Time Series Coursework

Alix Vermeulen (CID: 01564896)

December 2020

Question 1

(a) The following code is a function that evaluates the parametric form of the spectral density function (sdf) for an AR(p) process on a set of frequencies. The inputs in this function are f, a vector of frequencies at which we want to evaluate the sdf, $phis$, a vector of the $\phi_{1,p}$ to $\phi_{p,p}$ parameters and $sigma2$, the variance of the white noise term in an $AR(p)$ process. The spectral density function for an AR(p) is given by:

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

```
function S = S_AR(f, phis, sigma2)
N = length(f);
p = length(phis);
for i = 1:N
    sum_p(i) = sum(phis(1:p).*exp(-j*2*pi*f(i)*(1:p)));
end
S(i) = sigma2/abs(1-sum_p(i))^2;
end
```

(b) The following code is a function that simulates a Gaussian AR(2) process of length N. The inputs are $phis$, the vector $[\phi_{1,2}, \phi_{2,2}]$, $sigma2$, a scalar for the variance σ_ϵ^2 of the white noise and N, the length of process to be simulated. We discard the first 100 values and keep values 101 to $100 + N$, which returns X, a vector a vector of length N. An AR(2) process is of the form:

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

```
function X = AR2_sim(phis, sigma2, N)
X = zeros(1,100+N); % create an empty row vector of length 100+N to store time series
epsilon = normrnd(0,sqrt(sigma2),[1,100+N]); % normrnd generates normal random numbers
    for t = 3:N+100
        X(t) = phis(1)*X(t-1) + phis(2)*X(t-2) + epsilon(t);
    end
X = X(101:end); % discard first 100 value
end
```

(c) The following code is a function that computes the estimate of autocovariance $\hat{s}_\tau^{(p)}$. The inputs are the time series X, and τ, the designated values at which we want to evaluate $\hat{s}_\tau^{(p)}$. The output, *shat* is a vector of values of the autocovariance sequence estimate evaluated at the elements of τ. The estimate of the autocovariance $\hat{s}_\tau^{(p)}$ is given by:

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

where $\tau = 0, \pm 1, \pm 2, \pm 3..., \pm(N-1)$. Note also that the mean of the process is taken to be 0.

```
function s_hat = acvs_hat(X, tau)
N = length(X);
P = length(tau);
s_hat = zeros(P,1); % storing values
for i = 1:P % loop over each value of tau
    s_hat(i) = (1/N)*sum((X(1:N-abs(tau(i))).*(X(abs(tau(i))+1:N))));
end
end
```

Question 2

(a) The following is a function to compute the periodogram at the Fourier frequencies of a time series X. The output *sphat*, is a vector with all the periodogram values. The periodogram is given by:

$$\hat{S}^{(p)}(f) = \frac{1}{N} \left| \sum_{t=1}^{N} X_t e^{-i2\pi f t} \right|^2$$

We use the inbuilt fft function because the sum inside the absolute value in $\hat{S}^{(p)}(f)$ is simply the Fourier Transform of the time series data X.

```
function sp_hat = periodogram(X)
N = length(X);
sp_hat = (1/N)*abs(fft(X)).^2;
end
```

The following is a function that computes the direct spectral estimate at the Fourier frequencies using the Hanning taper, the input is a time series X stored as a vector and the output *dse* are the estimates. The direct spectral estimator is given by:

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

where h_t is known as the data taper. Here we will use the Hanning taper:

$$h_t = \frac{1}{N} \left[\frac{8}{3(N+1)} \right]^{1/2} \left[1 - cos\left(\frac{2\pi t}{N+1} \right) \right], t = 1, ..., N$$

```matlab
function dse = direct(X)
N = length(X);
ht = zeros(1,N);
for t = 1:N
  ht(t) = (1/2)*(8/(3*(N-1)))^(1/2)*(1-cos((2*pi*t)/(N+1))); % Hanning taper
end
dse = abs(fft(ht.*X)).^2;
```

(b) (A) & (B) We will now simulate 10000 realizations, each of length $N = 16$ of an AR(2) process. Here, the function will output the values of the bias for the direct spectral estimate and the bias of the periodogram. In line 28 of the code below, we use such indexing because periodogram(i) corresponds to the frequency of $\frac{i-1}{N}$ which we want equal to $\frac{1}{8}$. Hence,

$$\frac{i-1}{N} = \frac{1}{8}$$

Therefore $i = \frac{N}{8} + 1$. And if we want the frequencies $\frac{2}{8}$ and $\frac{3}{8}$ we must multiply N by 2 and 3.

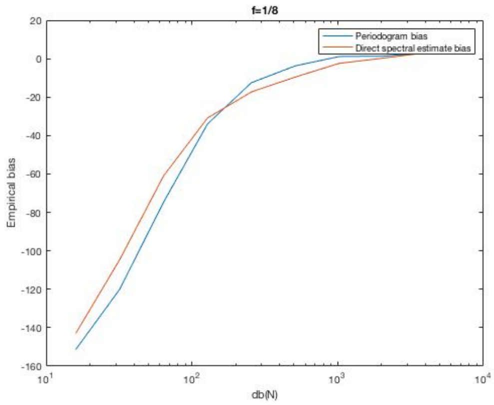
```matlab
1  function [bias_direct, bias_periodogram] = qu2(N)
2  % set values to obtain phi vector
3  f_dash = 1/8;
4  r = 0.95;
5  sigma2 = 1;
6  M = 10000;
7  phis = [2*r*cos(2*pi*f_dash),-r^2];
8
9  % Create empty Nx10000 matrices to store time series simulations,
10 % periodogram values and direct spectral estimate values
11 T = zeros(N,M);
12 P = zeros(N,M);
13 D = zeros(N,M);
14 for i = 1:M % loop over the colums
15     T(:,i) = AR2_sim(phis,sigma2,N);
16     P(:,i) = periodogram(T(:,i));
17     D(:,i) = direct(transpose(T(:,i)));
18 end
19
20 % Create a 3x10000 matrix where we will store value of the periodogram and
21 % dsf values at the relevant frequencies, each row will be one frequency
22 % (hence the 3 rows)
23 P_at_freq = zeros(3,M);
24 D_at_freq = zeros(3,M);
25
26 % We take the following indexing below because
27 for i = 1:3
28     P_at_freq(i,:) = P(((i*N)/8)+1,:);
29     D_at_freq(i,:) = D(((i*N)/8)+1,:);
30 end
31
32 % Step 1: find the true spectral density function using S_AR
33 f = [1/8, 2/8, 2/8];
```

```
true_sdf = S_AR(f, phis, sigma2);

% Step 2: take the sample mean of the periodogram and direct spectral
% estimate at each frequency
sample_periodogram = zeros(3,1);
sample_direct = zeros(3,1);
for i = 1:3
    sample_periodogram(i,:) = sum(P_at_freq(i,:))/M;
    sample_direct(i,:) = sum(D_at_freq(i,:))/M;
end
% Step 2: compute bias
bias_periodogram = sample_periodogram - transpose(true_sdf);
bias_direct = sample_direct - transpose(true_sdf);
end
```

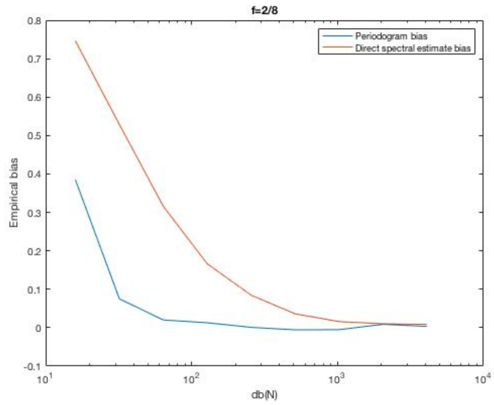
(C) & (D) Below is my code for step (A) and (B) for values of N = 16, 32, 64, 128, 256, 512, 1024, 2048 and 4096. To compare the two spectral estimators for different values of N, I will plot the bias of these two estimators.

```
bias_periodogram = zeros(3,9);
bias_direct = zeros(3,9);
N = zeros(1,9);

for i = 4:12
    N(i-3) = 2^i;
    [B_P, B_D] = qu2(2^i); % store bias of periodogram and sde
    bias_periodogram(:,i-3) = B_P; % bias of periodogram
    bias_direct(:,i-3) = B_D; % bias of sde
end

figure(1)
semilogx(N, bias_periodogram(1,:));
hold on;
semilogx(N, bias_direct(1,:));
hold off;
title('f=1/8')
xlabel('db(N)')
ylabel('Empirical bias')
legend('Periodogram bias','Direct spectral estimate bias')

figure(2)
semilogx(N, bias_periodogram(2,:));
hold on;
semilogx(N, bias_direct(2,:));
hold off;
title('f=2/8')
xlabel('db(N)')
ylabel('Empirical bias')
legend('Periodogram bias','Direct spectral estimate bias')

figure(3)
semilogx(N, bias_periodogram(3,:));
```

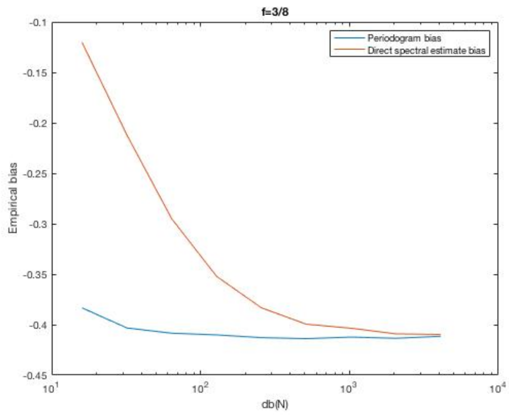
```
hold on;
semilogx(N, bias_direct(3,:));
hold off;
title('f=3/8')
xlabel('db(N)')
ylabel('Empirical bias')
legend('Periodogram bias','Direct spectral estimate bias')
```



(a)



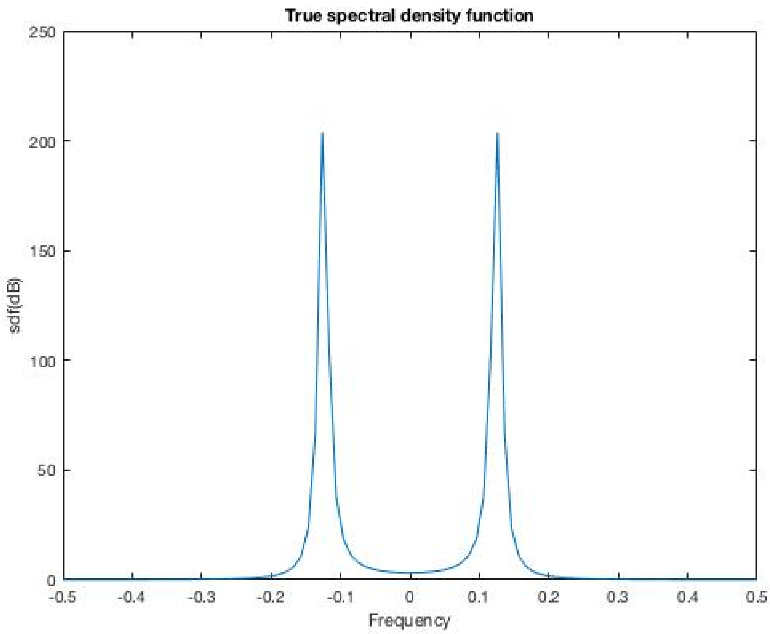
(b)



(c)

Figure 1: Bias of periodogram and direct spectral estimate for (a) f=1/8, (b) f=2/8 and (c) f=3/8.

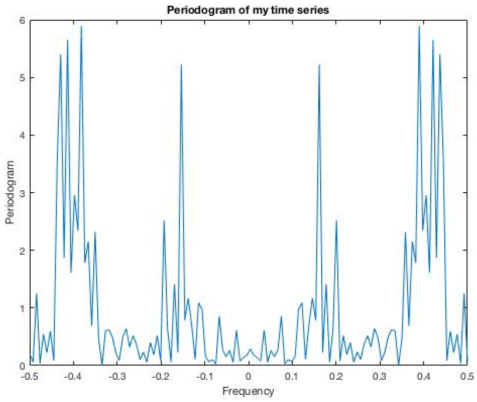
(c) As we can clearly see from the graphs, as N increases, the bias tends towards 0 which is what we expect from a good estimator.



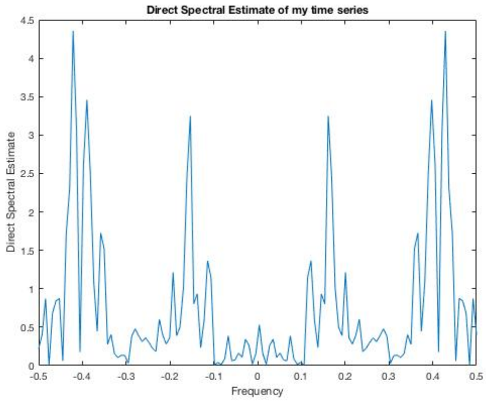
Question 3

(a) Below, we compute the periodogram and the direct spectral estimate using the Hanning taper from Question 2(a) for our time series data. Here is my code to do so:

```
figure(1)
f = linspace(-0.5,0.5,length(tsdata));
x = periodogram(tsdata);
plot(f,x);
xlabel('Frequency')
ylabel('Periodogram')
title('Periodogram of my time series')

figure(2)
f = linspace(-0.5,0.5,length(tsdata));
x = direct(tsdata);
plot(f,x);
xlabel('Frequency')
ylabel('Direct Spectral Estimate')
title('Direct Spectral Estimate of my time series')
```



(a)



(b)

Figure 2: Periodogram (a) and Direct Spectral Estimate (b) of my time series

(b) Here, we will be using 3 methods to fit an $AR(p)$ model. I use lecture notes pages 65 to 76 to do this. Firstly, we have the code for the Yule-Walker method. The output of my function are $phiyw$, the vector of ϕ's estimated and $sigmayw$, the estimated variance term.

```
function [phi_yw, sigma_yw] = yw(X, p)
tau = 0:p;
s_hat = transpose(acvs_hat(X,tau)); % autocovariance terms from s_0 to s_p
gamma_p_hat = s_hat(2:p+1);
cap_gamma_hat = zeros(p,p);
for i = 1:p % creating the Toeplitz matrix
    cap_gamma_hat(i,:) = [flip(s_hat(1:i)),s_hat(2:p-i+1)];
end
phi_yw = inv(cap_gamma_hat)*transpose(gamma_p_hat);
sigma_yw = s_hat(1) - sum(phi_yw(1:p).*transpose(s_hat(2:p+1)));
end
```

Next, we have the forwards Least Squares estimate:

```
function [phi_flsqr,sigma_flsqr] = flsqr(X,p)
N = length(X);
F = zeros(N-p,p);
for i = 1:p % creating the (N-p)xp F matrix
    F(:,i) = X(p-i+1:N-i);
end
X = transpose(X(p+1:N));
phi_flsqr = inv(transpose(F)*F)*transpose(F)*X;
sigma_flsqr = (transpose(X-F*phi_flsqr)*(X-F*phi_flsqr))/(N-2*p);
end
```

And lastly, the approximate maximum likelihood estimate:

```
function [phi_mle, sigma_mle] = mle(X, p)
[phi_flsqr,sigma_flsqr] = flsqr(X,p);
```

```
    N = length(X);
    phi_mle = phi_flsqr;
    sigma_mle = sigma_flsqr*((N-2*p)/(N-p));
end
```

(c) We now compute the AIC. It can be shown that for stationary Gaussian AR processes, AIC $= 2p - Nln(\hat{\sigma}_{\epsilon}^2)$.

```
X =  tsdata;
N = length(X);
AIC_yw = zeros(1,20);
AIC_mle = zeros(1,20);
AIC_lse = zeros(1,20);
tau = 0:20;

for p = 1:20
    [phi_yw, sigma_yw] = yw(X, p);
    AIC_yw(p) = 2*p + N*log(sigma_yw);
    [phi_flsqr, sigma_flsqr] = flsqr(X, p);
    AIC_flsqr(p) = 2*p + N*log(sigma_flsqr);
    [phi_mle, sigma_mle] = mle(X, p);
    AIC_mle(p) = 2*p + N*log(sigma_mle);
end

p = 1:20;
t = table(p',AIC_yw', AIC_flsqr', AIC_mle', 'VariableNames',{'p','AIC_yw', '
    AIC_flsqr', 'AIC_mle'});
```

Below is the table where I store the AIC values for each estimation method.

p	AIC_yw	AIC_flsqr	AIC_mle
1	-15.902	-16.706	-17.718
2	-14.064	-15.867	-17.915
3	-12.101	-12.652	-15.761
4	-44.925	-46.57	-50.767
5	-44.792	-52.749	-58.06
6	-42.801	-49.075	-55.531
7	-41.816	-46.237	-53.865
8	-39.853	-42.079	-50.91
9	-37.957	-37.777	-47.843
10	-36.931	-36.763	-48.098
11	-35.514	-33.376	-46.014
12	-33.963	-30.32	-44.298
13	-32.474	-29.028	-44.383
14	-31.72	-27.912	-44.683
15	-30.311	-24.334	-42.563
16	-28.321	-20.076	-39.808
17	-28.073	-18.07	-39.348
18	-27.203	-15.776	-38.649
19	-25.645	-14.784	-39.301
20	-24.174	-15.61	-41.824

Figure 3: Table of AIC values for each estimation method

(d) In order to select which model fits our data the best, we pick the value for which we have the best AIC, this is the smallest one. For the Yule-Walker method, we get $p = 5$ and for both forward least squares and approximate maximum likelihood, we get $p = 4$. The p+1 estimated parameter values for each method are:

```
[phi_yw, sigma_yw] = yw(tsdata, 4);
[phi_flsqr, sigma_flsqr] = flsqr(tsdata, 5);
[phi_mle, sigma_mle] = mle(tsdata, 5);
```

Parameters	Yule-Walker
$\hat{\sigma}$	0.6613
$\hat{\phi}_{1,4}$	0.3748
$\hat{\phi}_{2,4}$	-0.0622
$\hat{\phi}_{3,4}$	0.1957
$\hat{\phi}_{4,4}$	-0.4881

Parameters	Least Squares	Maximum Likelihood
$\hat{\sigma}$	0.6125	0.5876
$\hat{\phi}_{1,5}$	0.3189	0.3189
$\hat{\phi}_{2,5}$	-0.0174	-0.0174
$\hat{\phi}_{3,5}$	0.2079	0.2079
$\hat{\phi}_{4,5}$	-0.4968	-0.4968
$\hat{\phi}_{5,5}$	-0.1245	-0.1245

(e) We now plot the associated spectral density functions on a single axis for the three associated models. We notice from the Figure 4 below that both the maximum likelihood and least squares methods for estimating parameters overlap. This is because they are similar estimating methods (have the same ϕ parameters and proportional variance). This is why it is very difficult to distinguish them from the graph. The Yule-walker parameters to get the spectral density of an $AR(4)$ process on the other hand is more visible.

```
f = linspace(-0.5,0.5,100);
a = S_AR(f,phi_yw,sigma_yw);
plot(f,a)
xlabel('Frequency')
title('Spectral Density function using estimates from the 3 methods above')

hold on

b = S_AR(f,phi_flsqr,sigma_flsqr);
plot(f,b)
c = S_AR(f,phi_mle,sigma_mle);
plot(f,c)
```

```
legend('Yule-Walker','Least Squares','Max Likelihood')

hold off
```

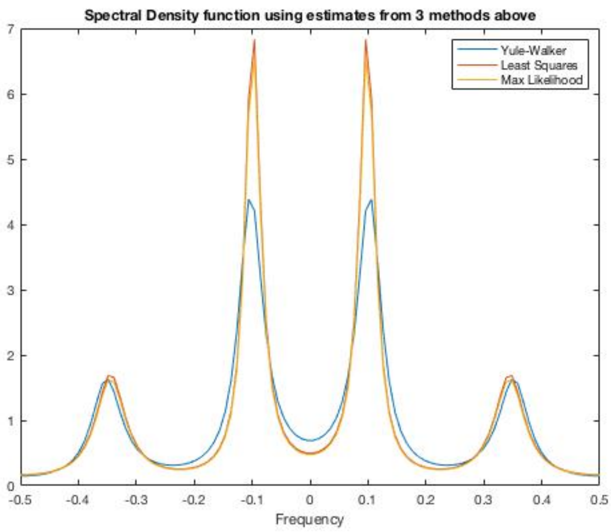


Figure 4: Spectral Density plotted using parameters from Yule-Walker, Least Squares and Maximum Likelihood.

Question 4

In this final part, we assume that we have only observed values $X_1, ..., X_{118}$. We will forecast $X_{119}, ..., X_{128}$ using the selected models and parameter estimates for each of the three methods (YW, LS and ML). Figure 5 and 6 below compare them to the actual values from time point 110 to 128.

```
function Y = forecast(X)

% obtain estimated parameters for each method
[phi_yw, sigma_yw] = yw(X, 4);
[phi_flsqr, sigma_flsqr] = flsqr(X, 5);
[phi_mle, sigma_mle] = mle(X, 5);

% where we will store the forecasted time series
Y_yw = zeros(0,128);
Y_flsqr = zeros(0,128);
Y_mle = zeros(0,128);

for t=119:128
    Y_yw(t) = phi_yw(1)*X(t-1) + phi_yw(2)*X(t-2) + phi_yw(3)*X(t-4) +
   phi_yw(4)*X(t-4);
```

```
    Y_flsqr(t) = phi_flsqr(1)*X(t-1) + phi_flsqr(2)*X(t-2) + phi_flsqr(3)*X(t-4) + phi_flsqr(4)*X(t-4) + phi_flsqr(5)*X(t-5);
    Y_mle(t) = phi_mle(1)*X(t-1) + phi_mle(2)*X(t-2) + phi_mle(3)*X(t-3) + phi_mle(4)*X(t-4) + phi_mle(5)*X(t-5);
end

X = X(110:118);
% forecasted values for t=119 to t=128
Y_yw = Y_yw(119:128);
Y_flsqr = Y_flsqr(119:128);
Y_mle = Y_mle(119:128);

%concatenate to get value from t=110 to t= 128
Y_yw_final = [X,Y_yw];
Y_flsqr_final = [X,Y_flsqr];
Y_mle_final = [X,Y_mle];

t = 110:128;
plot(t, tsdata(110:128));
xlabel('Time');
ylabel('Value of time series');
title('True time series with forecasted data from t=119 onwards')
hold on
a = Y_yw_final;
plot(t,a)
b = Y_flsqr_final;
plot(t,b)
c = Y_mle_final;
plot(t,c)
legend('YW forecast','LS forewcast','ML forecast', 'True time series')
hold off
end
```

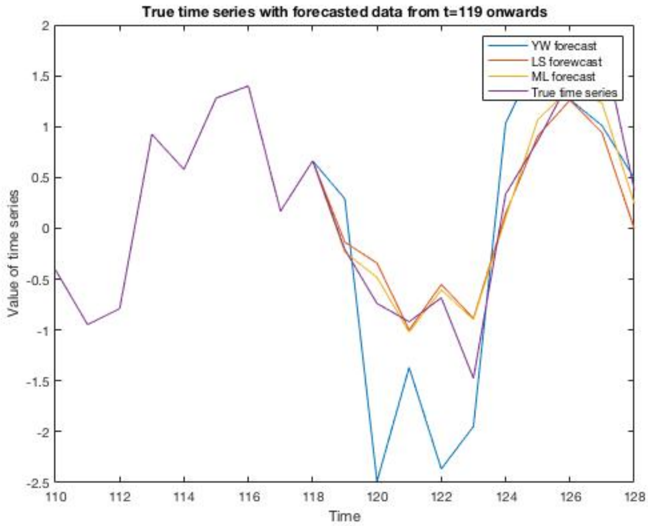


Figure 5: True times series data until t=119 and then forecasted data using the selected models and parameter estimates from Question 3(d).

As above, we can see that the Least Squares and Maximum Likelihood lines are very close because the parameters are very similar.

True X	YW Forecast	LS Forecast	ML Forecast
-0.4076	-0.4076	-0.4076	-0.4076
-0.9464	-0.9464	-0.9464	-0.9464
0.7887	-0.7887	-0.7887	-0.7887
0.9235	0.9235	0.9235	0.9235
0.5786	0.5786	0.5786	0.5786
1.2803	1.2803	1.2803	1.2803
1.3995	1.3995	1.3995	1.3995
0.1679	0.1679	0.1679	0.1679
0.6628	0.6628	0.6628	0.6628
0.2896	-0.1363	-0.2335	-0.2088
-2.4881	-0.3418	-0.4829	-0.7390
-1.3684	-0.9996	-1.0212	-0.9183
-2.3656	-0.5518	-0.6055	-0.6831
-1.9477	-0.8861	-0.8968	-1.4743
1.0337	0.1445	0.1028	0.3356
1.6890	0.9086	1.0687	0.8613
1.2652	1.2602	1.3745	1.4614
1.0088	0.9385	1.2314	1.8513
0.4918	-0.0028	0.2436	0.3799

Figure 6: Data to plot above graph from time point 110 to 128