Imperial College London

Advanced Signal Processing Coursework

Prof. Danilo P. Mandic

TA: Sithan Kanna

Alexandre Benoit

CID: 00747144

Email: ab7012@ic.ac.uk

Department of Electrical and Electronic Engineering

Due Date: 9th April 2015

TABLE OF CONTENTS

Exercise 1: Random signals and stochastic processes

1.1.1 Statistical estimation for uniform distributions (rand)

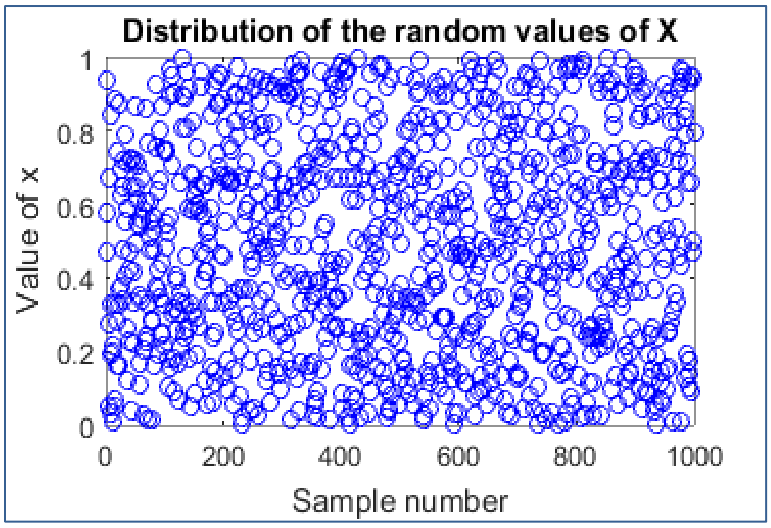


Figure 1: Uniform distribution of the 1000 random variables

i) Theoretical mean vs. sample mean

The theoretical mean or expected value is: $\mathbb{E}(x) = \int_{-\infty}^{+\infty} xp(x)dx = \int_0^1 x\,dx = \left[\frac{x^2}{2}\right]_0^1 = 0.5$

The sample mean was calculated by MATLAB to be: $\hat{m} = 0.49315$
The accuracy of the estimator increases with the sample size, as the sample mean will converge towards the theoretical mean with larger sample sizes. In other words, the estimator will be more accurate for sample sizes of 1000 compared to samples sizes of 10, for example.

ii) Theoretical standard deviation vs. sample standard deviation

The theoretical standard deviation is:

$$\sigma = \sqrt{E\{x - E(x)\}^2} = \sqrt{E\left\{x - \frac{1}{2}\right\}^2} = \sqrt{\int_{-\infty}^{+\infty}\left(x^2 - x + \frac{1}{4}\right)dx} = \sqrt{\left[\frac{x^3}{3} - \frac{x^2}{2} + \frac{x}{4}\right]_0^1} = \sqrt{\frac{1}{12}} \approx 0.2887$$

The sample standard deviation was calculated by MATLAB to be: $\hat{\sigma} = 0.29027$
As for the sample mean, the value of the sample standard deviation will converge towards the theoretical standard deviation, and therefore be more accurate, as the sample size increases.

iii) Sample mean and sample standard deviation estimator bias

Mean of the respective 10 ensembles of 1000 random values between 0 - 1									
0.5003	0.4831	0.4949	0.5057	0.5021	0.5016	0.5084	0.4948	0.5013	0.5159
Standard deviation of the respective 10 ensembles of 1000 random values between 0 - 1									
0.2870	0.2837	0.2960	0.2957	0.2831	0.2875	0.2885	0.2913	0.2941	0.2955

Table 1: The sample means and sample standard deviations for the respective 10 uniform distributed ensembles

From the data calculated using MATLAB, we get an average of the 10 ensemble means of: 0.50081 and an averaged standard deviation of: 0.29024. Knowing that their theoretical values are 0.5 and 0.2887 respectively, we can conclude that both estimators have no bias as they are very clustered and vary only slightly around their respective theoretical values, as seen in Figure 2 (next page).

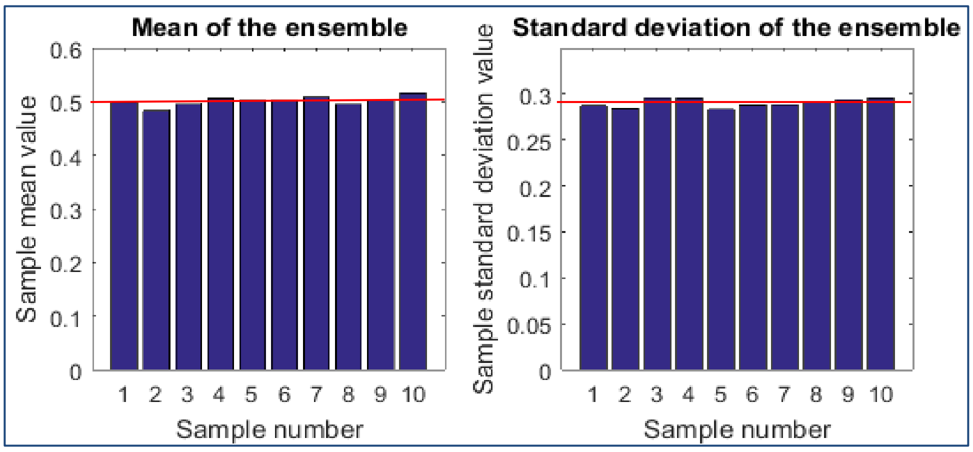


Figure 2: Sample mean and standard deviation for each of the uniform distributed ensembles

iv) Probability density function of the uniform distribution

Figure 3 below represents the different empirical and theoretical pdf's for different ensemble sizes and bin numbers. Subplot 1 and 3 depict that as the number of samples increase, the estimated pdf converges to the theoretical pdf. In addition, looking at subplots 1 and 2, we can say that, in this case of the uniform distribution, having fewer bins increases the smoothness of the estimate, although is less overall precise. This is because a finer grid involves fewer samples to be considered per bin, therefore being more unpredictable (e.g. subplot 2 there are only 10 samples per bin, we should have at least 100+ samples per bin). This is also true for subplot 3 and 4, however less due to the much greater number of samples per bin.

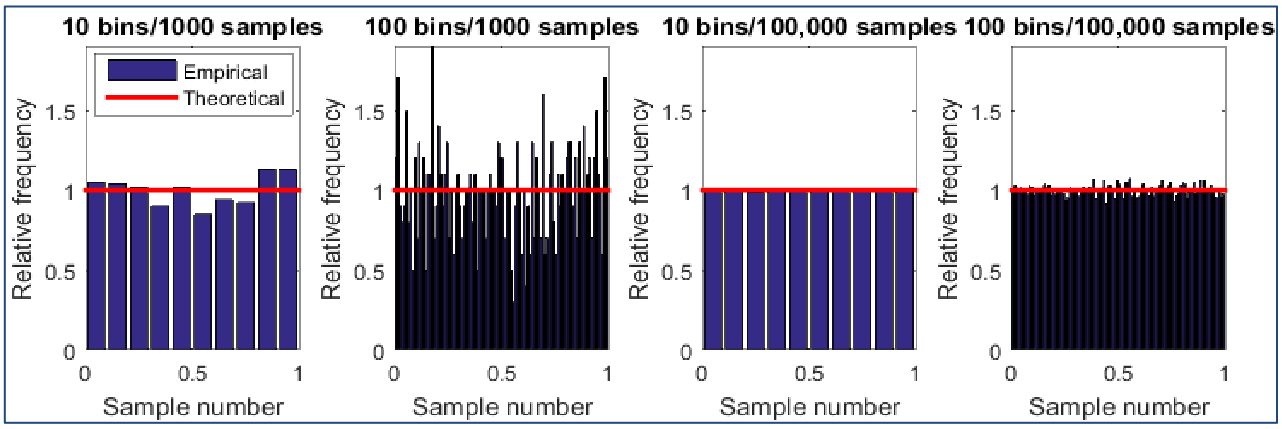


Figure 3: Empirical and theoretical pdf's of uniform distributions for different sample sizes and bin numbers

1.1.1 Statistical estimation for Gaussian/normal distributions (randn)

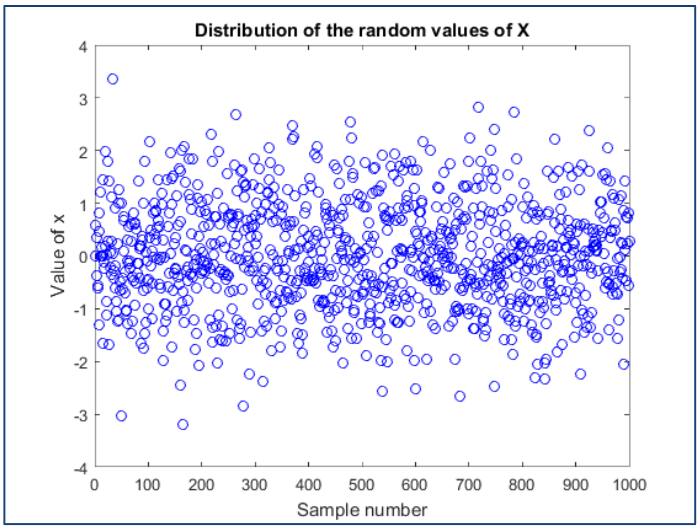


Figure 4: Gaussian distribution of the 1000 random variables

i) Theoretical mean vs. sample mean

By definition, the randn function generates samples normally distributed with 0 mean and a standard derivation of 1. Therefore the theoretical mean or expected value is: $\mathbb{E}(x) = 0$
The sample mean was calculated by MATLAB to be: $\hat{m} = 0.0068913$
Again, the difference between the theoretical and calculated mean will decrease with increasing numbers of samples.

ii) Theoretical standard deviation vs. sample standard deviation

The theoretical standard deviation, due to the nature of the randn function, is: $\sigma = 1$
The sample standard deviation was calculated by MATLAB to be: $\hat{\sigma} = 0.99768$
An increase in the number of samples would, once again, would increase the estimator's accuracy.

iii) Sample mean and sample standard deviation estimator bias

Mean of the respective 10 ensembles of 1000 Gaussian random variables									
0.0369	0.0383	0.0273	0.0306	-0.0590	0.0038	0.0041	-0.0681	0.0354	0.0071
Standard deviation of the respective 10 ensembles of 1000 Gaussian random variables									
0.9981	1.0050	0.9949	0.9758	1.0084	0.9830	0.9795	1.0014	0.9613	0.9524

Table 2: The sample means and sample standard deviations for the respective 10 Gaussian distributed ensembles

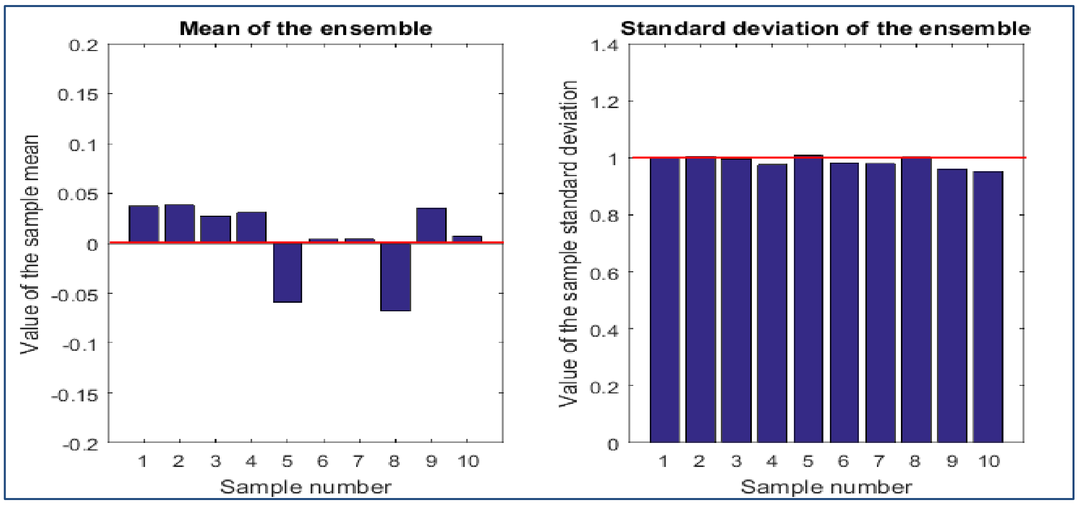


Figure 5: Sample mean and standard deviation for each of the Gaussian distributed ensembles

From the data calculated using MATLAB, we get an average of the 10 ensemble means of: 0.00564 and an averaged standard deviation of: 0.98598. As in the previous case in 1.1.3, we can see from the graphs that the sample means and sample normal distributions cluster very closely around their respective theoretical values, therefore the estimators have no bias.

iv) Probability density function of the uniform distribution

We can see on Figure 6, subplot 1 and 3, (next page) that the more samples available, the more the empirical pdf converges towards the theoretical normal distribution. Regarding the number of bins affecting the analysis, more bins mean more precision, but we need at least 50+ samples per bin so that graphs are not misleading and unpredictable. In subplot 2, we only have 10 samples per bin leading to potentially confusing results due to the random nature of the sample generation. However, subplot 4 having 1000 samples per bin is much more precise that the subplot 3 that is wasting a lot of information by just having 10 bins.

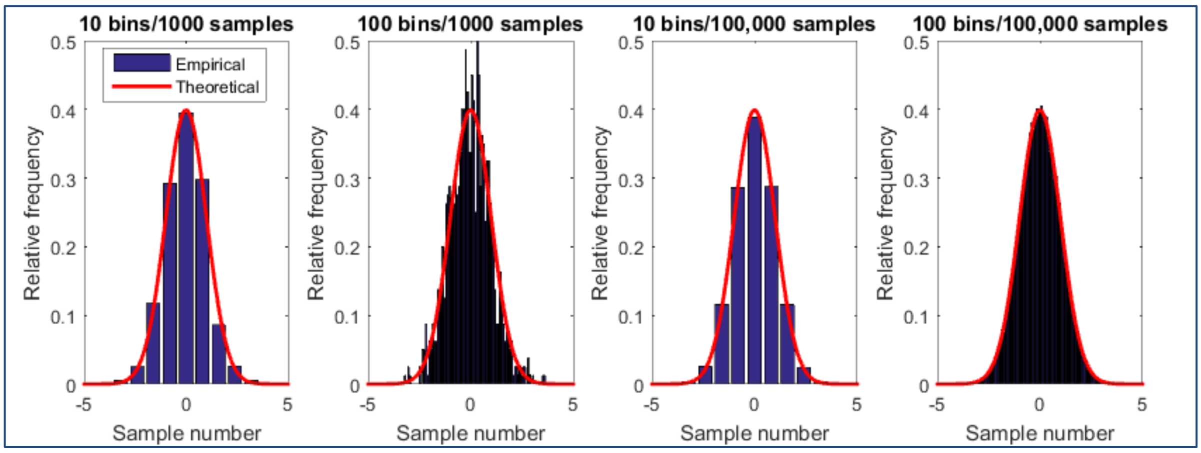


Figure 6: Empirical and theoretical pdf's of Gaussian distributions for different sample sizes and bin numbers

1.2 Stochastic processes

In this section, we will be analysing three different random processes: rp1, rp2 and rp3. We will be calculating using MATLAB the mean and standard deviation for the three signals as a function of time and for different realisations. The signals will be characterised as stationary if its characteristics calculated will not vary significantly over a sufficiently long time. They will be characterised as ergodic if this is the case for multiple different instances or realisations of the given signal. We will finally compute these values theoretically and compare them with the ones obtained by sample averaging. We will be analysing each signal one after the other for comprehension and easy comparison.

i) Random process 1: RP1

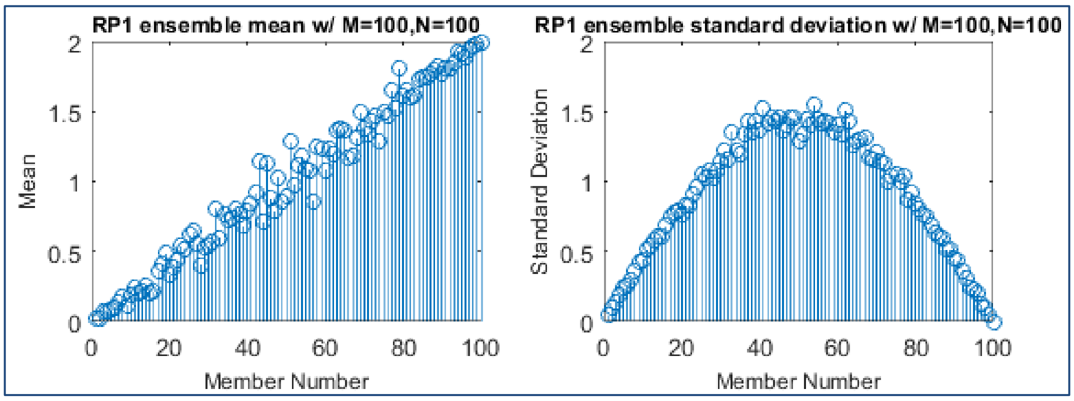


Figure 8: Ensemble mean and standard deviation for RP1

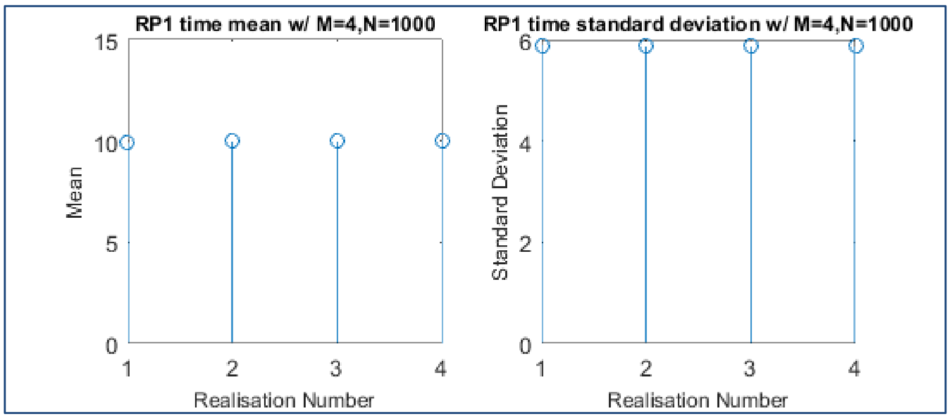


Figure 7: Time means and standard deviations for RP1

Non-Stationary: From Figure 7, we can clearly see that the ensemble mean and ensemble standard deviation vary with time. The ensemble mean varies linearly in time with the number of samples and the ensemble standard deviation varies in a similar fashion to the standard deviation of a sinusoidal signal.
Non-Ergodic: From Figure 8, we can see that the time mean and time standard deviation do not vary for different realisations of RP1. However, this process cannot be ergodic as we have previously concluded that it was not stationary.

Mathematical Description of Process:
From the code, we can derive a mathematical description of the process RP1:
$rp1(n) = [u(n) - 0.5] \times 5\sin\left(\frac{n\pi}{N}\right) + 0.02n$
Theoretical Mean:
$\mathbb{E}\{rp1(n)\} = \mathbb{E}\left\{[u(n) - 0.5] \times 5\sin\left(\frac{n\pi}{N}\right) + 0.02n\right\} = 0.02n$ *because E{u(n) – 0.5} = 0*
This expression of the theoretical mean confirms Figure 7, subplot 1's ensemble mean linearity with number of samples n.
Theoretical Variance:
$\sigma^2 = Var\{rp1(n)\} = \mathbb{E}\{(X-\mu)^2\} = \mathbb{E}\left\{\left[[u(n) - 0.5] \times 5\sin\left(\frac{n\pi}{N}\right) + 0.02n - 0.02n\right]^2\right\}$
$= 25\sin^2\left(\frac{n\pi}{N}\right) \times \mathbb{E}\{[u(n) - 0.5]^2\} = \frac{25}{12}\sin^2\left(\frac{n\pi}{N}\right)$
With N=100, we get a theoretical variance of: $\sigma^2 = \frac{25}{12}\sin^2\left(\frac{n\pi}{100}\right)$ and therefore a theoretical standard deviation of: $\sigma = \frac{5}{\sqrt{12}}\sin\left(\frac{n\pi}{100}\right)$
This result is consistent with our graph and its estimated deviation similar to the deviation of a sinusoidal wave. In addition, we can see that it peaks around 1.44, which is the approximate value of $\frac{5}{\sqrt{12}}$.

ii) Random process 2: RP2

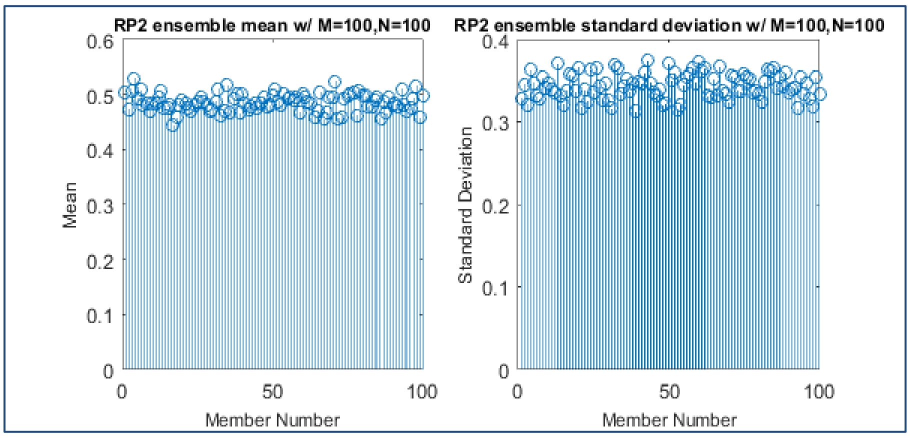


Figure 9: Ensemble mean and standard deviation for RP2

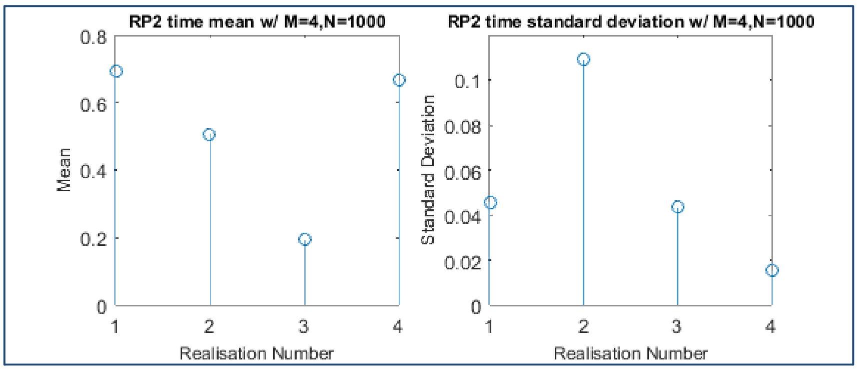


Figure 10: Time means and standard deviations for RP2

Stationary: From Figure 9, we can say that the ensemble mean and standard deviation vary only slightly around a constant value. This depicts the time-invariances of both estimates and that rp2 is stationary.
Non-Ergodic: From Figure 10, the different realisations do not have the same mean and standard deviation over time, therefore, rp2 is not ergodic.

Mathematical Description of Process:
From the code, we can derive a mathematical description of the process RP2:
$rp2(n) = [u(n) - 0.5] \times U(j) + U(k)$ *with U(j) and U(k) chosen uniformly in (0,1)*
Theoretical Mean:
$\mathbb{E}\{rp2(n)\} = \mathbb{E}\{[u(n) - 0.5] \times U(j) + U(k)\} = \mathbb{E}\{U(k)\} = 0.5$ *because E{u(n) – 0.5} = 0*
The calculated theoretical value of the mean corresponds to Figure 9, subplot 1 as the empirical mean varies around the same value.
Theoretical Variance:

$$\sigma^2 = Var\{rp2(n)\} = \mathbb{E}\{X^2\} - (\mathbb{E}\{X\})^2 = \mathbb{E}\{[(u(n) - 0.5) \times U(j) + U(k)]^2\} - 0.5^2$$
$$= \mathbb{E}\{[(u(n) - 0.5) \times U(j)]^2\} + 0.25 = \frac{1}{9}$$

We get a theoretical variance of: $\sigma^2 = \frac{1}{9}$ and therefore a theoretical standard deviation of: $\sigma = \frac{1}{3}$
Looking at Figure 9, subplot 2, we see that the value of the standard deviation varies around 0.33, which is confirmed by the calculation above.

iii) Random process 3: RP3

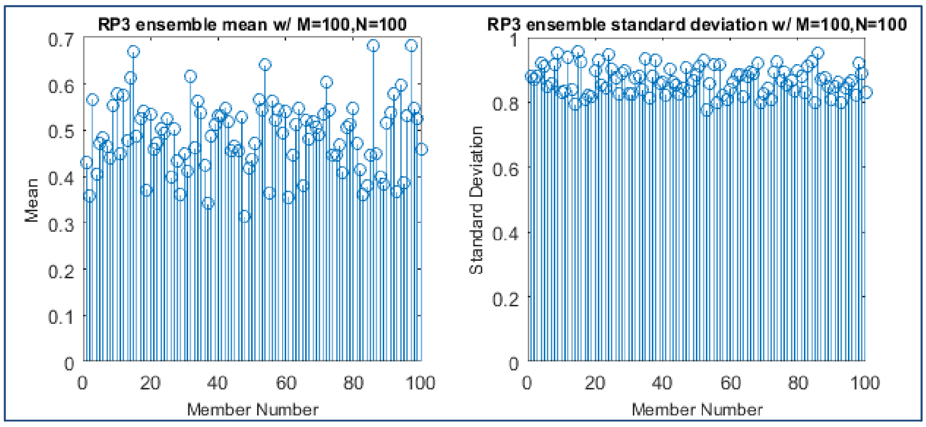


Figure 11: Ensemble mean and standard deviation for RP3

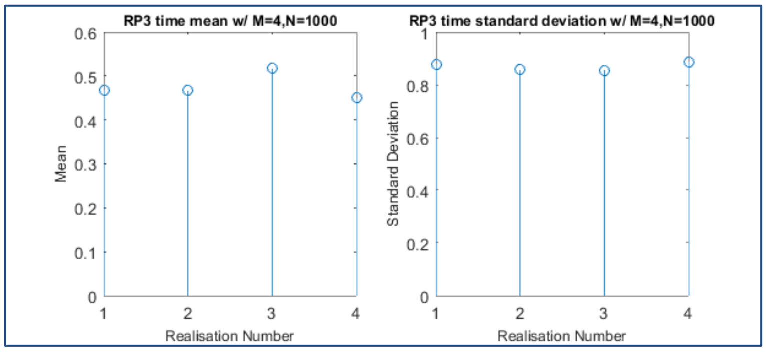


Figure 12: Time means and standard deviations for RP3

Stationary: From Figure 11, we observe that both the ensemble mean and standard deviation vary negligibly around a constant value. We can say they are time-invariant and that process rp3 is stationary.
Ergodic: From Figure 12, the mean and standard deviation of the different realisations of process rp3 seem to be constant around a fixed value over time. Therefore, the process rp3 is both stationary and ergodic.

Mathematical Description of Process:

From the code, we can derive a mathematical description of the process RP3:

$rp3(n) = 3[u(n) - 0.5] + 0.5$

Theoretical Mean:

$\mathbb{E}\{rp2(n)\} = \mathbb{E}\{3[u(n) - 0.5] + 0.5\} = \mathbb{E}\{3[u(n) - 0.5]\} + \mathbb{E}\{0.5\} = 0.5$ *because E{u(n) – 0.5} = 0*

This value of the theoretical mean confirms what we can see in Figure 11, subplot 1 as the empirical mean clusters around 0.5.

Theoretical Variance:

$$\sigma^2 = Var\{rp3(n)\} = \mathbb{E}\{(X-\mu)^2\} = \mathbb{E}\{[3(u(n) - 0.5) + 0.5]^2\} = 3^2 \times \mathbb{E}\{[u(n) - 0.5]^2\} = \frac{9}{12} = \frac{3}{4}$$

We get a theoretical variance of: $\sigma^2 = \frac{3}{4}$ and therefore a theoretical standard deviation of: $\sigma = \sqrt{\frac{3}{4}} \approx 0.86$

This theoretical value for the standard deviation corresponds to what we see in Figure 11, subplot 2, where the ensemble standard deviation varies around 0.86.

iv) Conclusion

All the theoretical values for the means and standard deviations correspond with the ones obtained by sample averaging. In addition, we have concluded that process 1 is non-stationary and non-ergodic, process 2 is stationary but non-ergodic and process 3 is stationary and ergodic.

1.3 Estimation of probability distributions

In this section, we will be using a designed pdf estimator to comparing the pdf's of processes with their respective theoretical pdf's. The code used for this pdf estimator can be seen below in Figure 14.

i) Pdf estimate for a stationary collection of 10,000 samples with Gaussian distribution

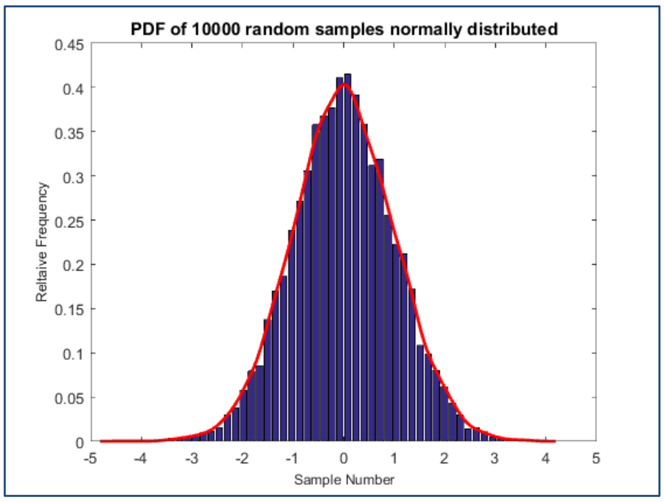
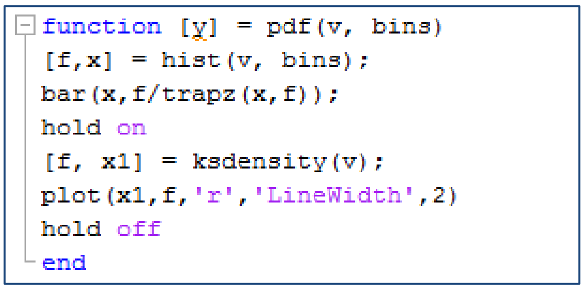


Figure 13: estimated pdf for the 10,000 sample Gaussian process



```
function [y] = pdf(v, bins)
 [f,x] = hist(v, bins);
 bar(x,f/trapz(x,f));
 hold on
 [f, x1] = ksdensity(v);
 plot(x1,f,'r','LineWidth',2)
 hold off
end
```

Figure 14: MATLAB function used to estimate the pdf's

Code explanation and analysis:

We have normalised the pdf in Figure 13 to have an area of one under the estimated pdf curve by dividing each frequency by the area beneath the actual curve (not shown)using the MATLAB function trapz(). The required number of histogram bins is passed as an argument sin the pdf estimator function. The function will plot the estimated pdf and smoothed-out version (red) on the figure currently being worked on when the function is called. As we can see from Figure 13, our pdf estimator generates the correct estimate of our stationary Gaussian process, which confirms that the function works for stationary processes.

ii) Pdf estimate for stationary and ergodic processes (e.g. RP3)

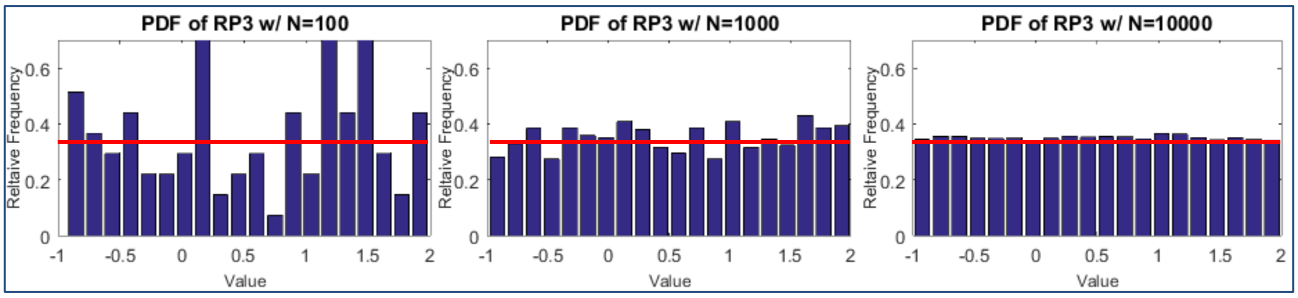


Figure 15: Pdf estimates for stationary process RP3 for different sample sizes with the theoretical pdf in red

We will be using rp3 for this question as it was the only process from the previous part to be stationary and ergodic. As we increase the number of samples for the stationary process, our estimated pdf converges towards the theoretical pdf in red, calculated using the formula: $pdf = \frac{1}{b-a} = \frac{1}{2-(-1)} = \frac{1}{3}$.

iii) Pdf estimate for non-stationary processes (e.g. RP1)

For this question we will use rp1, as it has been proven to be non-stationary and non-ergodic. First of all, looking at the theory, it would not be possible to estimate the pdf of a non-stationary process using our pdf estimator function due to the fact the MATLAB 'hist' function is time-averaged. In other words, for increasing number of samples, the pdf will expand will decreasing the height of its histogram bins, as can be seen in Figure 16 with the decreasing relative frequency of the bins as the number of samples increase.

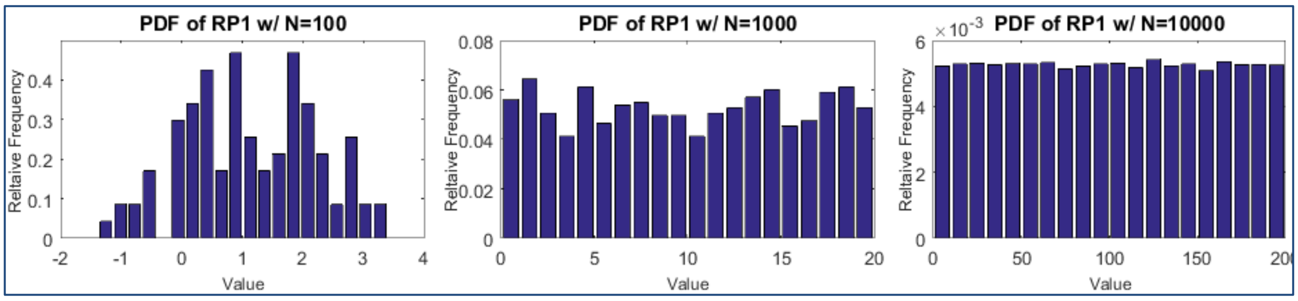


Figure 16: Pdf estimates for non-stationary process RP1 for different sample sizes

For the case where the mean of a 1000 sample long signal changes from 0 to 1 at sample 500, we would need to estimate the pdf for the first half of the signal then again for the second half. Assuming the stable standard deviation, this would separate the signal into two stationary signals.

Exercise 2: Linear stochastic modelling

2.1 ACF of uncorrelated sequences

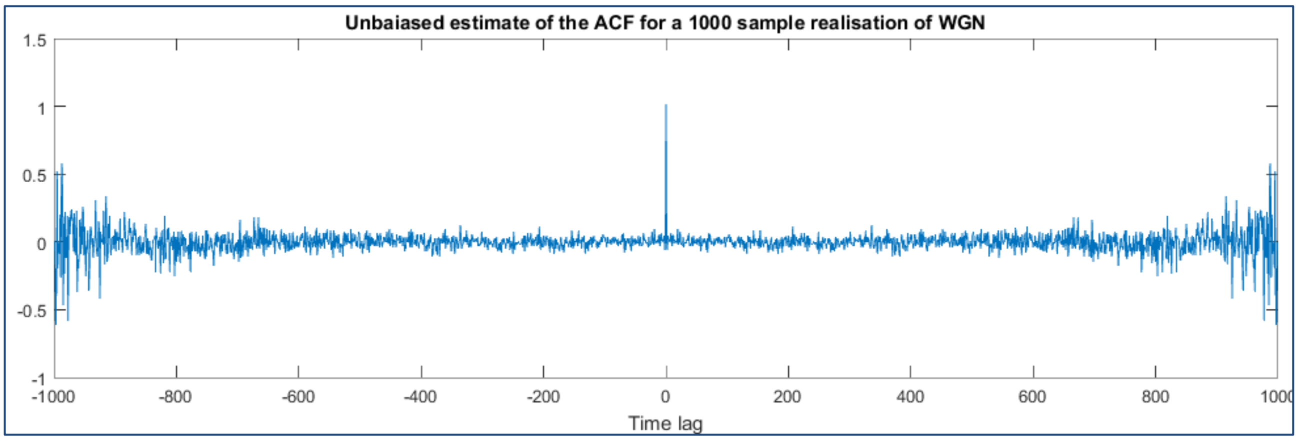


Figure 17: Unbiased estimate of the autocorrelation function for a 1000 sample realisation of white Gaussian noise

Question 1: The MATLAB function xcorr() is used to estimate the unbiased ACF for the generated white Gaussian noise. The ACF is used to determine the correlation of a signal with a time-shifted version of itself. As we can see in Figure 17, the ACF is equal to 1 when the lag is 0 due to the fact that for 0 lag, the waveform is essentially overlapping a copy of itself.
For any other non-zero time lag, the ACF fluctuates around 0 because of the random nature of white Gaussian noise. It does not equal 0 due to the inaccuracy of the MATLAB estimator used to calculate the ACF. We will explore why it has increasing errors for greater time lag in a subsequent question.
Finally, the signal is symmetric with respect to the y axis due to fact ACF function is even, being a convolution between two identical signals.

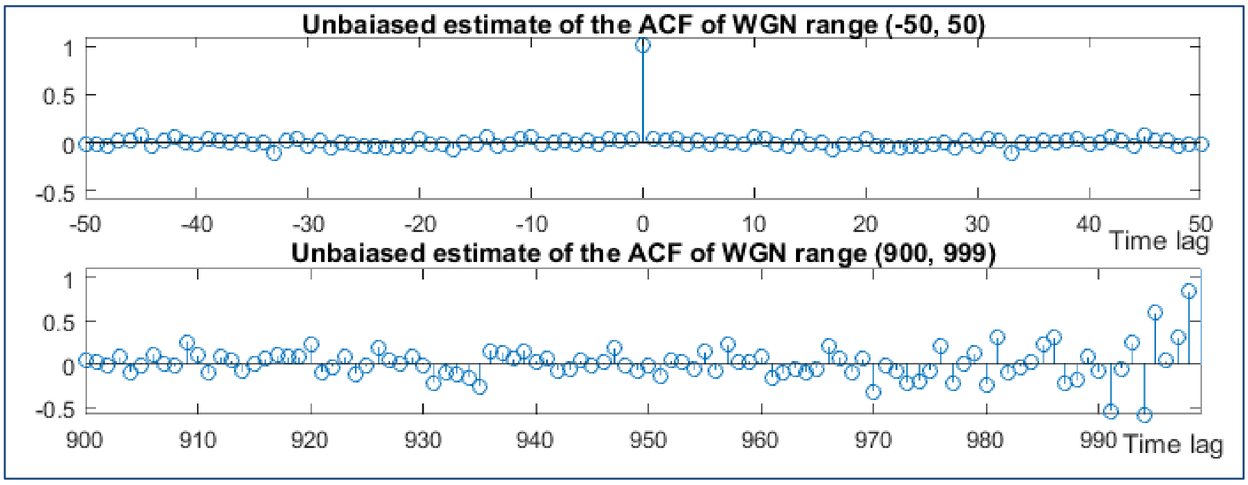


Figure 18: Unbiased estimate of the ACF for the white Gaussian noise for different ranges of time lag (-50,50) and (900,999)

Question 2: As we can see in Figure 18, the shape of the ACF for small time lag fluctuates very closely to 0 (in the range of ±0.2) whereas for large time lag, its shape has a more erratic behaviour (in the range of up to nearly ±0.9).

Question 3: The phenomenon above is due to MATLAB using an ACF estimator to calculate the autocorrelation between signals that does not consider an infinite number of samples:

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

The first fractional factor (in red) increases the variance of the ACF for larger time lag as the estimator will be using fewer points. For example, when the time lag τ = 999, only one data point is considers to estimate the ACF, for time lag τ = 998, two data points are considered, etc. This inevitably increases the variance of the ACF for larger time lag.

We can set a confidence limit on$|\tau|$, for example $\tau < \pm 2\sqrt{N} \Leftrightarrow |\tau| < 200$ which also corresponds to a variation of the ACF of ±15% from 0, or a y-axis bound of ±0.15.

2.2 ACF of correlated sequences

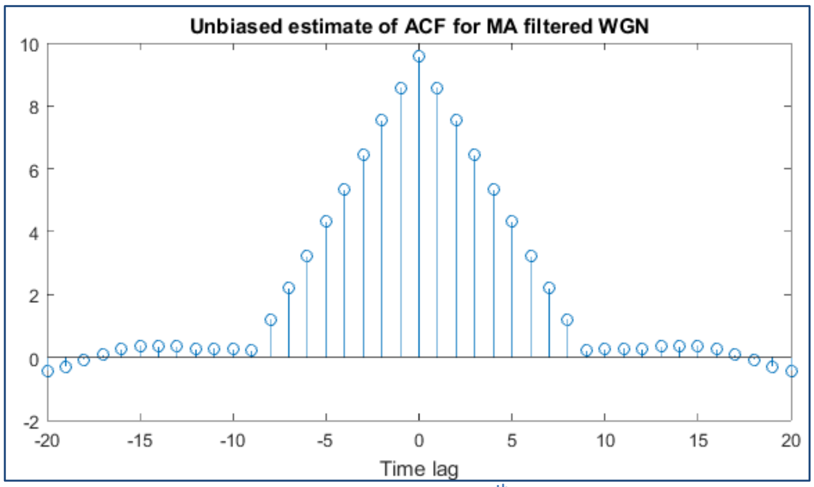


Figure 19: Unbiased estimate of the ACF for the 9th order MA filtered WGN

Question 1: The white Gaussian noise is filtered through a 9th order moving average filter that generates an ACF that can be seen in Figure 19 above. The FIR filter enables a smoother estimation of the ACF due to the attenuation of the high frequency components in the WGN. As before, the ACF is symmetrical with respect to the y axis.

We can see the effects of the MA filter through the introduced autocorrelation at time lag -9 ≤ τ ≤ 9, which also corresponds to the order of the MA filter. Outside of that range, the filtered ACF fluctuates very slightly and smoothly around 0, which is expected for WGN through a moving average filter.

At 0 time lag, the estimated ACF is equal to: $\frac{1}{N}\sum_{n=0}^{N-1}(x[n])^2$, which corresponds to the sum of the mean squared, from which we can derive the sample mean.

Question 2: $R_Y(\tau)$ is the convolution between the autocorrelation of the input and the autocorrelation of the impulse response. If the input X_n is an uncorrelated process, $R_Y(\tau)$ would simply be $R_h(\tau)$ scaled by the magnitude of the delta in $R_X(\tau)$, which is 1. Therefore, $R_Y(\tau)$ is the autocorrelation of the impulse response and $R_Y(\tau) = R_h(\tau)$.

2.3 Cross-correlation function

Question 1: The cross-correlation of two signals measures the similarity of these two signals with respect to a certain time lag. From Figure 20, subplot 1 (next page), we can see that x and y are cross-correlated for -9 < τ ≤ 0. On the other hand, x and y are not cross-correlated outside this range, which is expected because the filter is only of 9th order, and the output only depends on the past 9 input values x.

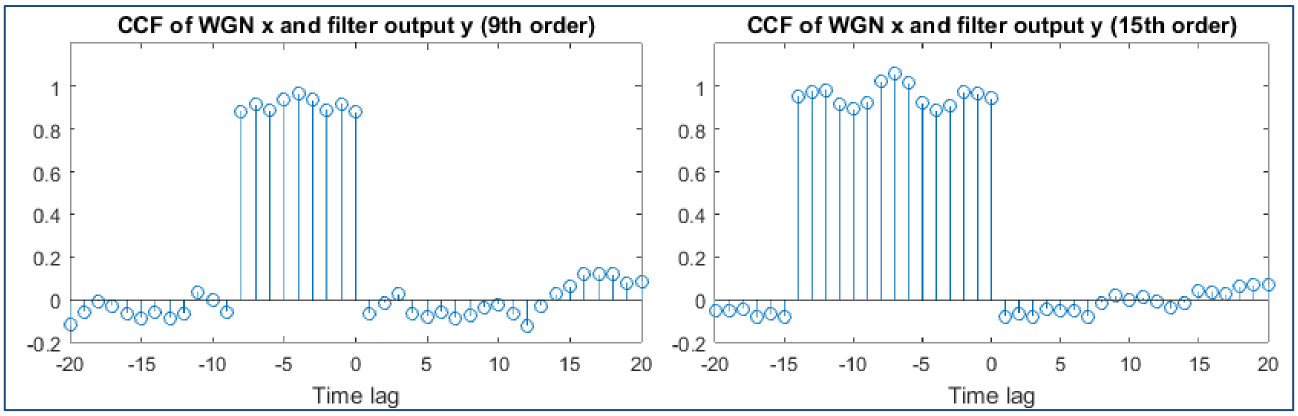


Figure 20: Unbiased estimate CCF of WGN and MA filtered version for different orders of MA filters (9 and 15)

To confirm this, we increase the MA filter order to 15 from 9 to see if we would get the same reaction. We see from Figure 20 above, subplot 2, that x and y are cross-correlated for -15 < τ ≤ 0, and weakly cross-correlated outside this range, which is the same scenario as with the 9th order MA filter, therefore confirming our explanation.
The above plots can be expressed as: $R_{XY}(\tau) = h(\tau) * R_X(\tau)$. If the input X_n is an uncorrelated stochastic process, $R_{XY}(\tau)$ would simply be $h(\tau)$ convoluted with the magnitude of the delta in $R_X(\tau)$, which is 1. Therefore, $R_{XY}(\tau)$ is the impulse response of the filter and $R_{XY}(\tau) = h(\tau)$.

Question 2: The result above can be used for system identification by using as input X_n being an uncorrelated stochastic process, therefore generating the system impulse response by the cross-correlation of the input x and the filtered input y.
This would be more precise for higher orders of the MA filter as the estimate of the CCF would be more accurate.

2.4 Autoregressive modelling

Question 1: The AR(2) process considered can be defined as: $x[n] = a_1x[n-1] + a_2x[n-2] + w[n]$ with $w[n]{\sim}N(0,1)$. The condition for the stability of the AR(2) process depends on the convergence of the signal x[n] converges, which is only obtain for specific ranges of the coefficients a_1 and a_2, which are:

$$\begin{cases} a_2 + a_1 < 1 \\ a_2 - a_1 < 1 \\ |a_2| < 1 \end{cases}$$

This can be shown in the figure below representing the pairs (a_2, a_1) for which x[n] converges in blue and for which x[n] diverges in red.

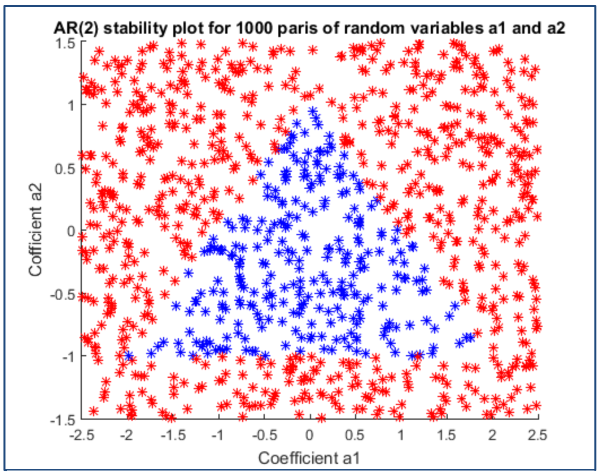


Figure 20.5: Pairs of coefficients for which AR(2) is stable

Question 2:

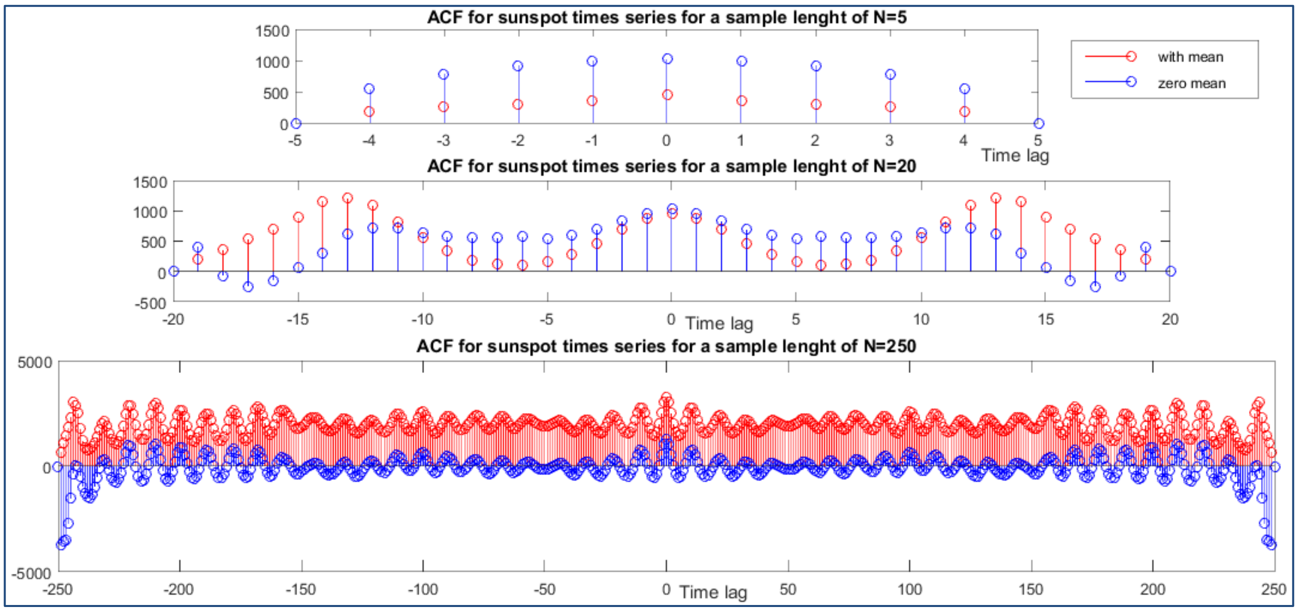


Figure 21: ACF for the sunspot series for different sample lengths (5, 20, and 250)

We can see in the Figure 21 above the different ACF's for the different sunspot series of different lengths with (red) and without (blue) mean. Regarding the overall shapes of the ACF's, as expected, they are all symmetrical with respect to the y axis.

For a small number of samples, i.e. 5, it is hard to evaluate the auto-correlation of the sunspot series. As we can see in Figure 21, subplot 1, even with zero mean, the small sample size makes it hard for us to predict the nature and behaviour of the correlation function. One could potentially say that we see a small auto-correlation for small time lag that decreases for increasing time lag.

For 20 samples, this trend can be seen more clearly, but since the number of samples is still relatively low, the ACF correlates only for small time lag, with and without the mean. In addition, an oscillatory pattern can be distinguished in the ACF's. This periodicity can be explained by, an external circular trend, even though the sunspots appear in random locations. For example, if the samples were taken every month, the peak would be occurring every 12 months, involving the possibility that there is an area of sunspot generation more predictable/less random (higher or lower generation) that is observed every 12 months at the same location in space after the Earth has revolved around the Sun. EDIT: I have discovered that the data is collected yearly, therefore the cycles would be of 11/12 years instead of months.

For a large sample size of 250, we can clearly see the auto-correlation of the sunspot series and its oscillatory tendency, as explained above. This model however breaks down at the extremities of the plot, but this is due to the fact that unbiased estimators use fewer data samples for larger time lag, therefore become more unpredictable.

For all these plots, the mean adds a bias that can be easily observed when the sample size is large. It can be noted that the ACF with mean never goes negative, probably due to the superiority of the large mean that counters any less significant change in the ACF.

Question 3: The Yule-Walker equations are used to calculate the partial correlation functions up until the model order p = 10 so that we can choose the most likely AR model order of the sunspot time series. We can use the aryule() function in MATLBAB to generate the coefficients for both the original sunspot series (Table 3) and standardised sunspot series with zero mean (Table 4).

p	Coefficients										
1	1.0000	-0.9295									
2	1.0000	-1.4740	0.5857								
3	1.0000	-1.5492	0.7750	-0.1284							
4	1.0000	-1.5167	0.5788	0.2638	-0.2532						
5	1.0000	-1.4773	0.5377	0.1739	-0.0174	-0.1555					
6	1.0000	-1.4373	0.5422	0.1291	-0.1558	0.2248	-0.2574				
7	1.0000	-1.3669	0.4807	0.1718	-0.1912	0.0764	0.1359	-0.2736			
8	1.0000	-1.3016	0.4483	0.1535	-0.1456	0.0355	0.0212	0.0523	-0.2384		
9	1.0000	-1.2615	0.4395	0.1500	-0.1515	0.0599	-0.0046	-0.0230	-0.0197	-0.1680	
10	1.0000	-1.2573	0.4400	0.1506	-0.1514	0.0584	-0.0007	-0.0268	-0.0308	-0.1362	-0.0252

Table 4: AR coefficients for different model orders for the original sunspot series

We can see from the table above that the first two partial correlation coefficients are much larger than the rest, making the latter negligible. The AR model order is most likely 2, as it is the last coefficient non-negligible. Nevertheless, we need to apply the same practice on the sunspot series with zero mean to further decay the coefficients.

p	Coefficients										
1	1.0000	-0.8212									
2	1.0000	-1.3783	0.6783								
3	1.0000	-1.2953	0.5097	0.1223							
4	1.0000	-1.3011	0.4856	0.1836	-0.0473						
5	1.0000	-1.3018	0.4885	0.1912	-0.0676	0.0156					
6	1.0000	-1.3044	0.4994	0.1602	-0.1469	0.2269	-0.1623				
7	1.0000	-1.2760	0.4597	0.1859	-0.1749	0.1394	0.0661	-0.1751			
8	1.0000	-1.2361	0.4447	0.1541	-0.1351	0.0971	-0.0385	0.1154	-0.2276		
9	1.0000	-1.1959	0.4243	0.1609	-0.1523	0.1210	-0.0658	0.0368	-0.0093	-0.1766	
10	1.0000	-1.1952	0.4243	0.1608	-0.1520	0.1205	-0.0652	0.0362	-0.0109	-0.1721	-0.0038

Table 3: AR coefficients for different model orders for the standardised sunspot series with zero mean

To standardise the sunspot series, we subtracted the mean from each sample and divided the result by the standard deviation. We then used the same methods as above and used the Yule Walker equations to calculate the AR coefficients. For this standardised sunspot series, we can see that the difference between the first two coefficients and the other ones is more pronounced, mostly due to the zero mean samples. As prior, we can further deduce that the estimated AR model order is 2.

We can conclude this question by using the reflection coefficients to confirm our estimated model order of the sunspot series. Below in Figure 22, we can see that the reflection coefficient becomes very small for model order 3 and converges to zero by model order 10, confirming our AR estimated model order of 2.

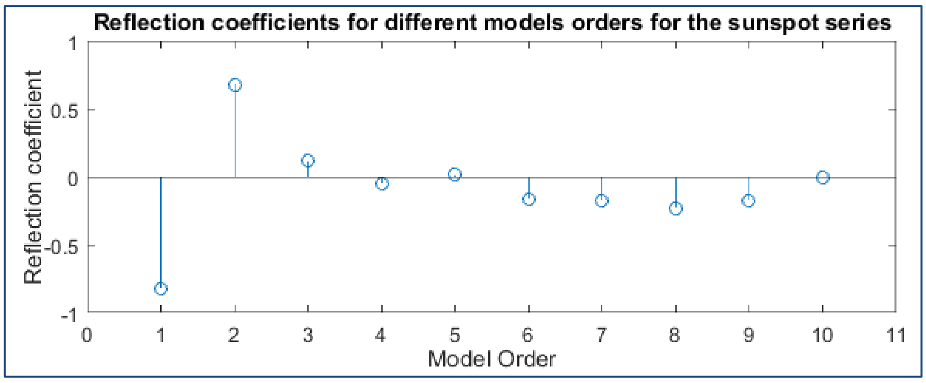


Figure 22: Different reflection coefficients for the standardised sunspot series

Question 4: The minimum description length (MLD) and Akaike Information Criterion (AIC) were used to determine the correct model order for the standardised data. The optimal model order would find a compromise between the accuracy (but complexity) of higher model orders with the simplicity (but inaccuracy) of lesser ones. The model order with lowest MDL or AIC should be selected as it would be the best model order. As we can see in Figure 23, model order 2 or 9 could be chosen, however to limit the complexity of higher model orders and over-sampling, model order 2 is preferred. This is in accord of our conclusions in the previous parts.

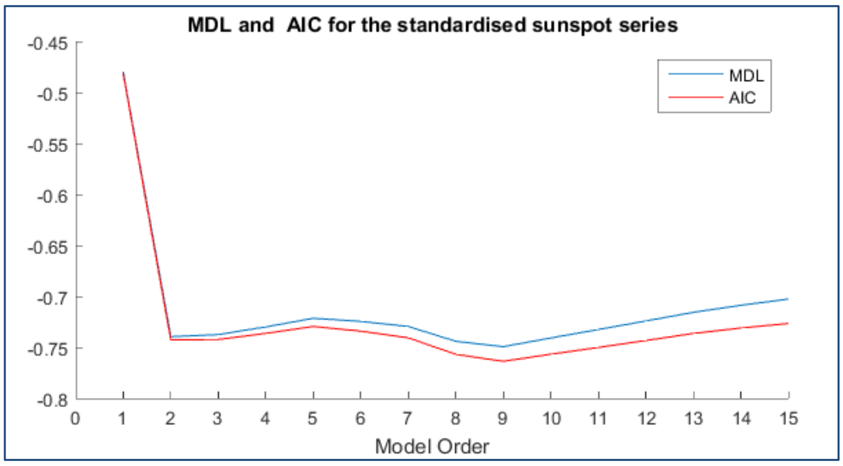


Figure 23: MDL and AIC for the standardised sunspot series

Question 5: For this question we will be comparing the accuracy of different AR models when predicting the sunspot time series m steps ahead. The AR models will vary in number of model order (1, 2, 10) and prediction horizon (1, 2, 5, 10).

Due to the cyclical nature of the sunspot series, AR models with greater prediction horizons will predict more poorly the sunspot time series. This is logical as predicting 10 steps ahead will involves more risk than predicting only 1 or 2 steps ahead.

In addition, considering the model order, we know that the AR(10) model will be performing better than the AR(2) and AR(1) model due to the capability of the AR(10) model to remember the past 10 outputs whereas the AR(2) and AR(1) models only can store 2 and 1 receptively. This makes it hard for these models to predict 5 or 10 steps ahead with only the information about the past few outputs.

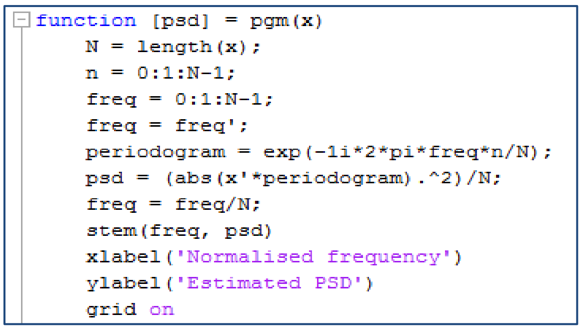
However, we need to remember that the choice of an AR(10) model involves drawbacks such as system and computational complexity which really is only necessary for very specific applications. Therefore for this situation a AR(2) model is sufficient.

2.5 ECG from iAmp experiment

Exercise 3: Spectral estimation and modelling

3.1 Averaged periodogram estimates

For real world applications, we can only estimate the PDF of a stochastic process using a periodogram as the exact value of this process's ACF is undeterminable. Below in Figure 24, we can see our MATLAB function used to approximate the PSD of a given process. We can also see in Figure 25 the generation of the estimation of the PSD, in theory constant, of WGN for different sample lengths of 128, 256 and 512.



```
function [psd] = pgm(x)
    N = length(x);
    n = 0:1:N-1;
    freq = 0:1:N-1;
    freq = freq';
    periodogram = exp(-1i*2*pi*freq*n/N);
    psd = (abs(x'*periodogram).^2)/N;
    freq = freq/N;
    stem(freq, psd)
    xlabel('Normalised frequency')
    ylabel('Estimated PSD')
    grid on
```

For this section we assume for graphs, unless otherwise specified, PSD expressed in dB/Hz and frequency in Hz

Figure 25: PGM function used to estimate process's PSD

Question 1:

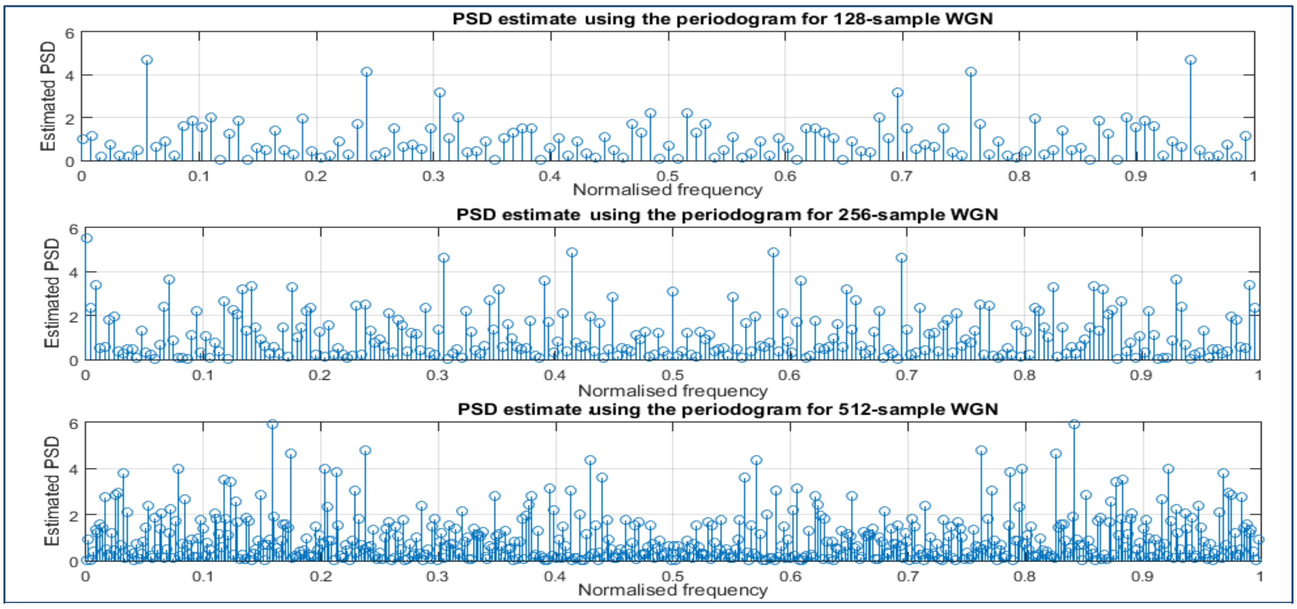


Figure 24: Periodogram for WGN for different sample sizes (128, 256 and 512)

First of all, we can see that the estimated PSD is symmetrical with respect to the y axis. Furthermore, we can also say that the estimation suffers multiple errors that deviate from the theoretical constant PSD of 1, mostly due to the WGN not having its power spread equally over all frequencies due to finite sample numbers and finite frequency ranges. A filter can be used to smoothen out the periodogram.

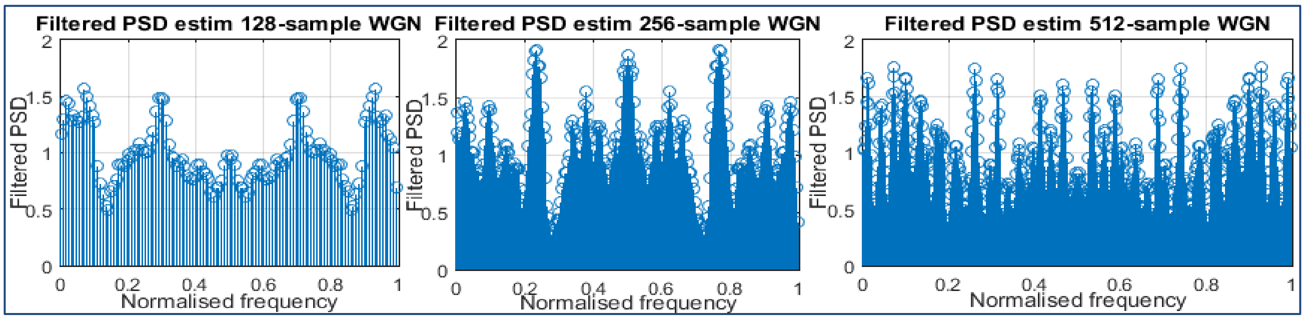


Figure 26: Filtered periodogram for WGN for different sample sizes (128, 256 and 512)

In the Figure 26 (previous page), we can see that the zero-phase FIR filter removes the high frequency components of the PGM that leads to a more constant and predictable estimated PSD. In other words, the value of the estimated PSD converge more towards the constant expected value of the PSD of 1. It can be noted that the efficiency of the filter does not depends on the length of the signal.

Question 2: The generated 1024 sample sequence of WGN is separated into eight non-overlapping 128 sample segments. We will use the average of the different segment's individual PSD's in order to create a better PSD estimator called the average periodogram. The estimated PSD's of the different segments can be seen below in Figure 27, which are similar to the results shown in question 1.

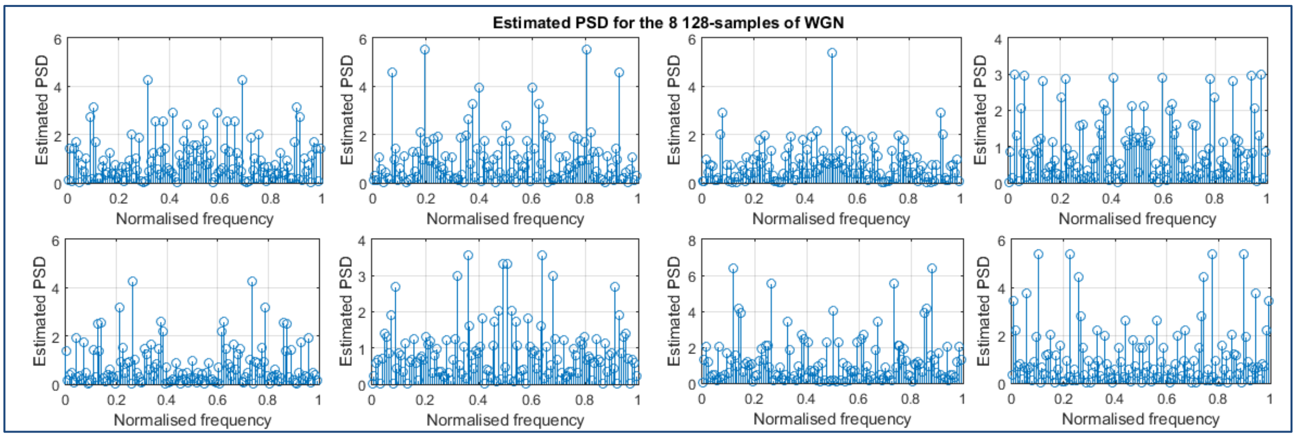
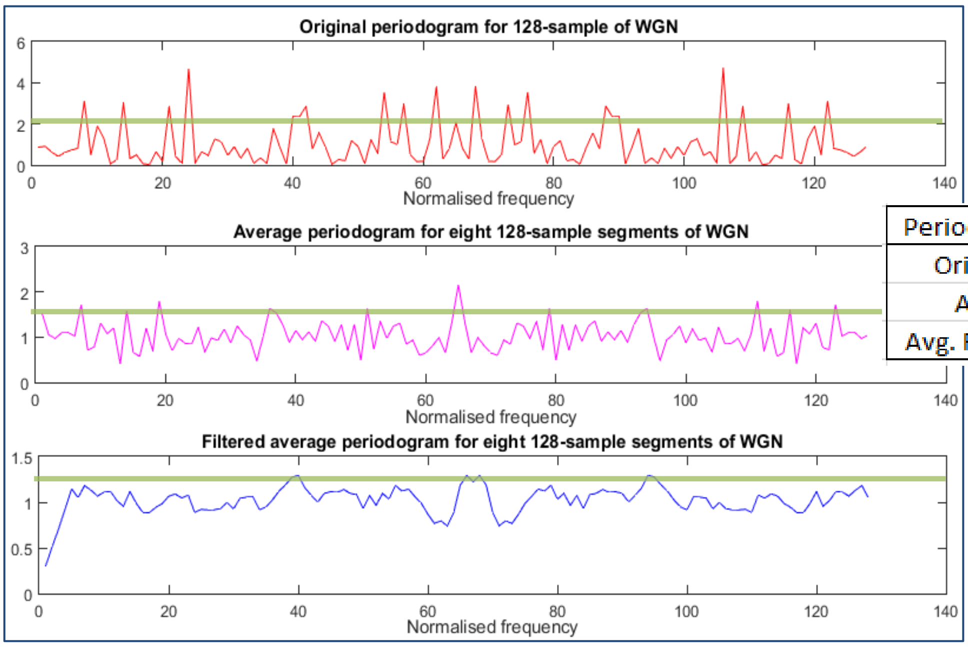
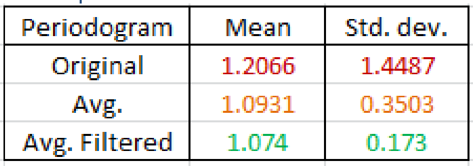


Figure 27: Periodogram for the eight 128-samples segments of WGN

Question 3: As the different segments of WGN are independent from each other, we can do an ensemble average of the segments to obtain a better PSD estimate for the initial WGN signal. As we can see in Figure 28 below, the averaged periodogram offers significant improvement from the original 128-sample segment of WGN. If we want to go even further, we could filter this averaged estimated PSD to get even better results. A small table regrouping the mean and standard deviation of these three periodograms can be seen below, in which we can clearly see quantitatively that the averaged filtered periodogram is the best, having a mean of 1.074 and only a standard deviation of only 0.173 (compared to 1.4487 for the original est. PSD).





Periodogram	Mean	Std. dev.
Original	1.2066	1.4487
Avg.	1.0931	0.3503
Avg. Filtered	1.074	0.173

Figure 28: Original, averaged and filtered average periodograms for 128 samples WGN

3.2 Spectrum of autoregressive processes

Question 1: Figure 29 below shows in blue the exact PSD of the considered filtered signal (WGN) and in red the actual periodogram of the filtered WGN. We can see by observing the blue curve, that the filter is in fact a high pass filter with a cut-off frequency at approximately 0.483Hz which is determined using the 3db roll off point.

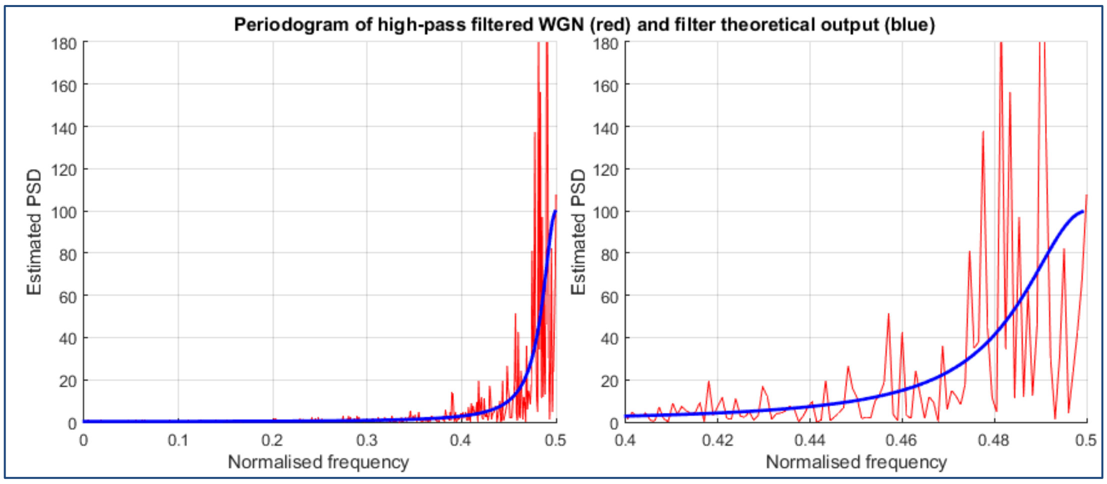


Figure 29: Periodogram of the high-pass filtered WGN (in red) compared to its theoretical value (in blue)

Question 2: On the same plot, we can see in red the periodogram of the filtered WGN. We can observe that the periodogram and the theoretical PSD share a common trend but the periodogram tends to oscillate and have random frequency spikes as the frequency gets closer to 0.5Hz, probably due to the periodogram and our PGM function to not be a perfect estimator of the PSD.

Question 3: We can see the plot zoomed in on the section of the frequency range (0.4, 0.5) in Figure 29, subplot 2. We can see that the oscillations start to increase heavily after the Nyquist frequency. These inaccuracies can be explained by the fact we are using a rectangular window within the periodogram estimator, which is non-linear in the frequency domain. This leads to additional oscillations when convoluting in the frequency domain with our WGN sequence. A solution to this issue would be to apply a Hamming window to the periodogram estimator to reduce the effect of these oscillations.

Question 4: Using the xcorr() MATLAB function, we calculated the AR(1) model parameters to be $\hat{\alpha}_1 = \frac{-\hat{R}_Y(1)}{\hat{R}_Y(0)} = 0.9015$ and $\hat{\sigma}^2{}_X = \hat{R}_Y(0) + \hat{\alpha}_1 . \hat{R}_Y(1) = 1.0151$. We can now find the model based PSD estimate using the formula below, and plot it against the theoretical PSD in Figure 30 below:

$$\hat{P}_y(f) = \frac{\hat{\sigma}^2{}_X}{|1 + \hat{\alpha}_1 . e^{-j2\pi f}|^2} = \frac{1.0151}{|1 + 0.9015 . e^{-j2\pi f}|^2}$$

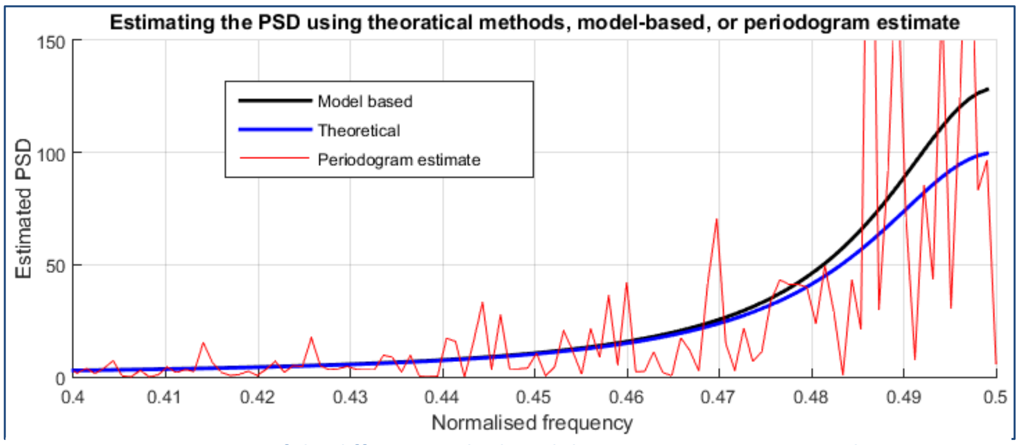


Figure 30: Representation of the different methods and their accuracy at estimating the PSD

It is clear that compared to the periodogram estimate (red), the model based (black) is much more accurate, being nearly identical to the theoretical PSD.

Question 5: We used the AR(2) model and its coefficients to estimate the PSD of the sunspot series. Looking at the Figure 31 below, we can say that the estimated PSD is very similar to the actual sunspot series PSD, with even more similarity for the zero-mean sunspot series.

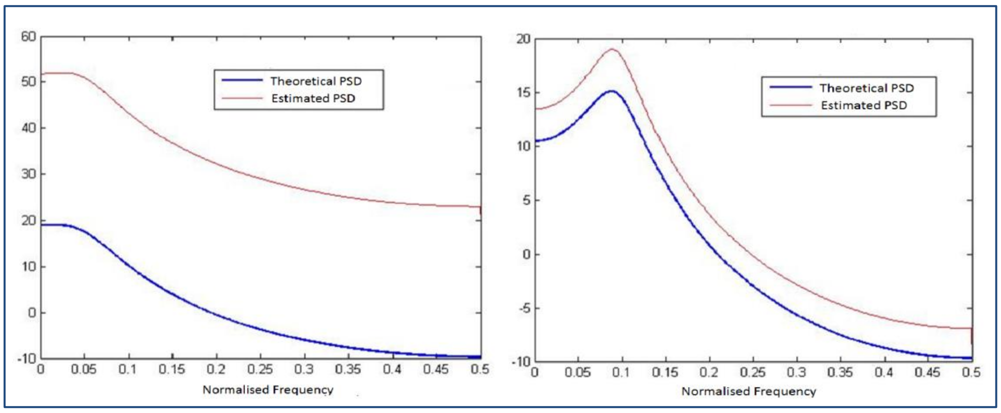


Figure 31: Periodogram for the original and zero-mean sunspot series, along with the theoretical PSD

Over modelling would be better compared to under-modelling when considering the impact of the accuracy and resemblance to the theoretical PSD, but one would need to consider the drawbacks such as system complexity when choosing to over-model.

3.3 Spectrogram for time-frequency analysis: dial tone pad

Question 1: Using a MATLAB function call dial-tone-generator, we create the dial tone signal based on an 11 digit random UK landline number. Below in Figure 32, we can see the signal at different time intervals, with more or less detail.

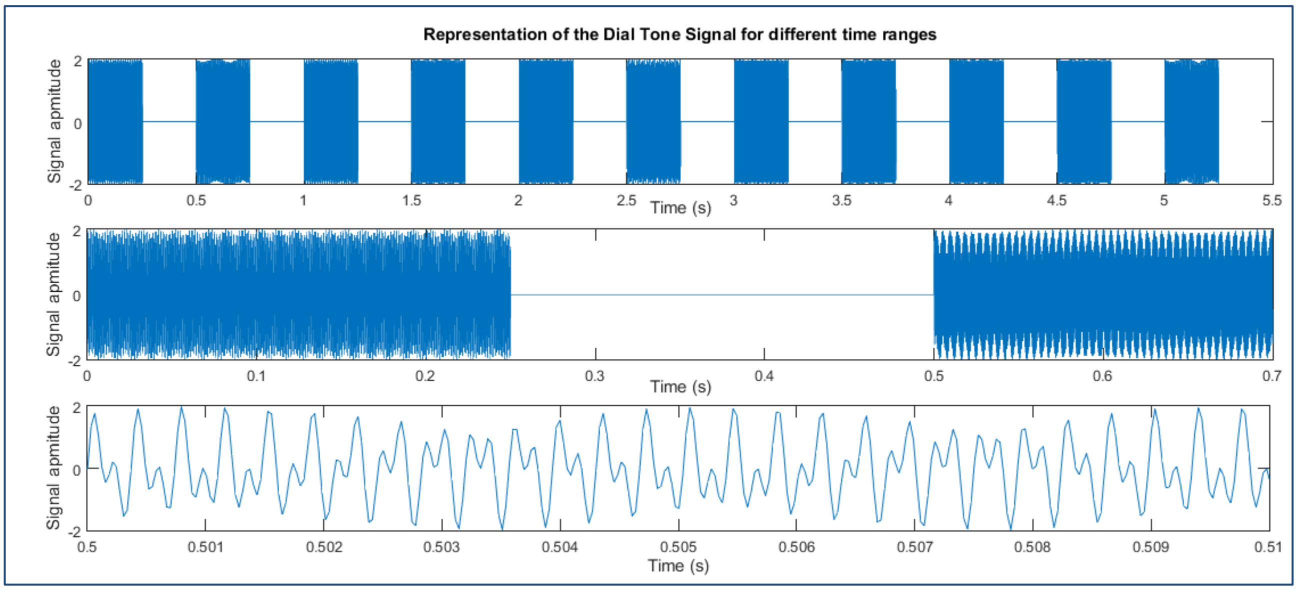


Figure 32: Representation of the dial tone signal for a given random UK landline number

We need a sampling frequency of 32768Hz as the dial tone signal generator overlays sine waves of different frequencies. Sampling at the closest power of 2 of 10 times the Nyquist rate ensures the accuracy of our readings, thus giving 32768Hz. Each key pressed generates a different signal, the one in Figure 32, subplot 3, is the signal generated for the key 2.

<u>Question 2:</u> The figure below represents the dial tone signal spectrogram. We can see in blue the idle time between the pressed digits. In contrast, the light blue/yellow sections represent when a key is pressed. The bright yellow dashes show the dominating frequencies at that time of the signal. We can see that each key pressed is represented by two yellow dashes, which is expected according to the dial tone generator description.

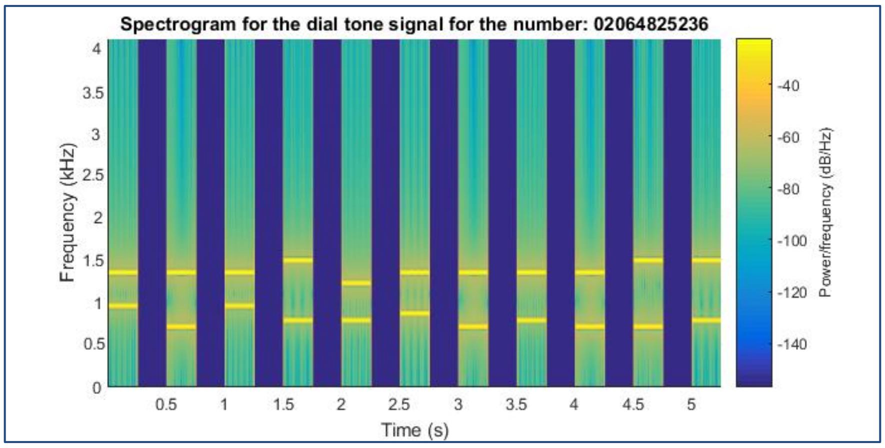


Figure 33: Spectrogram of the dial tone signal for the landline: 02064825236

<u>Question 3:</u> We can identify the different digits in the spectrogram above by looking at the yellow dashes. For example, the first pressed digit has a signal with frequencies at just around 950Hz and another at around 1300Hz. Using the table below, we can identify this signal as representing digit 0. The same process can be done for the rest of the dial tone signal spectrogram.

	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 5: Sine frequencies for each digit

<u>Question 4:</u> To simulate real world application, we added WGN to the dial tone signal. We simulated 3 cases, with increasing exposure to external noise until we the signal is undistinguishable in time domain. The corresponding signals and spectrograms can be seen below in Figure 34 and 35 respectively. The noise added represents 10%, 55% and 200% of the dial tone signal.

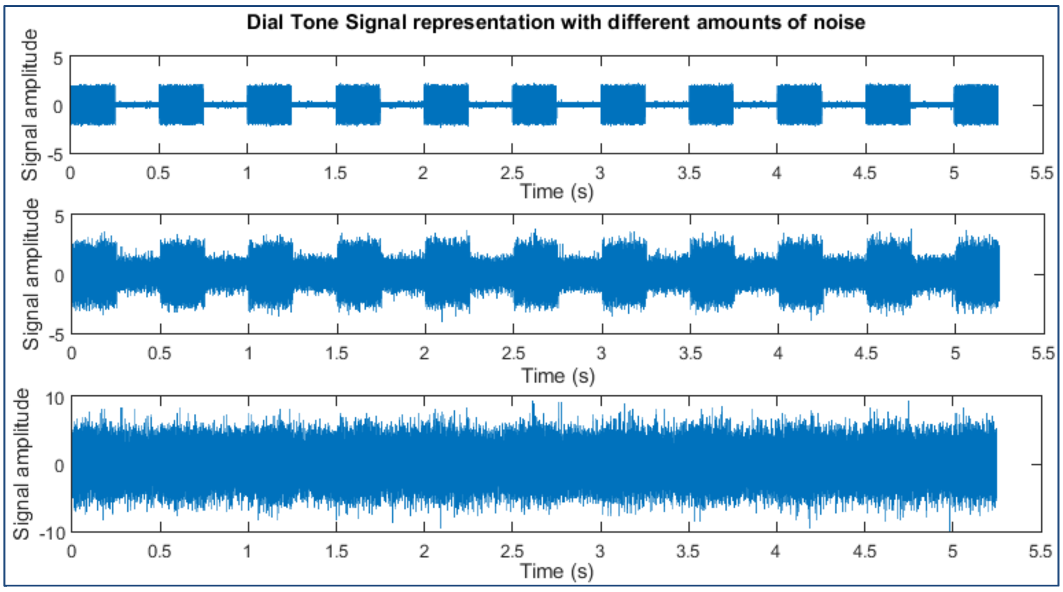


Figure 34: Representations of the dial tone signal with different amounts of added WGN

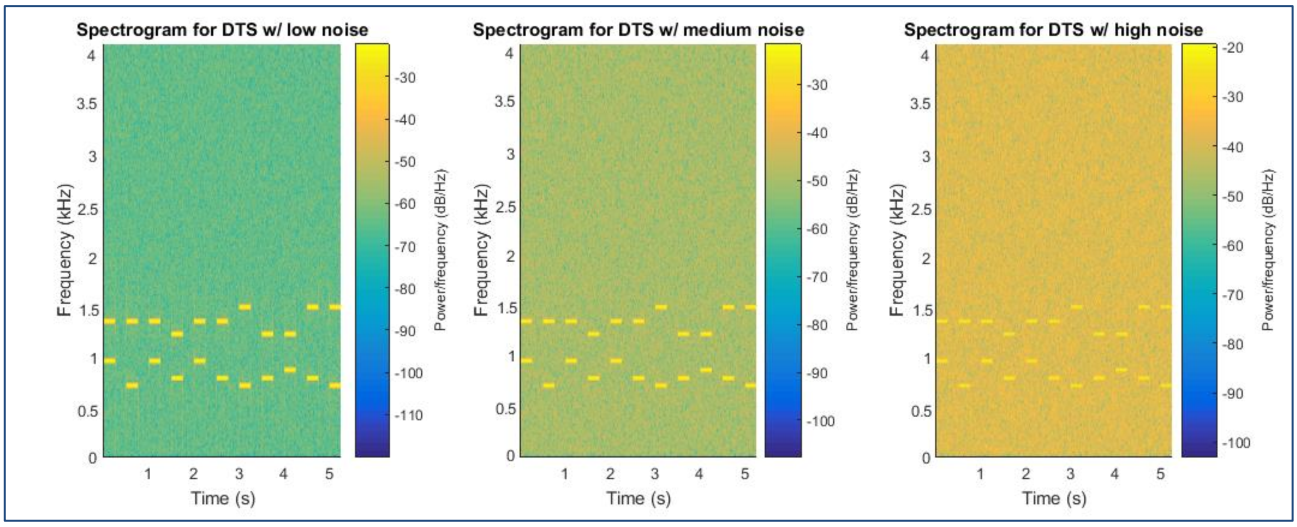


Figure 35: Spectrogram for the dial tone signal for different amounts of added WGN

We can see the effect of adding the WGN to the dial tone signal through the fuzziness of the signal in time domain and the blurriness of the background in frequency domain due to the added frequencies in the spectrum due to WGN. It can be noted that even though we cannot distinguish the signal in time domain when twice as much noise is added compared to the signal, the yellow dashes are still distinguishable in the frequency domain.

3.4 Respiratory sinus arrhythmia from RR-Intervals

Exercise 4: Optimal filtering – fixed and adaptive

4.1 Weiner filter

Question 1: The optimum Weiner filter coefficients can be found using the formula: $w_{opt} = R_{xx}{}^{-1}.p_{zx}$
Below is a table regrouping the values for R_{xx} and p_{zx}, and the calculated values for the optimal coefficients for the Weiner filter compared to those of the unknown system. Part of the code used to calculate these coefficients can also be found below in Figure 26. We have to scale the optimal coefficients up by the same amount we scaled y down.

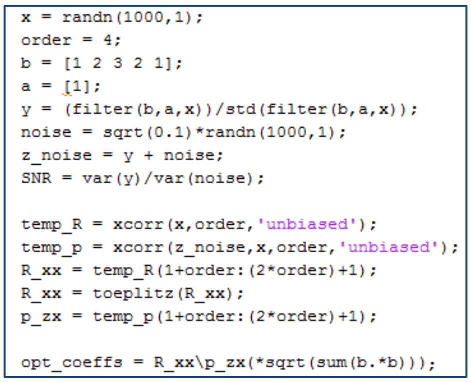
R_{xx}					p_{zx}	w_{unk}	w_{opt}
1.0082	0.0144	-0.0028	-0.0185	0.0232	0.2263	1	0.9721
0.0144	1.0082	0.0144	-0.0028	-0.0185	0.4521	2	1.9221
-0.0028	0.0144	1.0082	0.0144	-0.0028	0.6925	3	2.9439
-0.0185	-0.0028	0.0144	1.0082	0.0144	0.4580	2	1.9468
0.0232	-0.0185	-0.0028	0.0144	1.0082	0.2303	1	0.9890

Table 6: The different value of R_{xx} and p_{zx} used to find the optimal coeffs for the Weiner filter for WGN σ^2=0.1

Question 2: This process can be repeated for other value of the noise variance, i.e. 0.1, 0.5, 1, 2, 5, and 10. The results for the different w_{opt} can be seen in Table 7 below, as well as the SNR.

Noise Var.	0.1	0.5	1	2	5	10
W_{opt}	0.9988	1.0337	1.2086	0.8198	0.6771	1.3738
	2.0386	1.8981	2.1164	1.9496	1.9806	2.4648
	3.0998	2.9685	3.1899	2.9515	3.0154	3.2055
	2.0599	2.0342	2.1171	1.6744	2.1195	2.0945
	0.9705	0.9702	1.1582	0.8015	0.8947	1.4676
SNR	9.6926	2.0204	0.9954	0.5062	0.2194	0.1024

Table 7: The calculated w_{opt}'s for different variance of added WGN



```
x = randn(1000,1);
order = 4;
b = [1 2 3 2 1];
a = [1];
y = (filter(b,a,x))/std(filter(b,a,x));
noise = sqrt(0.1)*randn(1000,1);
z_noise = y + noise;
SNR = var(y)/var(noise);

temp_R = xcorr(x,order,'unbiased');
temp_p = xcorr(z_noise,x,order,'unbiased');
R_xx = temp_R(1+order:(2*order)+1);
R_xx = toeplitz(R_xx);
p_zx = temp_p(1+order:(2*order)+1);

opt_coeffs = R_xx\p_zx(*sqrt(sum(b.*b)));
```

Figure 36: Code used to calculate the w_{opt}'s

We can notice that as we increase the variance of the added WGN, and the SNR goes down, the calculated coefficient for the Weiner filter diverge away from their values at low variance noise.

We will now analyse the effect of increasing the value of N_w (i.e.6, 8, and 10), while keeping the original noise variance at 0.1. In the Table 8, we can see that the first five optimal coefficients remain similar to what we found previously but the rest are equal to approximately 0, which is expected.

N_w	4	6	8	10
W_{opt}	0.9988	1.1053532	0.9974	1.0181
	2.0386333	2.0294	1.9909	2.0460
	3.0998	3.1216	3.0504	3.0602
	2.0599	2.0815	1.9949	2.0736
	0.9705	1.0612	1.0617	1.0253
		0.0637	-0.0194	0.0149
		0.0247	-0.0342	-0.0210
			-0.0263	-0.0113
			-0.0664	-0.1191
				-0.0047
				-0.0021

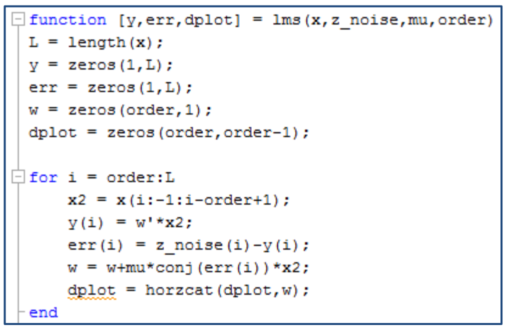
Table 8: Optimal coefficients for different values of N_w

Question 3: We can estimate the number of multiplications and additions in the calculation of the Weiner filter to be: multiplications: N^2 and additions: N^2.

4.2 The least mean square (LMS) algorithm

Question 1: We can see below the MATLAB function used to generate the Least Mean Square (LMS) estimate. We will first look at the accuracy of the estimate compared to the theoretical output we would expect for different values of μ, as seen in Figure 38.

We can see that, as μ increases, the LMS estimation starts to perfectly match the theoretical output. The estimation need some time at the start to match the theoretical signal, which according to our results, depends on the step size μ .



```
function [y,err,dplot] = lms(x,z_noise,mu,order)
L = length(x);
y = zeros(1,L);
err = zeros(1,L);
w = zeros(order,1);
dplot = zeros(order,order-1);

for i = order:L
    x2 = x(i:-1:i-order+1);
    y(i) = w'*x2;
    err(i) = z_noise(i)-y(i);
    w = w+mu*conj(err(i))*x2;
    dplot = horzcat(dplot,w);
end
```

Figure 38: Function to implement LMS estimate

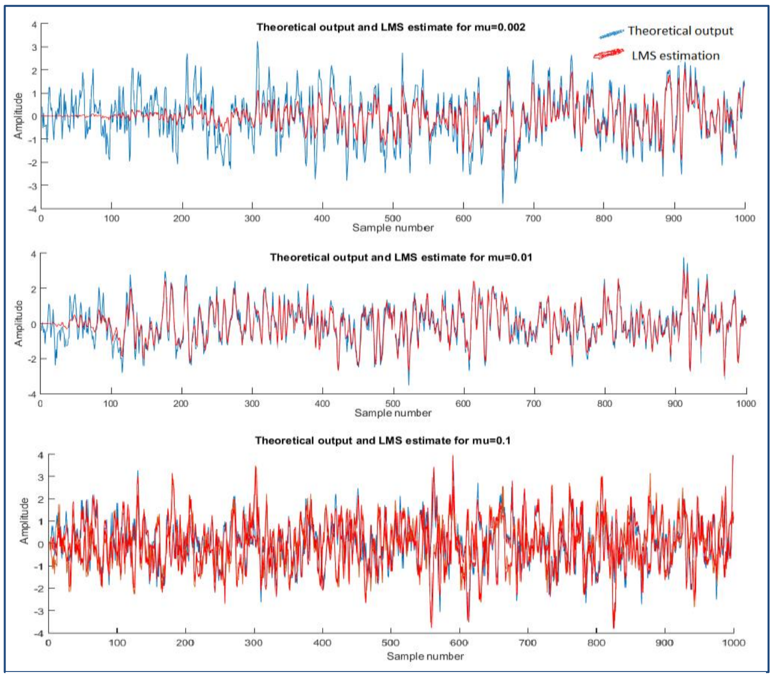


Figure 37: Theoretical output compared to the LMS estimation

Question 2: We now change μ is the same way but look at the filter coefficients and the estimate error evolve over time. We expect the coefficients to rise to a certain value and stabilise and the error to decrease to zero more or less fast, depending on the step size and then oscillate about 0. This is what we can see in Figure 39 and 40 below. It is noted that after increasing μ above the critical 0.1, our model breaks down, we can already see this for $\mu = 0.1$, where the error and coefficients should be stable but oscillate relatively heavily.

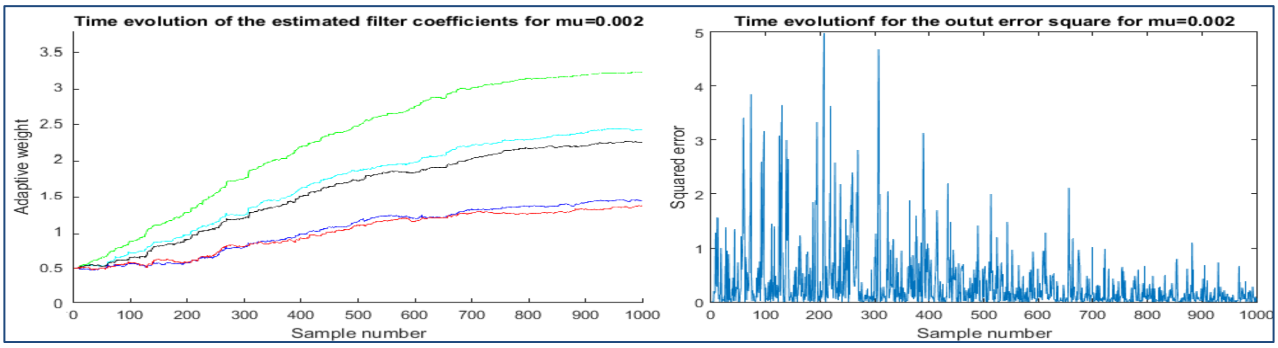


Figure 39: Time evolution of the estimated filter coefficients and output error squared for $\mu = 0.002$

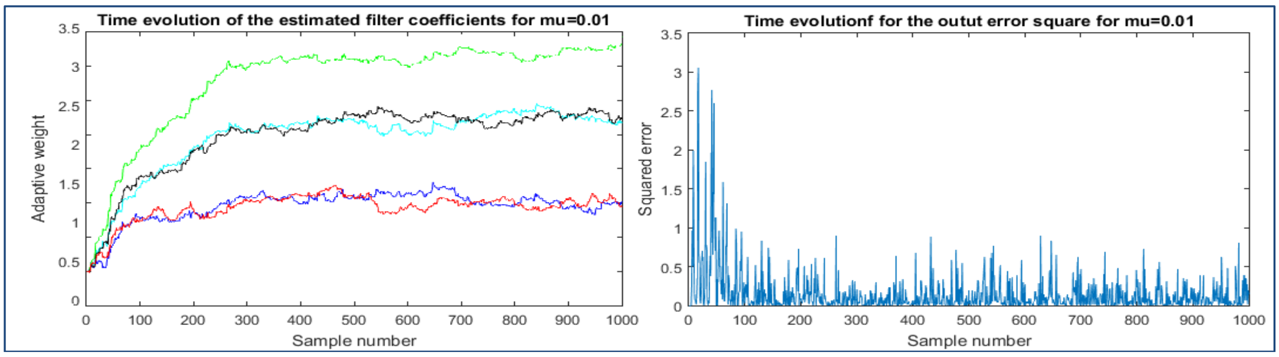


Figure 40: Time evolution of the estimated filter coefficients and output error squared for $\mu = 0.01$

For $\mu = 0.1$, both the coefficients and output error follow the same trend and settle to their respective values even faster that for $\mu = 0.01$.

Question 3: We can estimate the number of multiplications and additions in the calculation of the LMS algorithm to be: multiplications: 4N+3 and additions: 4N+1.

4.3 Gear Shifting

The Figure 41 below shows the resulting graph comparing the filter coefficients and output error varying over time for the LMS algorithm using a varying adaptation gain μ.

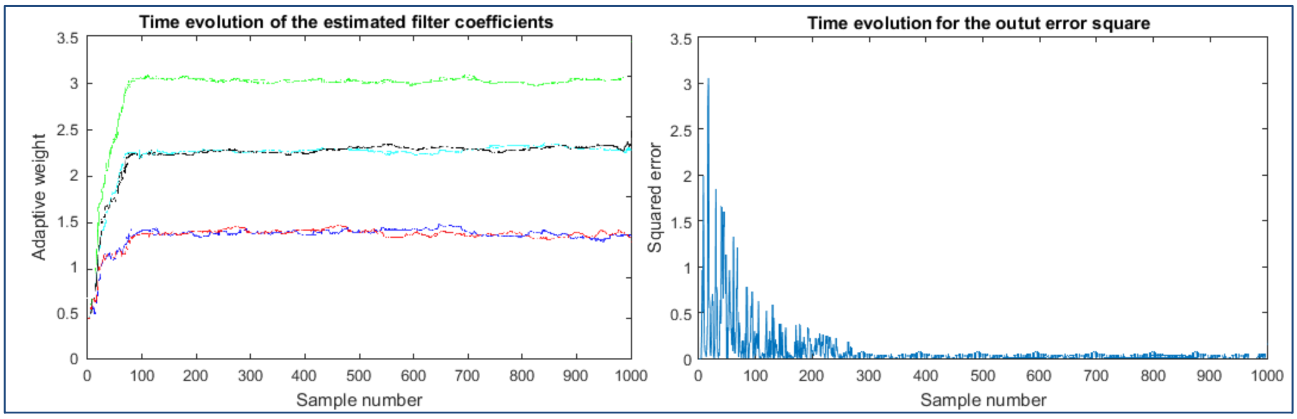


Figure 41: Evolution of the filter coefficients and output error using a varying adaptation gain (using sample number, not error)

We can see that the rise time is much shorter compared to the LSM algorithm with a non-varying adaptation time, which also leads to the output error square to be decaying faster.

Conclusions: We can conclude that the error and rise time are both greater if the adaptation gain is greater and vice versa when the adaptation ain is smaller. However, at large adaptation rates, the system becomes unstable and we see overshoots just before the system become unstable. Therefore, one could say that depending on the sample number, we should use different values of the adaptation gain, a larger one at the start to increase the error converging time/rise time of the coefficients and later a smaller value to have more accurate estimates and less oscillations. This gear shifting method accomplishes that by setting the adaption gain to: $\mu(i) = \mu(i = 0)/i + 1$ with i: sample number. Another possibility involves feeding back the error into an equation giving out an adaptation gain that would not depends on the sample number but on the current error of the output. The equation defining μ would fluctuate between 0.001 for example when the error is low and 0.1 when the error is high.

4.4 Identification of AR processes

We can see from the Figure 42 (next page) that the AR coefficients converges towards -0.2 and -0.9 (due to the formatting of the coefficients by MATLAB, they are negative), and oscillate slightly around their respective value. As the adaptation gain increases, the rise time of the coefficients (until they get to their nominal value_ decreases but the oscillations after that increase. For example for μ = 0.01, the evolution in time of the AR coefficients are in the range ±0.1 of -0.2 whereas for μ = 0.05, they are in the range ±0.4 of -0.2. For μ > 0.01, the model breaks down and the evolution of the coefficients is unreliable.

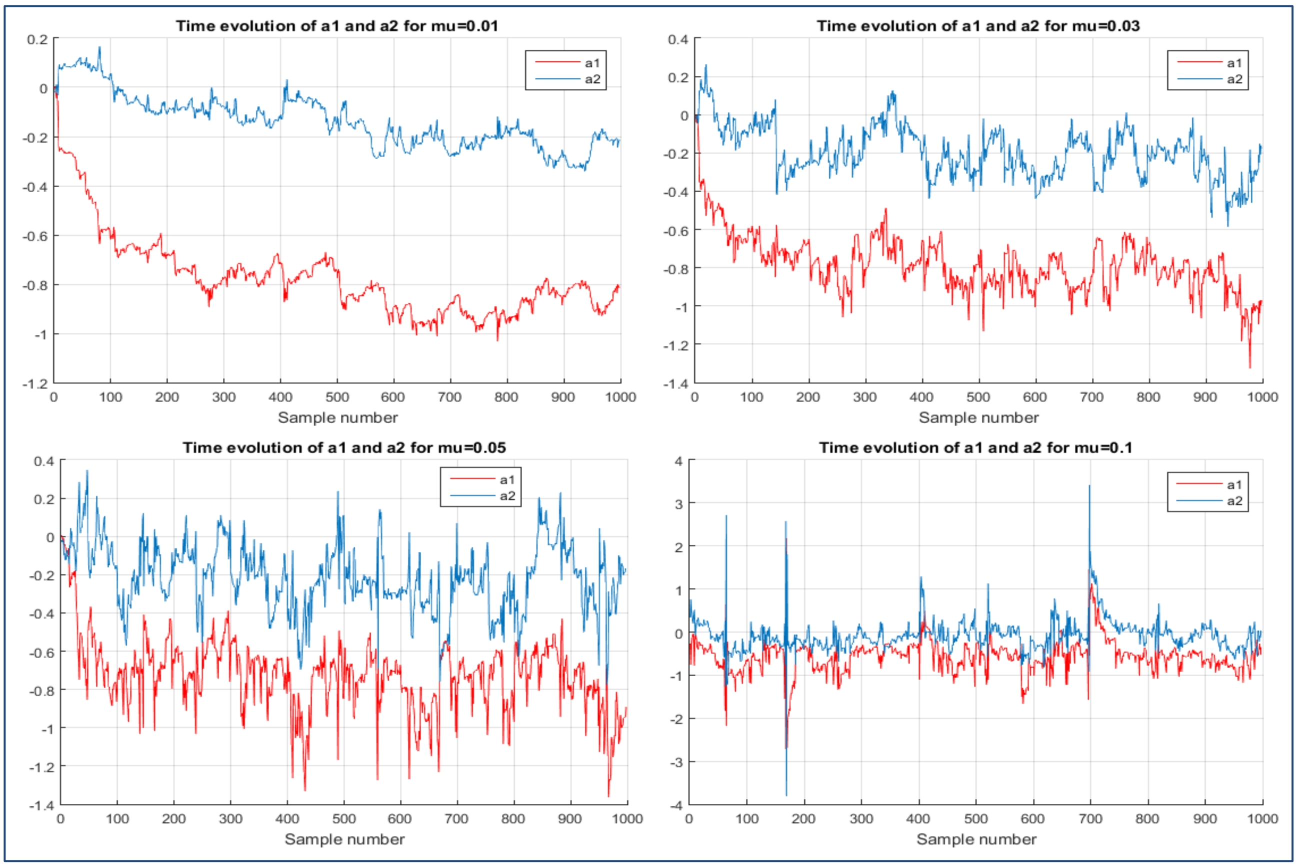


Figure 42: Time evolution of the AR coefficients a1 and a2 for different adaptation gains (0.01, 0.03, 0.05, and 0.1)

4.5 Speech recognition

Question 1: For each letter sampled at 44100Hz, we extract a segment of 1000 samples in order to input into the predictor. We varied the adaptation gain (5, 15, and 30) as well as varied the order of the predicator (5, 15, and 30). The results can be seen below for the letter A and S in Figures 43 and 44 respectively.

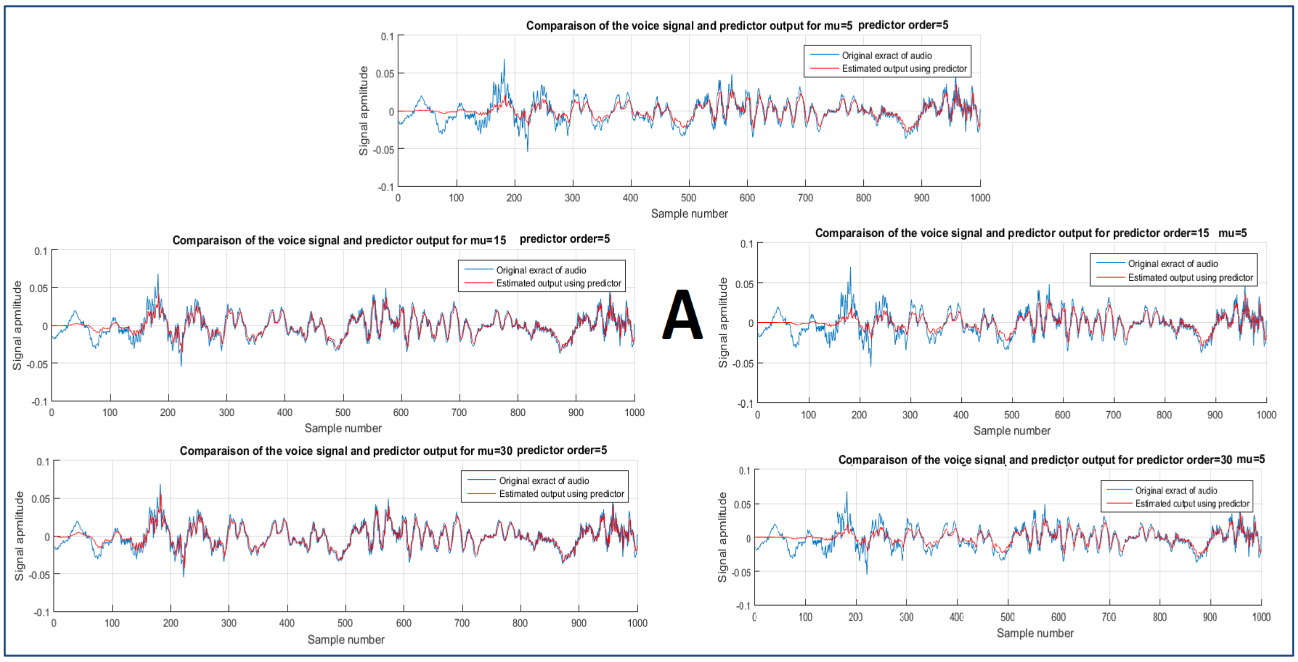


Figure 43: Voice signal compared to the output of the predictor for different predictor orders and adaptation gain for the letter A

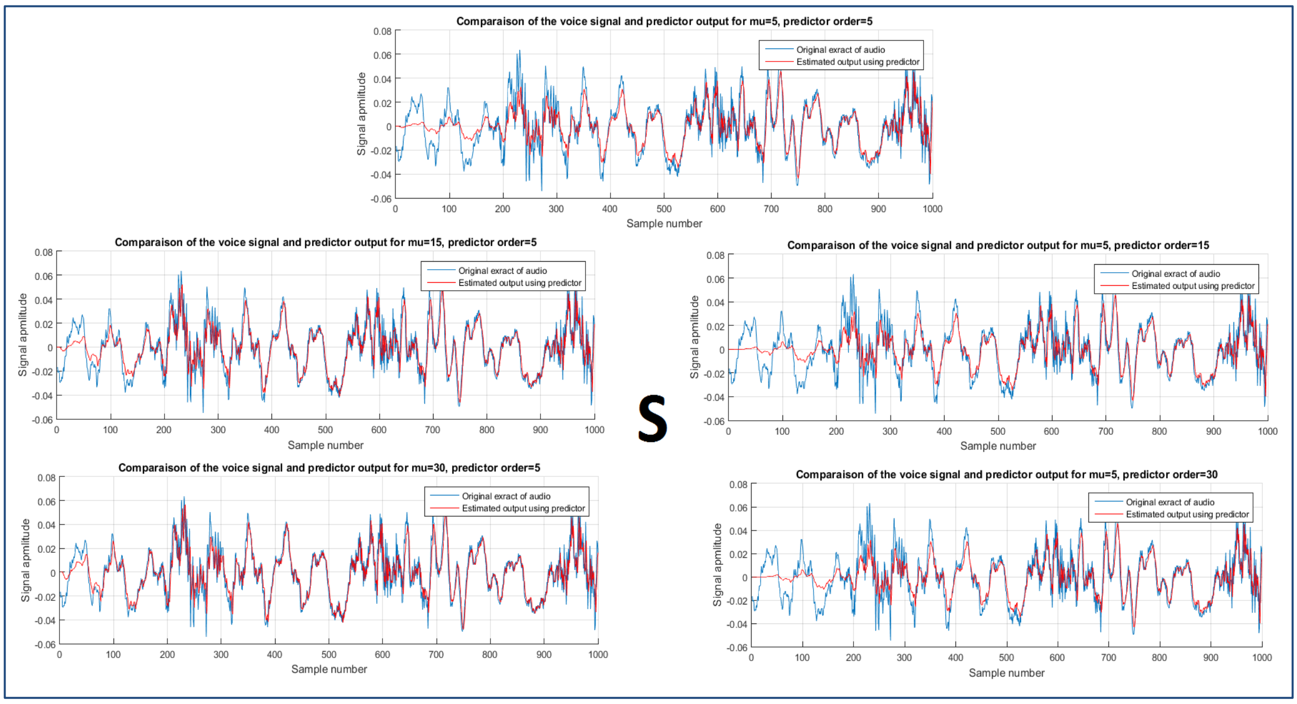


Figure 44: Voice signal compared to the output of the predictor for different predictor orders and adaptation gain for the letter S

We can clearly see that as the value of the adaptive gain µ is increased (left column), the predictor output converges more quickly to the original voice signal extract. Nevertheless, as we have seen before, increasing the adaptive gain by too much will lead to oscillations and inaccuracies. Regarding the predictor order, for a fixed adaptive gain, the graphs do not show much of a change for a varying predictor order (right column), as the estimation with a predictor order of 5 is already fairly accurate. However, we should expect higher accuracy for higher predictor order, at the cost of system complexity.
The rest of the letters follow a similar trend, but for the sake of the readability of this report have been excluded it.

Question 2: We can define the optimal filter length to be the shortest predictor length that still correctly outputs a close estimate to the original voice signal extract. As we have seen, there is a trade-off between accuracy and complexity when considering the order of the filter. A heuristic approach to find the best filter length would be to repeat the tests above with a broader spectrum of orders for all the letters and choose the smallest order number overall that still estimated correctly the original signal extract. An analytical approach would use tests like the Minimum Description Length (MDL) or Akaike Information Criterion (AIC) for example.

Question 3: The predictive gain calculated using its given equation can be found in Table 9. The decreased sampling frequency can be translated into a decrease of data points but an increase in change of value between successive data points. This will lead to the predictor making mistakes of larger amplitude. Therefore, as can be seen in Table 9, the prediction gain is larger for the higher sampling rate. To conclude, we can also say that to obtain quasi-stationary vowel sounds, we would require less data points at lower sampling frequency, but we would still require more data points for the learning curves to converge. This is due to the monotony of the signal being determined with less data points at lower sampling frequencies.

Letter	Prediciton Gain	
	Fs = 44100Hz	Fs = 16000Hz
a	20.47	11.88
e	21.71	10.70
s	13.94	6.79
t	15.03	8.16
x	13.19	7.69

Table 9: Prediction gains for different F_{samp}

4.6 Dealing with computational complexity: sign algorithms

In order to demonstrate dealing with computational complexity, we used an adaptation gain of µ = 0.0001. In addition, we will use a simplified version of the LMS algorithm known as the class of sign LMS algorithm. This modified LMS algorithm is good when dealing with computationally heavy signals and for high speed applications.

We can see in Figure 45 below the estimated AR(2) coefficient a1 and a2 evolve over time. We can see that all configurations lead to the same desired output with a2 = -0.2 and a1 = -0.9 but they all take different converging trends with the no sign implementation converging the quickest and the sign sign implementation the slowest.

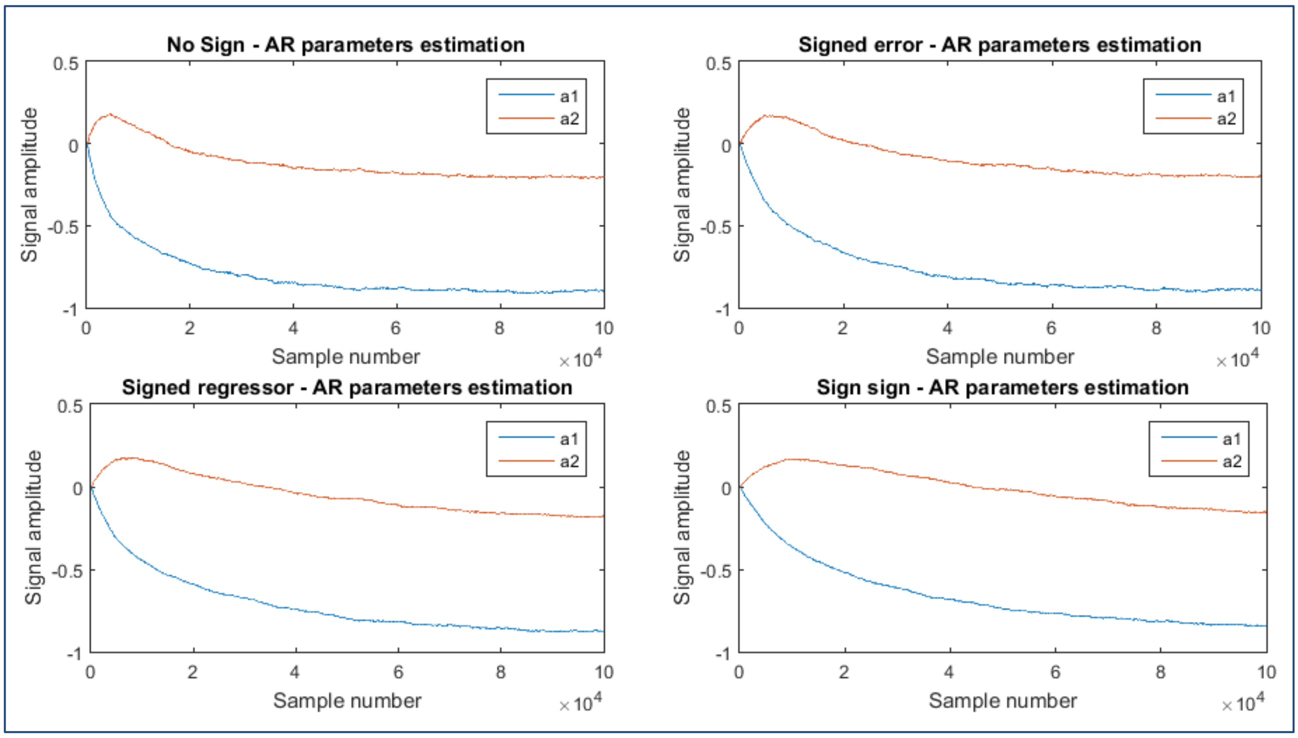


Figure 45: Estimation of the AR(2) model coefficients a1 and a2 using different sign LMS algorithms used for high computational complexity application

Exercise 5: A real world case study: Vinyl Denoising

Question 1: Below in Figure 46 we can see the PSD estimate using the periodogram for the Led Zeppelin corrupted signal for the right and left channel.

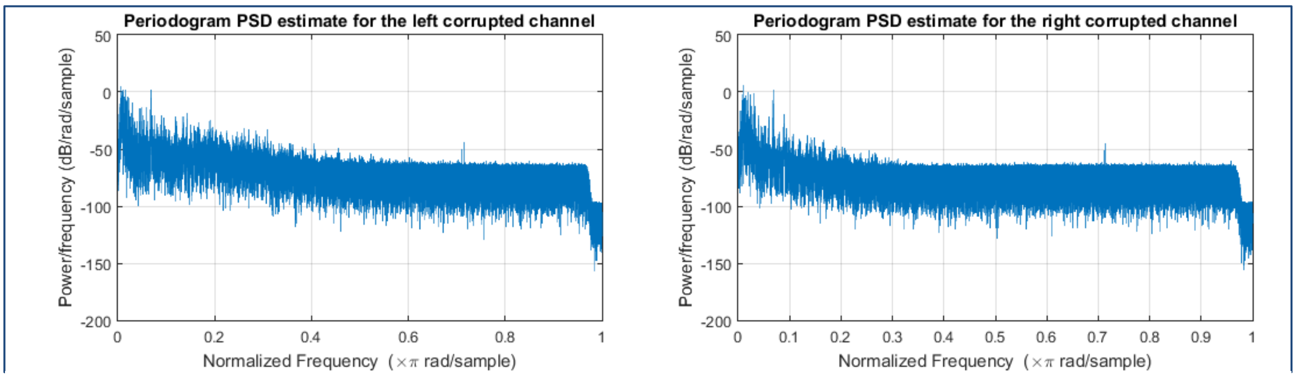


Figure 46: Periodogram PSD estimate for the left and right channel corrupted Led Zeppelin extract

Using just the PSD estimate, it is impossible to differentiate the voice signal spectral component to the tick spectral component. Even when comparing to the PSD of the original signal, it would be very hard. However, we can use tool such as the spectrogram and FFT's to try and singularise the frequencies of the tick sounds that are present in the corrupted signal and not in the original signal.

Question 2: We can first see below in Figure 47, the estimated PSD of the original signal. We will then be looking at a comparison of the spectrograms of the original and corrupted signals to see if we can notice any differences in Figure 48. We can notice on the spectrogram a slight difference in spectral components with an added line of small dots along the axis where the black arrow has been drawn.

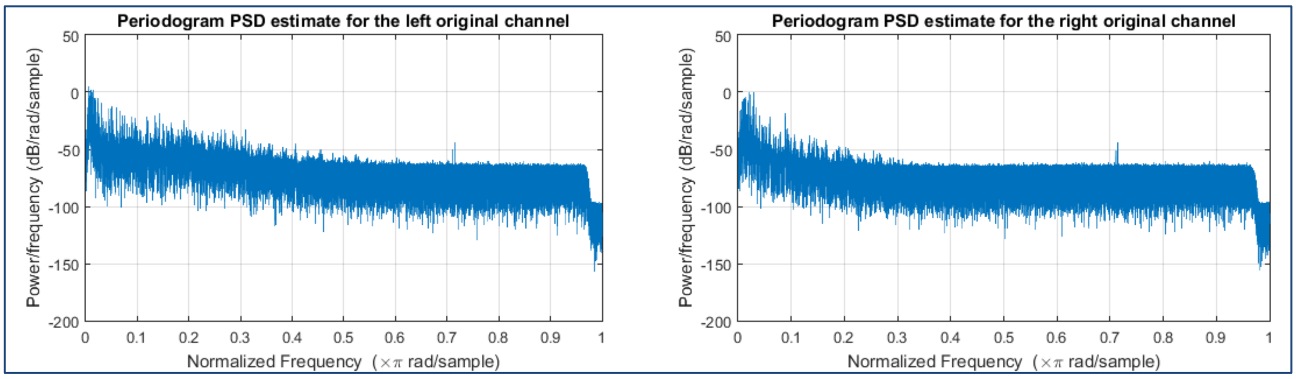


Figure 48: Periodogram PSD estimate for the left and right channel original Led Zeppelin extract

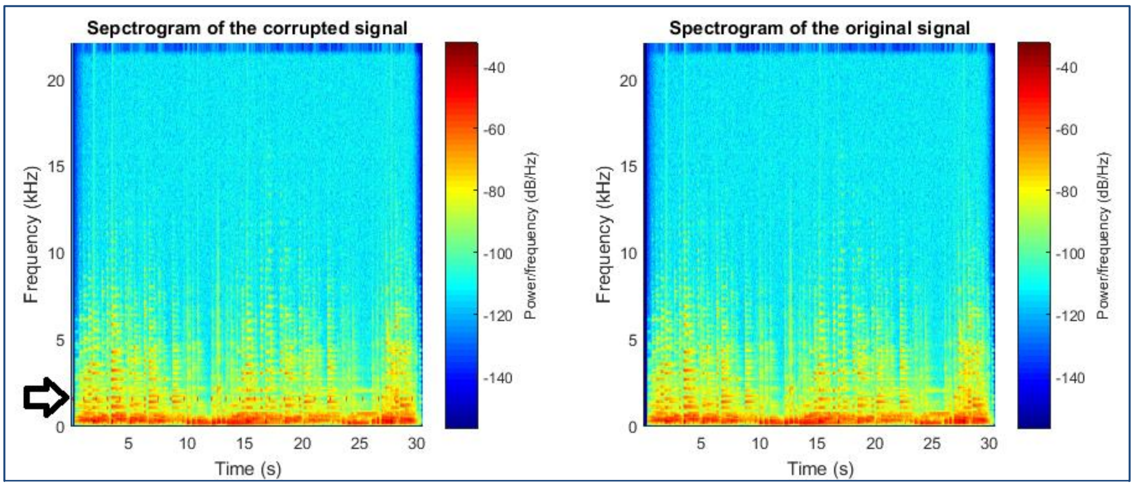


Figure 47: Spectrogram for the corrupted and original audio extract

Finally, to find the exact range of frequencies that cause the tick, we will be taking the FFT of both the original and corrupted signal and calculated their difference. This will then allow us to build an appropriate filter to supress these unwanted ranges of frequencies. Note that in the intent of graph visibility, the x-axis has been truncated to only show the relevant parts.

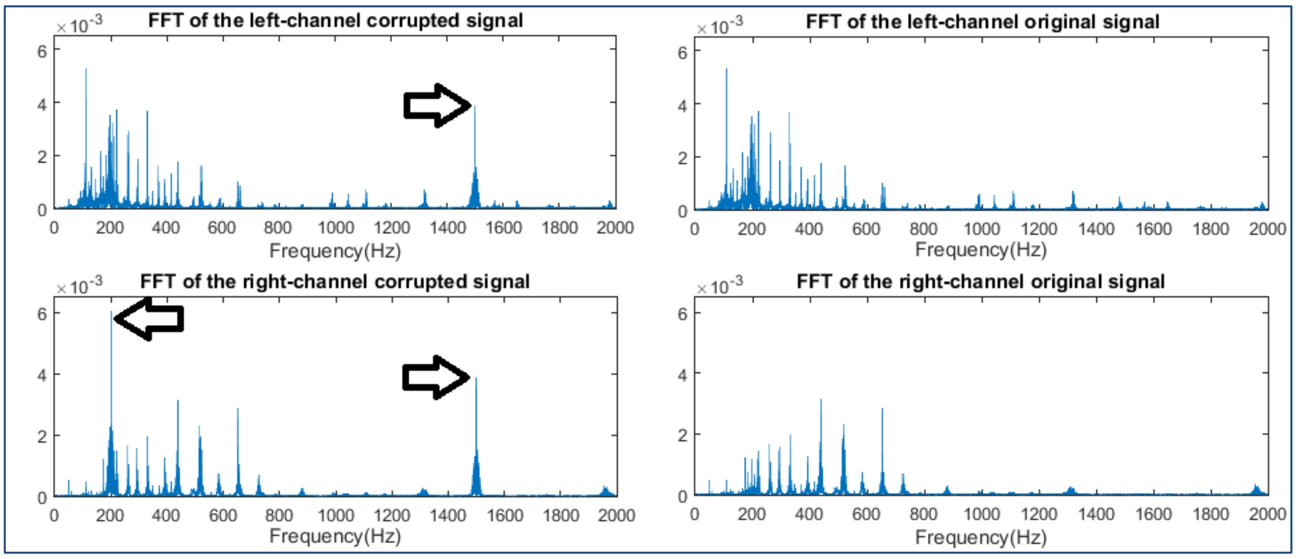
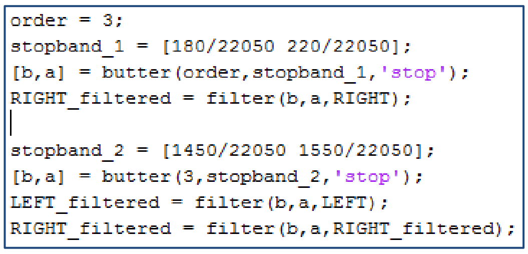


Figure 49: FFT of the left and right channel original and corrupted signal of the audio extract

We can see that the left channel corrupted signal FFT has an extra spike at around 1500 HZ compared to the original signal's FFT. On the other hand the right channel has a spike at 200 Hz and another one at 1500 Hz.

Question 3: Now that we know the frequencies that cause the imperfection we can design a filter to supress those frequencies. We will be using a Butterworth filter as we require removing frequencies within a small range (±20Hz & ±50Hz) around the spike but not above that range. We can see below in Figure 51, an extract of our MATLAB code that performs this filter and also the resulting FFT post-filtering in Figure 50.



```
order = 3;
stopband_1 = [180/22050 220/22050];
[b,a] = butter(order,stopband_1,'stop');
RIGHT_filtered = filter(b,a,RIGHT);

stopband_2 = [1450/22050 1550/22050];
[b,a] = butter(3,stopband_2,'stop');
LEFT_filtered = filter(b,a,LEFT);
RIGHT_filtered = filter(b,a,RIGHT_filtered);
```

Figure 50: Code used to filter the corrupted signals

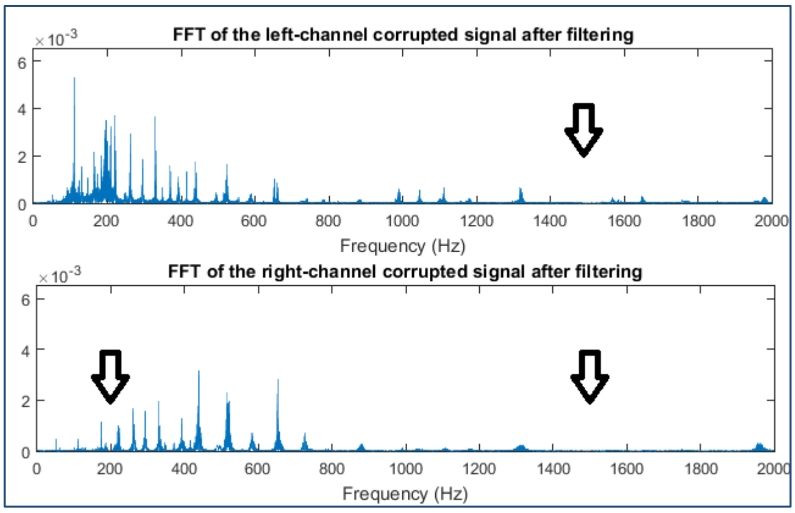


Figure 51: FFT of the filtered corrupted signals

We can see that the spikes have now been supressed successfully by the Butterworth filter. This is also confirmed when listening to the audio sample back in the time domain as the ticks have now been removed.

Butterworth filter coefficients used to remove the tick at 200 Hz							
a	1.00	-5.84	14.32	-18.92	14.19	-5.73	0.97
b	0.99	-5.78	14.26	-18.92	14.26	-5.78	0.99
Butterworth filter coefficients used to remove the tick at 1500 Hz							
a	1.00	-5.99	14.93	-19.87	14.88	-5.94	0.99
b	0.99	-5.96	14.91	-19.87	14.91	-5.96	0.99

Table 10: Representation of the Butterworth filter coefficients to filter at 200Hz and 1500Hz

<u>Question 4:</u> We can now compare the PSD's of the three signals, as can be seen in Figure 51 below. We can see that the tick's PSD in red has been removed but however we also removed in the process some spectral components of the original signal. However, this has little to no effect to the audio signal when played back while still deleting the tick. Note that in the intent of graph visibility, the x-axis has been truncated to only show the relevant parts.

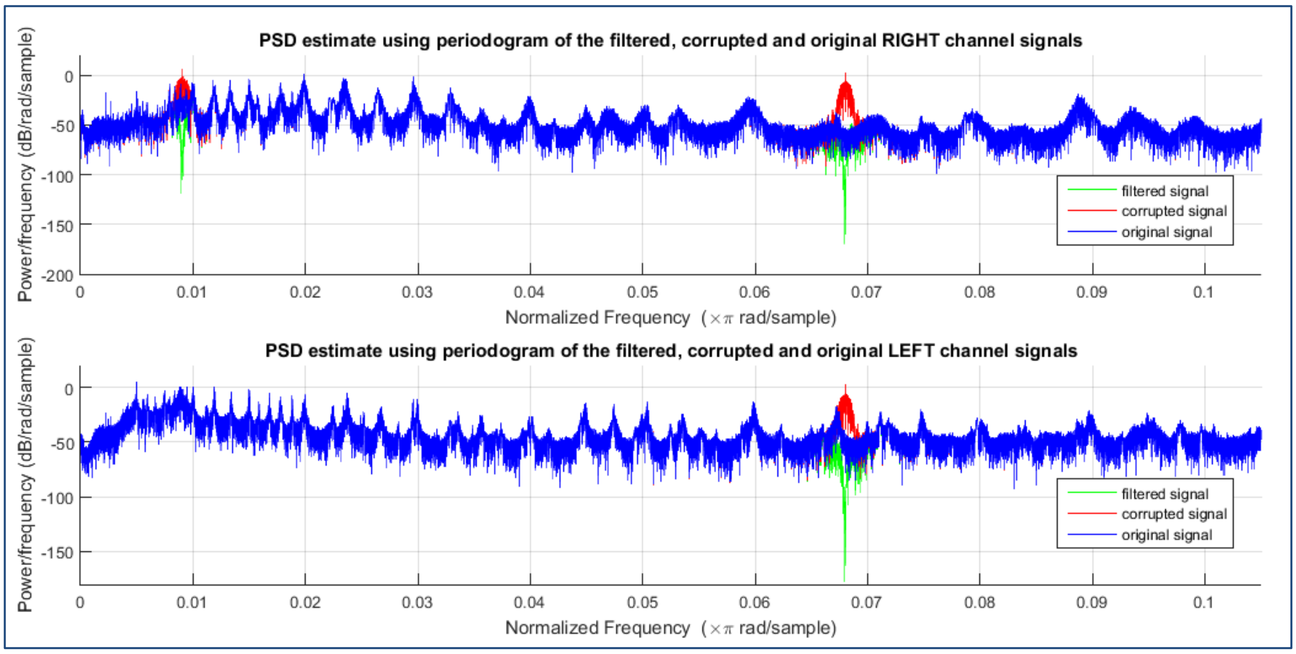


Figure 52: PSD estimate of the filtered signal using a fixed-coefficient digital filter compared to the original signal

We can also compute the relative error of our estimate according to the original sample according to the performance measure and the predication gain. These values can be seen in the table below. It can be noted that the large difference of values between the channels is due to the 2 ticks that were present on the right channel, leading to more estimation error.

Signal Channel	Quantit. Perf. Measure	Prediction Gain
RIGHT	0.1254	5.2013
LEFT	0.0072	19.3793

Table 11: Accuracy measure for the right and left channel filtered signal

<u>Question 5:</u> We developed a supervised adaptive denoising algorithm which does not require the analysis of the whole audio signal that uses the LMS estimator and the original signal as a teaching signal to eliminate the noise. We can see in figure 53 below that, with a step size of 0.1, all the noise components have been removed from the corrupted signal without impacting the quality of the filtered signal.

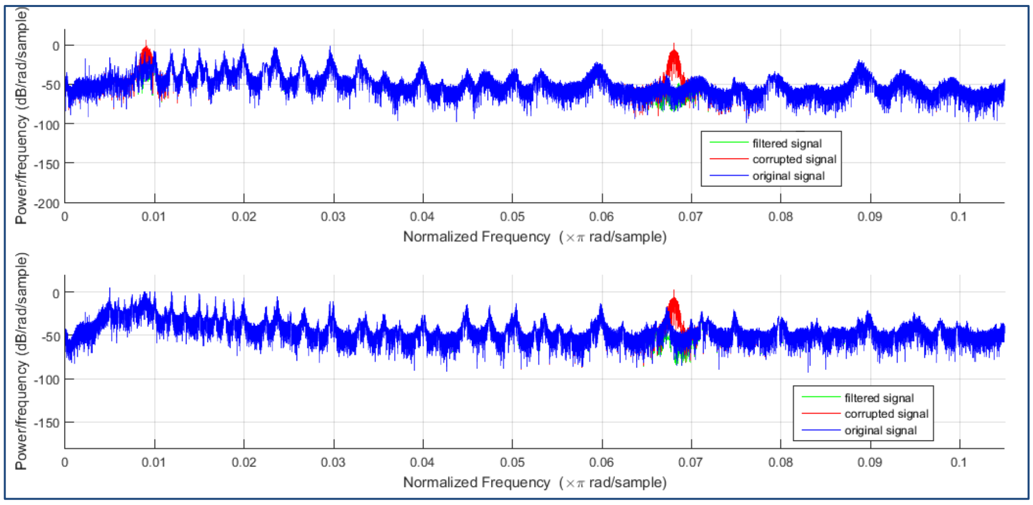


Figure 53: PSD estimate of the filtered signal using an adaptive denoising algorithm

This Normalised LMS algorithm used allows us to implement a time-varying step size because it uses the original signal as a teaching signal. This removes the trade-off between convergence and rise rate and accuracy as it can be both. We can see this with the evaluation of this algorithm in the table below. We can see that both the qualitative performance measure and the prediction gain increase compared to the previous method using a fixed-coefficient digital filter.

Signal Channel	Quantit. Perf. Measure	Prediction Gain
RIGHT	0.0290	23.95
LEFT	0.0070	33.76

Table 12: Accuracy measure for the right and left channel filtered signal

Question 6: We applied the adaptive filter from the previous section to a different music sample. The new recording has much more frequency components compared to the previous sample, which will lead to the spectral components of the tick sound mixed in with the spectral components of the music. Therefore we cannot apply a fixed-coefficient digital Butterworth filter as it would take away a larger chunk of frequencies, including those from the melody. However, we can see from Figure 54 that the adaptive filter, by lowering the step size to around 0.01, is able to supress the frequency causing the tick without damaging the spectral frequencies associated to the actual music.

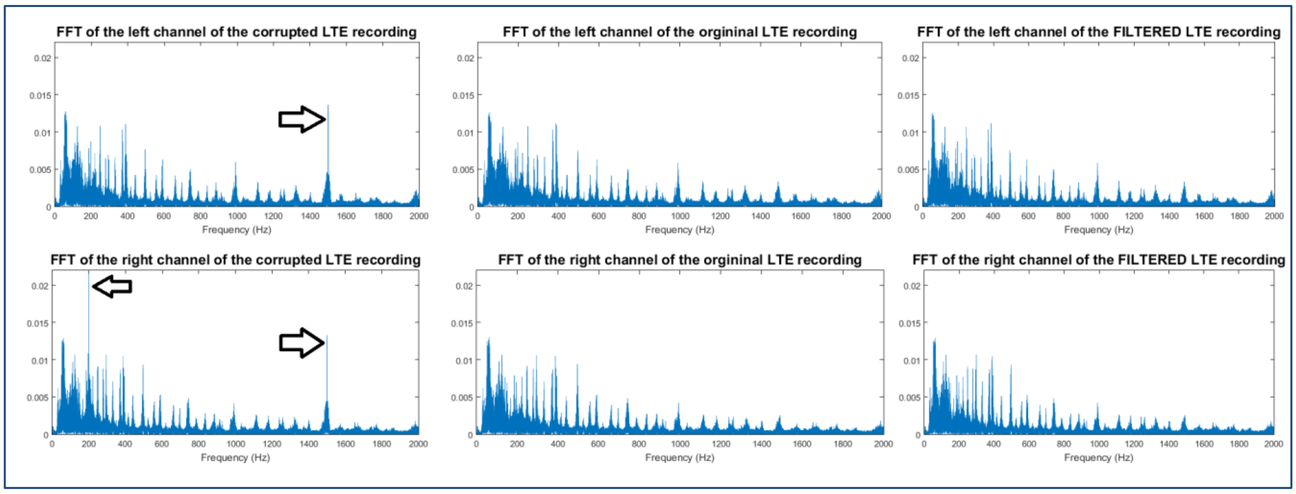


Figure 54: PSD estimate of the signal with high frequency compnents using an adaptive denoising algorithm compared to original

We can analyse the accuracy of this implementation but we would expect similar results at the previous scenario because it is the same filter. The results of the calculation of the two standards can be found in the table below.

Signal Channel	Quantit. Perf. Measure	Prediction Gain
RIGHT	0.0230	29.58
LEFT	0.0100	35.73

Table 13: Accuracy measure for the right and left channel filtered signal