Advanced Signal Processing

Hao Ding

CID 00734091

Login hd1812

CONTENTS

1. Random signals and stochastic processes

1.1 Statistical estimation

Using MATLAB command ‘rand’, 1000 samples of a uniform random variable X ~ U (0, 1) are generated and plotted in Figure 1.

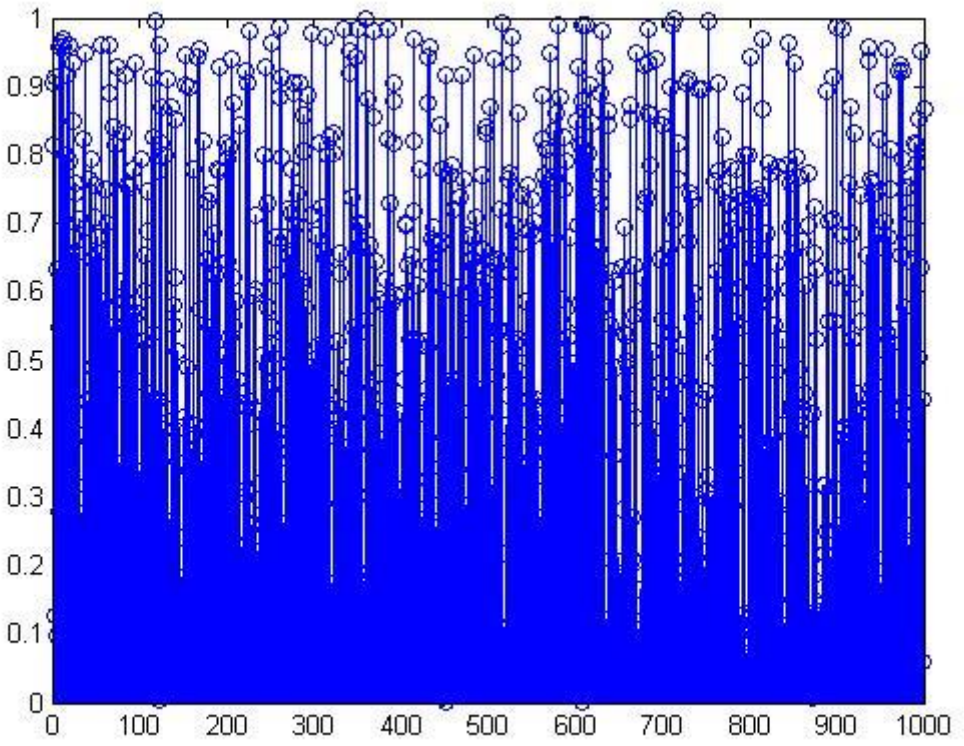

Figure 1. 1000 samples of uniform distribution

Although all samples have stochastic nature, they show uniformity since they are time-invariant and statistically stationary. Each sample is also referred as a realisation of the random process: X ~ U (0, 1), ∀n.

1.1.1 The theoretical mean, m = E{X}, of a uniform distribution between (0, 1) is 0.5. Using MATLAB function ‘mean’ to calculate sample mean, which performs as:

$$\widehat{m} = \frac{1}{N}\sum_{n=1}^{N} x[n],$$

MATLAB code:

```
N=1000;
x=rand(N,1);     %create a matrix of 1000*1
A=mean(x);       %Sample mean
```

Result:

A=0.488

Sample mean calculated is has a deviation 2.4% from theoretical value and it is a good approximation of theoretical mean. However, different set of samples from the same distribution can have different sample average. Increasing sample number will lead to a more accurate estimator of theoretical average.

1.1.2 Similar to 1.1.1, theoretical and sample standard deviation values are compared. MATLAB function 'std' computes sample standard deviation using the formula:

$$\hat{\sigma} = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(x[n]-\hat{m})^2}.$$

Whilst expected standard deviation can be derived from the expression:

$$\sigma = \sqrt{\mathbb{E}\{X-\mathbb{E}\{X\}\}^2}$$

For a uniform distribution, theoretical standard deviation is:

$$\boldsymbol{\sigma = \frac{b-a}{\sqrt{12}}}$$

(b: higher boundary, a: lower boundary)

This can be calculated by taking integrals in pdf and in this example, result is approximately 0.2887.

MATLAB code:

```
-    std_devs=std(x);    %calculate sample deviation
```

Result:

Std_devs=0.2853

Sample value is 1.12% from the theoretical value. Therefore, deviation taken from 1000 realisation is a good estimator of theoretical deviation. Similar to sample mean, larger sample number results in a better estimation.

1.1.3 Generate ten 1000-sample realisations and compare plot their bias of mean and standard deviation.

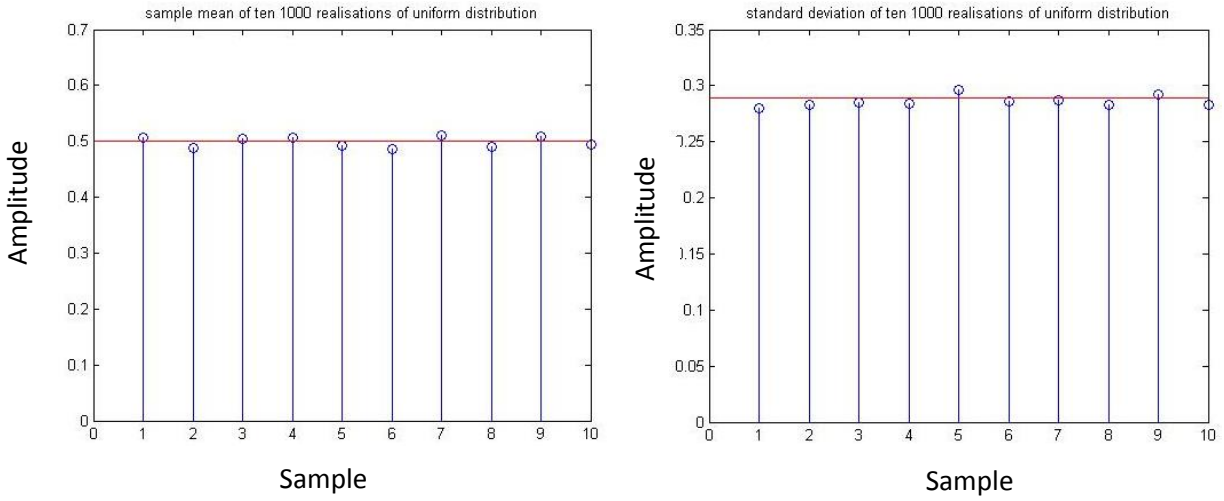

Figure 2. Ten 1000-samples realisations. Sample mean and standard deviation

Bias is defined as $B = \mathbb{E}\{X\} - \widehat{m}.$ Figure 2 illustrates that sample values are clustered closely around theoretical value. Therefore, bias is expected to be small, as shown below.

1.1.4 Use MATLAB to generate pdf of uniform distribution.

MATLAB code:

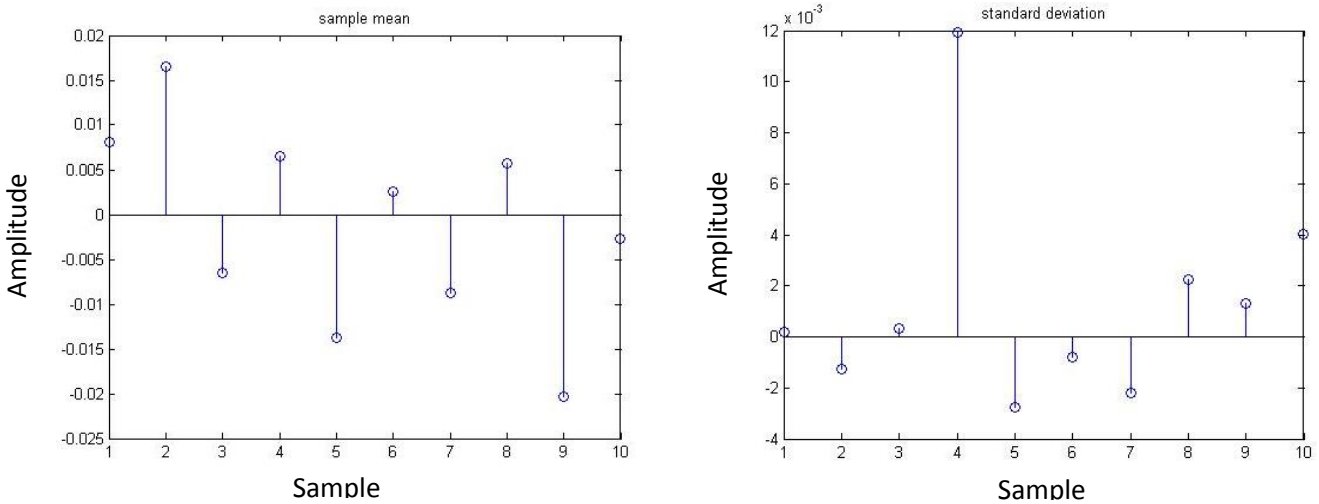

Figure 3. Bias of ten 1000-samples realisations. Sample mean and standard deviation

```
%y is assigned to pdf of uniform distribution
y=unifpdf(i,0,1);

%total area is sum(hfreq)*(1/numofbins). After normalisation, total area
%under pdf is 1.
bar(centre,hfreq/sum(hfreq)*numofbins);hold on
plot(i,y,'r');hold off
```

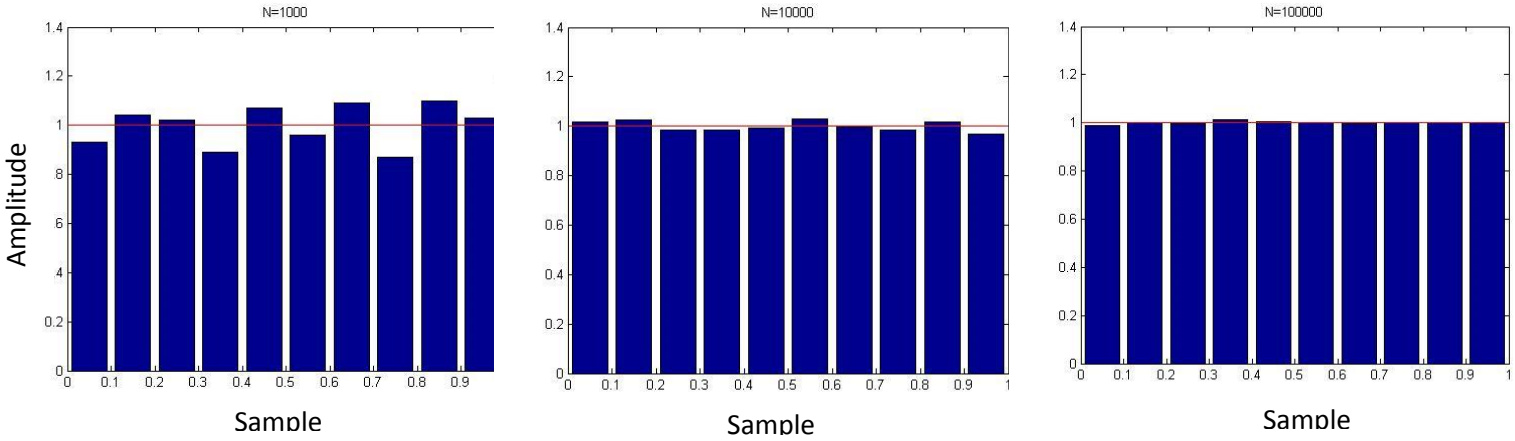

Figure 4. Pdf of uniform distribution for different N. (red line: theoretical value)

Figure 4 shows that the estimate converge as sample number increases. Regarding number of bins, if histogram is not normalised by number of bins, average height is inversely correlated to bin number. But larger bin number provides more details about the sample number in narrow intervals

1.1.5 Repeat the process from 1.1.1 to 1.1.4 using 'randn' function and investigate the properties of random normal distribution.

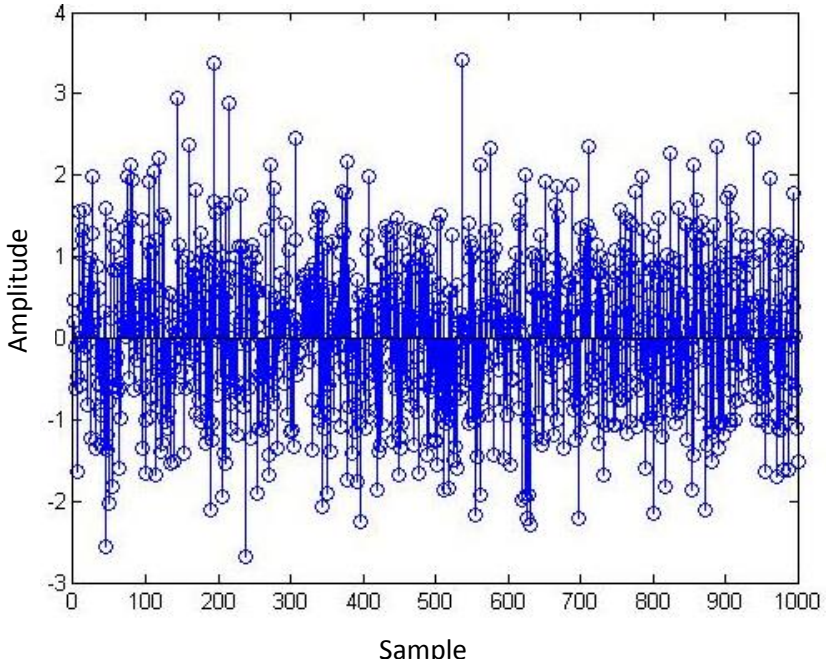

Figure 5. 1000 samples of Gaussian distribution

Theoretical mean:	0	Theoretical deviation:	1
Sample mean:	0.0155	Sample deviation:	1.0002

In this 1000-sample realisation, mean and deviation are good estimator and they are within 1% deviation of theoretical value. Larger sample number intended to reduce the estimation error.

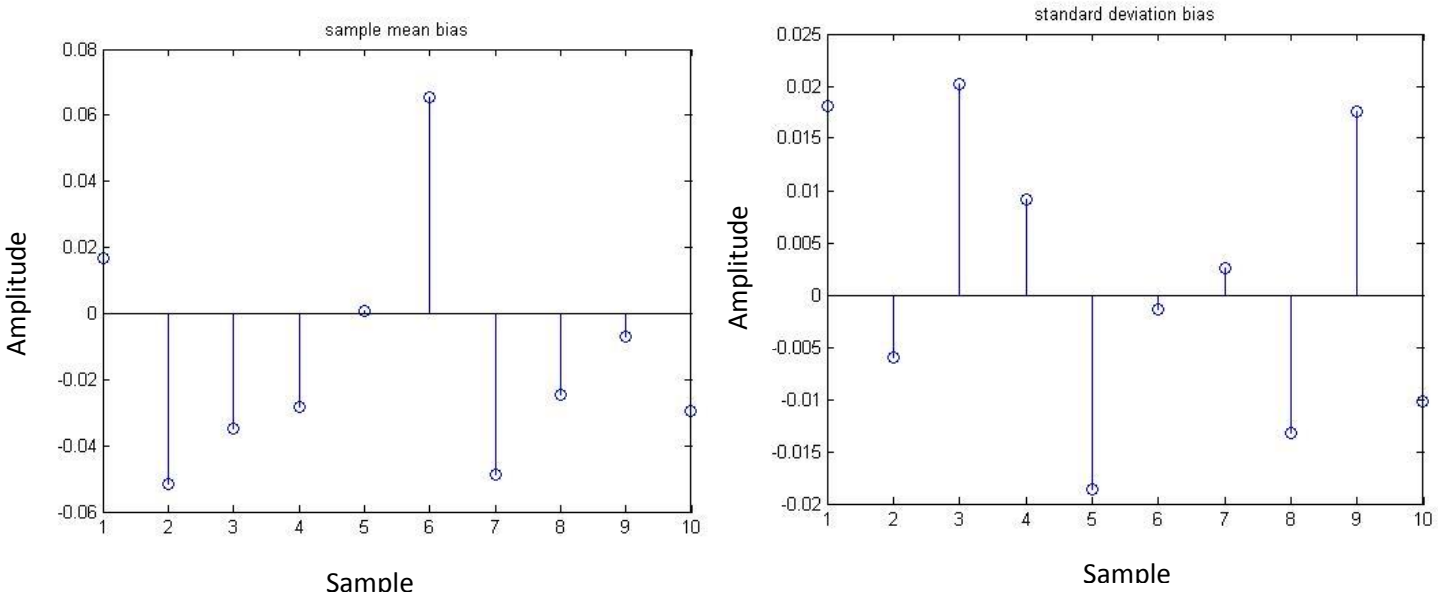

Figure 6. Bias of ten 1000-samples realisations. Sample mean and standard deviation

Therefore, bias of mean and sample estimator is expected to be small. Figure 6 shows the bias of ten 1000-sample realisations and proves the accuracy of estimators.

PDF of Gaussian distribution

```
Norm=randn(N,1);
[hfreq,centre]=hist(Norm,numofbins);
g=1/sqrt(2*pi)*exp(-0.5*centre.^2);

figure
bar(centre,hfreq/trapz(centre,hfreq));hold on
plot(centre,g,'r');hold off
```

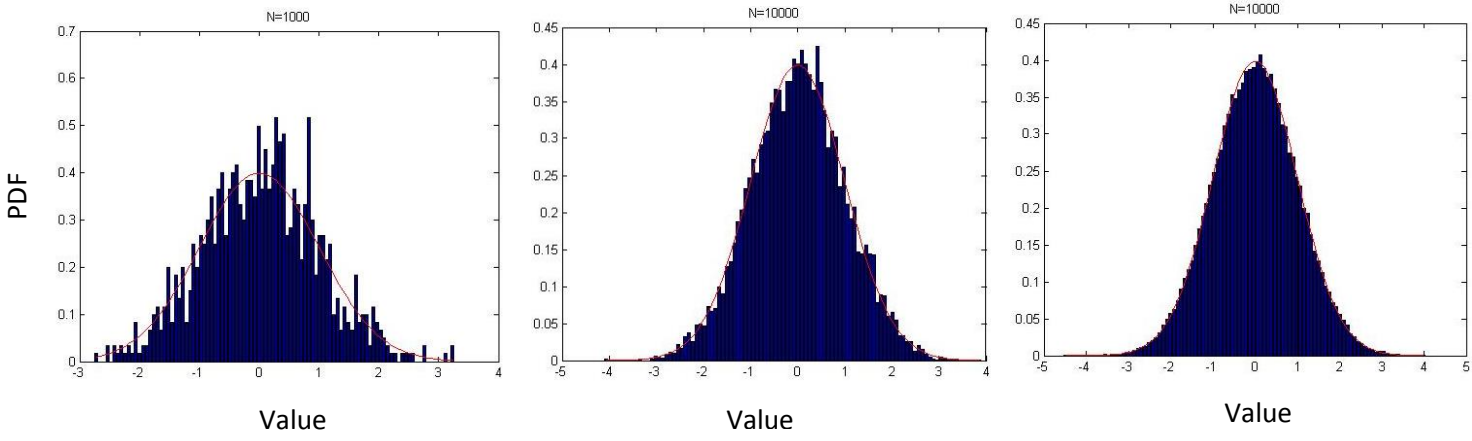

Figure 7. PDF of Gaussian distribution when N=1000, 10000,100000. (Bin Number is 100)

According to Figure 7, pdf converges as number of sample increases. Larger bin numbers provides a detailed view of small interval as well as a general trend.

1.2 Stochastic process

1.2.1 Stationarity

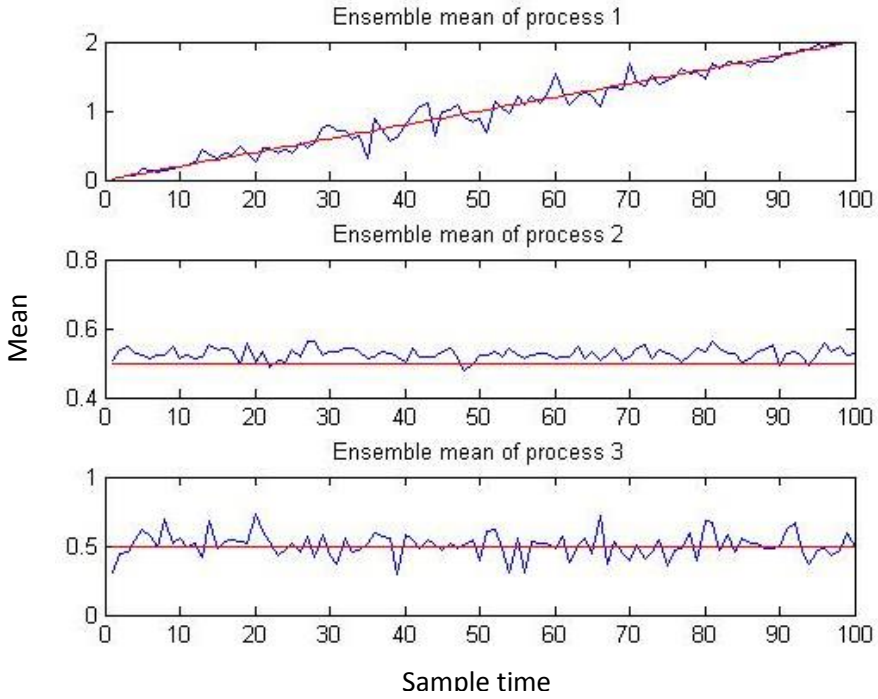

Figure 8. Ensemble mean for three processes (red line: theoretical value, proved later)

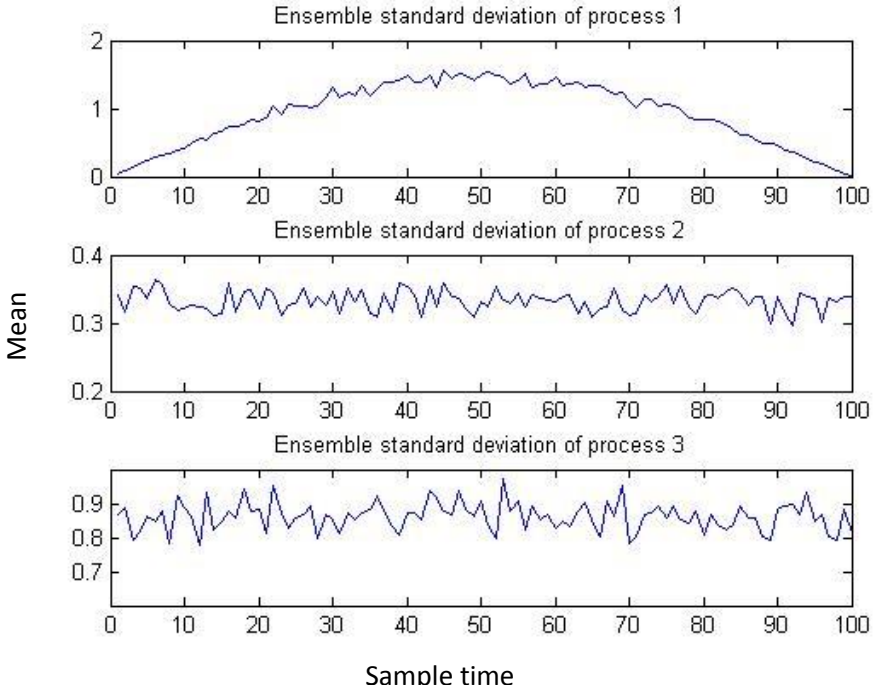

Figure 9. Ensemble standard deviation for three processes

Stationary processes can be categorised into Wide Sense Stationary (WSS) and Strict Sense Stationary (SSS). For WSS process, which has less restrictions, time average is constant and covariance only depends on time difference.

Figure 8 shows that process 1 is not stationary and process 2, 3 are stationary, since ensemble mean of first process varies with time.

1.2.2 Ergodicity

A stochastic process is ergodic if its properties, including mean and standard deviation, can be deduced from a sufficiently long sample.

Time average:

Process Number	Sample 1	Sample 2	Sample 3	Sample 3
1	9.963	10.016	10.035	9.987
2	0.435	0.167	0.999	0.717
3	0.518	0.472	0.444	0.440

Standard deviation:

Process Number	Sample 1	Sample 2	Sample 3	Sample 3
1	5.885	5.845	5.843	5.879
2	0.015	0.158	0.179	0.057
3	0.875	0.869	0.869	0.860

Process 1 is not ergodic because it is not a stationary process.

Process 2 is not ergodic since time average is not determinable.

Process 3 is ergodic.

1.2.3 Mathematical Description of Sample Stochastic Processes

Process 1

Mathematical Expression: $5 * \text{rand[n]} * \sin\left(\text{n} * \frac{\pi}{N}\right) + 0.02n$

rand[n] ~ U (-0.5 , 0.5)

Ensemble mean: $\text{E}\left(5 * \text{rand[n]} * \sin\left(\text{n} * \frac{2*Pi}{N}\right) + 0.02n\right) = \text{E}(0.02n)$

= 0.02 * n

Ensemble variance: Let $a[n] = 5 * \text{rand}[n] * \sin\left(n * \frac{2*Pi}{N}\right)$,

$b[n] = 0.02n$

$\text{Var}(a[n]+b[n])$ = $\text{Var}(a[n])+\text{Var}(b[n])$, a[n] and b[n] are independent at given n

$\text{Var}(a[n])$ = $E(a^2[n]) - E^2(a[n])$

= $E\ (25*\text{rand}^2\ [n]*\sin^2\ (n\pi/N)) - 0$

= $\frac{25}{12} \sin^2\ (n\pi/N)$

$\text{Var}(b[n])$ = 0

Process 2

Mathematical Expression: **(**a[n]-0.5)*b[n] + c[n]

a[n] ~ U (0 , 1), b[n] ~ U (0 , 1), c[n] ~ U (0 , 1)

Ensemble Mean: E(a[n]-0.5)*b[n] + c[n]) = 0.5

Ensemble Variance:

Var**(**a[n]-0.5)*b[n] + c[n]) = Var (a[n]-0.5)*b[n]) + Var**(**c[n]**)**

= Var $(a[n] - 0.5)E^2(b[n])$ + Var (b[n]) E^2(a[n]-0.5) + Var $(a[n] - 0.5)$ Var (b[n])+ Var**(**c[n]**)**

$= \frac{1}{12} \times 0.5^2 + \frac{1}{12} \times 0 + \frac{1}{12} \times \frac{1}{12} + \frac{1}{12}$

$= \frac{1}{9}$

$\sigma = 1 / 3 \approx 0.333$

Process 3

Mathematical Expression: (a[n]-0.5)*3 + 0.5

a[n] ~ U (0 , 1)

Ensemble Mean: E (a[n]-0.5)*3 + 0.5) = 0.5

Ensemble Variance: Var(a[n]-0.5)*3 + 0.5) = 9 * Var (a[n]) = ¾

$\sigma = \sqrt{\frac{3}{4}} = 0.866$

Summary

Results of ensemble mean and variance above match the curves plotted in Figure 8 and Figure 9.

1.3 Estimation of probability distributions

1.3.1 Probability Density Function design

Function pdf takes v as input argument and plot a histogram. Bin number is chosen to 10 in this case since the sample number required to test is 100, 1000, 10000 respectively.

```
function pdf(v)
numofbins=10;          %number of bins
[hfreq,centre]=hist(v,numofbins); % create histogram
bar(centre,hfreq/sum(hfreq))
```

Pdf of a Gaussian distribution with 100 samples is shown below:

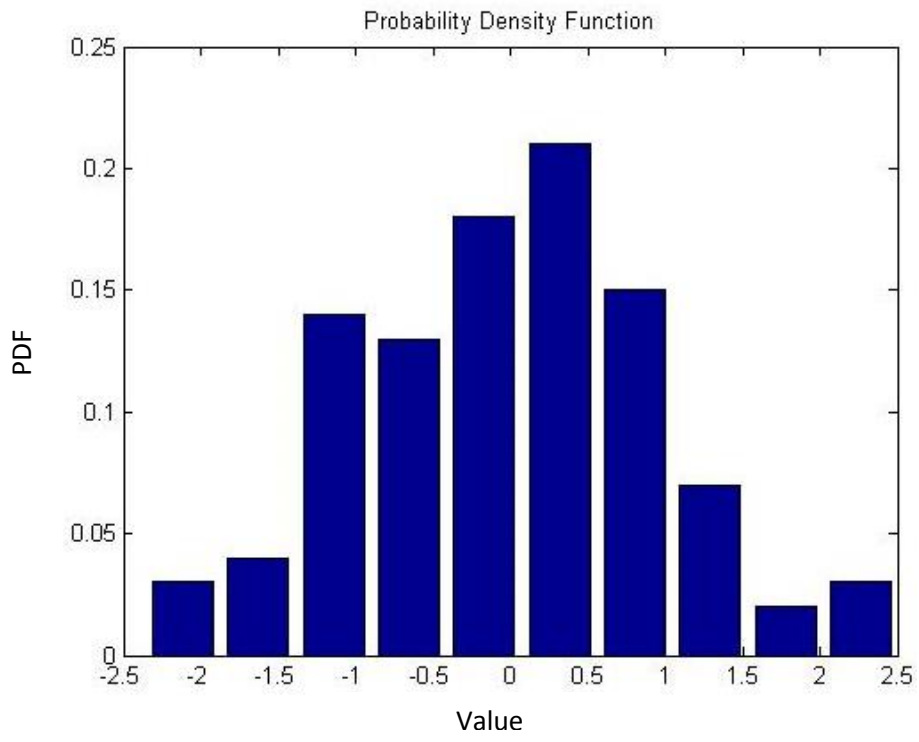

Figure 10. Pdf of a Gaussian distribution with 100 samples

1.3.2 Comparisons on pdf with different sample number

Process 1:

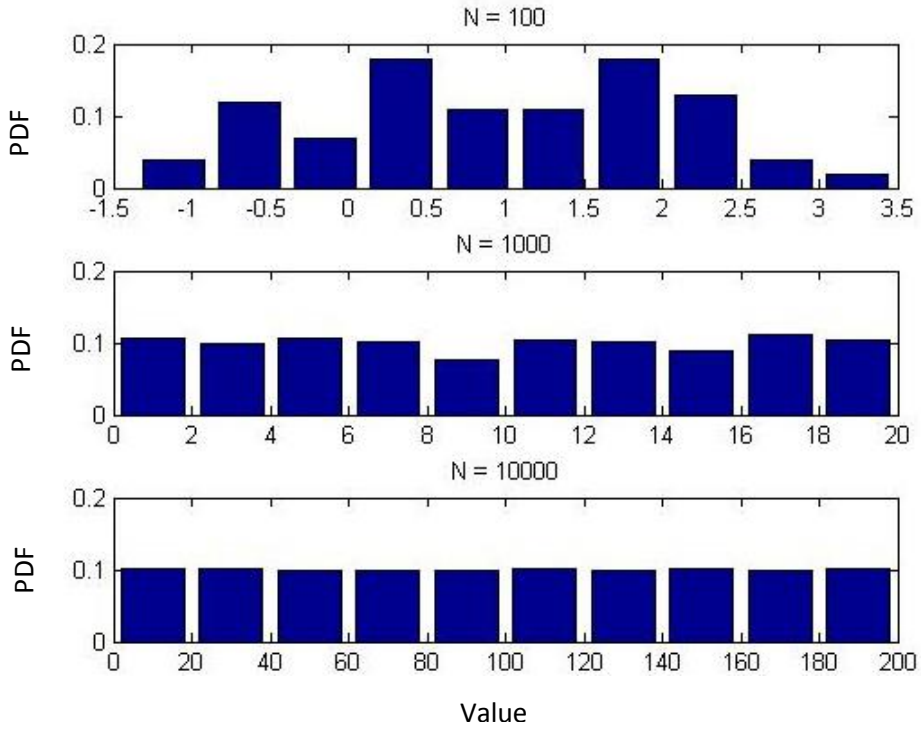

Figure 11. Pdf of a Process 1 with various sample number

Process 2:

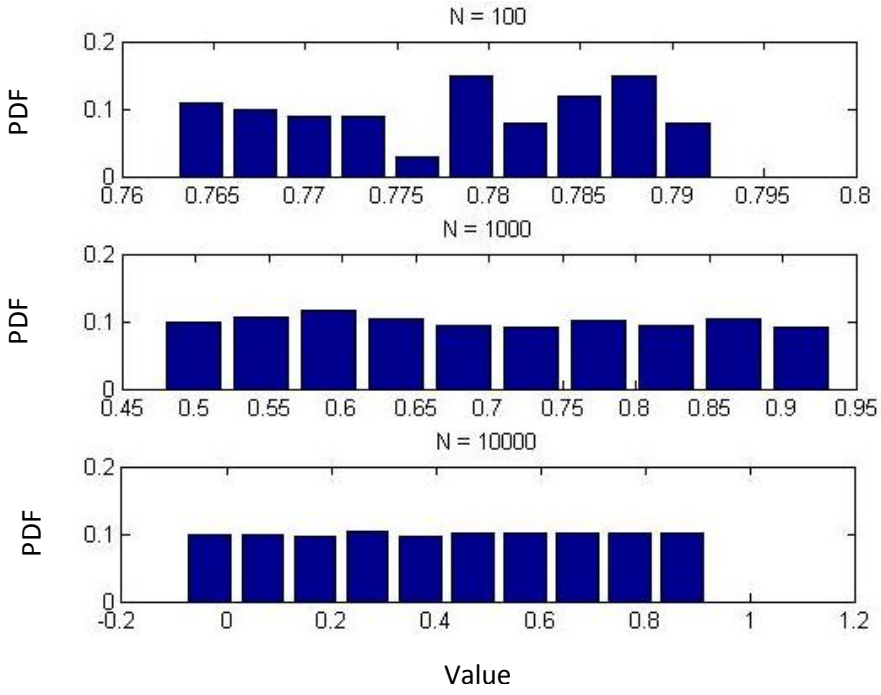

Figure 12. Pdf of a Process 2 with various sample number

Process 3:

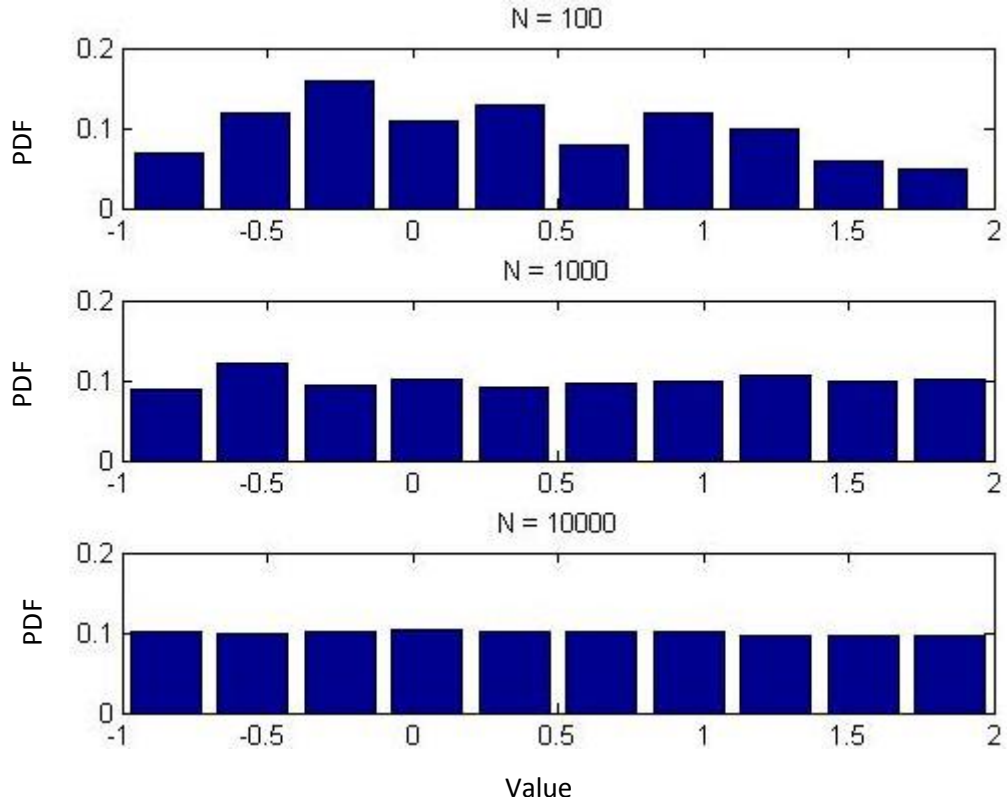

Figure 13. Pdf of a Process 3 with various sample number

As N increases, variance converges

1.3.3

It is impossible for my code to estimate the pdf of a non-stationary process since the ensemble mean varies with time. Regarding a non-stationary process whose mean of a 1000-sample-long signal rise from 0 to 1 when N=500, my code only provides pdf from limited number of samples, so the result is not valid. In order to provide a more accurate estimation, more samples should be taken, since the information contained in this 1000 samples is always restricted.

2 Linear stochastic modelling

2.1 ACF of uncorrelated sequence

2.1.1

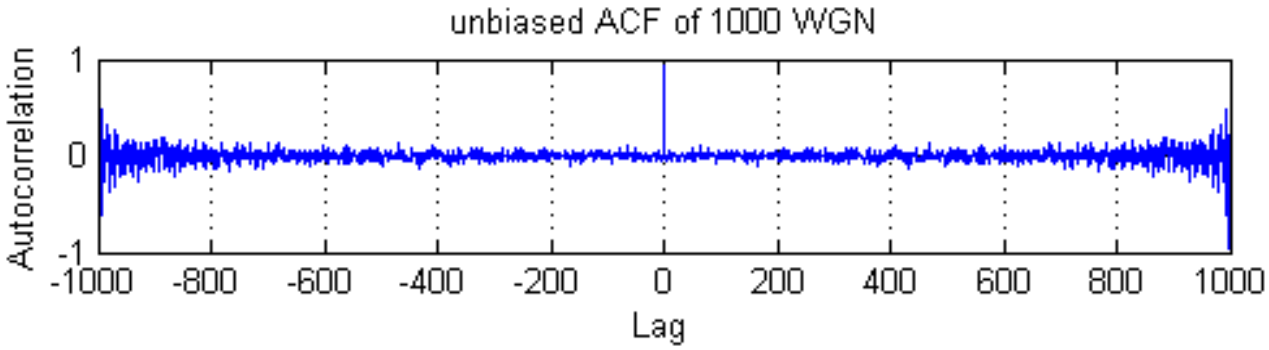

Figure 1. Unbiased autocorrelation function for a 1000-sample realisation of white Gaussian noise

It is observed that the ACF with lag zero has the highest peak in diagram since for a random signal, sample taken each time is expected to be uncorrelated with samples taken at different time. Note that ACF here are discrete values calculated from the function: $\hat{R}_x[l] = \frac{1}{N-|l|} \sum_{n=0}^{N-|l|} x[n]x[n+l]$. Also, figure 1 shows obvious symmetric property of autocorrelation due to the fact that when calculating autocorrelation at - τ or τ lags, essentially same samples are used to do calculation.

2.1.2

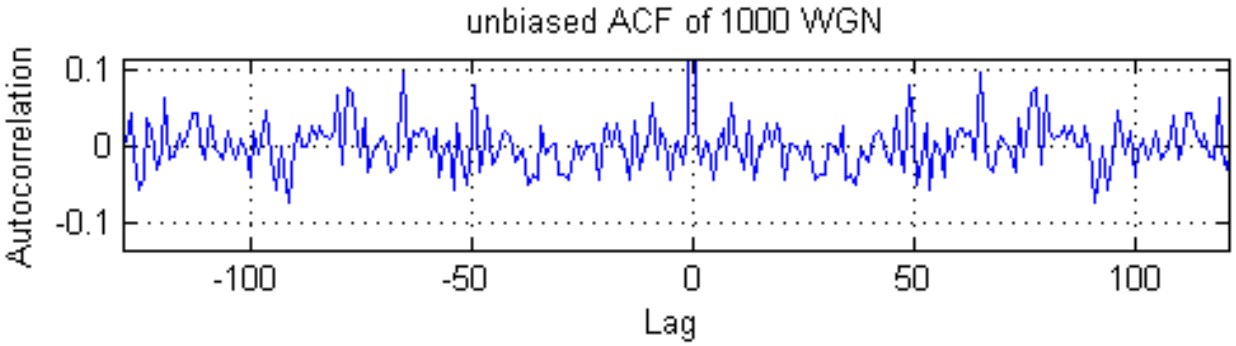

Figure 2. Zoomed version of figure 1

Figure 2 is a zoomed version to show the ACF when$|\tau| < 50$. The magnitude of ACF is almost always within -0.1 to 0.1. Comparing the figure1, autocorrelation magnitude gradually increases as absolute values of lags increase. This is due to the discrete ACF: $\hat{R}_x[l] = \frac{1}{N-|l|} \sum_{n=1}^{N-|l|} x[n]x[n+l]$. For higher lags, there are less samples available to calculate autocorrelation, leading to larger variance. Regarding small lags, sample number is large. Value calculated cancel out each other and approaches zero in average.

2.1.3

As is explained in 2.1.1 and 2.1.2, a large autocorrelation lag $|\tau|$ results in larger variance in general. Considering $\hat{R}_x[l] = \frac{1}{N-|l|} \sum_{n=1}^{N-|l|} x[n]x[n+l]\,,\ \tau = -N+1, \dots, N-1$. Total number of samples used to calculate autocorrelation is $N - |l|$. For small lags, more samples available and autocorrelation values cancel out each other and approaches zero in average. For larger lags, however, less samples are available, leading to larger variation of average result. Data at high lag numbers are therefore less reliable.

The autocorrelation value is still statistically reliable. If repeated experiments are carried out and results are taken average, correlation at high order is expected to approach zero as well. Note that at the extreme case, where only a pair of sample is available. Variance diverges but is still bounded on pdf of random sample. In our example where $[n]{\sim}N(0,1)$, maximum variance of autocorrelation is calculated as the following: $Var(\hat{R}_x[999]) = Var(x[0] * x[999]) = Var(x[0]) * Var(x[999]) = 1$.

In figure1, variance of autocorrelation is observed to increase gradually as lag number changes. It is reasonable to set the empirical boundary as a ratio of total sample number. Here, I select half of total sample number as an empirical boundary because for lags within the boundary, variance is not noticeable compared to large lags. Also the empirical boundary is easy to apply to other sample number, even in the case that sample number is small.

2.2 ACF of correlated sequences

2.1.1

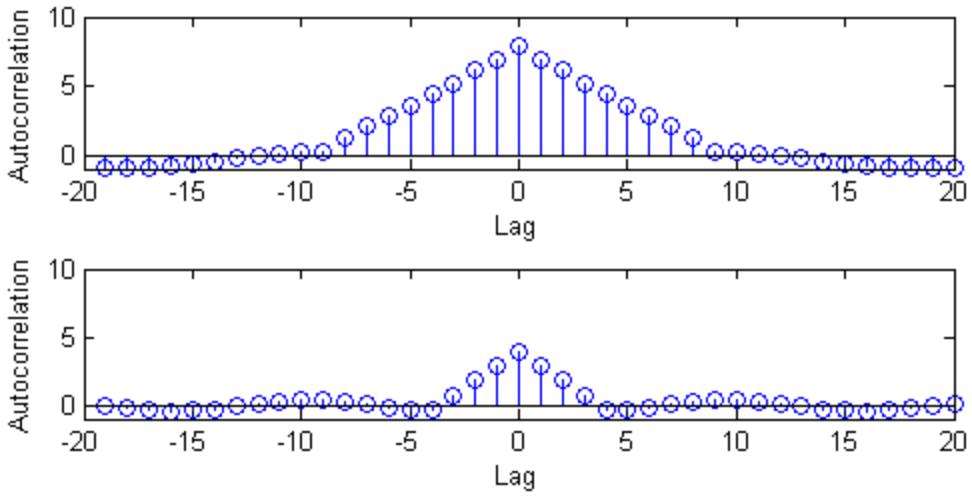

Figure 3. (Top) ACF after 9th order filter, (Bottom) ACF after 4th order filter

Figure 3 shows the ACF of 1000-sample WGN after moving average filters. A triangular shape is seen in both filters showing a linear relation with negative gradient when lag changes from 0 to N (N is order number). For a moving average filter implemented above, $\mathrm{y[n]} = \sum_{k=0}^{N-1} x[n-k]$, each value is calculated by averaging current and past N-1 samples. The smaller the lag number, the more overlapped samples used in averaging. If lag is larger than order number, outputs are no longer correlated to each other since there is no sample used as filter input. Hence the size of this triange is positively correlated to filter order.

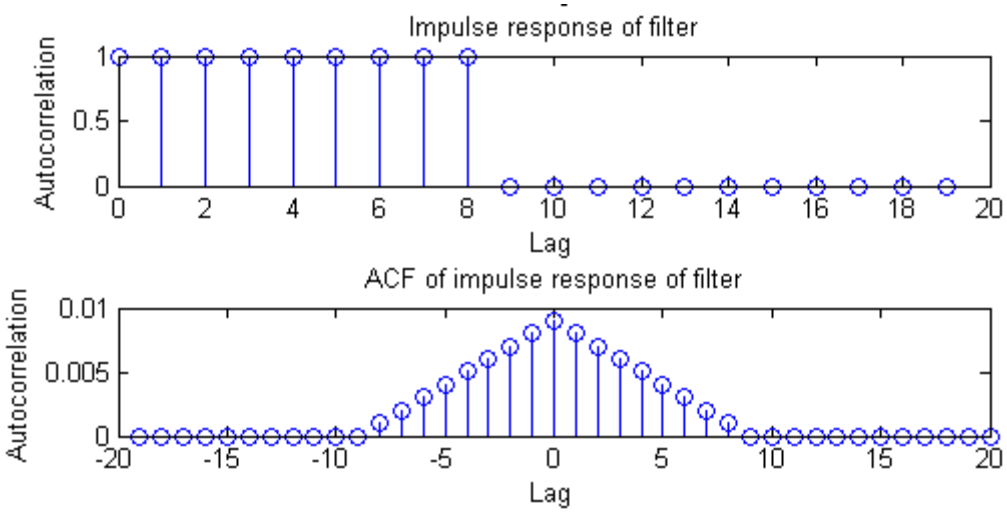

Figure 4. (Top) impulse response of 9th order filter, (Bottom) ACF of the 9th order filter

Here I plotted the impulse response of the filter which is a rectangular function and its ACF, which is a perfect triangle, whose size depends on filter order. As a result, ACF of filtered WGN is caused and dominated by the filter response with some noise added on. Theory is explained in the latter section. The filter function $y[n] = \sum_{k=0}^{N-1} x[n-k]$ is essentially calculating the local average of the input Gaussian noise.

2.2.2

The autocorrelation of filtered signal is given in the equation: $R_Y(\tau) = R_X(\tau) * R_h(\tau)$. where R_X is the autocorrelation of the input, R_X is the autocorrelation of the impulse response. R_y here represents the ACF of the filtered signal. In our example, R_x is shown in Figure 1, R_h is shown in Figure 4. The convolution of these two functions is plotted in figure 3, which agrees with our prediction. R_x is composed of an impulse at zero lag and noise in the rest part. After convoluting with the triangular function, result is a triangular with noise components added on.

2.3 Cross-correlation function

2.3.1

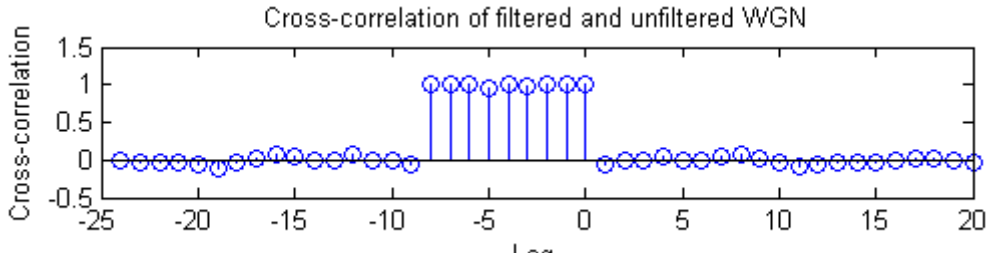

Figure 5. MATLAB plot of CCF of filtered and unfiltered WGN

It is observed from Figure 5 that the cross-correlation of signal before and after filter is close to a square wave with width N-1, N is order number 9 in this case.

Considering the cross-correlation equation: $R_{XY}(\tau) = h(\tau) * R_X(\tau)$. R_y is expected to the convolution between impulse response, which is a rectangular function and ACF of WGN, which is composed of an impulse at time zero and noise. The resultant is expected to be a rectangular function with noise, which is proved in Figure 5.

2.3.2

After passing WGN through a LTI system and calculated the cross-correlation of the output signal. System response can be approximated through the equation: $R_{XY}(\tau) = h(\tau) * R_X(\tau)$. R_{XY} is a good approximation of system impulse response if the variance of WGN is in a reasonable range.

Hence, filter order determines the effective range of CFF, which shows the shape of impulse response. In our example, larger filter order simply leads to the larger width of impulse response. After convolution, this effect is shown as extending the rectangular shape along the x-axis with the direction to negative x.

2.4 Autoregressive modelling

2.4.1

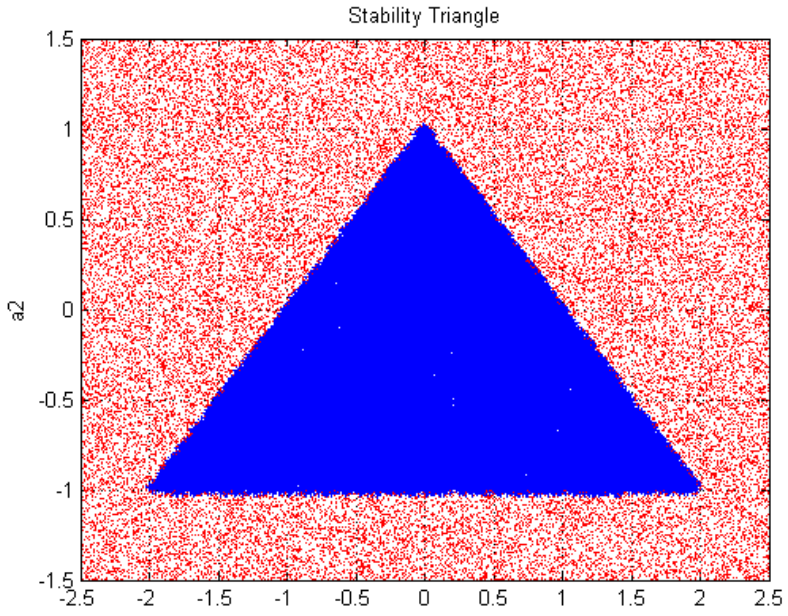

Figure 6. (Blue) variable pairs preserve stability, (Red) variable pairs cause non-stability

Figure 6 shows the MATLAB plot of stable and unstable variable pairs from 10000 samples, which gives a clear estimate than 100 samples. Resultant stable variable pairs form a triangle shape.

For an AR(2) process: $x[n] = a_1x[n-1] + a_2x[n-2] + w[n]$, stationary requires a_1 and a_2 to satisfy the following conditions: $a_1 + a_2 < 1, a_2 - a_1 < 1, -1 < a2 < 1$, which is proved in the following. In z-domain, $\mathrm{X[z]} = \frac{w[z]}{1-a_1z^{-1}-a_2z^{-2}} = \frac{z^2w[z]}{z^2-a_1z-a_2}$. Due to stability, the root should be inside the unit circle, apply this condition to $z^2 - a_1z - a_2$gives us the constraints of filter coefficients.

2.4.2

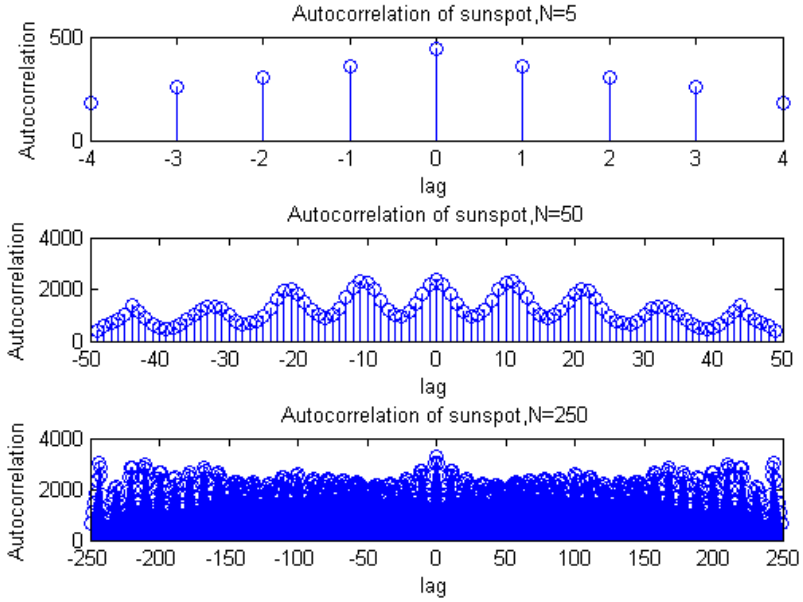

Figure 7. ACF of sunspots for N= 5, 50, 250 years

Figure 7 is the ACF of sunspots data for various periods. When N=5, some correlation is shown, but cannot summarised due to limited sample number. When N =50, data number is sufficient to illustrate the periodicity of 13 years of sunspots. When N=250, clear periodicity is still observed after zooming in. However, as year lag increases, variance of ACF rises and becoming less variable.

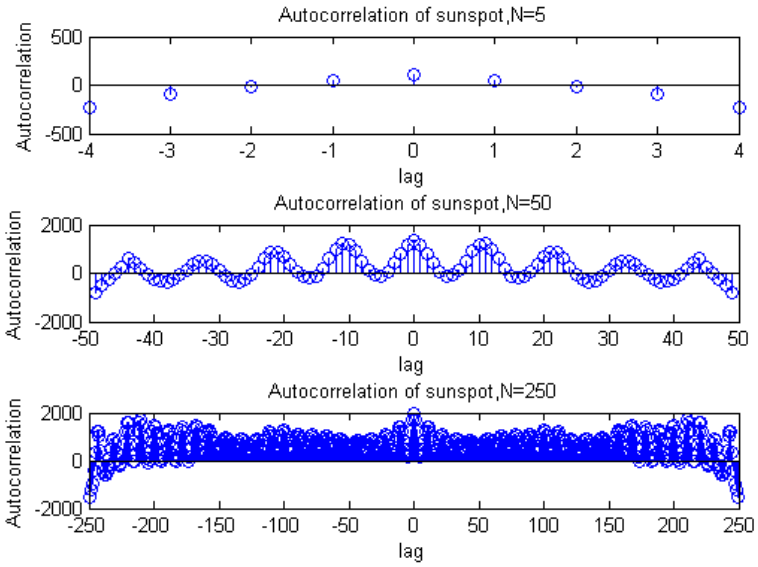

Figure 8. Zero-mean ACF of sunspots for N=5, 50, 250 years

Figure 8 shows the zero-mean autocorrelation of sunspots. It is observed that, in general, the amplitude of ACF decreases or even become negative as the input sample is centred at zero after subtracting mean. In this case ACF is now becoming auto-covariance function, which shows a clear correlation between samples. Changing in ACF shape will affects the coefficients calculated using Yule-walker equations.

2.4.3

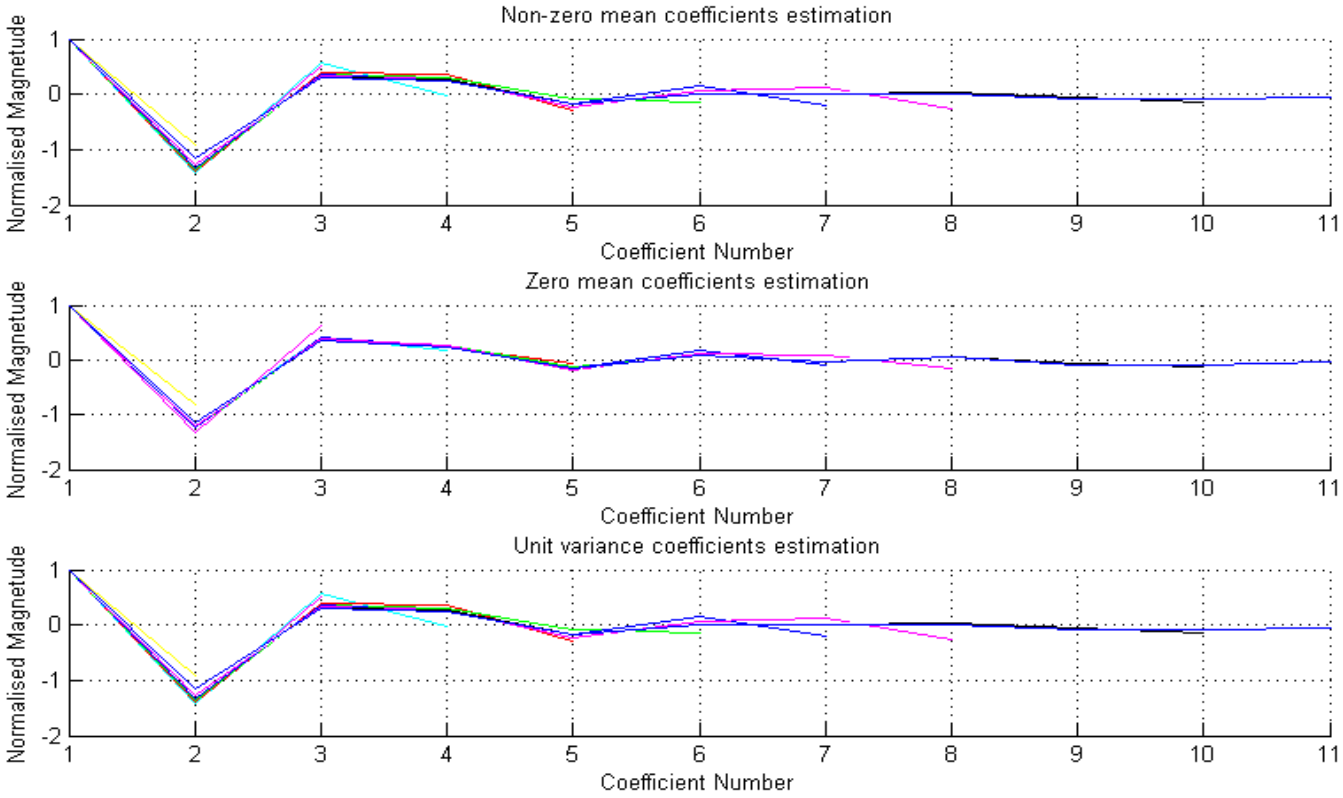

Figure 9. (Top) Non-zero mean coefficients plot. (Middle) Zero mean coefficients plot. (Bottom) Unit variance coefficients plot.

In figure 9, coefficient estimations of Yule-Walker model using non-zero mean, zero mean and unit variance input are shown on graph. Estimated coefficients are different for zero and non-zero mean input. Intuitively, solving the Yule-walker equations with different correlation values as input should lead to different results. While variance does not affect solution to YW equations

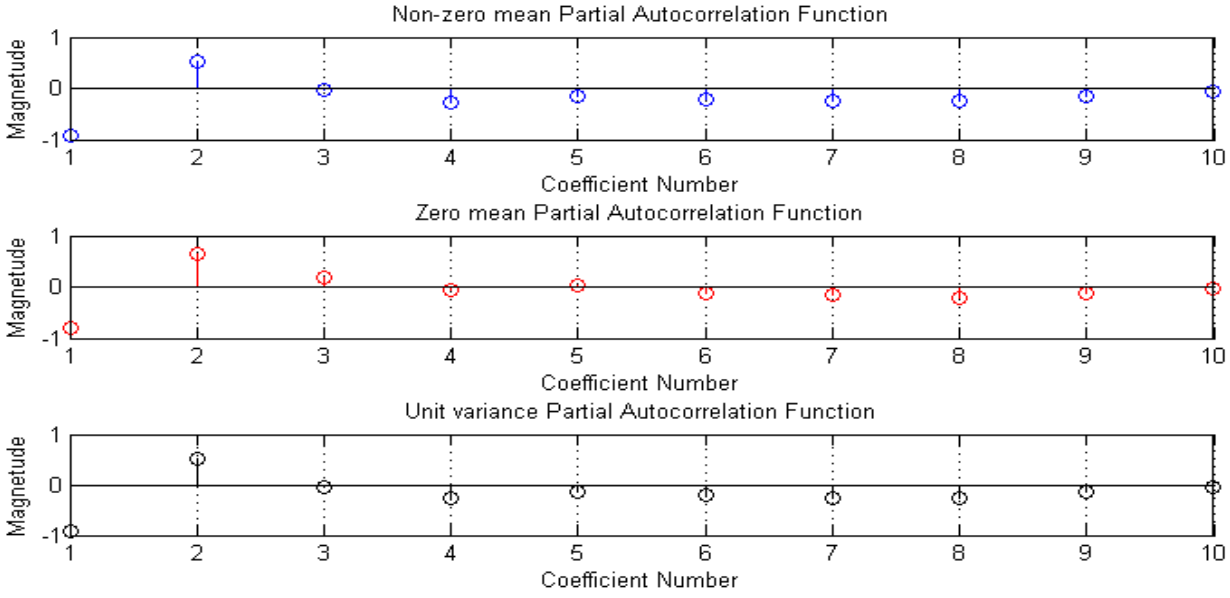

Figure 10. (Top) Non-zero mean PAC. (Middle) Zero mean PAC. (Bottom) Unit variance PAC.

Partial autocorrelation function using Yule-Walker equation is plotted in figure 10. PACs of zero-mean and unit variance input are same, while zero-mean partial autocorrelation is different. This is caused by the same reason that subtracting mean from data changes the results of Yule-Walker equations. It is observed that partial autocorrelation of first three lags have large magnitudes, while other peaks converges. As a result, AR (2) or AR (3) is a good model for sunspots.

2.4.4

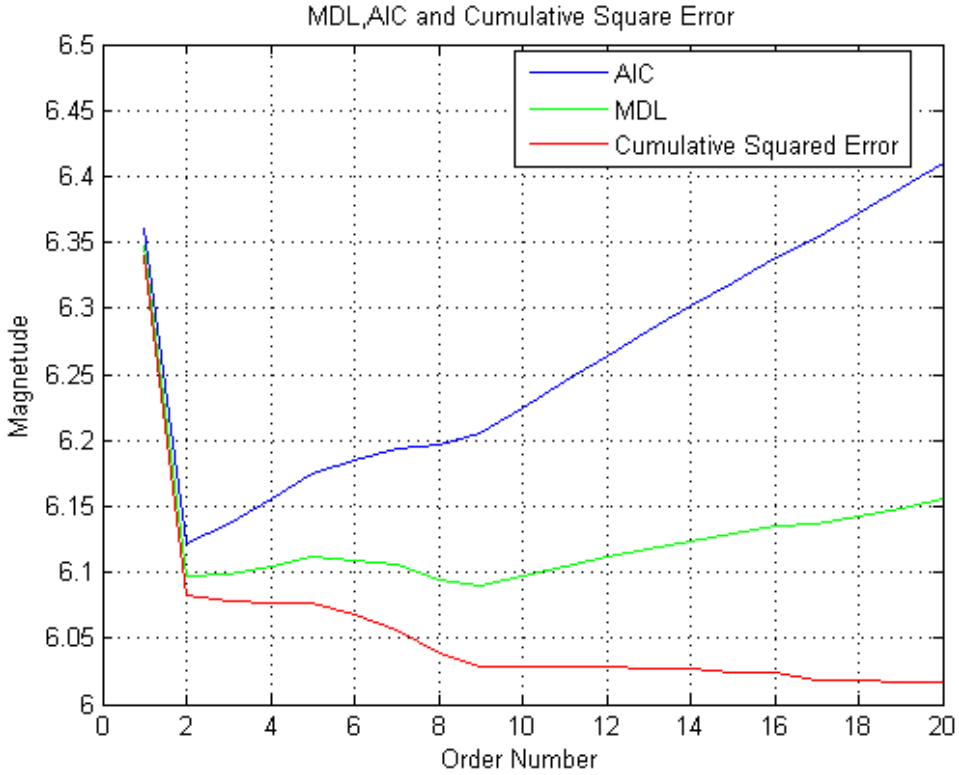

Figure 10. MDL, AIC and Cumulative Square Error

MDL and AIC are plotted using equations:

$$MDL(p) = \log(E_p) + \frac{p \cdot log(N)}{N}, AIC(p) = \log(E_p) + \frac{2 \cdot p}{N}$$

According to Figure 10, the best model order is the minimum value of MDL and AIC, which is three in this case. So considering both accuracy and complexity, AR (9) is the best model. AR(2) is also a reasonably good option.

2.4.5

Figure 11 shows the comparison between predicted values and real values using the past 250 data for various model order. 10 future values are predicted. In general, estimating further future time leads to larger inaccuracy. Regarding the relation between model order number and estimation accuracy, larger order number usually means better estimation of future data, at the cost of complexity. However, it is observed that AR2 model is sufficient to show the general shape of the curve. If we compare the coefficients generated using AR2 and AR10, they are pretty much similar. As the order number becomes higher, the difference between coefficients are smaller, so the improvement of result is tiny compared to cost. As a result, AR2 is a suitable model here.

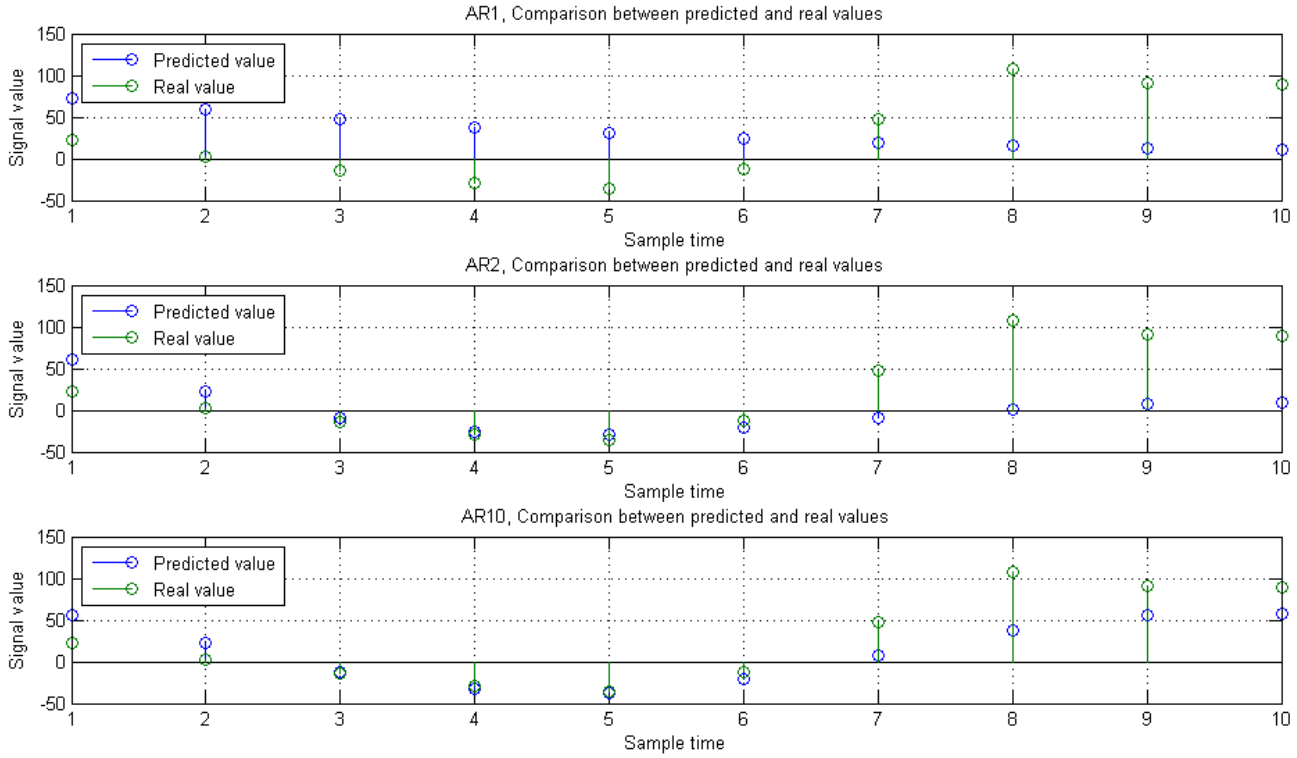

Figure 11. Prediction comparison of AR 1, 2 and 10

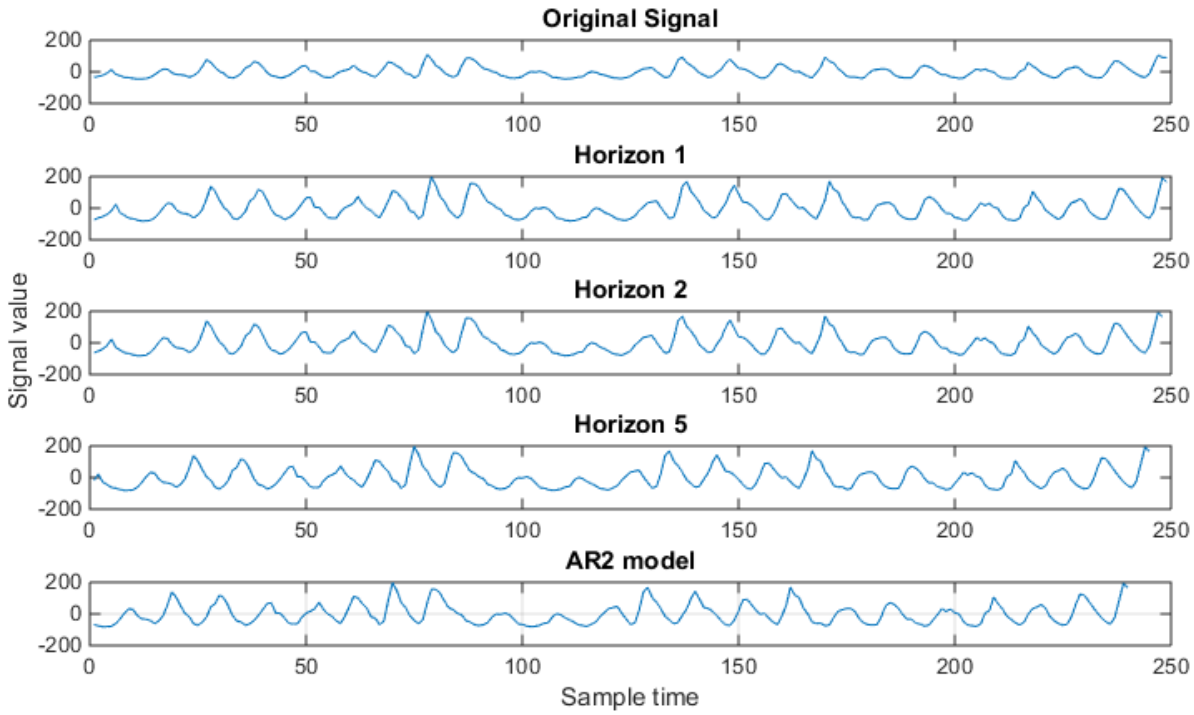

Figure 12. AR2 comparison with prediction horizon 1, 2,5,10

From the formula, $x[n+m] = a1 * x[n-1] + \cdots + ap * x[n-p]$,that effect of increasing prediction horizon is same as shifting the prediction with horizon 1 to left. This change only makes the prediction worse. Figure 12 shows the prediction of AR2 with different horizons.

2.5.1

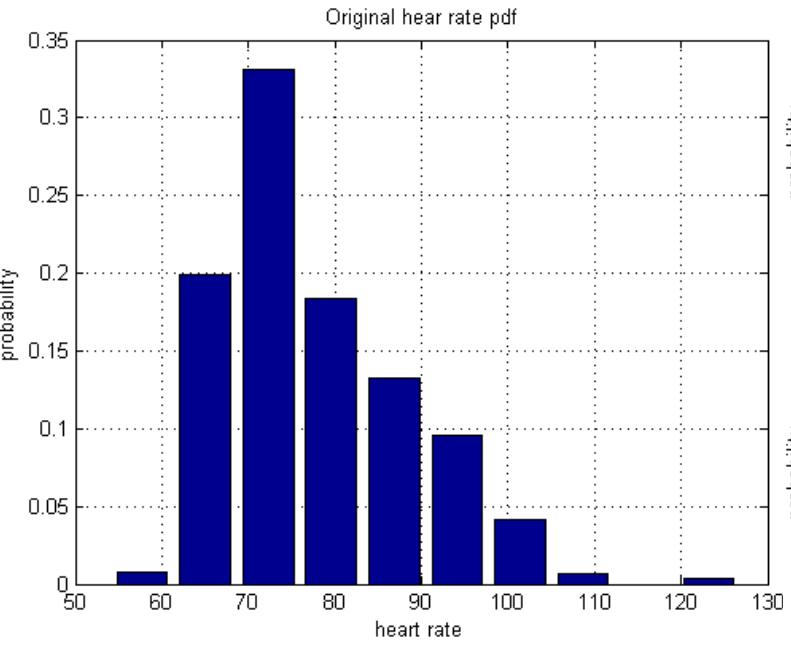

Figure 12. Original heart rate pdf plot

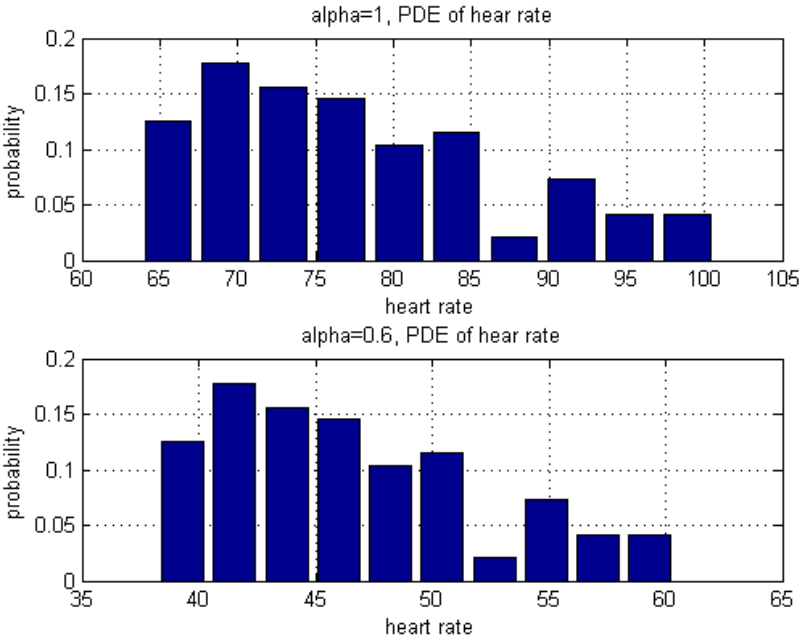

Figure 13. Averaged heart rate pdf plot

Figure 12 and Figure 13 show the PDE of the original hear rate and also the averaged heart rate.

2.5.2

PDE of the original heart rate is centralised around 70 but have some outliers. However, after averaging, most data are distributed in a small range between 65 and 100. Therefore, centralised data has smaller average and it is a better performance indicator over time.

The constant alpha behaves as the offset of heart rate. It does not affect the shape of heart rate distribution but it shifts the whole graph horizontally. Also, it affect the range of the averaged distribution since the difference between heart rate samples will be either reduced or increased depending on alpha value.

2.5.3

Figure 14 next page shows the autocorrelation function of three trials. It is observed that autocorrelation function tails off gradually. As a result, an autoregressive model is suggested. Since if data satisfy MA model, autocorrelation function should suddenly cut off after a certain lag.

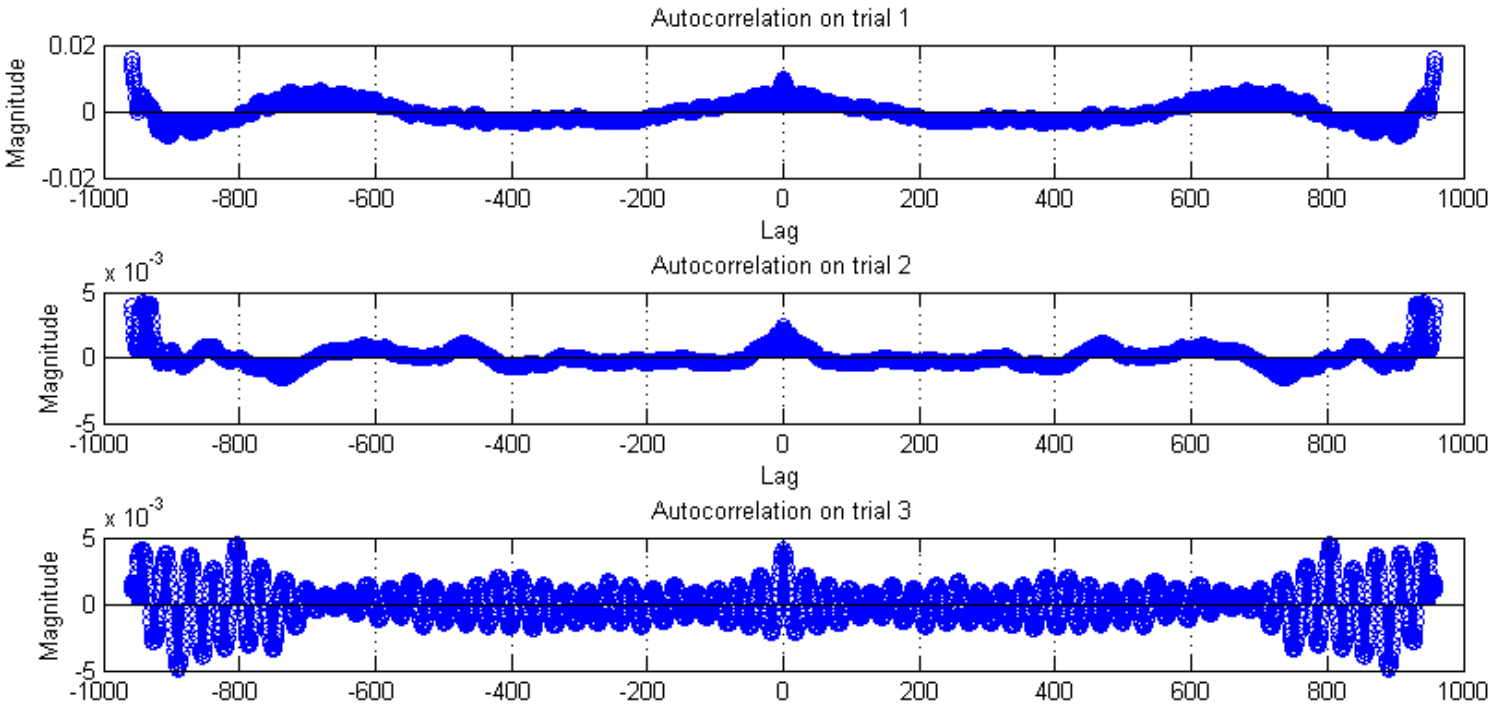

Figure 14. Averaged heart rate pdf plot

2.5.4

Figure 15 shows the trade-off between model accuracy and complexity. It indicates that autoregressive model can be used to describe heart rate. AR2 model is the most efficient model for trial two. Similarly, graph generated by trial three also shows the optimal order is 2.

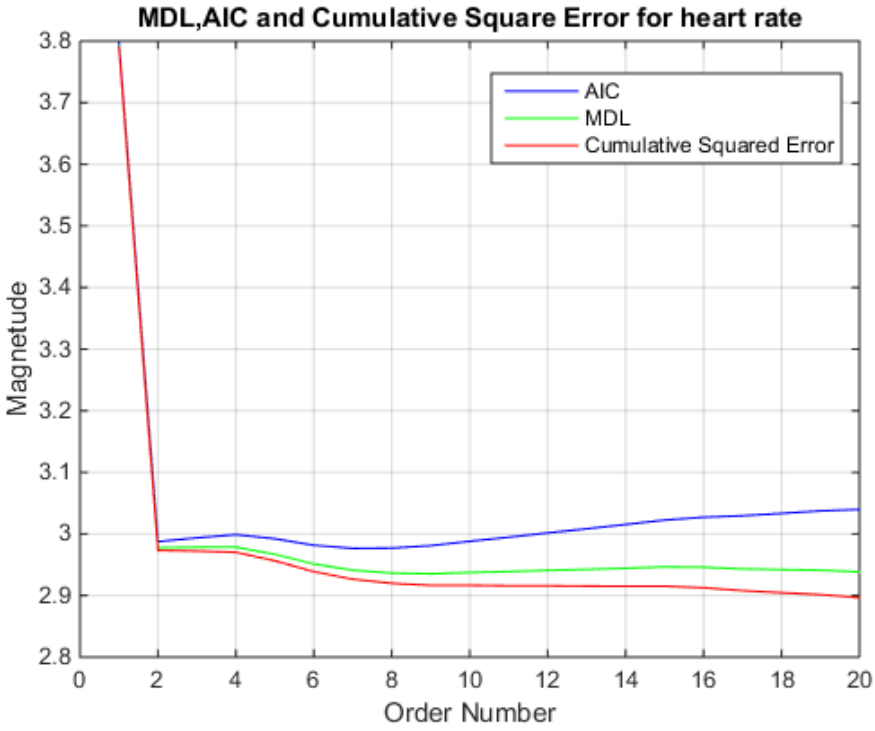

Figure 15. MDL, AIC and Cumulative Squared Error of heart rate trial two

3. Spectral estimation and modelling

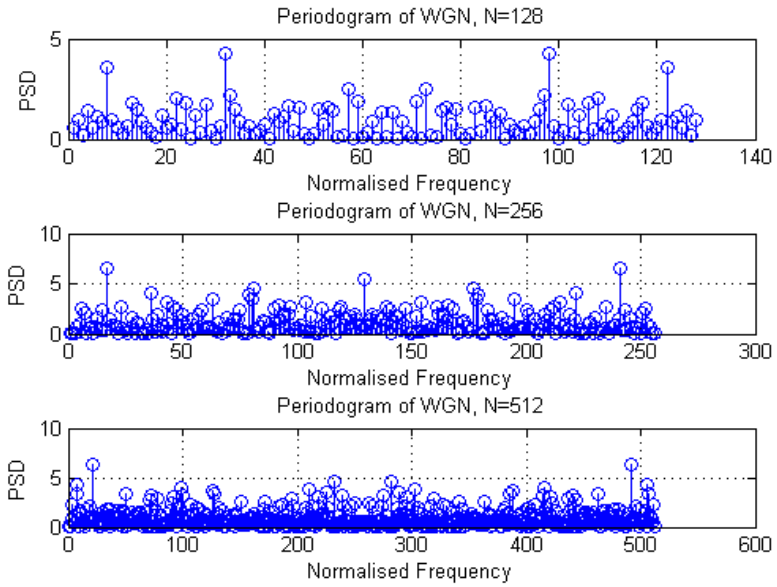

Figure 1. PSD estimation of WGN for N=128, 256,512

Figure 1 shows the PSD of WGN of various lengths. The theoretical PSD of WGN is a horizontal since all frequency components have same power magnitude. The above figure generally agree with theory. As sample number of WGN increases, a more accurate estimation of PSD is made. Power band is, hence, more flat over frequency axis.

3.1 Averaged periodogram estimates

3.1.1

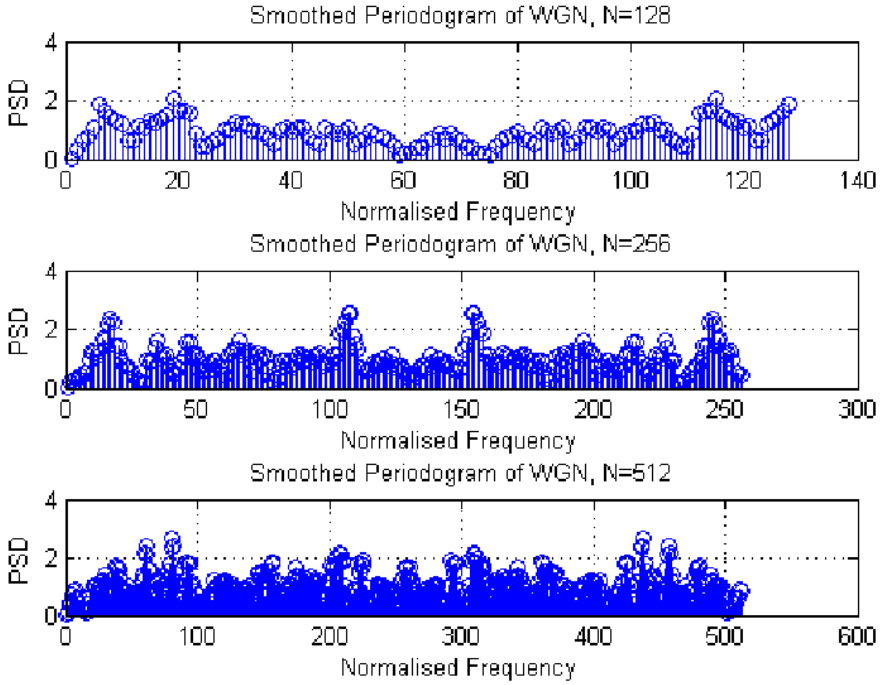

Figure 2. Smoothed PSD estimation of WGN for N=128, 256,512

Figure 2 shows the smoothed PSD of WGN of various lengths. Compared to figure 1, figure 2 is more close to the theoretical graph of PSD, which is a flat band across all frequencies.

3.1.2

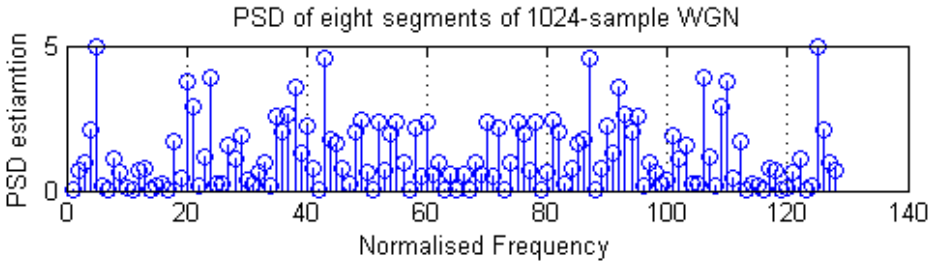

Figure 3. Sample PSD of one segment of 1024-sample WGN

One of the eight segments of 1024-sample WGN. It is observed that this shape is similar to the PSD of 128 WGN samples shown in figure 1. As a result, the inaccuracy of this PSD estimation is caused by its limited sample length.

3.1.3

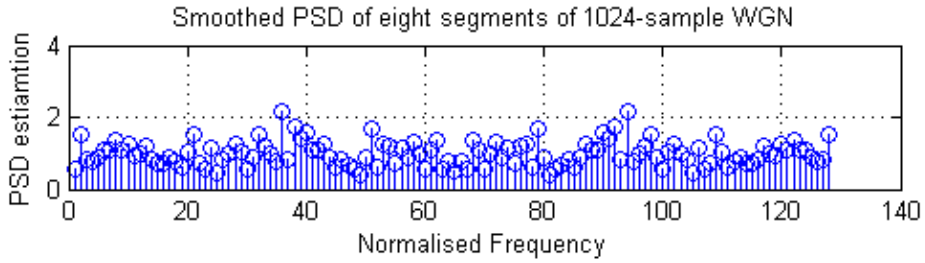

Figure 4. Averaged periodogram of one segment of 1024-sample WGN

Figure 4 shows the smoothed PSD of the segment in Figure 3. Averaged periodogram is more close to the theoretical PSD compared to individual PSD. Each frequency bin of averaged periodogram is the mean of the PSD estimation of eight segments. As a result, their differences to theoretical value cancel out each other and a better prediction is, hence, obtained.

3.2 Spectrum of autoregressive processes

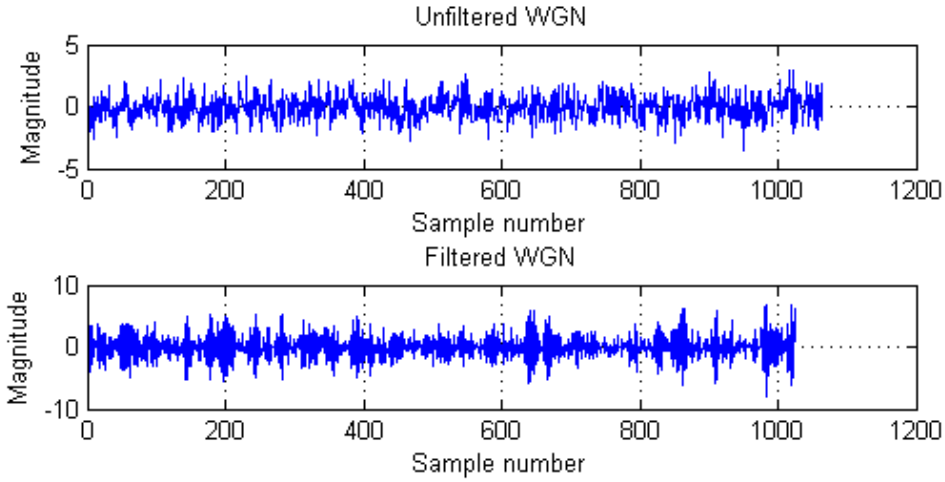

Figure 5. Comparison between filtered and unfiltered WGN

It is observed from Figure 5 that filtered WGN varies faster than the unfiltered because the AR1 filter with a=[1 0.9] is equivalent to a high-pass filter with a pole at (-0.9,0). As a result, the low frequency components of WGN is filtered and noise looks varying faster.

3.2.1

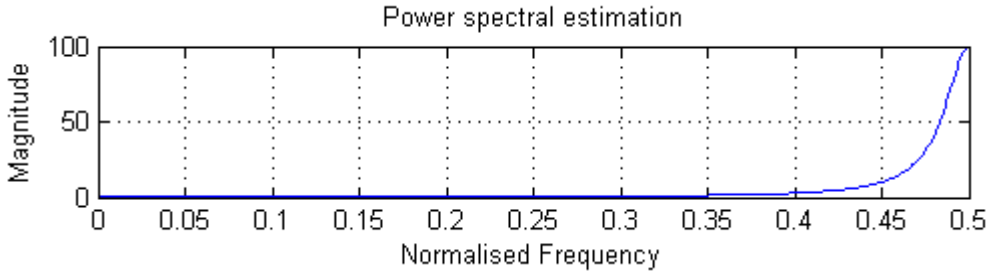

Figure 6. Power spectrum estimation of the filter with pole at (-0.9, 0)

PSD of the filter is shown in figure 6. $H(z) = \frac{1}{1+0.9z^{-1}}$ is the filter transfer function. By substitute $z = e^{jw}$. We obtain $\frac{e^{jw}}{e^{jw}+0.9}$. Since$e^{jw} = \cos(w) + jsin(w)$, w is calculated to be 2.69 rad/s, corresponding to 0.43 in this graph. Since figure 6 is a power graph, our calculated result approximately matches theory.

3.2.2

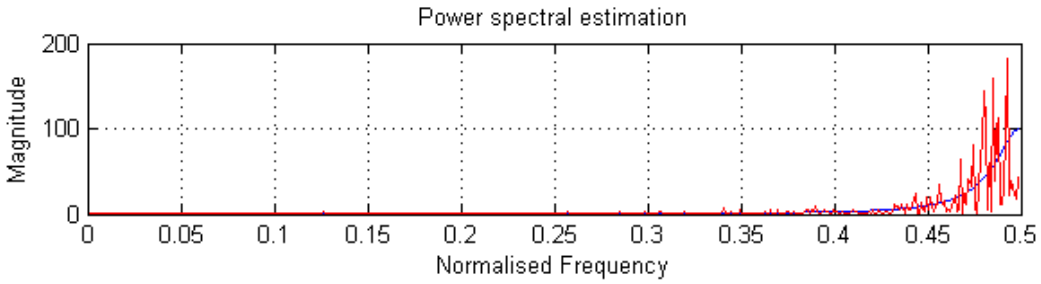

Figure 7. PSD comparison between filter and filtered WGN signal

Figure 7 compares the theoretical filter power spectrum and actual power spectrum of output after passing WGN through the filter. Since the input sample is limited, input spectrum is not a flat band over all frequency, spectrum always fluctuated around the average. As a result, In the transition and pass-band of the filter, fluctuated spectrum is shown rather than a smooth high pass curve.

3.2.3

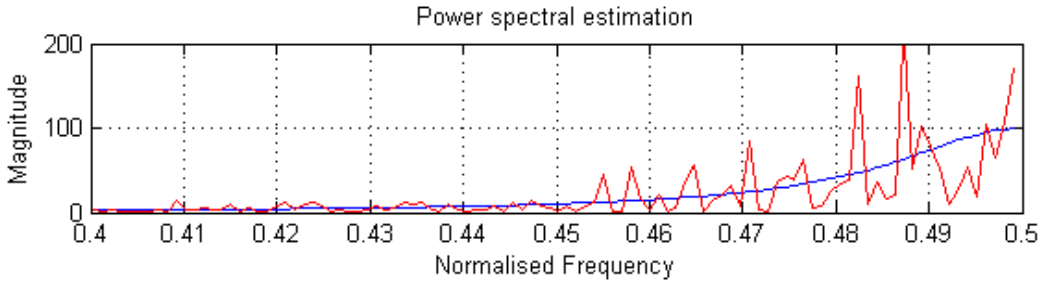

Figure 8. PSD comparison between filter and filtered WGN signal

The pgm function written in before takes a finite length of input data, which can be considered as multiplying a complete white noise and a rectangular window. The effect is equivalent to convolution of the theoretical filter spectrum with a sinc function, which is centred in the middle of frequency spectrum, i.e. 0.5. As a result, spectrum fluctuation increases when normalised frequency is getting close to 0.5. The fluctuation due to windowing adds to the fluctuation caused by incomplete input data.

3.2.4

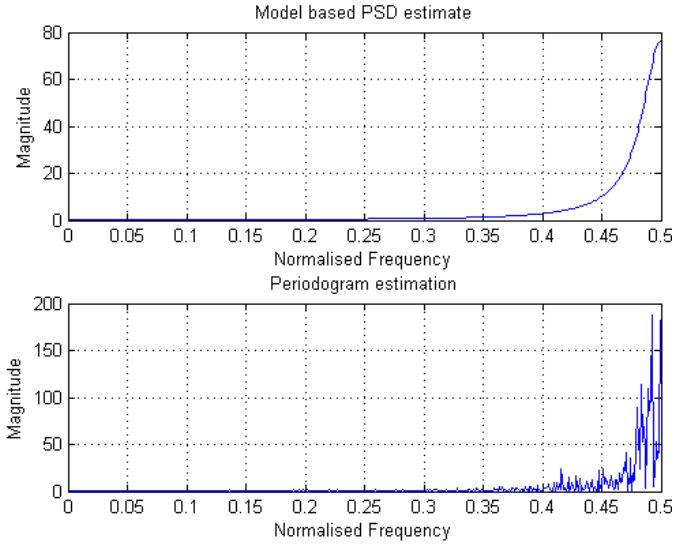

Figure 9. PSD comparison between model based and periodogram estimate

Figure 9 shows the comparison between model based and periodogram estimate. It can be seen that two curves are similar but not exactly same. Periodogram estimate is less close to the theoretical value than model based estimation since it directly transfer signal from time domain to frequency domain using FFT. Also, note that the magnitude of periodogram is larger than that of model based estimation, since the magnitude of periodogram mostly depends on signal variance. Therefore, AR model based estimate provides good result in estimating peaks, but not exact amplitude.

3.2.5

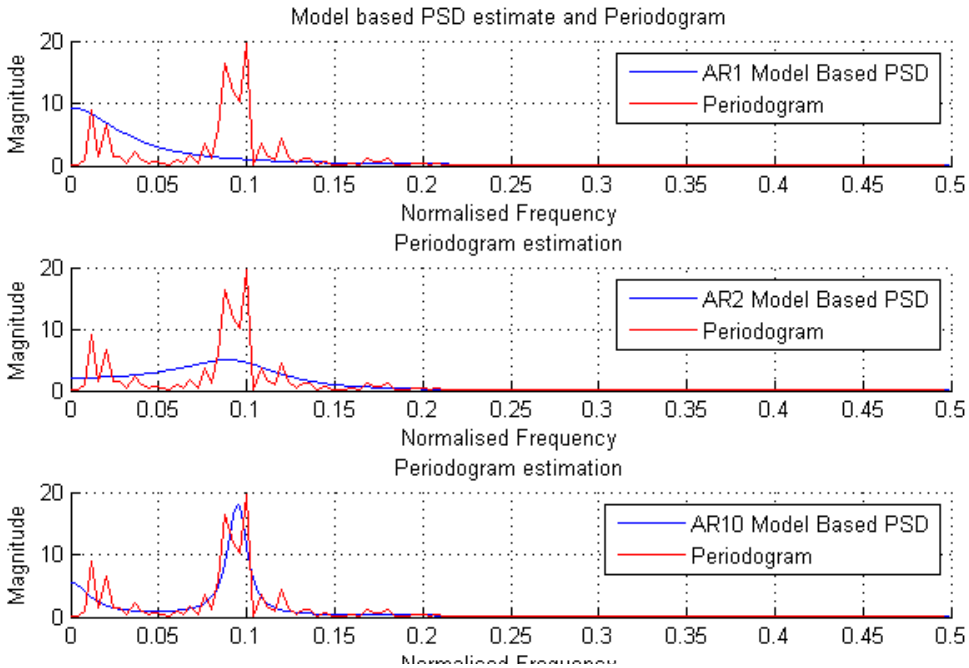

Figure 10. PSD comparison between model based and periodogram estimate for order 1,2 and 10

Figure 10 compares the model based PSD and periodogram of sunspot for AR1, AR2 and AR10. As we increase AR model number, simulated result approaching to ideal PSD. Compared to zero mean data, PSD of non-zero mean data consists of mostly are low-frequency components, which makes the PSD difficult to observe.

3.3 Spectrogram for time-frequency analysis: dial tone pad

3.3.1

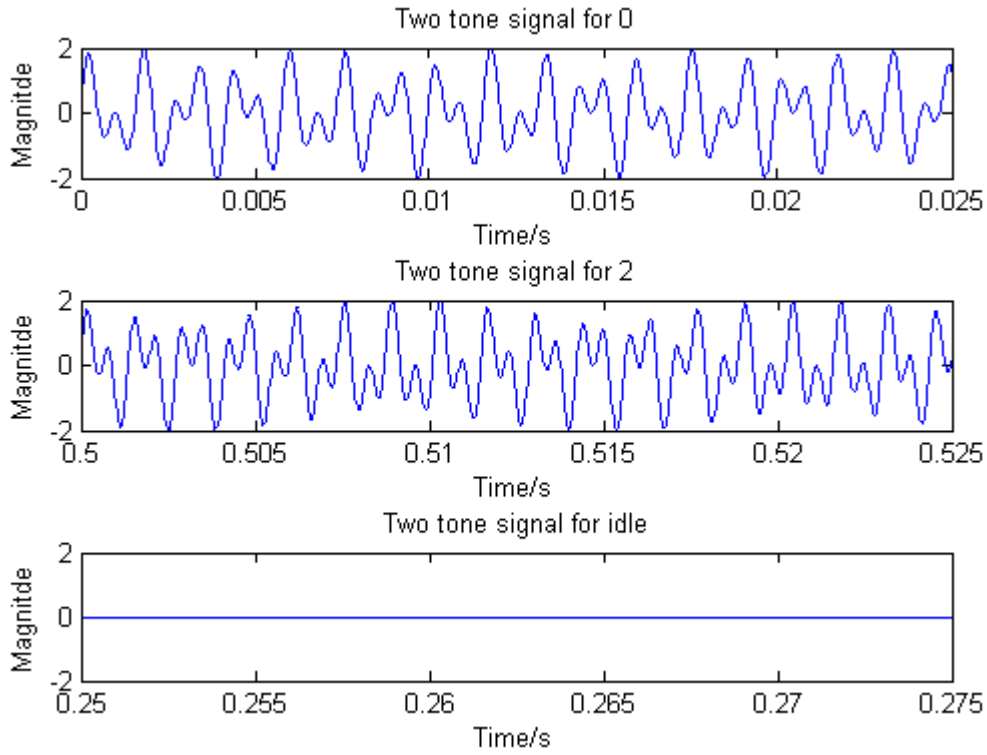

Figure 11. Two-tone signals for DTMF system

Figure 11 shows the two-tone signals for digit 0, 2 and idle. From the formula of sine wave addition: $\sin(2\pi * a * n) + \sin(2\pi * b * n) = 2\sin(2\pi * \frac{a+b}{2} * n)\cos(2\pi * \frac{a-b}{2} * n)$, $\frac{a+b}{2}$ is the frequency perceived by listener, while the other part $\frac{a-b}{2}$ controls the envelop of signal, and causes perception of 'beats'. In this case maximum possible frequency required to represent is Sampling $\frac{1477+941}{2} = 1209\ Hz$. Frequency 32768 is an appropriate number. It is around 10 times of the minimum frequency to represent 1209 Hz and good enough to represent an accurate time domain wave. Higher sampling frequency provides more accurate results at the cost of extra cost in time and space complexity.

3.3.2

Figure 12 shows the spectrogram of randomly generated London landline number. In the graph, the generated number is: 020 1832 7006. It can be seen that frequency of each dual tone signal has two main peaks at expected normalised frequencies.

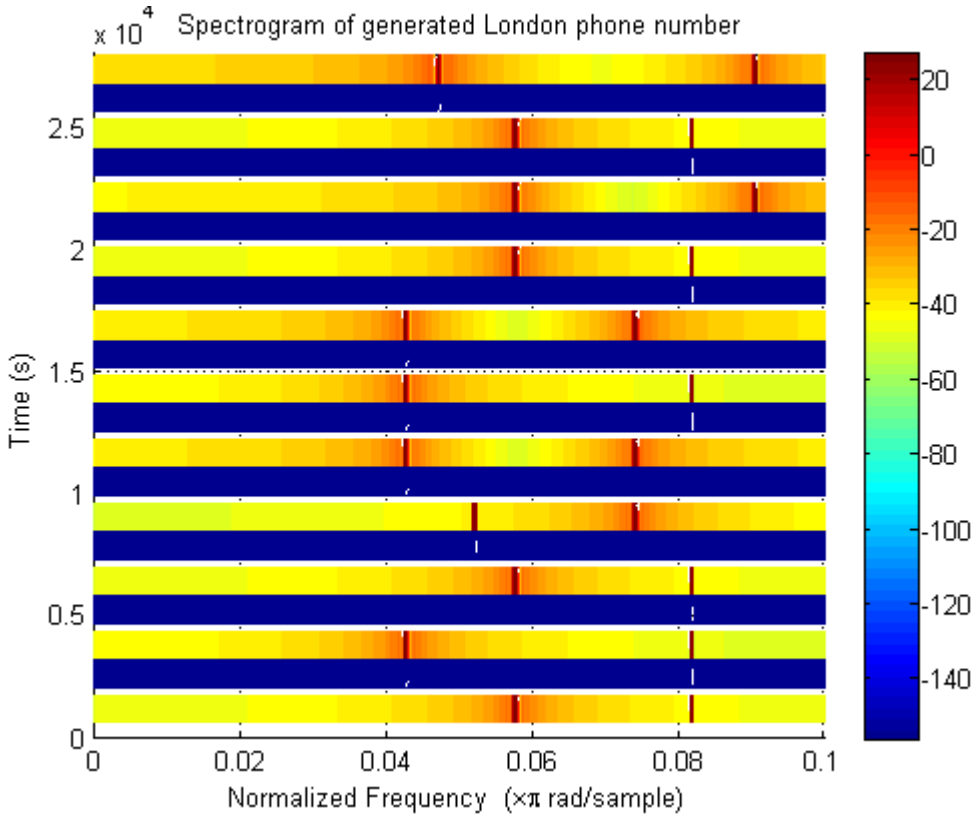

Figure 12. Spectrogram of landline 020 7121 0903

3.3.3

	1209 Hz	1336 Hz	1477 Hz
697 Hz	**1**	**2**	**3**
770 Hz	**4**	**5**	**6**
852 Hz	**7**	**8**	**9**
941 Hz	*	**0**	#

Table 1: Dial pad frequencies

Table 1 shows the actual frequencies representing each number. Converted to normalised frequency with sampling frequency 32768 Hz. The graph becomes:

	0.073792	0.081543	0.090149
0.042542	1	2	3
0.046997	4	5	6
0.052002	7	8	9
0.057434	*	0	#

Table 2. Dial pad normalised frequency with Fs=32768 Hz

For example, the first node has frequency components at around 0.057 and 0.081, which corresponds to the number 0. Using the same rule we can derive that the number series is 020 4204 8592, which agrees with the input number sequence.

3.2.4

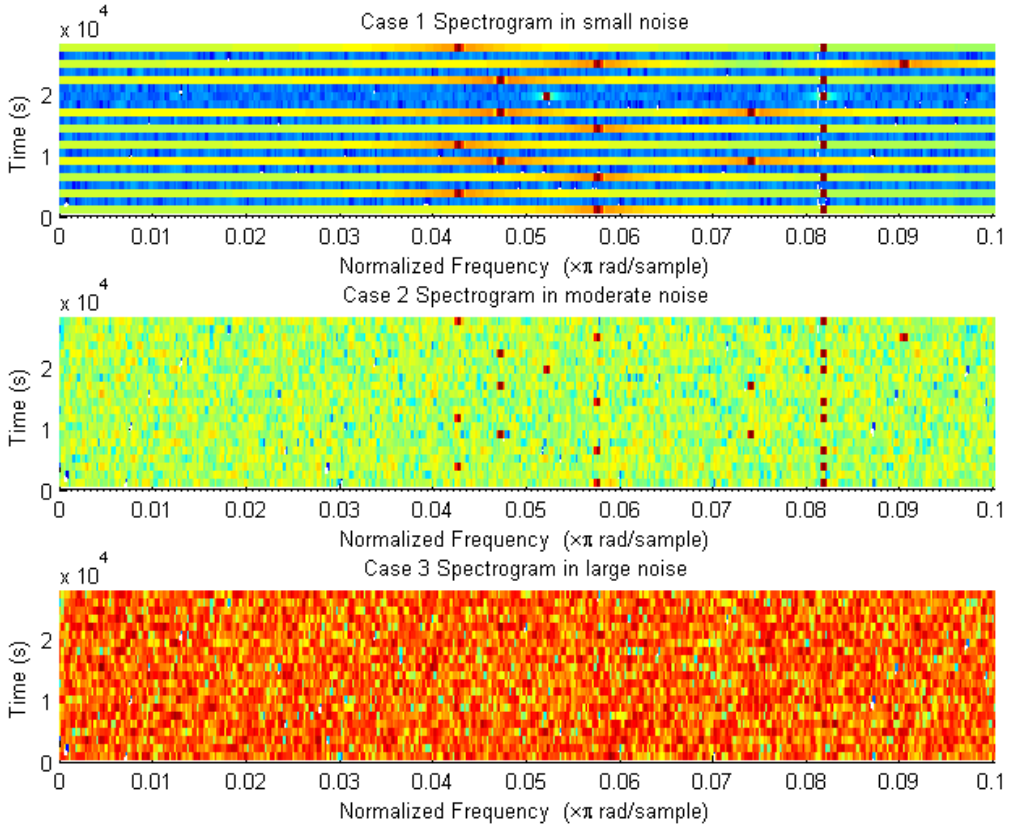

Figure 13. Spectrogram in various WGN

Figure 13 shows the spectrogram for various WGN amplitude. As we can see that, WGN introduces bins in all frequency regions. When the amplitude of WGN is small compared to signal, tone is still identifiable. When WGN has frequency amplitude larger than signal, spectrogram is not recognisable any more. The idle interval which is used to separate numbers are becoming less obvious to see.

3.4 Real world signals: Respiratory sinus arrhythmia from RR-Intervals

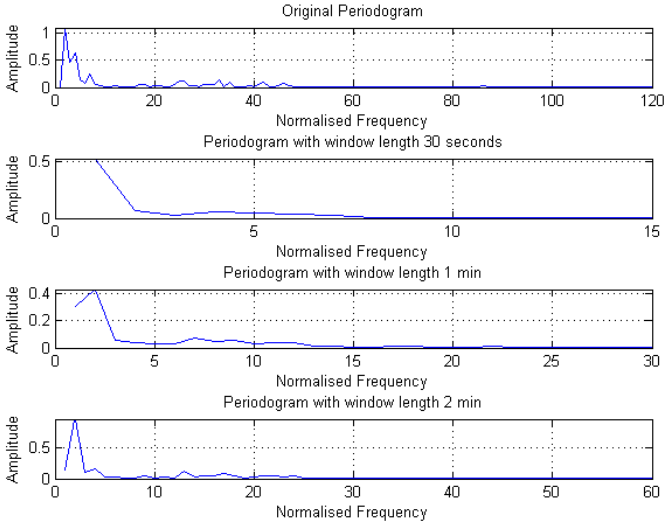

Figure 14. Periodograms of RRI, trial 1

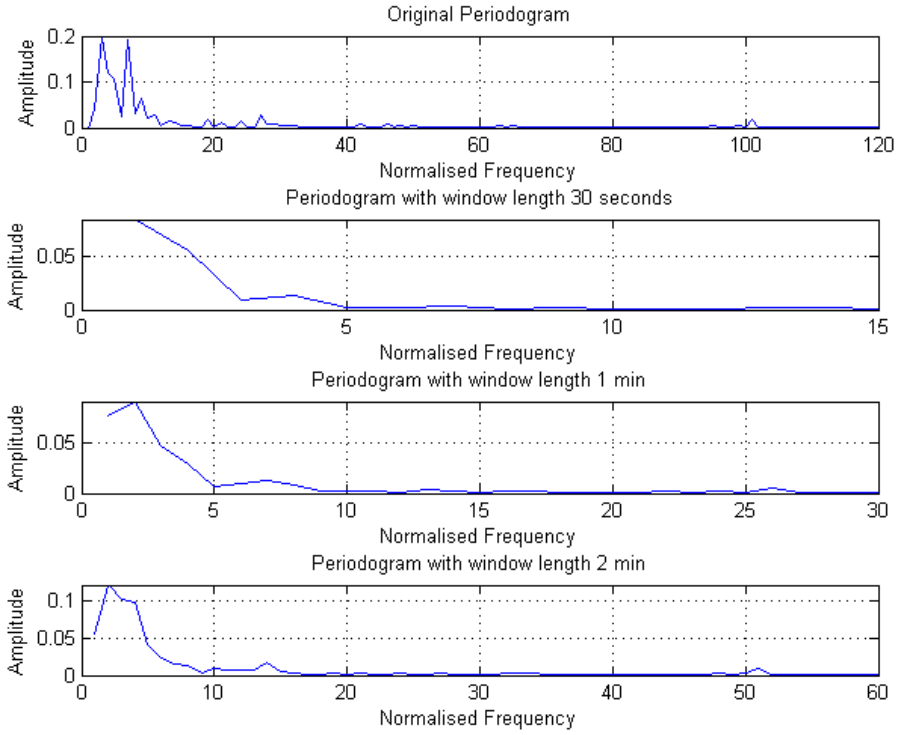

Figure 15. Periodograms of RRI, trial 2

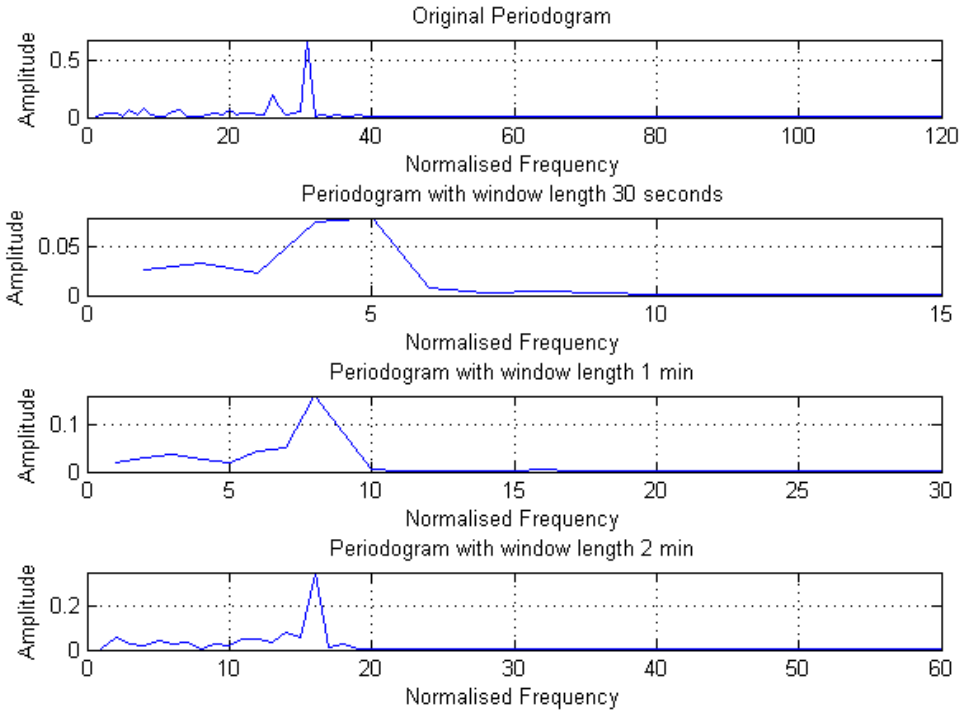

Figure 16. Periodograms of RRI, trial 3

Figure 14, 15 and 16 show the periodogram of RRI for different trials using different estimation parameters. It is observed that estimates of RRI from three trials have different peak frequency, regardless of estimation window length. Their valid frequencies are at 40 Hz, 100 Hz and 30 Hz correspondingly. The averaged data gives us more information about which peak is valid. Trial 2 has the highest peak frequency meaning that RRI changes more

rapidly. Trial 3 has the lowest peak frequency, meaning that RRI changes very slowly. The peak in trial 1 at 40 Hz means there are 40 cycles of respiration, i.e. 20 beats per minute, which is around the expected frequency of natural breathe. So corresponding RSA is $20 \times 2 \div 60 = 0.667$. Similarly, RSA of trial 2 and 3 are approximately 1.667 and 0.125.

If we zoom in the graph, we observe harmonics at higher frequencies. For example, in trial, there is a peak at 200 Hz, which is essentially a harmonic of the major peak at 100 Hz.

4. Optimal filtering – fixed and adaptive

By using MATLAB snr function, it is calculated that the signal-to-noise ratio is around 20 dB. It agrees with our theory since variance of system output is 1 while that of WGN is $\sigma^2 = 0.1^2 = 0.01$. So the SNR is $10\log(\frac{1}{0.01}) = 20$ dB.

4.1 Wiener filter

4.1.1

$$\mathbf{R}_{xx} = E\{\mathbf{x}[n]\mathbf{x}^T[n]\} = \begin{bmatrix} r_{xx}(0) & r_{xx}(-1) & \cdots & r_{xx}(-N_w) \\ r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(-N_w+1) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(N_w) & r_{xx}(N_w-1) & \cdots & r_{xx}(0) \end{bmatrix}$$

$$\mathbf{p}_{zx} = E\{z[n]\mathbf{x}[n]\} = \begin{bmatrix} E\{z[n]x[n]\} \\ E\{z[n]x[n-1]\} \\ \vdots \\ E\{z[n]x[n-N_w]\} \end{bmatrix} = \begin{bmatrix} r_{zx}(0) \\ r_{zx}(-1) \\ \vdots \\ r_{zx}(-N_w) \end{bmatrix}$$

Using the formulas above, we are able to calculate matrix R and P. After solving the Wiener-Hopf equation $\boldsymbol{w_{opt}} = \boldsymbol{R_{xx}^{-1}P_{zx}}$ **.** The optimal wiener filter coefficients are obtained. The first five coefficients of the optimal Wiener filters are: 1.00, 2.00, 3.00, 2.00 and 1.00 correspondingly, while all other coefficients are small enough to be ignored. It can be seen that the first five coefficients agree with our 'unknown' system. Note that when we calculate filter coefficients, filter output is not unit variance, since changing signal variance scales the wiener filter coefficients down.

4.1.2

Repeat the experiments with different noise power and investigate its effect on SNR. The relation between variance, SNR and filter coefficients are shown in the following table. Note that filter output variance is fixed at 1 to make comparison convenient. Wiener filter coefficients are scaled down but ratio is kept same.

Std_dev (σ)	SNR	First Five Optimal Wiener Coefficients				
		1	2	3	4	5
0.1	19.8	0.24	0.47	0.71	0.47	0.23
2	-6.06	0.27	0.43	0.56	0.29	0.26
4	-12	0.28	0.66	0.7	0.55	0.18
6	-15.1	0.36	0.47	0.67	0.3	0.07
8	-18	0.38	0.24	0.91	0.51	0.17
10	-20	0.18	0.38	0.78	0.47	0.55

Table 1. Relation between noise variance, SNR and Wiener filter coefficients

Since unknown system output is fixed at 1, relation between SNR and noise variance fits the equation: $SNR = 10\log(\frac{1}{\sigma^2})$. It is observed that, as noise power increases, there are more deviation between calculated wiener filter coefficients and theoretical values.

However, theoretically, noise have no effect on wiener filter coefficients if noise is zero-mean. $R_{xz}[k] = E\{x[n]z[n+k]\} = E\{x[n](y[n+k] + \mu[n+k])\} = E\{x[n]y[n+k]\} + E\{x[n]\mu[n+k]\}$

$E\{x[n]\mu[n+k]\} = E\{x[n]\}E\{\mu[n+k]\} = 0$, if given noise is zero-mean.

In reality, pure zero mean WGN does not exist, that is why in our test, noise power adds to the error of filter coefficients. In this case, noise window size is chosen to be the maximum, which is 999. Large window size makes mean of noise converge to zero. So more accurate results are obtained. Conversely, smaller window size adds to the influence on optimal Wiener filter coefficients brought by noise power. If N_w is smaller than 5, calculated coefficients are no longer complete and accurate, since 'unknown' system has five coefficients.

4.1.3

Complexity of Wiener filter solution comes from calculating cross- and auto-correlations as well as solving the Wiener-Hopf equations.

Finding each correlation matrix p requires O(Nw*N) multiplications and O(Nw*N) additions. While for matrix R, calculation process requires O(Nw*Nw*N) multiplications and O(Nw*Nw*N) additions. In total, computational complexity to calculate R and p is $O(N*N_w^2)$, this is also the number of multiplications and additions required.

Solving Wiener-Hopf equation, $\boldsymbol{w_{opt}} = \boldsymbol{R_{xx}^{-1}}\boldsymbol{P}_{zx}$ takes $O(N_w^3)$ to find its inverse. Multiplication between the two matrix has N_w*N_w multiplications and additions. In total, Solving Wiener-Hopf equation has complexity $O(N_w^3)$.

Combining the two processes, this algorithm has complexity of $O(N*N_w^2)$.

4.2 The least mean square (LMS) algorithm

4.2.1

Wiener filter solution cannot be used for time-varying waves. Hence, we use least mean square algorithm to approach the right coefficients iteratively. We start from one segment of input signal, keep tracking the error between filter output and theoretical output, and adjust the weights of past inputs based on error signal. Weights are updated iteratively using the formula: $\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu e[n]\boldsymbol{x}(n)$.

Result using least mean square algorithm is quite close to wiener filter solution for the case in 4.1 where signal is stationary. The first five coefficients are 1.0104, 1.9991, 2.9939, 2.0118 and 1.0183, while other coefficients are small enough to be neglected.

4.2.2

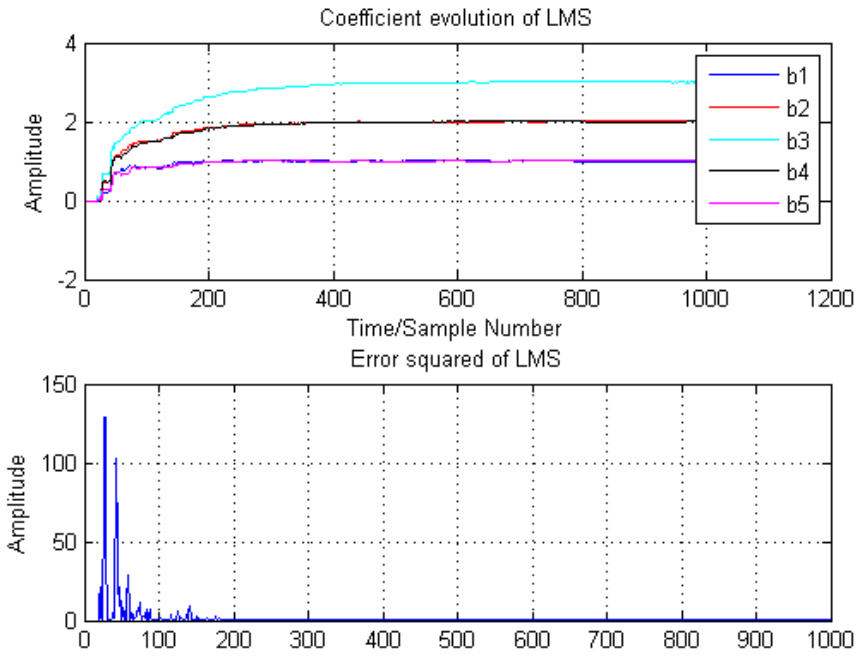

Figure 1. Coefficients evolution and error squared LMS algorithm, u=0.01

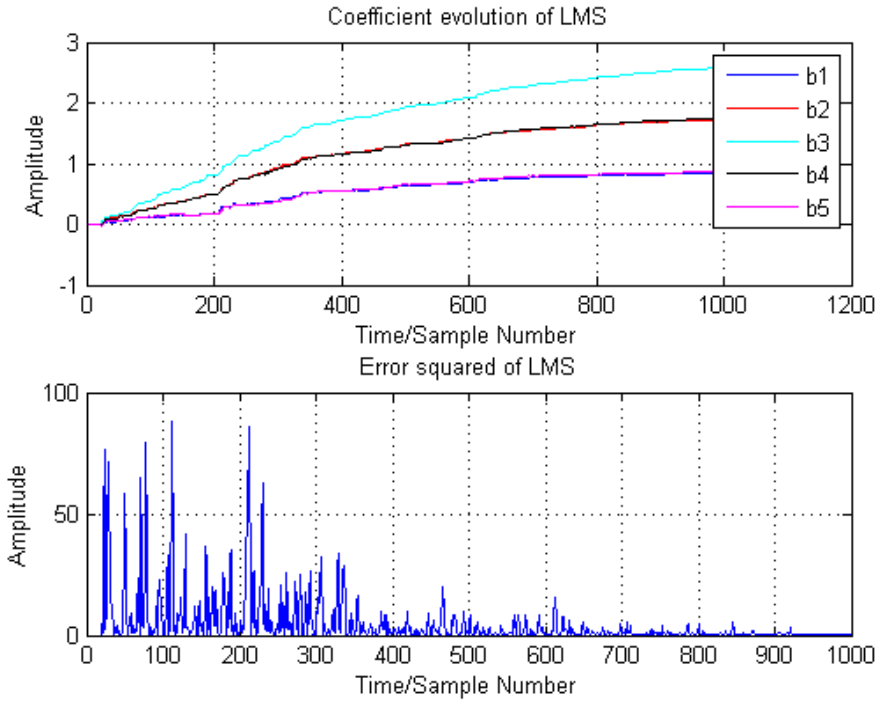

Figure 2. Coefficients evolution and error squared LMS algorithm, u=0.002

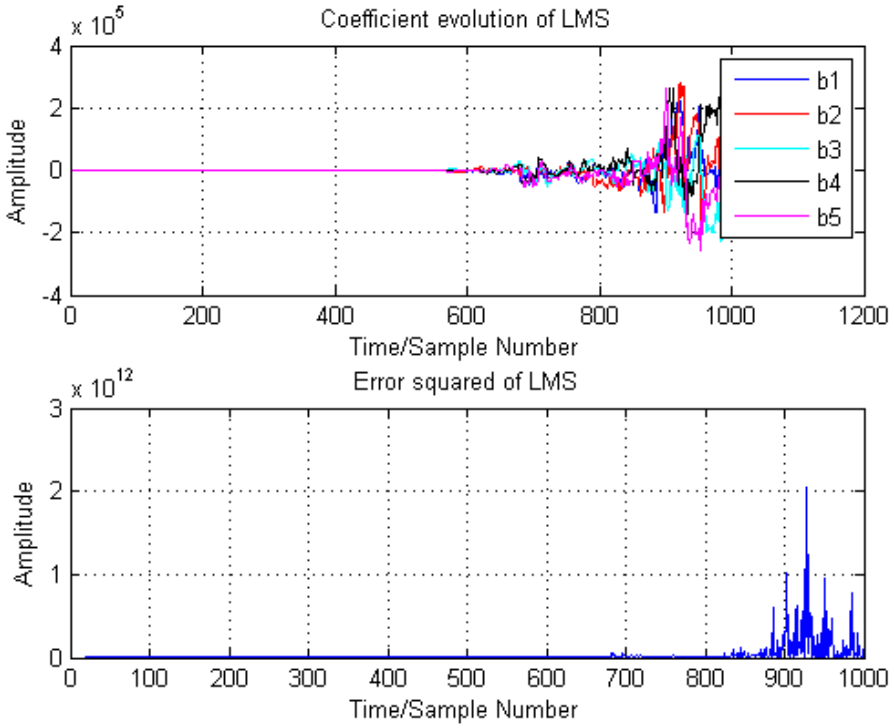

Figure 3. Coefficients evolution and error squared LMS algorithm, u=0.1

Figure 1, 2 and 3 show the evolution of LMS algorithm and error squared with different adaptation gain, which determines how fast estimated coefficients changes according to error signal. Larger adaptation gain means larger steps size, which makes coefficients quickly approach theoretical value but may results in inaccuracy. Figure 3 shows the example of large adaptation gain and error signal is becoming large at large sample number. Figure 2 shows the case when adaptation gain is small, coefficients change so slow that theoretical values may not be reached. If given a suitable adaptation gain and signal is stationary, adaptive filter converge to the Wiener solution within sample length.

4.2.3

The computational complexity of the LMS algorithm is composed of iterative computation of output y, error signal e and weights. Assuming N_w is window size, N is total sample length.

Calculating output y requires N_w*(N-N_w+1) multiplications and additions, which is O(N*N_w). Error signal requires N_w steps of addition. Calculating weights requires 2*N_w+(N-N_w+1) multiplications and additions, which is O(N_w). In total, its complexity is O(N*N_w).

4.3 Gear shifting

Instead of using a fixed adaptation gain, we use a time-varying gain instead. For sample with large error, we produce a large adaptation gain to adjust the weight more rapidly. For sample with small error, we apply small adaptation gain in order to obtain accurate final result, since small step size leads to higher accuracy.

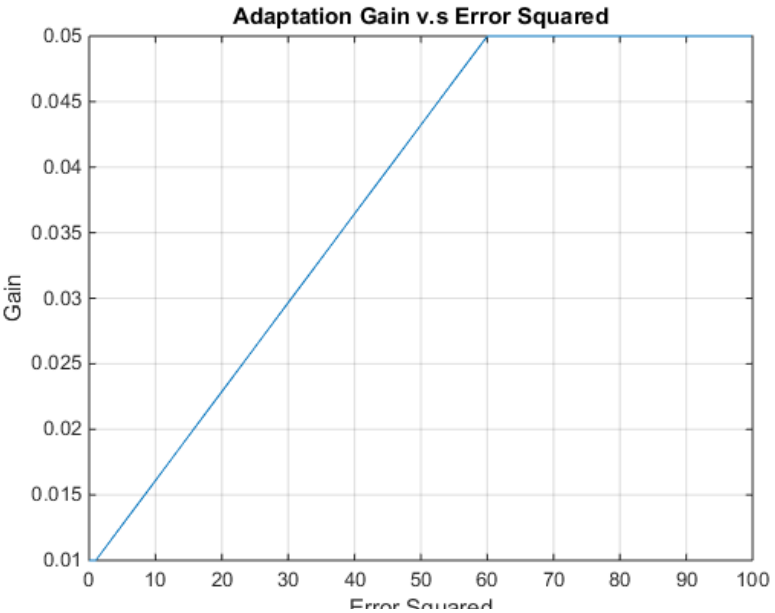

Figure 4. Relation between adaptation gain and error squared

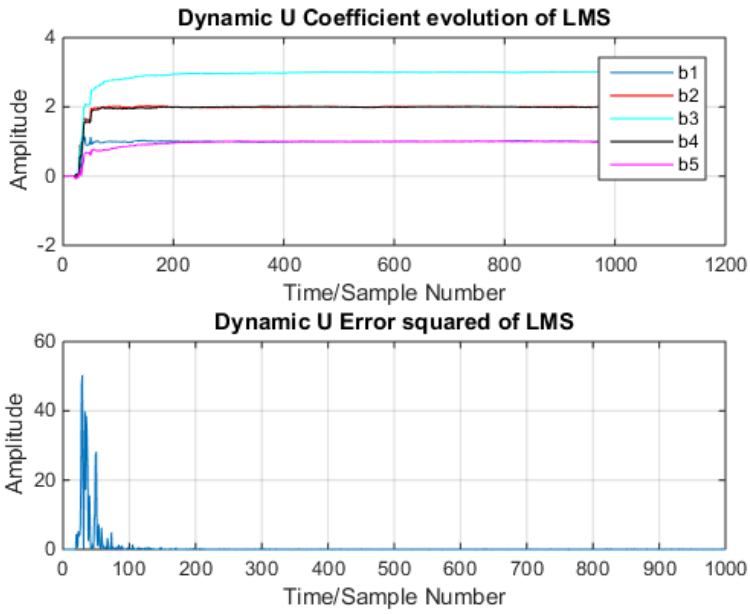

Figure 5. Evolution of coefficients and error square Dynamic u

Figure 4 shows the method I use to implement dynamic adaptation gain. When error square is between a certain ranges, adaptation gain is linearly proportional to error squared. When error square exceeds a maximum or minimum, a boundary adaptation gain is applied. Parameters are carefully selected to perform fast rise in the beginning and accurate prediction afterwards.

The result is shown in Figure 5. Compared to figure 1, where u is fixed at 0.01, coefficients prediction approaches to the theoretical value more rapidly. In the second case, coefficients reaches theoretical level in first 200 samples, while the first method requires around 400 samples. Predictions from both methods obtain reasonably good result.

4.4 Identification of AR processes

4.41

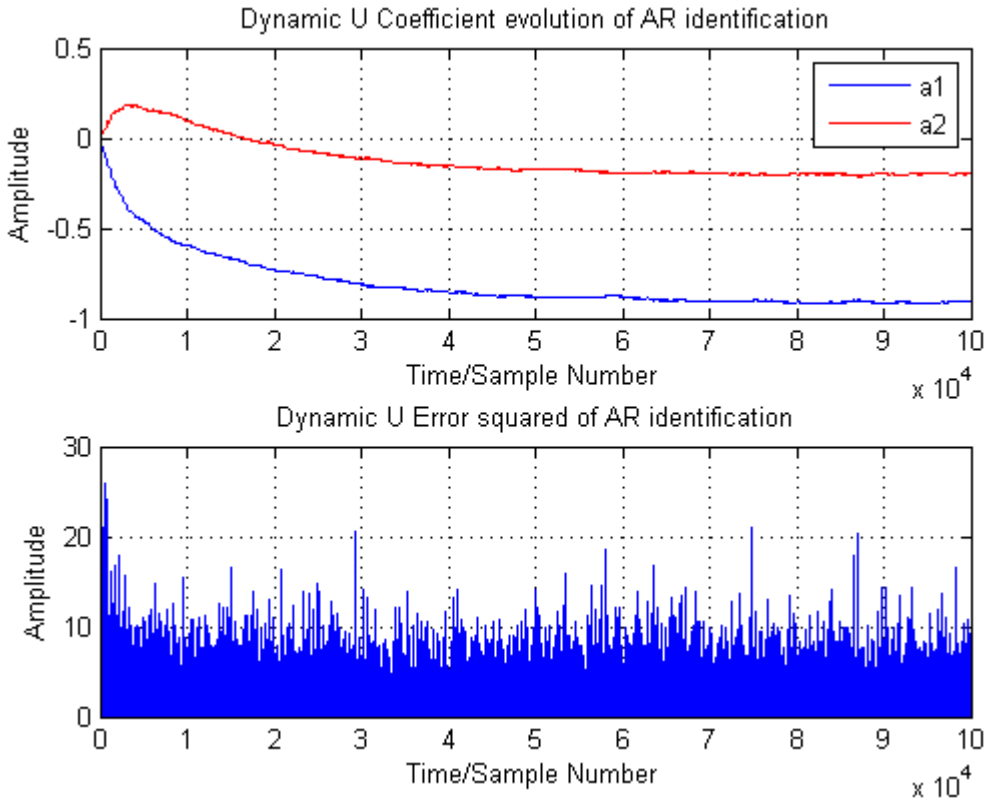

Figure 6. Coefficients and error evolution of AR process identification

In AR identification, we aim to predict AR coefficients from system output. Given an AR2 system with equation: $x[k] = a_1x[k-1] + a_2x[k-2] + n[k]$, we use adaptation algorithm and iteratively compare the output generated from the past two samples and the current sample. Predicted coefficients are then adjusted to approach theoretical coefficients.

After implementing the algorithm, we obtain Figure 6, which shows that estimated coefficients are around -0.2 and -0.9 correspondingly. We are given the AR filter has b=1 and a= [1 0.9 0.2]. Filter response can be represented as: $\mathrm{H(z)} = \frac{1}{1+0.9z^{-1}+0.2z^{-2}}$. It can be written in another form of $x[k] = -0.9x[k-1] - 0.2x[k-2] + n[k]$. As a result, the estimated coefficients are close to expected values.

4.4.2

Figure 7 shows the effect of different step size on predicted coefficients. It is observed that when adaptation gain is 0.01, coefficients approach theoretical values relatively fast and then oscillates around the correct value. For the case when adaptation gain is 0.1, step size is so large that the coefficients just oscillate around the actual values. When the adaptation gain is 0.001, coefficients approach the actual values slowly. In figure 8, sample number is not large enough for coefficients to converge to -0.9 and -0.2.

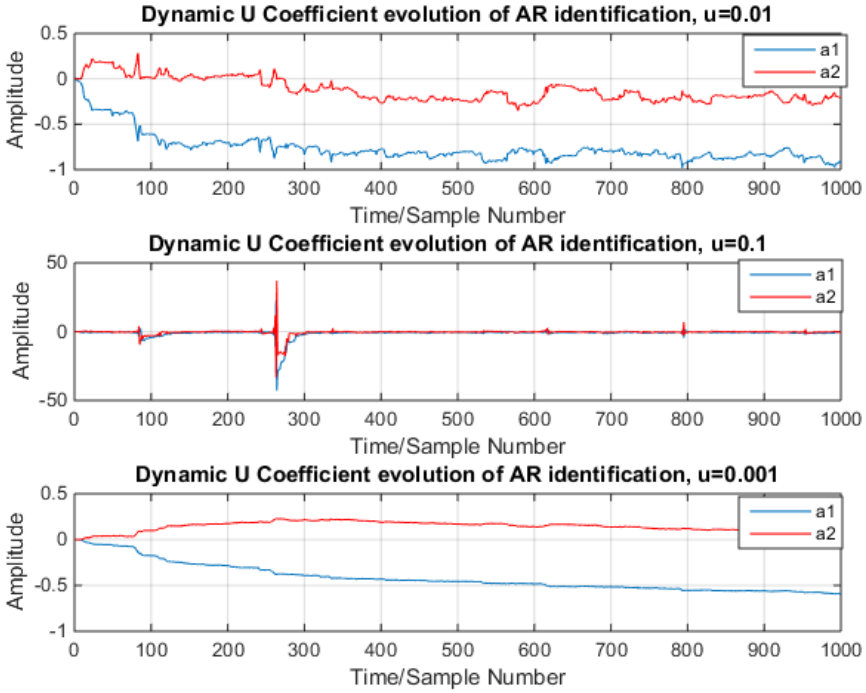

Figure 7. Coefficients and error evolution for various adaptation gain

4.5 Speech recognition

4.5.1

After recording the letters with sampling frequency 44100 Hz and taking 1000 samples in the valid region, various adaptation gains are used.

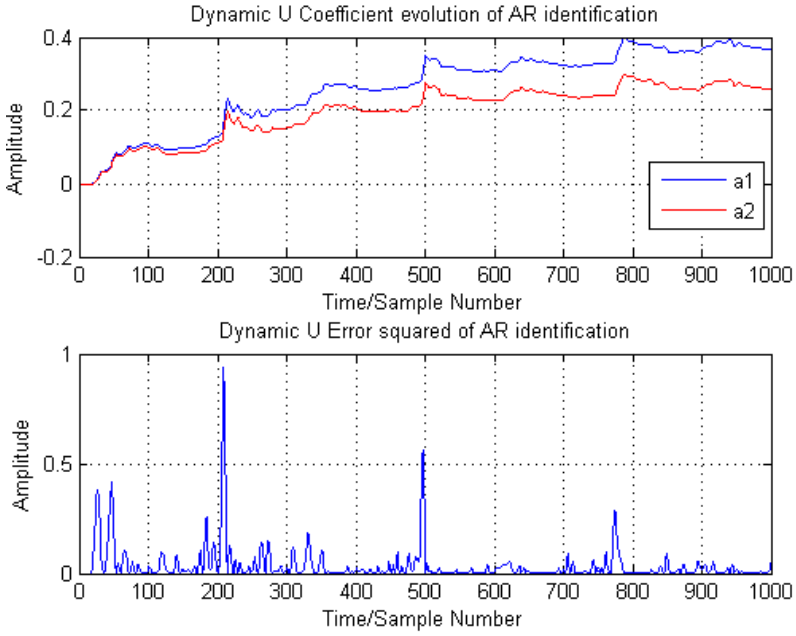

Figure 8. Coefficients and error evolution for various adaptation gain

Figure 8 shows the coefficients evolution when u is 0. 1. From the error evolution graph, 0.1 is a good option. Gear shifting requires different parameter setting from previous exercise and coefficients rise faster than fixed adaptation gain. If we observe the weights calculated using adaptive filter, coefficients are becoming negligible after order 3.

4.5.2

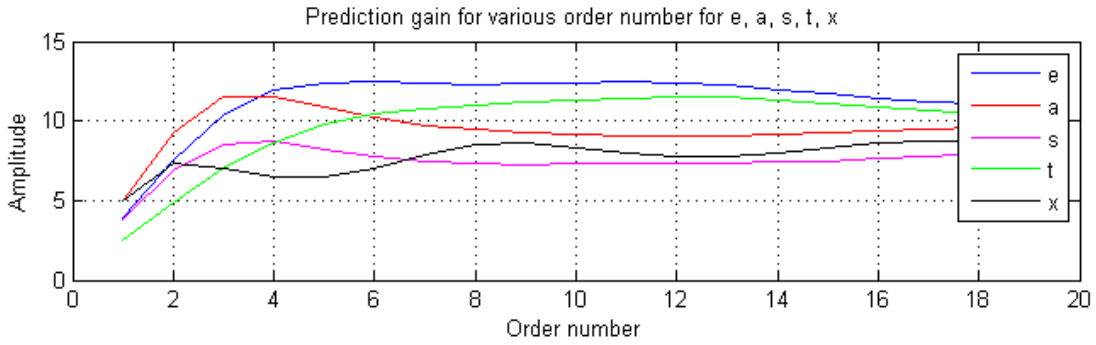

Figure 9. Prediction gain for e, a, s, t and x

Figure 9 is the example of using prediction gain to determine filter performance for different letters, using the formula: $R_p = 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_{e^2}}\right)$. In general, when order number rise above five, optimal performance is achieved. However, the best order is different for different letters. And prediction gain only measures performance and ignore the cost of computational complexity.

4.6 Dealing with computational complexity: sign algorithms

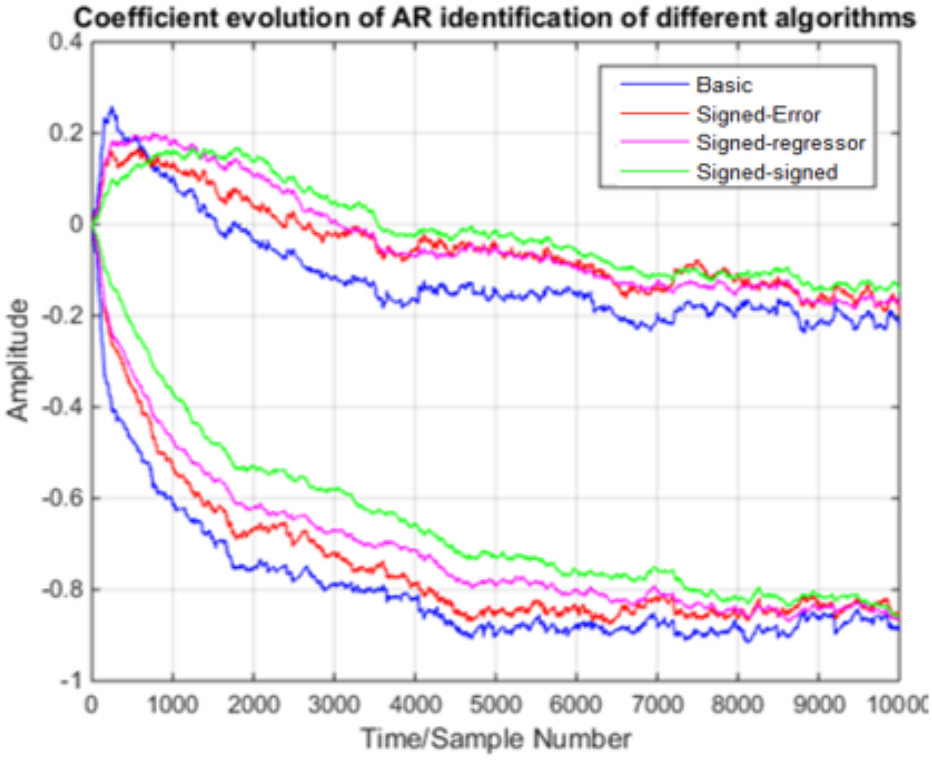

Figure 8. Coefficients and error evolution of AR identification using different algorithms

Figure 8 compares the performance of different algorithms. The basic algorithm rise fastest while the signed-error algorithm rise slowest among the four algorithms. It is observed that these algorithms obtain reasonably good result as the basic algorithm, while computational complexity is reduced.

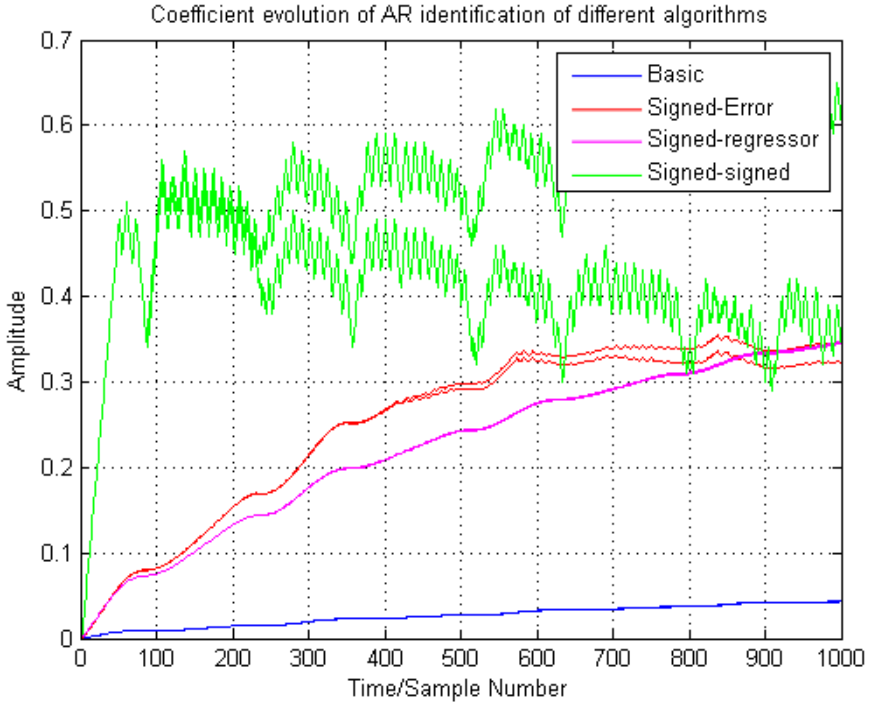

Figure 9. Coefficients and error evolution of AR identification using different algorithms

The above figure is the performance for different algorithms applied on speech signal when u=0.01, it can be observed that sign algorithm has the shortest reaction time among them and a reasonably good result can be obtained. While other algorithms either rise very slowly or produce result with large error for non-stationary signals.

5. A Real World Case Study: Vinyl Denoising

5.1

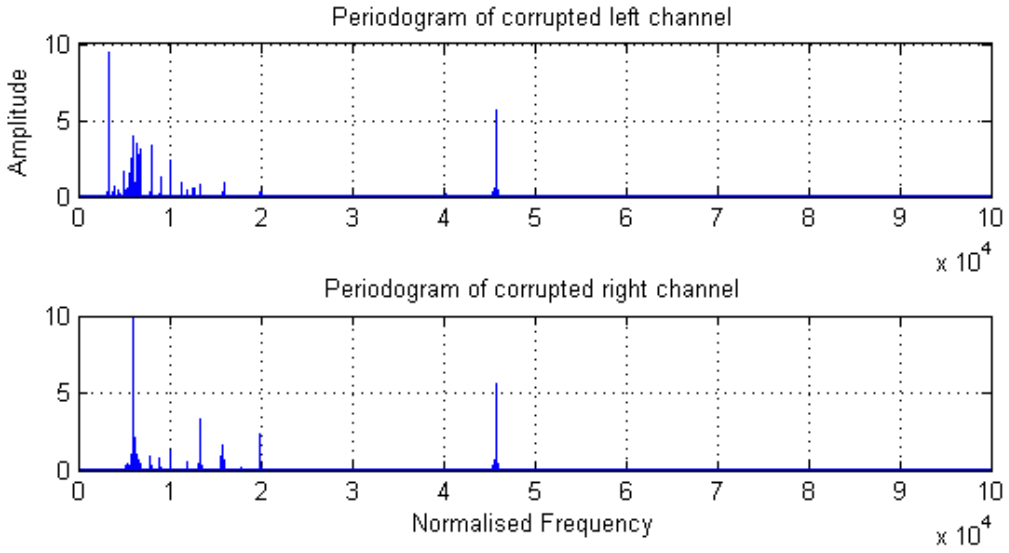

Figure 1. Periodograms of corrupted music

Figure 1 shows the corrupted periodograms of both channels. It is observed that there is one single peak at around 0.03 radians/second, corresponding to around 1500 Hz. From spectrum it seems that this peak is out of normal music frequency band, but it is still possible in some music to have this high frequency sound throughout the song. As a result, it is difficult to identify the noise peak without any comparison in this case.

5.2

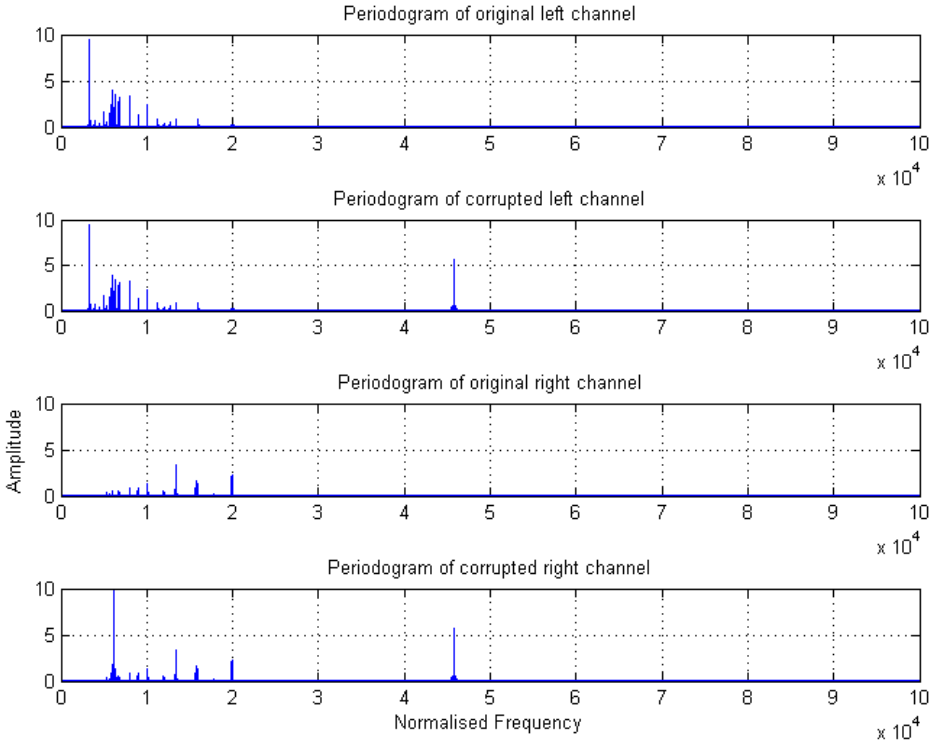

Figure 2. Periodograms of corrupted and uncorrupted music

Figure 1 shows the periodograms of original and corrupted music in both channels. It is obvious to identify noise spectrum in this case. For left channel, the noise is at 0.03 radians/second. While for right channel, noise peaks are at 0.002 and 0.03 radians/second. Comparing the corrupted and original data provide us with information about noise peaks.

5.3

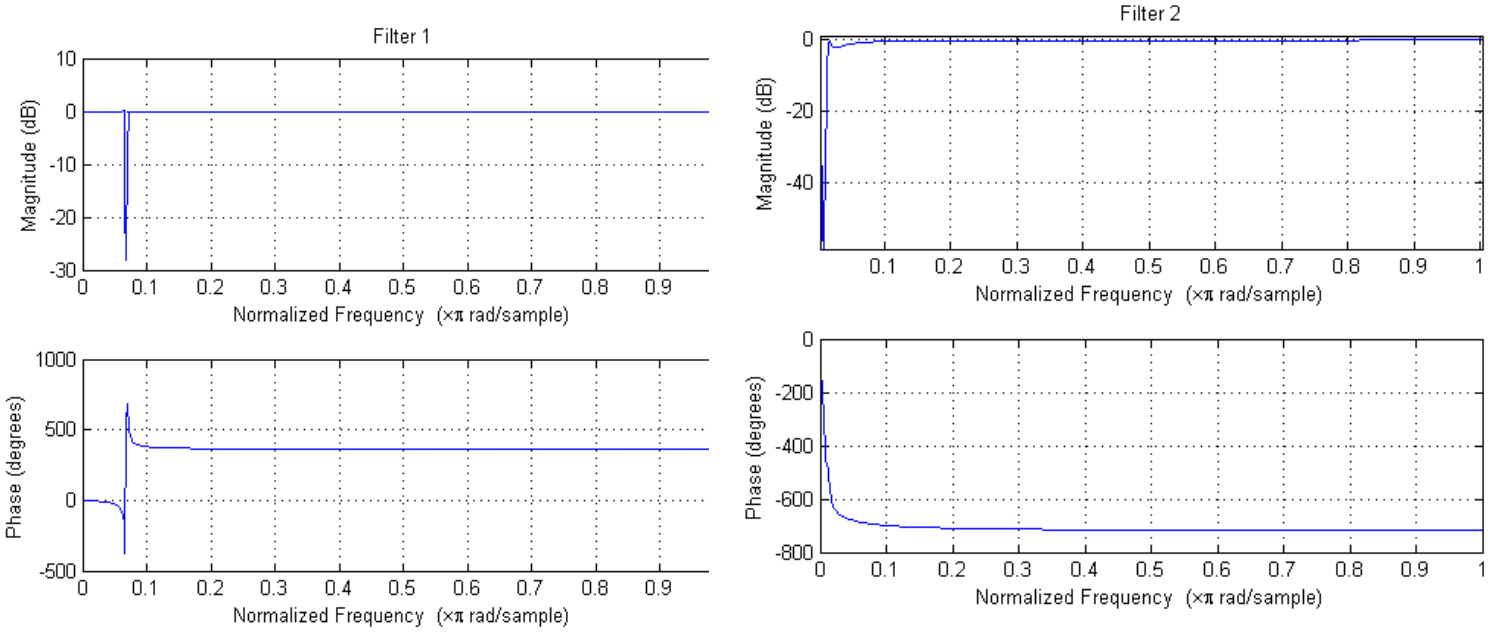

Figure 3. (Left) First filter response, (right) Second filter response

Figure 3 shows the filter design to remove the noise peak identified in different frequencies. In this section, I choose Chebyshev filter, which has sharp transition band in order to maximise the music quality. From corrupted music, we figure out that the noise peak is at 206.6 Hz and 1500 Hz. A band-stop filter is designed to attenuate the signal at that small frequency region. Normalised frequencies are then passed to the cheby1 function. Order number are chosen to be 10 and 6 correspondingly, and stop-band ripple are set to a reasonably small number such as 0.1. Note that filter 2 is only designed to remove the noise at low frequency, which does not exist in left channel. As a result, channel 1 only requires to pass the first filter.

5.4

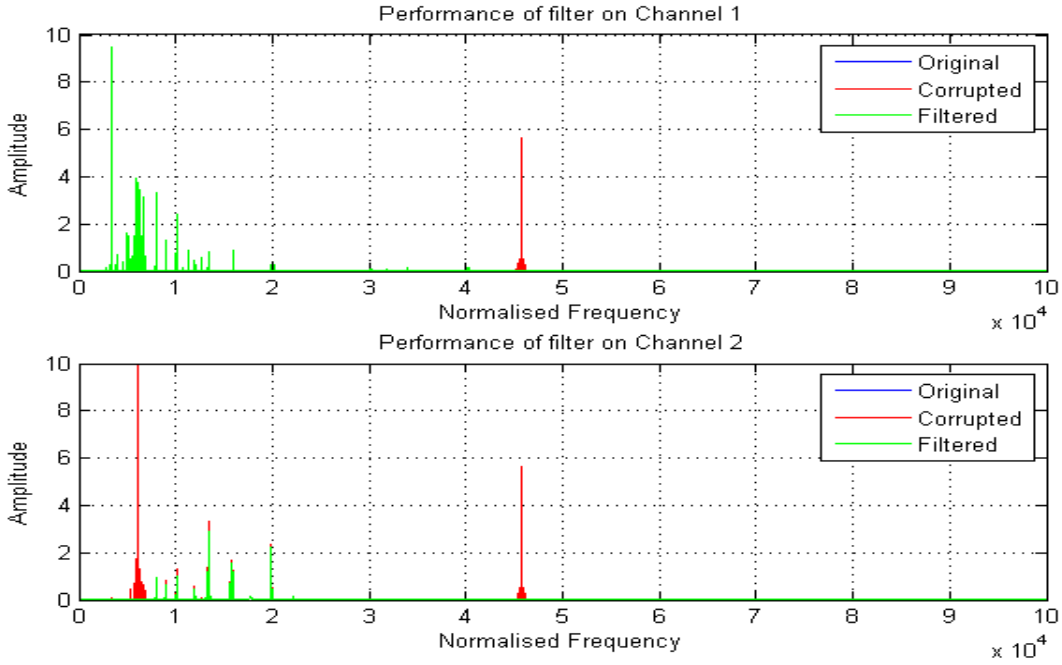

Figure 4. (Top) Channel 1 noise removal, (Bottom) Channel 2 noise removal

Figure 4 shows the effect of filter on both channels. It is obvious that the noise bins at stated frequencies are attenuated. Most original music nodes are kept after filtering. Also, in the listening test, tick noise are completely removed and not noticeable by listener.

Quantitatively, using the $\frac{\|P_c - \hat{P}_c\|}{\|P_c\|}$ formula , we obtain the difference between filtered signal and original signal. Smaller result means more signals are remained after filtering. However, it does not mean noise removal is better, since sometimes noise removal will sacrifice some music quality. In this experiment, the quantitative performance measure for channel 1 and 2 are 0.0072 and 0.24 correspondingly. Some frequency bins in channel 2 are attenuated in low frequency part in order to remove the low frequency noise completely.

5.5

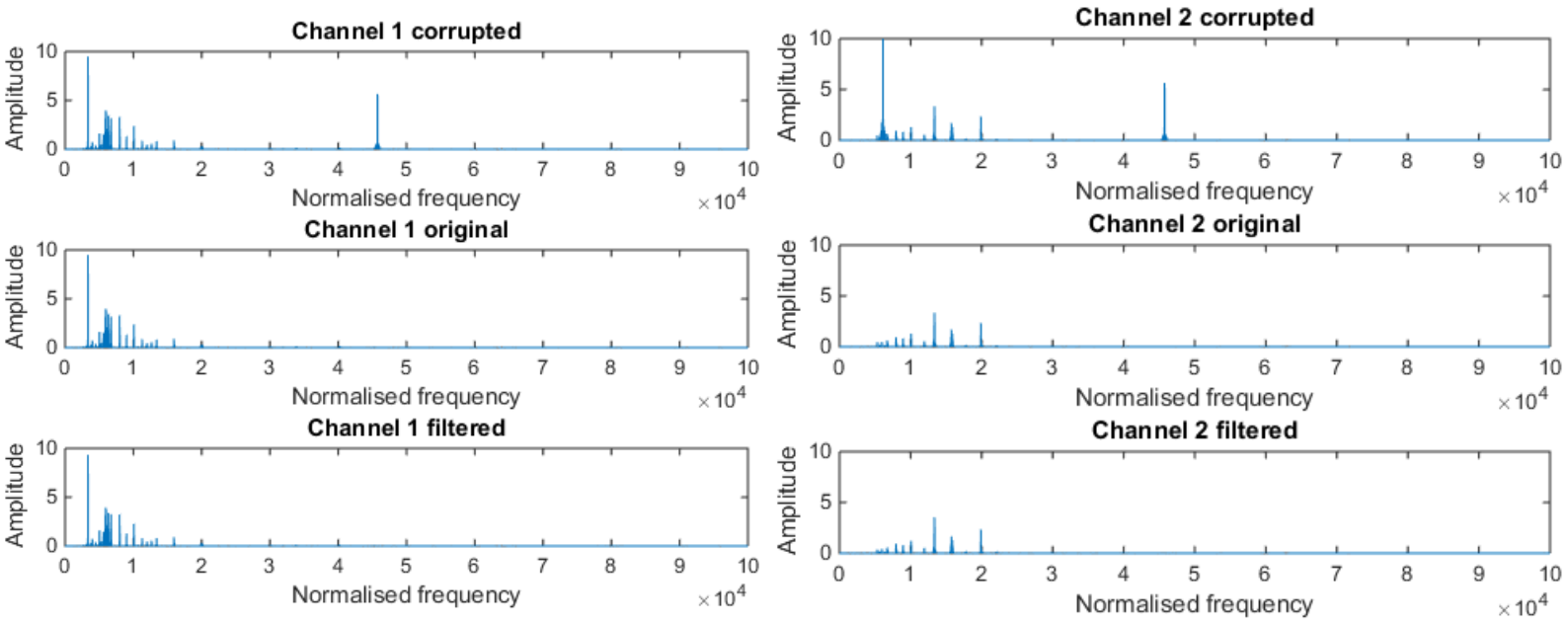

Figure 5. (Left) Channel 1 noise removal, (Right) Channel 2 noise removal for s2h

Figure 5 shows the nosie subtraction effect on both channels, it is obvious that noise peaks are significantly attenuated. Different parameter settings lead to different filtered result. Adaptation gain determines how fast coefficients react. Since this is a non-stationary case, reasonably higher adaptation gain will lead to better filter coefficients. But when adaptation gain is too large, inaccurate coefficients may be generated. Also, higher filter order increase the noise subtraction result at the cost of complexity. $\frac{\|P_c - \hat{P}_c\|}{\|P_c\|}$

Using quantitative performance measure we calcuated the values when order is set to 19. The value for channel 1 is 0.03 and for channel 2 is 0.054.

5.6

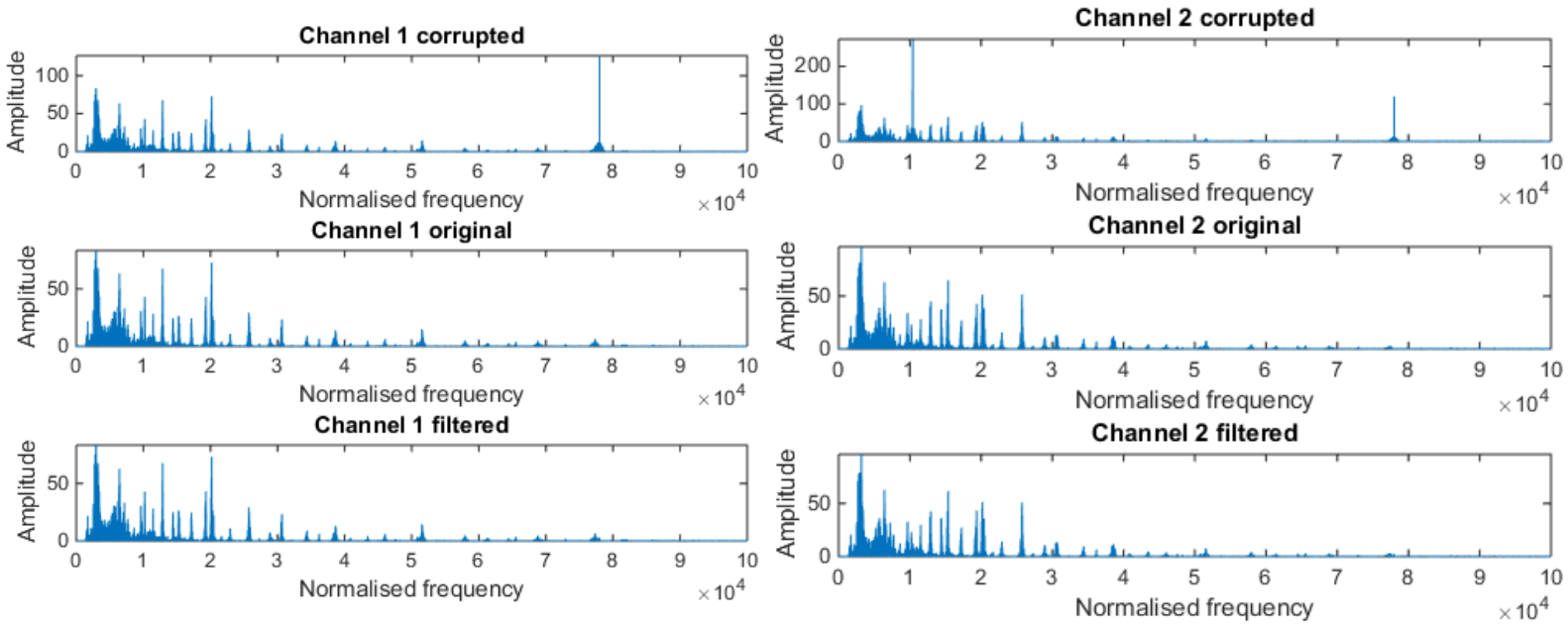

Figure 5. (Left) Channel 1 noise removal, (Right) Channel 2 noise removal for um

Figure 6 shows the noise removal performed on the second musical track. The change on periodogram shows clearly that noise peak is attenuated. Quanlitively, the sound removal effect is much better than the first track, this is because the noise peak is large, so that it is easy to be tracked my adaptive filter. Quantitatively, the performance measure for channel 1 and channel 2 are 0.01 and 0.07 correspondingly, which shows better nosie removal effect.