STOCHASTIC SIGNALS AND SYSTEMS

WS 2018

ASSIGNMENT 4

PARAMETER ESTIMATION, AR (p)- PROCESS

Date: 05/02/2019

by

Shinde Mrinal Vinayak (Matriculation No.: 5021349)
Yama Sanath Kumar (Matriculation No.: 5021387)

guided by

Prof. Dr. -Ing. Dieter Kraus

Contents

1 Preparation **2**

1.1 Discrete white noise 2

1.2 Generation of AR(p)-processes 3

1.3 LS-Estimation 4

1.4 Levinson-Durbin-Algorithm 6

2 Exercises with Matlab **8**

2.1 Sample covariance function 8

2.2 Generation of AR(p) - processes 12

2.3 LS-Estimation 14

2.4 Empiric Yule-Walker-Equation 15

2.5 Levinson-Durbin-Algorithm 17

2.6 Estimation of the model order 19

Bibliography **23**

List of Figures

2.1 Covariances found using functions `covfct`(Sample) and `xcov`(Biased) 10

2.2 Covariances found using functions `covfct`(Modified Sample) and `xcov`(Unbiased) . . 11

2.3 Comparison of the covariances 11

2.4 Impulse response of the filter 13

2.5 Estimation of model order 21

Chapter 1

Preparation

1.1 Discrete white noise

Task: Specify the constant component, the covariance function and the spectral density of discrete white noise.

Solution:

A discrete white noise is a discrete signal whose samples are regarded as a sequence of serially uncorrelated random variables with zero mean and finite variance. Mean, $E(Z_t) = \mu = 0$.

The covariance function is:

$$\text{cov}(Z_{t+\tau}, Z(t)) = \sigma_Z^2 \partial(\tau)$$

The spectral density is given by:

$$C_{ZZ}(\Omega) = \sum_{\tau=-\infty}^{\infty} C_{zz}(\tau) e^{-j\Omega t} = \sigma_Z^2$$

The spectral density has a constant value at all frequencies.

1.2 Generation of AR(p)-processes

Task: Make oneself familiar with the MATLAB-function `filter`. The function `filter` is used for the generation of an AR(p)-process. Which values have to be entered for the filter coefficient vector **b**? Which value has to be assigned to the first element a_0 of the filter coefficient vector **a**?

Solution:

The function `filter` can be used to generate an AR(p)-process in MATLAB. $Y = \texttt{filter}(b, a, X)$ filters the data in vector X with the filter described by vectors a and b in-order to create the filtered data, Y. It can be mathematically expressed into an equation as,

$$a_0 Y_t + a_1 Y_{t-1} + ...a_p Y_{t-p} = b_0 X_t + b_1 X_{t-1} + ...b_q X_{t-q}$$

In the case of AR(p)-process with an input Z and output X , it can be stated that:

$$a_0 X_t + a_1 X_{t-1} + ...a_p X_{t-p} = Z_t$$

So comparing the formula obtained by the Matlab function `filter` and the formula expressing an AR(p)- process, the vectors a and b can be entered as follows:

$$\underline{a^T} = (a_0, a_1, ...a_p) : \begin{cases} a_0 = 1 \\ a_i \qquad\qquad \text{i} \neq 0 \end{cases}$$

$$\underline{b^T} = (b_0, b_1, ...b_p) : \begin{cases} b_0 = 1 \\ \\ b_i = 0 \qquad \text{i} \neq 0 \end{cases}$$

1.3 LS-Estimation

Task: How can one determine the least squares estimates of the parameters $a_1, a_2, ..., a_p; {\sigma_z}^2$ for the given observations $x_i (i = 1, 2, ..., N)$? How large has to be the number of observations N? How large must be N, if the estimates shall be determined by solving the empirical Yule-Walker-Equations?

Solution:

For an AR(p)-process,

$$X_t + a_1 X_{t-1} + ...a_p X_{t-p} = Z_t; \qquad for \;\; t = 0, 1, ...N - 1$$

For $t = 0, 1, ..., N - 1$

$$X_{t+p} + a_1 X_{t+p-1} + ...a_p X_t = Z_{t+p}$$

$$N - 1 = t + p$$

$$t = N - p - 1$$

Therefore, we can write in matrix notation:

$$\underbrace{\begin{pmatrix} X_p \\ X_{p+1} \\ . \\ . \\ X_{N-1} \end{pmatrix}}_{\underline{X}} + \underbrace{\begin{pmatrix} X_{p-1}.....X_0 \\ X_p....X_1 \\ . \\ . \\ X_{N-2}....X_{N-p-1} \end{pmatrix}}_{\underline{\underline{H}}} + \underbrace{\begin{pmatrix} a_1 \\ . \\ . \\ . \\ a_p \end{pmatrix}}_{\underline{a}} = \underbrace{\begin{pmatrix} Z_p \\ Z_{p+1} \\ . \\ . \\ Z_{N-1} \end{pmatrix}}_{\underline{Z}}$$

$$\underline{X} + \underline{\underline{H}}.\underline{a} = \underline{Z}$$

So we can estimate the vector a using LS-estimation, and the result is,

$$\hat{\underline{a}} = -(\underline{\underline{H^T}}\ \underline{\underline{H}})^{-1}\underline{\underline{H^T}}\ \underline{X}$$

And the LS error is

$$q(\hat{\underline{a}}) = \sum_{i=p}^{N-1} Z_i^2 = \underline{Z^T}\ \underline{Z} = (\underline{X} + \underline{\underline{H}}.\hat{\underline{a}})^T(\underline{X} + \underline{\underline{H}}.\hat{\underline{a}}) = \underline{X^T}(\underline{\underline{I}} - \underline{\underline{P}})\underline{X}$$

$$\underline{\underline{P}} = \underline{\underline{H}}(\underline{\underline{H^T}}\ \underline{\underline{H}})^{-1}\underline{\underline{H^T}}$$

$$\sigma_Z^2 = \frac{q(\hat{\underline{a}})}{N-p} = \frac{\underline{X^T}(\underline{\underline{I}} - \underline{\underline{P}})\underline{X}}{N-p}$$

$$\hat{C}_{xx}(\Omega) = \frac{\sigma_Z^2}{|1 + \underline{a}_1 e^{j\Omega} + ... + \underline{a}_p e^{jp\Omega}|^2}$$

In order to get the LS-estimate the number of observations should satisfy,

$$\underbrace{N-p}_{\text{number of equations}} \quad \geq \quad \underbrace{p}_{\text{number of coefficients}} \qquad \Rightarrow N \geq 2p$$

Below expressed equation is the set of Yule- Walker equations

$$C_{xx}(m) + \sum_{n=1}^{p} a_n C_{xx}(m-n) = \sigma_z^2 \partial_m \quad , m = 0, ...p$$

So to determine the estimates of the parameters the number of observations has to be N $\geq$ p+1.

1.4 Levinson-Durbin-Algorithm

Task: Which particular property of the coefficient matrix of the following Yule-Walker-Equation system

$$\begin{pmatrix} c_{xx}(0) & c_{xx}(1) & ... & c_{xx}(p-1) \\ c_{xx}(1) & c_{xx}(0) & ... & c_{xx}(p-2) \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ c_{xx}(p-1) & c_{xx}(p-2) & ... & c_{xx}(0) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ . \\ . \\ . \\ a_p \end{pmatrix} = - \begin{pmatrix} c_{xx}(1) \\ c_{xx}(2) \\ . \\ . \\ . \\ c_{xx}(p) \end{pmatrix}$$

permits the application of the Levinson-Durbin-Algorithm? Is the Levinson-Durbin-Algorithm be suited for solving the equation system of the LS-Estimation procedure? Give reasons for your answer. Which advantage offers the Levinson-Durbin-Algorithm with respect to the model order estimation? How can the model order be estimated?

Solution:

It was concluded that,

$$-(\underline{\underline{H^T}}\ \underline{\underline{H}})\underline{\hat{a}} = \underline{\underline{H^T}}\ \underline{X}$$

Levinson-Durbin-Algorithm can be applied if the matrix (HT H) is Toeplitz, i.e. the diagonal elements of the matrix are equal. The Y-W equation system:

$$\begin{pmatrix} c_{xx}(0) & c_{xx}(1) & ... & c_{xx}(p-1) \\ c_{xx}(1) & c_{xx}(0) & ... & c_{xx}(p-2) \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ c_{xx}(p-1) & c_{xx}(p-2) & ... & c_{xx}(0) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ . \\ . \\ . \\ a_p \end{pmatrix} = - \begin{pmatrix} c_{xx}(1) \\ c_{xx}(2) \\ . \\ . \\ . \\ c_{xx}(p) \end{pmatrix}$$

Due to the structure of the matrix (diagonal elements are equal), Levinson-Durbin-Algorithm can be suited to solve this equation system. The advantage of using Levinson-Durbin-Algorithm, when the model order is to be estimated, is that has to be computed for different orders. First the computing for order 2, and then to do the computation for order 3 there is no need to start from the beginning. Hence the computation load is lower.

Estimation of model order: As mentioned earlier, $\hat{\sigma_Z}^2$ is computed for different orders. It is supposed to decrease rapidly at the beginning and then the slope becomes slight. To estimate the model order one of two criteria is applied:

1. Akaike criteria, AIC $(p) = \ln\hat{\sigma_{Z,p}}^2 + p\frac{2}{N}$
2. Rissanen criteria, MDL $(p) = \ln\hat{\sigma_{Z,p}}^2 + p\frac{\ln N}{N}$

And the correct order is the order which satisfies the minimum of the criteria.

Chapter 2

Exercises with Matlab

2.1 Sample covariance function

Task: Write a function `covfct` for determining the sample and the modified sample covariance function. To calculate both sample covariance functions take the first 200 random numbers from `dat1_1`. Display the results and explain the differences between the functions. What indicates that a sequence of random numbers can be interpreted as a realisation of discrete white noise? (Note: In MATLAB exists a similar function `xcov`. Ascertain the correctness of your function `covfct` by comparing the results of the experiments with `covfct` and `xcov`.)

Solution: First, we save the first 200 random numbers from `dat1_1` in a variable `X`. We then wrote a function `covfct` to determine the sample and modified sample covariance function. We also used a similar function, `xcov` built by MATLAB in-order to compare our results with the results from this function. A sequence of random numbers can be interpreted as a realization of discrete white noise if the mean is zero and there is no correlation between two different numbers as the variance of a

discrete white noise is,

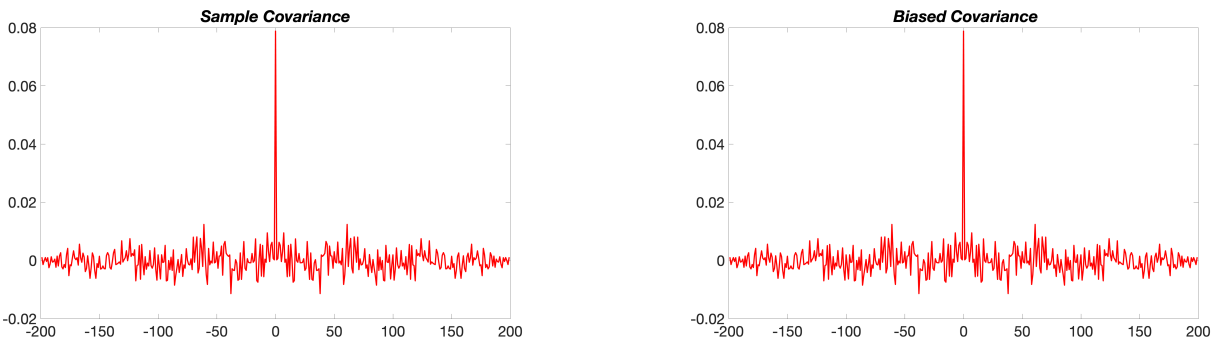
$$C_{xx}(t) = {\sigma_Z}^2 \delta_\tau$$

MATLAB code:

```
clear all
close all
clc

load dat1_1

X = x(1, 1:200);
tau = 0 : 199;
[C,cmod] = covfct(X, tau);
cov_biased = xcov(X,'biased');
cov_unbiased = xcov(X,'unbiased');
for i = 200 : -1 :1
    sample(i+199) = C(i);
    sample(i) = C(200-i+1);
    modified(i+199) = cmod(i);
    modified(i) = cmod(200-i+1);
end
tau = -199 : 199;
getPlot(tau, sample,'Sample Covariance', '','');
getPlot(tau, modified,' Modified Sample Covariance','','');
getPlot(tau, cov_biased,' Biased Covariance','','');
getPlot(tau, cov_unbiased,' Unbiased Covariance','','');
getPlot(sample, cov_biased,...
    'Biased Covariance v/s Sample Covariance', ...
    'Sample Covariance','Biased Covairance');
getPlot(modified, cov_unbiased, ...
    'Unbiased Covariance v/s  Modified Sample Covariance', ...
    ' Modified Sample Covariance ','Unbiased Covariance');

function [c,cmod] = covfct(x, tau)
n = length(x);
```

```
for i = length(tau) : -1 : 1
    co(i) = 0;
    for j = 1 : n-tau(i)
        sum = (x(j+tau(i))-mean(x))*(x(j)-mean(x));
        co(i) = sum+co(i);
    end
    c(i) = (1/n)*co(i);
    cmod(i) = (1/(n-tau(i)))*co(i);
end
end

function getPlot(x , y, title_str , x_label , y_label)
figure()
plot(x,y,'r','LineWidth',2.5)
set(gca,'Title',text('String',title_str , ...
    'FontAngle', 'italic','FontWeight', 'bold'), ...
        'xlabel',text('String', x_label , 'FontAngle', ...
        'italic'),'ylabel',text('String', y_label , ...
        'FontAngle','italic'),'FontSize',28)
end
```

Output: Figure 2.1 (a) shows the output by the `covfct` which Figure 2.1 (b) shows the simulated output of the matlab function `xcov` with biased argument. This means that the mean value is known.

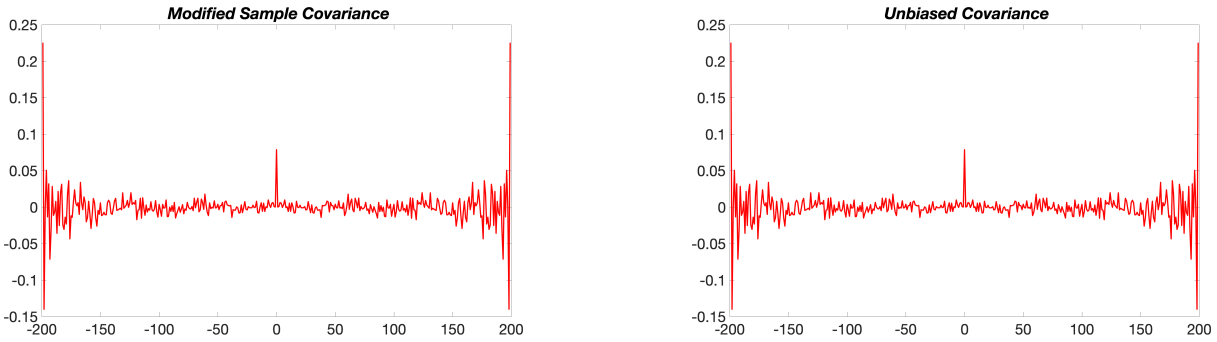


(a) Sample Covariance

(b) Biased Covariance

Figure 2.1: Covariances found using functions `covfct`(Sample) and `xcov`(Biased)

Similarly Figure 2.2 (a) shows the output by the `covfct` which Figure 2.1 (b) shows the simulated output of the matlab function `xcov` with unbiased argument. This means that the mean value is known. The point that there is a peak at the center of this sample covariance function means it shall be white noise in wide sense stationary, the `xcov` also shows almost no cross-correlation and peak at the center with symmetry at both sides.

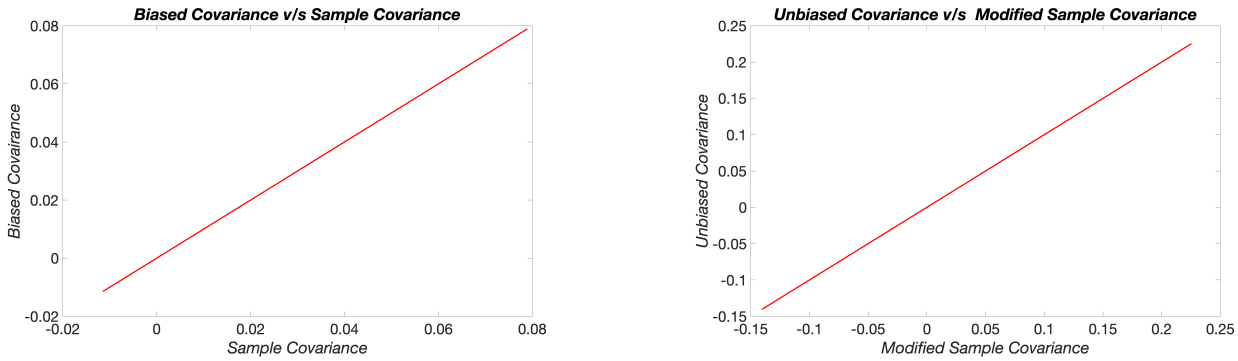


(a) Modified Sample Covariance

(b) Unbiased Covariance

Figure 2.2: Covariances found using functions `covfct`(Modified Sample) and `xcov`(Unbiased)

If one covariance function is plot on x-axis and the other is plot on y-axis, it would be clear that they are equal which is shown in Figure 2.3. It can be concluded that all the outputs are symmetric.



(a) Uniform distribution

(b) Standard normal distribution

Figure 2.3: Comparison of the covariances

Inference: The results obtained by the Matlab function `xcov biased` match the results of the

sample covariance function, and the results of `xcov unbiased` match the results of the modified sample covariance function. The sample covariance function is more accurate in case of a large number of observations because the variance converges to zero when the number of observations is getting very large. However, the modified sample variance function gives better results when the number of observations is limited.

2.2 Generation of AR(p) - processes

Task: Load `dat1_2` that includes a realization of white noise. For the generation of an AR(p) process you have to filter the white noise by a recursive filter with the filter coefficients $a_1 = 0.5, a_2 = 0.3, a_3 = 0.1, a_4 = 0.7, a_5 = 0.3$. Use the MATLAB-function `filter`. Save the white noise and the AR (ρ) - process for later use in `dat4_1`.

Solution: The data which includes a realization of white noise is loaded at first. This white noise is filtered using the recursive filter with the filter coefficients $a_1 = 0.5, a_2 = 0.3, a_3 = 0.1, a_4 = 0.7, a_5 = 0.3$.We used the MATLAB function `filter` for this process and both white noise as well as AR(ρ) - processes is saved in `dat4_1`

MATLAB code:

```
clear all
close all
clc

load dat1_2;
a = [1 0.5 0.3 0.1 0.7 0.3];
figure()
```

```
impz(a);
set(gca,'FontSize',20)
b = 1;
AR = filter(b,a,x1);
save dat4_1 x1 AR
```

Output: Figure 2.4 shows the impulse response of the filter.

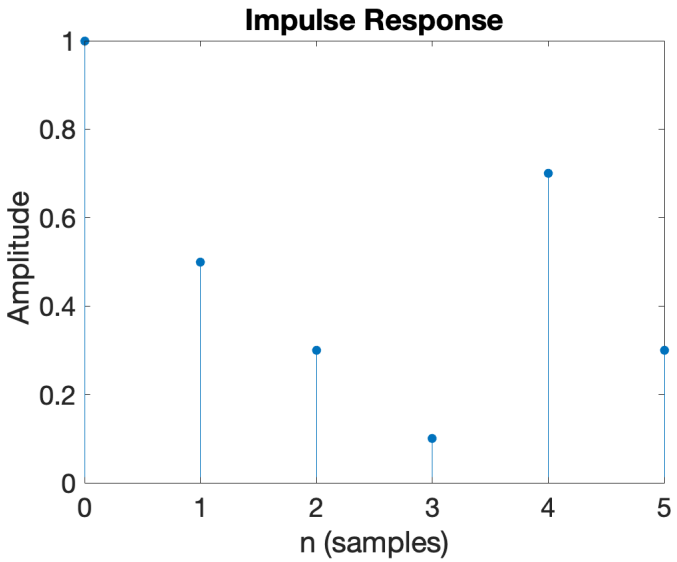


Figure 2.4: Impulse response of the filter

Inference: The Auto-Regression process is been realized by the filtering the normally distributed random numbers using the Matlab function filter, with the given coefficients: a = [1,0.5,0.3,0.1,0.7,0.3], b = [1]. The mean value of the impulse response is zero. After obtaining the coefficients the AR(p)-process is being generated, and the data are saved for further processing.

2.3 LS-Estimation

Task: Carry out a LS - Estimation of the parameters $a_i (i = 1, 2, ..., 5)$ and the variance ${\sigma_Z}^2$of the AR(p)-process.

Solution: The LS - Estimation of the parameters $a_i (i = 1, 2, ..., 5)$ and the variance ${\sigma_Z}^2$of the AR(p)-process is carried out by the following code.

MATLAB code:

```
clear all
close all
clc

load dat4_1
a = 5;
I = eye(length(AR)-a);
for i = length(AR)-a : -1 :1
    x(i,1) = AR(i+a);
    for j = 1 : a
        H(i,j)= AR(a+i-j);
    end
end
est_a = -(inv(H'*H)) * H' * x;
P_orthogonal = I - H * inv(H'*H) * H';
variance = (x'*P_orthogonal*x)/(length(AR)-2*a);
fprintf('\n The LS-estimation of the parameters:')
fprintf('\na1 = %f \na2 = %f \na3 = %f \na4 = %f \na5 = %f',...
    est_a(1), est_a(2), est_a(3), est_a(4), est_a(5))
fprintf('\n\n Variance = %f', variance)
```

Output: The output after execution of the code is shown below.

```
 The LS-estimation of the parameters:
a1 = 0.474946
a2 = 0.288384
a3 = 0.091040
a4 = 0.672543
a5 = 0.288343

 Variance = 0.893504>>
```

Inference: The parameters and the variance is calculated and shown above.

2.4 Empiric Yule-Walker-Equation

Task: Determine the first 11 values of the sample covariance function, e.g. $\hat{c}_{xx}(0), ...\hat{c}_{xx}(10)$ using function `covfct`. Estimate the parameters $a_1(i = 1, 2, ..., 5)$ and ${\sigma_z}^2$ by solving the empirical Yule-Walker-Equation via the Gaussian elimination algorithm. (Note: Use the MATLAB-function `toeplitz` to generate the coefficient matrix and read the MATLAB-help for the operator `backslash`.)

Solution: The MATLAB-function `toeplitz` is used to generate the coefficient matrix. The estimation of the first 11 values of the sample covariance function is carried out by the following code.

MATLAB code:

```
clear all
close all
clc

load dat4_1
tau = 0 : 10;
[C, cmod] = covfct(AR, tau);
```

```
for i = 5 : -1 :1
    x(i)=C(i);
    c(i)=C(i+1);
end
M = toeplitz(x);
a = -M\c';
variance = x(1) + c*a;
fprintf('The first 11 values of the sample covariance are:')
for i = 1 : 1 : 11
    fprintf('\nC_xx(%d) = %f',i-1,C(i))
end

fprintf('\n \nThe estimated parameters are:')
fprintf('\na1 = %f \na2 = %f \na3 = %f \na4 = %f \na5 = %f',...
    a(1), a(2), a(3), a(4), a(5))
fprintf('\n\nVariance = %f', variance)

fprintf('\n\nThe coefficient matrix is:')
fprintf('\n')
disp(M)
function [c,cmod] = covfct(x, tau)
n = length(x);
for i = length(tau) : -1 : 1
    co(i) = 0;
    for j = 1 : n-tau(i)
        sum = (x(j+tau(i))-mean(x))*(x(j)-mean(x));
        co(i) = sum+co(i);
    end
    c(i) = (1/n)*co(i);
    cmod(i) = (1/(n-tau(i)))*co(i);
end
end
```

Output: Applying `covfct` function to the AR(p)-process, the sample covariance function is obtained and the first 11 values are listed in Fig 2.5. The coefficient matrix, estimated parameters as well as

the variance is the output of the MATLAB code.

```
The first 11 values of the sample covariance are:
C_xx(0) = 2.100452
C_xx(1) = -0.859199
C_xx(2) = -0.207025
C_xx(3) = 0.789476
C_xx(4) = -1.395629
C_xx(5) = 0.423801
C_xx(6) = 0.482061
C_xx(7) = -0.674281
C_xx(8) = 0.861079
C_xx(9) = -0.182713
C_xx(10) = -0.468456

The estimated parameters are:
a1 = 0.476614
a2 = 0.288056
a3 = 0.091331
a4 = 0.669474
a5 = 0.289500

Variance = 0.891768

The coefficient matrix is:
    2.1005   -0.8592   -0.2070    0.7895   -1.3956
   -0.8592    2.1005   -0.8592   -0.2070    0.7895
   -0.2070   -0.8592    2.1005   -0.8592   -0.2070
    0.7895   -0.2070   -0.8592    2.1005   -0.8592
   -1.3956    0.7895   -0.2070   -0.8592    2.1005
```

Inference: The parameters and the variance is calculated and shown above.

2.5 Levinson-Durbin-Algorithm

Task: Write a function `levindur` for implementation of the Levinson-Durbin-Algorithm. Estimate with this function and the sample covariance function `covfct` the parameters $a_1(i = 1, 2, ..., 5)$ and the variance ${\sigma_z}^2$.

Solution: A function `levindur` is written using the Matlab function Levinson for the implementation of the Levinson-Durbin-Algorithm.

MATLAB code:

```
clear all
close all
clc

load dat4_1;
tau = 0:5;
[C, cmod] = covfct(AR,tau);
[a, variance] = levindur(C,length(tau));
fprintf('\n \nThe estimated parameters are:')
fprintf('\na1 = %f \na2 = %f \na3 = %f \na4 = %f \na5 = %f',...
    a(1), a(2), a(3), a(4), a(5))
fprintf('\n\nVariance = %f', variance)

function [a,sigma] = levindur(x,p)
[a,sigma] = levinson(x,p);
end

function [c,cmod] = covfct(x, tau)
n = length(x);
for i = length(tau) : -1 : 1
    co(i) = 0;
    for j = 1 : n-tau(i)
        sum = (x(j+tau(i))-mean(x))*(x(j)-mean(x));
        co(i) = sum+co(i);
    end
    c(i) = (1/n)*co(i);
    cmod(i) = (1/(n-tau(i)))*co(i);
end
end
```

Output: Again the parameters $a_i (i = 1, 2, ..., 5)$ and ${\sigma_Z}^2$ are estimated and the results are given as follows.

```
The estimated parameters are:
a1 = 1.000000
a2 = 0.476614
a3 = 0.288056
a4 = 0.091331
a5 = 0.669474

Variance = 0.891768>>
```

Inference: We can infer that the the values are similar as obtained by solving empirical-Walker-Equation.

2.6 Estimation of the model order

Task: Estimate the parameters $a_1(i = 1, 2, ..., 5)$ and ${\sigma_z}^2$ as in section 4.2.5, but now for the wrong model order $p = 4$ and $p = 6$. Compare the estimated parameters with those obtained in section 4.2.5. Which consequences have an underestimation or an overestimation of the model order? Estimate now for the model order $k(k = 1, 2, ..., 10)$ and the variance ${\sigma_{Z,k}}^2$ with the Levinson-Durbin-Algorithm. Present the result in a diagram. Draw also the values of the Akaike and Rissanen criterion and deduce from them the model order p of the AR(p)-process.

Solution: The parameters $a_i(i = 1, 2, ..., 5)$ and ${\sigma_Z}^2$ are estimated for two model orders p=4 and p=6 using the code given below. **MATLAB code:**

```
clc
clear all
close all
load dat4_1

for i = 10 : -1 : 1
tau = 0 : i;
[C, cmod] = covfct(AR, tau);
```

```
[a{i}, sigma(i)] = levindur(C, length(tau));
if i == 4 || i == 5 || i == 6
    fprintf('\n \nThe estimated parameters for p = %d are:',i)
    fprintf('\n%f',a{i})
    fprintf('\nVariance = %f', sigma(i))
end
Akaike(i) = log(sigma(i)) + i*2/length(AR);
Rissanen(i) = log(sigma(i)) + i*log(length(AR))/length(AR);
end
figure()
hold on
plot(sigma,'--om','LineWidth',2.5,'MarkerSize',10, ...
    'MarkerEdgeColor','k','MarkerFaceColor','m')
plot(Akaike,'--ob','LineWidth',2.5,'MarkerSize',10, ...
    'MarkerEdgeColor','k','MarkerFaceColor','b')
plot(Rissanen,'--or','LineWidth',2.5,'MarkerSize',10, ...
    'MarkerEdgeColor','k','MarkerFaceColor','r')
set(gca,'Title',text('String','Estimation of model order', ...
    'FontAngle', 'italic','FontWeight', 'bold'), ...
        'xlabel',text('String', 'Model order, p', 'FontAngle', ...
        'italic'),'ylabel',text('String','Variance, \sigma', ...
        'FontAngle','italic'),'FontSize',28)
hold off
legend('Estimated Variance','Akaike cretiria','Rissanen cretiria')
hold off

function [a,sigma] = levindur(x,p)
[a,sigma] = levinson(x,p);
end

function [c,cmod] = covfct(x, tau)
n = length(x);
for i = length(tau) : -1 : 1
    co(i) = 0;
    for j = 1 : n-tau(i)
        sum = (x(j+tau(i))-mean(x))*(x(j)-mean(x));
```

```
        co(i) = sum+co(i);
    end
    c(i) = (1/n)*co(i);
    cmod(i) = (1/(n-tau(i)))*co(i);
end
end
```

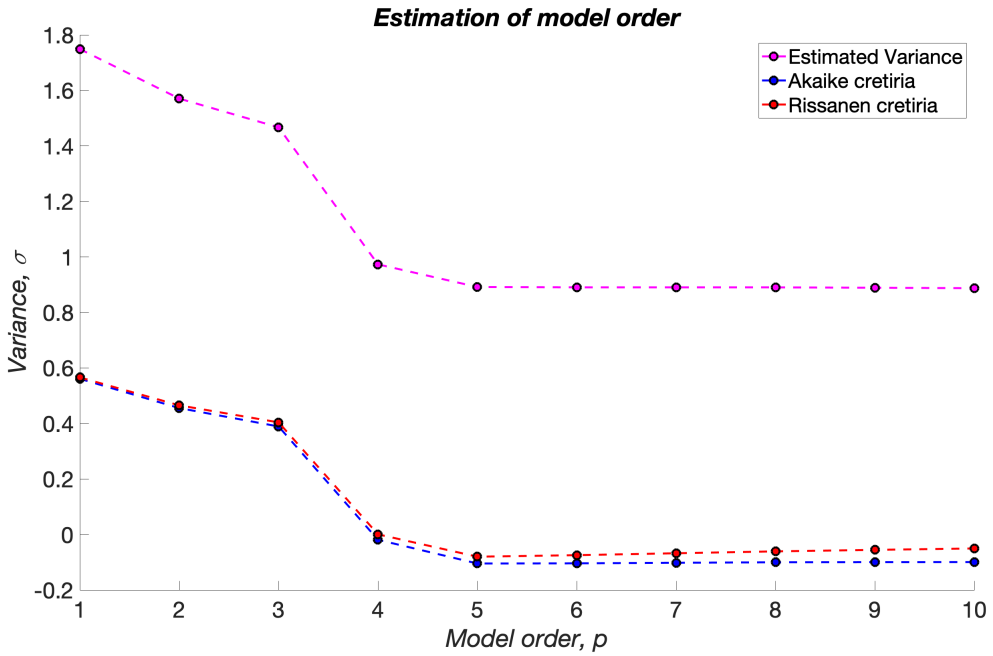


Figure 2.5: Estimation of model order

Output: It is evident that for lower order the variance is higher, hence high error. And for higher order it is less. Comparing the resulting parameters for model orders $p = 4$ and $p = 6$ with the parameters estimated for the order $p = 5$, it can be seen that slight differences between the parameters estimated for the order $p = 6$ and $P = 5$. Whereas the differences are big between the parameters estimated for the model with order $p = 4$ and $p = 5$. So it is better to have an overestimation of the model order than an underestimation, because underestimation will lead to a big error. In order to determine the correct model order, the variance is estimated for the orders from 1 to 10.

```
The estimated parameters for p = 6 are:
1.000000
0.487392
0.312980
0.094731
0.680198
0.307244
0.037230
Variance = 0.890532

The estimated parameters for p = 5 are:
1.000000
0.476614
0.288056
0.091331
0.669474
0.289500
Variance = 0.891768

The estimated parameters for p = 4 are:
1.000000
0.308671
0.285547
0.008665
0.580114
Variance = 0.973344>>
```

In information theory, the Akaike and Rissanen criterion is considered for computing the statistical quality of data. Here, the variance is measured along with its order. For $p = 5$, the minimum could be achieved for both the model (so this gives confidence). Rissanen (MDL) has slightly higher value for the slope than Akaike Information criterion, this is due to penalty factor (offset in equation), which is meant to be increase due to logarithmic behavior with increasing order.

Inference: It can be seen that the variance in all cases has a minimum at the order $p = 5$ which is the model order for the AR(p)-process.

Bibliography

[1] Prof. Dr.-Ing. Dieter Kraus, *"Stochastic Signals and Systems- Probability Theory* lecture notes.

[2] Matlab Documentation

[3] IIntuitive Probability and Random Processes Using MatLab, Steven M. Kay

[4] C. Grinstead, J. Snell and J. Snell, Introduction to probability, 1st ed. Providence, RI: American Mathematical Society, 1997.