Imperial College
London

COURSEWORK

IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL ELECTRONIC ENGINEERING

Advanced Signal Processing

Author:
Keidi Kapllani (CID: 00946701)

Date: July 6, 2018

Contents

1 Random Signals and Stochastic Processes

1.1 Statistical Estimation

1.1.1 Expected Value

We generate 1000 random variables using the Matlab function `rand`

Theoretical Mean	Sample Mean
0.5127	0.5127
0.5115	0.5115
0.5006	0.5006
0.4870	0.4870
0.5022	0.5022

Table 1: Theoretical and Sample mean

Table 1 represents the values obtained from calculating the theoretical mean, using Matlab function `mean` and sample mean. As we can see clearly the values obtained from the theoretical mean and the sample mean the values are consistent and have the same degree of accuracy. Therefore we can conclude that both the sample mean and theoretical mean are identical.

1.1.2 Standard Deviation

Table 2 shows the theoretical standard deviation and sample standard deviation for some values from our random signal.

Table 2: Standard Deviation

Theoretical Standard Deviation	Sample Standard Deviation
0.2890	0.2890
0.2897	0.2897
0.2837	0.2837
0.2883	0.2883
0.2883	0.2883

We can observe that similar results are obtained as with the Expected Value leading to the Theoretical and Sample Standard Deviation to be consistent and the same degree of accuracy.

1.1.3 Bias Calculation

We plot the bias for standard deviation and mean, shown in Figure 1.1.

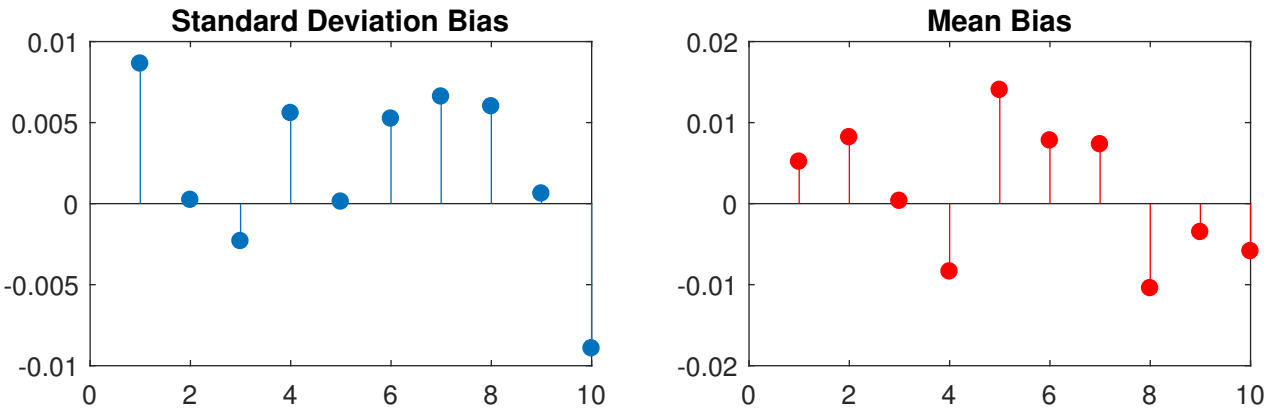

Figure 1.1: Standard deviation and mean bias

We can observe that the randomly generated samples tend towards 0, for both *Mean* and *Standard Deviation*, clustering around their respective theoretical values.

1.1.4 Probability Density Function

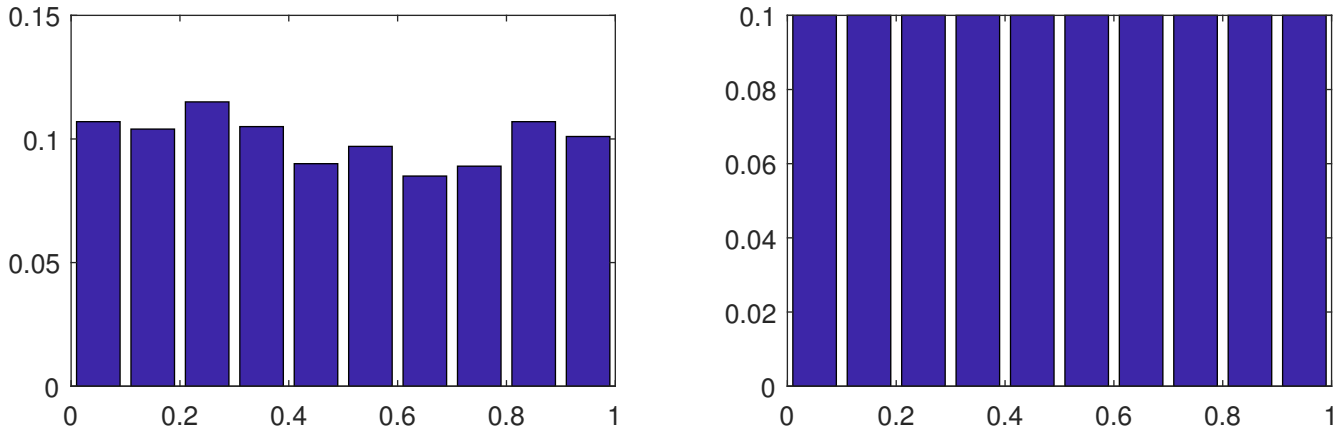

Figure 1.2: PDF

We can clearly observe from the bar graphs in Figure 2 that as the number of samples increases the results converge to *0.1* which is the theoretical value. The number of bins considered is 10 and since it allows to observe a clear pattern in the analysis must be considered viable.

1.1.5 Gaussian Random Variable

We repeat steps 1-4 with a Gaussian Random Variable generated using the *randn* MATLAB function and we observe a mean of approximately 0, namely 0.0478 and a standard deviation of 1, namely 1.0282. The experimental test agree with out theoretical predictions as we can observe from Figure 3:

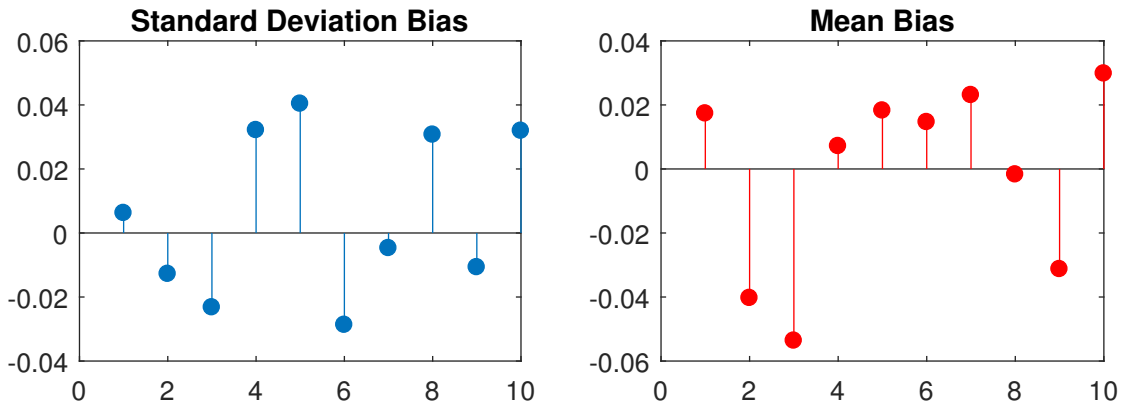

Figure 1.3: Gaussian Variable Bias

Both values fluctuate around their theoretical predictions. Furthermore by plotting the pdf of our random variable,using the hist function we can see that it resembles the theoretical pdf of a Gaussian Random Variable.

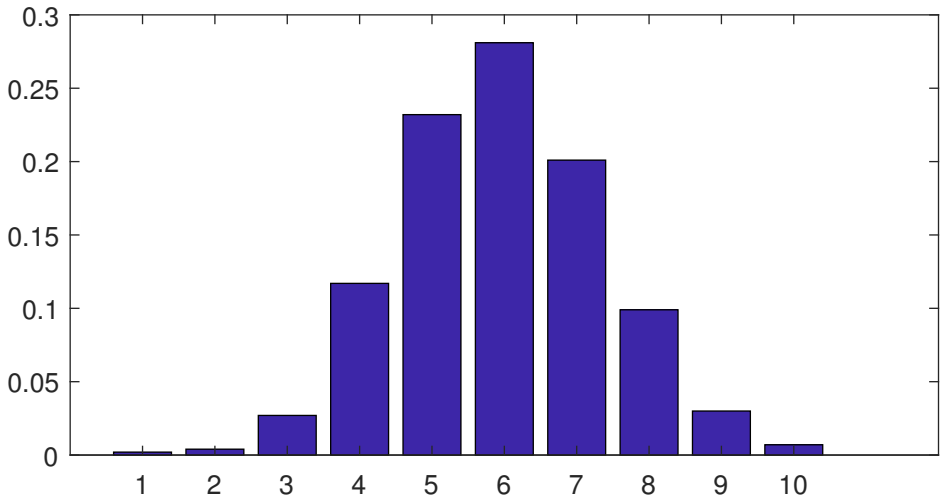

Figure 1.4: Gaussian Variable Bias

1.2 Stochastic Processes

1.2.1 Ensemble Mean and Standard Deviation

The graphs below show the ensemble mean and standard deviation for process rp1:

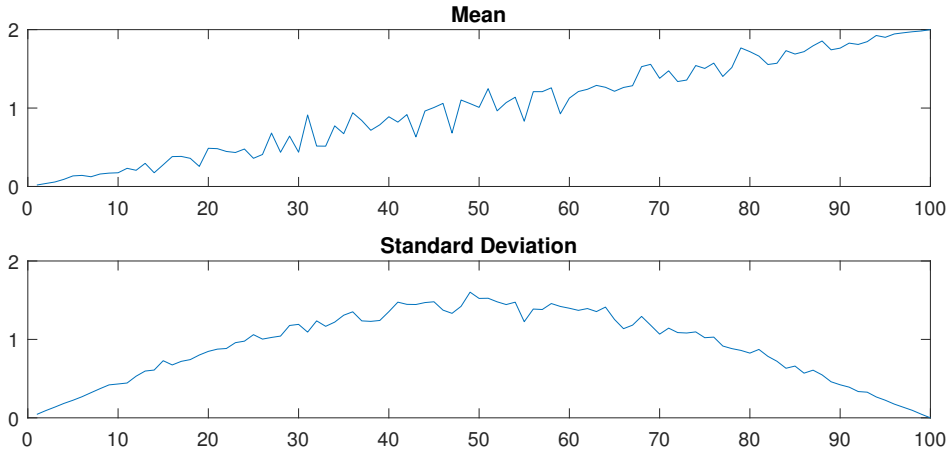

Figure 1.5: Ensemble Mean and Standard Deviation for rp1

We can observe clearly that this is not a stationary process as the ensemble mean changes with time, in this case it seems to be linearly increasing with time with an approximated function $mean = time/50$. The standard deviation also changes with time, not linearly but with an approximate function $Std = 1.44 \times sin(\frac{n \times \pi}{N})$, which leads us to the conclusion that rp1 is a non-stationary process.

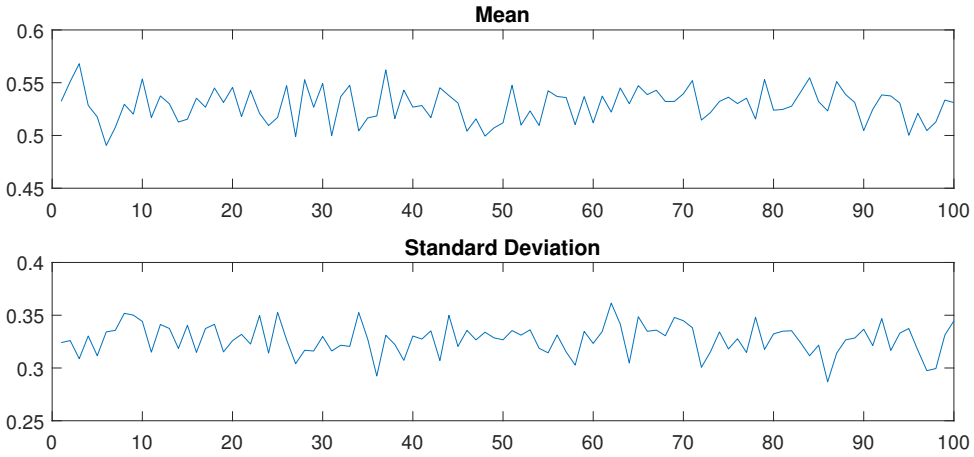

Figure 1.6: Ensemble Mean and Standard Deviation for rp2

Figure 6 are the graphs of ensemble mean and standard deviation for process rp2. In this process the ensemble mean is constant, with value fluctuating between 0.5-0.55 and a standard deviation of 1/3 which leads us to the conclusion that rp2 is a stationary process.

Figure 7 shows the graphs for process rp3.

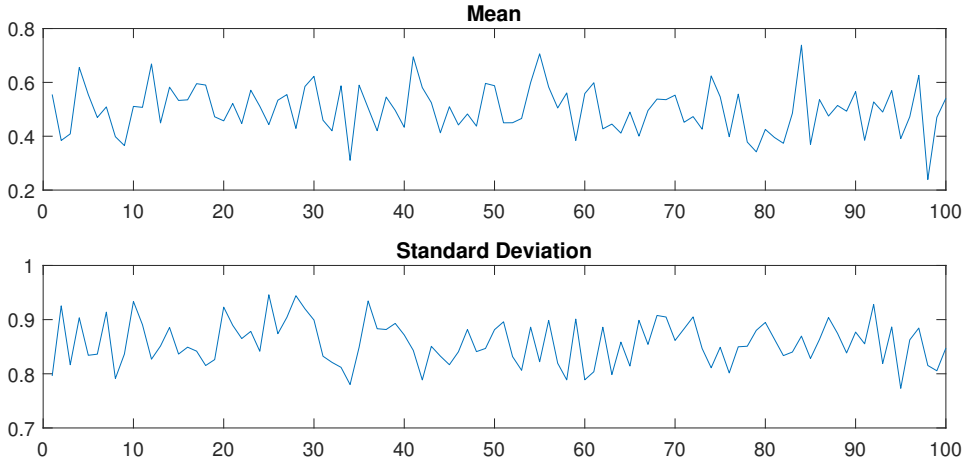

Figure 1.7: Ensemble Mean and Standard Deviation for rp3

Again the graphs suggest that this is a stationary process with mean 0.5 and standard deviation 0.85.

1.2.2 Ergodicity

In this step we take 4 samples at 1000 different instances in time. For process rp1 we can observe that it is not an ergodic process because the averages of ensembles calculated does not agree with the time average discussed in the previous step. Hence the inspection of many samples in one point in time will not be able to predict the complete properties of the system. Process rp2 is also not ergodic since each set of occurrence present significantly different values. We can see the difference for each occurrence in Table 3.

Table 3: Process rp3

Expectation of expectation of mean	Expectation of standard deviation
0.4803	0.8195
0.5188	0.8178
0.5105	0.8109
0.5052	0.8197

The only ergodic process is rp3. For rp3 we can define the properties of the system in one run since the expected value and the standard deviation of the system remain constant.

1.2.3 Mathematical Description

Process rp1

The process rp1 can be written as:

$$P_n = sin(\frac{n * \pi}{N}) * b * c + a * n \tag{1}$$

where a and b are constants, c is a randomly distributed value from -0.5 to 0.5 with mean 0 and standard deviation 1 and n is a constant which represents the iteration number over the total iterations N.
The expected value of process rp1 is:

$$\begin{aligned} E[P_n] &= E(sin[\frac{n\times\pi}{N})\times b\times c + a\times n] \\ &= E[sin(\frac{n\times\pi}{N})\times b]\times E[c] + E[a\times n] \\ &= E[sin(\frac{n\times\pi}{N})\times b]\times 0 + E[a\times n] \\ &= E[a\times n] = a\times n \end{aligned} \tag{2}$$

For the standard deviation we first calculate the variance:

$$\begin{aligned} \sigma_P^2 &= Var(P_n) = E[P_n^2] - E[P_n]^2 \\ &= E[sin^2(\frac{n\times\pi}{N})\times b^2c^2 + 2\times sin(\frac{n\times\pi}{N})\times abcn + a^2n^2] - a^2n^2 \\ &= E[c^2]E[sin^2(\frac{n\times\pi}{N}]\times b^2] + E[c]E[2\times sin(\frac{n\times\pi}{N})\times abn] + E[a^2n^2] - a^2n^2 \\ &= E[c^2]sin^2(\frac{n\times\pi}{N})\times b^2 \\ &= \frac{1}{12}sin^2(\frac{n\times\pi}{N})\times b^2 \end{aligned} \tag{3}$$

And the variance is:

$$\sigma_P = \frac{1}{\sqrt{12}}sin(\frac{n\times\pi}{N})\times b \tag{4}$$

From this we can calculate the ideal values for expectation and standard deviation, since we also have the parameters $a = \frac{1}{50}$ and $b = 5$.

$$E[P_n] = \frac{n}{50} \tag{5}$$

and

$$\sigma_P = \frac{5}{\sqrt[2]{12}}sin(\frac{n\times\pi}{N}) \approx 1.443\times sin(\frac{n\times\pi}{N}) \tag{6}$$

Both these values fit perfectly with the experimental values obtained earlier in section 1.2.1.

Process rp2

The process rp2 can be written as:

$$P_n = A_n + B_n\times C_{n,t} \tag{7}$$

where n and t represent what the variable is random to, i.e A and B being random to the number of instances and C being random to both the number of instances and the number of realisations.

A and B are uniformly distributed constants between 0 and 1 that have $E[A] = E[B] = 0.5$ and $\sigma_{A,b} = \frac{1}{2\sqrt[2]{3}}$. C is a uniformly distributed random variable with $E[C] = 0$ and $\sigma_C = \frac{1}{2\sqrt[2]{3}}$. Thus the expectation with respect to time is:

$$E[P_n] = E[A_n] + E[B_n \times C_{n,t}] = E[A_n] = 0.5 \tag{8}$$

and the standard deviation:

$$\begin{gathered} Var[P_n] = E[P_n^2] - E[P_n]^2 = E[A_n^2 + 2A_nB_nC_{t,n} + B_n^2C_{n,t}^2] - E[A_n]^2 \\ E[A_n^2] = E[B_n^2] = Var[A_n] + E[A_n]^2 = \frac{1}{3} \\ \therefore Var[P_n] = \frac{1}{3} + \frac{1}{4} \times \frac{1}{12} - \frac{1}{4} = \frac{1}{9} \\ \sigma_{P_n} = \sqrt{Var[P_n]} = \frac{1}{\sqrt{9}} \end{gathered} \tag{9}$$

Both the expected value and the standard deviation correspond to the experimental values for process rp2.

Process rp3

Process rp3 can be expressed mathematically

$$P_{n3} = m \times X + a \tag{10}$$

where a and m are constants of 0.5 and 3 respectively and X is a uniformly distributed random variable between $[-0.5, 0.5]$. The expected value of the process can be found nominally as it is the expected value of X which is 0.5. The standard deviation in turn can be calculated as:

$$\begin{gathered} Var[P_{n3}] = E[P_{n3}^2] - E[P_{n3}]^2 = E[m^2X^2 + a^2 + 2amX] - 0.25 = 0.25 \\ \sigma_{P_{n3}} = \sqrt{Var[P_{n3}]} = \frac{3}{2\sqrt{3}} \approx 0.86 \end{gathered} \tag{11}$$

1.3 Estimation of p.d.f

1.3.1 Pdf of Random Gaussian Variable

The following script is used to approximate the pdf:

```
N=20000;
x=randn(1,N);
[a,b]=hist(x,100);
figure
a=a/N;
bar(b,a);
xlabel('X');
ylabel('Probability');
```

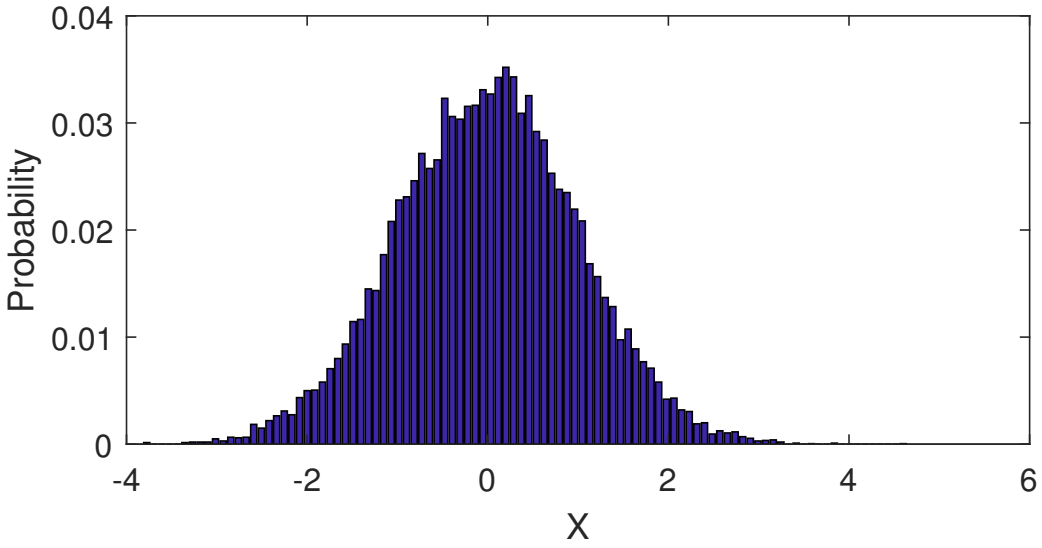

Figure 1.8: Pdf of Gaussian Variable

1.3.2 P.d.f of stationary ergodic process

The only stationary and ergodic process is rp3.

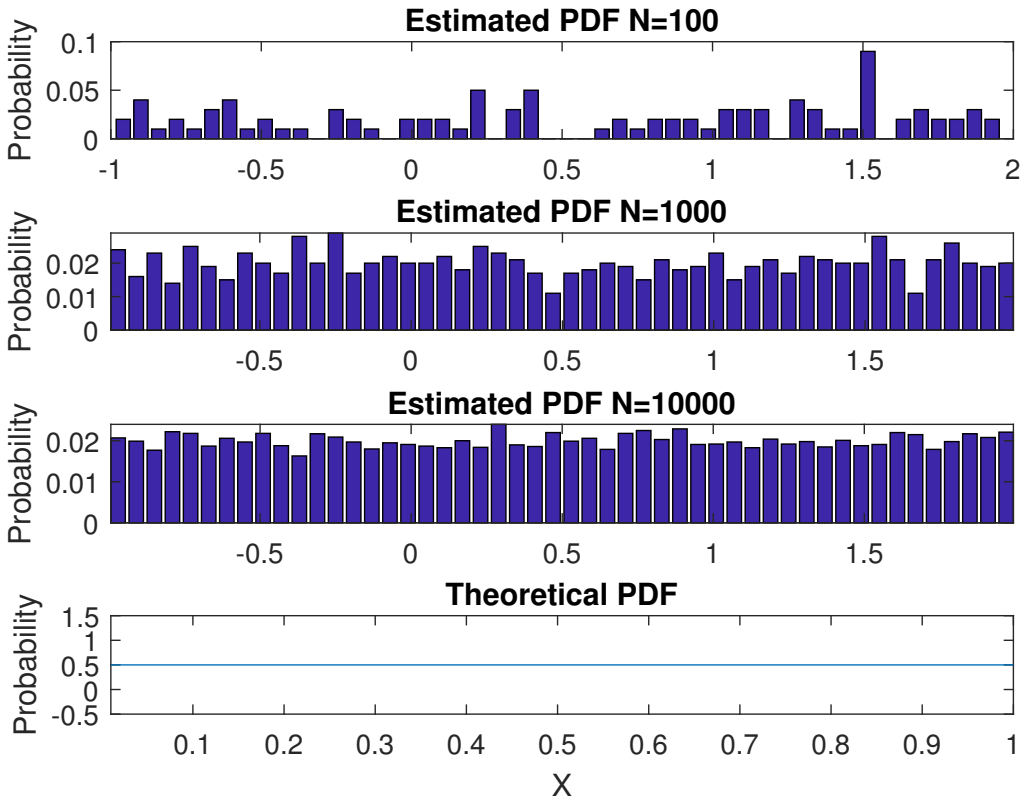

Figure 1.9: PDF of rp3 for various N

We can observe quite clearly that as N increases the sample pdf approaches increasingly the shape of the theoretical pdf.

1.3.3 Non-stationary process

A non-stationary process whose mean changes from 0 to 1 after N=500 is rp1. As such trying to apply the pdf function written in the previous step yields the following results:

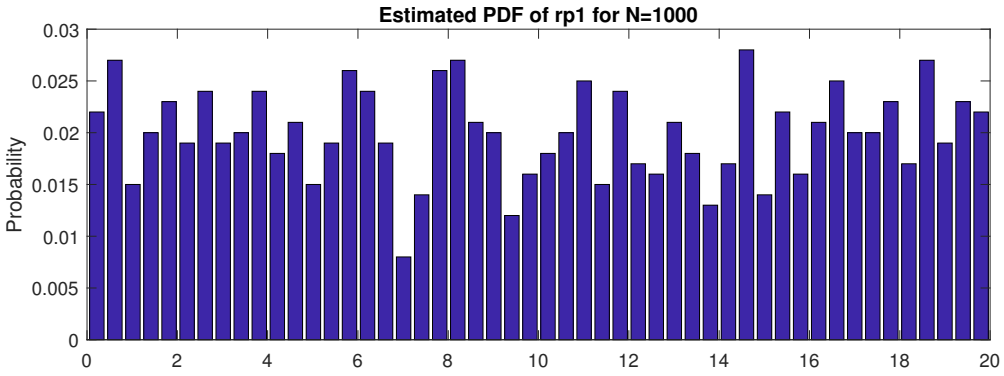

Figure 1.10: Pdf of rp1

We can see that the pdf implies the function amplitude is random and between 0 and 20. However since this is not a stationary process as discussed before, we know that this is not the case. Hence we can conclude that the use of the *hist* function causes a loss of information.

2 Linear Stochastic Modelling

2.1 Autocorrelation function

2.1.1 Matlab *"xcorr"* function

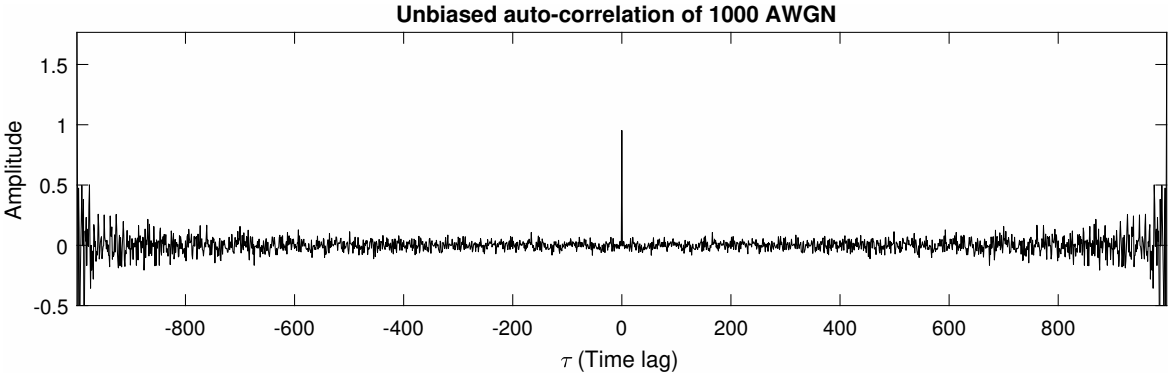

Figure 2.1: Unbiased correlation of white Gaussian noise.

The function for the unbiased estimate of the ACF is:

$$\hat{R}_x(\tau) = \frac{1}{N-|\tau|} \sum_{n=0}^{N-|\tau|-1} x[n]x[n+\tau],\ \tau = -N+1, ..., N-1 \tag{12}$$

Figure 11 is the plot of the ACF of 1000 AWGN samples. From the graph we notice firstly a discrete Dirac function at $\tau = 0$ and that the function is symmetric around $\tau = 0$. This is due to the autocorrelation being the correlation of a signal with itself hence at equals distances from $\tau = 0$ equal samples will be compared. Furthermore we can see an increase in amplitude as the value of τ increases from $\tau = \pm 500$ onwards. This is because after this value of τ, less than half the values of the vector **x** storing the AWGN samples are available for comparison. This also explains why the amplitude is largest at $\tau = \pm 999$ as in this point only one value is available for comparison.

2.1.2 Zoomed in version

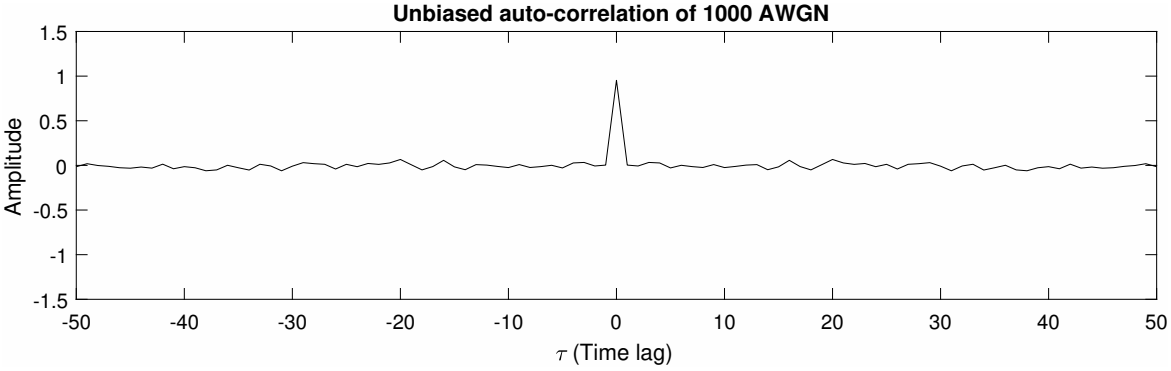

Figure 2.2: Zoomed in instance of figure 11.

In Figure 2.2 we notice that for values other than $\tau = 0$ the graphs remains fairly constant around 0. This, as explained in the above section, is due to the samples completely overlapping and cancelling each other at the origin giving a better estimate than for larger values of τ. The cleaner look of the graph comes from less samples being present in the zoomed instance.

2.1.3 Empirical bound on τ

We can clearly notice in equation (12) that for values of τ above a certain limit the value of the ACF starts to diverge suggesting that for a large lag the signal is similar to itself τ samples before. However this is not expected as the samples are taken AWGN.
In the case presented in above sections, where 1000 AWGN were taken, $\tau = \pm 500$ was a good limit as past this value less then half the samples will be available for comparison and the probability of resemblance increases. This is confirmed if we look at $\boldsymbol{R}_x$ where we see in the term $\frac{1}{N-|\tau|}$ that for an increasing τ the accuracy gets smaller due to less samples being available.
One way of explaining this is that the probability of two samples having the same sign (i.e constructive interference) decreases as the number of samples increases. As such if we had only one sample to compare the probability of it being the same sign is 50%, for two samples is 25% etc. Consequently, it is clear why the maximum amplitude is at $\tau = 0$, decreases as τ increases until it reaches the limit $\tau = \frac{N}{2}$ where it starts increasing again due to sample size increasing.

2.1.4 Moving Average Filter

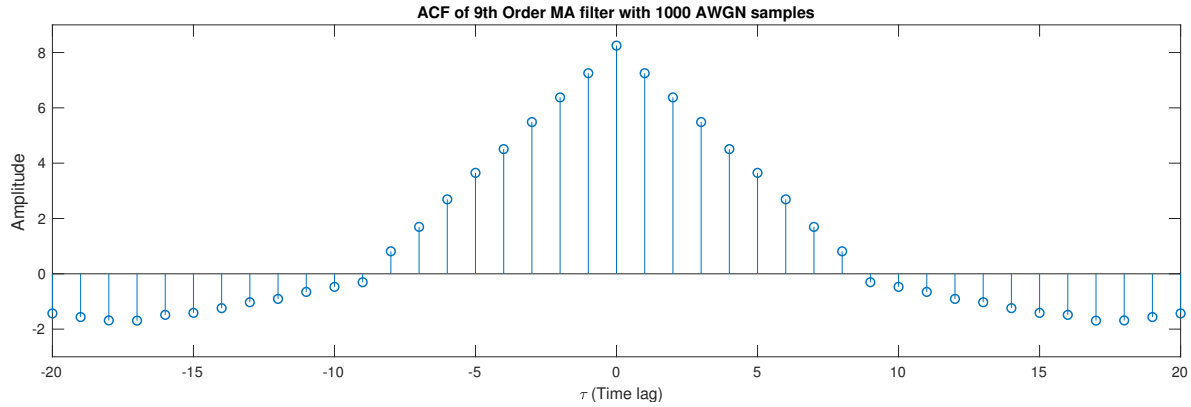

Figure 2.3: ACF of 9^{th} Order MA filter on 1000 AWGN Samples

In Figure 2.3 we can see the effect of passing the AWGN samples through a MA filter of order 9.
The ideal ACF for AWGN samples passed through MA filter is: $AFC_{ideal}(\tau) = N \times \Lambda(\frac{\tau}{N})$ where $\Lambda(t)$ is the triangle function. This is because the MA filter works by taking the coefficients it has and adding up past elements. Hence, by taking the average,

elements with index 1 to N will be correlated with correlation increasing as you approach the original element, explaining the triangle shape of the graph.
The filter order determines the width of the triangle in the graph, with the triangle in our case being at $\tau = \pm 9$ since we have a 9nth order filter.

2.1.5 Stochastic process correlation

We have that $\mathbf{X}_n$ is an uncorrelated process, $\mathbf{Y}_n$ is the results of a stochastic process and:

$$\boldsymbol{R}_Y(\tau) = \boldsymbol{R}_X(\tau) * \boldsymbol{R}_h(\tau) \tag{13}$$

Then we have that $\boldsymbol{R}_Y(\tau) = \boldsymbol{R}_h(\tau)$ since $\boldsymbol{R}_X(\tau) = \boldsymbol{R}_h(\tau)$, which is the results of the autocorrelation function of an uncorrelated process , and any function convoluted with $\delta(\tau)$ is equal to itself. Hence the ACF of process $\mathbf{Y}_n$ represents the autocorrelation of the impulse response.

2.2 Cross-correlation function

2.2.1 CCF of filtered with unfiltered samples

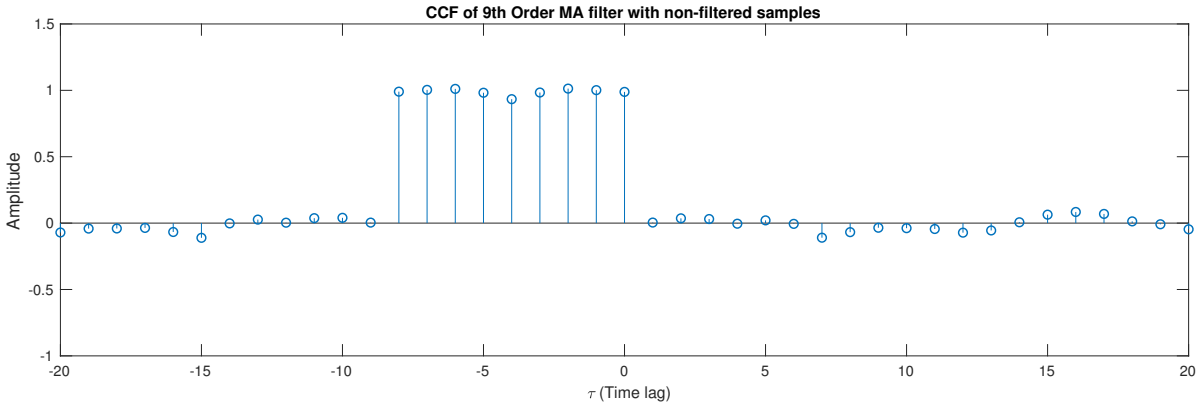

Figure 2.4: CCF of 9^{th} Order MA filter with unfiltered samples

Figure 2.4 shows the Cross-correlation of instances X and Y from the previous section where Y is the MA filtered version of X. The results, as it can be noticed clearly on the figure, is the impulse response (here it is time-reversed but it depends on whether you use *xcorr(x,y,'unbiased')* or *xcorr(y,x,'unbiased')*). This result is expected since we have the cross-correlation of X and Y which is $\boldsymbol{R}_{XY}(\tau) = \boldsymbol{R}_h(\tau) * \boldsymbol{R}_X(\tau)$, but we know that X is AWGN which is uncorrelated, that ultimately leads to $\boldsymbol{R}_{XY}(\tau) = \boldsymbol{R}_h(\tau)$ The impulse response is $h(\tau) = \sum_{i=0}^{N} \delta(-i)$ which is what we observed in Figure 2.4.

2.2.2 System identification using CCF

If we apply the CCF on two functions, one which we know is the result of a FIR filter and the other being the original function we can obtain the impulse response only

in the case when the original function is an uncorrelated process. From this we can deduce the order by the number of delta functions. Furthermore if the system is a FIR the coefficients can be approximated.

2.3 Autoregressive Modelling

2.3.1 Autoregressive Processes

In this section we need to verify the stability of an AR(2) process. 100 values of a1 and a2 were generated from the uniform distribution $a1 = [2.5, 2.5]$ and $a2 = [1.5, 1.5]$. Furthermore 1000 samples were generated using the function `randn`.The pairs were then examined to see whether they converge or diverge and the results were plotted in Figure 2.5. To obtain a clearer graph of the stability the number of pairs generated was increased from 100 to 1000.

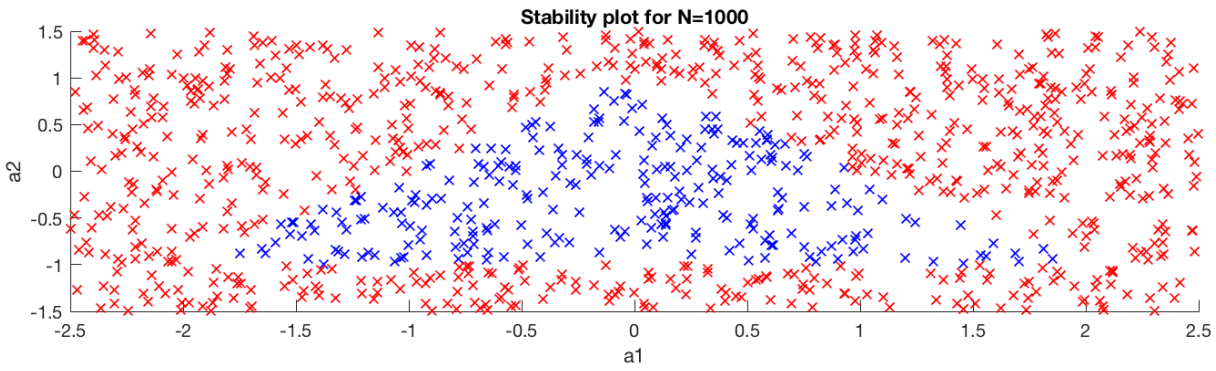

Figure 2.5: Stability plot of a1 and a2

2.3.2 Sunspot Data

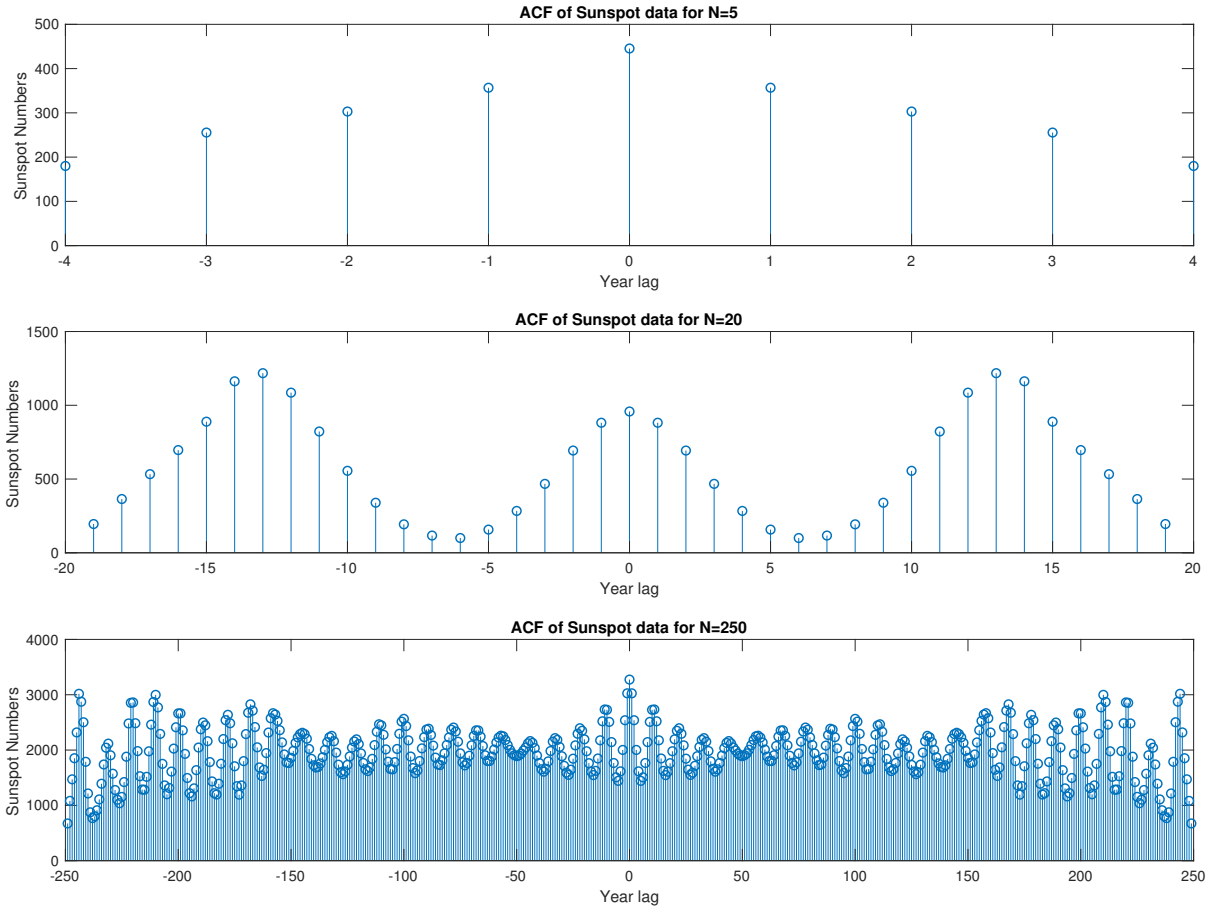

Figure 2.6: ACF of sunspot data for varying N

We begin by uploading sunspot data by using the command `load sunspot.da`. We then plot the ACF of sunspot data with varying N for N = 5, 20, 250. Initially we notice that for a small N(N=5) there is no information we can extract from the plot whereas for N=20 we can notice that the data is recurring every 13 years or so. In the plot for N=250 we notice that peaks are higher than in N=20 and some peaks are not visible anymore but this can be attributed to the issue of interpreting data past $\frac{N}{2}$ as discussed in Section 2.1.

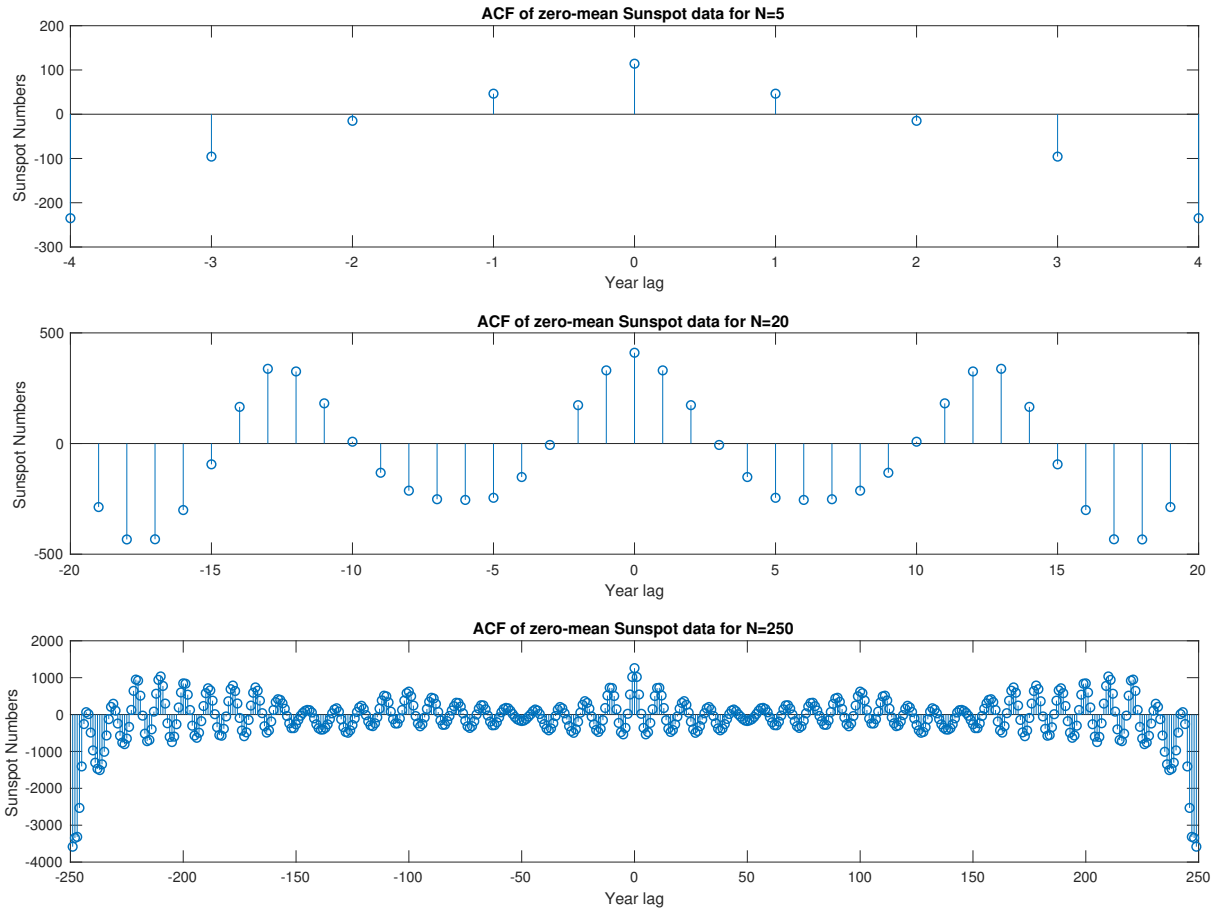

Figure 2.7: ACF of zero-mean of sunspot data for varying N

We proceed to zero-mean the data in order to see the effect that mean has on the ACF. The first thing to notice is that the apparent difference in peak amplitude between N=20 and N=250 is no longer observable. Furthermore the plot for N=250 can be observed better now. This can be explained by returning to the way ACF works, which is by multiplying every sample with a shifted version of itself. Hence by setting a zero mean we can force an equal representation in positive as well as negative magnitudes which means data does not accumulate throughout the ACF. This allows the periodicity to be more visible in the plot, with peaks representing periods of repetition.

2.3.3 Yule-Walker Equations

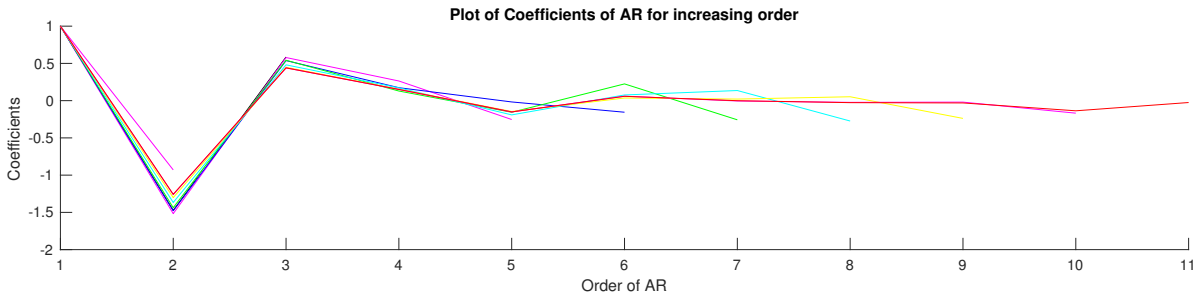

Figure 2.8: Plot of coefficients for various AR orders

In this section we try to calculate which model order of AR is a better approximate for the sunspot time series using the `aryule()` function in MATLAB. After calculating the coefficients We plot them for model orders $[1,10]$ in Figure 2.8. From this plot we can see that past $2^n d$ order all coefficients are smaller which leads us to believe this is the appropriate model. This is further confirmed when we plot the Partial Autocorrelation coefficients in Figure 2.9 where with the help of the guidance lines we can conclude that the sunspot time series is best approximated by AR(2).

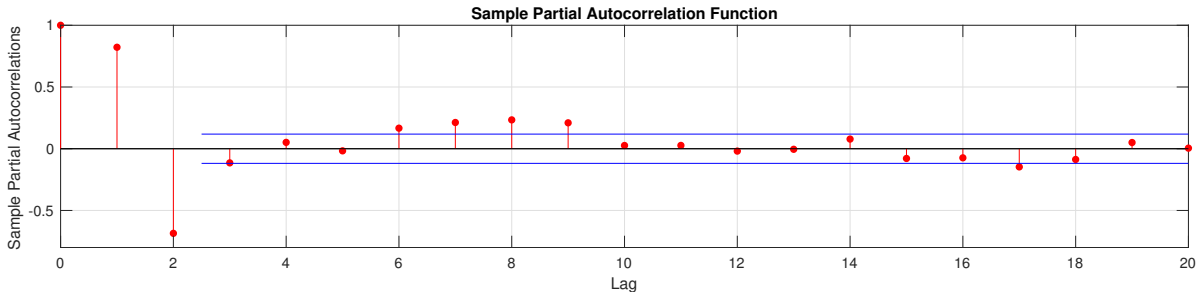

Figure 2.9: Partial Autocorrelation coefficients

2.3.4 MDL and AIC

We will now use the Minimum Description Length, Akaike Information Criterion and the corrected Akaike Information Criterion(AICc) in order to determine what is the most optimal order. The equations for which we will apply to the AR coefficients are the following:

$$\begin{aligned} MDL(p) &= \log(E_p) + \frac{p\log(N)}{N} \\ AIC(p) &= \log(E_p) + \frac{2p}{N} \\ AICc(p) &= AIC(p) + \frac{2p(p+1)}{N-p-1} \end{aligned} \tag{14}$$

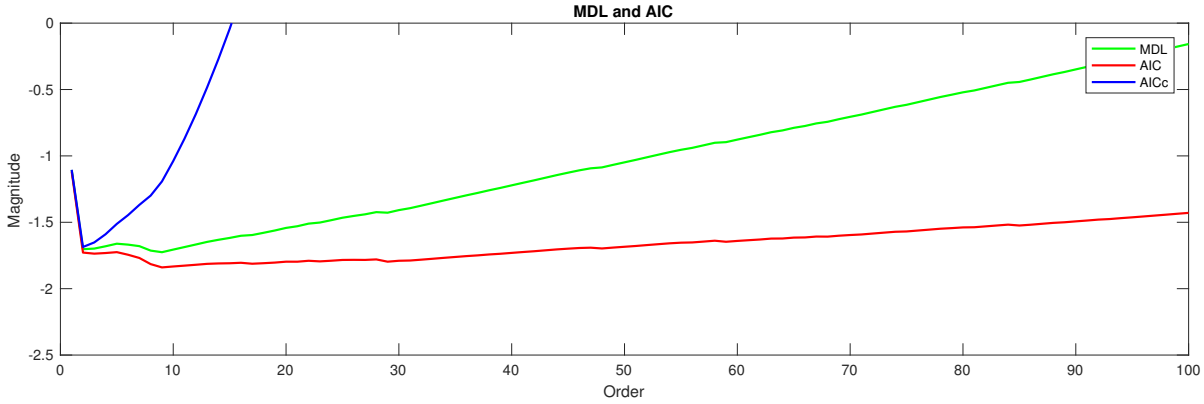

Figure 2.10: Expected error for increasing model order

The expected error for the different model orders are shown in Figure 2.10. We can see that the minimum value on both MDl and AIC occurs at model order 9 however for AICc it occurs at model order 2. Since the AICc penalises more for over-fitting the data we can conclude that the best choice would be model order 2, taken into account the values of AICc.

2.3.5 Predicting future sunspots

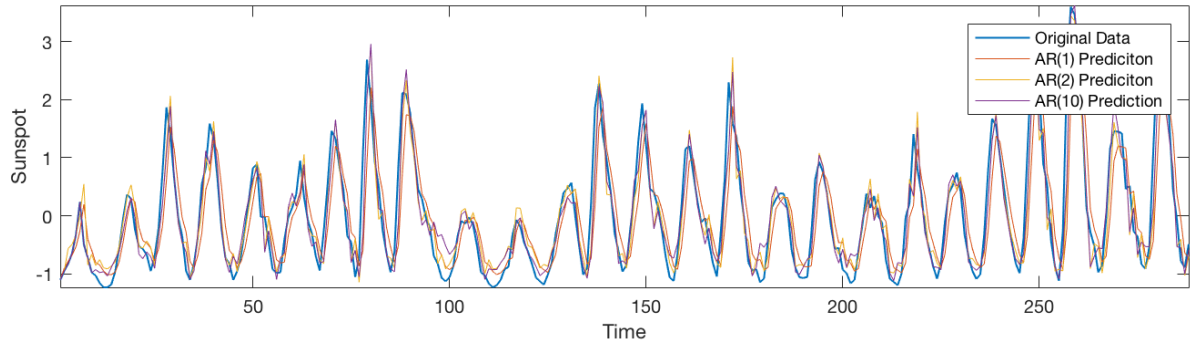

Figure 2.11: Prediction of sunspots for horizon 1

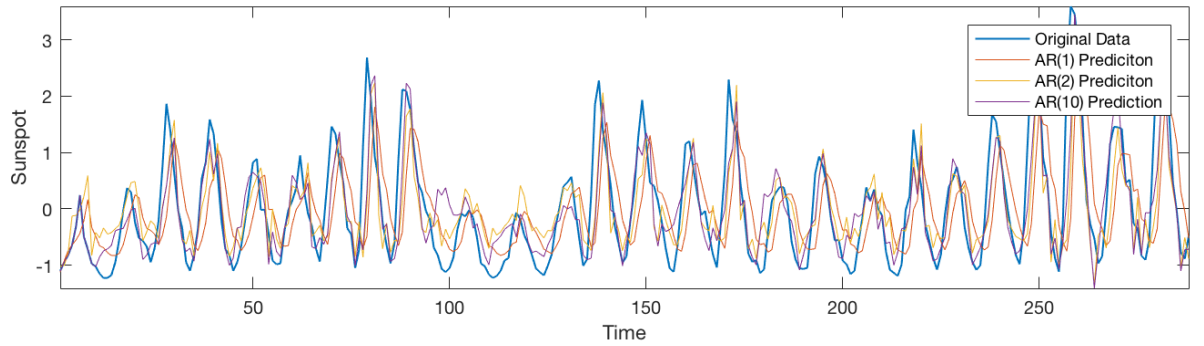

Figure 2.12: Prediction of sunspots for horizon 2

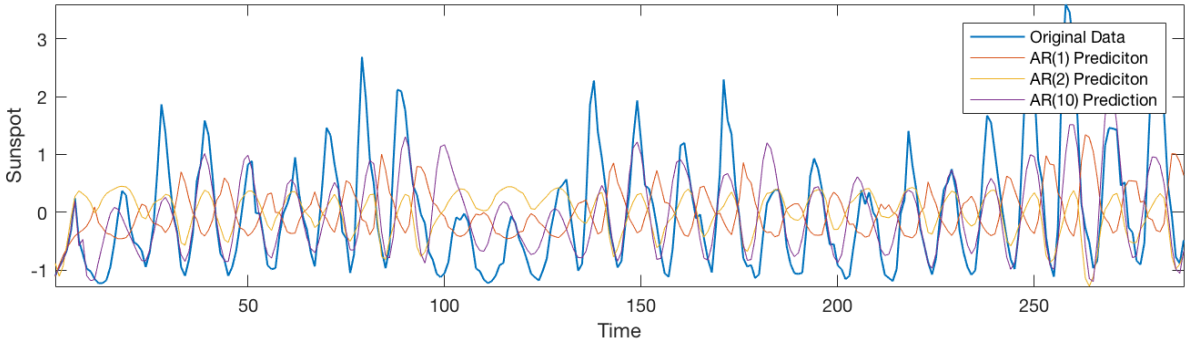

Figure 2.13: Prediction of sunspots for horizon 5

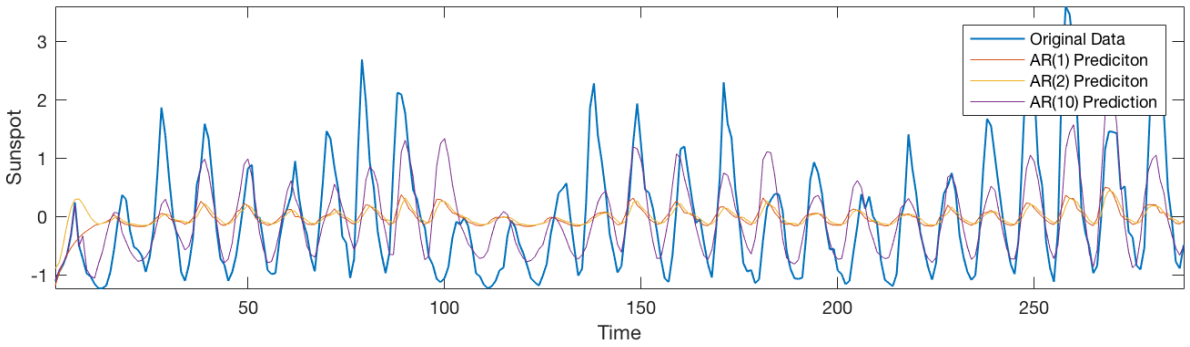

Figure 2.14: Prediction of sunspots for horizon 10

Figures 2.11 - 2.14 shows the prediction of sunspots using AR(1), AR(2) and AR(10) models for varying predictions horizons. It is noticeable that with increasing horizon the models all move away from the actual data. Furthermore model AR(10) over-fits the data and model AR(1) under-fits the data due to the difference in degrees of freedom.

2.4 Cramer-Rao Lower Bound

2.4.1 Choosing model order

Given the NASDAQ end-of-day prices for the index between June 2003 and Feb 2007 we are required to prove that AR(1) is sufficient to describe daily returns of the index.

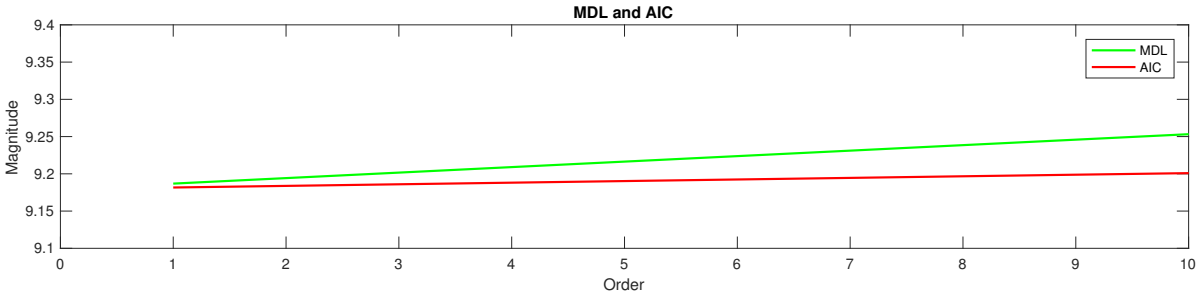

Figure 2.15: Expected error for NASDAQ data with increasing model order

If we compute the MDL and AIC of using the formulas stated in Section 2.3.4 we can see from Figure 2.15 that in fact the AR(1) model is sufficient to correctly describe daily returns as fro higher orders the magnitude of the error increases.

2.5 ECG from iAmp Experiment

Following the completion of the experiment we import the data to MATLAB and analyse it.

2.5.1 PDE of averaged heart rate

We calculate the heart rate in beats per minute (bpm) using the RRI according to $h[n] = \frac{60}{rr[n]}$. To obtain a smoother estimate of the data we averaged every 10 samples using $\hat{h}[n] = \sum_{i=10(n-1)}^{n+10} h[i]$. Using the `hist()` function we plot the PDE of the original and averaged heart rate, shown in Figure 2.16.

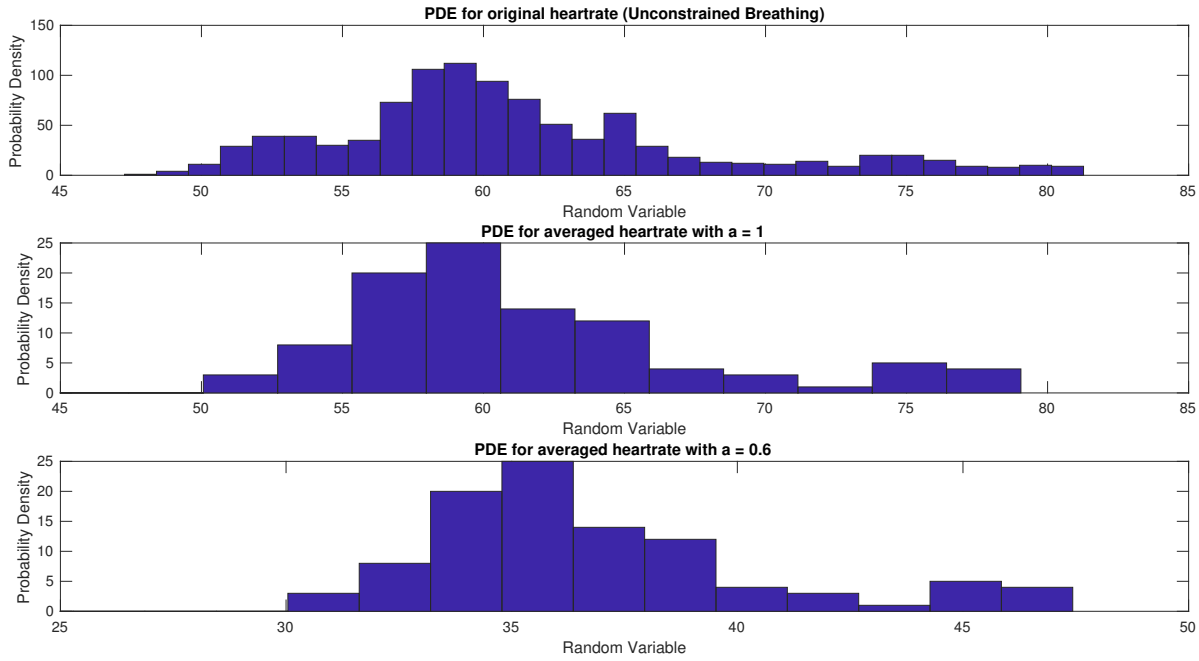

Figure 2.16: Probability Density Estimate for original and average heart rate

2.5.2 Effect of α on PDE

The average resting heart rate for an adult is between 60 and 100 bpm. Looking at Figure 2.11 we can see clearly that both the original and the $\alpha = 1$ averaged comply with these constrains whereas the $\alpha = 0.6$ doesn't. This is expected considering that for the averaged dataset with $\alpha = 0.6$ the inputs were scaled down by a factor of 0.6 hence the shift in heart rate displayed in the graph. For the non-averaged data the PDE is observed to be reasonably uniform indicating the range of heart rate rather then the exact shape. The PDE after averaging suggests for the case of $\alpha = 1$ that the heart rate is mostly likely close to 60 bpm.

2.5.3 ACF of RRI data

We calculate the ACF for the 3 trials from the RRI data. Firstly to ensure data is zero-mean we use the `detrend` and then we plot the ACF for the 3 trials, shown in Figure 2.17.

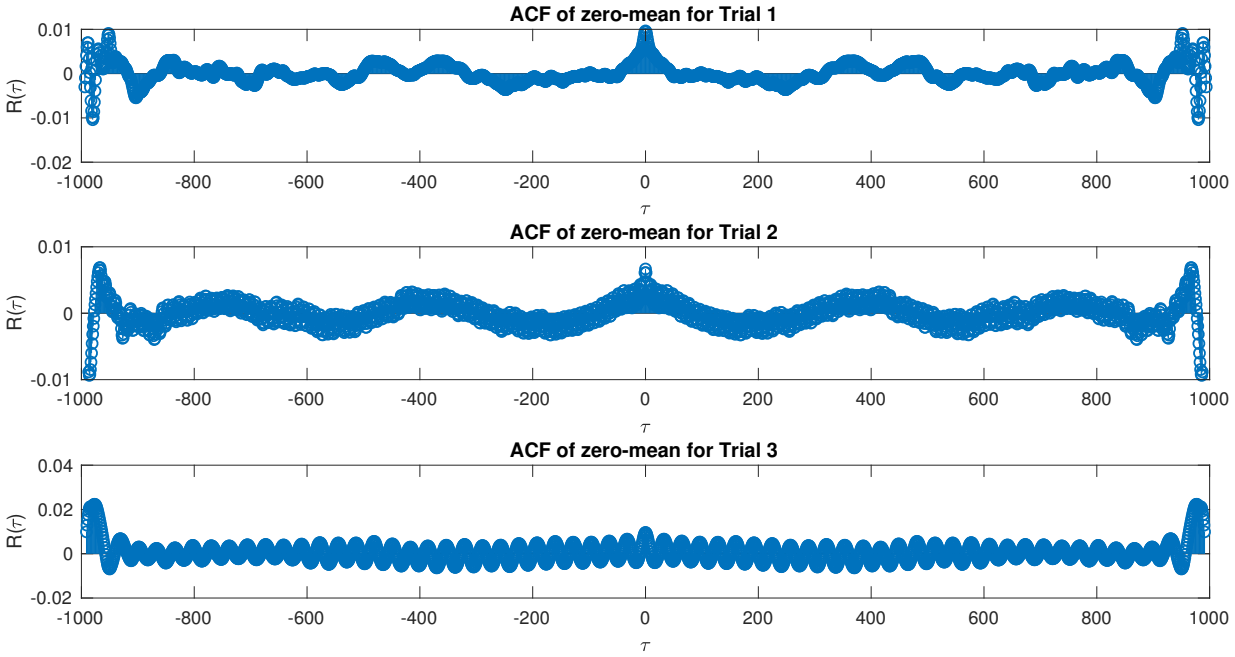

Figure 2.17: ACF for zero-mean RRI data from 3 trials

From the plot of the ACF in Figure 2.17 we can conclude that RRI data is an AR process. This is because we can see significant correlation in each trial where as if it were an MA filter it would be 0 for MA larger then the filter order.

2.5.4 Optimal AR(p) model

In the above section we concluded that the data could be an AR process. We now use the MDL and AIC criteria to determine the correct model order.

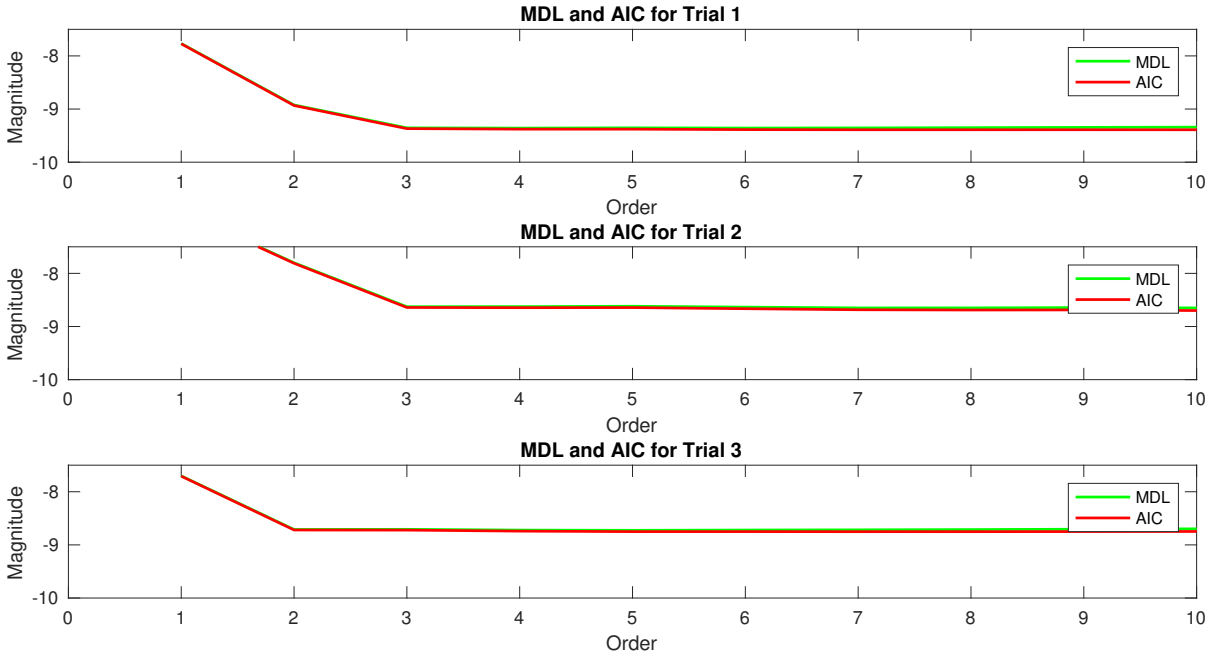

Figure 2.18: MDL and AIC applied to RRI data for all trials

From Figure 2.18, where we can see the plots of criteria for all trials, it is clear the the correct model for Trial 1 and 2 is AR(3), whereas for Trial 3 is AR(2).

3 Spectral Estimation and Modelling

The PSD gives a measure of the power present in each frequency for a signal. In practical situations since only a finite number of samples are available its autocorrelation function can only be approximated. One method to estimate the PSD is by using a periodogram. This approach makes use of the Fast Fourier Transform as follows:

$$\hat{\mathbf{P}}_X(f) = \frac{1}{N}|\sum_{n=0}^{N-1} x[n]e^{-j2\pi f\frac{n}{N}}|^2 \tag{15}$$

```
function [P,freq] = pgm(data)

N=length(data);
freq=(0:N-1)/N;
P=[];
for count=1:N
    e=exp((-1i)*2*pi*(count-1)*freq);
    P=[P (abs(data*e')^2)/N];
end
end
```

The above code is used to estimate the PSD of AWGN of 128, 256, and 512 samples and shown in Figure 3.1.

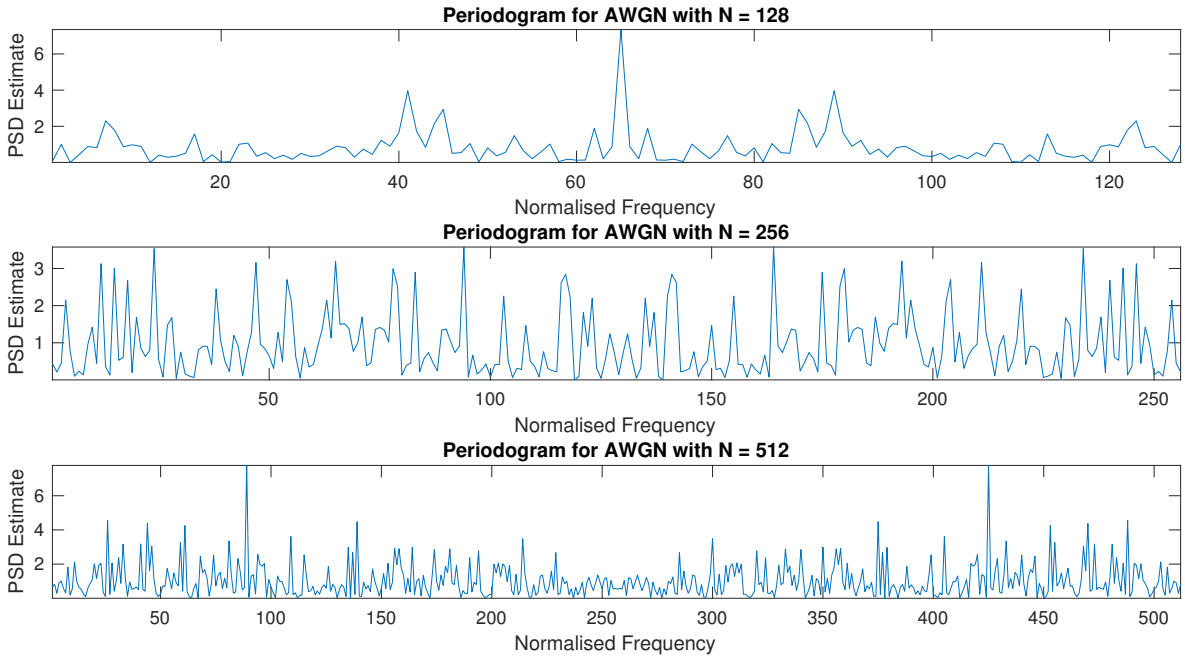

Figure 3.1: PSD Estimate for AWGN of N=128, 256, 512

The theoretical discrete time autocorrelation function for WGN derived in Section 2.1 was $R_X(\tau) = \delta(\tau)$. Using the Weiner-Khintchine relation, we can calculate the

theoretical PSD of AWGN: $\hat{\mathbf{P}}_X(f) = \frac{1}{N}|\sum_{\tau=\infty}^{\infty}\delta(n)e^{-j2\pi f tau}|^2 = 1$. It is observable from Figure 3.1 that as the number of N increases the periodogram tends to being constant more and more.

3.1 Averaged periodogram estimates

3.1.1 PSD Estimate Smoothing

An FIR filter with impulse response $0.2 \times [11111]$ is used to make the PSD smoother by taking the average of 5 consecutive samples. Figure 3.2 shows the filtered version of the PSD estimate. If we contrast Figure 3.1 and 3.2 we can see that be making the estimate smoother we can achieve a better PSD estimate tending towards unity.

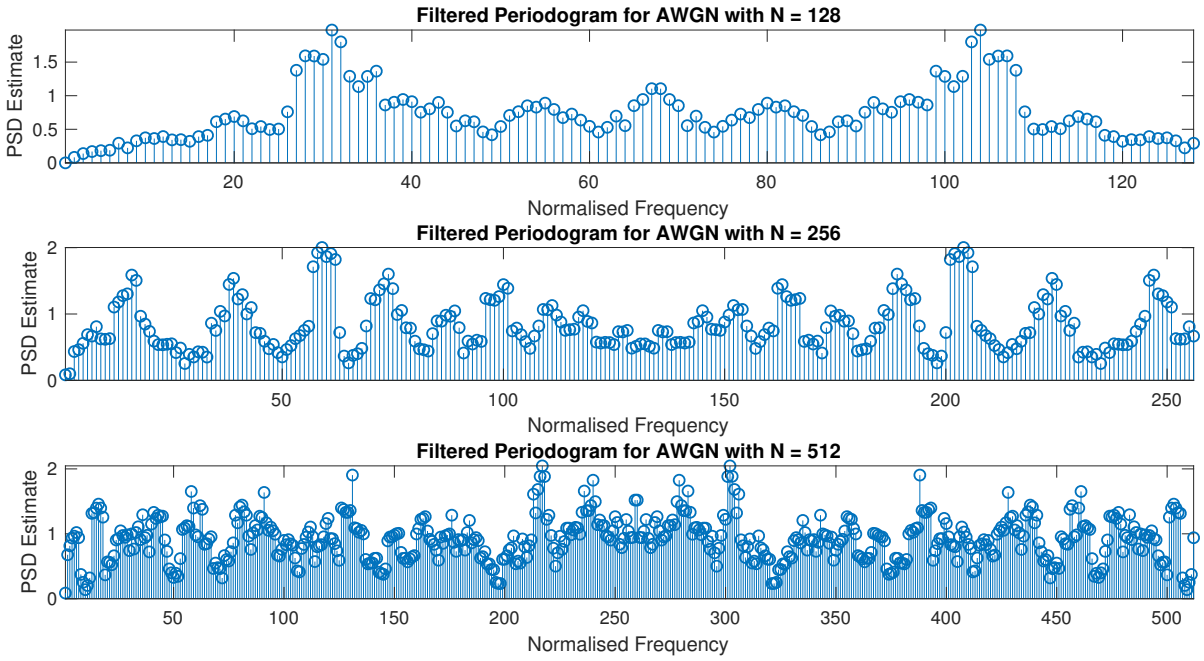

Figure 3.2: Filtered PSD Estimate for AWGN of N=128, 256, 512

3.1.2 PSD of subdivided AWGN realisation

In this section we are required to take a 1024-sample AWGN and subdivided in 8 128-sample non-overlapping realisations. Since the samples are i.i.d. this is equivalent of individually generating 8 realisations of 128 samples.

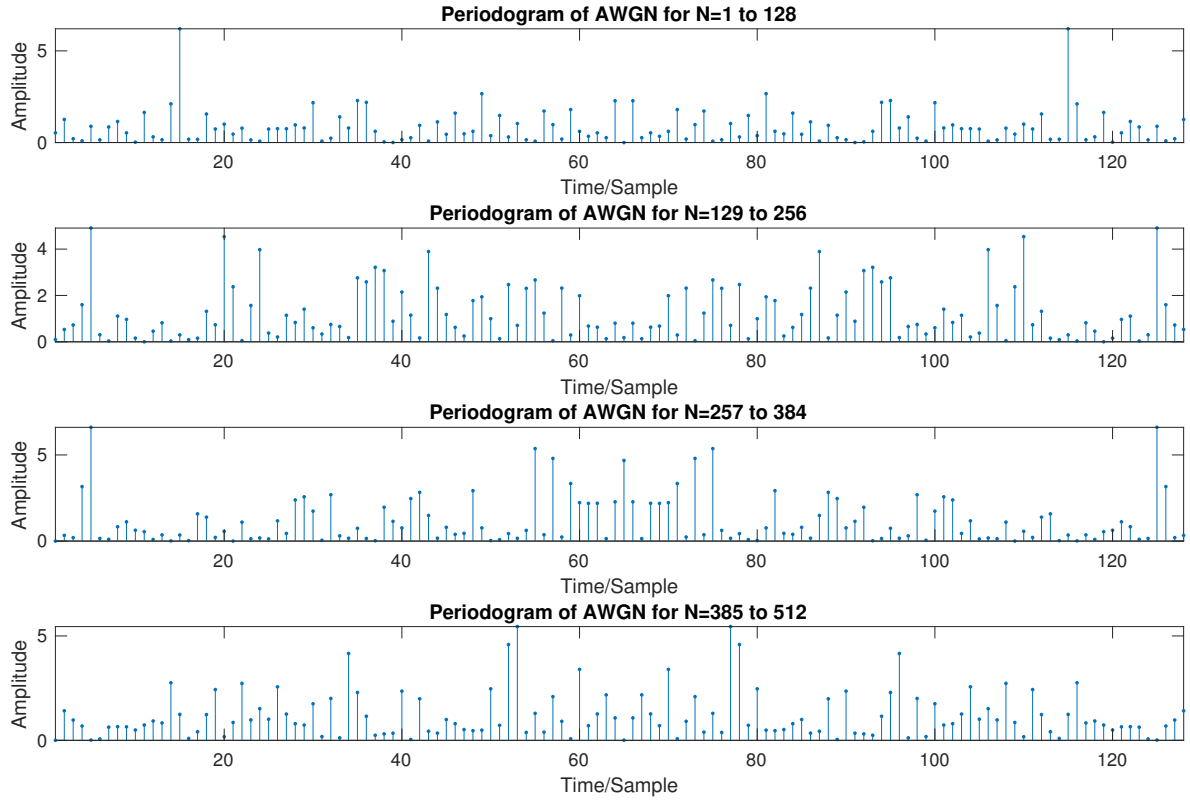

Figure 3.3: Filtered PSD Estimate for AWGN of N=128, 256, 512

The first 4 partitions are shown in Figure 3.3 where it is immediately visible that the plots have a great resemblance with the N=128 realisation in Figure 3.1. Also in Figure 3.3 we have not applied the averaging FIR filter as in Figure 3.2 which explains the large variance between samples.

3.1.3 Averaging of 8 partitions

Now we average the 8 128-sample realisations generated in the above section using function `mean` in order to create an averaged periodogram of 128 samples.

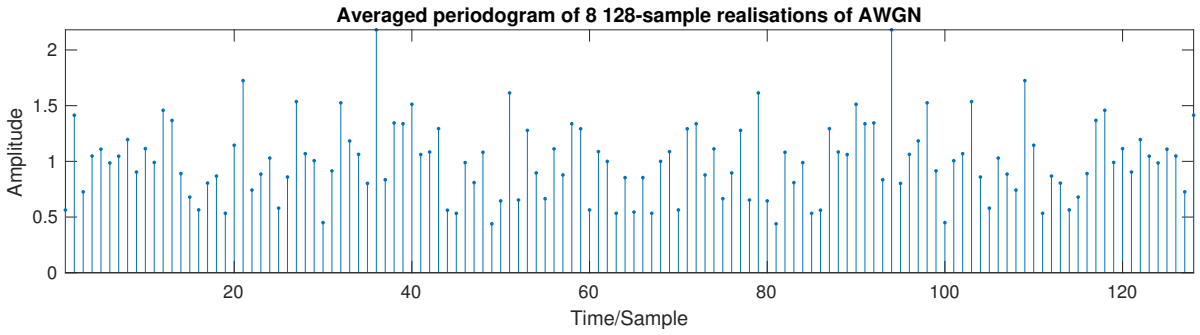

Figure 3.4: Averaged periodogram of 8 128-sample realisations of AWGN

The `stem()` plot of the averaged periodogram is presented in Figure 3.4. There is a clear improvement when contrasting it with the separate periodograms which indicates that the averaged is a vastly better approximation of the ideal PSD.

3.2 Spectrum of autoregressive processes

A 1064-sample WGN sequence was filtered using the MATLAB function `filter` with $b = [1]$ and $a = [1, 0.9]$, an AR(1) model, and then the first 40 samples of the new filtered signal y were removed as they are affected by the transient effects of the filter. Plots for both the unfiltered and filtered signal are shown in Figure 3.5.

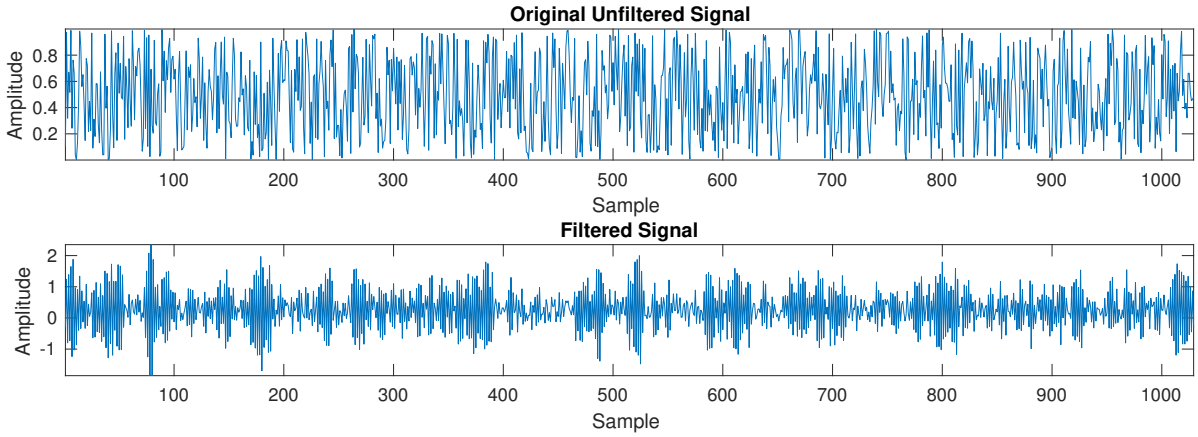

Figure 3.5: Filtered and unfiltered 1064-sample WGN

It is noticeable when we compare the plot that the AR(1) model acts as a high-pass filter, since the filtered plot oscillates at higher frequencies. We can derive the Variance for the AR(1) model in order to better explain:

$$\begin{gathered} Var(y[n]) = Var(a_1^2 y[n-1] + x[n]) = a_1 Var(y[n-1]) + Var(x[n]) \\ \text{Since } Var(y[n]) = \sigma_y^2 \text{ and } Var(x[n]) = \sigma_x^2 \\ \text{Giving the results: } \sigma_y^2 = \frac{\sigma_x^2}{1 - a_1^2} \\ \text{Which with our values is : } \sigma_y^2 = 5.26 \end{gathered} \tag{16}$$

3.2.1 PSD of AR(1)generated process

We plot the results of the process generated by an AR(1) model using the Matlab function `plot(w/(2*pi),abs(h).^2)`, which plot the PSD of y.The results are shown in Figure 3.6.

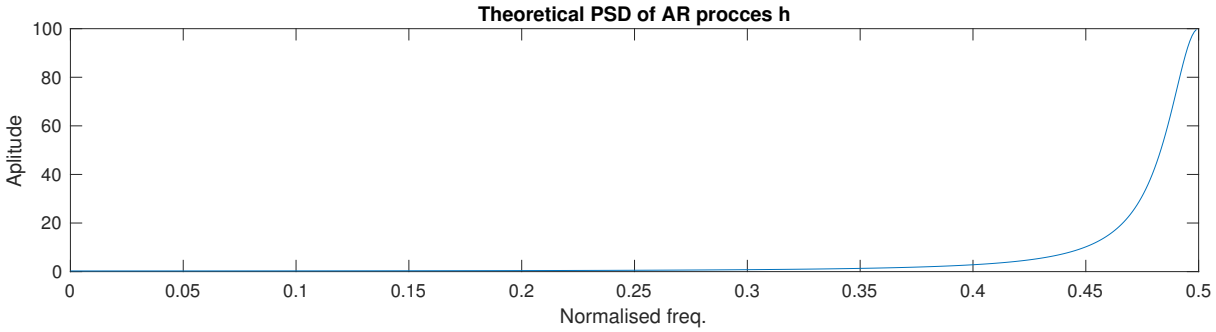

Figure 3.6: Theoretical PSD of y

This just shows again that AR(1) acts as a high-pass filter, as we can see that the higher frequencies have the most amount of power.

3.2.2 PSD and periodogram of AR(1) generated process

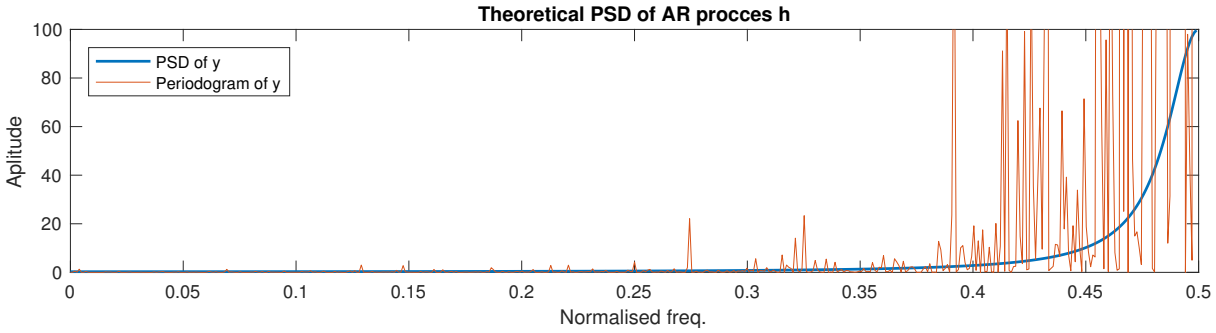

Figure 3.7: Theoretical PSD and periodogram of y

In this section we use the `pgm()` function written in the beginning of this Section on our y signal for values $[1:512]$ and we plot in the same graph as the PSD of y, shown in Figure 3.7. The periodogram seems to agree with the theoretical PSD estimation however some differences are also visible. These differences can be explained by the fact that the periodogram uses a finite number of samples whereas the theoretical PSD uses an infinite number of samples.

3.2.3 Rectangular windowing

Figure 3.8 shows the zoomed in version of Figure 3.7 for values between 0.4 and 0.5. In this plot it is more visible that the filter is not ideal because of the method the periodogram uses to generate the output.

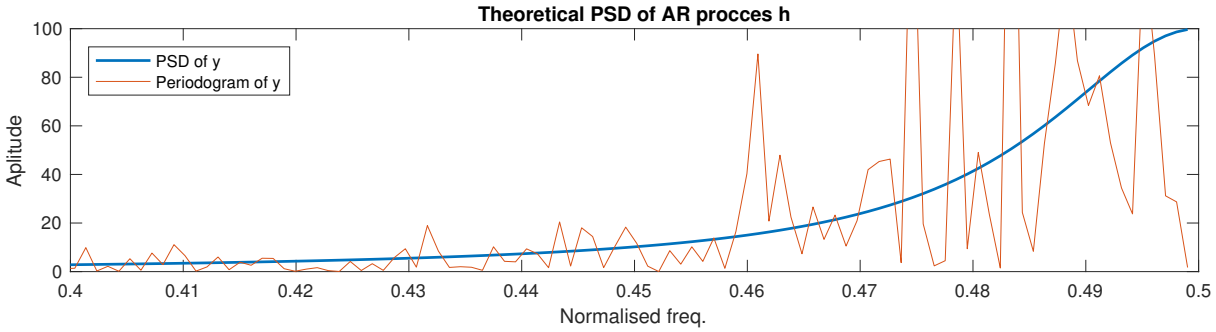

Figure 3.8: Theoretical PSD and periodogram of y

Firstly it applies a rectangular window in the time domain which means in the frequency domain it is convoluting with the `sinc()` function. This has an effect on the peak, whereby it spreads it around the theoretical value of 0.5, which is what we observe in Figure 3.8 aswell.

3.2.4 Model based PSD estimate

After observing the limitations of the periodogram method in estimating the PSD we are going to experiment with the model-based method. This method uses the Yule-Walker equations to express in order to obtain σ_x^2 and a_1 which we can then use to estimate the PSD.

$$\begin{aligned} \hat{R}_{xx}(0) &= a_1 \hat{R}_{xx}(1) + \sigma_x^2 \\ \hat{R}_{xx}(1) &= -a_1 \hat{R}_{xx}(0) \end{aligned} \tag{17}$$

If we rearrange equation (17) we can get the values of interest:

$$\begin{aligned} \hat{a_1} &= \frac{-\hat{R}_{xx}(1)}{hat R_{xx}(0)} \\ \sigma_x^2 &= a_1 \hat{R}_{xx}(1) - \hat{R}_{xx}(0) \end{aligned} \tag{18}$$

We use the `xcorr` function to calculate the parameters $\hat{a_1}$ and σ_x^2. We then use the `freqz` function to estimate the model-based PSD. Both are then plotted in Figure 3.9.

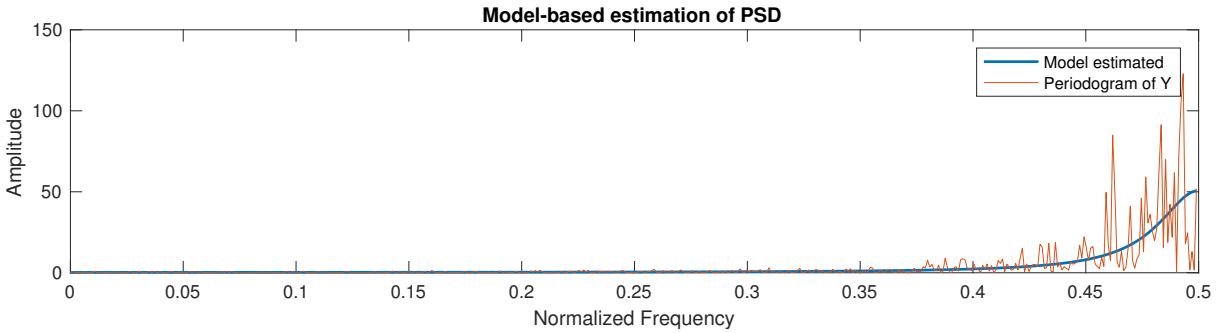

Figure 3.9: Model based PSD estimation

We can see that the model based PSD has the shape of the ideal PSD however they differ in amplitude. The amplitude is sensitive to both factors $\hat{a_1}$ and σ_x^2. $\hat{a_1}$ is calculated using the ratio of $\hat{R}_{xx}(1)$ and $\hat{R}_{xx}(0)$ thus it is sensitive to the imprecision of this ratio whereas σ_x^2 is calculated using $\hat{R}_{xx}(1)$, $\hat{R}_{xx}(0)$ and $\hat{a_1}$ so it just increases the inaccuracy of the model.

3.2.5 Sunspot data PSD

We now to try estimate the model-based PSD and the periodogram of the Sunspot data. We calculate the model-based for orders $[1, 2, 10]$ for both the original data and the zero-mean unit variance sunspots. The results were plotted in Figure 3.10.

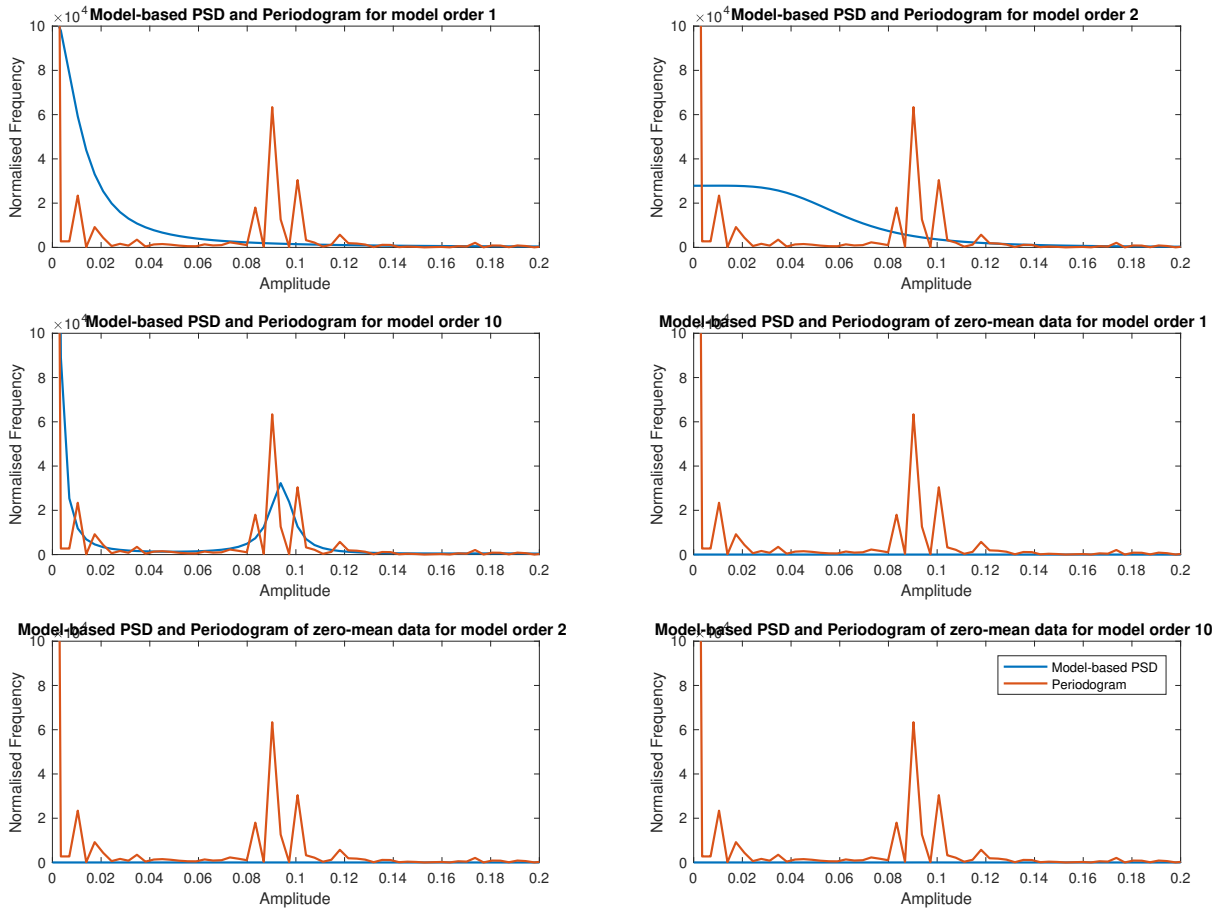

Figure 3.10: Model based PSD vs Periodogram for sunspot data

For model orders 1 and 2 the periodogram does is not similar to the model-based PSD. This is due to information loss because of the low order. For model order 10 with the original data the periodogram and model-based prediction agree. For the normalised data however, the model-based seems not to agree with periodogram in all cases.

3.3 Spectrogram for time-frequency analysis: dial tone pad

3.3.1 Dial tone for London Number

In this section a random London landline number consisting of 11 digits was generated using `num = randi([0 9], 1, 8);` with the first 3 digits being 020(London landline code).

	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

Figure 3.11: Table of dial-pad frequencies

The sequence then was used to generate a Dual Tone Multi-Frequency (DTMF) signal, according to the table given in Figure 3.11 and the equation $y[n] = sin(2\pi f_1 n) + sin(2\pi f_2 n)$. We assume each key is pressed for 0.25s followed by a pause of 0.25 seconds, giving the full number a time of 5.25s. Furthermore we are given the sampling frequency, f_s to be 32768Hz. Therefore the signal would consist of $5.25s \times 32768 = 172032$ samples, where each key corresponds to 8192 samples. The signal for the last two digits is shown in Figure 3.12. The Figure shows the signal for the last two digits with a pause as well as the zoomed in version for number 3 and number 1.

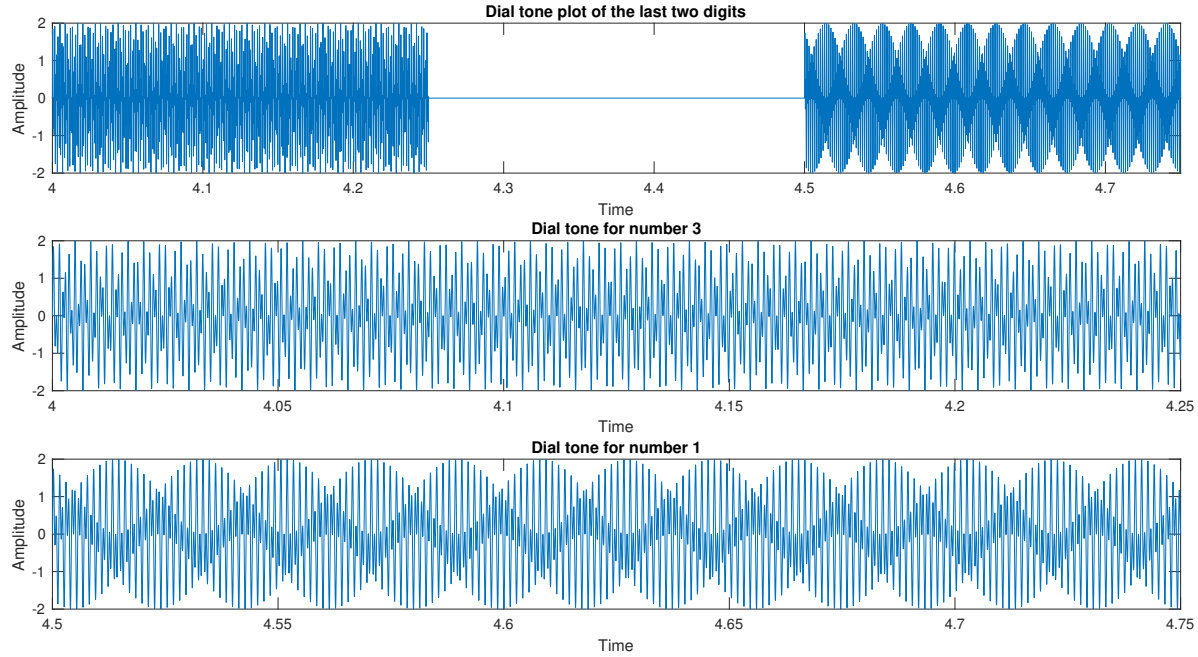

Figure 3.12: Dial-pad signal for last two digits

3.3.2 Spectrogram

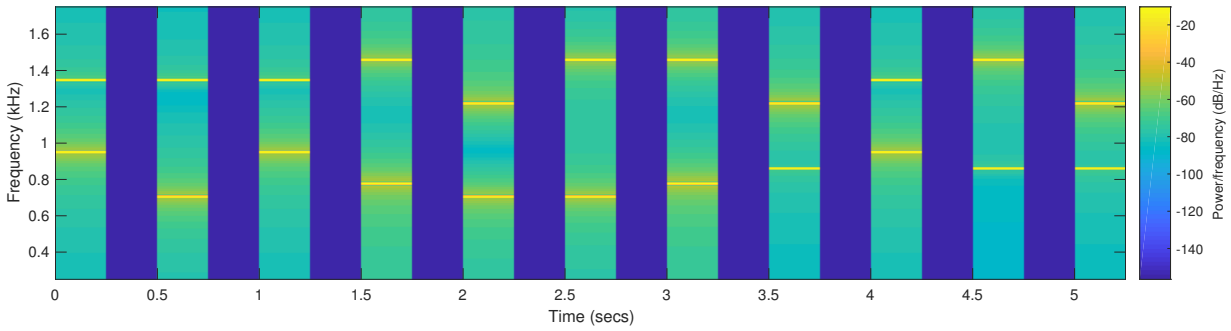

Figure 3.13: Spectrogram of signal y[n]

In Figure 3.13 we can see the spectrogram of signal y[n] generated in the above section. The spectrogram was generated using Matlab function `spectrogram` with carefully chosen parameters such that there is no over-lapping in segments.

3.3.3 Identifying the sequence from the Spectrogram

We can use the spectrogram in Figure 3.13 together with the table of frequencies in Figure 3.11 to decode the signal in order to extract the number sequence that was dialled. The yellow lines in the spectrogram represent a frequency and there are two yellow lines per number showing two frequencies as corresponding to the table of in Figure 3.11. The above spectrogram represents the sequence: 020 613 67097 (decoded from table).

3.3.4 Dial-pad sequence with added noise

In this section we repeat the steps from the above section but with a signal corrupted by noise. We will use 3 different white noise with low medium and high variance. In figure 3.14 the plot of the last 2 digits of the sequence for varying SNR are shown. It is clear that with decreasing SNR the plots become harder and harder to distinguish between separate tones until in the bottom plot it is impossible to do so.

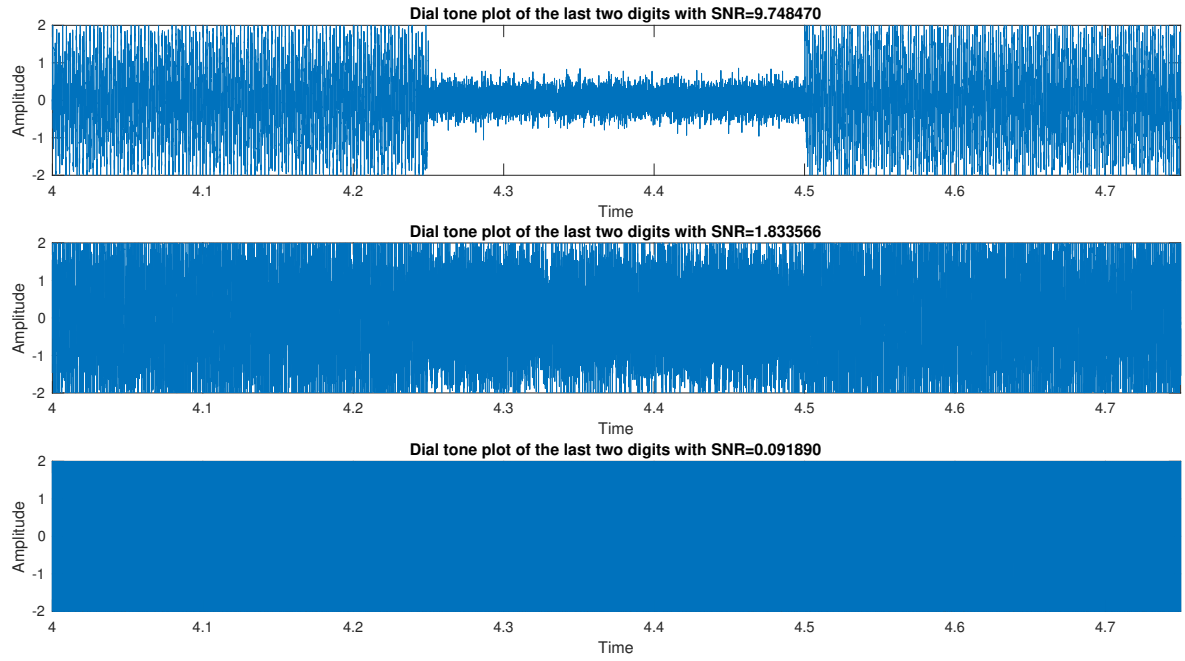

Figure 3.14: Plot of noisy y[n] signal for different SNR

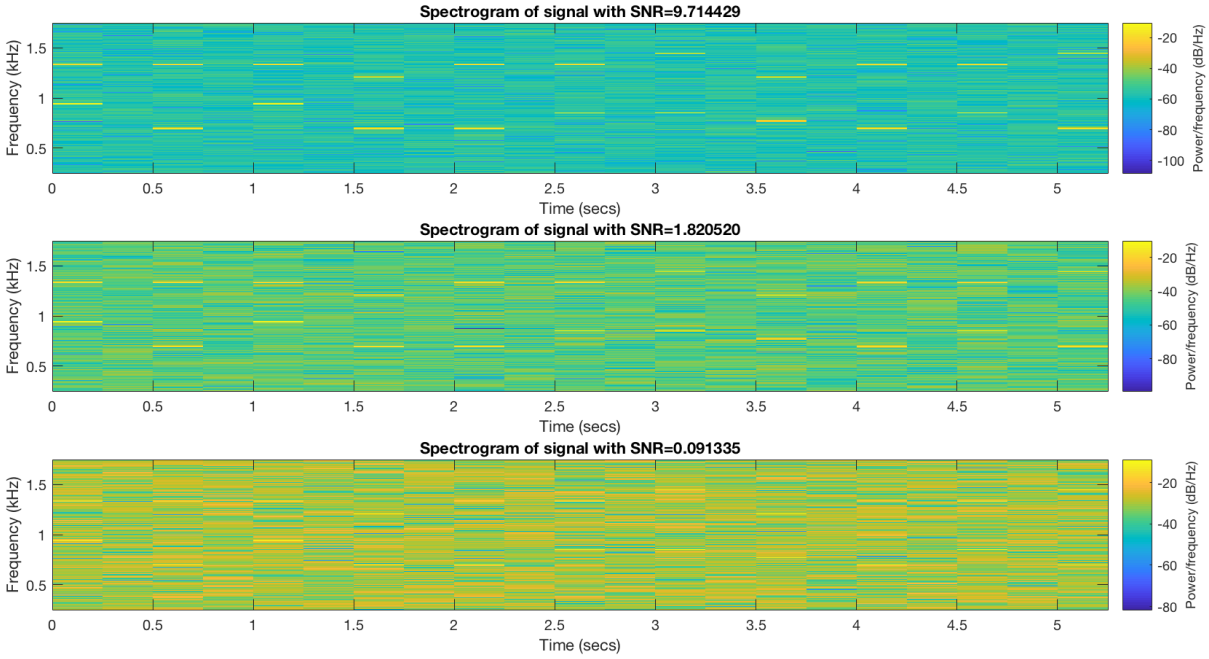

Figure 3.15: Plot of noisy y[n] signal for different SNR

For the spectrogram of the noisy signal, shown in Figure 3.15, we also observe something similar to the plots. With SNR 9.71 the separate tones are still distinguishable

and therefore can be decoded. As the SNR decreases the yellow lines, representing the frequency, become less and less clear until in the bottom spectrogram , SNR = 0.09, the frequencies are corrupted by the noise completely and therefore it would not be possible to decode.

List of Figures