes diverges. Why is that? Plot the poles and zeros of the different polynomials to see if this provides a clue.

QUESTION 1 *In Mozquizto, answer question 1.*

The theoretical covariance function $r_y(k)$ for an ARMA-process can be computed using the Yule-Walker equations using the provided function `kovarians`. The function assumes that the driving noise process has unit variance, i.e., $V(e(t)) = \sigma^2 = 1$. To instead compute the estimated covariance from a given data set, use the function `r_est = covf(y,m)`. Plot to compare the theoretical versus the estimated covariance for `y1` by (set `m=20`) by inputing

```
r_theo = kovarians( ARMA_poly.c, ARMA_poly.a, m );
stem( 0:m, r_theo*sigma2 )
hold on
rest = covf( y1, m+1 )
stem( 0:m, r_est ,'r')
```

Why are the estimated and theoretical covariance functions not identical?

QUESTION 2 *In Mozquizto, answer question 2.*

Your task is now to re-estimate your model using the data vector `y1`. Do some basic analysis by plotting the `acf`, `pacf`, and `normplot` of your data. This is something you will need to do many times during the course, so it might be worth the time to make a short script that, for instance, plots these three figures as three subplots.

For estimation, make your data an `iddata` object. This is a structure that Matlab uses to efficiently handle data. You create it by typing (recall that you can read more about this function by using `help`)

```
data = iddata(y1);
```

Estimate the model you wish to validate using the LS-based method `arx` for an AR(n_a) by typing

```
ar_model = arx( y1, [na]);
```

where `na` is the model order, or using the ML-based method `armax` for an ARMA(n_a, n_c) by typing

```
arma_model = armax( y1, [na nc]);
```

To get an overview of the estimated parameters, their standard deviation, and the model FPE, use the command `present`. To calculate the error residual of your estimated model, you can do this with `filter` by running your data through the inverse filter of your model. Do so by typing

```
e_hat = filter( ARMA_poly.a, ARMA_poly.c, y );
```

Important: An important thing to note when computing the residual using `filter` is that the initial values of the output will be corrupted, as the AR part will need non-existing samples to initialise the first states, therefore setting these to zeros. This means that the first n_a samples will be corrupted and should be removed before *any* further processing is done.

Try changing the order of the $A(z)$ polynomial and plot the first 20 samples of `e_hat`. Can you see that the initial n_a samples are corrupted? To avoid this from causing problems for you, you should *always* remove the first n_a samples when filtering any process that you are modelling. This is done by adding

```
e_hat = e_hat( length(ARMA_poly.a):end);
```

Hint: As you will do the filtering operation many times, and it is very easy to forget omitting these n_a initial samples, it is highly recommended that you create your own function `myFilter` that does the filtering and omits the samples; that way you will not to do forget this. It is only a couple of lines of code, but it will likely save you a lot of pain... However, also note that there are cases when you do not want to do this automatically, especially in the case of prediction that we will cover later - in that case, you will filter two different

sequences with filter with different lengths - and then want omit the same number of samples as the longer filter for both sequences. In this case, you do *not* want to use `myFilter`.

Proceed to find an appropriate model model for the process by adding one extra parameter at a time, and then analyze the residual using `acf`, `pacf`, and `normplot`. Make a note of the FPE for each of the models you examine.

QUESTION 3 *In Mozquizto, answer question 3.*

It is worth noting that, for shorter sequences, the FPE values are often unreliable, as are, e.g., the AIC and BIC values. For such sequences, there is simply not enough data to reliably estimating the model order using an information criteria. However, when there is enough data, these models are often reasonably close to the correct order, and can in this sense be used to determine a rough order estimate that one can use as a starting point.

Hint: For the data considered in this course, you should be very wary to trust the results you get from FPE, AIC, or BIC, as they will likely not be reliable!

2.2 Model order estimation of an ARMA-process

In the file `data.dat` you will find 200 observations of the ARMA(1,1)-process

$$y_t - 0.6y_{t-1} = e_t + 0.8e_{t-1}$$

Assuming that you do not know what might be an appropriate model structure for the process, we will begin by trying to estimate it using an AR(p) model, for $p = 1, \ldots, 5$, using the command `arx`. As an alternative to using `filter`, we will here use the command `resid` from the Matlab SIT toolbox. This command directly computes the residual for the given model. The function is called as

```
data=iddata(data);
rar1=resid(ar1_model,data);
rar2=resid(ar2_model,data);
```

with, for instance, the first two model orders. You may then, for comparison plot the residuals (available as `rar1.y`) together with the noise that was used for generating `data` (can be found in `noise.dat`). Note that you need to remove the initial samples just as if you used `filter` command! Also try typing `resid(ar1_model,data)`, i.e., without assigning an output variable.

When examining which model order to use, type `present(arp_model)` (for model order `p`) to see the parameter estimates and some statistics. Try modelling `data` as an AR(p) model, for $p = 1, \ldots, 5$.

QUESTION 4 *In Mozquizto, answer question 4.*

Instead, try to model the data using an ARMA(p, q)–models, for $p, q = 1, 2$, using the command

```
am11_model = armax(data,[p q])
```

Comparing the models using the variance of the residuals, which was the best model of the ones you considered, i.e., an AR(p), for $p = 1, \ldots, 5$, or an ARMA(p, q), for $p, q = 1, 2$? Examine the ACF and PACF as well. Which model would you pick if you did not know it was an ARMA(1,1)?

Be prepared to answer this question when discussing with the examiner at the computer exercise!

2.3 Estimation of a SARIMA-process

We proceed to examine the seasonal ARMA model

$$A(z)\nabla_s y_t = C(z)e_t,$$

We begin with simulating data, and then use the ML-based estimator `pem` to re-estimate the parameters to see how this is done. Simulate the process using

```
rng(0)
A = [1 -1.5 0.7];
C = [1 zeros(1,11) -0.5];
A12 = [1 zeros(1,11) -1];
A_star = conv(A,A12);
e = randn(600,1);
y = filter(C,A_star,e);
y = y(101:end);
plot(y)
```

We here set the seed just to get the same result as used in *Mozquizto*.

Perform basic analysis of the signal using `ACF`, `PACF`, and `normplot`. What characteristics does it have? The ringning behaviour you see in both the `ACF` and the `PACF` indicates strong seasonality. To be able to model the process taking the season into account, you create a differentiated process.

To do so, start by removing the season and then create an `iddata` object

```
y_s  = filter(A12,1,y);
y_s  = y_s( length(A12):end);
data = iddata(y_s);
```

Note that we again have to omit the initial samples - otherwise these initial corrupt samples will harm our continuing processing! Then, set up an initial model with the Matlab object `idpoly`.

```
model_init = idpoly(A,B,C);
```

This object will set the maximum order and initial values of each of the polynomials. If one does not use, for instance, the B-polynomial, then one should set `B=[]`. Building a model to this data, one parameter at a time, could for instance mean starting with an AR(1), as an initial guess, redo the basic analysis and then proceed to an AR(2). Thus, use `pem` to estimate a_1 and a_2 by inputing

```
model_init  = idpoly([1 0 0],[],[]);
model_armax = pem(data,model_init)
```

Form the residual and plot its ACF and PACF. What characteristics does it have?

To add a single parameter of a higher order, i.e., to set some parameters fixed to a value, then set this value in the initial polynomial, i.e., when creating `model_init` and then specify which parameters to be *free* in this polynomial (`idpoly` structure) using:

```
model_init.Structure.c.Free = [zeros(1,12) 1]
```

Here, setting an index to zero implies that this parameter is fixed to its initial value and will not be estimated, whereas a one sets the parameter as free, i.e., to be estimated. For the C-polynomial, we have thus made all parameters fixed except for c_{12}. Include this parameter in the model and estimate a_1, a_2, and c_{12} by inputing

```
model_init = idpoly([1 0 0],[],[1 zeros(1,12)]);
model_init.Structure.c.Free = [zeros(1,12) 1];
model_armax = pem(data,model_init)
```

The main questions to ask are now: Did this remove the season of the resulting model residual? Is the residual white noise? Are the ACF and/or PACF coefficients Gaussian distributed so that you can trust your whiteness test? Are the estimated parameters significant?

Compare the estimated parameters with the values used to simulate the process. Examine how the quality of the estimates improve if you redo the simulation but instead using $N = 10.000$ samples.

QUESTION 5 *In Mozquizto, answer question 5.*

Hint: If (or, rather, when) you want to check that your code is correct, it is always wise to test it using simulated data, preferably using a lot of samples. Simply simulate a process that has roughly the same characteristics as the process you are examining and check that your code predict the simulated process well. For a long sequence, it should work, otherwise, there is most likely a bug somewhere... It is highly recommended that you *always* do this when modelling data!

In summary, the typical modelling steps used when constructing a time series model generally works in accordance with the following steps:

1. Is there a trend? Try removing it.
2. Is there any seasonality? Try removing it.
3. Iterate between
   (a) Which is the lowest order strong AR- or MA-component? Try removing it by including it in the model. Always begin with the strongest AR-component, then inspect the MA-components in the next iteration.
   (b) Is the residual white noise? If not, go to (a). Can you trust your test, i.e., is the ACF and/or PACF Gaussian distributed? If not, what are the consequences?
4. Are all parameter estimates statistically significant? If not, redo the analysis and use a smaller model (order).

Can you improve your model by not removing the trend and/or season separately, but rather incorporating these parts in your model and estimate the corresponding coefficients? Is the resulting model better in some sense, i.e., is the variance of the model residual lower? Do you get a smaller model or one with higher confidence in the parameters without a significant increase in the variance of the model residual?

Be prepared to answer these questions when discussing with the examiner at the computer exercise!

2.4 Estimation on real data

We now proceed to use our knowledge on estimation of SARIMA-models on real data. Load the temperature measurements from Svedala, using the command `load(svedala)`. Using the working order stated above, create a suitable model for this data. Remember to reestimate all your parameters if you change your model.

QUESTION 6 *In Mozquizto, answer question 6.*

What parameter estimates did you get? Can you improve the model by not differentiating the data? Why or why not?

Be prepared to answer these questions when discussing with the examiner at the computer exercise!