HSB
Hochschule Bremen
City University of Applied Sciences

STOCHASTIC SIGNALS AND SYSTEMS

WS 2018

ASSIGNMENT 4

PARAMETER ESTIMATION, AR (p)- PROCESS

Date: 30/01/2019

by

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

guided by

Prof. Dr. -Ing. Dieter Kraus

Contents

List of Figures

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:

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:

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:

1.4 LS-Estimation

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:

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,

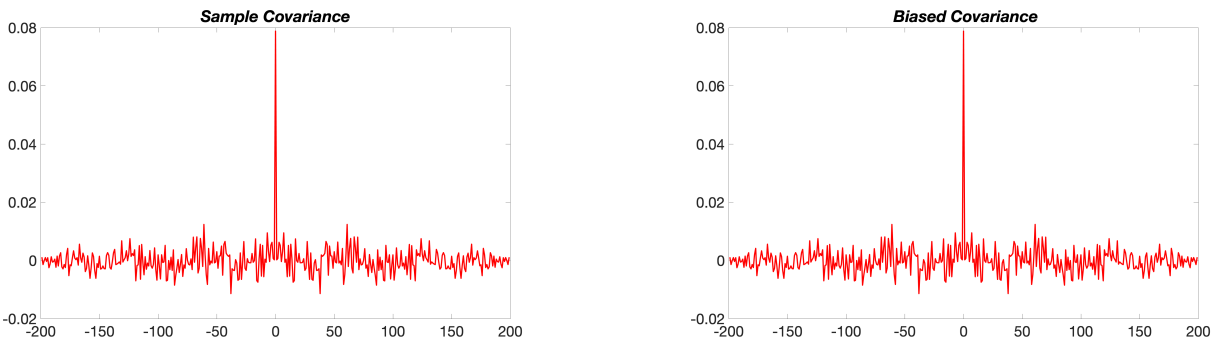
$$C_{xx}(t) = {\sigma_Z}^2 \delta_\tau$$

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.

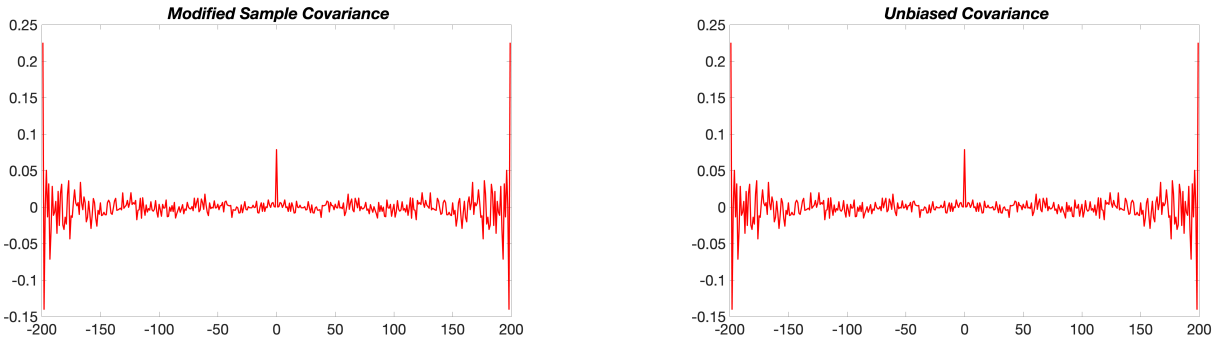


(a) Sample Covariance

(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.

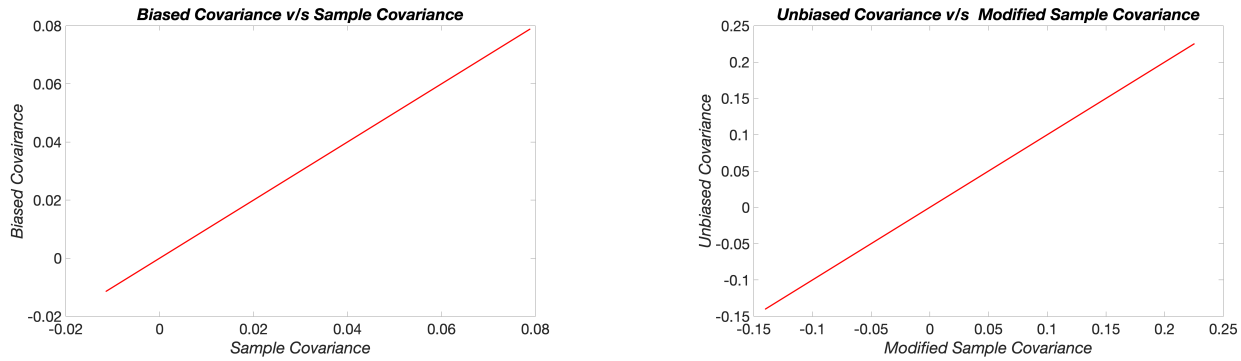


(a) Modified Sample Covariance

(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.

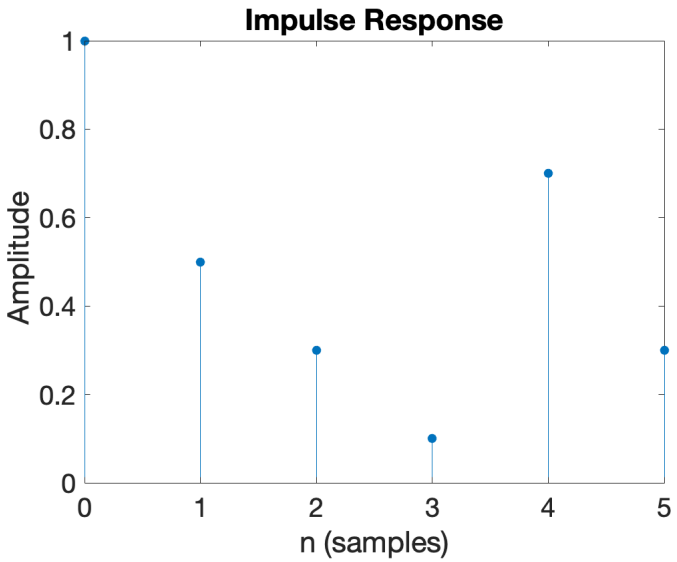


Figure 2.4: Impulse response of the filter

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 Estimating the order of a polynomial

Task: Estimate the order p of the polynomial model. Therefore you have to depict the estimated variance versus the model order $p = 1, 2, ..., 10$. Consider the order that provides the smallest variance as the correct order. Estimate for that order the parameters a_i for $i = 1, 2, ..., p$.

Solution: The same function `LSE` is used to obtain the variance of the polynomial model. The polynomial order p, is taken from 1 to 10 and the parameters at each order are shown in the output.

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: The written code plots variance versus polynomial order. It can be seen that the variance decreases drastically and after a point it remains constant. The plot can be seen in Figure 2.3.

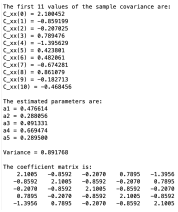


Figure 2.5: Variance as a function of the polynomial order

The output of the execution is shown below. The parameters for polynomial order 1 to 10 are printed and shown.

Inference: The variance reduces from 43.87 at $p = 1$ to 0.767 at $p = 3$. Hence the polynomial of order 3 is considered to be the correct order. This is because, as the order increases further than 3, it remains almost constant. The parameters for the polynomial of order 3 are: a = 4.77091, b = 2.47796, c = 1.187414 and d = -0.64021.

2.5 Comparison of observations and estimated model

Task: For each of the models estimated above show the observations of the model $(x_i, y_i) : i = 1, 2, ..., N$ and the corresponding reconstructed model curve $(x_i, g(x_i)) : i = 1, 2, ..., N$ in one figure.

Solution: We created a function `compare` inside the function `plotGraph` which used the values of the parameters from the previous exercise i.e., when $p = 3$ for polynomial model and $p = 1$ for the rest of the models to compare with the observed values from the data set which was previously created.

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: The output of the code is shown below. The estimated value is shown by a line while the observed value is represented by dots.

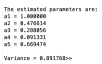


Figure 2.6: LS-adjustment to a circle: estimating the centre

Inference: We can infer that the observed value almost coincides with the estimated value.

2.6 LS-adjustment to a circle: estimating the centre

Task: Load `dat3_2` containing a 100×2 matrix that represent the coordinates of 100 measurement points in the xy-plane. Take a look at the measurement points and give from this the coordinates of the centre location.

Solution: `dat3_2` was loaded as mentioned in the task. It contains the matrix of data set xy. We estimate and find the centre of the circle to be located at (4,2).

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
```

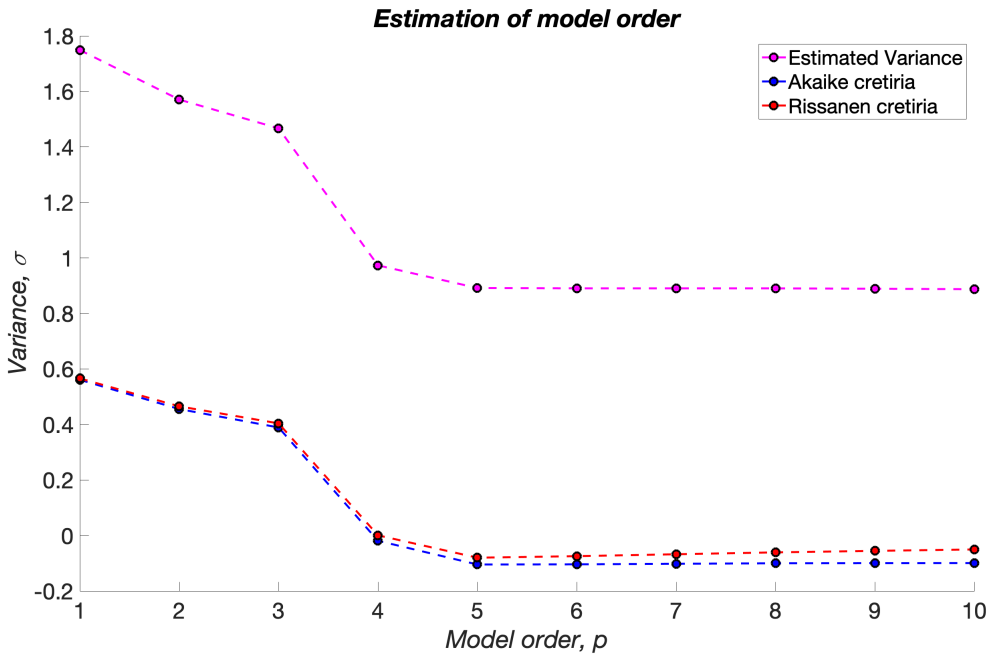


Figure 2.7: LS-adjustment to a circle: estimating the centre

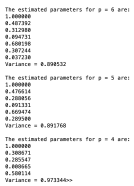


Figure 2.8: LS-adjustment to a circle: estimating the centre

Inference: The measurement points nearly takes the shape of a circle with centre at (4,2)

Bibliography

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

[2] Matlab Documentation

[3] Intuitive Probability and Random Processes using MATLAB, Steven M. Kay, Kluwer Academic Publishers, 2006

[4] Signal Processing Fundamentals of Statistical Signal Processing: Estimation Theory, Steven M. Kay, University of Rhode Island