Lund University
Mathematical Statistics
Centre for Mathematical Sciences

Time Series Analysis
Fall 2018

Computer Exercise 0

Simulation of ARMA-processes

The purpose of this computer exercise is to illustrate the statistical properties of AR-, MA-, and ARMA-processes. In the exercise, you will study how the choice of poles and zeros of the AR- and MA-polynomials affect different aspects of the process y_t. You will compute the covariance and correlation functions, $r_y(\tau)$ and $\rho_y(\tau)$, respectively, and the corresponding spectral density, $\phi_y(\omega)$. In the coming computer exercises, these parameters will typically be estimated from the observed measurements, but, for now, they are assumed known. We will also examine how one can simulate the processes.

1.1 Preparations before the lab

Read Chapter 3 in the course textbook as well as this guide to the computer exercise. Answer the preparatory exercises in the following section.

1.2 Preparatory exercises

1. Define
   a) an AR(p)-process.
   b) an MA(q)-process.
   c) an ARMA(p, q)-process.
2. Give the conditions for stationarity for an ARMA-process.
3. State the Yule-Walker-equations for an AR(p)-process and the corresponding equations for an ARMA(p, q)-process.
4. State the covariance function for an MA(q)-process.
5. Write down the spectral density for
   a) the AR(p)-process.
   b) the MA(q)-process.
   c) the ARMA(p, q)-process.

1.3 ARMA(p, q)-models in Matlab

The exercise will be done with help of the computer program Matlab and the functions that are in its System Identification Toolbox (SIT). Some extra functions are also available in the laboratory catalog. The function calls of some of the extra functions are attached at the end of this tutorial.

The lab is initialized by the command `initmstat` at the UNIX-prompt, before you start Matlab. After starting, the path to the particular files for this course are obtained by typing `fms051` at the Matlab-prompt. Use `edit` and save your code. You can run your program directly in `edit` via the `F5` key.

ARMA(p, q)-model

An ARMA(p, q)-model

$$y_t + a_1 y_{t-1} + \cdots + a_p y_{t-p} = e_t + c_1 e_{t-1} + \cdots + c_q e_{t-q}$$

can also be written as

$$A(z)y_t = C(z)e_t, \tag{1.1}$$

where

$$A(z) = 1 + a_1 z^{-1} + \cdots + a_p z^{-p},$$
$$C(z) = 1 + c_1 z^{-1} + \cdots + c_q z^{-q}$$

are the generating polynomials. The operator z^{-1} is a delay of one time unit. In Matlab, polynomial is represented as row matrix with the polynomial coefficients as the values of the matrix, e.g. $z^p A(z)$ is represented by the row matrix

```
A = [ 1 a1 ... ap ];
```

Each element in the vector is separated with a space and the angle brackets point out the start and end of the matrix. The matrix is now stored in the variable `A`. In order to examine the contents of the matrix `A` you type `A` at the prompt, getting the answer

```
A =
   1 a1 ... ap
```

Also try e.g. `A(2:3)` which gives the second to the third elements in `A`. In a general matrix `B` , one may similarly extract any submatrix, e.g. `B(:,1:4)` gives all the rows and the first four columns. The variable `A` is thus saved and can be used later, when necessary.

Sometimes it is convenient to specify the zeros of the polynomial

$$z^p A(z^{-1}) = 0$$

instead of the coefficients. The zeros are represented in Matlab as column matrices

```
r = [ r1 r2 ... rp ].'
```

where `.'` denotes the transpose of a matrix. Often, you will also want to include complex elements in `r`. In Matlab, this can be done by writing

```
r = [ 0.3+0.2*i 0.3-0.2*i ].'
```

which gives a column vector representing two complex conjugated numbers. Observe that spaces are not allowed around e.g. the `*`-sign, as spaces are used to separate the matrices element apart.

There are two Matlab-functions `poly` respective `roots`, that render possible translation between the two different representations described above. The function

```
A = poly(r)
```

computes the polynomial coefficients from its zeros, whereas

```
r = roots(A)
```

computes the zeros from the polynomial coefficients.

Apart from the standard Matlab functions, such as `poly` and `roots`, you can easily construct your own functions. In this computer exercise, you will use several such functions, which will simplify your work. A list of some useful functions are placed in the Appendix to this computer exercise.

A useful function that translate complex numbers written in polar form to the form $a + ib$ is the provided function `konvertera`. You provide a $p \times 2$-matrix

```
rpolar = [absr1 argr1; absr2 argr2; ... ; absrp argrp]
```

that contains the zeros to the polynomial $z^p A(z)$ written in polar form, and the function then converts this to the previously described form. (The `;` separate the rows in a matrix in the same way as the space separate the elements in a row from each other.)

One way to write this function is as follows:

```
% FUNCTION r = konvertera( rp )
% Input: rpolar = [ absr1 argr1; ... ; absrp argrp ]
% Output: r = [ r1; r2; ... ; rp ]
%
function [r] = konvertera( rp )
  r = rp(:,1).*( cos(rp(:,2))+sqrt(-1)*sin(rp(:,2)) );
```

It is worth noting that `rp(:,1)` means the first column in `rp`, i.e., the absolute value of the roots. In the same way `rp(:,2)` refer to the argument of zeros. Furthermore, `.*` implies element-by-element multiplication. The operations `cos` and `sin` operates in the same way element-by-element. In order to use the function you store the above text in a file with the name `konvertera.m`. Try

```
help konvertera
```

to see how the help function in Matlab works. It should be stressed that you may call the function naming the input parameter as you please. It is only internally in the function that it is called `rp` .

Another useful function is `A= polypolar(rpolar)`, which take the zeros as input parameters and return the coefficients in the polynomial. Due to numerical reasons there are sometimes a small imaginary part in the coefficients, whereas our coefficients should be real. A simple solution is to use `A=real(A)`, which gives you the real part of the matrix.

There is also a need for multiplying polynomials together. With the polynomial represented as row matrices `A1` and `A2` with their coefficients, this can be done in Matlab using its internal convolution function:

```
A3 = conv( A1,A2 );
```

Covariance function

The covariance function $r_Y(\tau)$ for an ARMA-model can be computed from the Yule-Walker-equations. The function `r=kovarians(C,A,m)` (see Appendix) computes a column matrix `r` with the covariance function values for $\tau = 0, \ldots, \mathtt{m}$ for an ARMA-model specified by the polynomials `A` and `C`. The covariance function `r` can be viewed with the command

```
plot( 0:m, r )
```

and the correlation function is obtained by

```
plot( 0:m, r/r(1) )
```

As an alternative to the function `plot` one can use the `stem` function.

Spectral density function

The spectral density $\phi_y(\omega)$ can be computed from

$$\begin{aligned}
\phi_y(\omega) &= \left|\frac{C(e^{-i\omega})}{A(e^{-i\omega})}\right|^2 \phi_e(\omega) \\
&= \frac{C(e^{i\omega})}{A(e^{i\omega})}\left(\frac{C(e^{i\omega})}{A(e^{i\omega})}\right)^* \phi_e(\omega) \\
&= \left|\frac{e^{-i\omega q}\prod_{k=1}^{q}(e^{i\omega}-z_k^C)}{e^{-i\omega p}\prod_{k=1}^{p}(e^{i\omega}-z_k^A)}\right|^2 \phi_e(\omega) \\
&= \frac{\prod_{k=1}^{q}\left|e^{i\omega}-z_k^C\right|^2}{\prod_{k=1}^{p}\left|e^{i\omega}-z_k^A\right|^2}\phi_e(\omega), \quad |f|<1/2,
\end{aligned}$$

where $()^*$ means complex conjugate, and where z_k^A and z_k^C are the real and complex conjugated poles respective zeros of the process, and $\phi_e(\omega) \equiv \sigma^2$ is the spectral density for the white noise. In Matlab, there is a function `freqz` that computes

$$H(e^{i\omega}) = \frac{C(e^{i\omega})}{A(e^{i\omega})},$$

where $\omega = 2\pi f$. Write

```
[H,w] = freqz( C, A, n );
```

which yields two column matrices `H` and `w` that contain $H(e^{i\omega_k})$ and ω_k, $k = 1, 2, \ldots, n$ where ω_k is equally spaced in the interval $[0, \pi]$. The polynomials A and C are defined as previous. If you choose n as a power of 2 , it is possible to use the FFT, which facilitates the computation. The spectral density can now be computed as

```
R = abs(H).^2;
f = w / (2*pi);
```

when $V(e_t) = \sigma^2 = 1$. The provided function

```
[R,f] = spekt( C, A, n )
```

computes the spectral density for an ARMA-process. Observe that if the ARMA-process has an innovation variance different from 1, you will have to scale with this variance to obtain the correct spectral density. The plotting is done with

```
plot( f, R )
```

or

```
semilogy( f, R )
```

if you want a logarithmic scale of the y-axis. As an alternative, you can use the functions in System Identification Toolbox to compute the spectral density `Ralt` using

```
Ralt = idfrd( idpoly( A, [], C, [], [], sigma2 ) )
```

where `sigma2` is the variance of the innovation process. In these computations, the number of points on the frequency axis are fixed (=128), unless you do not specify your own frequencies (these are then assumed to be given in rad/s in the function, see `help idfrd` to get advice on how to specify your own frequencies).

To get access to the used frequencies, write `Ralt.Frequency` and to get the spectrum write `Ralt.SpectrumData`. Plot the spectrum with the command

```
ffplot( Ralt )
```

Simulation

To simulate an ARMA-model, a sequence $\{e_t\}$ of independent normal distributed random variables is necessary. Generate such a sequence with variance 1, e.g. with the command:

```
e = randn( n, 1 );
```

which gives a row vector `e` with `n` Normal distributed variables. The simulation of

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

is then done with

```
Y = filter( C, A, e );
```

which gives a row vector `Y` with the simulated time series as the result, where `A`, `C` and `e` are defined as previous. If you want to see a plot of the time series, write e.g.

```
plot( Y )
```

To get more information about advanced types of plots, use the command `help`. The function `subplot` is very useful, and also the commands `hold on` and `hold off`.

1.4 Assignments

Amongst the computer files for the laboratory there is a function `armagui` which is made to make it easier to examine the connections between poles, zeros, covariance function, and spectrum for ARMA-models. Write

```
help armagui
```

to get information about the function. Preferably, you can use this function in the future, as an alternative to the other computational methods mentioned above.

AR(1)-process

To get familiar with Matlab and its functions, we will now examine an AR(1)-process,

$$y_t + a_1 y_{t-1} = e_t,$$

where a_1 is a constant, $-1 < a_1 < 1$, and $V(e_t) = 1$.

Preparation exercise: Compute the theoretical correlation function and the spectral density function of an AR(1)-process.

$\rho_y(\tau) =$

$\phi_y(\omega) =$

At which frequency is the maxima of the spectral density when
$a_1 > 0$:
$a_1 < 0$:

Exercise: Start with an AR(1)-process and choose two values of a_1 with different signs but with the same absolute value, close to the unit circle (of course inside); define the matrices `C = 1` and `A = [1 a1]`. Then compute the correlation function and spectrum using Matlab functions, and simulate a part of

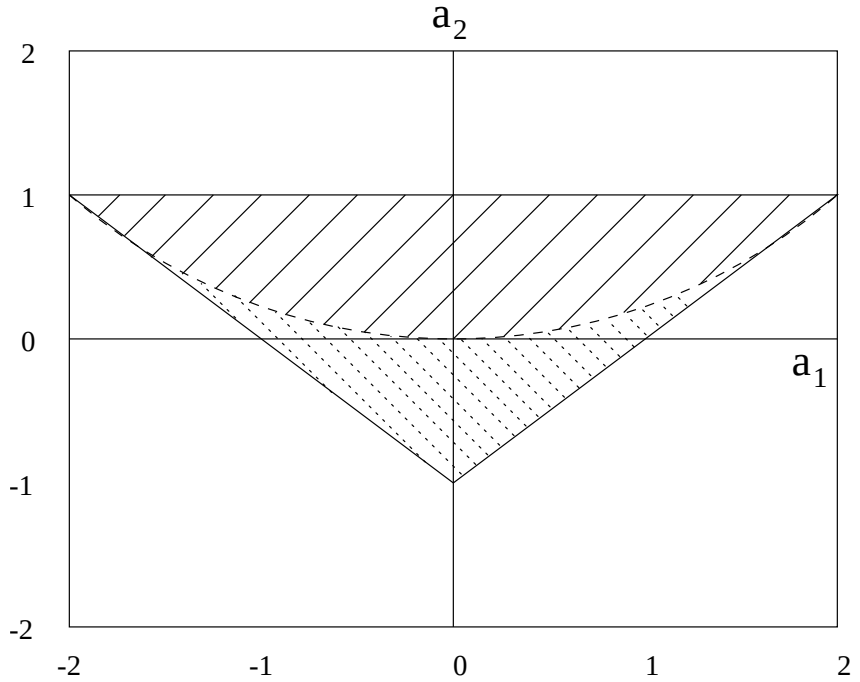

Figure 1.1: *The stationary area for the $AR(2)$-process. When (a_1, a_2) are inside the dashed area then the zeros are complex, in the dotted area they are real and different and on the dashed border curve they are real and equal.*

the process.

Compare the correlation functions and spectrums for both a_1-values. Explain the difference? Discuss intuitively starting from the process model.

AR(2)-process

Consider the process $\{y_t\}$, where

$$y_t + a_1 y_{t-1} + a_2 y_{t-2} = e_t$$

The parameter values that give a stable AR-filter are inside the triangle shown in Figure 1.1 above. The correlation functions have three different structures,

$$\begin{aligned} \rho_y(\tau) &= A_1 x_1^{|\tau|} + A_2 x_2^{|\tau|} && (1) \\ \rho_y(\tau) &= A x_1^{|\tau|}(1 + B \mid \tau \mid) && (2) \\ \rho_y(\tau) &= A\alpha^{|\tau|} \cos(\beta \mid \tau \mid + \omega), && (3) \end{aligned}$$

where the different types appear as the zeros to the equation $z^2 A(z) = 0$, are real-valued and different, real-valued and equal, or complex-valued, respectively.

Exercise: Now work with an AR(2)-process, i.e., let $p = 2, q = 0$ and choose a-parameters in the dashed area in Figure 1.1. Compute the correlation function and the spectral density. Then choose new a-parameters, this time in the dotted area, and do the same computations as before.

Result: Write down the results

a-parameters = ρ_y is of type:

a-parameters = ρ_y is of type:

Theory: The previous exercise showed that it is not easy to find a direct, obvious simple connection between the coefficients a_1 and a_2 and the correlation function, the spectrum, and the process appearance. It is much simpler if you choose to work with the zeros of the characteristic equation:

$$z^p A(z) = 0,$$

where $A(z) = 1 + a_1 z^{-1} + \cdots + a_p z^{-p}$ is the generating polynomial. For the AR(2)-process the characteristic equation is thus

$$z^2(1 + a_1 z^{-1} + a_2 z^{-2}) = 0$$

i.e.,

$$z^2 + a_1 z + a_2 = 0$$

Note that as $a_2 \neq 0$, you can exclude the possibility that $z = 0$ is a zero to the equation. If the process should be stationary, the filter has to be stable, i.e., the equation $z^2 A(z) = 0$ should have all its zeros strictly inside the unit circle. You will now study the connections between the zeros and the process properties more intensively in the case of complex zeros, and therefore complex conjugated. Hence, assume that the equation

$$z^2 + a_1 z + a_2 = 0$$

has complex conjugated zeros with $0 < \alpha < 1$ and $0 < \theta < \pi$. Then, the correlation function ρ_y (for $\tau \geq 0$) is

$$\begin{aligned}
\rho_y(\tau) &= K_1 z_1^\tau + K_2 z_2^\tau \\
&= \alpha^\tau \left(K_1 e^{i\tau\theta} + K_2 e^{-i\tau\theta}\right) \\
&= \alpha^\tau \Big((K_1 + K_2)\cos(\tau\theta) + i(K_1 - K_2)\sin(\tau\theta) \Big) \\
&= \alpha^\tau \left(L_1 \cos(\tau\theta) + L_2 \sin(\tau\theta)\right),
\end{aligned}$$

where L_1 and L_2 are real constants since the correlation function is real-valued. By writing

$$\begin{aligned}
\cos\phi &= \frac{L_1}{\sqrt{L_1^2 + L_2^2}} \\
\sin\phi &= -\frac{L_2}{\sqrt{L_1^2 + L_2^2}}
\end{aligned}$$

you get

$$\rho_y(\tau) = \alpha^\tau \sqrt{L_1^2 + L_2^2} \cos(\tau\theta + \phi).$$

You can thus see that ρ_y decrease exponentially with harmonic oscillations and the parameter θ decides the frequency, while the parameter α gives the amplitude degradation. It is further valid that

- the spectral density has a maximum close to $f = \theta/(2\pi)$
- the spectrum is narrow if α is close to 1
- the spectrum is wide if α is close to 0.

Observe that a spectrum that is nearly constant correspond to nearly uncorrelated random variables, while a narrow spectrum correspond to very correlated random variables.

Exercise: Take two complex conjugated zeros in polar form; choose the absolute value as $\alpha \geq 0.7$ and an argument θ between 0 and π, e.g. $\theta \approx$0.3-0.8 and compute the corresponding polynomial.

Observe: The coefficients in A should be real, but due to numerical uncertainties the functions sometimes will return coefficients with a very small imaginary part. Be sure that the coefficients are real before you continue.

Compute the correlation function and spectrum and perform a simulation. Try three different values on α, so that you get one spectrum with a vague maximum, one with a clear noticeable maximum and one with a very peaked maximum.

Results: Write down the results.

The absolute value: $\alpha_1 =$ $\alpha_2 =$ $\alpha_3 =$

The argument: $\theta =$

Is the correlation function of the correct type, i.e., decreasing harmonic?

Which frequencies?

Does the spectrum has a clear maxima? Where?

Are the processes realizations fairly periodical? With which frequency and wavelength?

Slowly varying AR-processes

Exercise: In the same way as in the previous exercise, it is possible to generate slowly varying AR(2)-processes, i.e., processes with strong positive dependence. How should α and θ be chosen? Choose a suitable θ and try different α so that you get two or three processes with different strong positive dependence. Note that you get different results if you approach $z = +1$ along the real axis or along the unit circle.

Results: Write down your results.

The absolute value: $\alpha_1 =$ $\alpha_2 =$ $\alpha_3 =$

The argument: $\theta =$

Is there a clear maximum in the spectrum? Where?

The stability of the AR-filter

Why is it so important that the generating AR-filter is stable, i.e., that the zeros of the characteristic equation are inside the unit circle? Examine what happen when you try to generate an AR(2)-process with poles on or outside the unit circle. Why is the covariance function or the spectrum in the graphical interface not computed for the process you have chosen?

Results: Write down your results.

$a_1, \quad a_2 =$ $(\alpha =$, $\theta =$ $)$

$a_1, \quad a_2 =$ $(\alpha =$, $\theta =$ $)$

How does the process behave?

The MA(q)-process

Generate an MA(q)-process,

$$y_t = e_t + c_1 e_{t-1} + \cdots + c_q e_{t-q}$$

for some different values of q, e.g. $q = 1, 2, \ldots$ by defining the zeros of the corresponding MA-filter. For which τ:s are the correlation function $\rho_y(\tau) \neq 0$? Discuss the properties of the spectrum starting from the location of the

zeros. Study especially what happens when you have complex zeros close to the unit circle. Compare with the AR(p)-process.

Results:

The ARMA(p, q)-process

Generate ARMA-processes by defining the corresponding poles and zeros in some interesting cases. Plot the poles and zeros in some of these cases.

What happens when the poles and zeros are close to each other?

Let a complex pole-zero-pair move closer to the unit circle and examine how the process, its correlation function and its spectrum change with a decreasing distance to the border. Examine if it matters how the pole and the zero are mutually located by keeping e.g. the pole fixed and rotating the zero around the fixed pole.

Results:

White noise

1. Generate 500 independent $N(0, 1)$-distributed data and plot them.

```
brus = randn( 500, 1 );
plot( brus )
```

2. Study the statistical properties, as e.g.

```
mean( brus )
std( brus )
hist( brus, 30 )
```

Estimate the covariance function and plot it

```
rbrus = covf( brus, 25 );
stem( 0:24, rbrus )
```

Result:

Each of the files `data1.dat` and `data2.dat` contains a sequence of white noise with 100 elements. Read the files into Matlab using the command `load`. Plot the data sequences in different diagrams using `subplot`. The command `subplot(111)` take you back to just one window. Use the command `figure` to make it possible to work with the graphics in different windows. Estimate the correlation function of the two white noise realizations by using `covf`. Remember to subtract the mean value, `mean(data)`, before using `covf`.

Discussion:

1.5 Matlab functions

`konvertera`	convert complex numbers from polar form to $(a + ib)$ form
`polypolar`	gives the coefficients to a polynomial with given zeros
`kovarians`	computes the theoretical covariance function for an ARMA-process
`spekt`	computes the theoretical spectrum for an ARMA-process (uses the function `freqz`)
`filter`	generates an ARMA-process from an input signal
`mean`	computes the mean value of a sequence
`std`	computes the standard deviation of a sequence
`covf`	estimates the covariance function for a sequence with mean value =0 (NB: you have to subtract `mean(data)`)

```
% FUNCTION r = konvertera( rp )
% Input: rpolar = [ absr1 argr1; ... ; absrp argrp ]
% Output: r = [ r1; r2; ... ; rp ]
function [r] = konvertera( rp )
r = rp(:,1).*( cos(rp(:,2))+sqrt(-1)*sin(rp(:,2)) );

function [A]=polypolar(rpolar)
%[A]=polypolar(rpolar)
%rpolar=[absr1 argr1; absr2 argr2; ... ; absrp argrp]
%A=[1 a1 ... ap]

function [R,f]=spekt(C,A,n)
%[R,f]=spekt(C,A,n)
%R=[R1 R2 ... Rn]
%f=[f1 f2 ... fn]
%C=[1 c1 ... cq]
%A=[1 a1 ... ap]
%n aer antalet oenskade vaerde

function [r]=kovarians(C,A,m)
%[r]=kovarians(C,A,m)
%C=[1 c1 ... cq]
%A=[1 a1 ... ap]
%m+1 aer antalet oenskade vaerde

function [rgles]=expa(rkomp)
%[rgles]=expa(rkomp)
%rgles=[1 0 0 0 0 0 -1 0 0 0 0 -1]
%rkomp=[1 1;7 -1;12 -1]
```