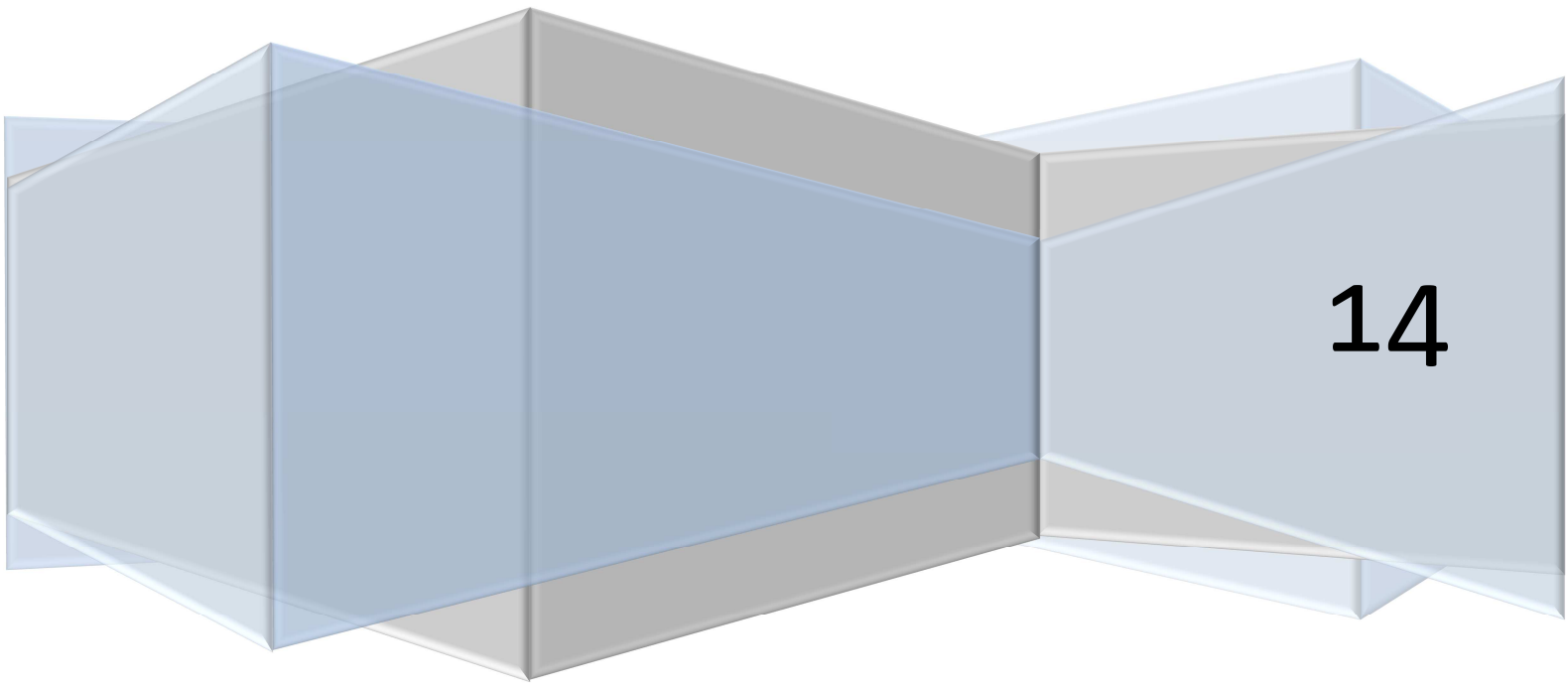
Imperial College

ASP Coursework

Bofeng Li

CID: 00683142, login:bl2111



14

Table of Contents

Coursework 1

1.1 statistical Estimation

In this section, I am going to discover some basic theories about uniform white noise and Gaussian white noise using Matlab. First of all, I have generated a 1000-sample of uniform random variable $X \sim N(0,1)$ which could be regarded as a realisation of uniform white noise with mean of 0.5.

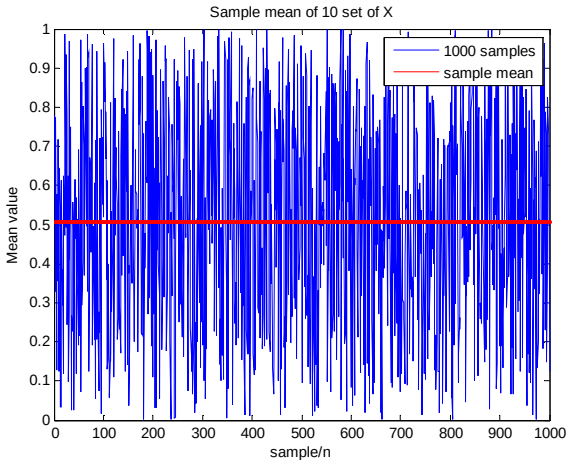


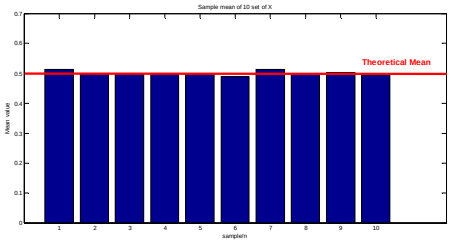
Fig.1 Graph of 1000-sample random signal with sample means

This graph demonstrates the stochastic nature of uniform white noise and it exhibits a degree of uniformity because numbers of samples with value within different intervals of value (for example, intervals of 0-0.1,0.1-0.2.....) are generally similar. Such random signals are also referred to statistically stationary.

1 The theoretical mean of this signal is 0.5 because samples of this signal follows X ~ N(0,1). For uniform distribution, it has a mean of (1+0)/2=0.5. Also, the sample mean has been computed by Matlab which was0.5127. The sample mean was a good estimator since the sample mean was close enough to the theoretical mean.

2 For uniform distribution with parameters of 0 and 1, it has a theoretical standard deviation of $\sqrt{\frac{1}{12}} = 0.2887$. By using Matlab, I also computed the sample standard deviation which was 0.2893. Hence, the sample standard deviation was close enough to the theoretical standard deviation and it was a good estimator of standard deviation.

3 In this part, I created an ensemble of ten 1000-sample realisations of X, and calculate the sample mean and standard deviation of each realisation respectively.



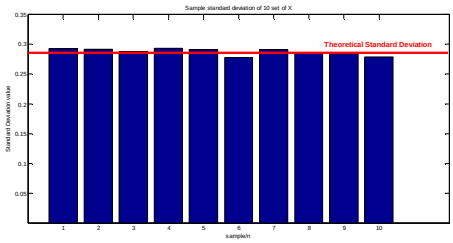


Fig. 2 Graph of sample means and sample standard deviations of ten 1000-sample realisations

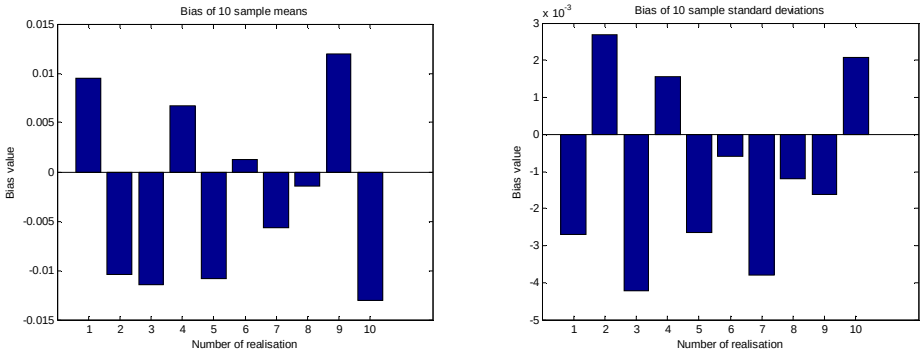


Fig. 3 Graph of bias of 10 sample means and sample standard deviations

These graphs illustrate that the biases of sample standard deviations are tiny and biases of sample means are also relatively small.

4 In this part, I have analysed the Probability Density Function(PDF) of sample using histogram of random signals which is normalised by the number of samples

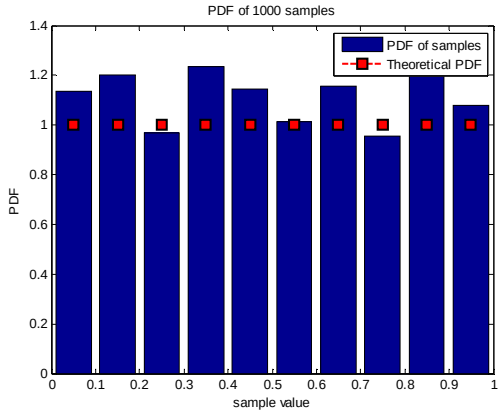


Fig. 4 Graph of histogram of the random signal with histogram bins of 10

This graph shows that the PDF of 1000 samples and the theoretical PDF of uniform distribution *X ~ N(0,1)* are very similar. The difference between theoretical PDF and sample PDF are highly related to number of sample and number of histogram bins.

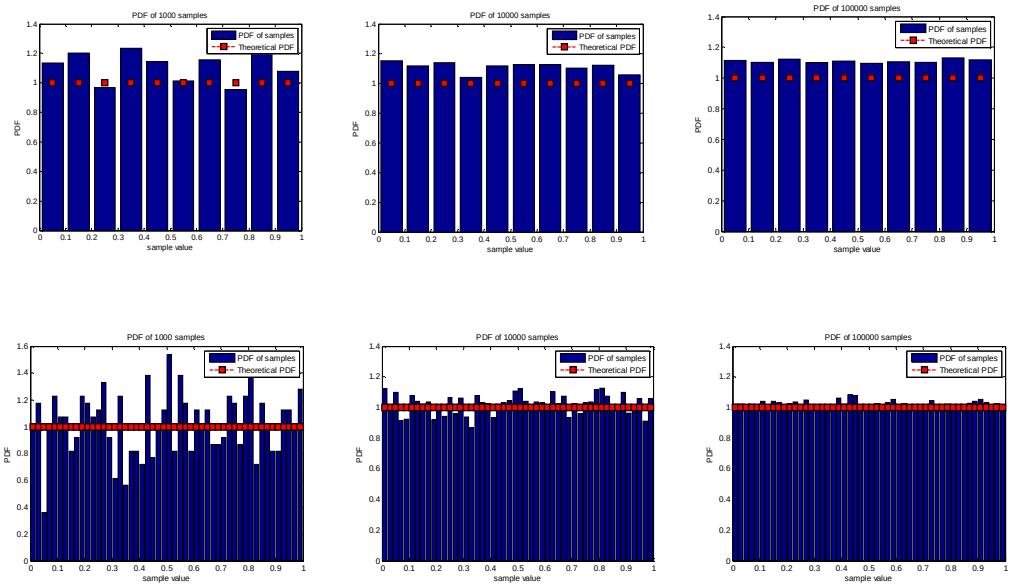


Fig.5upper 3 graphs: graphs of sample PDFs with different numbers of samples(1000 samples, 10000 samples and 100000 samples) and histogram bins of 10. bottom 3 graphs: graphs of sample PDFs with different numbers of samples(1000 samples, 10000 samples and 100000 samples) and histogram bins of 50

The top 3 graphs show that the sample PDF with larger number of samples seems converges more to a constant value but that value is slightly different from theoretical PDF. Comparing the bottom 3 graphs, it is clear that the sample PDF with more histogram bins converges more to the theoretical PDF.

5 I repeat part 1-4 with realisation of Gaussian random variables. The generated Gaussian random signal is showed below.The graph below demonstrates a degree of Gaussian distribution since most samples have value near 0.

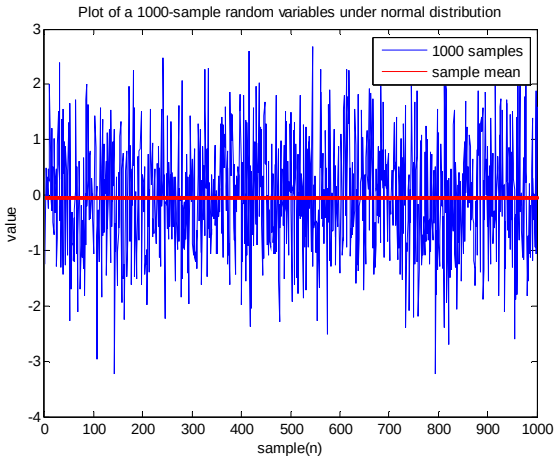


Fig. 6 graph of Gaussian random signal with 1000 samples

5.1 Since the random signal follows a standard Gaussian distribution, the theoretical mean is 0 and the sample mean in the case showed by above graph is 0.0168. Hence sample mean is a good estimator of theoretical mean.

5.2 The theoretical standard deviation of standard Gaussian distribution is 1 and the sample standard deviation in the case showed by above graph is 1.0101. Hence sample standard deviation is a good estimator of theoretical standard deviation.

5.3 The graphs below illustrate the sample means and sample standard deviations of an ensemble of 10 Gaussian realisations.

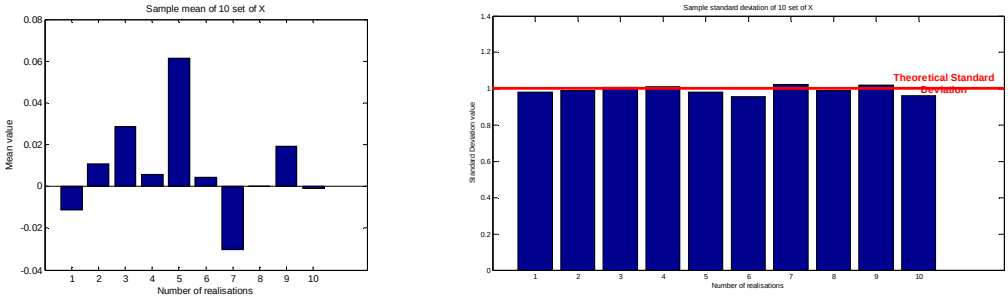


Fig. 7 graph of sample means and sample standard deviations of 10 realisations

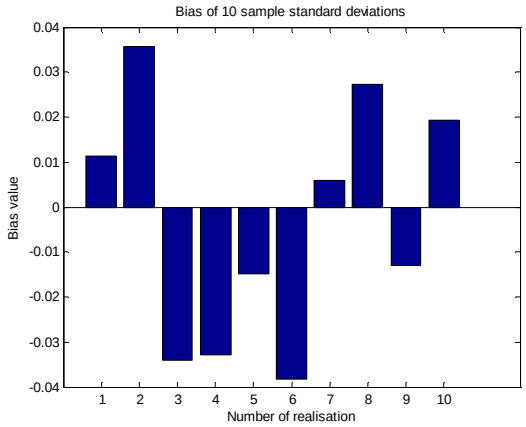


Fig. 8 Graph of biases of sample standard deviations of 10 realisations

The above 3 graphs show that the biases of sample means and sample standard deviations are small. Hence, the sample mean and sample standard deviation are good estimators.

5.4 In this part, I calculated the PDFs of sample by plotting histograms of random signals. Again, the number of samples and number of histogram bins still affect the analysis

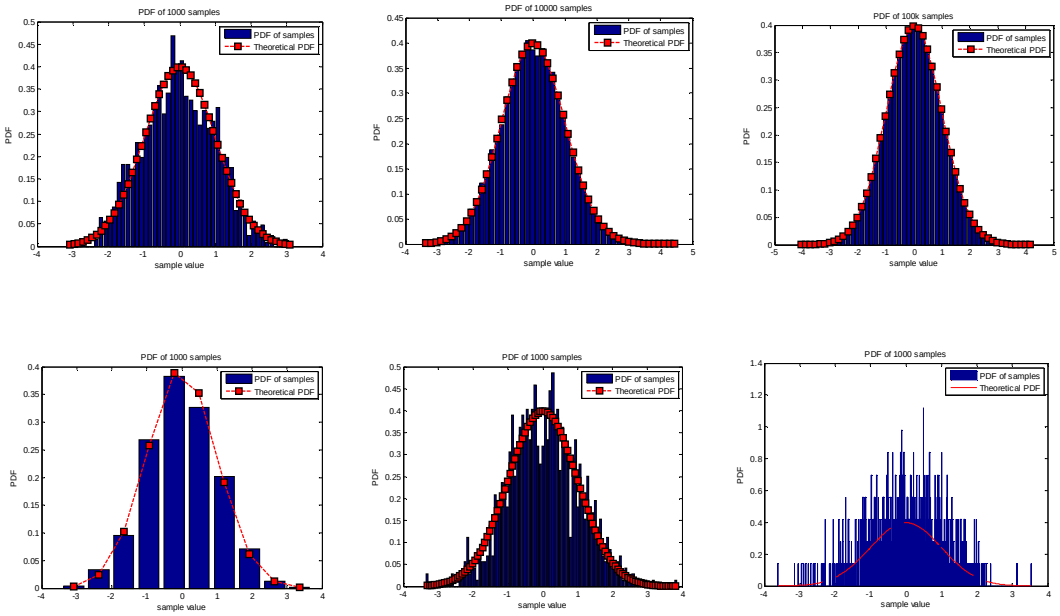


Fig.9 upper 3 graphs: graphs of sample PDFs with different numbers of samples (1000 samples, 10000 samples and 100000 samples) but same histogram bins of 50. Bottom 3 graphs: graphs of sample PDFs with same numbers of samples of 1000 but different histogram bins(10 bins, 100 bins and 1000 bins)

The top 3 graphs show that the sample PDF with larger number of samples seems converges more to theoretical PDF. Comparing the bottom 3 graphs, it is clear that the sample PDF with less histogram bins converges more to the theoretical PDF.

1.2 Stochastic Processes

In this section, I will discover stochastic processes which are an ordered collection of random variables. Given 3 stochastic processes generated by Matlab which gives an ensemble of M realistation of N sample, I could examine some of their features including stationarity, ergodicity, etc.

First process

1. I generated an ensemble of 100 realisations of 100 samples and the graph below shows the ensemble mean and standard deviation. Note that the ensemble mean is increasing and standard deviation is not constant. Hence, I can say that this process is not stationary.

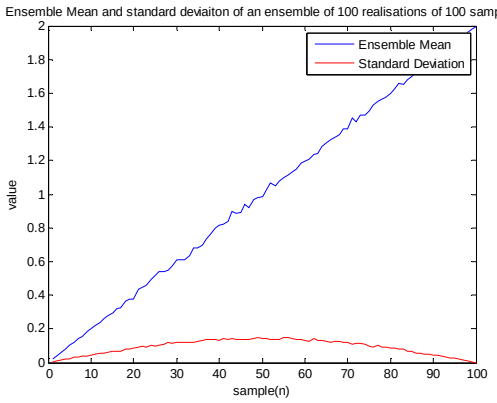


Fig. 10 graph of ensemble mean and standard deviation of an ensemble with 100 realisation of 100 samples

2. After that, I generated 4 realisations of 1000 samples and the graphs below show the means and standard deviations for each realisation. Graphs illustrate that 4 sample means have similar values and values of 4 sample standard deviations are also close to each other. Hence, it can be induced that the theoretical mean can be approximated by sample mean, which means that the first process is ergodic.

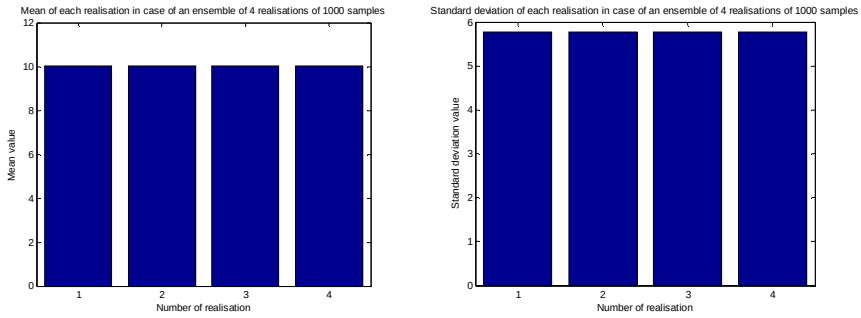


Fig. 11 graphs of sample means and sample standard deviations of 4 realisations

3. In order to write a mathematical description of the first process, I needed to closely look at the code. Mc is an array used to generate a sampled sine wave $5 \times \sin\left(\frac{n\pi}{N}\right)$from 0 to π. Ac is an array used to generate a linear function 0.02n. Also, note that rand() generates random variables *X ~ N(0,1)* with mean of 0.5, so **rand(M,N)-0.5** generates uniform white noise which follow a uniform distribution. Hence the overall mathematics description is

$$v = 0.02n + 5uw(n)\sin\left(\frac{n\pi}{N}\right)$$

where $uw(n)$ refers *to* uniform white noise. Hence the theoretical mean is

$$E(v) = E\left(0.02n + 5uw(n)\sin\left(\frac{n\pi}{N}\right)\right)$$
$$= E(0.02n) + E\left(5uw(n)\sin\left(\frac{n\pi}{N}\right)\right).$$

Since $uw(n)$ and $\sin\left(\frac{n\pi}{N}\right)$ are independent and $E\big(uw(n)\big) = 0$, we have

$$E(v) = E(0.02n) + 5E\big(uw(n)\big)E\left(\sin\left(\frac{n\pi}{N}\right)\right) = E(0.02n)$$

For theoretical variance, we have

$$Var(v) = Var\left(0.02n + 5uw(n)\sin\left(\frac{n\pi}{N}\right)\right)$$

$$= Var(0.02n) + 25Var\left(uw(n)\sin\left(\frac{n\pi}{N}\right)\right)$$

$$= Var(0.02n) + 25E(wn(n)^2)E\left(\sin^2\left(\frac{n\pi}{N}\right)\right)$$

Since $E\left(\sin^2\left(\frac{n\pi}{N}\right)\right) = 0.5$ and $E(wn(n)^2) = Var\big(wn(n)\big) = \frac{1}{12}$, we have

$$Var(v) = Var(0.02n) + 25 \times 0.5 \times Var\big(wn(n)\big)$$

$$Var(v) = Var(0.02n) + 12.5$$

Hence, by calculating $E(0.02n)$ and $Var(v) = Var(0.02n) + 12.5$, I generated theoretical mean and standard deviation.

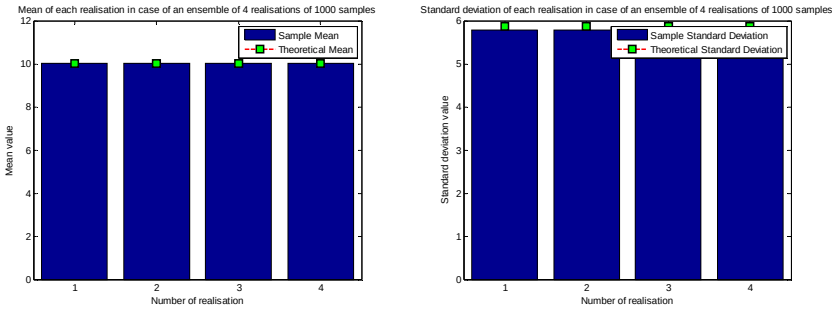


Fig. 12 Left: graphs of sample and theoretical means with sample size of 1000. Right: graph of sample and theoretical standard deviations of 4 realisations with sample size of 1000.

Second process

1.

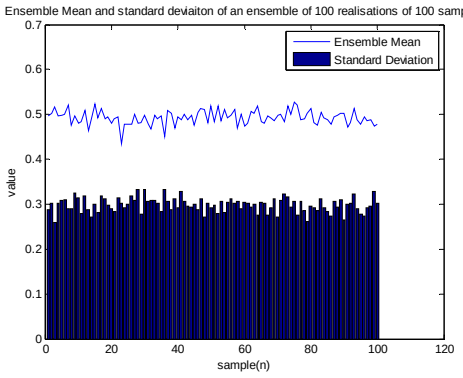


Fig. 13 graph of ensemble mean and standard deviation of an ensemble with 100 realisation of 100 samples

These graphs shows that the ensemble mean is a random noise centered on 0.5 and standard deviation is approximately around 0.25. Hence this process shows some extend of stationary.

2.

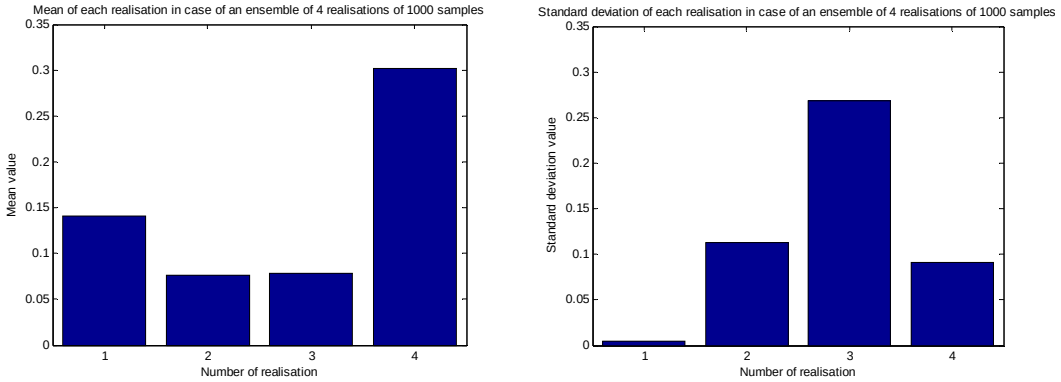


Fig. 14 graphs of sample means and sample standard deviations of 4 realisations

These graphs show that the sample means and sample standard deviations of realisations are completely random and there is not any trend to follow. Hence, it can be induced that 2nd process is non-ergodic.

3. In order to derive mathematical description, I had a look at the code generating 2nd process and assuming M=1.In this process, both **Ar** and **Mr** are generating uniform random variable *X ~ N(0,1),* which can be regarded as $\mathrm{uw(n)} + 0.5$. Hence

$$\mathrm{v} = \mathrm{uw(n)}(\mathrm{uw(n)} + 0.5) + (\mathrm{uw(n)} + 0.5)$$

$$\mathrm{v} = \big(\mathrm{uw(n)}\big)^2 + 1.5\mathrm{uw(n)} + 0.5$$

Hence, theoretical mean is

$$\mathrm{E(v)} = \mathrm{E}\left(\big(\mathrm{uw(n)}\big)^2\right) + \mathrm{E}\big(1.5\mathrm{uw(n)}\big) + 0.5$$

$$\mathrm{E(v)} = \mathrm{Var}\big(\mathrm{uw(n)}\big) + 0 + 0.5$$

$$\mathrm{E(v)} = \frac{1}{12} + 0 + 0.5 = 0.5833$$

For theoretical variance, we have

$$\mathrm{Var(v)} = \mathrm{Var}\left(\big(\mathrm{uw(n)}\big)^2\right) + \mathrm{Var}\big(1.5\mathrm{uw(n)}\big)$$

$$\mathrm{Var(v)} = \Big(\mathrm{Var}\big(\mathrm{uw(n)}\big)\Big)^2 + 1.5^2\mathrm{Var}\big(\mathrm{uw(n)}\big)$$

$$\mathrm{Var(v)} = \left(\frac{1}{12}\right)^2 + 1.5^2 \times \frac{1}{12} = 0.1944$$

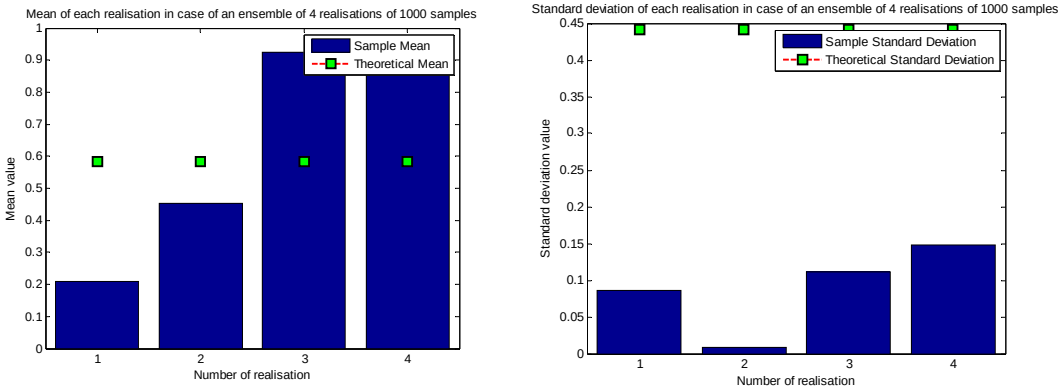


Fig. 15 Left: graphs of sample and theoretical means. Right: graph of sample and theoretical standard deviations of 4 realisations

Note that the theoretical mean and standard deviation do not match with sample means and standard deviations, and that is the reason why this process is non-ergodic.

Third process

1.

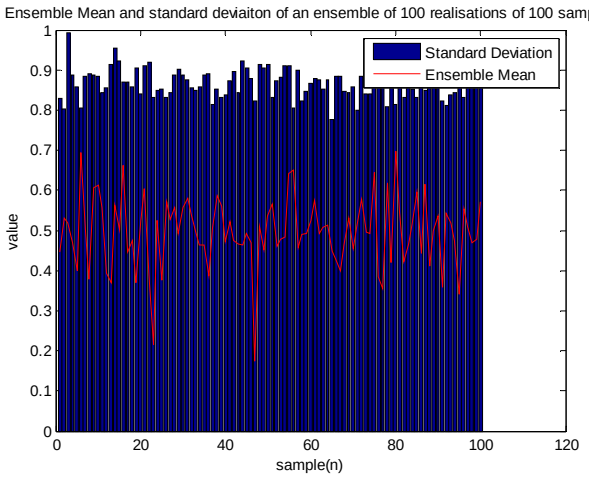


Fig. 16 graph of ensemble mean and standard deviation of an ensemble with 100 realisation of 100 samples

These graphs shows that the ensemble mean is a random noise centered on 0.5 and standard deviation is approximately around 0.9. Hence this process shows some extend of stationary.

2.

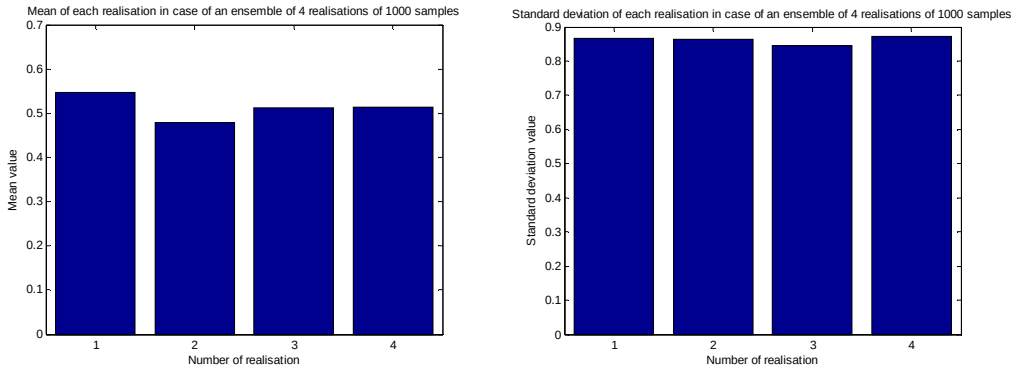


Fig. 17 graphs of sample means and sample standard deviations of 4 realisations

3. The mathematical expression is

$$v = 3uw(n) + 0.5$$

It is easy to directly state the theoretical mean is 0.5, and the theoretical standard deviation is

$$\mathrm{Var}(v) = \mathrm{Var}\big(3uw(n)\big) = 9\mathrm{Var}\big(uw(n)\big) = 9\left(\frac{1}{12}\right) = \frac{3}{4}$$

Hence the theoretical standard deviation is 0.8660

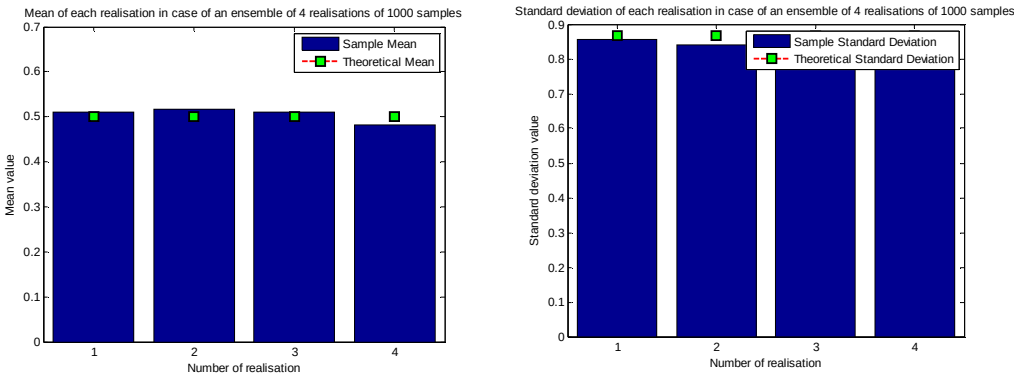


Fig. 18 Left: graphs of sample and theoretical means. Right: graph of sample and theoretical standard deviations of 4 realisations

1.3 Estimation of Probability distribution

1. I created a function named pdf and the code is showed below. This code work perfectly for

Gaussian pdf, **v=randn(1,N)**

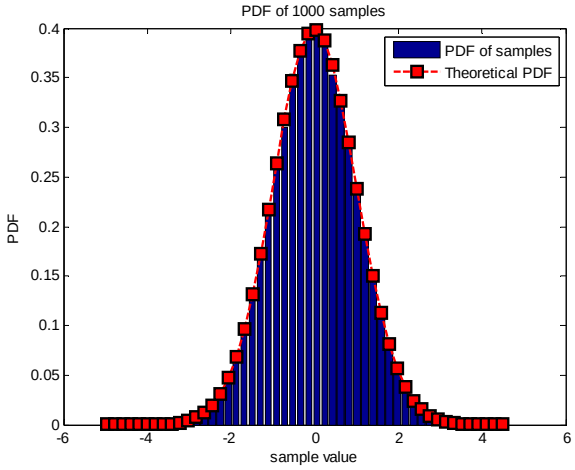


Fig. 19 Sample PDF generated by function pdf and Theoretical PDF

2. Since, among processes in part1.2 (a,b,c), process c) is the only stationary and ergodic one. Therefore, I generated both sample and theoretical pdf of process c) with different sample number (100, 1000, 10000). This is demonstrated in graphs below, and the graphs show that the sample pdf converges more to the theoretical pdf if the number of samples is high.

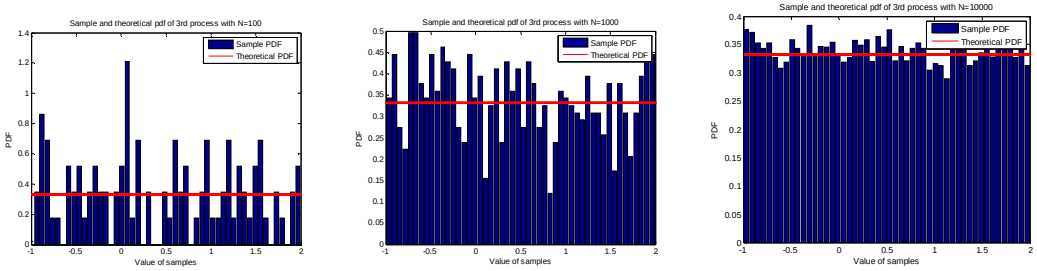


Fig. 20 graphs of sample and theoretical pdf with sample number of 100, 1000 and 10000 respectively

Coursework 2

2.1 ACF of Uncorrelated Sequences

In this session, I am going to discover some basic properties of an autocorrelation function of a White Gaussian Noise (WGN). WGN was firstly generated by **randn(1,1000)** in MATLAB.

1. Having the 1000-sample WGN, I calculated the unbiased estimate of the ACF for it using command **xcorr(x, 999, 'unbiased')** where **x** stored the values of 1000 samples and **999** helps display for $\tau \in [-999 : 999]$. The unbiased estimate of the autocorrelation function had an expression of:

$$\widehat{R_x}(\tau) = \frac{1}{N-|\tau|}\sum_{n=0}^{N-|\tau|-1} x[n]x[n+\tau], \qquad \tau = -N+1, \dots, N-1 \qquad (1)$$

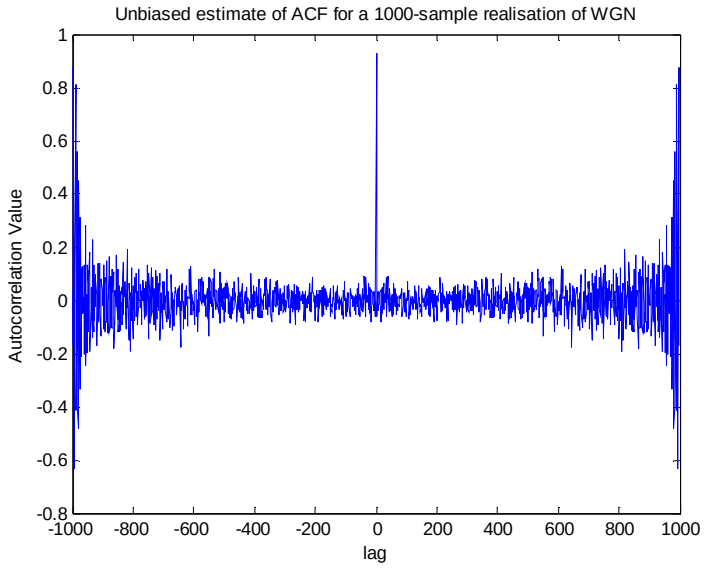


Fig. 1 Unbiased estimate of ACF for a 1000-sample WGN

According to theoretical expression of ACF of a wide-sense-stationary (WSS) random process:

$$R_x(\tau) = E\{x[n]x[x+\tau]\}$$

I found out that the theoretical ACF only had a value at $\tau = 0$. That was because value of a WGN at any specific time was independent to values at other time. Therefore, when $\tau = 0$, theoretically we have

$$R(0) = E\{x[n]^2\} = Var\{x[n]\} = 1$$

When $\tau \neq 0$,

$$R_x(\tau) = E\{x[n]x[x+\tau]\} = 0$$

Therefore, the theoretical ACF has the form of a Dirac function. Since theoretical ACF has a property of

$$R_x(\tau) = R_x(-\tau)$$

it should be an even function.

The graph above demonstrates the unbiased estimate of ACF of a WGN, which generally corresponds to our theoretical expectation especially for small lag. The graph shows that

ACF has a value of 1 when lag is 0. And for lag≠ 0 and small lag, the ACF is approximately 0 even though there are some minor errors. Also, it is obvious that the unbiased estimate of ACF is symmetric about y-axis.

2. After getting an ACF of WGN, I would like to look at the graph in some detail, so I focused onto the region $|\tau| < 50$ by using **axis** function in MATLAB.

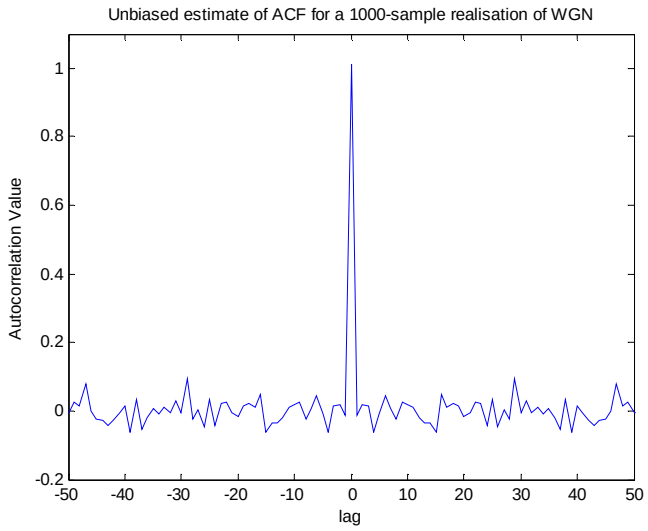


Fig. 2 Zoomed version of unbiased estimate of ACF for a WGN

The above graph shows that for $|\tau| < 50$ and $|\tau| \neq 0$, the ACF value is approximately 0 with small deviations. However, for large value of $|\tau|$ shown in Fig.1, the deviation of ACF increases as the $|\tau|$ increases even though the ACF in this case still generally centres at 0.

3. Fig.1 shows that for large $|\tau|$ (e.g $|\tau| > 400$), the values of ACF are abnormal. That is because, according to equation (1), the order of summation is relatively small in case of large $|\tau|$. That means that the calculation of $\widehat{R_x}(\tau)$ involves in less number of samples as $|\tau|$ increases. Hence, the result of $\widehat{R_x}(\tau)$ is statistically unreliable for large $|\tau|$. By carefully examining Fig.1, I suggest an empirical bound on $|\tau|$, which is $|\tau| < 300$.

2.2 ACF of Correlated Sequences

In this session, I am going to create a correlated sequence by filtering a 1000-sample WGN. Then, I am going to discover some properties of correlated sequences.

1. First of all, I generated a 1000-sample WGN and filtered it by a moving average filter with unit coefficients of order 9. It is noticeable that order 9 in MATLAB actually means moving average process with 8 coefficients MA(8). I then calculated the ACF of filtered WGN by using **stem** function for lags between -20 and 20 in MATLAB.

   The moving average process MA(q) is given by

$$x[n] = w[n] + b_1 w[n-1] + \cdots + b_q w[n-q]$$

Therefore the uncorrelated sequence of WGN going through the filter becomes a correlated sequences because each output sample of the filter is generated by weighted past input samples of the filter.

The theoretical ACF of a MA(q) process is given by:

$$R(\tau) = E\big[\big(w[n] + b_1 w[n-1] + \cdots + b_q w[n-q]\big)\big(w[n-\tau] + b_1 w[n-\tau-1] + \cdots + b_q w[n-k-q]\big)\big]$$

And therefore

$$R(0) = E\big[\big(w[n] + b_1 w[n-1] + \cdots + b_q w[n-q]\big)\big(w[n] + b_1 w[n-1] + \cdots + b_q w[n-q]\big)\big]$$

Since WGN is independent, we have

$$R(0) = E[w[n]w[n] + b_1^2 w[n-1]w[n-1] + \cdots + b_q^2 w[n-q]w[n-q]]$$

$$R(0) = \big(1 + b_1^2 + \cdots + b_q^2\big)E(w[n]w[n])$$

If the 8 coefficients are all 1 in case of order 9 moving average filter, we have

$$R(0) = 9 \times \sigma_w^2 = 9$$

Likewise,

$$R(1) = E\big[\big(w[n] + b_1 w[n-1] + \cdots + b_q w[n-q]\big)\big(w[n-1] + b_1 w[n-2] + \cdots + b_q w[n-q-1]\big)\big]$$

$$R(1) = (b_1 + b_1 b_2 + \cdots + b_7 b_8)\sigma_w^2 = 8$$

$$R(2) = E\big[\big(w[n] + b_1 w[n-1] + \cdots + b_q w[n-q]\big)\big(w[n-2] + b_1 w[n-3] + \cdots + b_q w[n-q-2]\big)\big]$$

$$R(2) = \big(b_2 + b_1 b_3 + \cdots + b_{q-2} b_q\big)\sigma_w^2 = 7$$

$$R(3) = \big(b_3 + b_1 b_4 + \cdots + b_{q-3} b_q\big)\sigma_w^2 = 6$$

$$\cdots\cdots$$

$$R(8) = b_8 \sigma_w^2 = 1$$

$$R(9) = 0$$

That means that the ACF has a cutoff after lag q. Using the codes in Fig. 5, I have generated a ACF graph of MA(8)

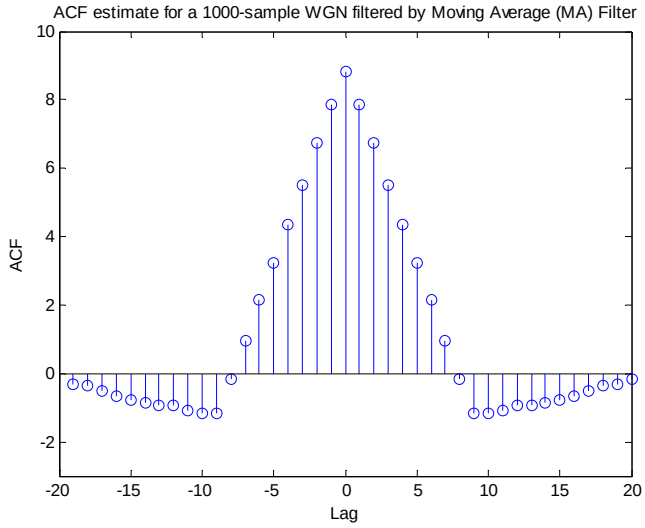


Fig. 3 ACF of filtered WGN for lags between -20 and 20

The graph above demonstrates a ACF of WGN filtered by a MA(8) process with 9 unit coefficients. It shows that the $R(0)$ has the largest ACF value which corresponds to my theoretical result, and it is also clear that the ACF is almost 0 after $lag = |q| = |8|$.

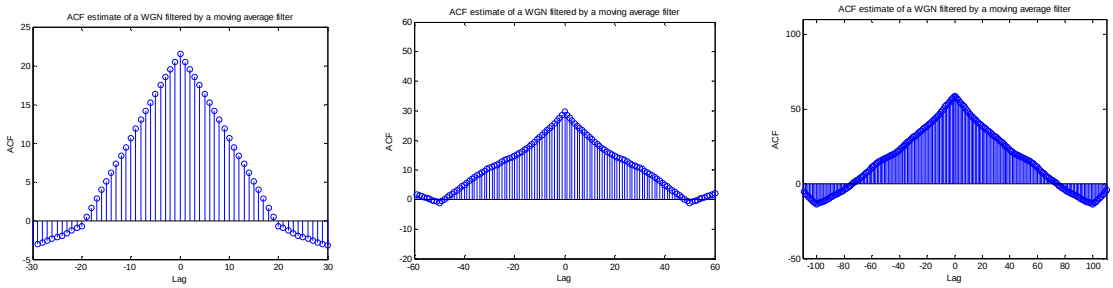


Fig. 4 Top left: ACF with filter order 20. Top right: ACF of with filter order 50. Bottom: ACF of with filter order 100

The above 3 graphs proof that the ACF of MA(q) has a cut off at approximately $lag = |q - 1|$ and the peak value of ACF is equal to $q + 1$.

2. Since X_n is an uncorrelated process, it has an ACF of Dirac delta function multiplied by variance of X. Hence, I have

$$R_Y(\tau) = R_X(\tau) * R_h(\tau)$$
$$R_Y(\tau) = \sigma_X^2 \delta(\tau) R_h(\tau) = \sigma_X^2 R_h(\tau)$$

2.3 Cross-correlation function

In this session, I am going to discover cross-correlation function (CCF) of 2 different stochastic processes.

1. For this question, I used 2 sequences **x** and **y** generated in last section and compute their CCF using **xcorr()** and **stem()** functions.

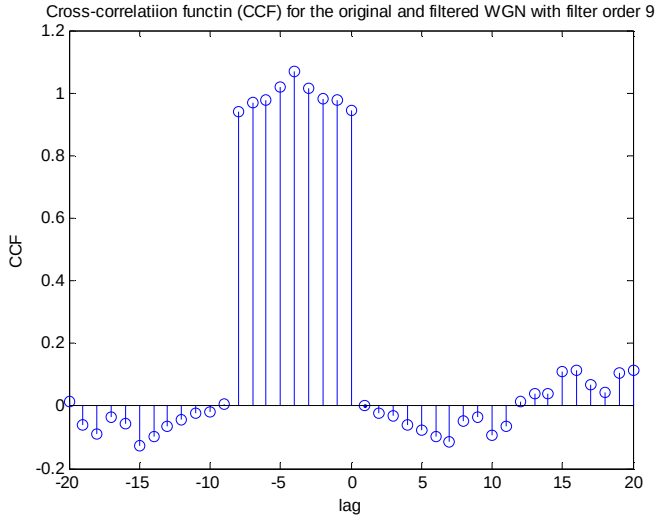


Fig. 5 CCF between a WGN and a filtered WGN

The above graph shows that the CCF function has a shape of rectangle. That is because the cross-correlation function R_{xy} between input and output of a filter is given by

$$R_{xy}(\tau) = h(\tau) * R_x(\tau)$$

In my case, the input signal has an ACF of a Dirac function since it is a WGN. Also, the impulse response of the filter is a rectangular function. Therefore we have

$$R_{xy}(\tau) = rectangular\ function * Dirac\ function = rectangular\ function$$

2. Based on the property discover above, I can use this method to discover the behaviour of a system. If a system is unknown, I can send a WGN as an input and measure the output of the system. Then, I can calculate the ACF of the input and the CCF between input and output. Given, the above relationship between input ACF and CCF, I am able to calculate the impulse response of the system which directly demonstrates the behaviour of the system.

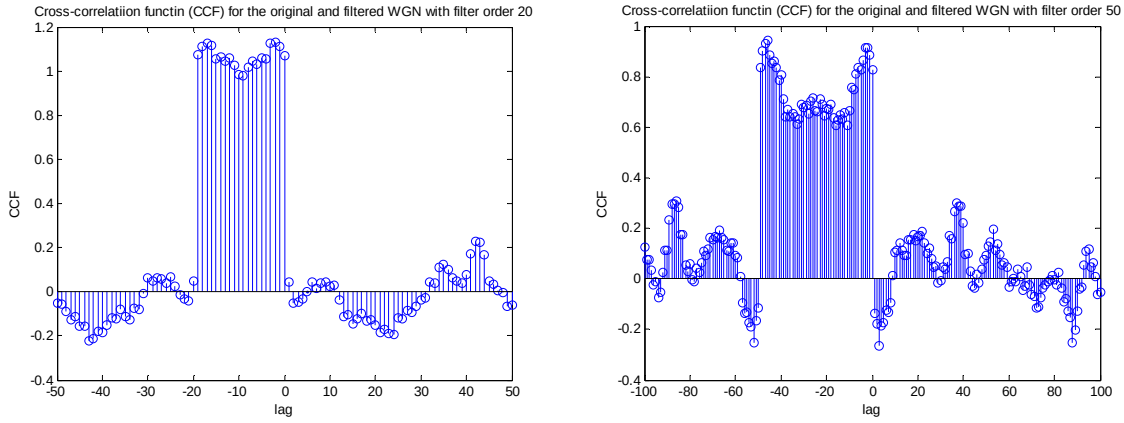


Fig. 6 Left: the CCF with filter order 50. Right: the CCF with filter order 100

The above graphs show that the CCFs with different filter orders (with unit coefficients) have generally the same shape and the length of the rectangular shape of a CCF is equal to the order of the filter. That is because the filter with higher order has longer impulse response function.

2.4 Autoregressive modelling

1. For this question, I generate 100 samples of $a_1 \in [-2.5,2.5]$ and $a_2 \in [-1.5,1.5]$, and I can generate 10000 combinations of (a_1, a_2) as the coefficients of the AR(2) process length 1000, given by

$$x[n] = a_1 x[n-1] + a_2 x[n-2] + w[n], \qquad w[n] \sim N(0,1).$$

If a specific combination of (a_1, a_2) makes x[n] converges, I plot the symbol * at the position (a_1, a_2). In practice, I take last 100 sample of the filtered signal and calculate its peak-to-peak value for each combination of a_1 and a_2. If the peak-to-peak signal is below 2, I believe that it is a convergent signal and plot corresponding a_1 and a_2 on a graph.

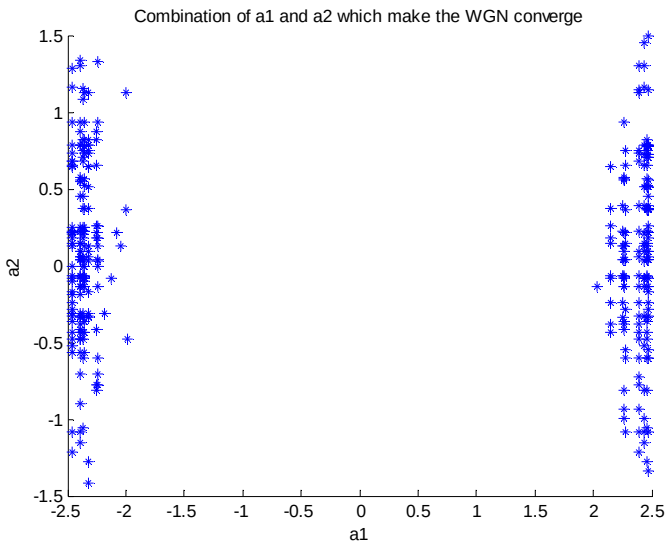


Fig. 71 Plot of combination of a1 and a2

2. In this question, I am going to analyse the sunspot data by loading **sunspot.dat** in MATLAB. The sunspot data is shown below.

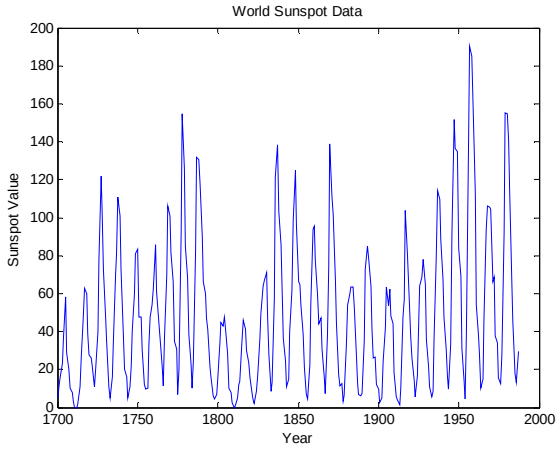


Fig. 8 Sunspot time series

After loading the sunspot data, I calculate the ACFs for data length N=5, 20, 250 by randomly taking successive 5, 20 or 250 samples from sunspot data and computing their ACFs. I calculate the ACFs of the first 5, 20 and 250 samples and ACFs of No.50-55, No.50-70 and No.38-288 samples.

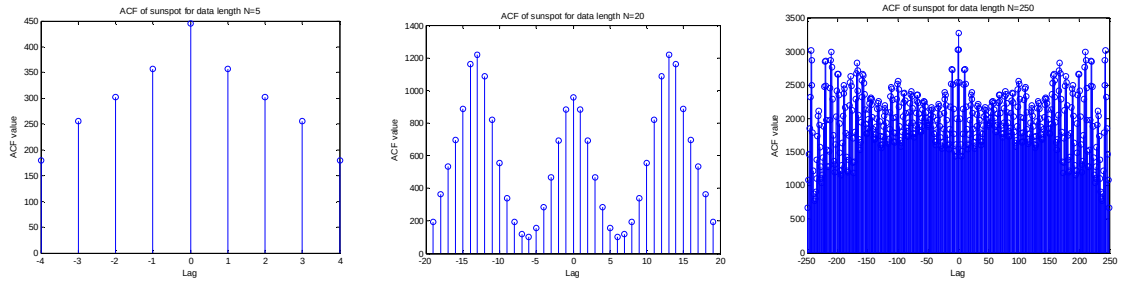


Fig. 9 Left: ACF of first 5 samples. Middle: ACF of first 20 samples. Right: ACF of first 250 samples

From the above graphs, one peak of ACF can be observed for N=5 and 3 peaks can be seen. For N=250, there exists multiple peaks.

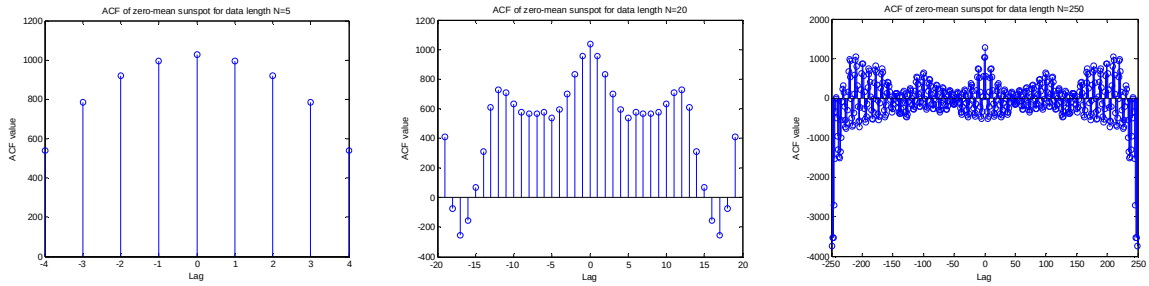


Fig. 9 Left: ACF of first 5 zero-mean samples. Middle: ACF of first 20 zero-mean samples. Right: ACF of first 250 zero-mean samples

Generally, ACF of zero-mean data has reduced ACF value whereas the ACF maintains a same pattern as the non-zero-mean ACF. The zero-mean ACF generally centres at 0

3. Given the sunspot data, I used Yule-Walker equations to estimate the coefficients of an autoregressive model up until the model order p=10. And I plot a partial correlation function to demonstrate the coefficients.

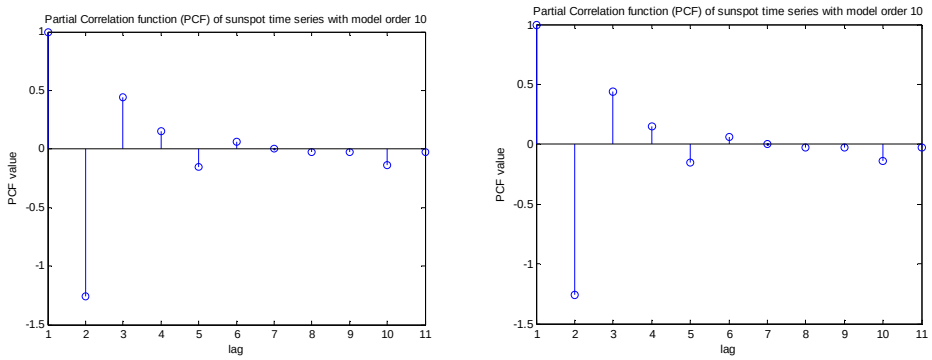


Fig. 10 Left: PCF of sunspot data. Right: PCF of sunspot data with zero mean and unit variance

The above 2 graphs show that the PCFs of original sunspot data and standardised data have the same coefficients. It is also noticeable that the coefficients above 3 become tiny and play an irrelevant role. Since the first 3 coefficients are dominating the AR model which generating sunspot data from white noise, the likely order of the model is 2.

4. By using the equation stated in the coursework notes, I obtain MDL and AIC value of different model order for sunspot data.

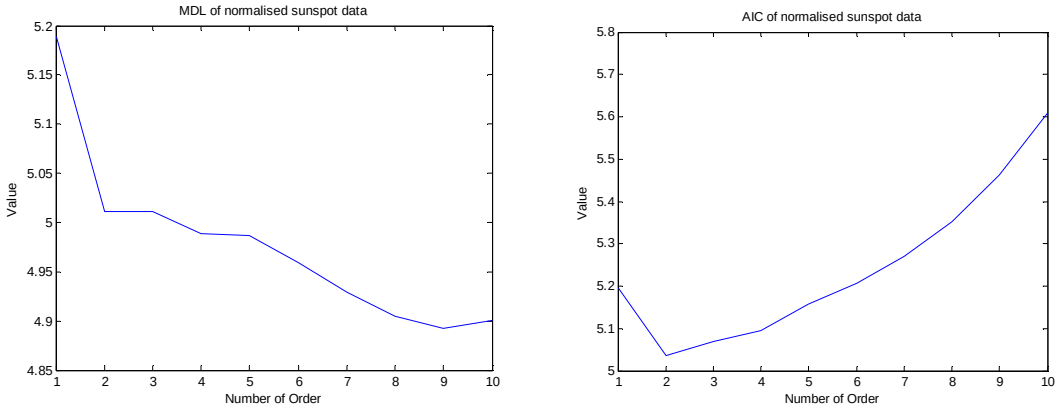


Fig. 11 Left: MDL of normalised sunspot data. Right: AIC of normalised sunspot data

From the above 2 graph, it is clear that both AIC and MDL value change significantly at model order 2. Hence, the model order of 2 is by definition the optimal.

5. In this question, I predict the future sunspot data using AR model with different prediction horizons and model order.

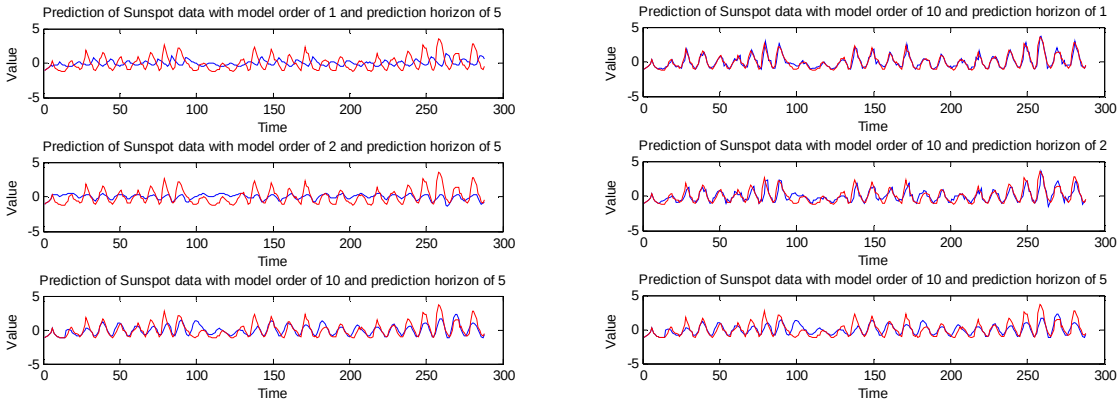


Fig. 12 Left: Sunspot prediction with horizon of 5 and model order of 1, 2, 10. Right: Sunspot prediction with model order of 10 and horizons of 1, 2, 5

From the above graphs, it is noticeable that the sunspot prediction is closer to the original sunspot data for higher model order. Also, sunspot prediction is better and more accurate if the horizon is the smallest (for example horizon of 1 in the above graphs). Hence, model order 10 with horizon of 1 generates the best result.

Coursework 3

In this coursework, I am going to discover some properties of power spectral density (PSD). First of all, I construct a function in MATLAB called **pgm()** to calculate the PSD of a input signal using built-in **periodogram()** function. Then, I test this function using WGN with sample number of 128, 256 and 512, and draw the PSD of WGNs. It is noticeable that, in MATLAB, the normalised frequency has the range of 0 to 1 and the Nyquist frequency is 0.5.

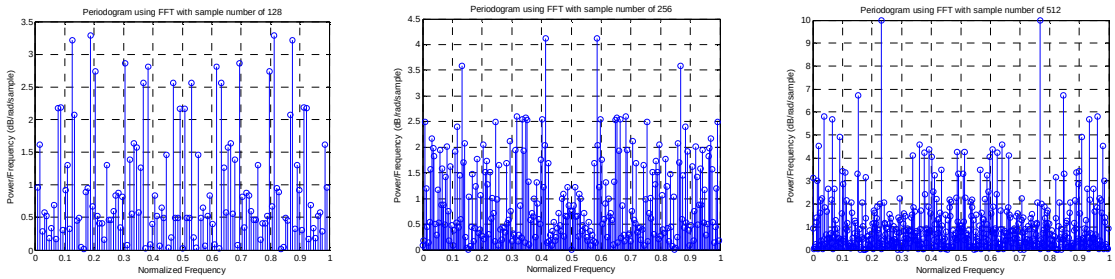


Fig. 2 Left: PSD of WGN with sample number of 128. Middle: PSD of WGN with sample number of 128. Right: PSD of WGN with sample number of 128.

The above 3 graphs show that the more samples the WGN has, the more samples the PSD has. Also, the PSD with larger number of samples seems to centre at a particular value with some deviations, since theoretically the PSD of WGN is 1 for all frequency components. However, it is still difficult to see a general pattern of PSDs of sample WGNs. Because the Nyquist frequency is 0.5, the above 3 plots are symmetric about 0.5.

3.1 Averaged periodogram estimates

Theoretically, the ACF of WGN is an Dirac delta function with area of 1 since samples generated by **randn** follow a Gaussian distribution with variance of 1. Therefore, the PSD of WGN should be a 1 for all frequencies.

1. I apply a FIR filter with impulse response of [0.2 0.2 0.2 0.2 0.2] to obtain a better PSD using the **filtfilt()** function

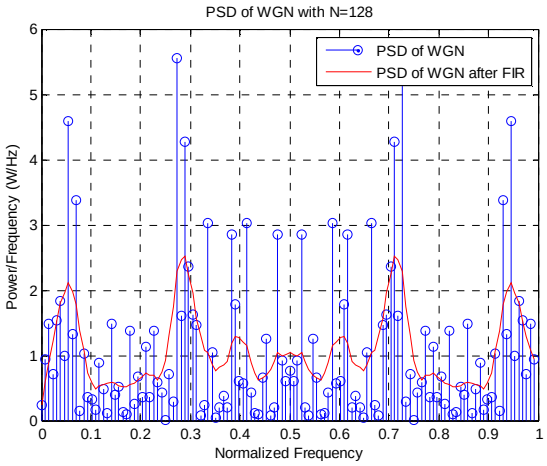


Fig. 3 PSDs of WGN and filtered WGN

The above graph shows that the FIR filters smooth the original PSD and the filtered PSD is somehow centred at 1. However, the deviations of PSD are still large

2. To further improve my PSD estimate, I generate a 1024-sample WGN and divide them into 8 non-overlapping 128-sample segments. I calculate the PSDs of each segment using **pgm()**.

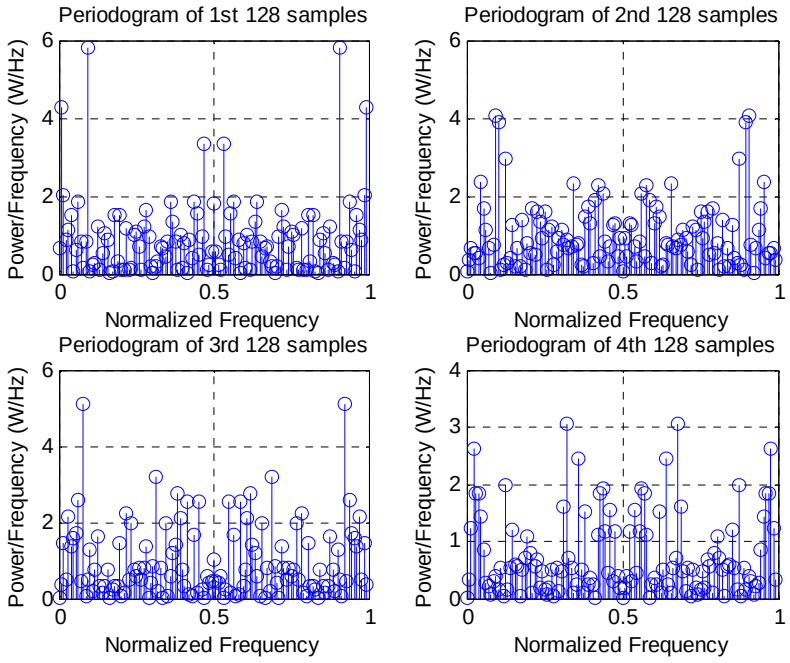


Fig. 3 PSDs of first 4 segments of WGN

The above graphs show that each segment has similar PSD. This method has little improvement on PSD and the deviation of PSDs is still very large for each segment.

3. After the above exercise, I average the PSDs of these 8 segments to generate a new PSD.

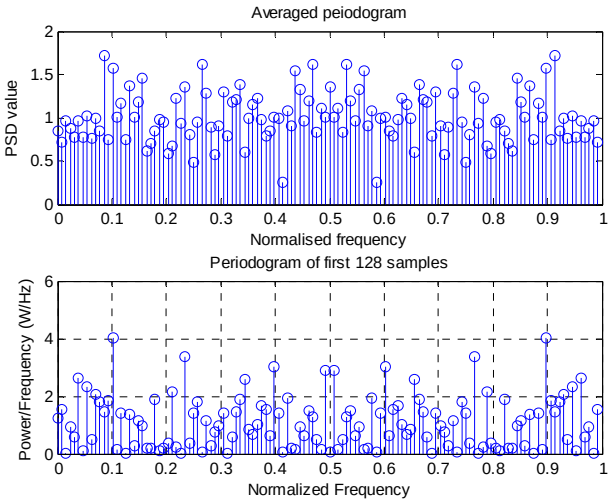


Fig. 4 Top: Averaged PSD of 8 segments. Bottom: PSDs of the first segments of WGN

The above graph demonstrates that the averaged PSD is much better and smoother compared to individual PSDs. And the averaged PSD is centred at 1 with acceptable deviations.

3.2 Spectrum of autoregressive processes

First of all, I generate a 1064-sample of WGN and generate another sequence **y** using a first-order Autoregressive model AR(1) with coefficients of [1, 0.9] by applying **filt ()** function to the WGN in the MATLAB. I also remove the first 40 samples of the sequence **y** as these are affected by the transient effects of the filter.

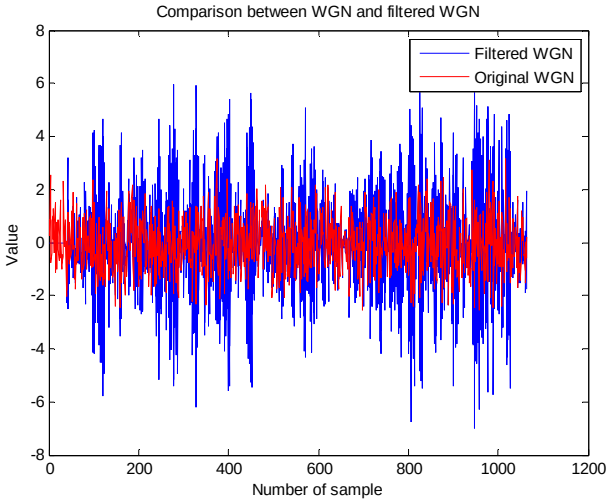


Fig. 5 Signals of original WGN and filtered WGN

1. According to the coursework notes, I can calculate and plot the theoretical PSD of the above AR(1) model.

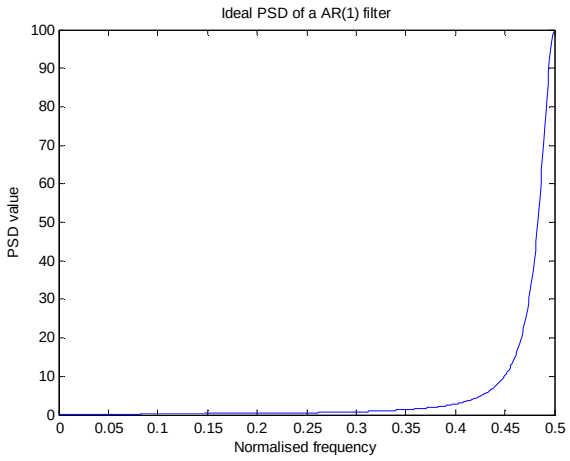


Fig. 6 Ideal PSD of the AR(1) model above

The above graph demonstrates that the PSD has a peak at Nyquist frequency. That is because, according to the notes, the AR has a transfer function of

$$H(z) = \frac{1}{1 + \sum_{k=1}^{p} a_p(k) z^{-k}}$$

Hence, the above AR(1) model has a transfer function of

$$H(z) = \frac{1}{1 + 0.9z^{-1}}$$

As shown in the above equation, this AR(1) model has a pole at $z = -0.9$ which is near the unit circle and corresponds to $\omega = \pi$ in the z-plane. Since a pole near the unit circle will cause the frequency response to increase in the neighbourhood of that pole, a peak is observed at Nyquist frequency in frequency response. Hence, a peak is also observed at Nyquist frequency in power spectrum.

2. I then plot a graph of the sample PSD of **y** using function **pgm()** and compare it with the above theoretical PSD.

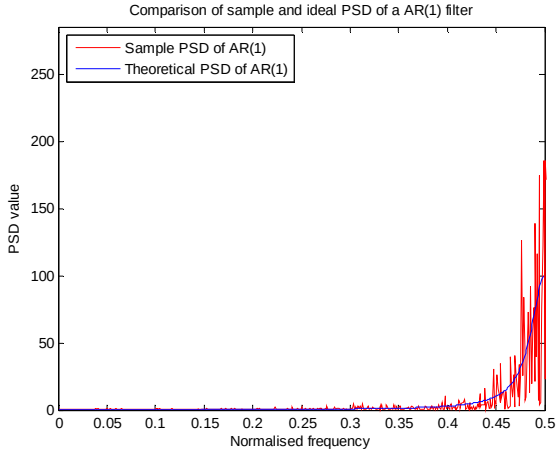


Fig. 7 Ideal and sample PSD of the AR(1) model

It is noticeable that the sample PSD is generally similar to theoretical PSD for frequency below 0.45. For high frequency (0.45 to 0.5), the sample and theoretical PSDs have the same

patten (having peak at Nyquist frequency) but the sample PSD has large deviation compared to theoretical PSD.

3. I also zoom in on the interval between 0.4 and 0.5 to examine in more detail.

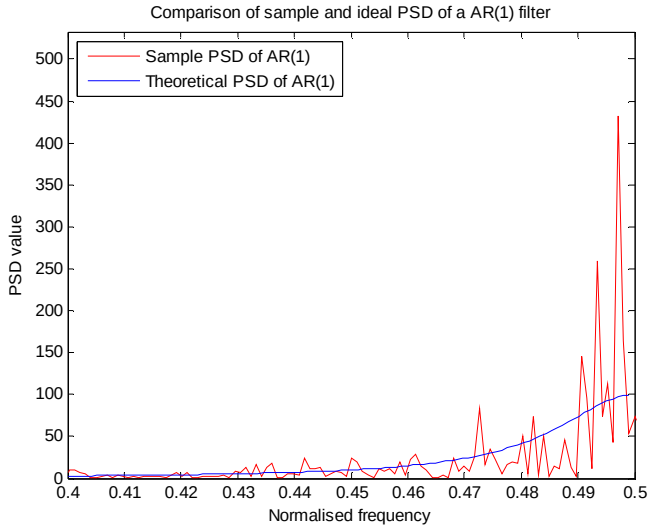


Fig. 8 Zoom-in version of Ideal and sample PSD of the AR(1) model

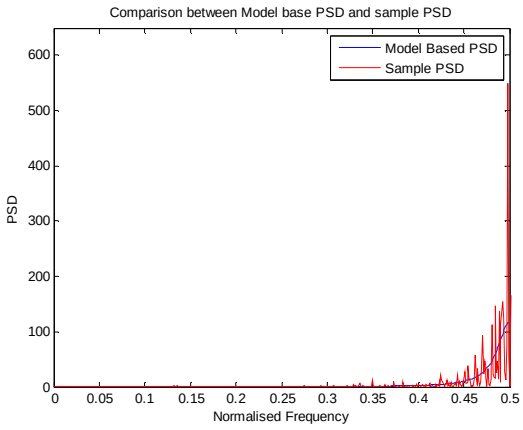
As stated in the above question, the sample PSD using periodogram has large deviations at high frequency. That is because, theoretically, I assume that the input of AR(1) model (i.e. WGN) has a perfectly constant PSD with a value of 1 for all frequencies. However, for above sample PSD, the PSD of the input WGN is centred at 1 but with some deviation. That causes the sample PSD of filtered WGN has large deviations at high frequencies.

4. In this question, I apply Model based PSD method to calculate PSD. I first of all assume that the sequence **y** is generated by a AR(1) model. And then I calculate the autocorrelation of **y** in order to estimate 2 parameters of AR(1) using

$$\widehat{a_1} = -\frac{\widehat{R_Y}(1)}{\widehat{R_Y}(0)}$$

$$\hat{\sigma}_X^2 = \widehat{R_Y}(0) + \widehat{a_1}\widehat{R_Y}(1)$$

In my case, $\widehat{R_Y}(1) = 6.004$, $\widehat{R_Y}(0) = 6.5008$. Hence, $\widehat{a_1} = 0.9236$ and $\hat{\sigma}_X^2 = 0.9556$. Therefore, I can again generate a PSD estimate using $\widehat{a_1}$ and $\hat{\sigma}_X^2$.



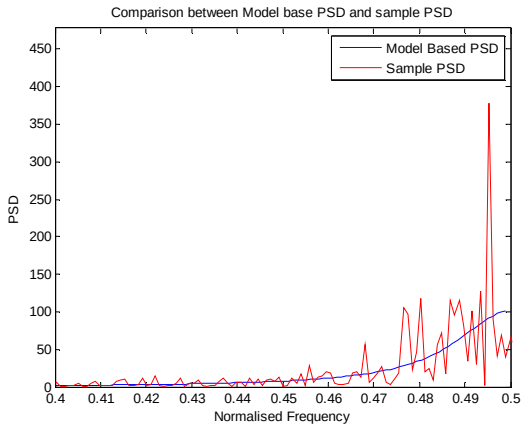


Fig. 9 Left: Model based and sample PSD of the AR(1) model. Right: Zoom-in version of Model based and sample PSD of the AR(1) model

The above 2 graph illustrate that the model based PSD has the same shape as the periodogram but periodogram has some large deviations for high frequency components. As a result, model based PSD is very similar to theoretical PSD using **freqz()** function. That is because the value of $\widehat{a_1}$ and $\hat{\sigma}_X^2$ calculated are highly close to the value of a_1 and σ_x^2 when we use **freqz()**.

5. I load the sunspot data and repeat the above procedure (i.e. I compare the periodogram of sunspot data with model based PSD). I also evaluate the PSD for both original and zero-mean version of sunspot and I also try different model orders. I use Yule-walker equation to generate the noise variance estimate and coefficient estimates under higher orders.

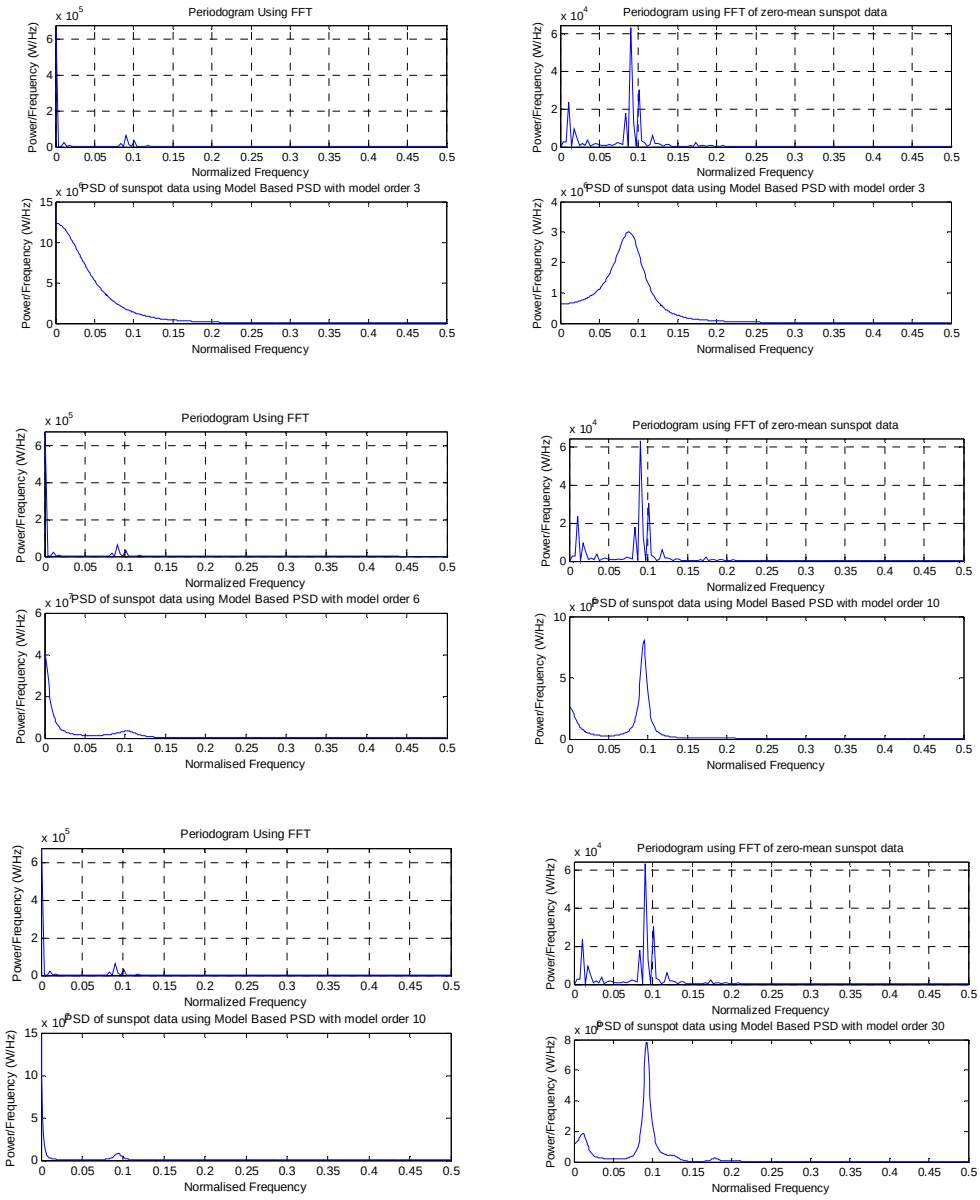


Fig. 10 Left 3 graphs: Model based and sample PSD of sunspot data with model order 3, 6, 10. Right 3 graphs: of Model based and sample PSD of zero-mean sunspot data with model order 3, 10, 30

It is clear that the model based PSD with higher number of order will be more similar to sample PSD. That means that higher number of order actually makes the PSD more accurate and closer to practical PSDs.

3.3 Spectrogram for time-frequency analysis: dial tone pad

In this session, I will discover touch-tone telephone dialling, which has an underlying concept of Dual Tone Multi-Frequency system. It assigns a signal composed of 2 sinusoids to each button of the keypad.

1. I first of all generate a random sequence of London landline number where the last 8 digit are drawn from 0 to 9 with uniform probability. Then I generate a dialling signal using the data from the note and sampling frequency of 32768Hz

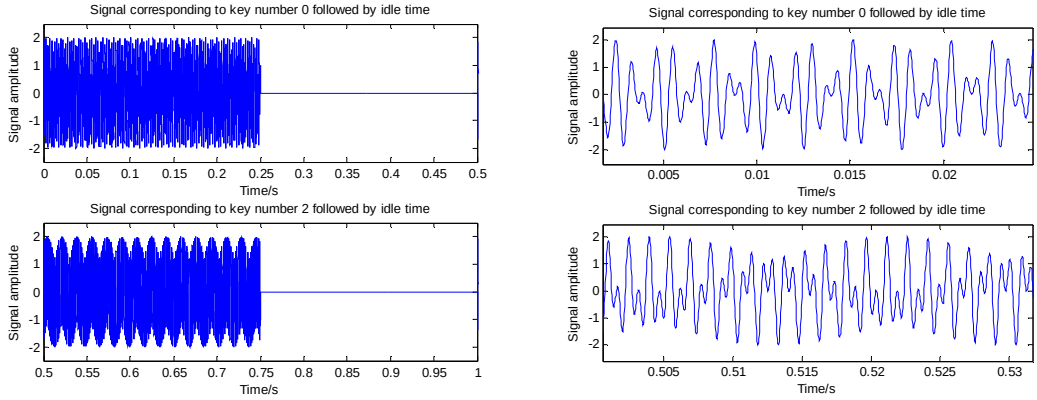


Fig. 11 Left: Signal of key number 0 and 2 followed by idle time. Right: Zoom-in signal of key number 0 and 2

The above graph demonstrates that the signal for key number 0 and 2 are different in period and shape. It is also noticeable that the sampling frequency is chosen at 32768Hz because the sampling frequency should be 2 times larger than the signal frequency (according to Nyquist Criterion). Since the maximum frequency component among all keys is 1477 Hz, my chosen sampling frequency definitely meet the criterion. Also, the sampling frequency should be large enough to reconstruct dialling signal, so the frequency several times beyond the Nyquist frequency will be better. Apart from that, we need our sampling frequency to be some power of 2 in order to perform efficient FFT in the following questions. Therefore, 32768Hz is appropriate.

2. I then use spectrogram to analyse the signal of whole 11 digit telephone number. According to the note, Hanning window is suggested. By testing, I discover that larger size of Hanning window may remove relatively large number of samples used for calculating spectrogram, meanwhile small number of window size does not provide PSD data in enough detail. Therefore, I choose window size of 1024. Also, a low FFT length will reduce the resolution of spectrogram. Therefore, I choose FFT length of 1024.

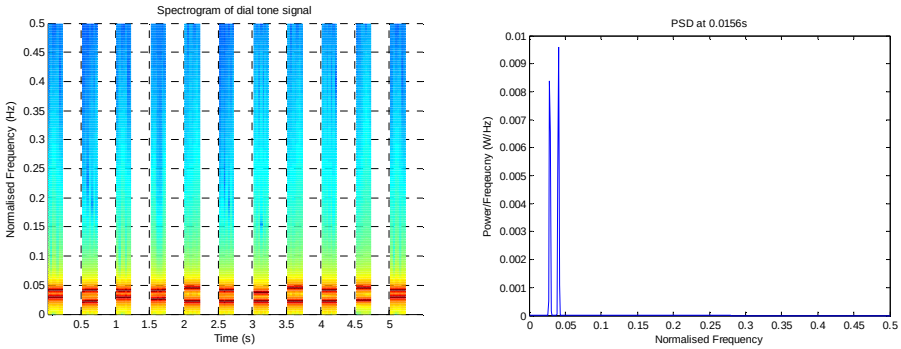


Fig. 12. Top: Spectrogram of the dialling singal. Bottom: PSD segment at 0.0156s

In spectrogram, the red parts represent high power components whereas blue parts represent low power component. For example the bottom graph demonstrates that, at 0.0156s, power is high for 2 specific frequency components.

3. Given spectrogram and the table of frequency pairs corresponding to each key, we are able to identify the landline number that has been dialled. In Fig. 12, the spectrogram indicates that, apart from the idle time, there are always 2 frequency components having high power. These high power frequency components correspond to a pair of frequency related to the key that is being pressed. Hence, in order to identify the landline number sequence, I should analyse the spectrogram and identify the pairs of frequency components with high power at each instant of time. Then, I can search in the table to look for keys corresponding to these frequency pairs.

4. In this question, I apply a WGN with variance (power) of 0.00001, 0.01 and 1 respectively to the dialling signal, and analyse these noise corrupted signal again using spectrogram.

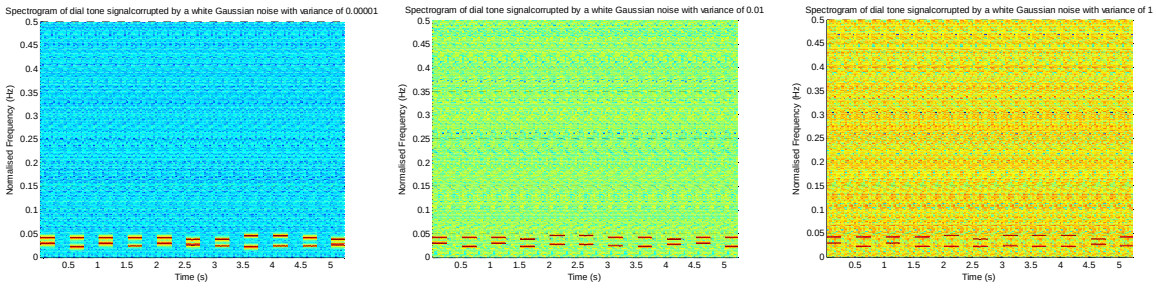


Fig. 13. Spectrograms of the dialling singal corrupted by noise with variance of 0.00001 (Left), 0.01 (middle) and 1 (right)

In above spectrograms, the background colour has been changed which corresponds to different power of noise. For high power noise (with variance of 1), even though the original signal is virtually immersed in noise, pairs of frequencies corresponding to keys are still clear enough to identify. Hence, noise has little effect on tone identification. However, if the power of noise is similar or even larger than the information signal, it is also unlikely to identify the tone.

3.4 Spectrum of dialling signal using AR model

For this session, I generate a noise corrupted (with SNRs of 3, 10 and 20) dialling signal corresponding to tone 3 and *. And then I model these dialling signals using AR model and calculate coefficients using Yule-Walker Equation. I also calculate the PSD of the AR model. Theoretically, since tone 3 apply frequency pair of 1477Hz and 697Hz, there are supposed to be peaks in PSD graph at normalised frequencies of 1477/32768=0.045 and 697/32768=0.0213. Likewise, for tone *, I should observe peaks in PSD graph at normalised frequencies of 0.0369 and 0.0287.

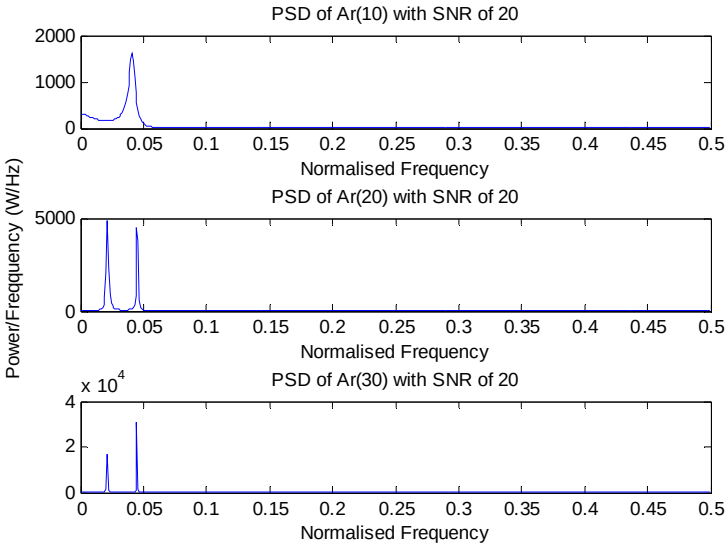


Fig. 13. PSDs of noise corrupted (SNR=20) dialling signal of tone 3 under model order of 10, 20 and 30

The above graphs demonstrate that the PSDs have more resolution and can display more detail if the model order is large enough. Also, the above 2 peaks occurs at 0.02148 and 0.04492 which corresponds to our calculation. Apart from that, I also observe that the signal with lower SNRs requires higher number of model order since the noise is larger and PSD requires more resolutions. Also, tone * requires more resolution because its 2 peaks are very close to each other. Hence, finally the order I determine is shown below.

SNR value	Number of Model Order	
	Tone 3	Tone *
20	17	23
10	20	30
3	25	45

Coursework 4

4.1 Wiener Filter

In this section, I am going to implement a Wiener filter to identify an unknown system and examine some of its properties.

First of all, I generate a 1000-sample WGN as x[n], and generate another 1000-WGN with standard deviation of 0.1 as η[n]. Hence the SNR of the signal is 10.12 dB. Given the filter coefficients of unknown system, I am able to obtain y[n] and z[n]. Hence, by using the algorithm described in the notes, I calculate the coefficients of Wiener filter.

1. According to the note, I obtain a matrix of R_{xx} and a vector of P_{zx} by calculating autocorrelation function of x[n] and cross-correlation function between z[n] and x[n].

$$R_{xx} = \begin{bmatrix} 0.9980 & 0.0370 & 0.0452 & -0.0261 & -0.0442 \\ 0.0370 & 0.9980 & 0.0370 & 0.0452 & -0.0261 \\ 0.0452 & 0.0370 & 0.9980 & 0.0370 & 0.0452 \\ -0.0261 & 0.0452 & 0.0370 & 0.9980 & 0.0370 \\ -0.0442 & -0.0261 & 0.0452 & 0.0370 & 0.9980 \end{bmatrix}$$

$$P_{zx} = \begin{bmatrix} 0.7102 \\ 0.4847 \\ 0.2433 \\ -0.0181 \\ -0.0247 \end{bmatrix}$$

   Hence the optimum Wiener filter solution is

$$w_{opt} = \begin{bmatrix} 0.6856 \\ 0.4545 \\ 0.1965 \\ -0.0284 \\ 0.0096 \end{bmatrix}$$

   It is noticeable that the Wiener filter coefficients are very different from the coefficients of unknown system. That is mainly because the Wiener filter is taking noise corrupted signal z[n] rather than y[n] to estimate coefficients. Therefore, the noise makes the coefficients of Wiener filter different from the coefficients of unknown system.

2. I repeat the experiment by trying six different variance of η[n]. For each η[n], I calculate the SNR and Wiener filter coefficients.

Variance	SNR	Wiener Filter Coefficients				
		W1	W2	W3	W4	W5
0.1	3.1929	0.6798	0.4525	0.2606	0.0097	0.0108
0.5	1.4563	0.6914	0.4387	0.2805	0.0054	-0.0175
1	0.9864	0.7577	0.4869	0.2196	0.0654	0.003
3	0.5923	0.7474	0.5348	0.2725	-0.0176	0.0364
6	0.4147	0.7090	0.4142	0.2952	0.0056	-0.0133
10	0.3177	0.5998	0.3800	0.1949	0.1144	-0.0187

Table 1 Table of Wiener filter coefficients under different noise power

From the table above, I observe that the coefficients are almost the same for all variance. Therefore, it is predictable that the power of noise does not affect the coefficients

Apart from that, I also change order of Wiener filter to 8, 10 and 20 with the same variance value of 0.1.

Order 8	Order 10	Order 20 (first 10 coefficients)
0.6798	0.6790	0.6792
0.4543	0.4533	0.4534
0.2332	0.2330	0.2340
0.0114	0.0107	0.0102
-0.0278	-0.0278	-0.0263
-0.0347	-0.0343	-0.0334
-0.0211	-0.0210	-0.0180
-0.0608	-0.0610	-00619
	-0.0082	-0.0061
	-0.0131	-0.0124

Table 2 Table of coefficients under different order number

The above table shows that for all orders, the first several coefficients are large and dominant, whereas tThe following coefficients of higher order become small and negligible.

3. This algorithm requires calculating autocorrelation and cross-correlation which has $(N+1)N$ multiplications. Also, the 3 for loop requires $4 \times (N_w+1)^2 + 2 \times (Nw+1)$ addition, and the matrix multiplication at the end requires N_w^2 multiplications and $N_w(N_w+1)$ additions. Since the inverse of matrix require complexity of $O(N^3)$, the total complexity of the algorithm is $O(N_w^3)$.

4.2 The least mean square (LMS) algorithm

1. I implement the LMS algorithm and write a MATLAB routine called **lms.** I test this algorithm using the signal x[n] and z[n] in 4.1.

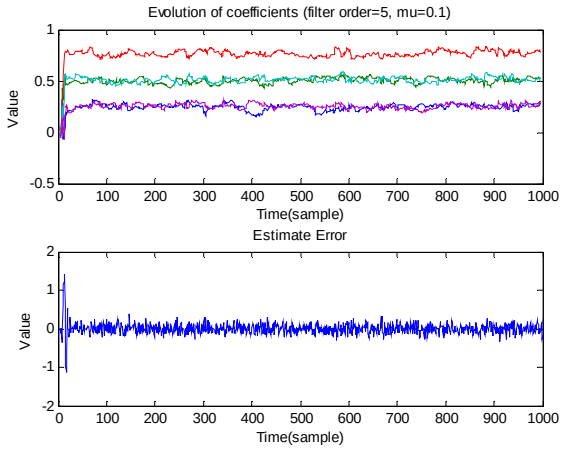


Fig. 1 Evolution of Coefficients and estimate error w.r.t time

2. I choose the adaptation gain to be 0.01 and calculate the evolution of coefficients. I also compare these coefficients with Wiener filter coefficients.

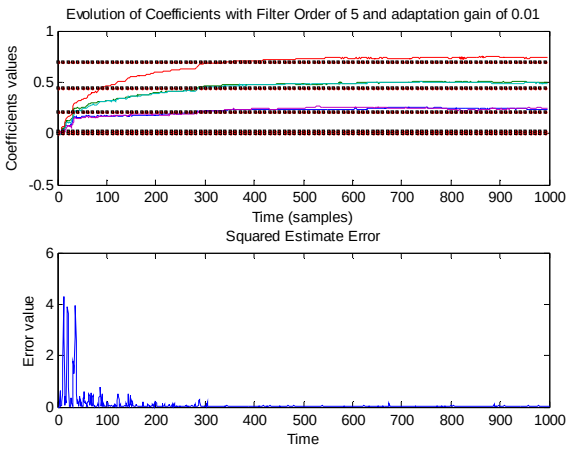


Fig. 2 Evolution of Coefficients and estimate error w.r.t time compared with Wiener filter coefficients

In the above graph, the black dot lines represent the Wiener filter coefficients, and it is clear that the LMS coefficients converge to Wiener filter coefficients. That is because the LMS algorithm and Wiener filter are doing the same thing i.e. estimating a unknown system coefficients. The only difference is that Wiener filter estimates the coefficients given the whole input and output signal and assumes statistical stationary. However, the LMS algorithm can calculate coefficients with time. Therefore, LMS coefficients will move close to Wiener filter solution with time since they are estimating the same system.

From the graph, I can see that the LMS coefficients start being very close to Wiener coefficients from No.300 sample and squared estimate error starts being negligible from the same point. After that, I also try different value of adaptation gain μ and I find out that the LMS coefficients with larger adaptation gain converge more quickly to the Wiener filter coefficients. But once the adaptation gain exceeds a specific value, I can no longer obtain acceptable coefficients. By testing, the maximum μ is approximately 0.19. Compared to larger adaptation gain, smaller μ makes the coefficients converges more closely to Wiener coefficients.

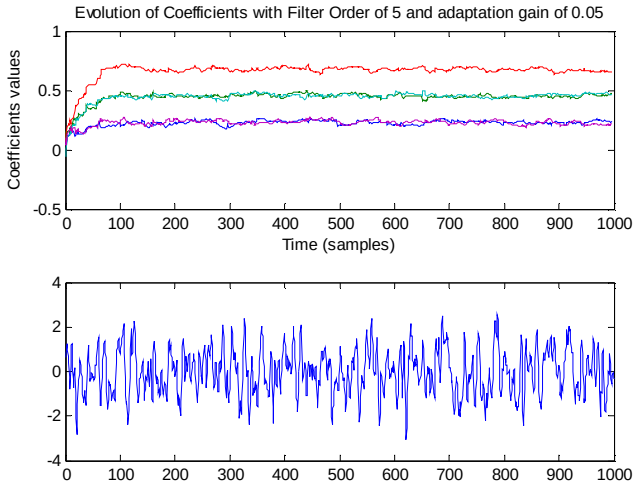


Fig. 3 Evolution of coefficients and estimate error with mu=0.05

3. In the LMS algorithm, it is clear that the command operate consume the most of the computational power and the complexity of the whole algorithm is determined by the equation:

$$\hat{y}[n] = w^T(n)x(n) = \sum_{m=0}^{N_w} w_m(n)x(n-m)$$

Therefore, the overall complexity of the algorithm is $O(NN_w)$;

4.3 Gear shifting

In this part, I start to modify μ in the lms function according to the previous value of e(i). This gear shifting method should smooth the LMS evolution curves of coefficients especially when it is getting closer to steady state values. While manipulating the adaptation gain μ should not exceed its maximum value, and it may not be very small since it will slow down the convergence. Also, the maximum overshoot is required to be lower than 20% of the steady state value, so I cannot decrease μ very quickly at the beginning. To actually implement this filter, I try 3 methods.

The first method calculates, in the for-loop, the mean square error (MSE) and minimum MSE which is the variance of the noise signal. Then, I calculate the difference between MSE and minimum MSE and assign the result as excess MSE. If the excess MSE is 1/10 of the minimum MSE, I scale μ by a factor of 0.5. However, this method has little effect on the coefficient convergence.

The second method simply scales μ by a factor of 0.95 for every loop cycle. However, this method changes the convergence value under different μ values.

The third method takes the current estimate error value e(i). If e(i)* e(i) is greater than 1, I use the original input μ. If e(i)*e(i) is in between of 0.3 and 1, I scale the μ by a factor of 0.8. If e(i)*e(i) is below 0.3, I scale the μ by a factor of 0.3. This method generate better evolution curves of coefficients

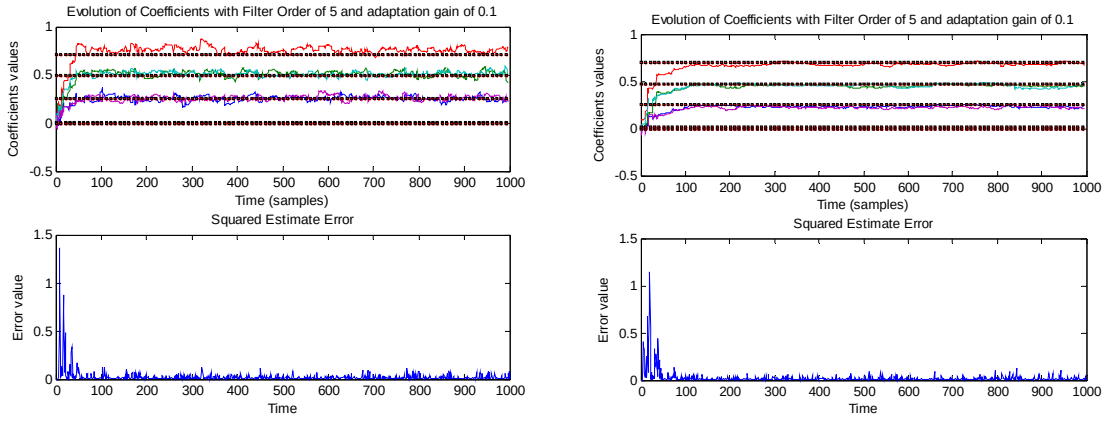


Fig. 4 Left: Evolution of coefficients and estimate error without gear-shifting. Right: Evolution of coefficients and estimate error with gear-shifting

The above graph demonstrates that the coefficients with gear-shifting are closer to convergent values once they start converging.

4.4 Identification of AR process.

1. I write a program to implement the synthesis and analysis structure in the coursework notes. I filter the input of the function by a AR model, and replace the N_w in **lms()**by 2 and the **w** by **a**. I also implement gear shifting to obtain a smoother evolution of coefficients.

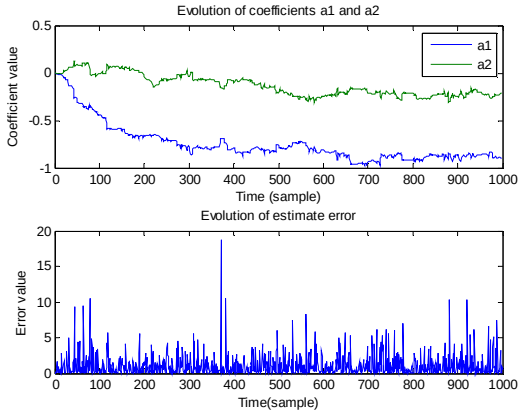
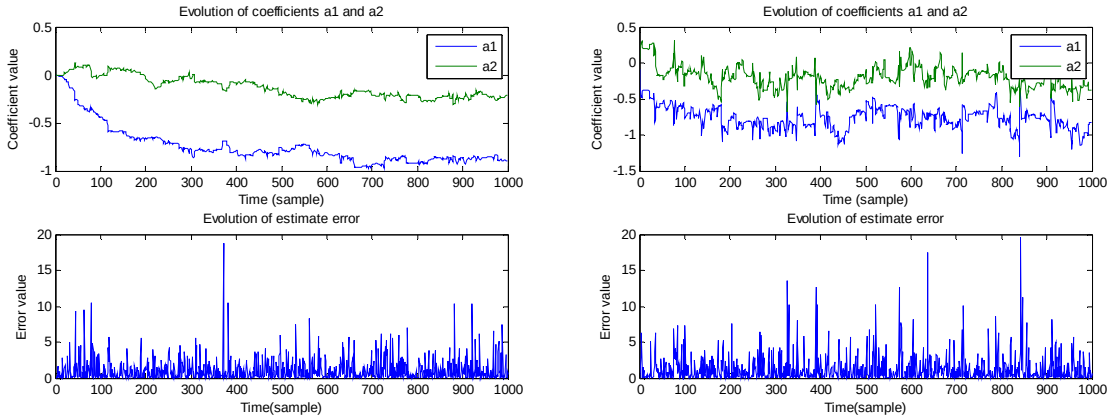


Fig. 5 Evolution of coefficients and estimate error with $\mu = 0.01$

The graph shows that the coefficients a_1 and a_2 converges to -0.2 and -0.9 respectively, which corresponds to the AR model coefficients.

2. Then, I try 4 different μ of 0.01, 0.05, 0.07 and 0.1.



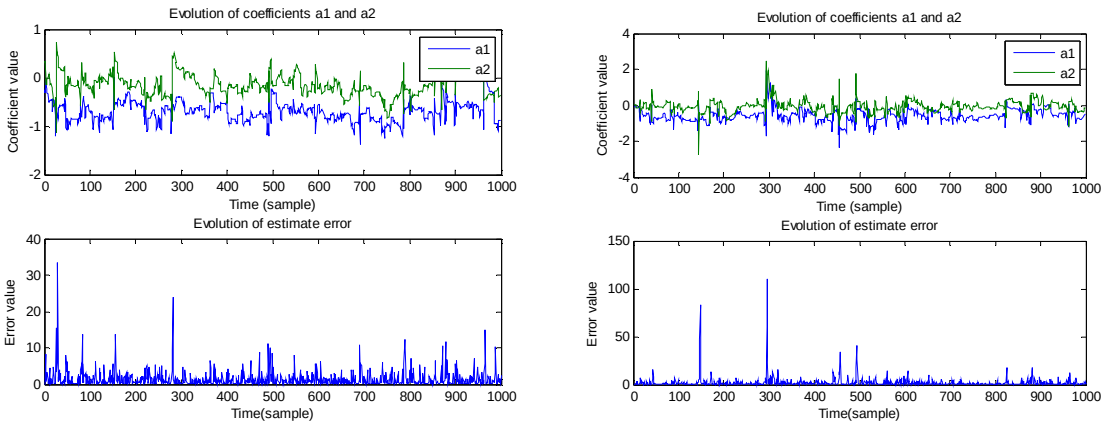


Fig. 6 Top left: the evolution of coefficients and estimate error with $\mu = 0.01$. Top right: the evolution of coefficients and estimate error with $\mu = 0.05$. Bottom left: the evolution of coefficients and estimate error with $\mu = 0.07$. Bottom right: the evolution of coefficients and estimate error with $\mu = 0.1$.

The 4 graphs above show that the coefficients converge to -0.9 and -0.2, and larger adaptation gain can result in faster convergence but larger errors.

4.5 Speech recognition

1. In this question, I modify the analysis structure in 4.4 into an p^{th} order analysis structure with coefficients $a_1, a_2, \dots, a_p$ to recreate a speech signal. To test this algorithm, I record my own voice corresponding to the sounds “e”, “a”, “s”, “t” and “x” with sampling frequencies of 44.1 kHz, and take 1000 sample in these voice file to test my code. It is noticeable that I also implement a gear shifting algorithm in this section. In this case, I set μ to be 0.5 and order number to be 20.

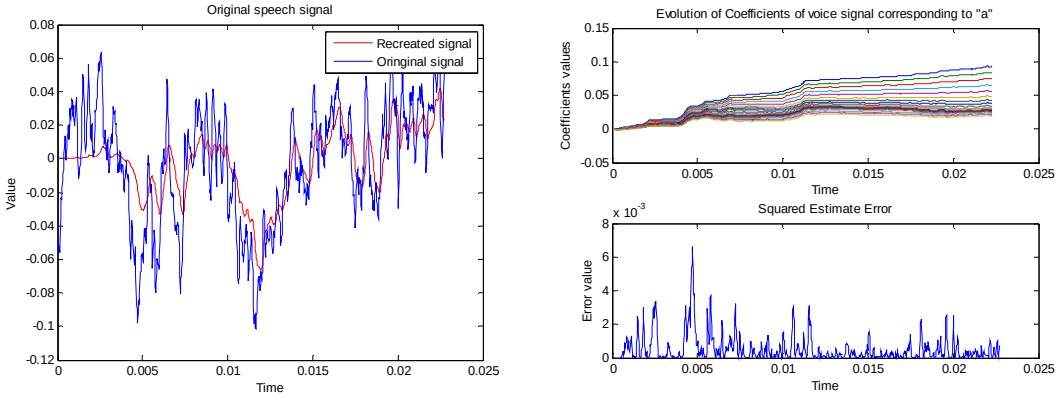


Fig. 7 Left: 1000-sample Original and recreated signal of voice”a”. Right: Evolution of coefficients and estimate error of voice "a"

2. I also test the algorithm using voice “a” with different number of order.

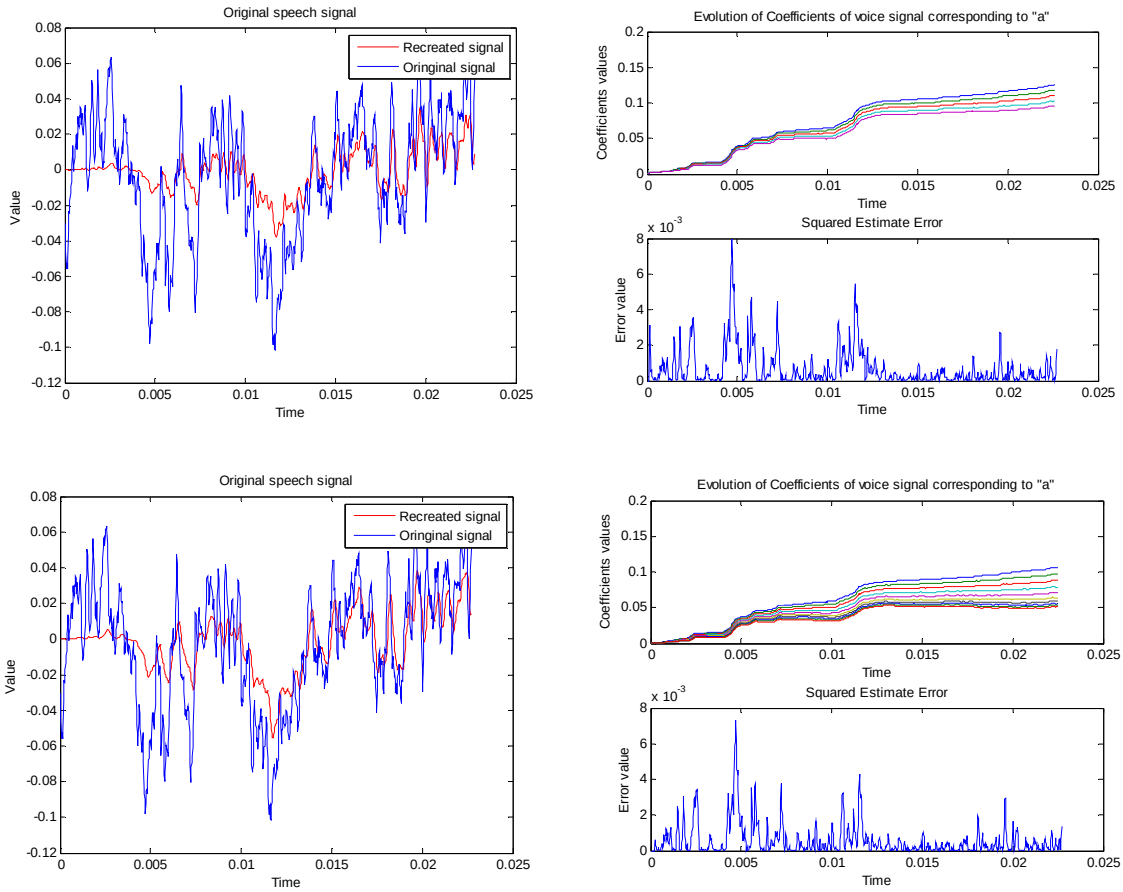


Fig. 8 Top left: 1000-sample Original and recreated signal of voice "a" with order number of 5. Top right: Evolution of coefficients and estimate error of voice "a" with order number of 5. Bottom left: 1000-sample Original and recreated signal of voice "a" with order number of 10. Bottom right: Evolution of coefficients and estimate error of voice "a" with order number of 10.

The above graphs show that, for larger number of order, the recreated signal is more similar with original signal. Also, larger order number produces more coefficients and requires more computational power whereas smaller order number reduces the number of coefficients and improves efficiency. Therefore, suitable order number is a trade-off between efficiency and convergent speed. It is also noticeable that $\boldsymbol{\mu}$ should be further changed for each voice file and different 1000-samples obtained from different part of a particular voice file may need different $\boldsymbol{\mu}$. Generally, larger $\boldsymbol{\mu}$ is able to produce the recreated signal more similar in shape with original signal. Hence, larger $\boldsymbol{\mu}$ (below a certain maximum value) and larger order number can produce a more detailed recreated signal, but larger order number requires more computational power. Hence, I can choose a relatively small order value but a larger value of $\boldsymbol{\mu}$.

3. To further examine the performance of the adaptive filter, I calculate prediction gain of each voice signal. By trying different combination of order number and μ, I obtain suitable μ values for each voice signal while setting order number to be 10 for all signals. That is mainly because the order number of 10 gives me an acceptable convergent speed while requiring less computational power. It is also noticeable that a larger μ can increase prediction gain and recreated signal are more similar in shape with original signal. That is because larger prediction gain means less error and larger μ can reduces the error. Hence, suitable μ is the maximum available value that makes prediction gain below 25.

Corresponding Voice File	μ	R
a	0.6	24.296
e	0.2	24.398
s	0.3	24.468
t	0.07	24.918
x	0.4	23.92

Table 3 Table of suitable adaptation gains and results prediction gain for different voice file

4.6 Dealing with computational complexity: sign algorithm

In this section, I implement sign algorithms into section 4.4 and 4.5 according to the notes. I also test their performance by examining the convergence properties and accuracy of the sign algorithm. For section 4.4, the improvement brought by sign algorithm is tiny but the improvement in section 4.5 is significant. The table below demonstrates the improvement that sign algorithm brings.

Section 4.4 with adaptation gain of 0.02				
Algorithm	Basic	Signed-error	Signed-regressor	Signed-signed
No. of sample when convergence starts	300	500	500	600
Accuracy	/	Same as basic	Same as basic	Better than basic

Table 4 Table of performance of sign algorithm for Section 4.4

Section 4.5 using speech voice "a" with adaptation gain of 0.02				
Algorithm	Basic	Signed-error	Signed-regressor	Signed-signed
No. of sample when convergence starts	200	250	250	50
Accuracy	/	A lot better than basic	A lot better than basic	Best

Table 5 Table of performance of sign algorithm for Section 4.5

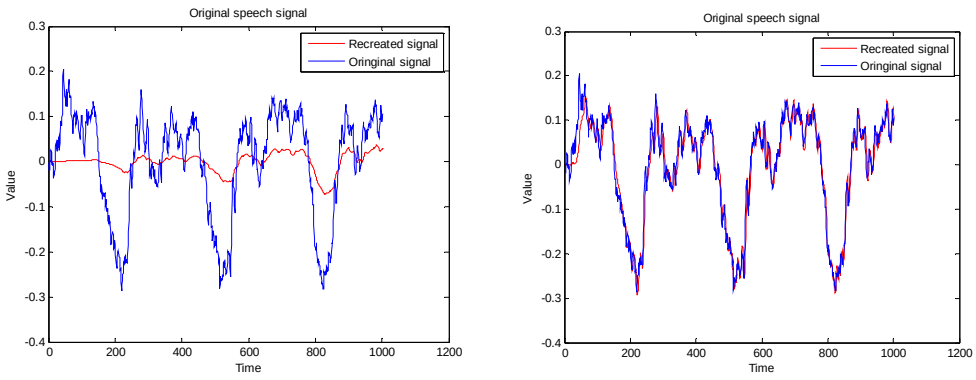


Fig. 9 Left: Recreated signal and original signal generated by basic LMS with adaptation gain of 0.01 and order of 10. Right: Left: Recreated signal and original signal generated by sign-sign LMS with adaptation gain of 0.01 and order of 10.

From the above graphs, it is clear that the recreated signal is more similar in shape with the original signal and the improvement is huge. Also, it is noticeable that the number of sample when it starts convergence becomes smaller than basic algorithm.

Coursework 5

1. In this section, I load the Vinyl music signal into MATLAB and analyse its corrupted signal. I first of all use **sound** command to listen to the music signal, and I find out that there is irregular tickling in the signal. To further examine the signal, I draw periodograms of both channel.

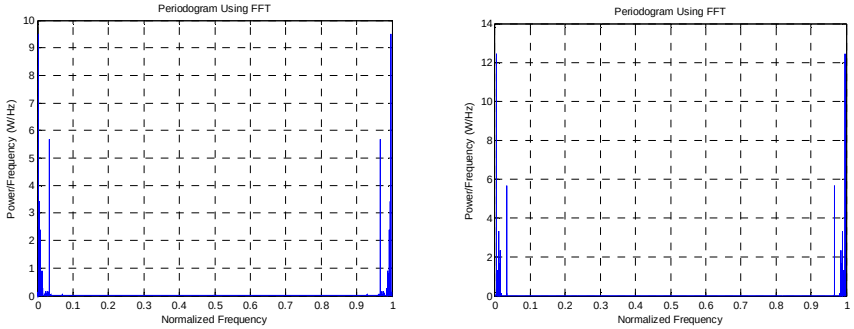


Fig. 4 Left: Periodogram of noise corrupted signal of channel 1. Right: Periodogram of noise corrupted signal of channel 2

From the above graphs, it is impossible to tell exactly which components correspond to noise or signals. That is because we do not have enough information about periodogram of original signals. However, according to the listening, I can roughly tell that one of the main noise components locates at frequency lower than most of signal frequency components because this component corresponds to time between 2 adjacent tickling. Another main component locates at relatively high frequency which corresponds to the internal frequency of any specific tickling.

2. After obtaining an original signal of Vinyl music, I can compare it with previous noise corrupted signal.

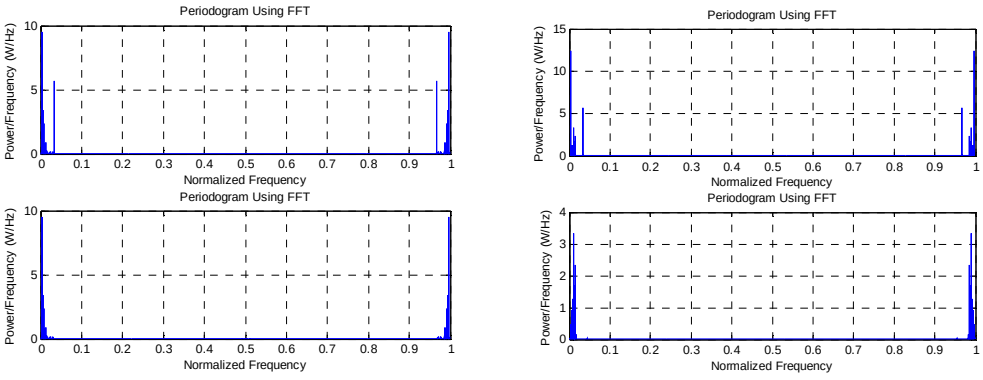


Fig. 5 Top left: Periodogram of noise corrupted signal of channel 1. Top right: Periodogram of noise corrupted signal of channel 2. Bottom left: Periodogram of original signal of channel 1. Bottom right: Periodogram of original signal of channel 2.

Compare the bottom 2 graph with the top 2 graph, it is clear that there are 2 main components of noise at 200 Hz, 1500 Hz for both channel 2 and 1 at 1500 Hz for channel 1.

3. According to above observations, I apply an IIR notch filter to filter out the specific frequency components at 200 Hz and 1500 Hz for channel 2 and 1500 Hz for channel 1. IIR notch filter is chosen because noise only occurs at these 2 frequencies and there are also other wanted components around 200 Hz noise components. Therefore, it will be good if I can only eliminate the components at 200 and 1500 Hz. Hence, the IIR notch filter is a good choice. By implementing IIR notch filter, I also need to specify the bandwidth at the -3 dB points which is equal to notch frequency divided by a Q factor. If the Q factor is large, the notch will be sharp and vice versa. By carefully examining the periodogram of signal, I find out that there still exist some other noise components with smaller values surrounding the main noise components. Therefore, I choose Q factor to be small to remove these smaller noise components as well. After implementing this filter, the noise has been significantly reduced.

4. By using the equation in the coursework notes, I calculate error of my estimate with respect to original signal. The signal of channel 1 has relative error of 0.0192 and signal of channel 2 has error of 0.2366. A good relative error should be less than 1 because if the error is greater than 1, it means that the estimate signal is larger in power than original signal. That extra power of estimate signal is due to the power of noise. Also, the smaller the relative error, the better. The periodograms of original, noise corrupted and filtered signal are shown below

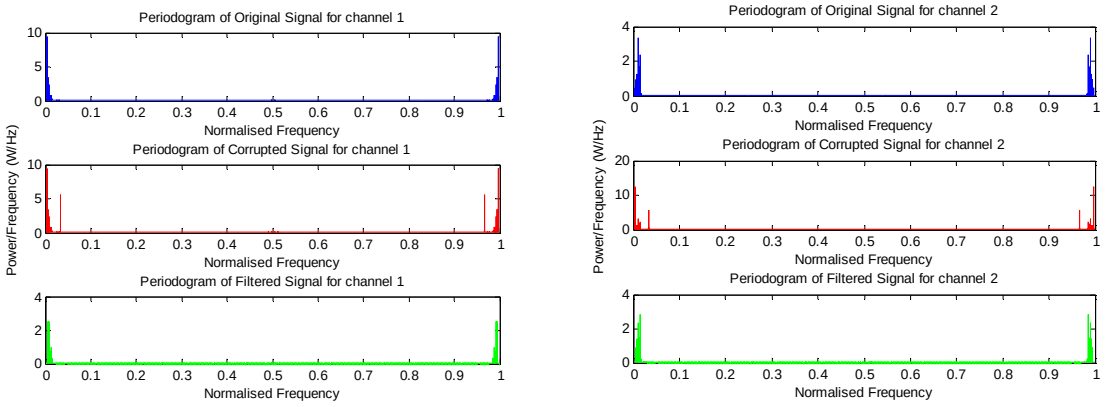


Fig. 6 Left: periodograms of original, noise corrupted, filtered signals of channel 1. Right: Periodograms of original, noise corrupted, filtered signals of channel 2.

From the above graphs, it is clear that the filtered signal periodograms are very similar to original signal periodograms and errors are relatively small and acceptable.

5. In this question, I modify the function **lms()** in Coursework 4 and use it to filter the Vinyl music signal. By testing the performance of LMS and NLMS, I set the adaptation gain to be 0.1 and order number to be 10.

μ =0.1, order=10	NLMS	LMS

Relative Error	Channel 1	0.2520	0.2191
	Channel 2	0.2851	0.3519

Table 1 Table of relative errors

By comparing the relative error (as in part 4), I find out that NLMS has better performance for channel 2 and LMS has better performance for channel 1. However, the differences of performance are tiny. I plot the signal and squared error for NLMS and LMS as shown below.

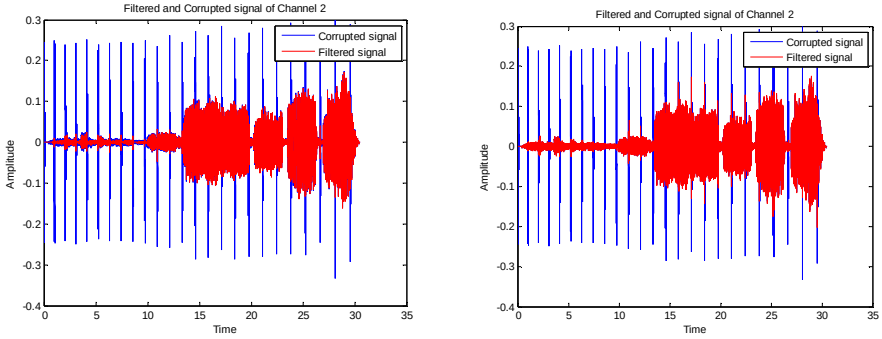


Fig. 4 Left: Noise corrupted and filtered signals of channel 2 with LMS. Right: Noise corrupted and filtered signals of channel 2 with NLMS.

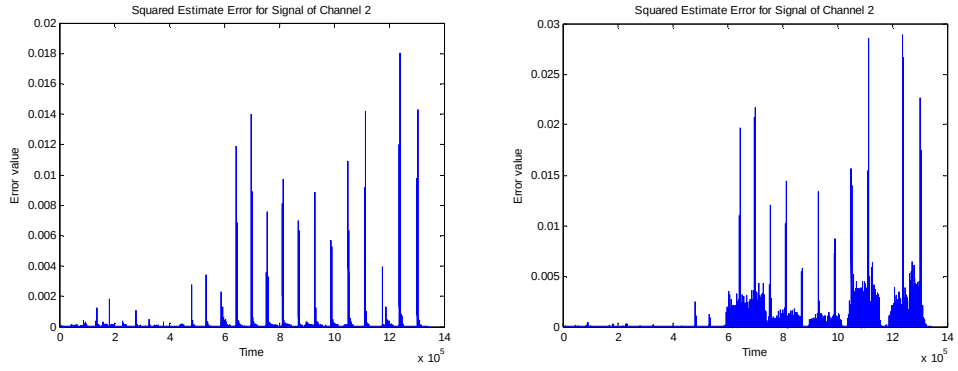


Fig. 5 Left: Squared error of noise corrupted and filtered signals of channel 2 with LMS. Right: Squared error of noise corrupted and filtered signals of channel 2 with NLMS.

From Fig. 4, the noise in the corrupted signal has been significantly reduced and the output filtered signals for both LMS and NLMS are very similar in shape to the original signal. Compared to LMS, even though output of NLMS has better shape and is more similar to original signal, there still exists some noise in output of NLMS. The Fig. 5 can also proof this result since the squared error of NLMS is actually larger than LMS. Hence, LMS has better performance. Since NLMS and LMS have differences but differences are tiny, both algorithms are good at reducing noise. Therefore, I use LMS straightforward because LMS can save some computational power. To maximise the performance of LMS, adaptation gain is chosen to be 3.5 for channel 1 and 4 for channel 2 because larger adaptation gain can reduce squared error and produce better output signal. Adaptation gain of 3.5 and 4 are maximum available for channel 1 and 2. Number of orders is chosen to be 10 because changes of number of orders do not significantly improve the performance while larger order number requires more computations. Hence, a smaller and acceptable number of

order is 10. In this case, the error (as in Part 4) is 0.1005 for channel 1 and 0.1795 for channel 2.

6. In this question, I need to filter the noise in another music data in matrices **um**. By analysing the periodogram of the original and noise corrupted signal, I find out that, similar to **s2h** signal, **um** signal has noise components at 200 and 1500 Hz for channel 2 and 1500 Hz for channel 1. In order to choose between LMS and NLMS, I set adaptation gain to be 0.1 and order to be 10 and calculate the relative error.

μ =0.1, order=10		NLMS	LMS
Relative Error	Channel 1	0.4646	0.1019
	Channel 2	0.4404	0.1517

Table 2 Table of relative errors

The table above show that the LMS has way better performance than NLMS, so I choose LMS filter. It is also noticeable that the both NLMS has better performance in the previous case but LMS in the previous case perform worse than this case. The signal of channel 2 is shown below.

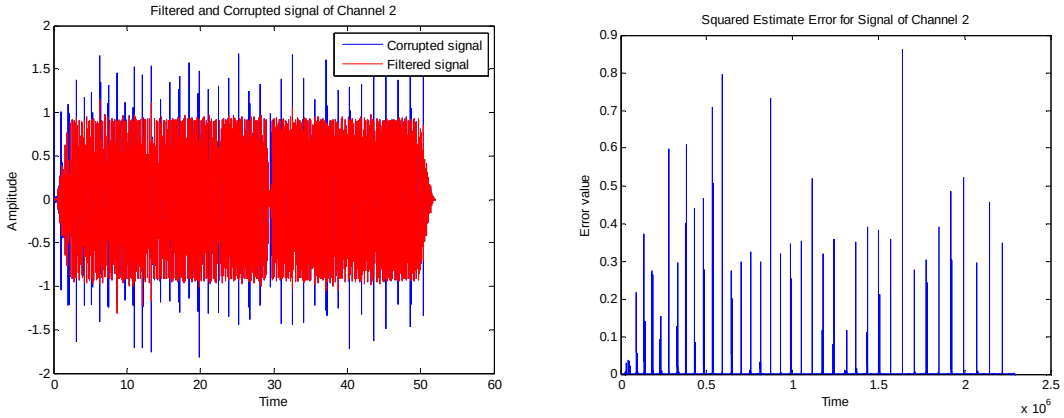


Fig. 6 Left: Noise corrupted and filtered signals of channel 2 with LMS. Right: Squared error of noise corrupted and filtered signals of channel 2 with LMS.

In this case, the maximum adaptation gain is reduce to approximately 0.5 for channel 1 and 0.3 for channel 2, so I set adaptation gain to 0.5 and 0.3 to maximise the LMS performance. And the number of order is kept 10. The performance measurements (as in Part 4) are 0.0869 for channel 1 and 0.1350 for channel 2. By carefully examining periodogram, the noise components at 200 Hz and 1500 Hz of channel 2 are removed while keeping other frequency components almost unchanged. However, noise at 1500 Hz of channel 1 is reduced rather than removed.

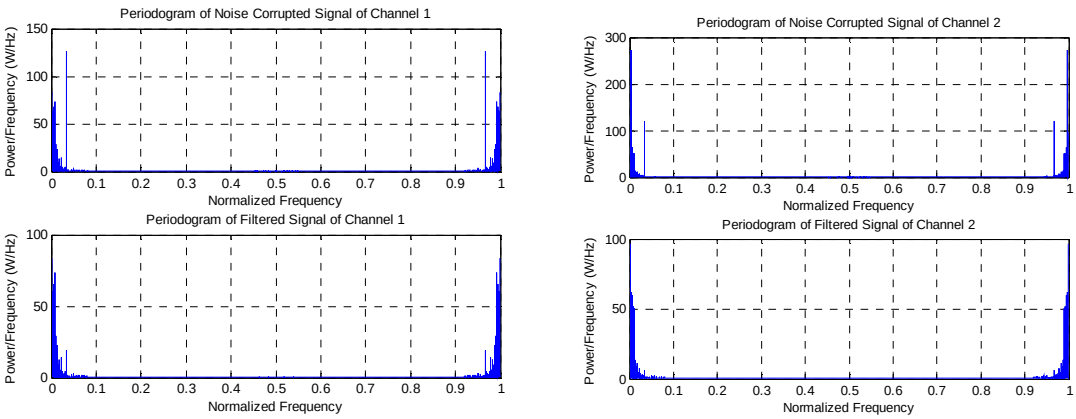


Fig. 6 Left: Periodograms of Noise corrupted and filtered signals of channel 1. Right: Periodograms of Noise corrupted and filtered signals of channel 2.