Imperial College London

Department of Mathematics

Time Series Coursework

Author:
Hermine Tranié

Lecturer:
Dr E. Cohen

Submitted in partial fulfillment of the requirements for the BSc degree in Mathematics with Statistics of Imperial College London

December 2020

Question 1

Part a

In this function *S_AR(f,phis,sigma2)* , the goal is to evaluate the parametric form of the spectral density function for an AR(p) process on a designated set of frequencies. Note that by definition from the lecture notes [1] we have that an AR(p) is defined as follows:

$$X_t - \phi_{1,p}X_{t-1} - ... - \phi_{p,p}X_{t-p} = \epsilon_t$$

where $\{\epsilon_t\}$ is a zero mean white noise process with variance σ^2, and $[\phi_{1,p}, ...\phi_{p,p}]$ the vector of parameters phis. Now we can define the spectral density function for an AR(p) as follows:

$$S_X(f) = \frac{\sigma_\epsilon^2}{|1 - \phi_{1,p}e^{-i2\pi f} - ... - \phi_{p,p}e^{-i2\pi fp}|^2}$$

```
function S = S_AR(f,phis,sigma2)
% STEP 1: SET VALUES
p = length(phis); % find order of process
param = 1; % set first value

% STEP 2: LOOP TO COMPUTE SUM
for k = 1:p
    param = param - phis(k)*exp(-1i*2*pi*f*k);
end

% STEP 3: CONCLUSION
param = abs(param).^2;
S = sigma2./param;
```

Part b

In this function *AR2_sim(phis,sigma2,N)*, the goal is to simulate a Gaussian AR(2) process of length N by using a burn in method. Let us define an AR(2) process [1]:

$$X_t = \phi_{1,2}X_{t-1} + \phi_{2,2}X_{t-2} + \epsilon_t$$

where $\{\epsilon_t\}$ is a zero mean white noise process with variance σ^2.

```
function X = AR2_sim(phis,sigma2,N)
% STEP 1: SET VALUES
X = zeros(1,N+100);
X(1) = 0;
X(2) = 0;
white_noise = randn(1,N+100)*sqrt(sigma2); %randn simulates normally distributed random numbers

% STEP 2: LOOP
for i = 3:N+98
    X(i) = phis(1)*X(i-1) + phis(2)*X(i-2) + white_noise(i);
end

% STEP 3: TRIM VALUES OF TS AND DISP FROM t=101 to t=100+N
X = [X(101:end)];
```

Part c

In this function *acvs_hat(X,tau)*, the goal is to compute the estimate of autocovariance $\hat{s}_\tau^{(p)}$ for an input time series X, at designated values of tau. We can define this estimator [1] as follows (with the zero mean assumption given to us at the beginning of this coursework):

$$\hat{s}_\tau^{(p)} = \frac{1}{N}\sum_{t=1}^{N-|\tau|} X_t X_{t+\tau}$$

```
function s_hat = acvs_hat(X,tau)
% STEP 1: SET VALUES
N = length(X);
est = zeros(1,length(tau));

% STEP 2: LOOP
for k = 1:length(tau)
   est(k) = (1/N)*sum(X(1:(N-tau(k))).*X((1+tau(k)):N));
end

% STEP 3: CONCLUSION
s_hat=est;
```

Question 2

Part a

The goal of the *periodogram(X)* function is to compute the periodogram [1] at the Fourier frequencies of a time series stored as a vector (line) X. Let's define it as follows:

$$\hat{S}^{(p)}(f) = \sum_{\tau=-(N-1)}^{(N-1)} \hat{s}_\tau^{(p)} e^{-i2\pi f\tau}$$

Note that we can sub in the estimate of the autocovariance defined in Q1c above to get:

$$\begin{aligned}\hat{S}^{(p)}(f) &= \frac{1}{N} \sum_{\tau=-(N-1)}^{(N-1)} \sum_{t=1}^{N-|\tau|} X_t X_{t+\tau} e^{-i2\pi f\tau} \\ &= \frac{1}{N} \sum_{j=1}^{N} \sum_{k=1}^{N} X_j X_k e^{-i2\pi f(k-j)} \\ &= \frac{1}{N} |\sum_{t=1}^{N} X_t e^{-i2\pi f t}|^2\end{aligned}$$

Here we have that $\sum_{t=1}^{N} X_t e^{-i2\pi f t}$ is the Fourier transform of time series X which we can perform using the Fast Fourier Transform algorithm (in-built Matlab *fft* function) as follows: (note that the algorithm computes the transform at the Fourier frequencies $f_k = \frac{k}{N}, k = 0, ..., N-1$.

```
function [pd,f] = periodogram(X)
N = length(X);
pd = (1/N)*abs(fft(X)).^2; % periodogram
f = [0:N-1]/N; % fourier frequencies
end
```

The goal of the *direct(X)* function is to compute the direct spectral estimate at the Fourier frequencies of a time series stored as a vector (line) X using the Hanning taper. Let's define the direct spectral estimate [1] as follows:

$$\hat{S}^{(d)}(f) = |\sum_{t=1}^{N} h_t X_t e^{-i2\pi f t}|^2$$

where $h_t = \frac{1}{2}\left(\frac{8}{3(N+1)}\right)^{\frac{1}{2}}\left(1 - \cos\left(\frac{2\pi t}{N+1}\right)\right), t = 1, ..., N$ is the Hanning taper and $\sum_{t=1}^{N} h_t X_t e^{-i2\pi f t}$ is the Fourier transform of the tapered time series X which we can perform using the fft in-built function once again as follows:

```
function [dr,f] = direct(X)
N = length(X);
t = 1:N;
```

```
h = 0.5*((8/(3*(N+1)))^0.5)*(1-cos(2*pi*t/(N+1))); % Hanning taper
dr = abs(fft(h.*X)).^2; % direct spectral estimate
f = [0:N-1]/N; % fourier frequencies
end
```

Part b

In this exercise, we want to explore the behavior of the empirical bias of the periodogram and the direct spectral estimate of an AR(2) process with complex conjugate roots:

$$z_1 = \frac{1}{r}e^{-i2\pi f'} and z_2 = \frac{1}{r}e^{i2\pi f'}$$

To do this, we will use our function from Q1 *AR2_sim(phis,sigma2,N)*. We still need to compute the vector of constants [phis] which is defined [1] as follows for our AR(2) process: For an AR(2) process, we have the following characteristic equation:

$$\phi(z) = 1 - \phi_{1,2}z - \phi_{2,2}z^2 = (1-az)(1-bz) = 1-(a+b)z+abz^2$$

So the roots are $z_1 = \frac{1}{a}$ and $z_2 = \frac{1}{b}$. The coefficients are $\phi_{1,2} = (a+b), \phi_{2,2} = -ab$. Then $a = re^{i2\pi f'}$ and $b = re^{-i2\pi f'}$ and $\phi_{1,2} = 2r\cos(2\pi f')$ and $\phi_{2,2} = -r^2$. Let's now define the empirical bias as the difference between the mean of the periodogram or direct spectral estimate values and the actual spectral density function value.

```
% PART A. Simulate 10000 realizations, of length N=16
% STEP 1: SET PARAMETERS
    % Step 1.1: Define phi & sigma2
        r = 0.95;
        f_dash = 1/8;
        phis = [2*r*cos(2*pi*f_dash) , -r^2]; % by definition from the notes top of p51
        sigma2 = 1;
    % Step 1.2: Compute true sdf
        f_true = [1/8 2/8 3/8]; % vector of frequencies
        S_true = S_AR(f_true,phis,sigma2);
    % Step 1.3: Set simulation parameters
        N = 16 ; % length of the simulation
        N_sim = 10000; % number of simulations

% STEP 2: SIMULATION
    % Step 2.1: Initialize matrices to store values
        bias_p = zeros(1,3); % empirical bias of periodogram
        bias_d = zeros(1,3); % empirical bias of direct
        X = zeros(N_sim,N); % AR(2) process
        S1_p = zeros(N_sim,N); % periodogram
        S2_d = zeros(N_sim,N); % direct
    % Step 2.2: Loop
        for i = 1:N_sim % simulate N_sim times
            X(i,:) = AR2_sim(phis,sigma2,N); % use our AR2_sim function to generate an AR(2) process
            S1_p(i,:) = periodogram(X(i,:));
            S2_d(i,:)= direct(X(i,:));
        end

% STEP 3: STORE VALUES FOR EACH FREQUENCY
% take out the values at freq 1/8, 2/8, 3/8, when N=16 on [0,1/2),
% we have the points 0, 1/16, 2/16=1/8, 3/16, 4/16=2/8, 5/16,
% 6/16=3/8, etc so pick the 3rd, 5th and 7th value of the vector
    p_1 = [S1_p(:,3),S1_p(:,5),S1_p(:,7)];
    d_1 = [S2_d(:,3),S2_d(:,5),S2_d(:,7)];

% PART B. Compute empirical bias
    bias_p = mean(p_1) - S_true;
    bias_d = mean(d_1) - S_true;

```

```
% PART C. Repeat steps A & B for N= [32 64 128 256 512 1024 2048 4096]

% STEP 1: SET PARAMETERS
N = [32 64 128 256 512 1024 2048 4096];
% for 16: 3rd (2^1 + 1), 5th (2^2 + 1) and 7th (3*2^1 +1) position - steps of 2^1
% for 32: 5th (2^2 + 1), 9th (2^3 + 1) and 13th (3*2^2 +1)position - steps of 2^2
% for 64: 9th (2^3 + 1), 17th (2^4 + 1) and 25th (3*2^3 +1)position - steps of 2^3
pow_1 = 2^2; % pick freq 1/8 for value of N=32,
pow_2 = 2^3; % pick freq 2/8 for value of N=32
pow_3 = 3*2^2; % pick freq 3/8 for value of N=32,

% STEP 2: LOOP OVER LENGTH OF N
for k = 1:length(N)
    % Step 2.1: Initialize matrices to store values
        X = zeros(N_sim,N(k));
        S1_p = zeros(N_sim,N(k));
        S2_d = zeros(N_sim,N(k));
    % Step 2.2: Loop to simulate N_sim times
        for i = 1 : N_sim
            X(i,:) = AR2_sim(phis,sigma2,N(k)); %use our AR2_sim function to generate an AR(2) process
            S1_p(i,:) = periodogram(X(i,:));
            S2_d(i,:) = direct(X(i,:));
        end
    % Step 2.3: Store values for each frequencies
        p = [p_1, S1_p(:,pow_1+1), S1_p(:,pow_2+1), S1_p(:,pow_3+1)];
        d = [d_1, S2_d(:,pow_1+1), S2_d(:,pow_2+1), S2_d(:,pow_3+1)];

  % Step 2.4: compute empirical bias (each column is a frequency)
        bias_p = [bias_p; mean(p(:,end-2:end)) - S_true];
        bias_d = [bias_d; mean(d(:,end-2:end)) - S_true];

  % STEP 2.5: Update the place at which we compute the frequency for the
  % value of N, the steps increase by a power of 2 as N is only powers of 2
        pow_1 = pow_1*2;
        pow_2 = pow_2*2;
        pow_3 = pow_3*2;
end

% PART D. Plot to compare the two spectral esimators for different values of N
% STEP 1: INITIALIZATION
N_plot = [16,N]; % concetenate vector N

% STEP 2: PLOT
figure
    % Step 2.1: Plot bias at frequency 1/8
        subplot(1,3,1)
        semilogx(N_plot,bias_p(:,1)) %log axis as suggested per the notes
        hold on
        semilogx(N_plot,bias_d(:,1))
        xlabel('log(N)')
        legend({'Periodogram','Direct Spectral estimate'},'Location','southwest')
        title('f=1/8')
    % Step 2.2: Plot bias at frequency 2/8
        subplot(1,3,2)
        semilogx(N_plot,bias_p(:,2))
        hold on
        semilogx(N_plot,bias_d(:,2))
        xlabel('log(N)')
        legend({'Periodogram','Direct Spectral estimate'},'Location','northeast')
        title('f=2/8')
    % Step 2.2: Plot bias at frequency 3/8
        subplot(1,3,3)
        semilogx(N_plot,bias_p(:,3))
        hold on
        semilogx(N_plot,bias_d(:,3))
        xlabel('log(N')
        legend({'Periodogram','Direct Spectral estimate'},'Location','northeast')
```

```
        title('f=3/8')
```

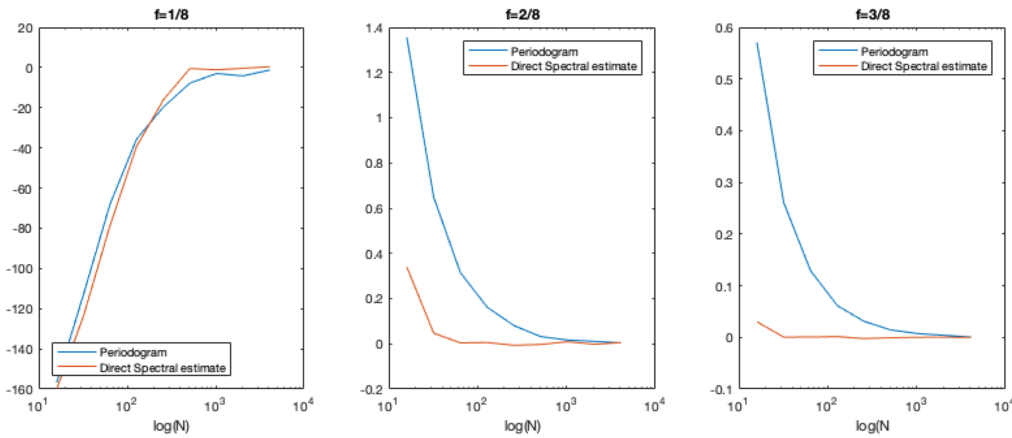


Figure 1: Empirical bias of periodogram and direct spectral estimate at 3 different frequencies

Part c

Interpretation of results:: First when we look at the plots of the bias between the periodogram and the direct spectral estimate, we see a clear distinction between the behavior at the frequency 1/8 and the frequencies 2/8 and 3/8, see Figure 1. While the latter decrease bias as we increase the length of each 10000 simulations to tend to zero, the former has increasing bias until it reaches approximately zero for very significant lengths of the realizations. We want to understand why such a behavior occurs. Now if we plot the true spectral density function, see Figure 2. Clearly it has spikes at frequencies around ± 0.12 and is very close to zero else. Note that $1/8 = 0.125$ which matches with our high bias shown before. As the true sdf spikes at around 1/8, the bias of both estimators is very high but tends to zero as the length of the realizations N go to infinity. Similarly at 2/8 and 3/8, the sdf is near zero and so the bias decreases as N goes to infinity.

```
% Plot of true sdf
plot([-0.5:0.001:0.5],S_AR([-0.5:0.001:0.5], phis, sigma2));
xlabel('frequency')
legend('True SDF','Location','southwest')
title('Plot of true spectral density function')
```

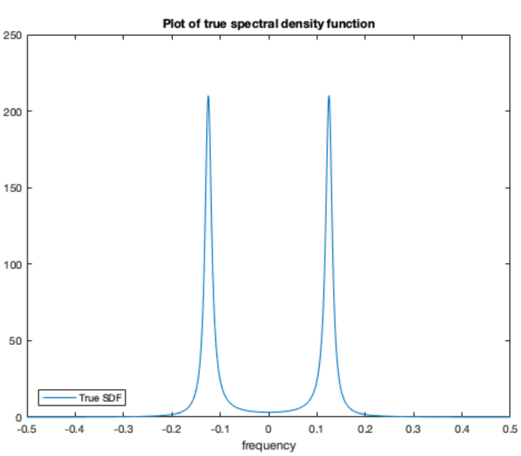


Figure 2: True spectral density function

Question 3

Part a

In this question, we plot the periodogram and the direct spectral estimate (as defined in Q2) for our given time series (number 226) over frequency $[-\frac{1}{2}, \frac{1}{2}]$. In order to do this, we need to use the in-build Matlab function *fftshift* which allows to shift our frequency range. See Figure 3.

```
% Initialize values
X_3 = table2array(TimeSeriesQ3); % transform time series to array
X_3_t = transpose(X_3); % to get a line vector of my time series
N = length(X_3);
f = -1/2 + [0:N-1]/N; % fourier frequencies

% Plot Direct Spectral Estimate
subplot(2,1,1)
[dr,~] = direct(X_3_t);
dr_shift = fftshift(dr); % shift values of direct
plot(f,dr_shift)
xlabel('frequency')
legend('direct','Location','southwest')
title('Direct spectral estimate')

% Plot Periodogram
subplot(2,1,2)
[pd,~] = periodogram(X_3_t);
pd_shift = fftshift(pd); % shift values of periodogram
plot(f,pd_shift)
xlabel('frequency')
legend('periodogram','Location','southwest')
title('Periodogram estimate')
```

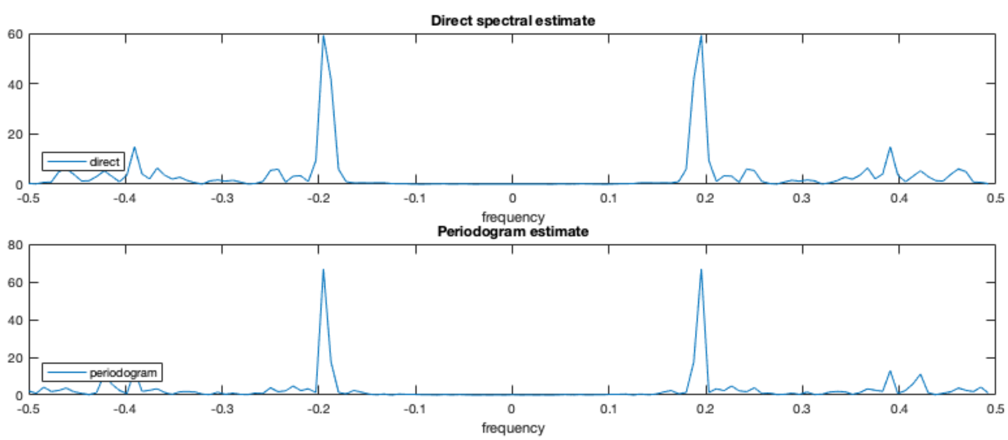


Figure 3: Periodogram and Direct Spectral Estimate using the Hanning taper

Part b

Yule-Walker method:

The goal of the function *YW(X,p)* is to fit an AR(p) model using the untapered Yule-Walker method. First define the autocovariance sequence at a lag of k: $s_k = \sum_{j=1}^{p} \phi_{j,p} s_{k-j}$ which in matrix notation is $\gamma_p = \Gamma_p \phi_p$ where $\gamma_p = [s_1, s_2, ..., s_p]^T$ and $\phi_p = [\phi_{1,p}, \phi_{2,p}, ..., \phi_{p,p}]^T$ and $\Gamma_p = \begin{bmatrix} s_0 & s_1 & ... & s_{p-1} \\ s_1 & s_0 & ... & s_{p-2} \\ \vdots & \vdots & ... & \vdots \\ s_{p-1} & s_{p-2} & ... & s_0 \end{bmatrix}$ which is known as the symmetric Toeplitz matrix. We don't know directly the autocovariance at the lag so we use the following estimates:

$$\hat{\phi}_p = \hat{\Gamma}_p^{-1}\hat{\gamma}_p$$

and

$$\hat{\sigma}_\epsilon^2 = \hat{s}_0 = \sum_{j=1}^{p} \hat{\phi}_{j,p}\hat{s}_j$$

In order to implement the Yule-Walker method as follows:

```
function[phi_YW, sigma2_YW] = YW(X,p)
% STEP 1: SET VALUES
s = acvs_hat(X,0:p);
sum = 0;
gamma = s(2:p+1);
T = toeplitz(s(1:p));
phi_YW = T^(-1)*transpose(gamma);

% STEP 2: LOOP
for k = 1:p
    sum = sum + phi_YW(k)*s(k+1);
end

% STEP 3: CONCLUSION
sigma2_YW = s(1) - sum;
```

Forward Least Squares method:
The goal of the function *LS(X,p)* is to fit an AR(p) model using the untapered forward Least Squares method. Let's describe it succintly: with our data $X_1, X_2, ..., X_N$ defined

$$X = F\phi + \epsilon$$

where,

$$F = \begin{bmatrix} X_p & X_{p-1} & ... & X_1 \\ X_{p+1} & X_p & ... & X_2 \\ \vdots & \vdots & ... & \vdots \\ X_{N-1} & X_{N-2} & ... & X_{N-p} \end{bmatrix}$$

and,

$$X = \begin{bmatrix} X_p \\ X_{p+2} \\ \vdots \\ X_N \end{bmatrix}; \phi = \begin{bmatrix} \phi_{1,p} \\ \phi_{2,p} \\ \vdots \\ \phi_{p,p} \end{bmatrix}; \epsilon = \begin{bmatrix} \epsilon_{p+1} \\ \epsilon_{p+2} \\ \vdots \\ \epsilon_N \end{bmatrix}$$

. Hence want to compute the estimates as follows:

$$\hat{\phi} = (F^T F)^{-1} F^T X$$

and,

$$\hat{\sigma}^2 = \frac{(X - F\hat{\phi})^T (X - F\hat{\phi})}{N - 2p}$$

```
function [phi_LS,sigma2_LS] = LS(X,p)
% STEP 1: SET VALUES
N = length(X);
F = zeros(N-p,p);
X_b = X((p+1):end);

% STEP 2: LOOP
for k = 1:N-p
    F(k,:) = fliplr(X(k:p+k-1));
end
```

```
% STEP 3: CONCLUSION
phi_LS = (transpose(F)*F)\transpose(F)*X_b;
sigma2_LS = (norm(X_b-F*phi_LS).^2) / (N-2*p);
```

Maximum Likelihood method:
The goal of the *MLE(X,p)* is to fit an AR(p) model using the approximate maximum likelihood method. Similar to the forward least squares method, we have the following estimates for the maximum likelihood method [1]:

$$\hat{\phi} = (F^T F)^{-1} F^T X$$

and,

$$\hat{\sigma}^2 = \frac{(X - F\hat{\phi})^T (X - F\hat{\phi})}{N - p}$$

where we note the only difference is in the denominator of the variance.

```
function [phi_MLE,sigma2_MLE] = MLE(X,p)
% STEP 1: SET VALUES
N=length(X);
F=zeros(N-p,p);
X_F = X((p+1):end);

% STEP 2: LOOP
for k = 1:(N-p)
    F(k,:) = fliplr(X(k:p+k-1));
end

% STEP 3: CONCLUSION
phi_MLE = (transpose(F)*F)\transpose(F)*X_F;
sigma2_MLE = (norm(X_F-F*phi_MLE).^2)/ (N-p);
```

Part c

Let's compute the Akaike Information Criterion for each $p = 1, 2, ..., 20$ for each method defined as:

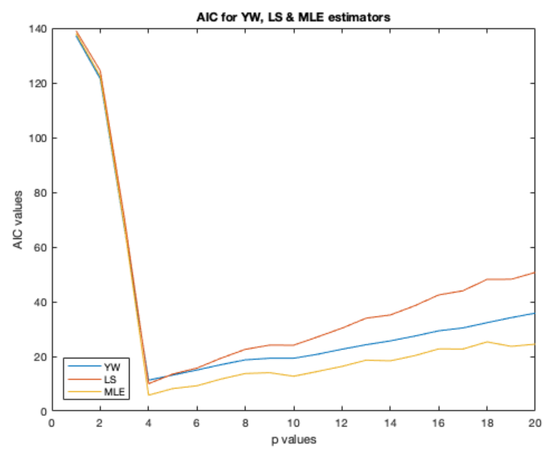
$$AIC = 2p + N \ln(\hat{\sigma}_\epsilon^2)$$

```
% STEP 1: SET VALUES
N=length(X_3);
p = linspace(1, 20, 20);
AIC_YW = zeros([20 1]);
AIC_LS = zeros([20 1]);
AIC_MLE = zeros([20 1]);

% STEP 2: COMPUTE AIC FOR EACH METHOD
    % STEP 2.1: Yule-Walker Method
for k=linspace(1, 20, 20)
[phi_YW, sigma2_YW] = YW(X_3,k);
AIC_YW(k) = 2*k + N*log(sigma2_YW);
end

    % STEP 2.2: Forward Least Squares Method
for k=linspace(1, 20, 20)
[phi_LS, sigma2_LS] = forwardLS(X_3,k);
AIC_LS(k) = 2*k + N*log(sigma2_LS);
end

    % STEP 2.3: Maximum Likelihood Method
for k=linspace(1, 20, 20)
[phi_MLE, sigma2_MLE] = MLE(X_3,k);
AIC_MLE(k) = 2*k + N*log(sigma2_MLE);
end

% STEP 3: DISPLAY RESULTS
```

```
p=transpose(p);
    % STEP 3.1: Create table of AIC values
        T = table(p,AIC_YW,AIC_LS,AIC_MLE);
        figure(1);
        uitable('Data',T{:,:},'ColumnName',T.Properties.VariableNames,...
        'RowName',T.Properties.RowNames,'Units', 'Normalized', 'Position',[0, 0, 1, 1]);
    % STEP 3.1: Create plot of AIC values against p
        figure(2);
        plot(p, AIC_YW) % plot Yule Walker AIC values
        hold on
        plot(p, AIC_LS) % plot forward Least Squares AIC values
        hold on
        plot(p, AIC_MLE) % plot maximum likelihood AIC values
        xlabel('p values')
        ylabel('AIC values')
        box on
        legend({'YW','LS', 'MLE'},'Location','southwest')
        title('AIC for YW, LS & MLE estimators')
```

p	AIC_YW	AIC_LS	AIC_MLE
1	137.1816	139.0612	138.0493
2	121.4959	124.4607	122.4126
3	66.9947	70.6352	67.5257
4	11.3343	10.0299	5.8328
5	13.1848	13.5505	8.2385
6	15.0306	15.7264	9.2712
7	16.9913	19.3655	11.7377
8	18.7480	22.5744	13.7433
9	19.3228	24.1255	14.0592
10	19.3277	24.0929	12.7581
11	20.8173	27.1681	14.5301
12	22.6224	30.3303	16.3528
13	24.2841	33.9874	18.6326
14	25.6705	35.1583	18.3867
15	27.4175	38.4947	20.2649
16	29.3611	42.4822	22.7509
17	30.4172	43.9872	22.7091
18	32.3294	48.1765	25.3039
19	34.1863	48.1937	23.6768
20	35.8286	50.7129	24.4992

(a) Table of AIC values



(b) Plot of AIC values

Figure 4: Values of AIC for the 3 studied methods

Part d

As seen on the table and on the graph, we want to take the smallest value of AIC for each methods to fit best the model [1]. Here we can see in Figure 4 that for p=4 we have the lowest AIC in each method. See the associated parameters below for each method in Figure 5:

```
% Create table of estimated parameter values for each method
param = ["phi(1,4)"; "phi(2,4)"; "phi(3,4)"; "phi(4,4)"; "sigma2"];
YW = [phi_YW_p ; sigma2_YW_p];
LS = [phi_LS_p ; sigma2_LS_p];
MLE = [phi_MLE_p ; sigma2_MLE_p];
T = table(param,YW,LS,MLE);
```

param	YW	LS	MLE
"phi(1,4)"	-0.7086	-0.7346	-0.7346
"phi(2,4)"	-0.7040	-0.7141	-0.7141
"phi(3,4)"	-0.8075	-0.8194	-0.8194
"phi(4,4)"	-0.6022	-0.6248	-0.6248
"sigma2"	1.0264	1.0160	0.9832

Figure 5: Estimated parameter values for each method

Part e

Here we plot the spectral density function for each method (see Figure 6).

```
% STEP 1: SET VALUES
p_e=4; % AR(4) has smallest AIC
[phi_YW_p, sigma2_YW_p] = YW(X_3,p_e);
[phi_LS_p, sigma2_LS_p] = forwardLS(X_3,p_e);
[phi_MLE_p, sigma2_MLE_p] = MLE(X_3,p_e);

%STEP 2: PLOT SPECTRAL DENSITY FUNCTION FOR EACH SELECTED MODELS
x_h = [-0.5:10^(-4):0.5]; % frequencies to plot on
plot(x_h ,S_AR(x_h, phi_YW_p, sigma2_YW_p));
hold on
plot(x_h,S_AR(x_h, phi_LS_p, sigma2_LS_p));
hold on
plot(x_h,S_AR(x_h, phi_MLE_p, sigma2_MLE_p));
box on
xlabel('frequency')
ylabel('method')
legend({'YW','LS', 'MLE'},'Location','southwest')
title('YW, LS & MLE spectral estimates')
```

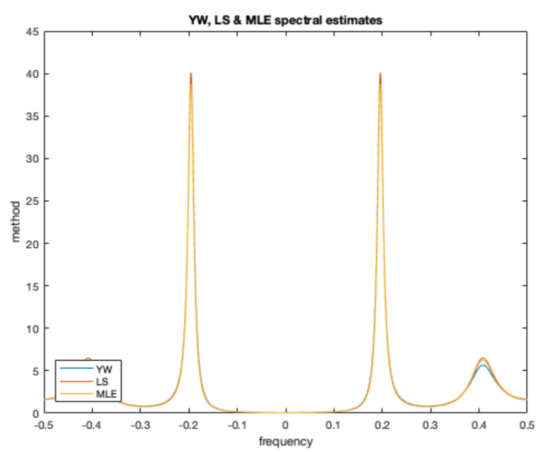


Figure 6: Spectral density function associated with the 3 studied methods

Question 4

In this question we will explore forecasting of values AR(p) process. Say the prediction step we want is called l. For l=1, we have:

$$X_t(1) = \phi_{1,p}X_t + ... + \phi_{p,p}X_{t-p+1} + e$$

Note that we set the future innovation term e to be zero, as per the hint. We can generalise to a l-step prediction step with $1 \leq l \leq p$:

$$X_t(l) = \phi_{1,p}X_t(l-1) + \phi_{2,p}X_t(l) + ... + \phi_{l-1,p}X_t(1) + \phi_{l,p}X_t + ... + \phi_{p,p}X_{t-p+l} + 0$$

As we base ourselves from the best estimate as found in Q3d, we choose p=4 and so we only need to compute the following:

$$\begin{aligned}
X_t(1) &= \phi_{1,4}X_t + \phi_{2,4}X_{t-1} + \phi_{3,4}X_{t-2} + \phi_{4,4}X_{t-3} \\
X_t(2) &= \phi_{1,4}X_t(1) + \phi_{2,4}X_t + \phi_{3,4}X_{t-1} + \phi_{4,4}X_{t-2} \\
&= \phi_{1,4}(\phi_{1,4}X_t + \phi_{2,4}X_{t-1} + \phi_{3,4}X_{t-2} + \phi_{4,4}X_{t-3}) + \phi_{2,4}X_t + \phi_{3,4}X_{t-1} + \phi_{4,4}X_{t-2} \\
&= (\phi_{1,4}^2 + \phi_{2,4})X_t + (\phi_{1,4}\phi_{2,4} + \phi_{3,4})X_{t-1} + (\phi_{1,4}\phi_{3,4} + \phi_{4,4})X_{t-2} + (\phi_{1,4}\phi_{4,4})X_{t-3} \\
X_t(3) &= \phi_{1,4}X_t(2) + \phi_{2,4}X_t(1) + \phi_{3,4}X_t + \phi_{4,4}X_{t-1} \\
&= \phi_{1,4}(\phi_{1,4}X_t(1) + \phi_{2,4}X_t + \phi_{3,4}X_{t-1} + \phi_{4,4}X_{t-2}) + \phi_{2,4}X_t(1) + \phi_{3,4}X_t + \phi_{4,4}X_{t-1} \\
&= (\phi_{1,4}^2 + \phi_{2,4})X_t(1) + (\phi_{1,4}\phi_{2,4} + \phi_{3,4})X_t + (\phi_{1,4}\phi_{3,4} + \phi_{4,4})X_{t-1} + (\phi_{1,4}\phi_{4,4})X_{t-2} \\
X_t(4) &= \phi_{1,4}X_t(3) + \phi_{2,4}X_t(2) + \phi_{3,4}X_t(1) + \phi_{4,4}X_t \\
&= (\phi_{1,4}^2 + \phi_{2,4})X_t(2) + (\phi_{1,4}\phi_{2,4} + \phi_{3,4})X_t(1) + (\phi_{1,4}\phi_{3,4} + \phi_{4,4})X_t + (\phi_{1,4}\phi_{4,4})X_{t-1}
\end{aligned}$$

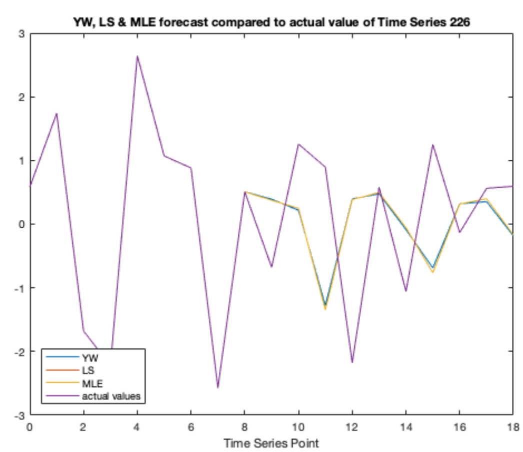
From lecture notes [1], we know that for general AR(p) processes, $X_t(l)$ depends only on the last p observed values of X_t. So for example for our time series, for X_{119} forecast only depends on $X_{115}, X_{116}, X_{117}, X_{118}$.

```
% STEP 1: SET VALUES
mat_param= [phi_YW_p, phi_LS_p, phi_MLE_p]; % make matrix of parameters phi

% STEP 2: YW METHOD FOR FORECAST
    param_YW = mat_param(:,1); % pick the column of parameters for YW method
    phi_fut_YW = [param_YW(1)^2+ param_YW(2); param_YW(1)*param_YW(2)+param_YW(3); param_YW
    (1)*param_YW(3)+param_YW(4); param_YW(1)*param_YW(4)];
    ts_YW = X_3_t(115:118); % as AR(4) only dependant on last 4 observed values ie X 115 to
    X 118
    next_YW = ts_YW*param_YW; % compute X 119
    ts_YW = [ts_YW, next_YW]; % update TS
    for k=1:9 % Loop to compute next 9 forecast values
        next_YWk = flip(ts_YW(1+k:end)); % re-order coefficients
        xt_YW = next_YWk*phi_fut_YW; % compute next value
        ts_YW = [ts_YW, xt_YW]; % update our time series with forecast
    end

% STEP 3: LS METHOD FOR FORECAST
    param_LS = mat_param(:,2); % pick the column of parameters for LS method
    phi_fut_LS = [param_LS(1)^2+ param_LS(2); param_LS(1)*param_LS(2)+param_LS(3); param_LS
    (1)*param_LS(3)+param_LS(4); param_LS(1)*param_LS(4)];
    ts_LS = X_3_t(115:118); % as AR(4) only dependant on last 4 observed values ie X 115 to
    X 118
    next_LS = ts_LS*param_LS; % compute X 119
    ts_LS = [ts_LS, next_LS]; % update TS
    for k=1:9 % Loop to compute next 9 forecast values
        next_LSk = flip(ts_LS(1+k:end)); % re-order coefficients
        xt_LS = next_LSk*phi_fut_LS; % compute next value
        ts_LS = [ts_LS, xt_LS]; % update our time series with forecast
    end

% STEP 3: MLE METHOD FOR FORECAST
    param_MLE = mat_param(:,3); % pick the column of parameters for MLE method
    phi_fut_MLE = [param_MLE(1)^2+ param_MLE(2); param_MLE(1)*param_MLE(2)+param_MLE(3);
    param_MLE(1)*param_MLE(3)+param_MLE(4); param_MLE(1)*param_MLE(4)];
    ts_MLE = X_3_t(115:118); % as AR(4) only dependant on last 4 observed values ie X 115 to
     X 118
    next_MLE = ts_MLE*param_MLE; % compute X 119
    ts_MLE = [ts_MLE, next_MLE]; % update TS
    for k=1:9 % Loop to compute next 9 forecast values
        next_MLEk = flip(ts_MLE(1+k:end)); % re-order coefficients
```

```
        xt_MLE = next_MLEk*phi_fut_MLE; % compute next value
        ts_MLE = [ts_MLE, xt_MLE]; % update our time series with forecast
    end

% STEP 4: PLOT AND TABLE
    ts_YW = [X_3_t(110:114), ts_YW]; % update our time series with previous known values
    ts_LS = [X_3_t(110:114), ts_LS];
    ts_MLE = [X_3_t(110:114), ts_MLE];
    % STEP 4.1: Table
    xvals = transpose(110:128);
    ts_YW_t = transpose(ts_YW);
    ts_LS_t = transpose(ts_LS);
    ts_MLE_t = transpose(ts_MLE);
    actual = X_3(110:128);
    T = table(xvals, actual,ts_YW_t, ts_LS_t, ts_MLE_t);
    figure(1);
    uitable('Data',T{:,:},'ColumnName',T.Properties.VariableNames,...
        'RowName',T.Properties.RowNames,'Units', 'Normalized', 'Position',[0, 0, 1, 1]);
    % STEP 4.2: Plot
    figure(2);
    m = 0:18; % numbers of values to plot
    plot(m,ts_YW);
    hold on
    plot(m,ts_LS);
    hold on
    plot(m,ts_MLE);
    hold on
    plot(m,X_3_t(110:128));
    xlabel('Time Series Point')
    legend({'YW','LS', 'MLE', 'actual values'},'Location','southwest')
    title('YW, LS & MLE forecast compared to actual value of Time Series 226')
```

xvals	actual	ts_YW_t	ts_LS_t	ts_MLE_t
110	0.5828	0.5828	0.5828	0.5828
111	1.7396	1.7396	1.7396	1.7396
112	-1.6773	-1.6773	-1.6773	-1.6773
113	-2.2231	-2.2231	-2.2231	-2.2231
114	2.6429	2.6429	2.6429	2.6429
115	1.0707	1.0707	1.0707	1.0707
116	0.8811	0.8811	0.8811	0.8811
117	-2.5726	-2.5726	-2.5726	-2.5726
118	0.5077	0.5077	0.5077	0.5077
119	-0.6767	0.3925	0.3749	0.3749
120	1.2577	0.2174	0.2483	0.2483
121	0.8945	-1.2781	-1.3463	-1.3463
122	-2.1771	0.3959	0.3861	0.3861
123	0.5819	0.4754	0.4959	0.4959
124	-1.0576	-0.0870	-0.0555	-0.0555
125	1.2518	-0.6864	-0.7633	-0.7633
126	-0.1326	0.3201	0.3154	0.3154
127	0.5624	0.3527	0.3989	0.3989
128	0.5946	-0.1865	-0.1705	-0.1705

(a) Table of forecasted values



(b) Plot of forecasted Time Series

Figure 7: Forecasted Time Series values for the 3 studied methods

Interpretation of results: On this Figure 7, we can see our forecasting plot: as our forward least squares and our maximum likelihood are very similar, their forecasts are not distinguishable. However they follow the same trend as the Yule-Walker method, and in general the actual values of our time series, as per the table of values.

Reference

[1] E. Cohen. Time Series, Lecture Notes, Imperial College London, 2020. 2020.