ELEC9782 Assignment 1

z5036602 - Zhengyue LIU

Semester 2 2019

I, Zhengyue LIU(student number z5036602), declare that the following assignment is my own work and that I have read and understood the University Rules in respect of Student Academic Misconduct.

Question 1

Problem (a)

(i)

Apply FIR filter on input to get output s_t, then perform the Z transform(back-shift operator) on equations will give us following steps:

$$\begin{aligned}
s_t &= (h * u)_t \\
&= h_0 u_t + h_1 u_{t-1} + h_2 u_{t-2} \\
\mathcal{Z}\{s_t\} &= \mathcal{Z}\{h_0 u_t + h_1 u_{t-1} + h_2 u_{t-2}\} \\
&= (h_0 + h_1 z^{-1} + h_2 z^{-2}) u_z \\
&= A(1 - 2\alpha z^{-1} + z^{-2}) u_z \qquad (1)
\end{aligned}$$

Then we apply Z transform on input u_t which is an AR sequence, we will get following steps:

$$\begin{aligned}
\mathcal{Z}\{u_t\} &= \mathcal{Z}\{\phi u_{t-1} + \eta_t\} \\
u_z &= \phi z^{-1} u_z + \eta_z \\
u_z &= \frac{\eta_z}{1 - \phi z^{-1}} \qquad (2)
\end{aligned}$$

Then we substitute the u_z in eqn(2) back into eqn(1) to get:

$$\begin{aligned}
s_z &= A(1 - 2\alpha z^{-1} + z^{-2}) \frac{\eta_z}{1 - \phi z^{-1}} \\
(1 - \phi z^{-1}) s_z &= A\eta_z (1 - 2\alpha z^{-1} + z^{-2})
\end{aligned}$$

Finally, we transform the equation back to time domain

$$\begin{aligned}
s_t - \phi s_{t-1} &= A\eta_t - 2A\alpha\eta_{t-1} + A\eta_{t-2} \qquad (3) \\
s_t &= A\eta_t - 2A\alpha\eta_{t-1} + A\eta_{t-2} + \phi s_{t-1}
\end{aligned}$$

Eqn(3) is the description of an ARMA process. LHS is the AR component, RHS is the MA component. The s_t is an ARMA process.
The parameters of the model are $[A, -2A\alpha, A, \phi]$

(ii)

$$s_z = A(1 - 2\alpha z^{-1} + z^{-2}) \frac{\eta_z}{1 - \phi z^{-1}}$$

Stability / stationarity requires $|\phi| < 1$ (pole of the transfer function must lie within unit circle)

For second question, caculate the mean first.

$$\begin{aligned}
\mathbb{E}(s_t) &= A\mathbb{E}(\eta_t) - 2A\alpha\mathbb{E}(\eta_{t-1}) + A\mathbb{E}(\eta_{t-2}) + \phi\mathbb{E}(s_{t-1}) \\
&= 0 + 0 + 0 + \phi\mathbb{E}(s_t)
\end{aligned}$$

As, it is a stationary process, $|\phi| < 1$, hence $\mathbb{E}(s_t) = 0$

To calculate the σ_s^2, first we got:

$\mathbb{E}(u_t^2) = \gamma_0 = \frac{\sigma^2}{1-\phi^2}$
$\gamma_r = \phi\gamma_0$

$$
\begin{aligned}
\sigma_s^2 &= \mathbb{E}(A(\mu_t - 2\alpha\mu_{t-1} + \mu_{t-2}))^2 \\
&= A^2(\gamma_0 - 2\alpha\phi\gamma_0 + \phi^2\gamma_0 - 2\alpha\phi\gamma_0 + 4\alpha^2\gamma_0 - 2\alpha\phi\gamma_0 + \phi^2\gamma_0 - 2\alpha\phi\gamma_0 + \gamma_0^2) \\
&= A^2\gamma_0(2 + 4\alpha^2 - 8\phi\alpha + 2\phi^2) \\
&= 2A^2\gamma_0(1 + 2\alpha^2 - 4\phi\alpha + 1\phi^2) \\
&= \frac{2A^2\sigma^2}{1-\phi^2}(1 - 4\alpha\phi + \phi^2 + 2\alpha^2)
\end{aligned}
$$

Problem (b)

(i)

Apply Z transform (back-shift operator) on equation:

$$
\begin{aligned}
\mathcal{Z}\{Y_t\} &= \mathcal{Z}\{a + \phi Y_{t-3} + \epsilon_t\} \\
&= a + \phi z^{-3} Y_z + \epsilon_t \\
&= \frac{a + \epsilon_t}{1 - \phi z^{-3}}
\end{aligned}
$$

Stability / stationarity requires $|\phi| < 1$ (pole of the transfer function must lie within unit circle)

(ii)

$$
\begin{aligned}
\mu = \mathbb{E}(Y_t) &= \mathbb{E}(a + \phi Y_{t-3} + \epsilon_t) \\
&= \mathbb{E}(a) + \mathbb{E}(\phi Y_{t-3}) + \mathbb{E}(\epsilon_t) \\
&= a + \phi\mathbb{E}(Y_t) + 0 \qquad (E(Y_{t-3}) = E(Y_t) \text{ as it is stationary}) \\
&= \frac{a}{1-\phi}
\end{aligned}
$$

The acs can be represented by difference equation. Firstly, we calculate the acvs of the Y_t. The method here is not smart, I calculated lots of γ value and find the underlying principle.

$$
\begin{aligned}
\gamma_0 &= \mathbb{E}(Y_t, Y_t) - \mu^2 \\
&= \mathbb{E}(Y_t^2) - \mu^2 \\
&= \mathbb{E}((a + \phi Y_{t-3} + \epsilon_t)^2) - \mu^2 \\
&= \mathbb{E}(a^2) + \mathbb{E}(a\phi Y_{t-3}) + \mathbb{E}(\phi Y_{t-3} a) + \mathbb{E}(\phi^2 Y_{t-3}^2) + \mathbb{E}((\epsilon_t)^2) - \mu^2 \\
&= a^2 + 2a\phi\mu + \phi^2(\gamma_0 + \mu^2) + \sigma^2 - \mu^2 \\
&= a^2 + 2a\phi\mu + \phi^2\mu^2 + \phi^2\gamma_0 + \sigma^2 - \mu^2 \\
&= (a + \phi\mu)^2 + \phi^2\gamma_0 + \sigma_2 - \mu^2 \\
&= \frac{(a + \phi\mu)^2 + \sigma^2 - \mu^2}{1 - \phi^2}
\end{aligned}
$$

Proof for $a + \phi\mu = \mu$:

$$\mu = \frac{a}{1-\phi} \Rightarrow a + \phi\mu = a + \phi\frac{a}{1-\phi} = \frac{a(1-\phi)+\phi a}{1-\phi} = \frac{a - a\phi + \phi a}{1-\phi} = \frac{a}{1-\phi} = \mu$$

Hence

$$\gamma_0 = \frac{\alpha^2}{1-\phi^2}$$

$$\begin{aligned}
\gamma_1 &= \mathbb{E}(Y_t, Y_{t-1}) - \mu^2 \\
&= \mathbb{E}((a + \phi Y_{t-3} + \epsilon_t)(a + \phi Y_{t-4} + \epsilon_{t-1})) - \mu^2 \\
&= \mathbb{E}(a^2) + a\phi\mathbb{E}(Y_{t-4}) + a\phi\mathbb{E}(Y_{t-3}) + \phi^2(\gamma_1 + \mu^2) \\
&= a^2 + 2a\phi\mu + \phi^2(\gamma_1 + \mu^2) - \mu^2 \\
&= a^2 + 2a\phi\mu + \phi^2\mu^2 + \phi^2\gamma_1 - \mu^2 \\
&= \frac{(a+\phi\mu)^2 - \mu^2}{1-\phi^2} \\
&= 0 \\
\gamma_2 &= \mathbb{E}(Y_t, Y_{t-2}) - \mu^2 \\
&= \mathbb{E}((a + \phi Y_{t-3} + \epsilon_t)(Y_{t-2})) - \mu^2 \\
&= a\mu + \phi(\gamma_1 + \mu^2) - \mu^2 \\
&= a\mu + \phi\mu^2 - \mu^2 \\
&= a\mu + \mu^2(\phi - 1) \\
&= \frac{a^2}{1-\phi} + \frac{a^2(\phi-1)}{(1-\phi)^2} \\
&= \frac{a^2}{1-\phi} - \frac{a^2}{1-\phi} \\
&= 0 \\
\gamma_3 &= \mathbb{E}(Y_t, Y_{t-3}) - \mu^2 \\
&= \mathbb{E}((a + \phi Y_{t-3} + \epsilon_t)(Y_{t-3})) - \mu^2 \\
&= a\mu + \phi(\gamma_0 + \mu^2) - \mu^2 \\
&= a\mu + \phi\frac{\sigma^2}{1-\phi^2} + \phi\mu^2 - \mu^2 \\
&= a\mu + \frac{\phi\sigma^2}{1-\phi^2} - \mu^2(1-\phi) \\
&= \frac{a^2}{1-\phi} + \frac{\phi\sigma^2}{1-\phi^2} - \frac{a^2(1-\phi)}{(1-\phi)^2} \\
&= \frac{a^2}{1-\phi} + \frac{\phi\sigma^2}{1-\phi^2} - \frac{a^2}{1-\phi} \\
&= \frac{\phi\sigma^2}{1-\phi^2}
\end{aligned}$$

$$
\begin{aligned}
\gamma_r &= \mathbb{E}(Y_t, Y_{t+r}) - \mu^2 \\
&= \mathbb{E}(Y_t(a + \phi Y_{t+r-3} + \epsilon_{t+r})) - \mu^2 \\
&= a\mathbb{E}(Y_t) + \phi\mathbb{E}(Y_t, Y_{t+r-3}) + \mathbb{E}(Y_t, \epsilon_{t+r}) - \mu^2 \\
&= a\mu + \phi(\gamma_{r-3} + \mu^2) + \mathbb{E}(Y_t, \epsilon_{t+r}) - \mu^2 \\
&= a\mu + \phi(\gamma_{r-3} + \mu^2) - \mu^2 \\
&= a\mu + \phi\gamma_{r-3} - (1-\phi)\mu^2 \\
&= \frac{a^2}{1-\phi} + \phi\gamma_{r-3} - (1-\phi)\frac{a^2}{(1-\phi)^2} \\
&= \frac{a^2}{1-\phi} + \phi\gamma_{r-3} - \frac{a^2}{1-\phi} \\
&= \phi\gamma_{r-3} \qquad (\text{r} \geq 3\ )
\end{aligned}
$$

From above, we have:

$$
\begin{aligned}
&\gamma_0 = \frac{\alpha^2}{1-\phi^2} \\
&\gamma_1 = 0 \\
&\gamma_2 = 0 \\
&\gamma_r = \phi\gamma_{r-3} \qquad (\text{r} \geq 3)
\end{aligned}
$$

Hence:

$$
\begin{aligned}
&\gamma_r = \phi^{\frac{r}{3}}\gamma_0 \qquad (r = 3n, n = 0, 1, 2, ...) \\
&\gamma_r = 0 \qquad (r \neq 3n, n = 0, 1, 2, ...)
\end{aligned}
$$

Then, we calculate the acs by finding the ratio between acvs:

$$
\begin{aligned}
&\rho_r = \frac{\gamma_r}{\gamma_0} = \phi^{\frac{r}{3}} \qquad (r = 3n, n = 0, 1, 2, ...) \\
&\rho_r = 0 \qquad (r \neq 3n, n = 0, 1, 2, ...)
\end{aligned}
$$

Question 2)

Problem (a)

(i)

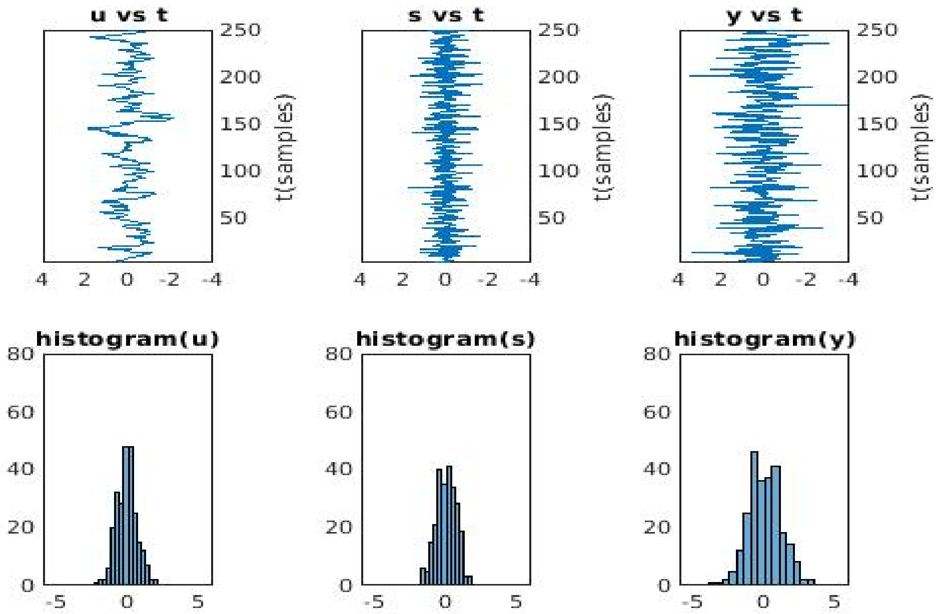

Figure 1: $\phi = 0.8 \quad \alpha = 1 \quad vsnr = 0.5$

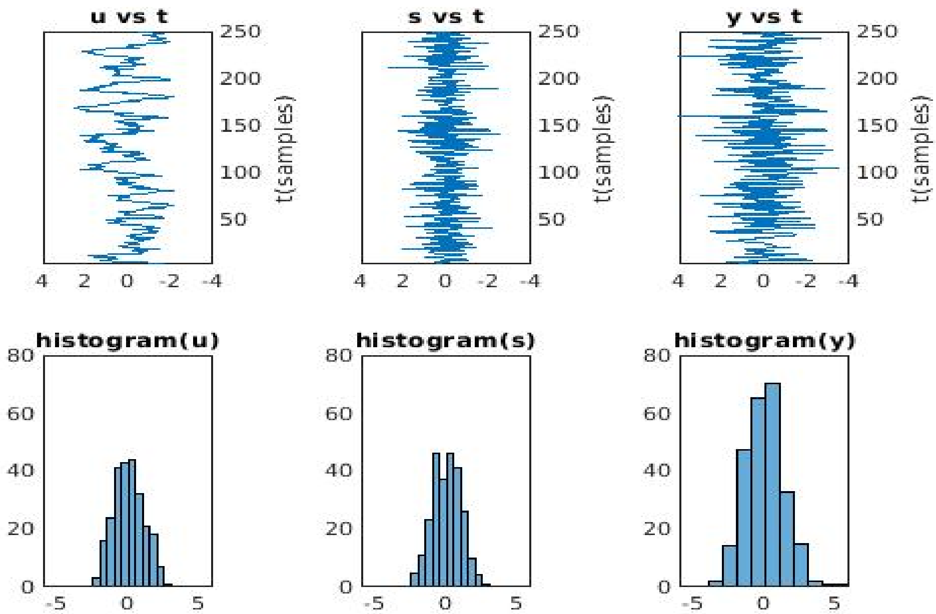

Figure 2: $\phi = 0.8 \quad \alpha = 1 \quad vsnr = 1$

Empirical var(u)	Empirical var(s)	Empirical var(y)
0.63	0.51	1.45
Theoretical var(u)	Theoretical var(s)	Theoretical var(y)
0.57	0.50	1.50

Table 1: $\phi = 0.8 \quad \alpha = 1 \quad vsnr = 0.5$

The empirical value is closed to theoretical value, signal y is much noisier than s and u

Empirical var(u)	Empirical var(s)	Empirical var(y)
0.81	0.91	2.07
Theoretical var(u)	Theoretical var(s)	Theoretical var(y)
1.13	1.00	2.02

Table 2: $\phi = 0.8 \quad \alpha = 1 \quad vsnr = 1$

The empirical value for s and y is closed to their theoretical value. However, the empirical u is bit far from its theoretical value.It might be caused by high snr which leads greater fluctuation for observation. In both tables, we can tell the var(y) is much larger than previous 2 which is caused by the added white noise component.

Matlab implementation

In this question, I used `output_sim.m`. The explanations were commented on the code. And I call the function on command line with different inputs.

(ii)

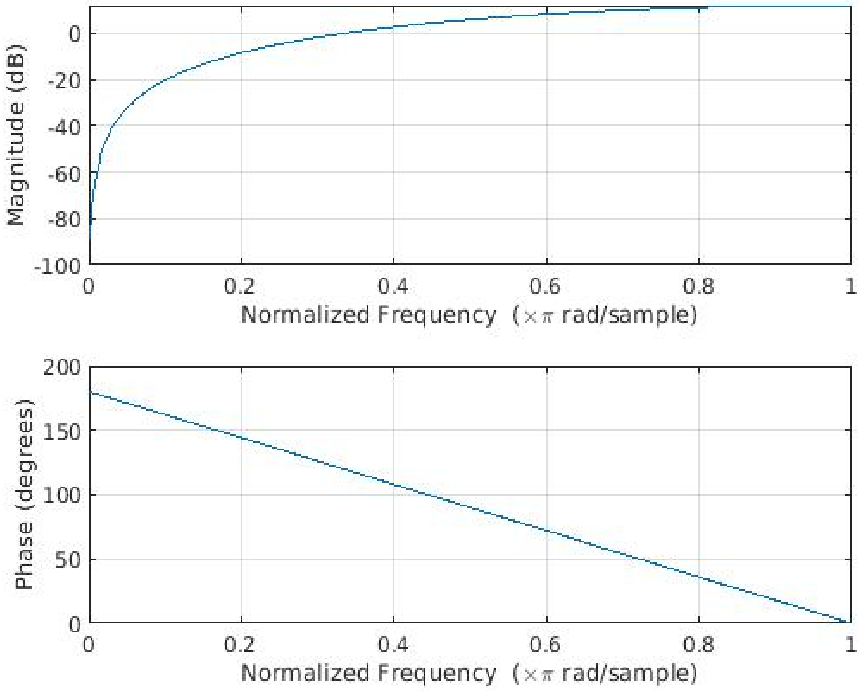

Figure 3: Bode plot of the FIR filter

Top pic of Figure 3 shows the bode plot of our FIR filter. It can be seen that it is a high-pass filter which heavily suppresses the dc and low frequency component, and add a little bit gain on high frequency(highest 0.9 dB). If we look at the output data plots, the dc components of s is lower than u. The s signal's fluctuations are strongly aligned to zero mean. It sort of looking like a "mean corrected version of u" with high-frequency noise amplified a little.

Matlab implementation

In this question, I used `output_sim.m`. On line 71 of the code, I used freqz() to plot the frequency response of the FIR filter system.

Problem (b)

(i)

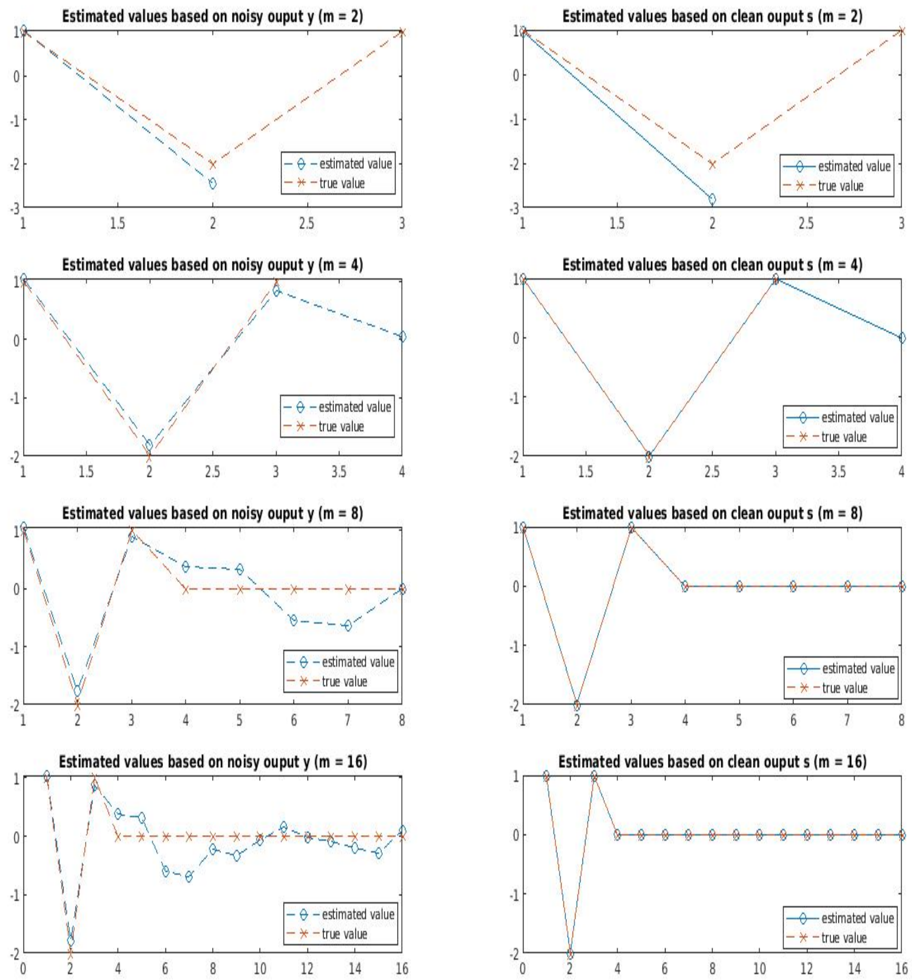

Figure 4: Estimated Impulse response and the true Impulse response

Here, I plotted the impulse response for models with order m = 2,4,8,16, which are estimated by the least square estimator. As the question did not clearly specified which input data to use for modelling, I plotted 8 graphs here. Left 4 graphs show the impulse response of the model estimated on noisy data y_t, and right 4 graphs show the impulse response of the model estimated on original convoluted data s_t. We can tell the clean data s_t gave us much better estimation. So if a signal has very low signal to noise ratio, it might not be a good idea to model the signal

The original impulse response of the model is [1,-2,1] with order 3. High order estimators are completely wasted, because the coefficients higher than order 3 are all estimated as zero. It could lead to over-fitting as well.

Matlab implementation

In this question, I used `Q2.m`, `LSE.m` and `output_sim.m`. Explanations are commented on code.

(ii)

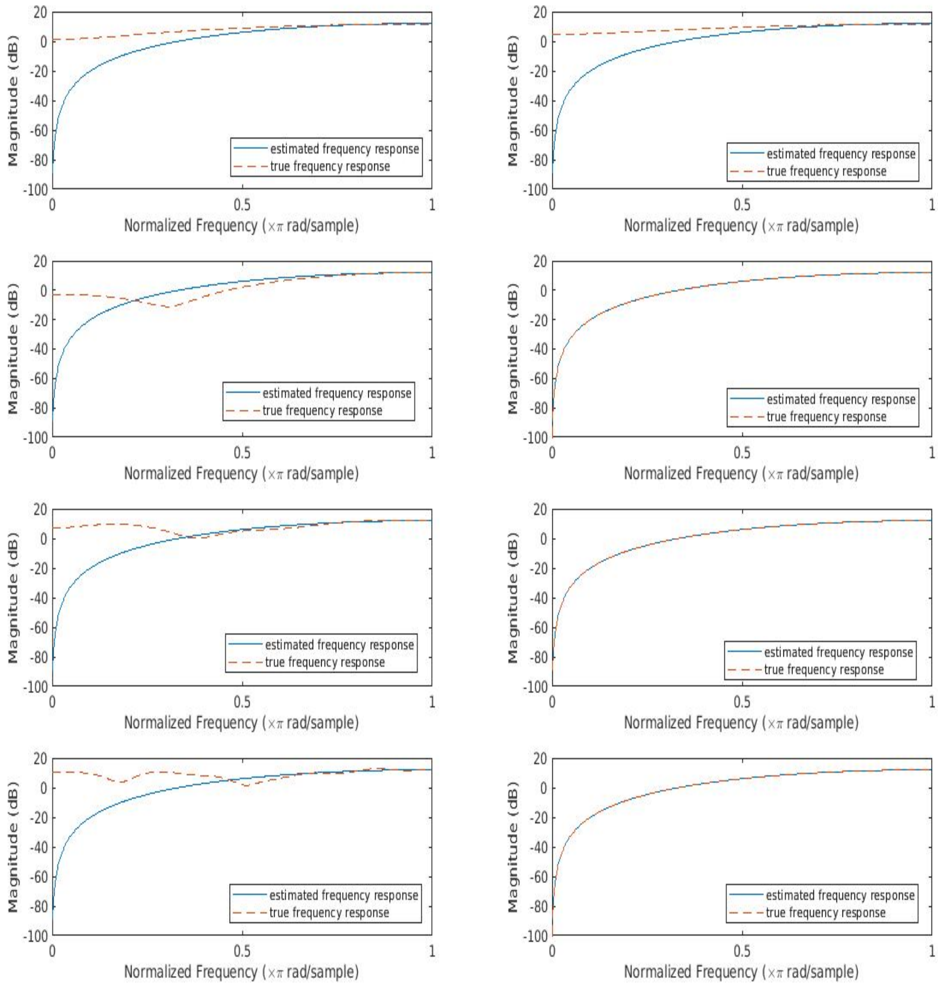

Figure 5: Estimated frequency response and the true frequency response

Same here, I plotted 8 graphs, Left 4 graphs show the frequency response of the model (m = 2,4,8,16) estimated on noisy data y_t, and right 4 graphs show the frequency response of the model estimated on original convoluted data s_t. We can tell the clean data s_t gave us much better estimation.

I also noticed that a small amount of estimation error on filter coefficients might lead to huge error on its estimated frequency response. Even though the estimated coefficients is not far away from its true value, its frequency response is very different from original system.

Matlab implementation

In this question, I used `Q2.m`, `LSE.m` and `output_sim.m`. Explanations are commented on code.

(iii)

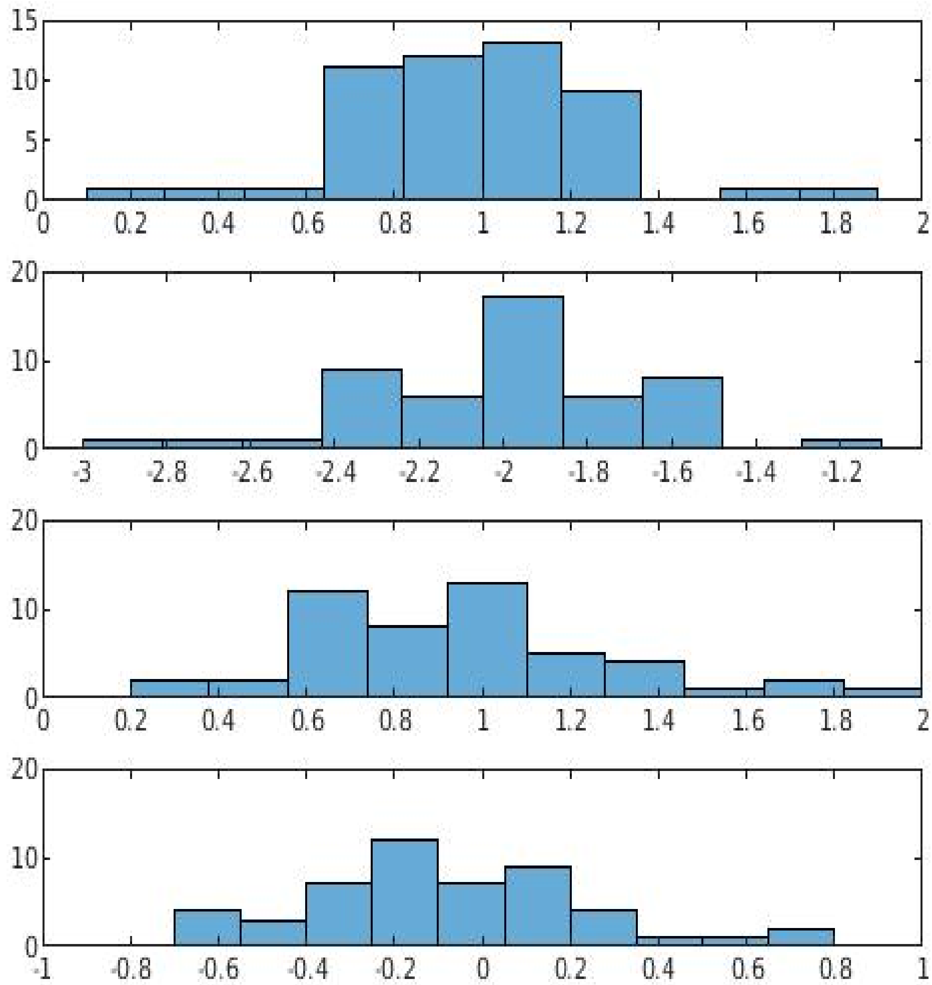

Figure 6: Histograms of each of the FIR parameters of model with order 4.

The plots from top to below corresponding to first impulse coefficients to fourth impulse coefficients. Here, I only plotted the histogram of coefficients which are estimated on noisy output data y.The true impulse response is [1,-2,1]. I will expect histogram show a normal distribution centered at [1,-2,1,0]. Closer the center to these values(less bias) and less deviation(variance) from these values will represent a more accurate estimator. The estimator here is not good. Because, the estimated values are sort of "everywhere" on histogram(large variance). On the first and last histogram, the distributions are not closely centered on 0(large bias).

In conclusion, such inaccuracy shows that our estimators are not robust!!

Matlab implementation

In this question, I used `FIR_hist.m`, `LSE.m` and `output_sim.m`. Explanations are commented on code.

Question 3

Problem (a)

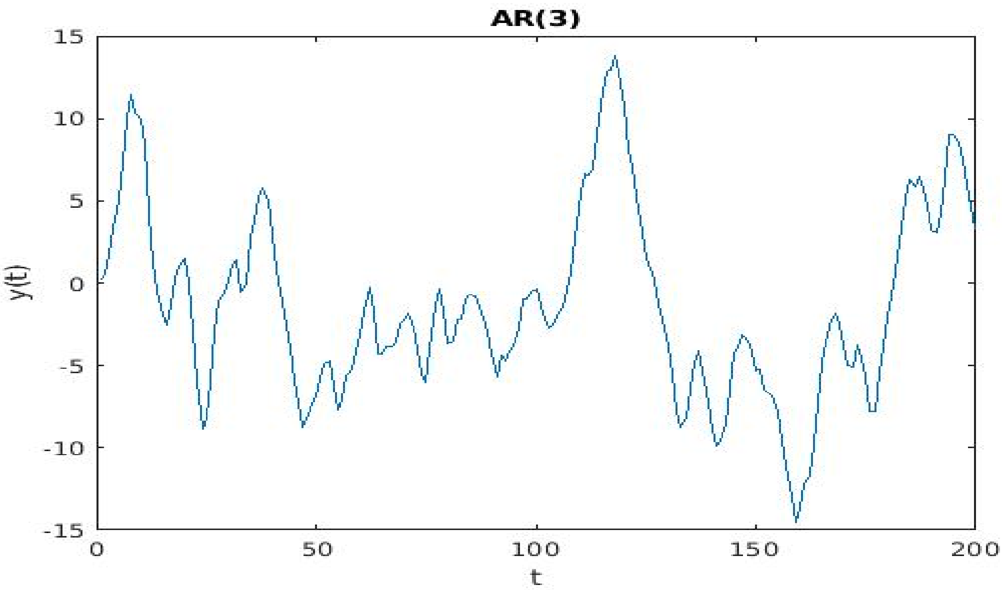

Figure 7: Simulation of AR(3) with $root_1 = 0.6$,$root_2 = 0.8$,$root_3 = 0.4$ $\gamma_0 = 31.9$. True parameters of the model are $[1.8, -1.04, 0.19]$

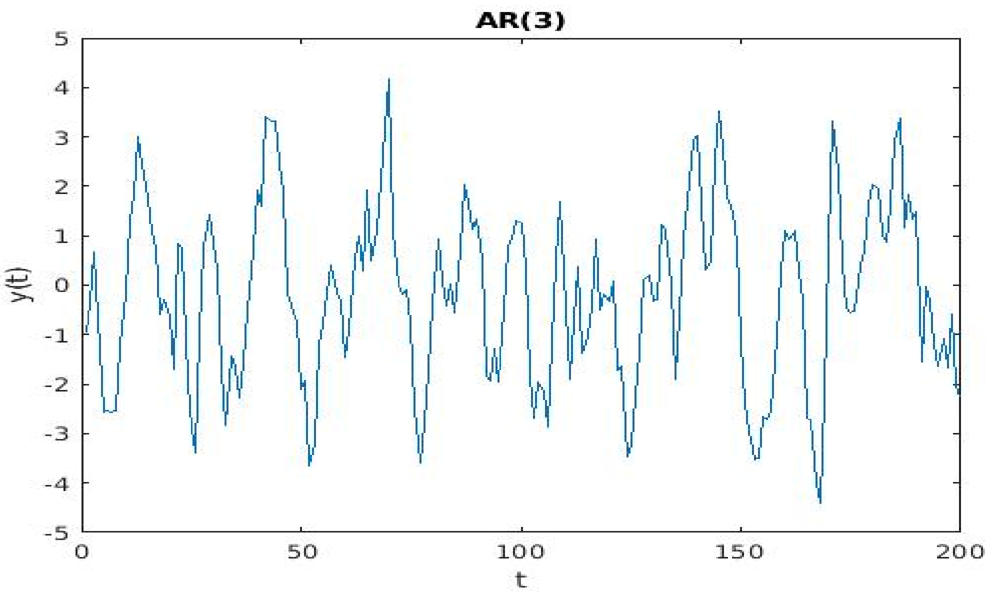

Figure 8: Simulation of AR(3) with $root_1 = 0.6$,$root_2 =$ 0.3+0,2j,$root_3 =$ 0.3-0.2j $\gamma_0 = 3.96$. True parameters of the model are $[1.20, -0.49, 0.078]$

Matlab implementation

In this question, I used `ar3_sim.m` . The explanations were commented on the code. Output were generated by calling the function on command line with different inputs.

(b)

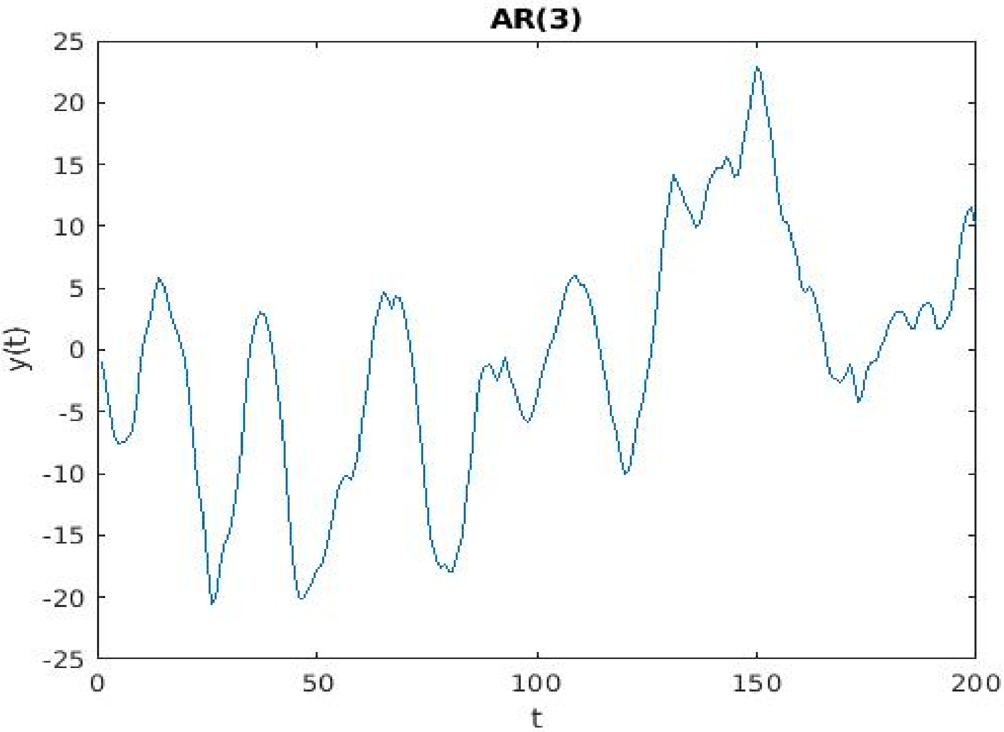

Figure 9: Simulation of AR(3) with $root_1 = 0.9$,$root_2 = 0.7$,$root_3 = 0.5$ $\gamma_0 = 173$. True parameters of the model are $[2.10, -1.43, 0.32]$

1. The estimated coefficients are $[2.12, -1.19, 0.34]$.
2. The standard error of the estimates are $[0.66, 1.24, 0.66]$.
3. The difference between ground-truth and estimates are $[0.02, 0.24, 0.06]$
   Hence we, the estimates are within 2 standard error of the true value.
4. The noise variance is 0.8822

Matlab implementation

The noise variance estimator is deducted by the acf of γ_0 to get relationship between γ_0 and σ^2(noise variance). Then, I just use empirical value to perform the calculation. In this question, I used `Q3_main.m` `ar3_sim.m` and `OLS_AR.m` I tried estimation with mean correction and without mean correction. It does not have huge effect. As the theoretical mean is zero, most of time, the empirical mean is also a tiny value. As we knew the mean is zero, for this question specifically, I will recommend not to perform mean correction. However, for real data, mean correction is absolutely necessary.
More explanation were commented on the code.

(c)

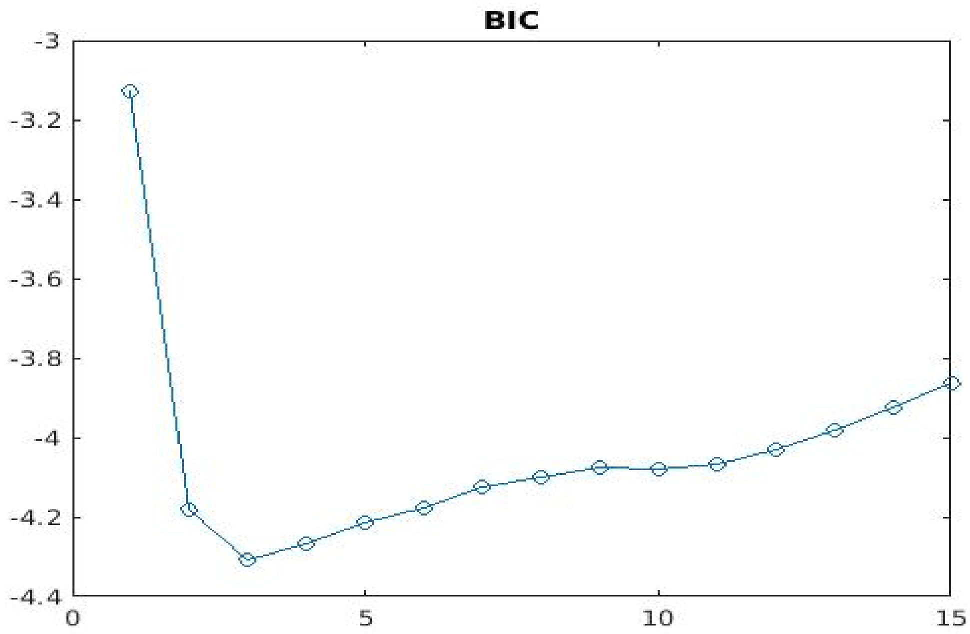

Figure 10: BIC for m = 0...15

Figure 7 is suggesting to use order 3 which makes sense. As our original model has order 3. Then I create model estimate again using order 3. It gives following parameters and standard errors(true parameters are [2.1,-1.43,0.32]):

Estimated Parameter 1	Estimated Parameter 2	Estimated Parameter 3
2.08	-1.39	0.27
Standard error	Standard error	Standard error
1.12	2.14	1.13

Table 3: Parameters estimation and standard error for order 3

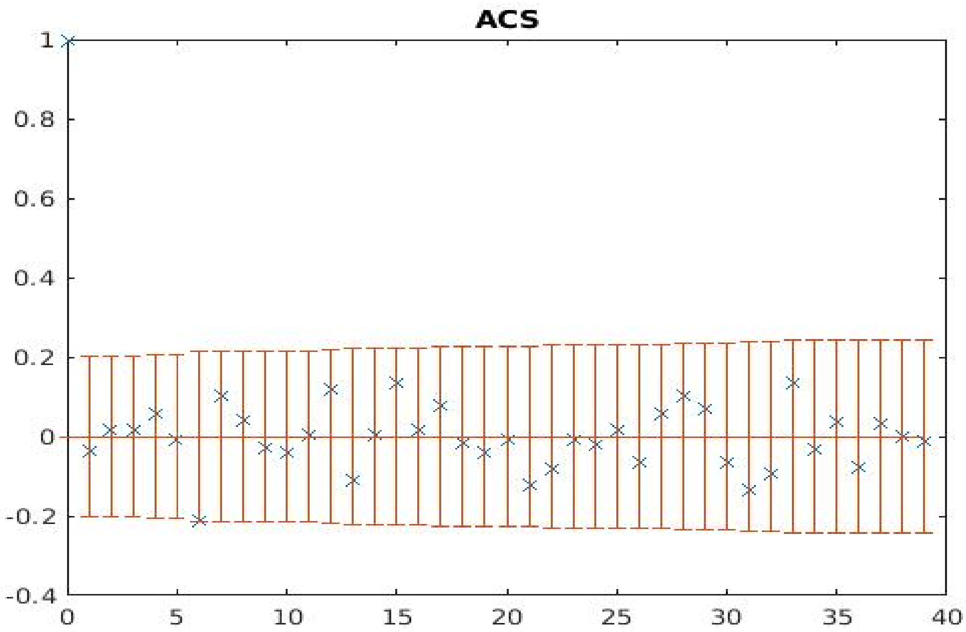

Figure 11: Auto-correlation sequence of the residuals

It can be seen that the value of acs data points, at different time shift, are very closed to zero and within the standard error around zero mean. It indicates that the residuals might just be some white noises, which is pretty normal for predication application. Hence, we can conclude our estimator's performance and model order are good.

Matlab implementation

In (C), I used `Q3BIC.m`, `my_acs.m` and `OLS_AR.m`. Explanations were commented on the code.

Appendix

`output_sim.m`

```
%%Input:
% A  —-> parameter A   fau—-> \phi     T—-> samples of output
% alpha —->\alpha      vsnr —->variance of signal to noise ratio

% Output:
% u —->u_t s—->s_t  y—->y_t h—->FIR filter var_y —->empirical variance of y
% var_s —->empirical variance of s   var_u —->empirical variance of u
% var_y —->empirical variance of y   theo_y —-> theoretical variance of y
% theo_u —-> theoretical variance of s theo_u —-> theoretical variance of u
function [u,s,y,h, var_u, var_y, var_s,theo_u,theo_y,theo_s] = ...
    output_sim(A,fau,alpha,vsnr,T)
    h = [A,-2*A*alpha,A];                % create the h filter
    r_0 = vsnr;                          % r_0 = sigma_s^2 = var(n)*vsnr = 1*vsnr(n)
    eps   = normrnd(0, 1, T, 1);         % white noise u = 0; sigma = 1

    %%% cacluating the sigma_eta by the relationship deducted in Q1
    sigma_eta = sqrt((r_0*(1-fau*fau))/((1-4*alpha*fau+fau*fau+2*alpha*alpha)*(2*A*A)));
    eta   = normrnd(0, sigma_eta, T+length(h)-1, 1);
    u = zeros(T,1);
    u(1) = eta(1);
    %%%simulate the u which is an AR 1 process
    for t = 2:T+length(h)-1              % add few extra points for get 250 and correctly ...
        convolved points
        u(t) = fau*u(t-1)+eta(t);        %  to simulate the u
    end

    s = conv(h,u);                       % perform covolution
    s = s(length(h):end-2);              % trim out the appropriate the signal
    u = u(length(h):end);
    y = s+eps;                           % add noise
    var_s = var(s);                      % empirical variance s
    var_y = var(y);                      % empirical variance y
    var_u = var(u);                      % empirical variance u
    theo_s = r_0;                        % theorectical variance
    theo_y = r_0 + 1;                    % var(s+n) = var(s) + var(n) + 2cov(s,n) = ...
        var(s)+var(n) = r_0+1
    theo_u = (sigma_eta^2)/(1-fau^2);    %AR 1 variance is sigma_eta^2/(1-fau^2)



    figure(1)                            % plot  u_t , s_t , y_t figure
    subplot 231
    plot(u);
    axis([1 250 -4 4]);
    xlabel("t(samples)");
    title("u vs t");
    camroll(90)
    subplot 232
    plot(s);
    axis([1 250 -4 4]);
    xlabel("t(samples)");
    title("s vs t");
    camroll(90)
    subplot 233
    plot(y);
    axis([1 250 -4 4]);
    xlabel("t(samples)");
    title("y vs t");
    camroll(90)
    subplot 234
    histogram(u);
    title("histogram(u)");
    axis([-6 6 0 80]);
    subplot 235
    histogram(s);
```

```
    title("histogram(s)");
    axis([-6 6 0 80]);
    subplot 236
    histogram(y);
    title("histogram(y)");
    axis([-6 6 0 80]);

    figure(2);                        %plot frequency response
    freqz(h,1);
end
```

Q2.m

```
clear;clc;
[u,s,y,h] = output_sim(1,-0.8,1,0.5,1000);       %simulating T= 1000 data points
figure;
subplot 421
FIR_estimates_2 = LSE(y,u,2);                    %find estimates with m = 2 and noisy ...
    data y
plot(FIR_estimates_2,'o--');                     %plot the impulse response which are ...
    estimates
hold on
plot([h,zeros(1,(length(FIR_estimates_2)-length(h))-1)],'x--');  %plot oringinal ...
    impulse response
title('Estimated values based on noisy ouput y (m = 2)');
legend('estimated value','true value')

subplot 423
FIR_estimates_4 = LSE(y,u,4);                   %find estimates with m = 4 and noisy ...
    data y
plot(FIR_estimates_4,'o--');                    %plot the impulse response which are ...
    estimates
hold on
plot([h,zeros(1,(length(FIR_estimates_4)-length(h))-1)],'x--');
title('Estimated values based on noisy ouput y (m = 4)');
legend('estimated value','true value')          %plot oringinal impulse response

subplot 425
FIR_estimates_8 = LSE(y,u,8);                   %find estimates with m = 8 and noisy ...
    data y
plot(FIR_estimates_8,'o--');                    %.....
hold on
plot([h,zeros(1,(length(FIR_estimates_8)-length(h)))],'x--');
title('Estimated values based on noisy ouput y (m = 8)');
legend('estimated value','true value')          %.....


subplot 427
FIR_estimates_16 = LSE(y,u,16);                %find estimates with m = 16 and noisy ...
    data y
plot(FIR_estimates_16,'o--');                  %.....
hold on
plot([h,zeros(1,(length(FIR_estimates_16)-length(h)))],'x--');
title('Estimated values based on noisy ouput y (m = 16)');
legend('estimated value','true value')         %.....

%From here, the procedure is the same, except I used clean data s to do
%estimation
subplot 422
FIR_estimates_ture_2 = LSE(s,u,2);
plot(FIR_estimates_ture_2,'o-');
hold on
plot([h,zeros(1,(length(FIR_estimates_ture_2)-length(h))-1)],'x--');
title('Estimated values based on clean ouput s (m = 2)');
legend('estimated value','true value')


subplot 424
FIR_estimates_ture_4 = LSE(s,u,4);
plot(FIR_estimates_ture_4,'o-');
```

```
hold on
plot([h,zeros(1,(length(FIR_estimates_ture_4)-length(h))-1)],'x--');
title('Estimated values based on clean ouput s (m = 4)');
legend('estimated value','true value')


subplot 426
FIR_estimates_ture_8 = LSE(s,u,8);
plot(FIR_estimates_ture_8,'o-');
hold on
plot([h,zeros(1,(length(FIR_estimates_ture_8)-length(h)))],'x--');
title('Estimated values based on clean ouput s (m = 8)');
legend('estimated value','true value')


subplot 428
FIR_estimates_ture_16 = LSE(s,u,16);
plot(FIR_estimates_ture_16,'o-');
hold on
plot([h,zeros(1,(length(FIR_estimates_ture_16)-length(h)))],'x--');
title('Estimated values based on clean ouput s (m = 16)');
legend('estimated value','true value')
%%
%Section below showed the frequency response of estimated FIR filter
%coefficients.
figure
subplot 421
[resp,w] = freqz(h,1);                          %% getting frequency response using freqz
plot(w/pi,20*log10(abs(resp)),'-')              % plot true frequency response of the ...
    FIR filter
ax = gca;
ax.YLim = [-100 20];
ax.XTick = 0:.5:2;
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude (dB)')
hold on
[resp,w] = freqz(FIR_estimates_2,1);            %% getting frequency response using freqz
plot(w/pi,20*log10(abs(resp)),'--');            %%normalize the gain to dB and ...
    frequency w to rad.
legend('estimated frequency response','true frequency response')
%%% Rest of sections are repeative, which plotted 8 graphs, left 4 are
%%% estimated on Noisy data y ,right 4 are on clean data s
subplot 423
[resp,w] = freqz(h,1);
plot(w/pi,20*log10(abs(resp)),'-')
ax = gca;
ax.YLim = [-100 20];
ax.XTick = 0:.5:2;
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude (dB)')
hold on
[resp,w] = freqz(FIR_estimates_4,1);
plot(w/pi,20*log10(abs(resp)),'--');
legend('estimated frequency response','true frequency response')

subplot 425
[resp,w] = freqz(h,1);
plot(w/pi,20*log10(abs(resp)),'-')
ax = gca;
ax.YLim = [-100 20];
ax.XTick = 0:.5:2;
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude (dB)')
hold on
[resp,w] = freqz(FIR_estimates_8,1);
plot(w/pi,20*log10(abs(resp)),'--');
legend('estimated frequency response','true frequency response')


subplot 427
[resp,w] = freqz(h,1);
```

```
plot(w/pi,20*log10(abs(resp)),'-')
ax = gca;
ax.YLim = [-100 20];
ax.XTick = 0:.5:2;
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude (dB)')
hold on
[resp,w] = freqz(FIR_estimates_16,1);
plot(w/pi,20*log10(abs(resp)),'--');
legend('estimated frequency response','true frequency response')

subplot 422
[resp,w] = freqz(h,1);
plot(w/pi,20*log10(abs(resp)),'-')
ax = gca;
ax.YLim = [-100 20];
ax.XTick = 0:.5:2;
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude (dB)')
hold on
[resp,w] = freqz(FIR_estimates_ture_2,1);
plot(w/pi,20*log10(abs(resp)),'--');
legend('estimated frequency response','true frequency response')

subplot 424
[resp,w] = freqz(h,1);
plot(w/pi,20*log10(abs(resp)),'-')
ax = gca;
ax.YLim = [-100 20];
ax.XTick = 0:.5:2;
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude (dB)')
hold on
[resp,w] = freqz(FIR_estimates_ture_4,1);
plot(w/pi,20*log10(abs(resp)),'--');
legend('estimated frequency response','true frequency response')


subplot 426
[resp,w] = freqz(h,1);
plot(w/pi,20*log10(abs(resp)),'-')
ax = gca;
ax.YLim = [-100 20];
ax.XTick = 0:.5:2;
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude (dB)')
hold on
[resp,w] = freqz(FIR_estimates_ture_8,1);
plot(w/pi,20*log10(abs(resp)),'--');
legend('estimated frequency response','true frequency response')

subplot 428
[resp,w] = freqz(h,1);
plot(w/pi,20*log10(abs(resp)),'-')
ax = gca;
ax.YLim = [-100 20];
ax.XTick = 0:.5:2;
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude (dB)')
hold on
[resp,w] = freqz(FIR_estimates_ture_16,1);
plot(w/pi,20*log10(abs(resp)),'--');
legend('estimated frequency response','true frequency response')
```

LSE.m

```
%input:
%y--> output data of the model u-->input data of the model
%model order to use for modelling
%output:
```

```
%FIR_estimates--> a vector contain estimated coefficients
function FIR_estimates = LSE(y,u,order)
    row = zeros(1,order);
    matri = zeros(length(y)-order,order);
%%%constructing X matrix                order is just m
    for t = order+1:length(y)
        for i = 1:order
            row(i) = u(t-i+1);                %FIR, so only u and its delay on each row
                                              % u_t + u_t-1... until hit the
                                              % order m
        end
        matri(t-order,:) = row;               %put rows together, we got our matrix X
    end
    X = matri;
    X_T = matri.';
    y = y(order+1:end);
    % performaning matrix caculating for estimates
    FIR_estimates = (X_T*X)\X_T*y;
end

 %y=y-mean(y);                                              % we knew there is no mean,
                                                             %So I skipped this step, in
                                                             %Q3 I had more
                                                             %discussion about
                                                             %whther or not to
                                                             %perform the mean
                                                             %correction
```

`FIR_hist.m`

```
FIR_estimated_matrix = zeros(50,4);
for B = 1:50
    [u,s,y,h] = output_sim(1,-0.8,1,0.5,1000);        %perform simulation for each ...
        estimation
    FIR_estimates_4 = LSE(y,u,4);                     %perform estimation with order 4
    FIR_estimated_matrix(B,:) = FIR_estimates_4;      %construct estimatied ...
        coefficients matrix for histogram plots
end                                                   %rows are 4 coefficients for one ...
    estimation
figure;
subplot 411
histogram(FIR_estimated_matrix(:,1),10);              %histogram on first column(first ...
    coefficient)
axis([0 2 0 15]);
subplot 412
histogram(FIR_estimated_matrix(:,2),10);              %histogram on second ...
    column(second coefficient)
subplot 413
histogram(FIR_estimated_matrix(:,3),10);              %histogram on third column(third ...
    coefficient)
axis([0 2 0 20]);
subplot 414
histogram(FIR_estimated_matrix(:,4),10);              %histogram on fourth ...
    column(fourth coefficient)
axis([-1 1 0 20]);
```

Q3_main.m

```
clear;
[rho1,rho2,rho3,r_0,output]  = ar3_sim(0.9,0.7,0.5); %%%generate output data points
[coeff,std_err,¬,¬,noise_var] = OLS_AR(output,3);             %%%perform ar lest ... square estimation
```

ar3_sim.m

```
%%input:
%three real number or one complex and one real;
%%output:
%rho1,2,3 --> parameters of AR(3)  %r_0 = \gamma_0
%output --> the output data of simulation
function [rho1,rho2,rho3,r_0,output] = ar3_sim(varargin)

%%%input checking
    if (nargin ≠ 2 && nargin ≠ 3) error('Wrong number of inputs');end
    if(nargin == 3)
        assert(isreal(varargin{1}),'all input must be real or one complex one real');
        assert(isreal(varargin{2}),'all input must be real or one complex one real');
        assert(isreal(varargin{3}),'all input must be real or one complex one real');
        root_vec = [varargin{1},varargin{2},varargin{3}];
        p = poly(root_vec);    %get the coefficients of a polynomial given roots of it.
    end
    if(nargin == 2)
        assert((isreal(varargin{1}) && imag(varargin{2}) ≠ 0)||(isreal(varargin{2}) && ...
            imag(varargin{1}) ≠ 0),...
        'all input must be real or one complex one real' );
        root_vec = [varargin{1},varargin{2},conj(varargin{2})];
        p = poly(root_vec);        %get the coefficients of a polynomial given roots of it.
    end
T = 200;                       % number of simulation points
y      = zeros(T+3,1);
y(1)   = 0;                    %y1,y2,y3 here are actually y(-1),y(-2),y(-3) for
y(2)   = 0;                    %true y(1) kicked in at y(4);
y(3)   = 0;
rho1   = -p(2);                %first coefficient
rho2   = -p(3);                %second coefficient
rho3   = -p(4);                %third coefficient
sigma = 1;
mu_e   = 0;
eps    = normrnd(mu_e, sigma, T, 1);   %creating white noises for final output

%%% the actual recursive simulation
for t=4:T+3
    y(t) = rho1*y(t-1) + rho2*y(t-2) + rho3*y(t-3) + eps(t-3);
end
%%% constructing the matrix to caculate the r_0,r_1,r_2,r_4
% this is contructed from system of linear equations of four unknowns
col_1 = [-1,rho1,rho2,rho3];
col_2 = [rho1,rho2-1,rho3+rho1,rho2];
col_3 = [rho2,rho3,-1,rho1];
col_4 = [rho3,0,0,-1];

A = [col_1;col_2;col_3;col_4].';
b = [-sigma^2,0,0,0].';
r = A\b;                        %we got result as [r_0,r_1,r_2,r_3]
r_0 = r(1);
output = y(4:end);              %trim out the proper output data


%Plot the series
figure
plot((y(4:end)));
title('AR(3)');
xlabel('t')
ylabel('y(t)')
```

```
end
```

OLS_AR.m

```
%input:
%y--> output data of the model u-->input data of the model
%model order to use for modelling
%output:
%coeff--> a vector contain estimated coefficients
%std_err--> standard error of estimates
%residual--> resiual of the esitmator
%sigma_sq -->variance of residual
%noise_var-->estimated noise variance
function [coeff,std_err,residual,sigma_sq,noise_var] = OLS_AR(y,order)
    y=y-mean(y);
    z = mean(y);
    row = zeros(1,order);
    matri = zeros(length(y)-order,order);
    for t = order+1:length(y)              %creating X matrix
        for i = 1:order
            row(i) = y(t-i);

        end
        matri(t-order,:) = row;
    end
    X = matri;                              %X
    X_T = matri.';                          %X^T
    y = y(order+1:end);                     %y m...T
    coeff = (X_T*X)\X_T*y;                  %inv(X*X^t)*X^T*y
    std_err_mat = var(y)*inv(X_T*X);
    std_err = sqrt(diag(std_err_mat));      % sqrt of diagonal of variance of ...
        beta(estimator)...
                                            %= standard error of estimator
    residual = y - X*coeff;                 % residual
    sigma_sq = var (residual);              % sigma_sq for BIC caculation (variance of ...
        residual)

 if (order == 3)
   rho1 = coeff(1);
   rho2 = coeff(2);
   rho3 = coeff(3);
%%%caculating empirical \gamma_0
    my_sum = 0;
       for k = 1:length(y)
           my_sum = my_sum + (y(k)-mean(y))*(y(k)-mean(y));
       end
    r_0 = my_sum/length(y);

%%% Noise variance estimator from relationship between r_0 and sigma^2,
%%% deducted from correation of the signal with itself.
    noise_var = (1-rho3^2-(4*rho1*rho2*rho3+rho1^2+rho2^2+rho1^2*rho2-rho2^3+...
    rho1^3*rho3+rho1^2*rho3^2-rho1*rho2^2*rho3+rho2^2*rho3^2)/(1-rho2-rho1*rho3-rho3^2))*r_0;
 end
    noise_var = 0;
end
```

Q3BIC.m

```
%%%simulate the data first
T = 100;
p1 = 0.9;
p2 = 0.7;
p3 = 0.5;
[rho1,rho2,rho3,¬,output] = ar3_sim(p1,p2,p3);
%%%define the upper bound of the system's order and perform BIC
total_order = 15;
BIC = zeros(1,total_order);
%%%BIC caulation
    for order = 1:total_order
        [¬,¬,¬,sigma_sq,¬] = OLS_AR(output,order);
        sigma_0 = var(output(order+1:end));                    %%empirical sigma_0;
        BIC(order) = log(sigma_sq/sigma_0)+ order * log(T)/T;   %%following the formula ...
            on slides
    end
%%%caculating BIC^2
BIC_sq = BIC.^2;
%%%find the optial order by minimizing BIC
optimal_order = find(BIC == min(BIC(:)));
plot(BIC,'o-');
title('BIC')
%%% use opitimal order to model again
 [coeff,std_err,residual,¬] = OLS_AR(output,3);
 t = 40;
 %%% caculating the acs
 [acs,acs_std_err] = my_acs_fun (residual,t);
 figure;
 plot(0:(t-1),acs,'x');
 title('ACS')
 hold on
 errorbar(0:(t-1),zeros(1,t),2*acs_std_err);    %%plotting 2 standard errorbar around zero
```

my_acs.m

```
function [acs,std_err] = my_acs_fun (residual,t)
   acs = ones(1,t);
   sigma_sq = var(residual);
   u = mean(residual);
   std_err = zeros(1,t);
   % caculating empirical autocorelation for each invidual residual point
   for i = 2:t
       my_sum = 0;
       for k = 1:length(residual)-i
           my_sum = my_sum + (residual(k)-u)*(residual(k+i)-u);
       end
       acs(i) = my_sum/(length(residual)*sigma_sq);
   end
   %caculating the standard error using formula on page 7 of slide 4b
   for i = 2:t
       std_err(i) = sqrt((1+2*sum(acs(2:i).^2))/length(residual));
   end
end
```