Lund University
Centre for Mathematical Sciences
Mathematical Statistics

Stationary stochastic processes
for scientists and engineers
October 21, 2013

Computer exercise 2 in Stationary stochastic processes

The purpose with this exercise is to examine the connection between the covariance function and the spectral density of a stationary stochastic process and to study different AR- and MA-processes.

1 Preparations

- Download and read `Matlabhints`.
- Study Chapter 6.1-6.2, 6.4 and 7.1-7.2, 7.6 in the textbook.
- Read through the entire computer exercise, paying extra attention to Section 3.
- Answer the exercises in the question dictionary below (use additional sheets, if needed); you are expected to be able to explain your answers during the exercise.
- Find out if you have access to the MATLAB Filter design toolbox `filterdesign`. If not, you have to skip a few examples that use the commands `freqz` and `zplane`.

1.1 Question dictionary

1. What are the conditions for a stochastic process to be weakly stationary?

   Your answer:

2. Define

   a) an $\mathrm{AR}(p)$-process
   b) an $\mathrm{MA}(q)$-process

   Your answer:

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.

   Your answer:

4. Determine the Yule-Walker-equations for an $\mathrm{AR}(p)$-process.

   Your answer:

5. What are the conditions on a_1 and a_2 that ensures that an $\mathrm{AR}(2)$-process is stationary?
   Your answer:

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?
   Your answer:

7. To ensure stationarity, what is required of the poles of an AR(p)-process?
   Your answer:

8. Determine the covariance function of an MA(q)-process.
   Your answer:

2 To start the MATLAB exercise

- Download the necessary m-files from the folder `ssproutines` and the data from `sspdata` and make them accessible from your MATLAB window.
- 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

This section is an introduction to how AR, MA and ARMA models are handled in MATLAB. It is included for you to familiarize yourself with the functions in advance, and for use as a reference during the lab session.

An ARMA(p,q)-model

$$X(t) + a_1 X(t-1) + \cdots + a_p X(t-p) = e(t) + c_1 e(t-1) + \cdots + c_q e(t-q)$$

can also be written as

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

where

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

and where the operator T^{-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. $z^p A(z^{-1}) = z^p + \cdots + a_{p-1}z + a_p$ 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. 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)
```

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)
```

If you have access to the toolbox `filterdesign` you can plot poles and zeros 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')`.

3.2 Spectral density - skip this if you don't have the filterdesign toolbox

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 in `filterdesign` 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 variable N is the 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[1] 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 and spectrum of AR(1)-processes

Using the previous section you should investigate some different AR(1)-processes.

1. Create an AR(1)-model in MATLAB. Use results of the preparation exercise 6 to calculate its spectral density and covariance function. Follow instructions in Section 3.2 to plot the spectral density. Also 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 results. Plot the poles and study their location.

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

Q. What are the coefficients of your AR(1) model? Write down the expressions for its spectral density and covariance function.

Q. Draw rough sketches of the covariance function for $a_1 < 0$ ***and*** $a_1 > 0$**.**

Q. At what frequencies is the spectral density maximum for $a_1 < 0$ ***and*** $a_1 > 0$**,** ***respectively?***

2. Simulate a few different realizations for each process and plot them (see Section 3.3). Analyze and discuss the relationship between realization, covariance function, and spectral density.

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
```

Read the help text; also in `Matlabhints`.

5.1 Poles of an AR(2)-process

1. Use `armagui` to produce an AR(2)-model with a complex conjugated pair of poles. Restart by removing the poles with the button ”Erase everything”. Add a complex pole, ”Complex pole” (gives you the pole as well as the conjugate). 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. Study how the frequency changes.

   Q. Does the frequency increase or decrease as you increase the angle? Why?

2. Change the distance of the poles from the origin and study the covariance function and the spectral density.

   Q. Does the spectral density have clearly visible peaks? Make a rough sketch of the observed spectral density for poles close to the origin.

   Q. Does the exponential decay of the covariance function increase or decrease as you move the poles closer to the origin? Is this the expected result?

5.2 Stability

To see what happens when the generating AR-filter is unstable, i.e., if the roots to the characteristic polynomial are outside the unit circle, de-selecting the button “Request stability”

in `armagui` and move one pole/pole pair outside the unit circle. [2]

Q. 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 "Remove everything". Add a complex zero using "Complex zero", 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 more zeros to form MA(3)-, MA(4)- and MA(5)-processes.

   Q. For which values of τ do the covariance functions of the three processes above become zero? How is this related to the order?

2. Examine 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.

   Q. Sketch two spectral densities, one with complex zeros selected close to the unit circle and second with complex zeros selected close to the origin.

[2] The button "Request stability" actually does more than ensure stability, it forces not only all poles, but also all zeros to be inside the unit circle. Zeros never causes stability issues, but having all zeros inside the unit circle ensures another nice property: minimum phase. This property will not be discussed further in the scope of this course.

6 Filter design with armagui

Different filters can be designed directly by placing poles and zeros in `armagui`. As the input process of ARMA-processes (as well as AR- and MA-) is white noise, the spectral density plot will also be the plot of $|H(f)|^2$ as $R_e(f) = 1$.

1. Try to make some interesting filters by placing poles and zeros. Free your mind and try different possibilities! Sort out how your movements of the poles and zeros effect the frequency function. Create e.g., a low-pass filter, a high-pass filter, a band-pass filter and a band-stop filter.

2. Try to create a low-pass filter using 6 poles and 6 zeros. The filter should have a flat spectral density (not necessarily with power one) below 0.25 and as small energy as possible at frequencies above 0.25. When you have found a model that you are satisfied with, export it. Thereafter, load the file `lowpass.mat` MATLAB which contain a Butterworth low-pass filter. (The function `butter` in the Signal processing toolbox of MATLAB can be used.) Then import the model `lowpass` to `armagui` and compare to your own filter.

   Q. Can you explain why this design work.

3. Create a "notch"-filter, which is a band-stop filter for a very small frequency interval. Use one pair of complex-valued poles and one pair of complex-valued zeros. (Hint: Put the zeros on the unit circle and move the poles close to the zeros and investigate the result).

7 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`.

Q. Does the data seem to be normally distributed? If not, can you think of a transformation that could make the sunspot data appear more like a normally distributed process?

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

Q. Why do we require the data to be zero mean for modelling as an AR-process?

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.

Q. What periodicity do you get? (Hint: Use the MATLAB-command `roots`).