University of New South Wales

School of Electrical Engineering and Telecommunications

INDIVIDUAL Assessment Task: ELEC9782 Mid-term

Course Code	Elec9782	Course Name	Data science
Week/Session/Year	4	Lecturer	Prof. Victor Solo

Student Number	z5036602
Family Name	LIU
Given Names	Zhengyue

Mark/Grade given (For official use only)

Marker's Comments:

Since this work counts toward your formal assessment for this course, please write your name and student number where indicated above, and sign the declaration below. Attach this cover sheet to the front of your submission, so that your name and student number can be seen without any cover needing to be opened.

For further information, please see http://www.lc.unsw.edu.au/plagiarism/index.html

---

I declare that this assessment item is my own work, except where acknowledged, and has not been submitted for academic credit elsewhere, and acknowledge that the assessor of this item may, for the purpose of assessing this item:

- ***Reproduce this assessment item and provide a copy to another member of the University ; and/or,***
- ***Communicate a copy of this assessment item to a plagiarism checking service (which may then retain a copy of the assessment item on its database for the purpose of future plagiarism checking).***

I certify that I have read and understood the University Rules in respect of Student Academic Misconduct.

Signature of student ____________ Date: 19/7/2019

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)

MA(1) is stationary, and theoretical acv of MA(1) process is thus given by:

$$\begin{aligned}
\gamma_r &= \sigma_v^2(1+\theta^2),\ r=0 \\
\gamma_r &= -\theta\sigma_v^2,\ r=1 \\
\gamma_r &= 0, \text{ otherwise}
\end{aligned}$$

The spectrum is the Fourier series of the covariances:

$$\begin{aligned}
\mathbb{F}(\omega) &= \sum_{-\infty}^{\infty} \gamma_r e^{-j\omega r} \\
&= \gamma_0 + 2\sum_{1}^{\infty} \gamma_r \cos(\omega r)
\end{aligned}$$

Hence, the spectrum sum of MA(1) collapses to:

$$\begin{aligned}
\mathbb{F}_x(\omega) &= \gamma_0 + 2\gamma_1 \cos(\omega) \\
&= \sigma_v^2(1+\theta^2) - 2\theta\sigma_v^2\cos(\omega) \\
&= \sigma_v^2(1+\theta^2 - 2\theta\cos(\omega))
\end{aligned}$$

We can easily get the transfer function h as following:

$$\begin{aligned}
s_t &= \frac{b}{1-a^2z^{-2}} x_t \\
h(z) &= \frac{b}{1-a^2z^{-2}}
\end{aligned}$$

From result F2 we have:

$$\begin{aligned}
\mathbb{F}_s(\omega) &= |h(e^{-j\omega})|^2\mathbb{F}_x(\omega) \\
&= h(e^{-j\omega})h(e^{j\omega})\mathbb{F}_x(\omega) \\
&= \frac{b}{1-a^2e^{-2j\omega}}\frac{b}{1-a^2e^{2j\omega}}\mathbb{F}_x(\omega) \\
&= \frac{b^2}{1-a^2(e^{2j\omega}+e^{-2j\omega})+a^4}\mathbb{F}_x(\omega) \\
&= \frac{b^2}{1-2a^2\cos(2\omega)+a^4}\mathbb{F}_x(\omega) \\
&= \frac{b^2}{1-2a^2\cos(2\omega)+a^4}\sigma_v^2(1+\theta^2-2\theta\cos(\omega)) \\
&= \frac{b^2\sigma_v^2(1+\theta^2-2\theta\cos(\omega))}{1-2a^2\cos(2\omega)+a^4}
\end{aligned}$$

From matrix filtering deduction in slide 8, L(5b), if we have:

$$\mathbb{Y}_t = h(z^{-1})x_t + n_t$$

Then

$$
\begin{aligned}
\mathbb{F}_y(\omega) &= |h|^2\mathbb{F}_x(\omega) + \mathbb{F}_\epsilon(\omega) \\
&= \mathbb{F}_s(\omega) + \mathbb{F}_\epsilon(\omega)
\end{aligned}
$$

Since for white noise $\gamma_0 = \sigma^2$, $\gamma_r = 0$, $r \neq 0$ so the spectrum has one term: $\mathbb{F}_\epsilon(\omega) = \sigma_\epsilon^2$,

$$
\begin{aligned}
\mathbb{F}_y(\omega) &= \mathbb{F}_s(\omega) + \mathbb{F}_\epsilon(\omega) \\
&= \frac{b^2\sigma_v^2(1+\theta^2-2\theta\cos(\omega))}{1-2a^2\cos(2\omega)+a^4} + \sigma_\epsilon^2
\end{aligned}
$$

From result F3 we have:

$$
\begin{aligned}
\mathbb{F}_{yx}(\omega) &= h(e^{-j\omega})\mathbb{F}_x(\omega) \\
&= \frac{b}{1-a^2e^{-2j\omega}}\sigma_v^2(1+\theta^2-2\theta\cos(\omega))
\end{aligned}
$$

For σ_s^2, firstly we modify the transfer function to difference equation.

$$
\begin{aligned}
s_t &= \frac{b(1-\theta z^{-1})v_t}{1-a^2z^{-2}} \\
s_t - a^2 s_{t-2} &= bv_t - b\theta v_{t-1} \\
s_t &= bv_t - b\theta v_{t-1} + a^2 s_{t-2}
\end{aligned}
$$

Then,

$$
\begin{aligned}
\sigma_x^2 &= \gamma_0 \\
&= \sigma_v^2(1+\theta^2) \\
\sigma_s^2 &= \mathbb{E}(s_t^2) \\
&= \mathbb{E}(bv_t - b\theta v_{t-1} + a^2 s_{t-2})^2 \\
&= b^2\mathbb{E}(v_t^2) + b^2\theta^2\mathbb{E}(v_{t-1}^2) + a^4\mathbb{E}(s_{t-2}^2) - 2b^2\theta\mathbb{E}(v_t v_{t-1}) + \\
&\quad 2a^2b\mathbb{E}(v_t s_{t-2}) - 2a^2b\theta\mathbb{E}(v_{t-1}s_{t-2}) \\
&= \frac{b^2\sigma_v^2 + b^2\theta^2\sigma_v^2}{1-a^4} \\
&= b^2\sigma_v^2\frac{\theta^2+1}{1-a^4} \\
\sigma_y^2 &= \mathbb{E}(y_t^2) \\
&= \mathbb{E}((s_t+\epsilon_t)^2) \\
&= \mathbb{E}(s^2 + 2s_t\epsilon_t + \epsilon_t^2) \\
&= Var(s_t) + Var(\epsilon_t) \\
&= b^2\sigma_v^2\frac{\theta^2+1}{1-a^4} + \sigma_\epsilon^2 \\
VSNR &= \frac{\sigma_s^2}{\sigma_\epsilon^2} \\
&= \frac{b^2\sigma_v^2(1+\theta^2)}{\sigma_\epsilon^2(1-a^4)}
\end{aligned}
$$

Problem (b)

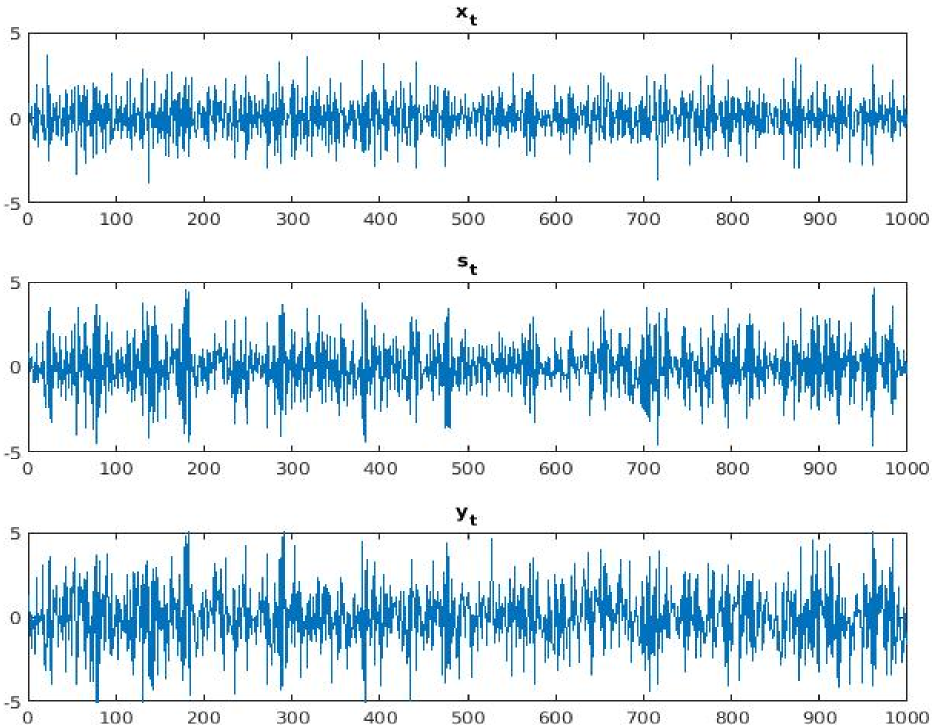

Figure 1: $a = 0.8 \quad b = 1 \quad \sigma_\epsilon = 1 \quad \theta = 0.7 \quad \sigma_v^2 = 1$

Empirical VSNR	Theoretical VSNR
2.61	2.52

Matlab implementation

I created x_t first by filtering the white noise v_t using the FIR filter $(1 - \theta z^{-1})$, then I filter the x_t by using IIR filter $\frac{b}{1-a^2z^{-2}}$ to get s_t. Finally, I add white noise ϵ_t with s_t to get y_t. To get empirical VSNR i just use $\frac{var(\sigma_s^2)}{var(\sigma_\epsilon^2)}$. To get theoretical VSNR, I used the formula I deduced in part (a).
The code involved in this part are `system_simulation.m` and first section of `spectrum_estimation.m`. More details are commened on code.

Problem (c)

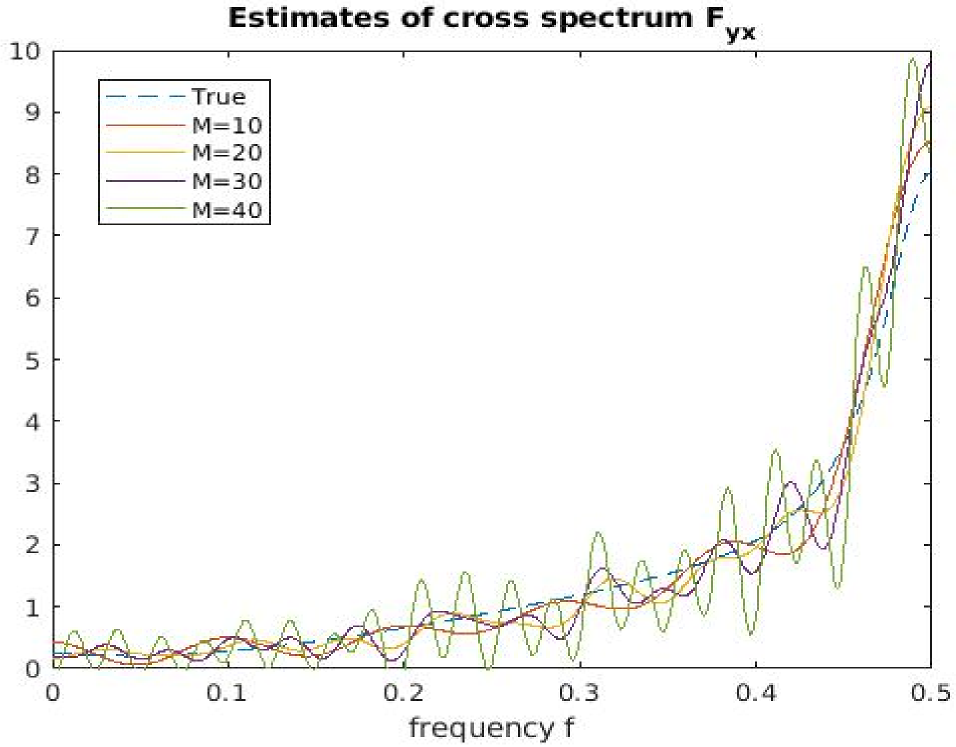

Figure 2: F_{yx}

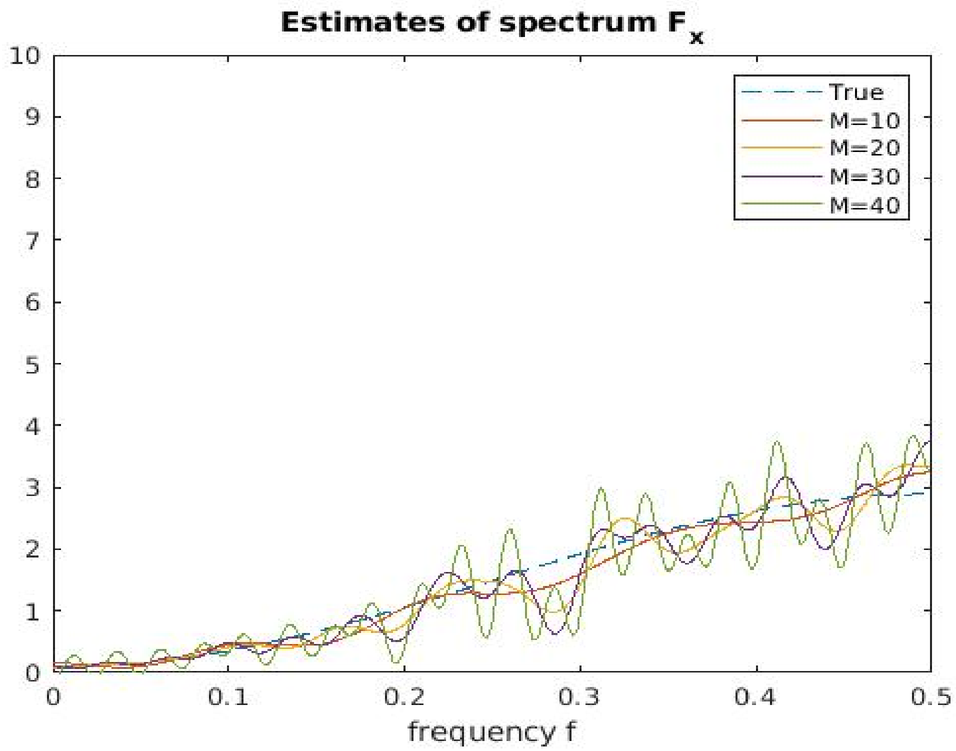

Figure 3: F_x

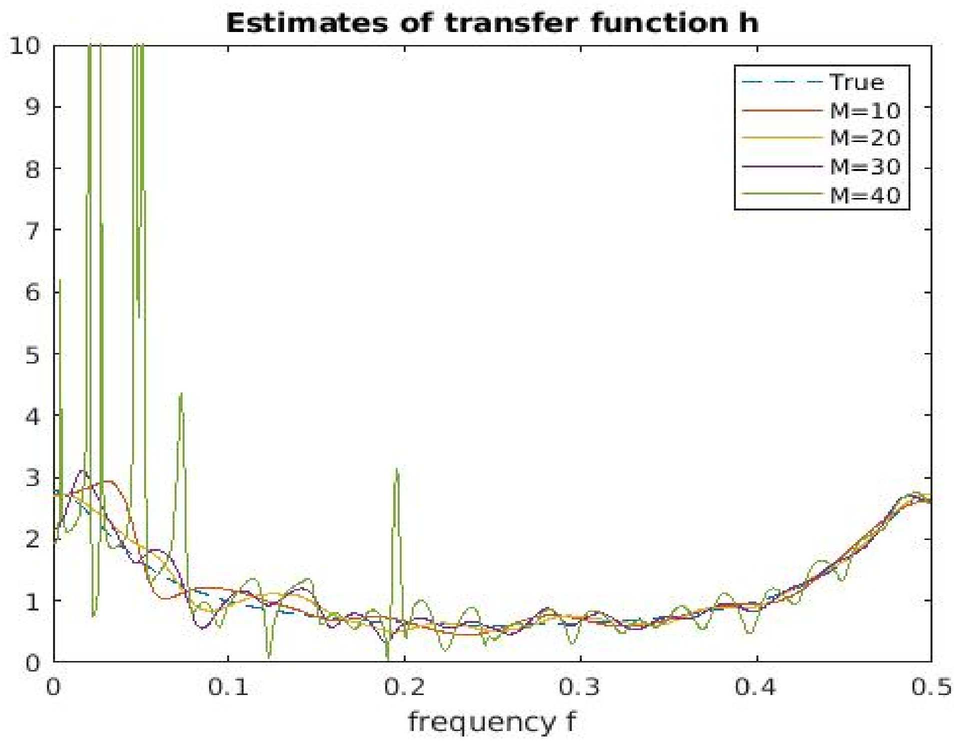

Figure 4: Transfer function H

It can be seen that, increase the co-variances lags for estimation can make the estimated value extremely noisy. We should decide the lags by trials and errors. Too few lags may lead to loss of details on estimated value.

Matlab implementation

The code in this part includes `spectrum_estimator.m` and `empirical_autocorr.m` and `empirical_cross_autocorr.m` and `spectrum_estimation.m`.

For empirical covariances. The formula that I am using:

$$C_r = \frac{1}{T}\sum_{n=1}^{T-|r|}(y_t - \overline{y}))(y_{t+r} - \overline{y})$$

For empirical cross-covariances. Firstly, I performed the mean correction, then I used formula:

$$C_{yx,r} = \frac{1}{T}\sum_{n=1}^{T-|r|} y_t x_{t+r} \qquad \gamma >= 0$$

For negative time lag covariances, I used the following formula, then flipped the result as the negative part.

$$C_{yx,k} = \frac{1}{T}\sum_{n=1}^{T-|k|} x_t y_{t+k} \qquad k = -\gamma \quad \gamma < 0$$

For spectrum estimator I used formula :

$$F_M(\omega) = \sum_{-M}^{M} C_r e^{-j\omega r}$$

More details were commented on code

More plots for having fun and confirmming the correctness

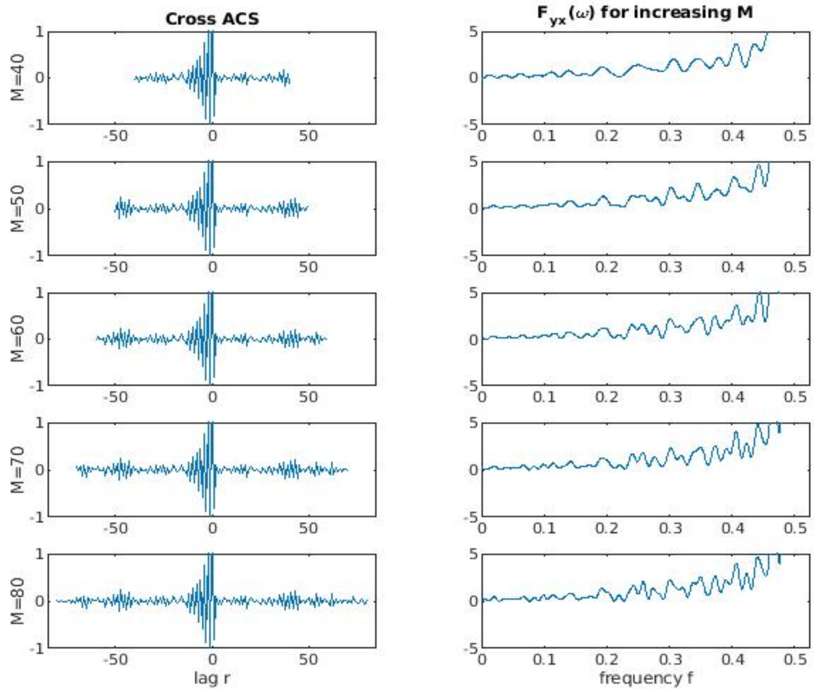

Figure 5: F_{yx} for $M = 40...80$ and covariances

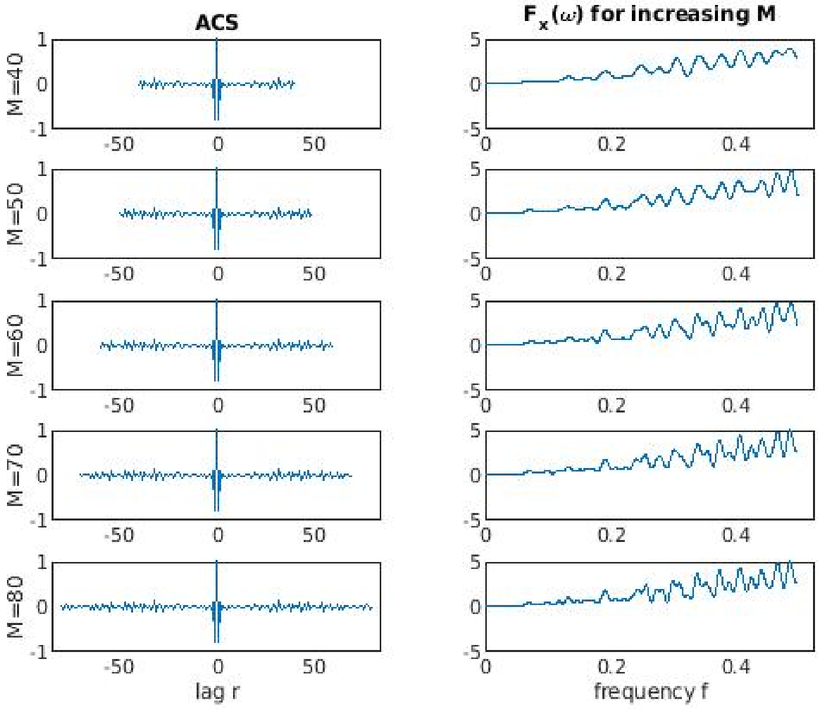

Figure 6: F_x for $M = 40...80$ and covariances

Question 2

Problem (a)

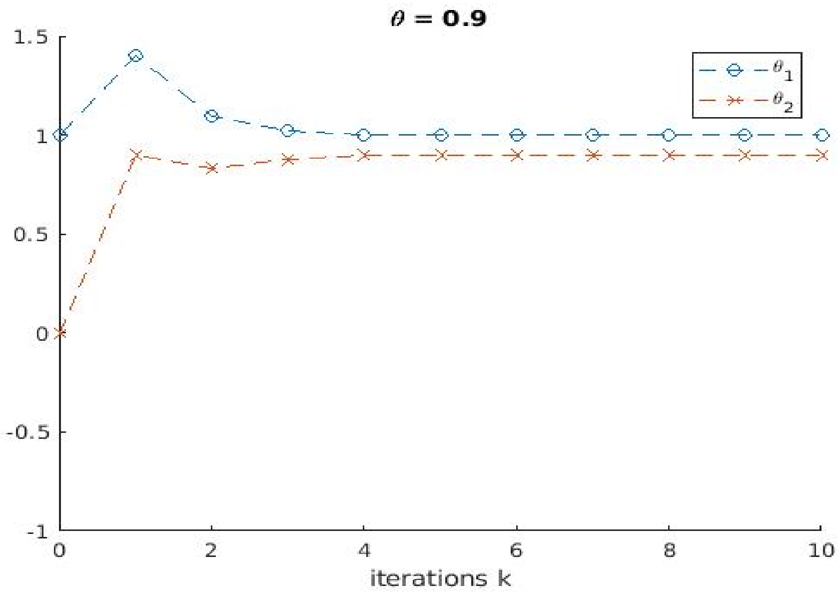

Figure 7: Spectral factorization for MA(1) prcoess with $\sigma^2 = 1, c_o = 1 + \theta^2, c_1 = -\theta, \theta = 0.9$

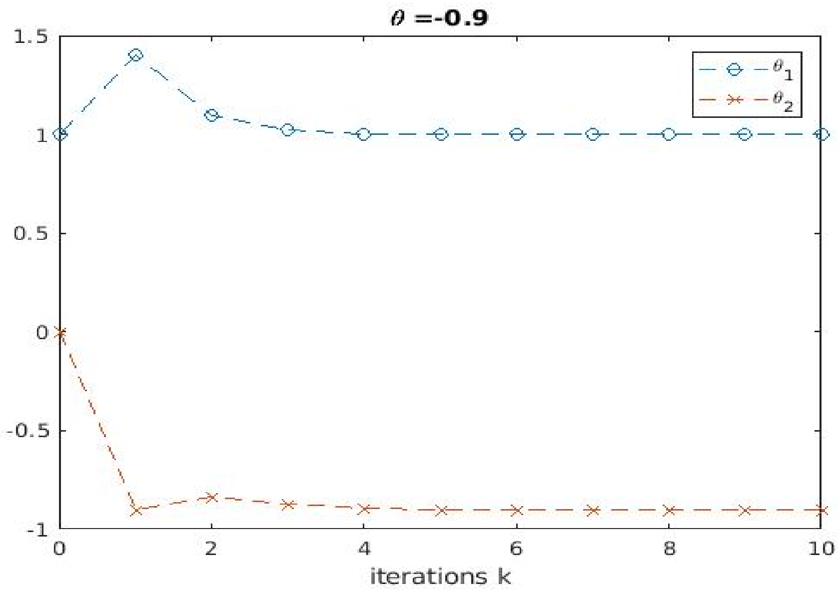

Figure 8: Spectral factorization for MA(1) prcoess with $\sigma^2 = 1, c_o = 1 + \theta^2, c_1 = -\theta, \theta = -0.9$

Matlab implementation

The code involved in this part includes `q2.m` and `wilson.m`. For wilson algorithm, I used exact same procedure shown on slide 8 lecture 6b. For building TL and TR, I used matlab default hankel and toeplitz function. More details were commented on code.

Problem (b)

(i)

Since s_t and n_t are independent, we have $F_y = F_s + F_n$. We can compute Wiener filter as:

$$
\begin{aligned}
F_s(\omega) &= \frac{\sigma_\epsilon^2}{|1-\phi e^{-j\omega}|^2} \\
F_n(\omega) &= \sigma_v^2|1-\theta e^{-j\omega}|^2 \\
F_y(\omega) &= F_s(\omega) + F_n(\omega) \\
&= \frac{\sigma_\epsilon^2}{|1-\phi e^{-j\omega}|^2} + \sigma_v^2|1-\theta e^{-j\omega}|^2 \\
&= \frac{\sigma_\epsilon^2 + \sigma_v^2|1-\theta e^{-j\omega}|^2|1-\theta e^{-j\omega}|^2}{|1-\phi e^{-j\omega}|^2} \\
\tilde{W} &= \frac{F_s(\omega)}{F_y(\omega)} \\
&= \frac{\sigma_\epsilon^2|1-\phi e^{-j\omega}|^2}{|1-\phi e^{-j\omega}|^2(\sigma_\epsilon^2 + \sigma_v^2|1-\theta e^{-j\omega}|^2|1-\phi e^{-j\omega}|^2)} \\
&= \frac{\sigma_\epsilon^2}{\sigma_\epsilon^2 + \sigma_v^2|1-\theta e^{-j\omega}|^2|1-\phi e^{-j\omega}|^2} \\
&= \frac{\frac{\sigma_\epsilon^2}{\sigma_v^2}}{\frac{\sigma_\epsilon^2}{\sigma_v^2} + |1-\theta e^{-j\omega}|^2|1-\phi e^{-j\omega}|^2} \\
&= \frac{\lambda}{\lambda + |(1-\theta e^{-j\omega})(1-\phi e^{-j\omega})|^2} \\
&= \frac{\lambda}{\lambda + |(1-(\theta+\phi)e^{-j\omega} + \theta\phi e^{-2j\omega})|^2} \\
&= \frac{\lambda}{\lambda + 1 - (\theta+\phi+\theta^2\phi+\theta\phi^2)(z^1+z^{-1}) + \theta\phi(z^2+z^{-2}) + (\theta+\phi)^2 + \theta^2\phi^2)} \\
&= \frac{\lambda}{-(\theta+\phi+\theta^2\phi+\theta\phi^2)z^1 + (\theta+\phi+\theta^2\phi+\theta\phi^2)z^{-1} + \theta\phi z^2 + \theta\phi z^{-2} + (\theta+\phi)^2 + \theta^2\phi^2 + \lambda + 1}
\end{aligned}
$$

(ii)

The denominator is finite MA spectrum. The spectrum formula is:

$$F(\omega) = \gamma_0 + 2\sum_1^\infty \gamma_r \cos(r\omega)$$

For $MA(2)$ it collapses to: $\gamma_0 + 2\gamma_1\cos(\omega) + 2\gamma_2\cos(2\omega)$, and we can calculate $F(\omega)$ as follows:

$$
\begin{aligned}
F(\omega) &= \lambda + (1-(\theta+\phi)e^{-j\omega} + \theta\phi e^{-2j\omega})(1-(\theta+\phi)e^{j\omega} + \theta\phi e^{2j\omega}) \\
&= \lambda + (1-(\theta+\phi)e^{j\omega} + \theta\phi e^{2j\omega} - (\theta+\phi)e^{-j\omega} + (\theta+\phi)^2 \\
&\quad - \theta\phi(\theta+\phi)e^{j\omega} + \theta\phi e^{-2j\omega} - \theta\phi(\theta+\phi)e^{-j\omega} + \theta^2\phi^2) \\
&= \lambda + 1 - (\theta+\phi+\theta^2\phi+\theta\phi^2)(e^{j\omega}+e^{-j\omega}) + \theta\phi(e^{2j\omega}+e^{-2j\omega}) + (\theta+\phi)^2 + \theta^2\phi^2 \\
&= \lambda + 1 + (\theta+\phi)^2 + \theta^2\phi^2 - 2(\theta+\phi+\theta^2\phi+\theta\phi^2)\cos(\omega) + 2\theta\phi\cos(2\omega)
\end{aligned}
$$

So by comparing the coefficients of $cos(\omega)$ and $\cos(2\omega)$, we have:

$$\begin{aligned}\gamma_0 &= \lambda + 1 + (\theta+\phi)^2 + \theta^2\phi^2 \\ \gamma_1 &= -(\theta+\phi+\theta^2\phi+\theta\phi^2) \\ \gamma_2 &= \theta\phi \\ \gamma_r &= 0, \text{ otherwise}\end{aligned}$$

(iii)

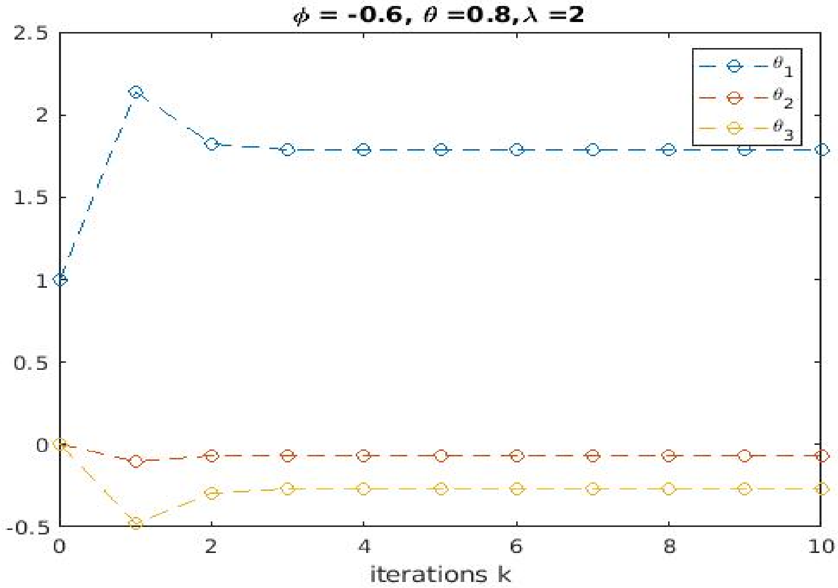

Figure 9: Spectral factorization on denominator of the Winner filter by using wilson algorithm

Once, values calculated by wilson algorithm get steady after iterations. We pick the final three steady values

$$\theta_1 = 1.7871, \theta_2 = -0.0685, \theta_3 = -0.2686$$

We can form the forward and backward filter as following:

$$\frac{\sqrt{\lambda}}{1.7871 - 0.0685z^{-1} - 0.2686z^{-2}} * \frac{\sqrt{\lambda}}{1.7871 - 0.0685z^{1} - 0.2686z^{2}}$$

Matlab implementation

The code involved in this part includes section (b) of `q2.m` and `wilson.m`. The steps are same as problem (a). Firstly, I construct the covariances vector. Then plug in the covariances vector and desired iterations to the wilson algorithm function. Fianlly, we store the outputs into $3*(iterations)$, 3 is from the order of system plus one. Then we pick values in last column which converged to the true MA coefficients. More details were commented on code.

Problem (c)

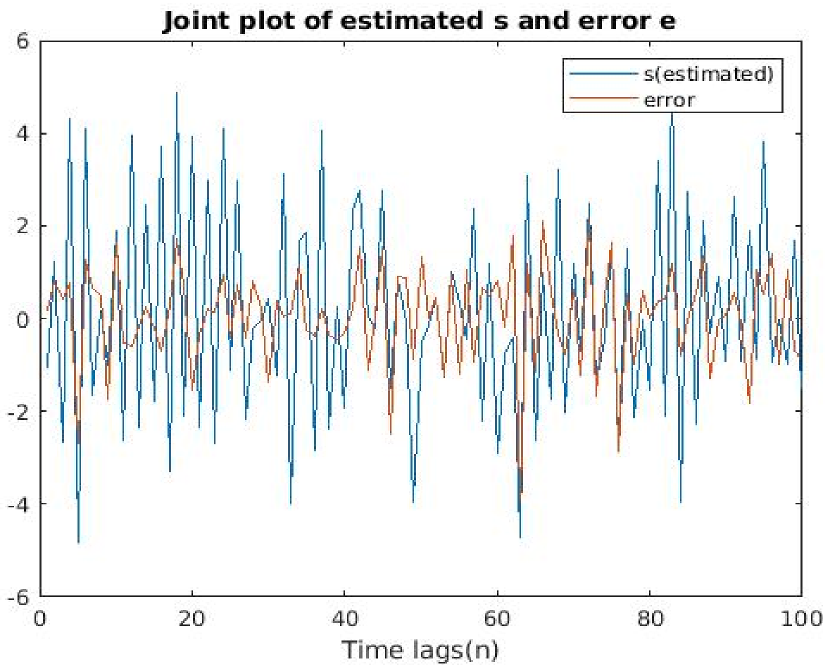

Figure 10: Estimated s after two one-sided filtering and its empirical error

Matlab implementation

The code involved in this part includes section (b) of `q2.m` and `AR_MA_simulation`. For ARMA simulation, I filtered the white noise ϵ_t by using IIR filter $\frac{1}{1-\phi z^{-1}}$ to get s_t. Then I filter the white noise v_t with $1-\theta z^{-1}$ to get n_t. Finally, I add the n_t with s_t to get the MA plus AR process y_t.

For second step, I construct the IIR forward filter with MA denominator coefficients calculated from part (b).

For third step, I filter the AR plus MA process y_t with IIR forward filter $\frac{\sqrt{\lambda}}{1.7871-0.0685z^{-1}-0.2686z^{-2}}$. Then I flip the filtered signal and pass it into the same filter again to get $output_1$. Then, flip the $output_1$ back again to get our estimated s hat.

For error, I just calculated the difference between true s and estimated s ($s - s(hat)$).

Question 3

Problem (a)

When the system is at steady state, state equation adjusted to DARE.

$$g = FPc + N \tag{1}$$
$$V = c^T Pc + R \tag{2}$$
$$P = FPF^T + Q - gV^{-1}g^T \tag{3}$$

Substitute (1) and (2) into (3):

$$P = FPF^T + Q - (FPc + N)(c^T Pc + R)^{-1}(FPc + N)^T$$

As F=1, c=1 and state is a scalar, the above equation can be calculated as:

$$\begin{aligned}
P &= P + Q - \frac{(P+N)^2}{(P+R)} \\
P^2 + (2N - Q)P + N^2 - QR &= 0 \\
P &= \frac{Q - 2N + \sqrt{(2N-Q)^2 - 4(N^2 - QR)}}{2} \\
&= \sqrt{QR}\left(\sqrt{\frac{Q}{4R}} - \frac{N}{\sqrt{QR}} + \sqrt{\frac{1}{4QR}((Q-2N)^2) - 4(N^2 - QR)}\right) \\
&= \sqrt{QR}\left(\sqrt{\frac{Q}{4R}} - \frac{N}{\sqrt{QR}} + \sqrt{\left(\sqrt{\frac{R}{4Q}} - \frac{N}{\sqrt{QR}}\right)^2 + 1 - \frac{N^2}{QR}}\right)
\end{aligned}$$

Since it is given that: $\rho = \frac{N}{\sqrt{QR}}$ and $\alpha^2 = \frac{Q}{4R}$,

$$P = \sqrt{QR}(\alpha - \rho + \sqrt{(\alpha - \rho)^2 + 1 - \rho^2})$$

Since $P_o = (\alpha - \rho + \sqrt{(\alpha - \rho)^2 + 1 - \rho^2})$,

$$P = P_o\sqrt{QR}$$

Problem (b)

$$\begin{aligned}
V &= c^T Pc + R \\
&= P + R \\
&= P_o\sqrt{QR} + R
\end{aligned}$$

Problem (c)

From equation (1) and (2) in problem (a),

$$\begin{aligned}
k &= \frac{g}{V} \\
&= \frac{P+N}{P+R} \\
&= \frac{P_o\sqrt{QR}+N}{P_o\sqrt{QR}+R} \\
&= \frac{P_o+\frac{N}{\sqrt{QR}}}{P_o+\sqrt{\frac{R}{Q}}} \\
&= \frac{P_o+\rho}{P_o+\frac{1}{2\alpha}}
\end{aligned}$$

Problem (d)

$$\begin{aligned}
\hat{\xi}_{t+1} &= \hat{\xi}_t + k(y_t - \hat{\xi}_t) \\
\hat{\xi}_{t+1} &= (1-k)\hat{\xi}_t + k y_t \\
\hat{\xi}_{t+1} &= \frac{k}{1-(1-k)z^{-1}} y_t
\end{aligned}$$

It can be seen that it is one pole filter. The pole is at (1-k). For stability. I am trying to consider the matrix stability from matrix's controllablity and observablity.
(F, g) is stabilizable (e.g. controllable)
(F, c) is detectable (e.g. observable

Appendix

Q1

`system_simulation.m`

```
%output:
% y-->y_t s-->s_t x-->x_t VSNR --> VSNR empirical_VSNR --> empirical_VSNR
% T-->lags a-->a b-->b sigma_eps -->sigma_eps sigma_v -->sigma_v
function [y,s,x,VSNR,empirical_VSNR] = system_simulation(T,a,b,sigma_eps,theta,sigma_v)
mu_e  = 0;                                  %mean zero
eps   = normrnd(mu_e, sigma_eps, T, 1);     %creating white noises eps
v_ = normrnd(mu_e, sigma_v, T, 1);          %creating whilte noise v
NUM_1 = [1,0-theta];                        %x_t = (1-theta*z^-1) v_t
DEN_1 = 1;
x = filter (NUM_1,DEN_1,v_);                %filter the signal
NUM_2 = b;                                  %s_t = b/(1-a^2*z^-2) x_t
DEN_2 = [1,0,-(a^2)];
s = filter(NUM_2,DEN_2,x);                  %filter the signal
y = s+eps;
%%%VSNR formula deduced from Q1
VSNR = (b^2 *sigma_v^2*(theta^2+1))/(sigma_eps^2*(1-a^4));
empirical_VSNR= var(s)/var(eps);      %Emepirical VSNR
```

`spectrum_estimator.m`

```
function [spectrum] = spectrum_estimator(data,lag)
    f_res = 1000;
    spectrum = ones(1,f_res);
    counter = 0;
    for f_c = 1:f_res
        w_0 = (f_c/f_res)*pi;
        total_acs = 0;
        counter = 1;
        for r = -lag:lag
            total_acs = total_acs+data(counter)*exp(-1i*w_0*r);
            counter = counter +1;
        end
        spectrum(f_c) = total_acs;
    end
end
```

`empirical_autocorr.m`

```
function [total_acs] = empirical_autocorr(data,t)
    t = t+1;
   acs = ones(1,t);
    u = mean(data);
   % caculating empirical autocorelation for each invidual data point
   for i = 1:t
       my_sum = 0;
       for k = 1:length(data)-i+1
           my_sum = my_sum + (data(k)-u)*(data(k+i-1)-u);
       end
       acs(i) = my_sum/length(data);
   end
   % flip the positive covariances (symetrical)
   acs_negative = flip(acs);
   total_acs = [acs_negative(1:end-1),acs(1:end)];
end
```

`empirical_cross_autocorr.m`

```
% %%%output :
% total_acs --> covariance vector'
% %input:
% data_y -->y    r-->lags
% data_x -->x
function [total_acs] = empirical_cross_autocorr(data_y,data_x,r)
    r = r+1;
    acs = ones(1,r);
    acs_negative = ones(1,r-1);
    u_x = mean(data_x);
    u_y = mean(data_y);
    data_y = data_y-u_y;
    data_x = data_x-u_x;
   % caculating empirical autocorelation for each invidual data point
   for i = 1:r
       my_sum = 0;
       for k = 1:length(data_y)-i+1
           my_sum = my_sum + (data_y(k))*(data_x(k+i-1));
       end
       acs(i) = my_sum/length(data_y);
   end
   counter = 1;
   %caculate the negative part by shift the y and leave the x stay the same
   for i = 2:r
       my_sum = 0;
       for k = 1:length(data_x)-i+1
          my_sum = my_sum + (data_x(k))*(data_y(k+i-1));
       end
       acs_negative(counter) = my_sum/length(data_x);
       counter = counter+1;
   end
   total_acs = [flip( acs_negative),acs];
end
```

spectrum_estimation.m

```
%%% simulate the system to generate (y,x,s)
T = 1000;
a = 0.8;
b = 1;
sigma_eps =1;
theta = 0.7;
sigma_v = 1;
[y,s,x,VSNR,empirical_VSNR] = system_simulation(T,a,b,sigma_eps,theta,sigma_v);
figure;
subplot 311
plot(x);
title('x_t')
axis ([0 1000 -5 5]);
subplot 312
plot(s);
title('s_t')
axis ([0 1000 -5 5]);
subplot 313
plot(y);
title('y_t')
axis ([0 1000 -5 5]);
%%
frequency_range = (0:pi/1000:(pi-pi/1000))/(2*pi);
figure;
subplot 522
%%%caculate the cross-covariances for different time lags t=40...80
%%%then estimate the spectrum from the empirical cross-correlation
%%%fianlly plot the result
t = 40;
acs = empirical_cross_autocorr(y,x,t);
[spectrum] = spectrum_estimator(acs,t);
plot (frequency_range,real(spectrum));
title('F_{yx}(\omega) for increasing M')
axis([0 3.3/(2*pi) -5 5]);
```

```
subplot 521
plot([-t:t],acs);
title('Cross ACS');
ylabel('M=40');
axis([-85,85,-1,1])
subplot 524
t = 50;                         %% lag numebrs
acs = empirical_cross_autocorr(y,x,t);
[spectrum] = spectrum_estimator(acs,t);
plot (frequency_range,real(spectrum));
axis([0 3.3/(2*pi) -5 5]);
subplot 523
plot([-t:t],acs);
ylabel('M=50');sigma_v^2*(1+theta^2-2*theta*cos(w));
axis([-85,85,-1,1])
subplot 526
t = 60;                         %% lag numebrs
acs = empirical_cross_autocorr(y,x,t);
[spectrum] = spectrum_estimator(acs,t);
plot (frequency_range,real(spectrum));
axis([0 3.3/(2*pi) -5 5]);
subplot 525
plot([-t:t],acs);
ylabel('M=60');
axis([-85,85,-1,1])
subplot 528
t = 70;                         %% lag numebrs
acs = empirical_cross_autocorr(y,x,t);
[spectrum] = spectrum_estimator(acs,t);
plot (frequency_range,real(spectrum));
axis([0 3.3/(2*pi) -5 5]);
subplot 527
plot([-t:t],acs);
ylabel('M=70');
axis([-85,85,-1,1])
subplot (5,2,10)
t = 80;                         %% lag numebrs
acs = empirical_cross_autocorr(y,x,t);
[spectrum] = spectrum_estimator(acs,t);
plot (frequency_range,real(spectrum));
xlabel('frequency f')
axis([0 3.3/(2*pi) -5 5]);
subplot 529
plot([-t:t],acs);
ylabel('M=80');
axis([-85,85,-1,1])
xlabel('lag r')

%%
frequency_range = (0:pi/1000:(pi-pi/1000))/(2*pi);
figure;
subplot 522
%%%caculate the auto-covariances for different time lags t=40...80
%%%then estimate the spectrum from the empirical cross-correlation
%%%fianlly plot the result
t = 40;                         %% lag numebrs
acs = empirical_autocorr(x,t);
[spectrum] = spectrum_estimator(acs,t);
plot (frequency_range,real(spectrum));
title('F_x(\omega) for increasing M')
axis([0 3.3/(2*pi) -5 5]);
subplot 521
plot([-t:t],acs);
title('ACS');
ylabel('M=40');
axis([-85,85,-1,1])
subplot 524
t = 50;                         %% lag numebrs
acs = empirical_autocorr(x,t);
[spectrum] = spectrum_estimator(acs,t);
plot (frequency_range,real(spectrum));
```

```
axis([0 3.3/(2*pi) -5 5]);
subplot 523
plot([-t:t],acs);
ylabel('M=50');
axis([-85,85,-1,1])
subplot 526
t = 60;                        %% lag numebrs
acs = empirical_autocorr(x,t);
[spectrum] = spectrum_estimator(acs,t);
plot (frequency_range,real(spectrum));
axis([0 3.3/(2*pi) -5 5]);
subplot 525
plot([-t:t],acs);
ylabel('M=60');
axis([-85,85,-1,1])
subplot 528
t = 70;                        %% lag numebrs
acs = empirical_autocorr(x,t);
[spectrum] = spectrum_estimator(acs,t);
plot (frequency_range,real(spectrum));
axis([0 3.3/(2*pi) -5 5]);
subplot 527
plot([-t:t],acs);
ylabel('M=70');
axis([-85,85,-1,1])
subplot (5,2,10)
t = 80;                        %% lag numebrs
acs = empirical_autocorr(x,t);
[spectrum] = spectrum_estimator(acs,t);
plot (frequency_range,real(spectrum));
xlabel('frequency f')
axis([0 3.3/(2*pi) -5 5]);
subplot 529
plot([-t:t],acs);
ylabel('M=80');
axis([-85,85,-1,1])
xlabel('lag r')


%%
frequency_range = (0:pi/1000:(pi-pi/1000))/(2*pi);
%%%plot original F_x for f=0...w/2pi
F_x = zeros(1,1000);
w = 0;
for i = 1:1000
    F_x(i) = sigma_v^2*(1+theta^2-2*theta*cos(w));
    w = w+pi/1000;
end
figure;
plot (frequency_range,real(F_x),'--');
hold on
%%%plot the estimates of spectrum F_x for lag t= 10...40
for t = 10:10:40
    acs = empirical_autocorr(x,t);
    [spectrum] = spectrum_estimator(acs,t);
    plot (frequency_range,real(spectrum));
end
legend('True','M=10','M=20','M=30','M=40');
xlabel('frequency f');
title('Estimates of spectrum F_x');
axis([0 0.5 0 6])
hold off
%%
frequency_range = (0:pi/1000:(pi-pi/1000))/(2*pi);
F_yx = zeros(1,1000);
w = 0;
%%%plot original F_yx for f=0...w/2pi
for i = 1:1000
    F_yx(i) = (b/(1-a^2*exp(-2j*w)))*sigma_v^2*(1+theta^2-2*theta*cos(w));
    w = w+pi/1000;
end
```

```
figure;
plot (frequency_range,real(F_yx),'--');
hold on
%%%plot the estimates of spectrum F_yx for lag t= 10...40
for t = 10:10:40
    acs = empirical_cross_autocorr(y,x,t);
    [spectrum] = spectrum_estimator(acs,t);
    plot (frequency_range,real(spectrum));
end
legend('True','M=10','M=20','M=30','M=40');
xlabel('frequency f');
title('Estimates of cross spectrum F_{yx}');
axis([0 0.5 0 6])
hold off
%%
frequency_range = (0:pi/1000:(pi-pi/1000))/(2*pi);
mag_h = zeros(1,1000);
w = 0;
%%%plot original transfer function h for f=0...w/2pi
for i = 1:1000
    mag_h(i) = b/(1-a^2*exp(-2j*w));
    w = w+pi/1000;
end

figure;
plot (frequency_range,abs(mag_h),'--');
hold on
%%%plot the estimates of transfer function h for lag t= 10...40
for t = 10:10:40
    acs = empirical_autocorr(x,t);
    [spectrum_x] = spectrum_estimator(acs,t);
    acs = empirical_cross_autocorr(y,x,t);
    [spectrum_yx] = spectrum_estimator(acs,t);
    estimated_h = spectrum_yx./spectrum_x;
    plot (frequency_range,abs(estimated_h));
end
legend('True','M=10','M=20','M=30','M=40');
xlabel('frequency f');
title('Estimates of transfer function h');
axis([0 0.5 0 6])
hold off
```

Q2

wilson.m

```
%%%input:
% c_star ––> MA covariances order ––>system order iteration ––> iteration
% theta ––> MA coefficients
function [theta] = wilson(c_star,order,iteration)

    theta = zeros(1,order+1).';
    theta(1) = 1;
    c = zeros(order+1,1);
    T_L = zeros(order+1,order+1);
    T_R = zeros(order+1,order+1);
    %%%caculate the covariances
    for counter = 1:iteration
        for r = 1:(order+1)
            cov_index = r–1;                %%%rejust the index back to zero
            my_sum = 0;                     %%%this rejustment is caused by matlab is 1 ...
                index
                for k = 1:(order–cov_index+1)
                    my_sum = my_sum + (theta(k))*(theta(k+cov_index));
                end
            c(r) = my_sum;
        end
     a = [theta(1),zeros(1,length(theta)–1)];
     b = theta;
     T_R = toeplitz(a,b);                   %%construct toeplitz TR
     a =  [theta(end),zeros(1,length(theta)–1)];
     T_L = hankel(b,a);                     %%construct hankel TL
     theta = (T_L+T_R)\(c+c_star);
    end
end
```

AR_MA_simulation.m

```
function [y,s] = AR_MA_simulation(phi,theta,lamda,T)
mu_e  = 0;                              %mean zero
eps   = normrnd(mu_e, lamda, T, 1);     %creating white noises eps
v_ = normrnd(mu_e, 1, T, 1);            %creating whilte noise v

NUM_1 = [1,0–theta];                    %x_t = (1–theta*z^–1) v_
DEN_1 = 1;
n = filter (NUM_1,DEN_1,v_);            %filter the signal to get n

NUM_2 = 1;                              %s_t = 1/(1–phi*z^–1) esp
DEN_2 = [1,–phi];
s = filter(NUM_2,DEN_2,eps);             %filter the signal to get s

y = s+n;                                 %final output y
end
```

q2.m

```
%%(a)
theta1 = 0.9;
theta2 = –0.9;
sigma=1;
iteration = 10;
order = 1;
theta_1 = zeros(order+1,iteration);
theta_2 = zeros(order+1,iteration);
%%% caculate the wilson for two sets of coefficients
%%% and run them for numbers of iteration and store the result into the
%%% matrix, finally we plot the row of matrix against iterations.
for k = 1:iteration+1
```

```
    theta_1(:,k)=wilson([1+theta1^2;-theta1],order,k-1);
    theta_2(:,k)=wilson([1+theta2^2;-theta2],order,k-1);
end
figure;
plot([0:10],theta_1(1,:),'o--');
hold on
plot ([0:10],theta_1(2,:),'x--');
legend('\theta_1','\theta_2')
xlabel('iterations k')
axis([0 10 -1 1.5]);
title('\theta =-0.9')
hold off

figure;
hold on
plot([0:10],theta_2(1,:),'o--');
plot ([0:10],theta_2(2,:),'x--');
legend('\theta_1','\theta_2')
xlabel('iterations k')
axis([0 10 -1 1.5]);
title('\theta = 0.9')
hold off


%%
%%%(b)
%%%construct MA covariances r0 r1 r2 caculated in (ii)
phi = -0.6;theta = 0.8;lamda = 2;
r0 = lamda+1+(theta+phi)^2+theta^2*phi^2;
r1= -((theta+phi)+(theta+phi)*theta*phi);
r2 = theta*phi;
iteration = 10;
theta_1 = zeros(2+1,iteration);
%%%run for numbers of iteration to perform spectral factorization for
%%%denominator
for k = 1:iteration+1
    theta_1(:,k)=wilson([r0;r1;r2],2,k-1);
end
%%%take steady value as the best fit coefficients for forward and backward
%%%filter.
theta_best = theta_1(:,end);
figure;
plot([0:10],theta_1(1,:),'o--');
hold on
plot ([0:10],theta_1(2,:),'o--');
plot ([0:10],theta_1(3,:),'o--');
legend('\theta_1','\theta_2','\theta_3');
xlabel('iterations k')
title('\phi = -0.6, \theta =0.8,\lambda =2')
hold off
NUM = sqrt(lamda);
DEN = theta_best;
%%
%%(c)
phi = -0.6;
theta = 0.8;
lamda = 2;
T=100;
%%%simulate the AR+MA peocess
[y,s] = AR_MA_simulation(phi,theta,lamda,T);
%%%construct filter caculated from (b)
NUM = sqrt(lamda)/1.7871;
DEN = [1,-0.0383,-0.1503];
%%%foward filtering
output_1 = filter(NUM,DEN,y);
%%%backward filtering
s_est = filter(NUM,DEN,flip(output_1));
%%%flip the time series back.
s_est=flip(s_est);
 figure;
 plot (s_est);
```

```
hold on
plot (s-s_est);
legend("s(estimated)",'error');
xlabel('Time lags(n)')
title('joint plot of s(estimated) and error')
```