Lund University
Centre for Mathematical Sciences
Mathematical Statistics

Stationary stochastic processes
VT 2009

Computer exercise 1 in Stationary stochastic processes, VT 09
Stationary processes

The purpose with this computer exercise is to examine the connection between the covariance function and the spectral density of a stationary stochastic process.

1 Preparations

Carefully read through the entire computer exercise and study especially Section 3. Also read Chapter 5.1-5.2 and 6.1 in the compendium. Answer the exercises in the question dictionary below; you are expected to be able to answer these questions during the exercise.

1.1 Question dictionary

1. What defines a weakly stationary process? Describe in words and using formulas.
2. Define
   a) an AR(p)-process
   b) an MA(q)-process
3. What is a white noise process? Describe the covariance function and the spectral density for white noise in
   a) discrete time.
   b) continuous time.
4. Determine the Yule-Walker-equations for an AR(p)-process.
5. What are the conditions on a_1 and a_2 that ensures that an AR(2)-process is stationary?
6. What is the covariance function and spectral density for an AR(1)-process? Where does the spectral density reach its maximum for $a_1 < 0$ and for $a_1 > 0$, respectively?
7. To ensure stationarity, what is required of the poles of an AR(p)-process?
8. Determine the covariance function of an MA(q)-process.

2 To start the exercise

During the exercise, you will be given a temporary computer account[1]. Log on to the account, open a shell-window (icon with shell) and initiate the Matlab environment by typing:

```
matstat/mc1> initmstat
matstat/mc1> matlab
< M A T L A B S T A R T A R>
>> fms045mas210
WAFO toolbox paths set: normal initiation
>>
```

It is often convenient to gather Matlab commands in a program script; the Matlab editor can be used for this purpose. This is started with the command `edit` in the Matlab window.

3 ARMA(p,q)-models in Matlab

An ARMA(p,q)-model

$$X(t)+a_1X(t-1)+\cdots+a_pX(t-p)=e(t)+c_1e(t-1)+\cdots+c_qe(t-q)$$

can also be written as

$$A(z^{-1})X(t)=C(z^{-1})e(t),$$

where

$$\begin{aligned}A(z^{-1}) &= 1+a_1z^{-1}+\cdots+a_pz^{-p}\\ C(z^{-1}) &= 1+c_1z^{-1}+\cdots+c_qz^{-q}\end{aligned}$$

and where the operator z^{-1} is the unit time delay. If you choose $C(z)\equiv 1$, the process becomes an AR(p)-process, whereas if you choose $A(z)\equiv 1$, it becomes an MA(q)-process. In Matlab, polynomials are represented as vectors; e.g. $z^pA(z^{-1})=z^p+\cdots+a_{p-1}z+a_p$ is represented as `[1 a1 ... ap]`.

3.1 Poles and zeros

It is often convenient to specify the roots instead of the coefficients of the A- and C-polynomials. Let

$$z^pA(z^{-1})=z^p+a_1z^{p-1}+\cdots+a_p=0$$

and denote the roots to this equation $z_1^A,\dots,z_p^A$, i.e.,

$$z^pA(z^{-1})=\prod_{k=1}^{p}(z-z_k^A).$$

[1]This account will be removed after the course. Do not store files you wish to retain on this account.

It is worth noting that for a real-valued process, the roots are always real or pairwise complex conjugated. Similarly, we may form

$$z^q C(z^{-1}) = \prod_{k=1}^{q} (z - z_k^C).$$

The spectral density $R_X(f)$ can then be formed as

$$\begin{aligned} R_X(f) &= \left| \frac{C(e^{-i2\pi f})}{A(e^{-i2\pi f})} \right|^2 R_e(f) \\ &= \left| \frac{e^{-i2\pi f q} \prod_{k=1}^{q} (e^{i2\pi f} - z_k^C)}{e^{-i2\pi f p} \prod_{k=1}^{p} (e^{i2\pi f} - z_k^A)} \right|^2 R_e(f) \\ &= \frac{\prod_{k=1}^{q} \left| e^{i2\pi f} - z_k^C \right|^2}{\prod_{k=1}^{p} \left| e^{i2\pi f} - z_k^A \right|^2} R_e(f), \quad |f| < 1/2, \end{aligned}$$

where $R_e(f) \equiv \sigma^2$ is the spectral density of the driving (white) noise process. The roots of the A- and C-polynomials, z_k^A and z_k^C, are termed poles and zeros, respectively. We will later see how the placement of the poles and zeros affect the shape of the spectrum. In Matlab, the poles and zeros can easily be computed using the command `roots`

```
>> P = roots(A)
>> Z = roots(C)
```

and can then be plotted using

```
>> zplane(Z,P)
```

It is worth noting that when using the function `zplane`, the zeros and poles must be located in column vectors. If row vectors are used as input, `zplane` interprets this as polynomial vectors and plots `roots(C)` and `roots(A)`. Verify this by using the commands `zplane(C,A)` and `zplane(C',A')`. If the poles `P` and zeros `Z` are specified instead and you want to calculate the polynomials `A` and `C`, one may use the command `poly`

```
>> A=poly(P)
>> C=poly(Z)
```

3.2 Spectral density

As discussed above, an ARMA process is specified using two polynomials. Using these polynomials, one may compute the spectral density for the process as

```
>> [H,w]=freqz(C,A);
>> R=abs(H).^2;
```

where `freqz` is a Matlab function that calculates the frequency function corresponding to the polynomials `C` and `A` in a number of equally spaced frequency values (default 512) between $0 \leq \omega < \pi$, with $\omega = 2\pi f$. The squared absolute value of H yields the power spectral density, R. Plot the spectrum using

```
>> plot(w,R)
```

Alternatively, the command `semilogy(w,R)` can be used to plot the spectral density using a logarithmic scale on the y-axis.

3.3 Simulation of ARMA-processes

To simulate the realisation of an ARMA-process, we generate a sequence of independent and normally[2] distributed stochastic variables, $\{e(t)\}$. This can be done as

```
>> m = 0;
>> sigma = 1;
>> e = normrnd(m, sigma, 1, n);
```

or, alternatively,

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

Each call to the above function will generate a row-vector `e` with `n` normally distributed random numbers; here, choose `n`≈ 400. The simulation of the ARMA-process is done using

```
>> x = filter(C, A, e);
```

which result in the vector `x` with the simulated time-series as the result, where `A` and `C` are defined as earlier. You can plot the realisation using

```
>> plot(x)
```

4 Covariance function and spectral density of AR(1)-processes

Using the previous section you should investigate some different AR(1)-processes by using different values of a_1.

1. Create an AR(1)-model in Matlab, and calculate and plot the spectral density. Also calculate and plot the covariance function $r(\tau)$ for $\tau = 1, 2, \cdots, 50$. Try a positive and a negative value of a_1 (with $|a_1| < 1$) and study the result. What is the difference? Plot the poles. Compare with your preparation exercise.

2. Simulate a few different realizations for each process and plot them. Compare with the covariance function and the spectral density.

[2] We note that, in general, $e(t)$ does not need to be normally distributed.

5 ARMAGUI

The function `armagui` may be used to investigate the connections between poles, zeros, covariance functions and spectral densities for ARMA-models. Type

```
>> help armagui
```

to get information how to use it, and start the function by typing

```
>> armagui
```

The help text is also included at the end of this exercise. Use this function in the continuation.

5.1 Poles of an AR(2)-process

The AR(2)-process $\{X(t)\}$,

$$X(t) + a_1 X(t-1) + a_2 X(t-2) = e(t)$$

is described in the compendium, Example 6.2, Chapter 6. For the AR(2)-process to be stationary, the roots z_1^A and z_2^A of the characteristic equation

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

i.e.,

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

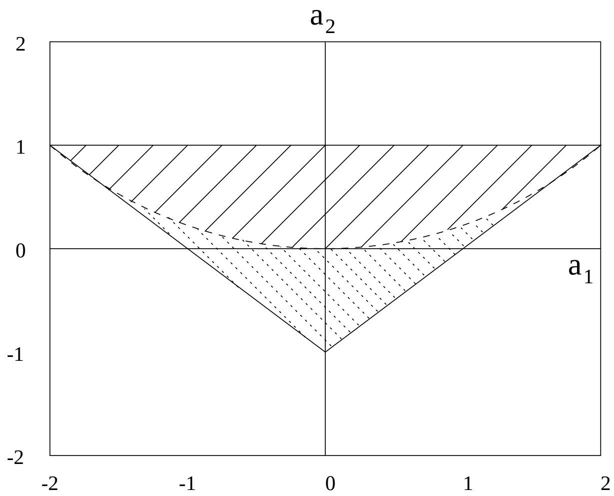
should be located strictly inside the unit circle. (As $a_2 \neq 0$, we can exclude the possibility that $z^A = 0$ is a root of the equation.) The allowed parameter values are in the triangle in Figure 1, as shown at the top of next page. One can show that the covariance function has one of the forms

$$\begin{aligned} r(\tau) &= K_1 z_1^{|\tau|} + K_2 z_2^{|\tau|} && (1) \\ r(\tau) &= K_1 z_1^{|\tau|}(1 + K_2 \mid \tau \mid) && (2) \\ r(\tau) &= K_1 \rho^{|\tau|} \cos(\beta \mid \tau \mid - \phi) && (3) \end{aligned}$$

where the different types appear if the roots to the equation $z^2 A(z^{-1}) = 0$ are 1) real-valued and different, 2) real-valued and equal, or 3) complex conjugated.

1. Choose a-parameters in the dashed area in Figure 1. Construct an AR(2) model according to

```
>> modell.A=[1 a1 a2];
>> modell.C=[1];
```



Figur 1: For the AR(2)-process to be stationary, the coefficients a_1 and a_2 must be inside the triangle.

and import it into `armagui` by writing `modell` in the box and press the import button. Study the poles. Which type of the three above have you constructed? What does the covariance function and spectral density look like?

2. Choose new a-parameters, this time in the dotted area and import into `armagui`. Which type is this?

As is clear from the above exercise, it is not easy to find a direct, obvious and simple connection between the coefficients a_1 and a_2 and the covariance function, spectral density and what the process look like. It becomes much easier if one instead choose to examine the roots of the characteristic equation of the process. Assume that we have the complex conjugated roots

$$z_1 = \rho e^{i2\pi f} \quad \text{and} \quad z_2 = \rho e^{-i2\pi f},$$

where $0 < \rho < 1$ and $0 < f \leq 1/2$. Then, the covariance function $r(\tau)$ is (for $\tau \geq 0$)

$$\begin{aligned} r(\tau) &= K_1 z_1^\tau + K_2 z_2^\tau = \rho^\tau (K_1 e^{i2\pi f\tau} + K_2 e^{-i2\pi f\tau}) \\ &= \rho^\tau \left((K_1+K_2)\cos(2\pi f\tau) + i(K_1-K_2)\sin(2\pi f\tau)\right) \\ &= \rho^\tau (K_3\cos(2\pi f\tau) + K_4\sin(2\pi f\tau)) \end{aligned}$$

where K_3 are K_4 are real constants (since $r(\tau)$ is real-valued). With

$$K_5 = |K_3 + iK_4| = \sqrt{K_3^2 + K_4^2}$$

and

$$\phi = \arg(K_3 + iK_4)$$

we can write

$$\begin{aligned} K_3 &= K_5\cos\phi \\ K_4 &= K_5\sin\phi \end{aligned}$$

and find that

$$r(\tau) = \rho^{\tau} K_5 \cos(2\pi f \tau - \phi).$$

Thus, $r(\tau)$ will decay exponentially with frequency f and decay ρ. We can also conclude that the spectral density has a peak near f and that this peak is narrow if ρ is close to 1 whereas it is wide if ρ is close to 0. Recall that a spectral density that is close to constant correspond to nearly uncorrelated random variables, whereas a narrow spectral density corresponds to heavily correlated variables.

1. Use `armagui` to produce an AR(2)-model with a complex conjugated pair of poles. Restart by removing the poles with the button "Ta bort allt". Add a complex pole, "Komplex pol" (gives you the pole as well as the conjugate). Is the covariance function of the correct type, i.e., is it a decaying harmonic function?
2. Move the poles (easily done with the mouse button). Change the angle frequency and keep quite close to the unit circle. Study the spectral density and the covariance function. How does the frequency change?
3. Change the distance of the poles from the origin. Does the spectral density have clearly visible peaks? How does the covariance function change?

5.2 Stability

What happens if the generating AR-filter is unstable, i.e., if the roots to the characteristic polynomial are outside the unit circle?

Examine this by de-selecting the button "Kräv stabilitet" in `armagui` and move one pole/pole pair outside the unit circle. What happens with the process realization? Why?

5.3 MA(q)-process

MA-processes behave a bit different from AR-processes and we will study MA-processes of some different orders. Restart by removing the poles with the button "Ta bort allt". Add a complex zero using "Komplext nollställe", generating an MA(2)-process.

1. Adding more zeros will increase the order of your MA-process.

   $$X(t) = e(t) + c_1 e(t-1) + \cdots + c_q e(t-q).$$

   Add zeros and examine how the order is affected. For which τ is the correlation function $\rho(\tau) \neq 0$? How is this related to the order.
2. Discuss the appearance of the spectrum from the view of the zeros placement. Study especially what happens when you have complex zeros close to the unit circle.

6 Estimation of parameters in an AR-process

Suppose we have observations $x_1,\ldots,x_N$ of an AR(2)-process. How may a_1 and a_2 be estimated?

We will now examine the sunspot index, which is a measure of sunspot activity. We will study the mean sunspot index for the years 1749 to 1984. Using these data points, we will try to determine the sunspot periodicity using an AR(2)-model.

Load the file `yearMeanSpots`. Plot and study the signal's periodicities. Do a normal distribution plot of the data using `normplot`. Does the data seem to be normal distributed? If not, can you think of a transformation that could make the sunspot data appear more like a normally distributed process?

Estimate an AR(2)-model for the sunspot data. Observe that the expected value of the process is not zero, so you either have to estimate the constant value or subtract the mean from data.

The following commands make a logarithmic transformation of the data and subtracts the mean before estimating the AR parameters:

```
>> yearMeanSpots_l=log(yearMeanSpots);
>> spots=yearMeanSpots_l-mean(yearMeanSpots_l);
>> U=[-spots(2:end-1) -spots(1:end-2)];
>> vhat=inv(U'*U)*U'*spots(3:end);
```

The AR(2)-model is

```
>> modell.A = [1 vhat'];
>> modell.C = 1;
```

Using the estimated AR-parameters, you can compute the periodicity of the sunspot activity. What periodicity do you get? (Hint: Use the Matlab-command `roots`). Does the answer correspond to your knowledge of sunspot periodicities?

6.1 Wrong model for data

Create a random sequence of Gaussian white noise and use this sequence of data to estimate an AR(2)-model, i.e.,

```
>> b=randn(128,1);
>> U=[-b(2:127) -b(1:126)];
>> vhat=inv(U'*U)*U'*b(3:128);
>> modell.A = [1 vhat'];
>> modell.C = 1;
```

Investigate your model in `armagui`. Is it appropriate? (What is the spectral density and covariance function of white noise?) Create another sequence of Gaussian white noise and investigate that model. What is the difference? Why?

7 Matlab–functions

armagui

```
function armagui(action)
% ARMAGUI
%
% armagui  opens a window for  ARMA(p,q)-modeling.
%
% Use the buttons to add poles and zeros. Move them by using the mouse.
% Delete poles and zeros with the right mouse button.
%
% If the button "Kräv stabilitet" is marked, poles and zeros
% are not allowed outside the unit circle.
%
% The button "Simulera" gives a realization of the process.
%
% A- and C-polynomial can be loaded into ARMAGUI by creating, e.g.,
%
% >> modell.A=[1 -0.5];
% >> modell.C=[1 0.4 0.7];
%
% and write the name (i.e.
% "modell") in the "Import"-box and push the button.
%
% Covariance function, spectral density and realizations can be exported
% with the "Export"-button.
% Then a variable with the name written in the "Export"-box is created,
% which includes poles, zeros etc..
%
```