Advanced Signal Proccessing Techinques-Exercise 3

Kitsios Konstantinos

April 30, 2020

In this 3rd assignment we will confirm the validity of Giannakis' formula using MATLAB. This formula provides a way to calculate the transfer function of a MA system if the order of the system q is known a priori. We will use a MA system of order $q = 5$ .Initially, we will create the 2 signals:

- $v[k]$: a random signal from exponential distribution with length $N = 2048$
- $x[k]$: the output of the MA system with coefficients $[1.0\, 0.93\, 0.85\, 0.72\, 0.59 - 0.1]$ when the input is $v[k]$

MATLAB code was written for the validations of the experiments below. The full code exists both at the .zip file and at the end of this document. The validation of the formula will take place according to the following steps

1. First, we will validate the non-Gaussian nature of $v[k]$. As we stated above, it comes from an exponential distribution which makes it non-Gaussian. Therefore, *the skewness* must be non zero. The value we calculated from MATLAB is 2.006777 which is significantly different than zero.

2. In the second step, we will calculate and plot the third order cumulants of $x[k]$ at the interval $(-\tau_1 : 0 : \tau_1) = (-\tau_2 : 0 : \tau_2) = (-20 : 0 : 20)$using the indirect methold. The library *bispec3cum.m* from the MATLAB file exchange was used for the calculations. The generated plot can be seen in figure 1

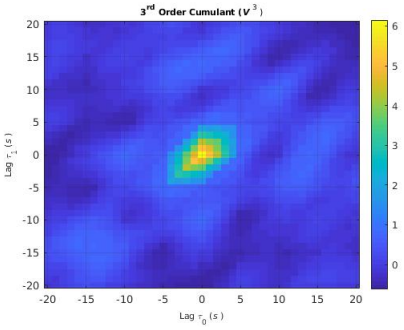
3. Next, having calculated the 3rd order cumulant, we will apply Giannakis' formula in order to compute the MA system transfer function

$$\hat{h}[k] = \frac{c_3^x(q,k)}{c_3^x(q,0)}, k = 0, 1, ..., q$$

$$\hat{h}[k] = 0, k > q$$

4. As we stated at the begining, in order for the formula to be applied we must know the order q of the MA system. This is not always the case so in this step we will assume that we miscalculated the order q. We

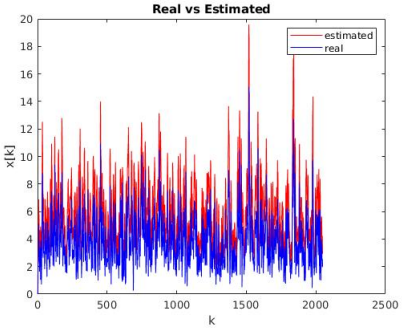
Figure 1:



will examine two cases: $q_{sub} = q - 2 = 3$ and $q_{sup} = q + 3 = 8$. For each case, we computed with Giannakis' formula the impulse responses $\hat{h}_{sub}$ and $\hat{h}_{sup}$. The comparison versus the actual q will take place in next steps.

5. In this step, we will estimate the output of the MA system x_{est} using the impulse response from Giannakis' formula. As a measure of similarity with the original one, we will calculate the NRMSE value between x_{est} and x. This value was found with MATLAB to be $NRMSE = 0.151344$. We see that it is a relatively low value(much closer to 0 than to 1) so we could deduce that the approximation of the impulse response with Giannakis' formula works well if we know the order q and if we have no noise at the output. The plot of x_{est} and x is shown at figure 2.

Figure 2:



6. Here, we will repeat the proccess of 5. but using $\hat{h}_{sub}$ and $\hat{h}_{sup}$ as impulse responses. The NRMSE for the 2 cases is $NRMSE_{sub} = 0.129251$ and $NRMSE_{sup} = 1.697673$. Here we see something interesting: $NRMSE_{sub}$is better than $NRMSE$ from the original impulse function. This means that Giannakis' formula performs slightly better if we pass as a parameter the order $q = 3$ while the actual MA system has $q = 5$. This could be hap-

pening because the formula works better with systems with lower order maybe. That would also explain the unordinarily high $NRMSE_{sup}$ which is greater than one, meaning that the prediction is totally wrong. This is obvious from the plots below where x and x_{sup} are totally different signals. Figure 3 shows $x\,vs\,x_{sub}$ and Figure 4 shows $x\,vs\,x_{sup}$

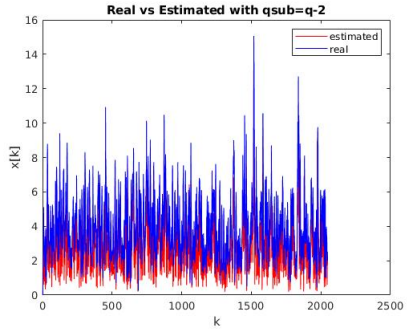


Figure 3:

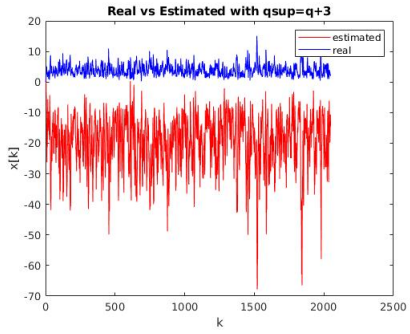
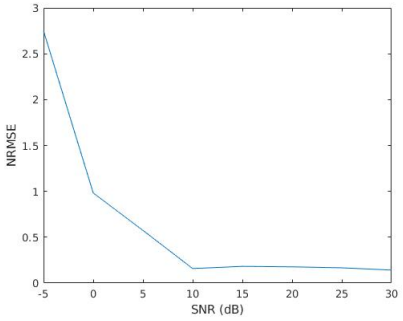


Figure 4:

7. Finally, we will examine how the formula works if we add noise to the signal output. To achieve this, we will not plug into the formula the time series $x[k]$ but the contaminated with noise $y_i[k] = x[k] + n_i[k]$, $i = 1, 2, ..., 8$. Each n_i is a white Gaussian noise such as the SNR will be $[30, 25, 20, 15, 10, 5, 0, -5]\,dB$ for the corresponding $i = 1, ..., 8$. The NRMSE for each i is computed and saved at a vector to be plotted against the corresponding SNR values. The plot can be seen below at Figure 5. It is obvious that for high values of SNR($[10 - 30]$) the NRMSE is not affected very much. This means that Giannakis' formula has an important stability regarding noise and can be used in noisy environments if the noise stays into legit levels. Of course when the SNR becomes smaller the accuracy of the formula also decreases and for $SNR = -5$(i.e noise contributes more than the acutal signal) we see that $NRMSE > 2.5$ which is a really bad

prediction. Even for $SNR = 0$ (i.e. noise contributes equal with the actual signal) we see that $NRMSE \sim 1$ which also indicates poor performance.

Figure 5:



Discussion

With the above analysis we confirmed Giannakis' formula using MATLAB code. The formula provides a method to calculate the impulse response of a MA system of order q using the 3rd order cumulants of a time series produced by the system. We've seen that the formula works well in noise-free environments when the order q of the MA system known a priori. This is indicated by the $NRMSE$ value of around 0.15 of the original signal versus the one predicted using the impulse response from Giannakis' formula.

What is also interesting is that when we do not know the order q and we miscalculated it for let's say $q_{sub} = q - 2$, Giannakis' formula works surprisingly well, even better than the original q value! On the other hand, when we miscalculate q as $q_{sup} = q + 3$ then the results are pretty dissapointing with the $NRMSE$ value being around 1.6 . This may be an indicator that the formula works better with low order systems. Having that said, we can deduce that if not sure about the acutal order of the system, lower predicted values should in general be preffered over higher predicted values if no other indicator exists in favor of high order values. This may significantly improve performance

Finally, we tested Giannakis' formula in a noisy environment with different SNR values. It is worth mentioning that the formula is robust to noise and produces very good results when the SNR values are realtively high, i.e. $SNR = 30 - 10dB$. For $SNR = 5dB$ the calculations seem to get worse while for $SNR = 0$(same amount of signal as noise) the $NRMSE$ value is near 1 which indicates a rather bad behaviour. If we keep adding more noise the calculations keep getting even worse, for example for $SNR = -5dB$(more noise than signal) the $NRMSE$ value explodes over 2.5. However, the robustness of the formula when the noise is kept in relatively low levels is remarkable.

Code

Below is the code for my main script *ex3.m*. The *bispec3cum.m* file I used for 3rd cumulant calculations from MATLAB file exchange site is included in the *.zip*

ex3.m

```
%%% Advanced Signal Proccessing Techniques - Exercise 3 - 2020
%%% Author: Kitsios Konstantinos 9182
%%% Validation of Giannakis' formula
N = 2048;
q = 5;
b = [1 0.93 0.85 0.72 0.59 -0.1];
rng(0)
v = exprnd(1, [1 N]);
x = zeros(1, N);
for i=1:N
for j=1:q+1
if i-j >= 1
x(i) = x(i) + b(j)*v(i-j);
end
end
end
muX = mean(x);
stdX = std(x);
muV = mean(v);
stdV = std(v);
%% 1. Estimage skewness
gamma3V = sum((v-muV).^3)/((N-1)*stdV^3);
fprintf("Skewness value is: %f which is different than zero so we deduce that V[k] is not Gaussian\n", gamma3V);
%% 2. 3rd order cumulant
L3 = 20; % number of lags
% window=0 uses the parzen window
figure(1)
[~, ~, cum, lag] = bisp3cum(x, 1, L3, 'none');
%% 3. Gianakis' formula for h[k]
% Compute h_hat for 10 elements. Only the q(=6) first will be nonzerosub
% It could be any other size >=6 with the same results below
h_hat = zeros(10, 1);
for k=1:q+1
h_hat(k) = cum(L3 + 1 + q, L3 + k)/cum(L3 + 1 + q, L3 + 1);
end
%% 4. Sub-estimation & Sup-estimation
q_sub = q-2;
h_sub = zeros(10, 1);
for k=1:q_sub
```

```
h_sub(k) = cum(L3 + 1 + q_sub, L3 + k)/cum(L3 + 1 + q_sub, L3 + 1);
end
q_sup = q+3;
h_sup = zeros(10, 1);
for k=1:q_sup
h_sup(k) = cum(L3 + 1 + q_sup, L3 + k)/cum(L3 + 1 +q_sup, L3 + 1);
end
%% 5. Comparison between estimation and original
x_est = conv(h_hat, v);
x_est = x_est(1:N);
figure(2);
plot(x_est, 'r');
hold on;
plot(x, 'b')
title('Real vs Estimated')
xlabel('k')
ylabel('x[k]')
legend('estimated', 'real')
rmse = sqrt( sum( (x_est - x).^2 )/N );
nrmse = rmse/(max(x) - min(x));
fprintf("NRMSE: %f\n", nrmse);
%% 6. Comparison with sub-estimation and sup-estimation
x_est_sub = conv(h_sub, v);
x_est_sub = x_est_sub(1:N);
figure(3);
plot(x_est_sub, 'r');
hold on;
plot(x, 'b');
title('Real vs Estimated with qsub=q-2')
xlabel('k')
ylabel('x[k]')
legend('estimated', 'real')
rmse = sqrt( sum( (x_est_sub - x).^2 )/N );
nrmse = rmse/(max(x) - min(x));
fprintf("NRMSE for sub-estimation with qsub=q-2: %f\n", nrmse);
x_est_sup = conv(h_sup, v);
x_est_sup = x_est_sup(1:N);
figure(4);
plot(x_est_sup, 'r');
hold on;
plot(x, 'b');
title('Real vs Estimated with qsup=q+3')
xlabel('k')
ylabel('x[k]')
legend('estimated', 'real')
rmse = sqrt( sum( (x_est_sup - x).^2 )/N );
```

```
nrmse = rmse/(max(x) - min(x));
fprintf("NRMSE for sup-estimation with qsup=q+3: %f\n", nrmse);
%% 7. SNR variations
NN = 8; % Number of SNR values to try
nrmseV = zeros(NN, 1); % Vector to hold NRMSE for each SNR value
snrV = 30:-5:-5; % Victor of SNR values
for i=1:NN
y = awgn(x, snrV(i), 'measured');
% 7.2. 3rd order cumulant
L3 = 20; % number of lags
% window=0 uses the parzen window
[~, ~, cumSNR, lagSNR] = bisp3cum(y, 1, L3, 'none');
% 7.3. Gianakis' formula for h[k]
% Compute h_hat for 10 elements. Only the q(=6) first will be nonzerosub
% It could be any other size >=6 with the same results below
h_hat_snr = zeros(10, 1);
for k=1:q+1
h_hat_snr(k) = cumSNR(L3 + 1 + q, L3 + k)/cumSNR(L3 + 1 + q, L3
+ 1);
end
x_est_snr = conv(h_hat_snr, v);
x_est_snr = x_est_snr(1:N);
rmse = sqrt( sum( (x_est_snr - x).^2 )/N );
nrmse = rmse/(max(x) - min(x));
nrmseV(i) = nrmse;
end
figure(5)
plot(snrV, nrmseV)
xlabel('SNR (dB)')
ylabel('NRMSE')
```