Imperial College London

ADVANCED SIGNAL PROCESSING

Name: Yao Lei Xu
Login: YLX15
CID: 01062231

Contents

(1) Random signals and stochastic processes

1.1 Statistical estimation

1.1.1 Uniform distribution: mean

For the signal generated in figure 1, the sample mean is $\hat{m} = \frac{1}{N}\sum_{n=1}^{N} x[n] = 0.5135$. This is close to the theoretical mean of the uniform distribution *U(0, 1)* which is 0.5, with an error of 0.0135 or 2.7%. With increasing samples, the sample mean should approach the theoretical mean by the law of large numbers

1.1.2 Uniform distribution: standard deviation

The sample standard deviation is instead found to be: $\hat{\sigma} = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(x[n] - \hat{m})^2} = 0.2897$. This is also close the theoretical value of $\sqrt{E(X - E(X))^2} = 0.2887$ with an error of 0.001 (or 0.35%).

1.1.3 Uniform distribution: bias

The sample means and standard deviations for 10 realisations of X are plotted in figure 2 below, which clearly cluster about their theoretical values:

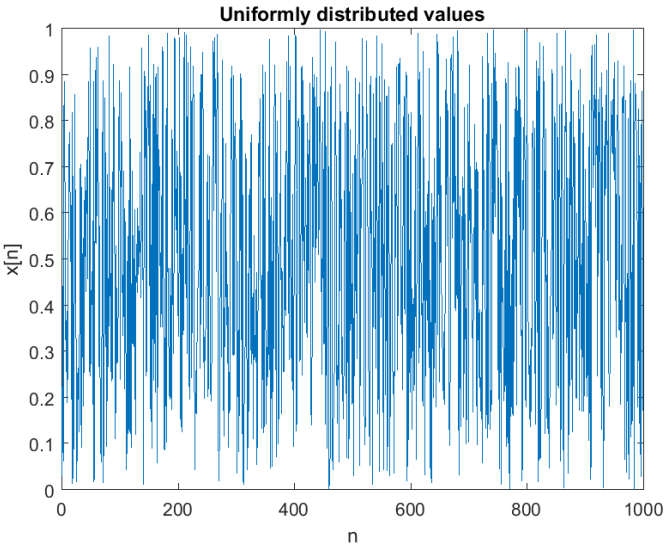


Figure 1 – stochastic process 1

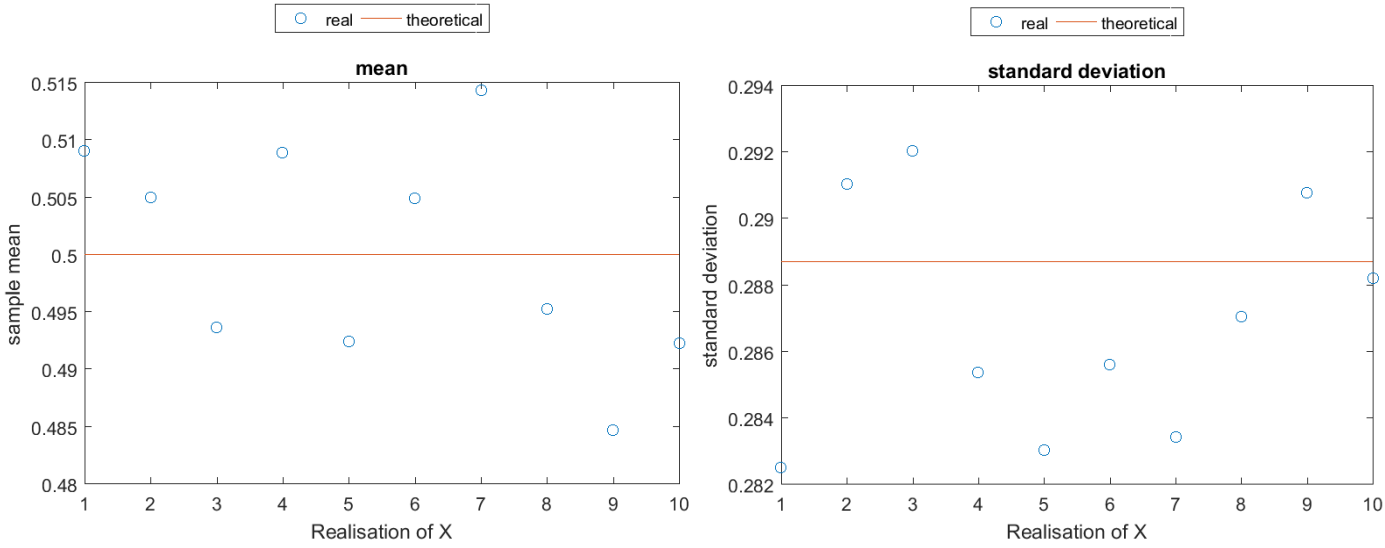


Figure 2 – mean and std of process 1

The computed values tend to oscillate about the theoretical value closely. Using MATLAB to compute the ensemble average and then subtracting it from the theoretical values, we get a mean bias of 0.0014 (0.3% error from the theoretical value) and the standard deviation bias of 0.00058 (0.2% error from the theoretical value).

1.1.4 Uniform distribution: PDF

To approximate the probability distribution function, it is possible to use histograms and normalise the values such that the area underneath is one. This allows us to gather more information about the random variable than the classical averages and standard deviations.

The top row of figure 3 below shows how the approximated PDF converge to the theoretical distribution as the number of samples increases, which is expected as for the law of large numbers. The bottom row of figure 3 instead shows how the histograms tend to approximate the theoretical distribution better with fewer bins. This is also expected as if less bins are present, the more realisations of a random variable will be contained in each bin, hence better approximating the theoretical statistics.

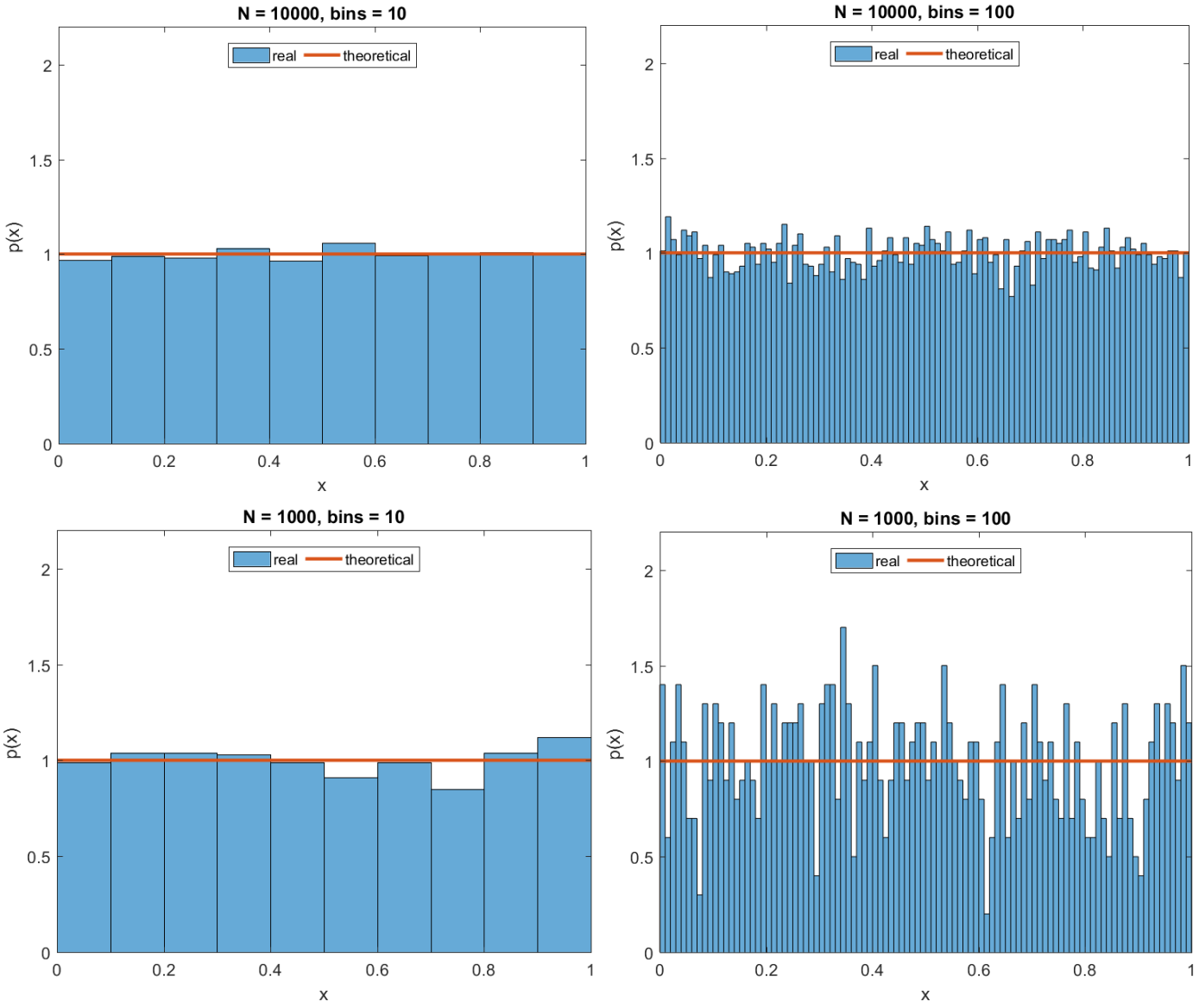


Figure 3 – uniform distribution approximation

1.1.5 Gaussian distribution: mean

For the Gaussian distribution in figure 4, the sample mean is $\hat{m} = \frac{1}{N}\sum_{n=1}^{N} x[n] = -0.0121$. This is close to the theoretical mean of the uniform distribution *N(0, 1)* which is 0. With increasing samples, the sample mean should approach the theoretical mean.

1.1.5 Gaussian distribution: standard deviation

The standard deviation is $\hat{\sigma} = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(x[n] - \hat{m})^2} = 0.9815$. This is also close the theoretical value of $\sqrt{E(X - E(X))^2} = 1$ with an error of 0.0185 or 1.85%

1.1.5 Gaussian distribution: bias

The sample means and standard deviations for 10 realisations of X are plotted in figure 5 below, which again are clustering about their theoretical values:

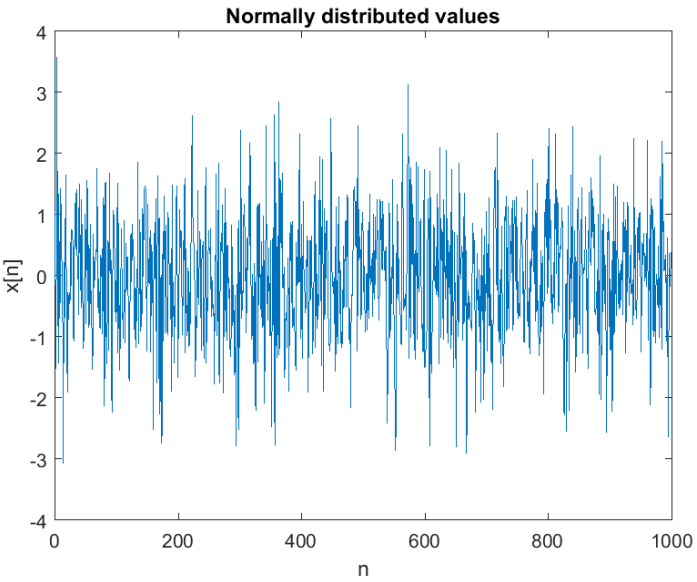


Figure 4 – stochastic process 2

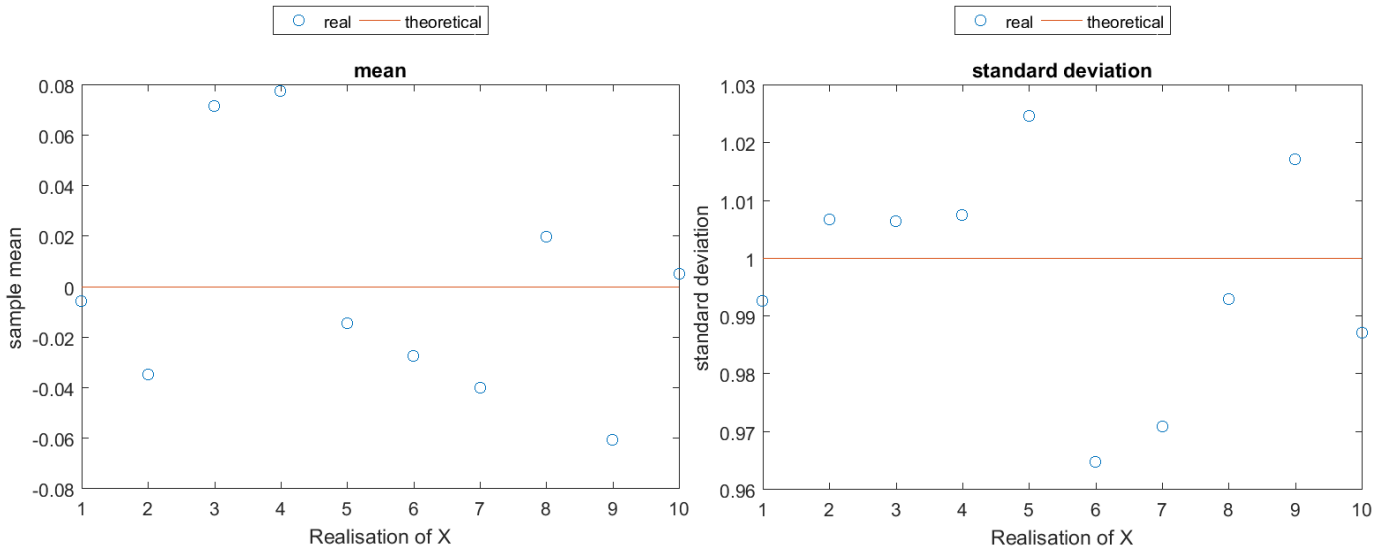


Figure 5 – mean and std of process 2

The computed values tend to oscillate about the theoretical value closely. Using MATLAB to compute the ensemble average and then subtracting it from the theoretical values, we get mean bias = 0.00091 and the standard deviation bias = 0.0022 (0.22% error from the theoretical value).

1.1.5 Gaussian distribution: PDF

The same conclusion for the number of samples and the number of bins applies here as well when approximating the underlying probability distribution, as shown in figure 6.

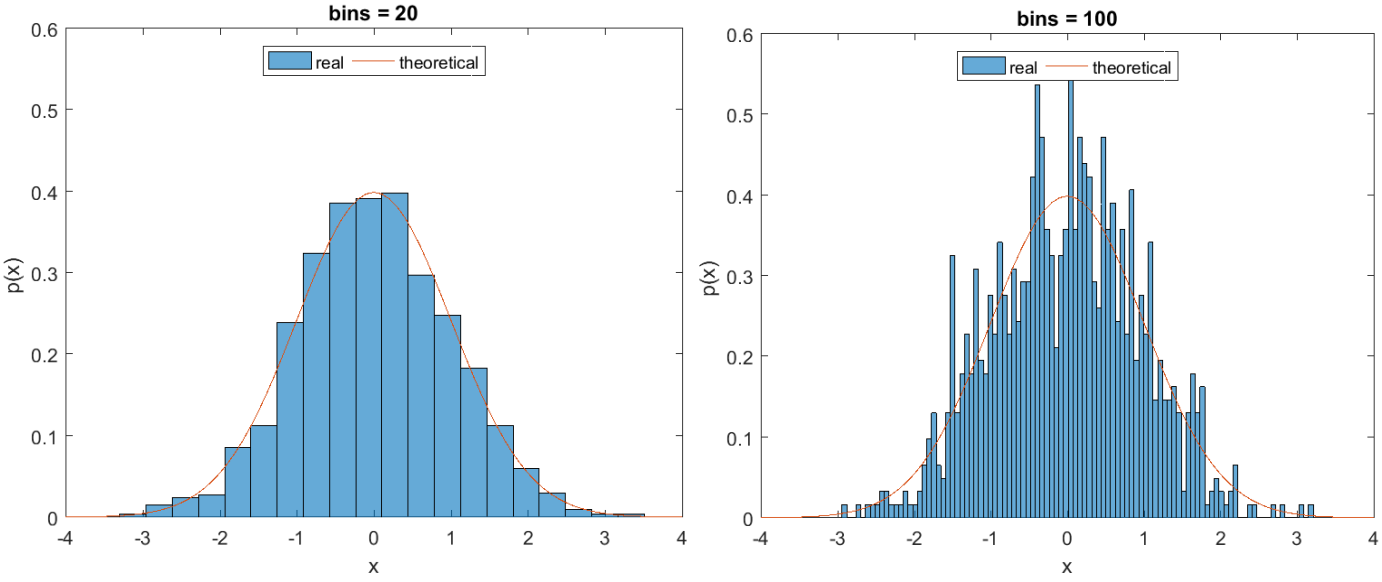


Figure 6 – Gaussian distribution approximation

1.2 Stochastic processes

Definitions

- Time average is the average of 1 realisation of the stochastic process over time.
- Ensemble average is the expected value of multiple realisations of a random variable at a given time.
- Ergodic process is a stochastic process for which the time average equals the ensemble average.
- A stationary process is a process whose probability distribution does not change when shifted in time (therefore no change in mean and variance over time)

1.2.1 Ensemble mean and standard deviation

Figure 7 below shows the ensemble and time statistics of the random process 1. The time average and standard deviation are taken along the time axis (along n as n ranges from 1 to 100), and they are plotted as individual points for each realisation of the random process. The ensemble statistics are taken instead along multiple realisations at a fixed time (fixed n value) and it is plotted as continuous lines that evolve over time.

It is clear from the plot that both the ensemble mean (due to the term Ac in the MATLAB implementation) and the ensemble standard deviation (due to Mc term) change over time. This means that the stochastic process is not stationary. In addition, time average fluctuates around one for all 100 realisations, which differs from the ensemble average that varies with time. This mean that the process is not ergodic neither.

By applying a similar analysis to random process 2 plotted in figure 8, it is clear that RP2 is not ergodic either. For instance, the ensemble mean and the standard deviation fluctuate around some fixed values over time, which shows that the random process is stationary. However, the time averages and standard deviations of 100 realisations do not cluster around the ensemble average and standard deviations, which suggests that the process is not ergodic.

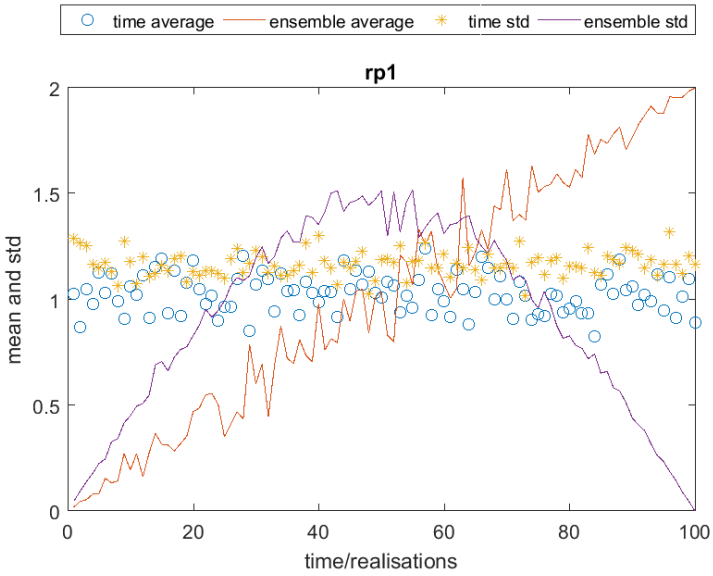


Figure 7 – RP1 stats

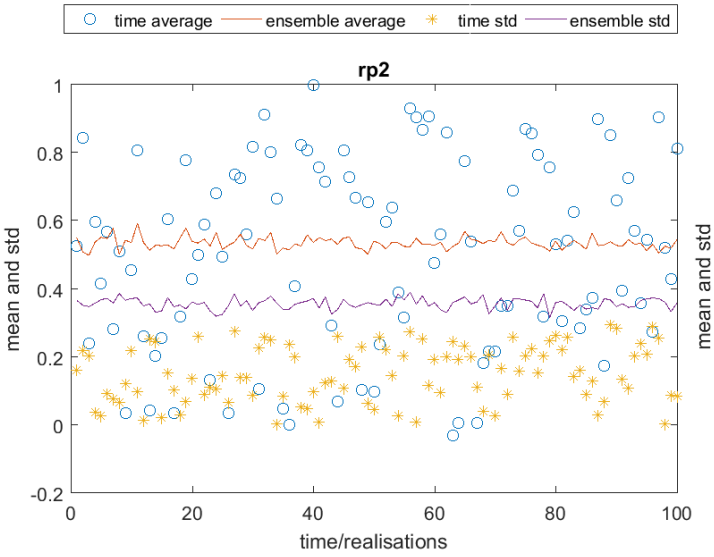


Figure 8 – RP2 stats

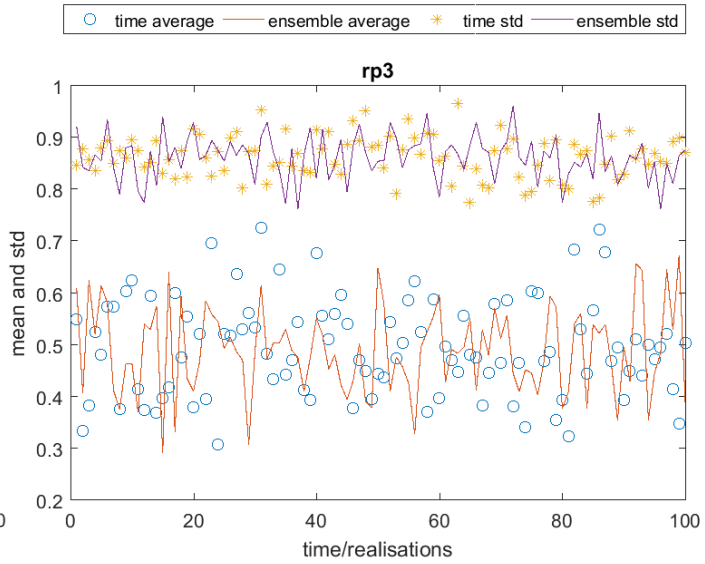


Figure 9 – RP3 stats

Finally, as shown by figure 9, the ensemble statistics fluctuate slightly over time and the time statistics cluster around the ensemble ones for RP3, meaning that it is both stationary and ergodic.

1.2.2 Ergodicity

	Time average			Time std		
	RP1	RP2	RP3	RP1	RP2	RP3
Realisation 1	9.993	0.651	0.507	5.856	0.193	0.883
Realisation 2	9.984	0.360	0.516	5.872	0.074	0.853
Realisation 3	10.022	0.850	0.494	5.867	0.035	0.882
Realisation 4	10.013	0.543	0.523	5.882	0.012	0.875
Average	10.003	0.601	0.510	5.869	0.079	0.873

Table 1 – RP1/2/3 stats summary

As discussed earlier, RP1 and RP2 are not ergodic as the ensemble statistics differ from time statistics, which means that we can't reach important conclusions about the random process with just one realisation over time. RP3 is the only

ergodic process, whose time average and the standard deviation fluctuate about 0.51 and 0.87 respectively for all 4 realisations, which fluctuate around its ensemble statistics.

1.2.3 RP1: Mathematical description

The random process can be described as:

$$RP1 = \left(b\sin\left(\frac{\pi}{N}n\right)\right)\varphi + an, \quad \varphi \sim U(-0.5, 0.5), \quad a = 5, \quad b = 0.02$$

The mean is:

$$E(RP1) = E\left(\left(b\sin\left(\frac{\pi}{N}n\right)\right)\varphi + an\right) = E\left(\left(b\sin\left(\frac{\pi}{N}n\right)\right)\varphi\right) + E(an) = 0 + an = an = 0.02n$$

The variance and standard deviation are:

$$variance(RP1) = E(RP1^2) - E(RP1)^2$$

$$= E\left(\left(b^2\sin^2\left(\frac{\pi}{N}n\right)\right)\varphi^2\right) + E\left(2ab\sin\left(\frac{\pi}{N}n\right)\varphi n\right) + (an)^2 - (an)^2$$

$$= E\left(\left(b^2\sin^2\left(\frac{\pi}{N}n\right)\right)\varphi^2\right) + 0 = \frac{1}{12}\left(b^2\sin^2\left(\frac{\pi}{N}n\right)\right) = \frac{25}{12}\left(\sin^2\left(\frac{\pi}{N}n\right)\right)$$

$$\therefore std(RP1) = 1.44\left(\sin\left(\frac{\pi}{N}n\right)\right)$$

As n ranges from 1 to N = 100, the mean and the standard deviation match the graph generated in the previous section.

1.2.3 RP2: Mathematical description

The random process can be described as:

$$RP2 = XY + Z, \quad X \sim U(0,1), \quad Y \sim U(-0.5, 0.5), \quad Z \sim U(0,1)$$

The mean is:

$$E(RP2) = E(XY) + E(Z) = 0 + E(Z) = 0.5$$

The variance and standard deviation are:

$$variance(RP2) = E(RP2^2) - E(RP2)^2$$

$$= E((XY)^2 + 2XYZ + Z^2) - 0.25 = E(X^2)E(Y^2) + 0 + E(Z^2) - 0.25 = \frac{1}{3}\frac{1}{12} + \frac{1}{3} - 0.25 = \frac{1}{9}$$

$$\therefore std(RP2) = 0.33$$

Hence the theoretical results match the graphs for RP2 as well.

1.2.3 RP3: Mathematical description

The random process can be described as:

$$RP3 = m\varphi + a, \quad \varphi \sim U(-0.5, 0.5), \quad m = 3, \quad a = 0.5$$

The mean is:

$$E(RP3) = E(m\varphi) + E(a) = 0 + a = 0.5$$

The variance and standard deviation are:

$$variance(RP3) = E(RP3^2) - E(RP3)^2$$

$$= E((m\varphi)^2 + 2am\varphi + a^2) - 0.25 = 0.75 + 0 + 0.25 - 0.25 = 0.75$$

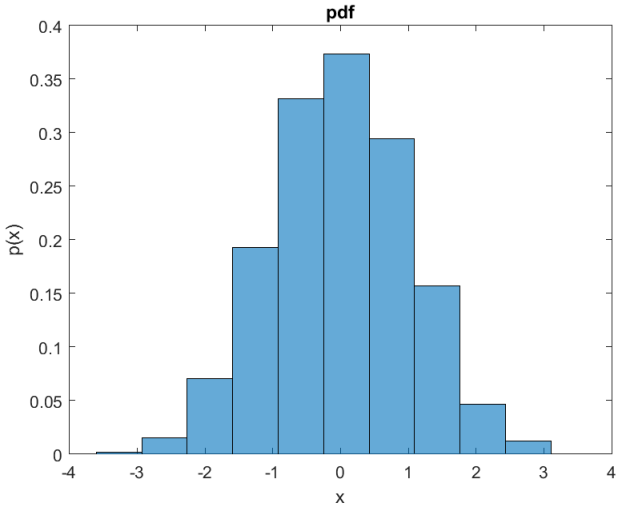
$$\therefore std(RP3) = 0.866$$

Hence the theoretical results match the graphs for RP3 as well.

1.3 Estimation of probability distributions

1.3.1 PDF function of a Gaussian distribution

Figure 10 below shows the probability distribution estimate for the zero mean and unit variance normal distribution, which is created by using the following histogram based function (with number of bins = 10) that normalises the values such that the area underneath the pdf is one, hence allowing us to express the frequency as probability of X=x occurring. The MATLAB function is as follows:




```
function v=pdf(x, bins)
    histogram (x, bins, 'Normalization', 'pdf')
    xlabel ('x')
    ylabel ('p(x)')
    title('pdf')
end
```

Figure 10 – Gaussian pdf estimate

1.3.2 PDF of RP3

From the previous analysis, RP3 is the only process that is both ergodic and stationary. Using the above PDF generating function, we obtained the figure on the right. As the graph indicates, the greater the value of N, the better the PDF approaches the theoretical distribution, which is $p(x) = \frac{1}{3}$, $-1 \leq x \leq 2$. Note that for this case the number of bins has been set to 100, which is relatively high. As discussed before, the greater the number of the bins the worse the probability distribution estimate looks. This is not the case here, which really shows the importance of law of large numbers (having a larger N).

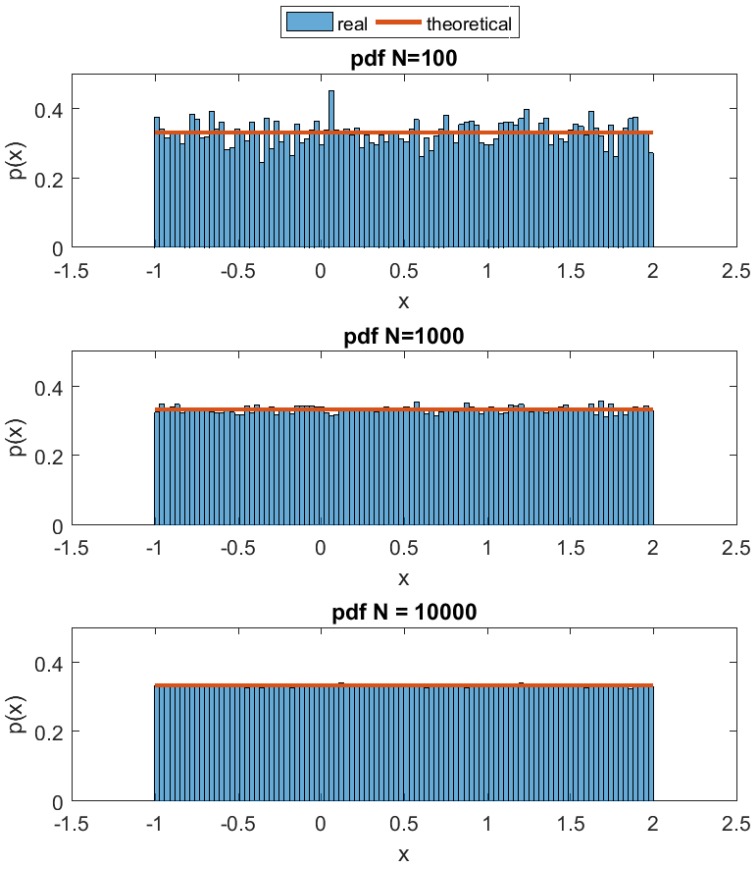


Figure 11 – RP3 pdf estimate

1.3.3 Estimation of non-stationary processes

Non-stationary processes have time-varying distributions (mean and variance change over time), and we can see that the above method for estimating the PDF is not good anymore. For instance, the performance of histogram as estimate for the true probability distribution is relies heavily on the law of large numbers, but this can be made obsolete if stationarity is not present anymore. In this particular scenario, the true PDF for N = 1:500 is a uniform distribution $f1 \sim U(-0.5, 0.5)$ with mean 0, while the true PDF for N = 501:1000 is a uniform distribution $f2 \sim U(0.5, 1.5)$, which has different means but the same variance over time (see figure 12), which means that it is not stationary.

In conclusion, unless it is known beforehand that there is a change in statistic at some given time, it is impossible to divide the process in multiple segments and find the corresponding PDF. In most real-life applications, we often will not know when the change in statistics will occur, hence making the above method obsolete.

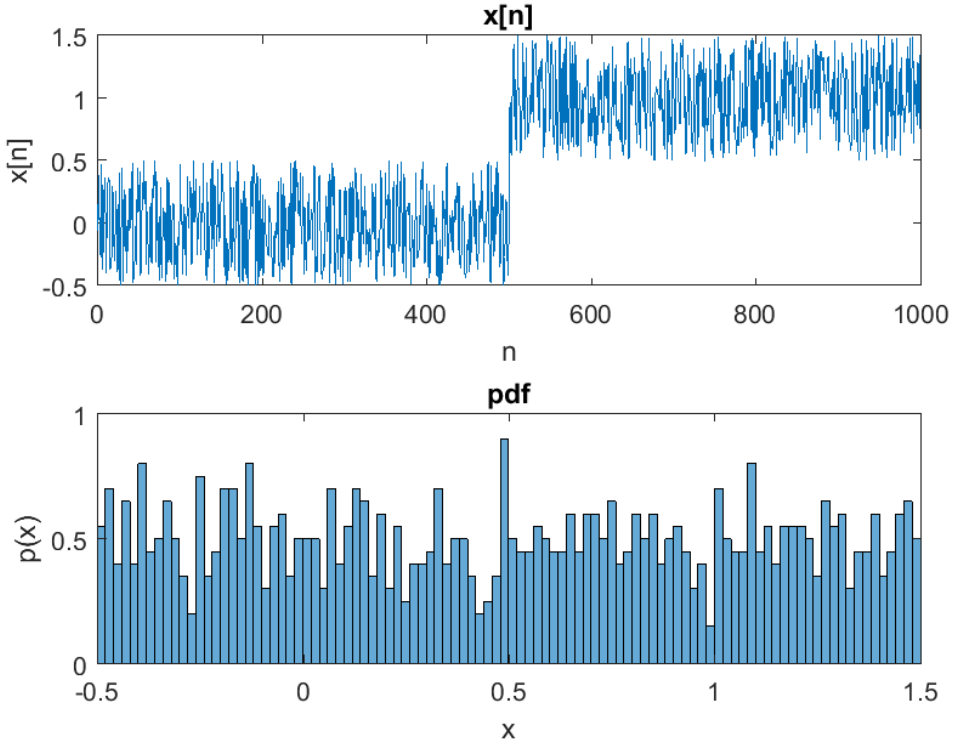


Figure 12 – non-stationary PDE

The PDE here is generated with the same method as in section 1.3.1, but with the following MATLAB code that aims to change the mean of the signal from 0 to 1 after N=500:

```
f1 = rand(500, 1)-0.5;
f2 = rand(500, 1)+0.5;
funct(1:500) = f1;
funct(501:1000) = f2;
subplot(2,1,1);
plot(funct);
xlabel('n')
ylabel('x[n]')
title('x[n]')
subplot(2,1,2)
pdf(funct, 100);
title('pdf')
```

(2) Linear stochastic modelling

2.1 ACF of uncorrelated and correlated sequences

2.1.1 ACF of WGN

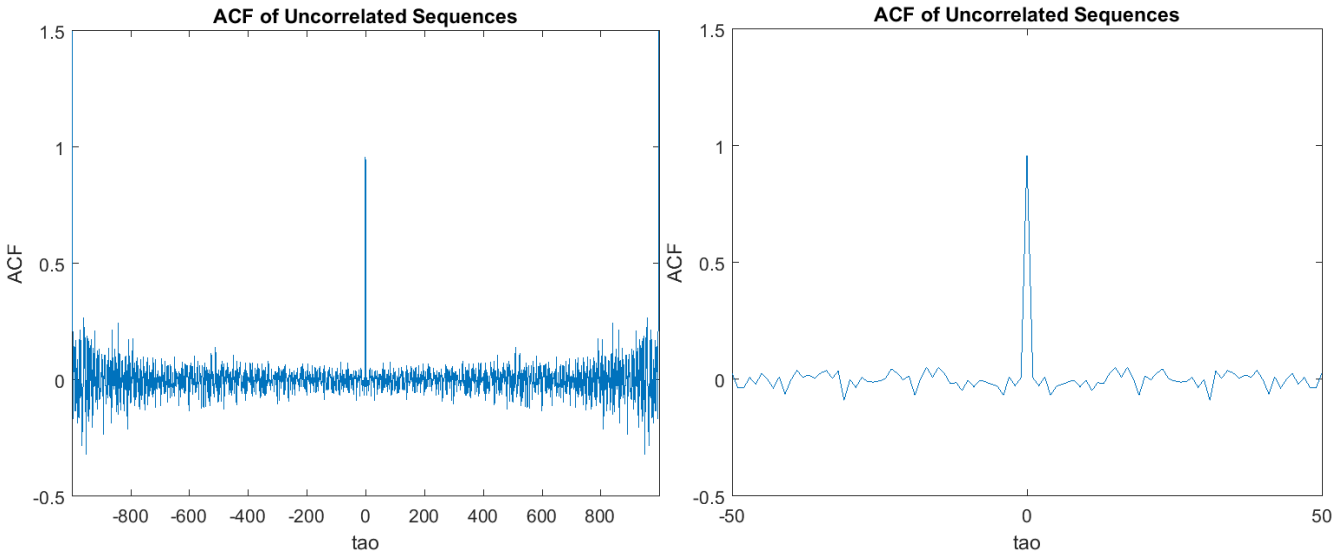


Figure 13 – uncorrelated ACF

The above figures show the unbiased estimate of the autocorrelation function, where the original signal has 1000 samples (N) and the ACF has 1999 samples (2N-1):

$$ACF = \frac{1}{N-|\tau|}\sum_{N=0}^{N-|\tau|-1} x[n]x[n+\tau], \quad \tau = -N+1, \dots, N-1$$

The ACF figure is centred around the origin with $\tau \in [-999, 999]$. The ACF has a symmetrical shape with a strong peak around the origin and much smaller value everywhere else. The signal is composed of white Gaussian noise, which means the signal is purely random, hence implying that the signal is correlated to itself only if there is no lag at all (τ=0), which explains the peak around the origin. The ACF is a symmetric function because the amplitude is proportional to the correlation of the signal to itself, and it decreases as the lag increases both in the positive and negative direction, producing same values at equal distances from the origin.

2.1.2 The value of τ

The signal increases as the lag increases towards the extremities, which should not happen since the input signal is purely random. However, ACF for larger lag values is less significant since the original signal is finite in length. This means that at increasing $|\tau|$ values, less samples of the original signal will be used to compute ACF, hence less reliable as result (for lags greater than N/2 = 500, less than half of the function is used to compute the ACF). Hence, the zoomed in version of the function provided above for τ = 50 is a more reliable description of the signal, with a strong peak at the origin and near zero values everywhere else.

2.1.3 Empirical bound on τ

As discussed previously, the larger the autocorrelation lag the less accurate the ACF. As a rule of thumb, ACF can be considered reliable for lags up to half of the number of samples, which is in this case τ = 500. However, depending on the tolerance for the probability of the ACF being correct, it is possible to fix an empirical bound on τ using the Hoeffding's inequality: $P(|error| > \varepsilon) \leq 2e^{-2\varepsilon^2 n}$. In this case, n is the number of samples being used to compute the ACF (from which we can derive the value of τ) and ε is the tolerance for which the computed ACF differs from the actual ACF (the error).

2.1.4 ACF of Moving Averages

The filtered ACF(x) on the top left is given by computing the ACF of y=filter(ones(9,1),[1],x), which means that the signal is being filtered with unit coefficients of order 9. The above function is also symmetrical and it follows a wide triangular shape, which suggests that the function is more correlated to itself than the original signal function for a wider

range of lags. This is expected as the moving average is taking into consideration multiple samples and computing an average cumulatively as it moves through the signal.

Therefore, the filtered signal is in a way making the successive samples more correlated to each other by averaging them. This is effect can be shown by using a filter of smaller order (order 3 for the top right figure). In this case, the filter takes into consideration less samples, leading a narrower triangular shape. The sample mean can be calculated in this way by using the filter that averages all the samples in the original signal.

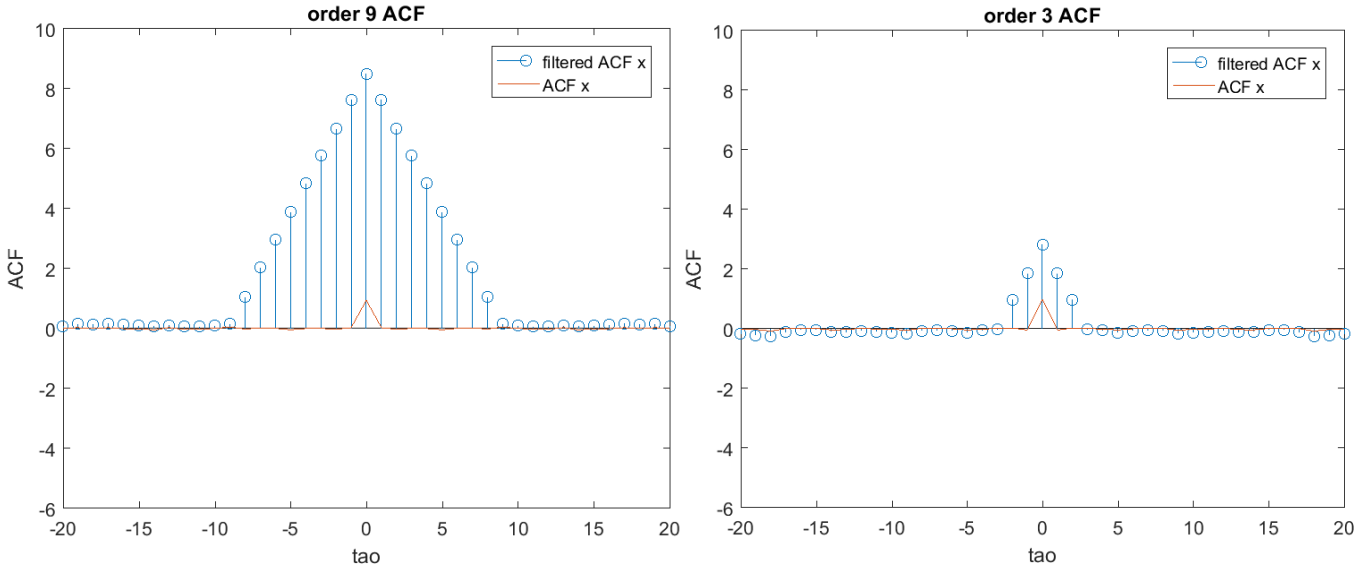


Figure 14 – correlated ACF

2.1.5 Theoretical Aspect of the Filtered ACF

$R_X(\tau)$ is the autocorrelation of the input, which is a Dirac function since it is equal to itself when there is no lag. The Dirac function has the property that when convoluted with another function, the result is the other function itself, which means $R_Y(\tau) = R_X(\tau) * R_h(\tau) = R_h(\tau)$, which is the autocorrelation of the impulse response of the filter

2.2 Cross-correlation function

2.2.1 CCF of the original and filtered signals

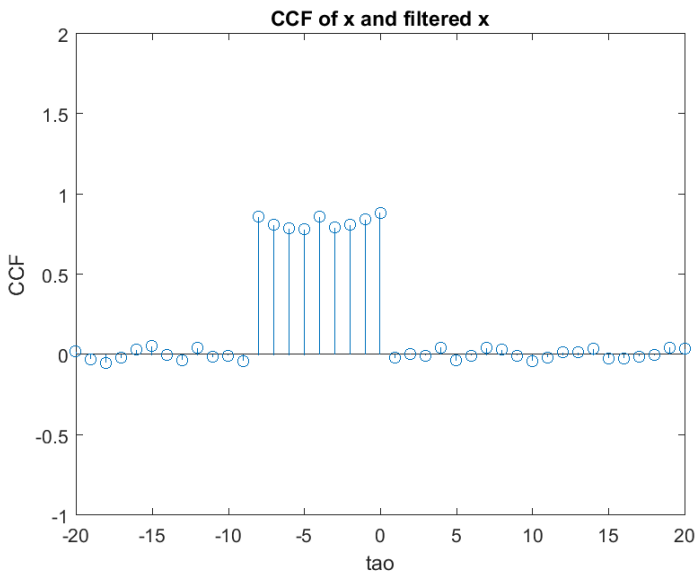


Figure 15 - CCF

The figure on the left shows that the CCF is an imperfect rectangular function (imperfect due to signal's finite length) ranging from -8 to 0 (9 samples). To explain the shape, we can refer to the equation:

$R_{XY}(\tau) = h(\tau) * R_X(\tau)$.

For similar reasons argued before, $R_X(\tau)$ is a Dirac function, which has the property that when convoluted with another function, the result is the other function itself. In this case $R_{XY}(\tau) = h(\tau)$ which is the impulse response of the filter. For this reason, it is expected that the CCF is a rectangular function since the autocorrelation of the rectangular function with width T = 9 is the triangular function with width 2T=18 (because the filter order is 9). Note that the function is correlated for negative τ, but if `xcorr(y, x, 'unbiased')` is computed instead it would be correlated for positive τ.

2.2.2 System identification

For similar reasons discussed above, given a Dirac function as the autocorrelation of a signal, the cross-correlation function can yield the impulse response of the filter, which can be used for system identification. The order of the filter that generated y can be found by counting the numbers of the Dirac functions we obtain. For instance, in the above example, a filter of order nine produced a rectangular wave composed with 9 Dirac functions (9 non zero samples).

2.3 Autoregressive modelling

2.3.1 AR2 Stability

$$x[n] = a_1 x[n-1] + a_2 x[n-2] + w[n], \quad w[n] \sim N(0.1), \quad a_1 \in [-2.5, 2.5], \quad a_2 \in [-1.5, 1.5]$$

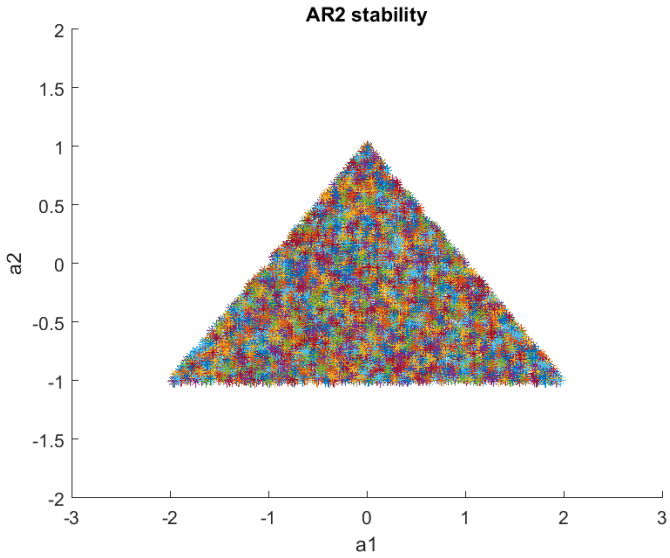


Figure 16 – stability triangle

The figure on the left plots the values of AR2 coefficients for which the autoregressive process indicated above did not diverge towards infinity over 100 iterations. It is clear that the stability region is a triangle, which confirms the theoretical boundaries defined by the following equations:

$$a_1 + a_2 < 1, \qquad a_2 - a_1 < 1, \qquad -1 < a_2 < 1$$

The above boundaries arise when solving for the transfer function of the AR process:

$$H(e^{jw}) = \frac{X(e^{jw})}{W(e^{jw})} = \frac{1}{1 - a_1 e^{-jw} - a_2 e^{-2jw}}$$

Given the Y-W equations for p=2, $a_1 = \frac{p_1(1-p_2)}{1-{p_1}^2}$, $a_2 = \frac{p_2-{p_1}^2}{1-{p_1}^2}$. Because $p_1 < p(0) = 1$, this produces a stability condition on a_1 and a_2 when solving for the denominator of the variance: $\sigma_x^2 = \left(\frac{1-a_1}{1+a_2}\right)\frac{\sigma_w^2}{(1-a_2)^2-a_1^2}$.

2.3.2 Sunspot ACF

The autocorrelation function often mimics the original signal. For this reason, if the original signal is periodic, so will the autocorrelation. However, we can see from the graphs for N = 5 that this is not true because there is not enough data that shows the periodicity nature of sunspot series. However, as we increase N, the ACF is becoming clearly more periodic. It is important to keep in mind that the ACF is increasingly inaccurate for greater values of the lag. For this reason, the figure below shows the zoomed in version of the zero mean N=250 ACF, which shows much nicer characteristics. For instance, it has the shape of an ACF of an AR2 process in the region 4 of the stability triangle. It is important to note that the following characteristics is only demonstrated with the zero-mean ACF, which contains much more information than their non-centred counterparts.

The zero mean ACF graphs differ from the original ones in other aspects as well. For instance, they are centred around the origin and have different comparative values in terms of peaks. Most notably, for N = 20, the zero mean version has the highest peak at the origin, which is expected since the signal should be most similar to itself at zero lag. This however is not the case for the non-zero mean ACF, which has successive peaks higher than the one at the origin.

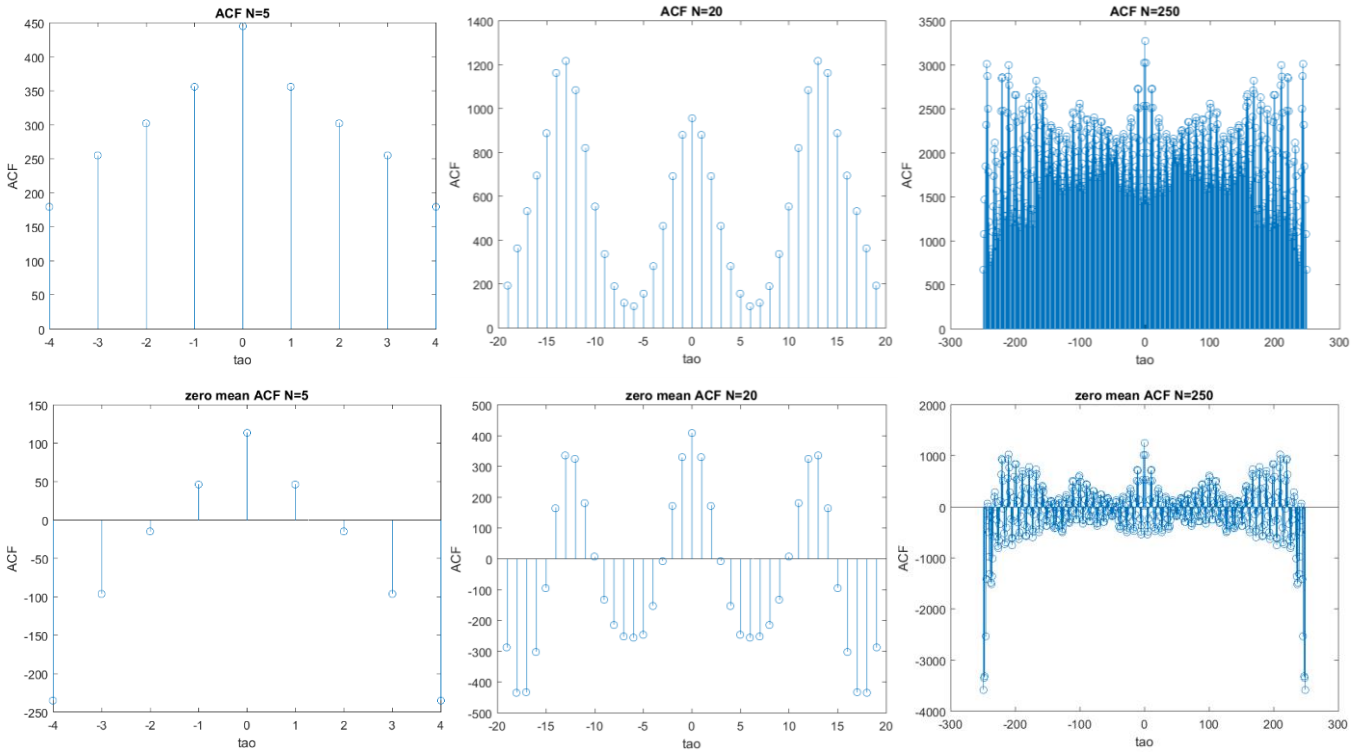


Figure 17 – Sunspot ACF

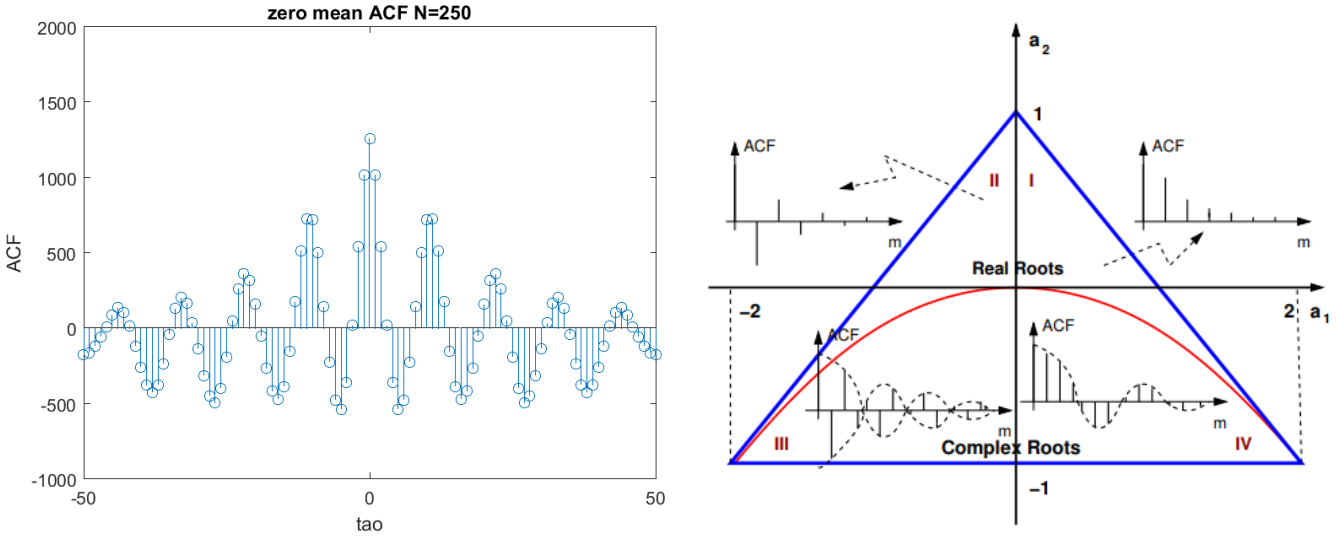


Figure 18 – Sunspot ACF shape

2.3.3 Yule-Walker Equations

Figure 19 show the partial ACF calculated using the Yule Walker equations both for the original and the normalised version of the sunspot series. Although there are slight variations in the correlation value, the normalised version should be the more reliable one as the biases due to its mean are removed. However, both the normalised and original series show that the correlation decreases rapidly after lag = 2, and it is negligibly small after lag = 3. For this reason, the sunspot series is most likely an AR2 process, although AR3 is also a plausible possibility.

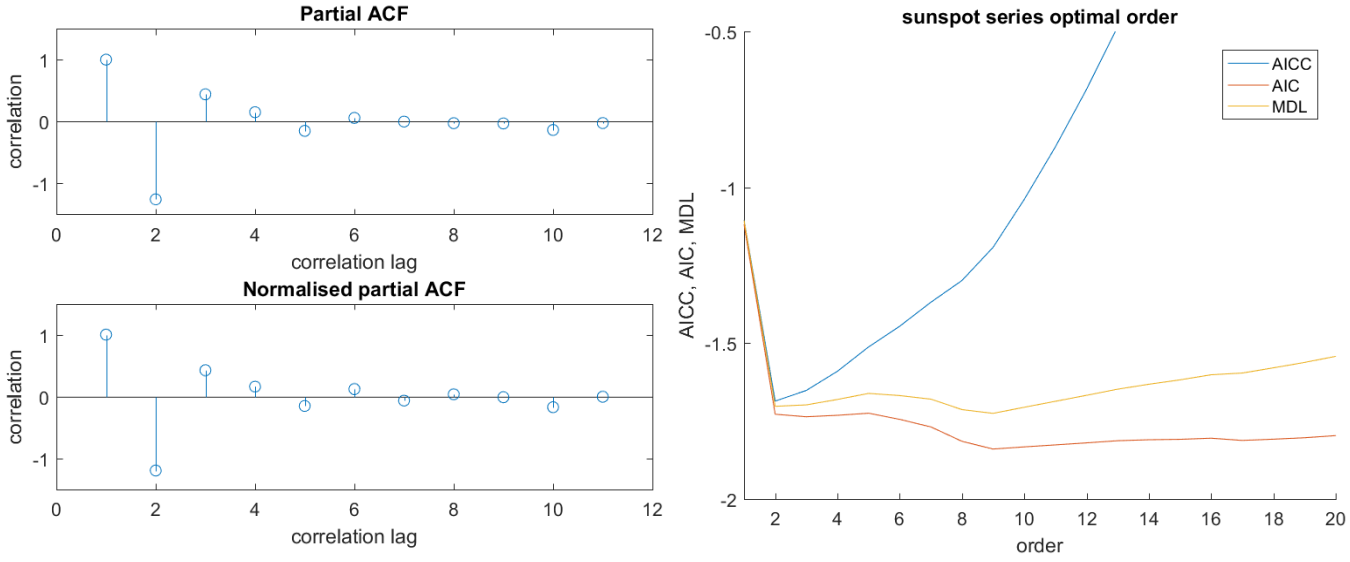


Figure 19 – Sunspot order

2.3.4 MDL and AIC for sunspot series

The figures below present the Minimum Description Length and the (Corrected) Akaike Information Criterion. Usually, given a polynomial of high enough order, it is possible to describe any desired functions. For this reason, during the in-sample training the higher order models are the ones that produce the smallest in-sample error. However, the higher the order, the more overfitting there will be, and hence we need to consider additional penalty term such as the ones in this example.

The AIC and MDL show that there are 2 minima worth considering around order =2 and order = 9. However, the corrected AIC (better reduces possibility of overfitting for small sample sizes) suggests that modelling sunspot series as AR2 is the safest choice, which confirms the previous argument using partial ACF.

2.3.5 AR modelling of sunspot series

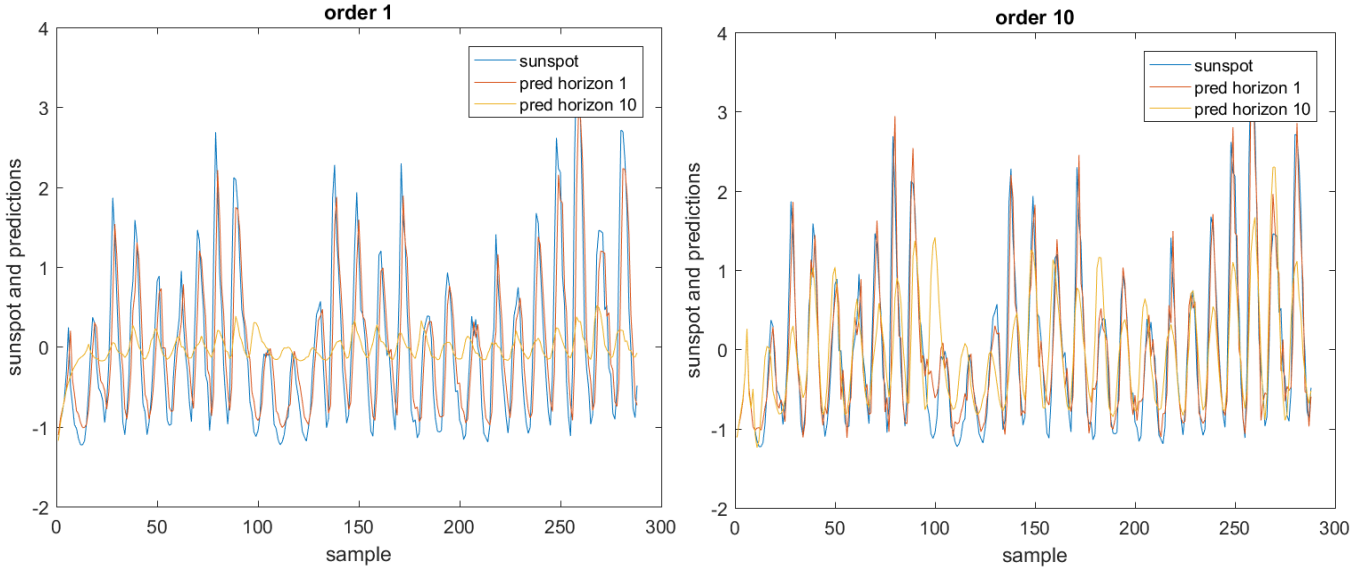


Figure 20 – Sunspot prediction

	horizon = 1	horizon = 2	horizon = 5	horizon = 10
order 1	0.43	0.72	0.99	0.72
order 2	0.31	0.50	0.69	0.72
order 10	0.27	0.40	0.48	0.50

Table 2 – Sunspot mean prediction error

The plots above show the performance of the smallest and biggest order model when predicting 1 or 10 steps ahead. The table summarises the performance for various model using the mean difference between the prediction and the actual values. In general, it seems that the greater the model order, the better it will be at predicting for greater prediction horizons. For instance, all the models considered performed well for pred-horizon = 1. However, as the models tried to predict 2 steps ahead, the order 1 model's performance degraded tremendously compared to the higher models. This trend seems to continue to apply for the higher order models and higher prediction horizons. Overall, the order 10 model performed the best for all prediction horizons, producing the smallest prediction error.

In terms of trade-off, one of the disadvantages of higher order models is their higher complexity. However, the most important aspect is that when fitting the model incredibly well for the given data and hence reducing the model error (interpolation), the overfitting of the model is likely to lead to bad results when generalizing outside the sample data, which increases the prediction error (extrapolation).

2.4 Cramer-Rao Lower Bound

2.4.1.a AR1 Modelling – NASDAQ

The right plot below shows that the AR1 predictions produces a lagged version of the real NASDAQ series. The partial ACF shows that the correlation up until lag = 2 is significant. However, considering the penalty of overfitting from the corrected AIC, the safest modelling order would be of order 1, minimising both AIC and MDL, which are non-decreasing functions of the order. Therefore, AR1 model is sufficient to describe the daily returns of the index.

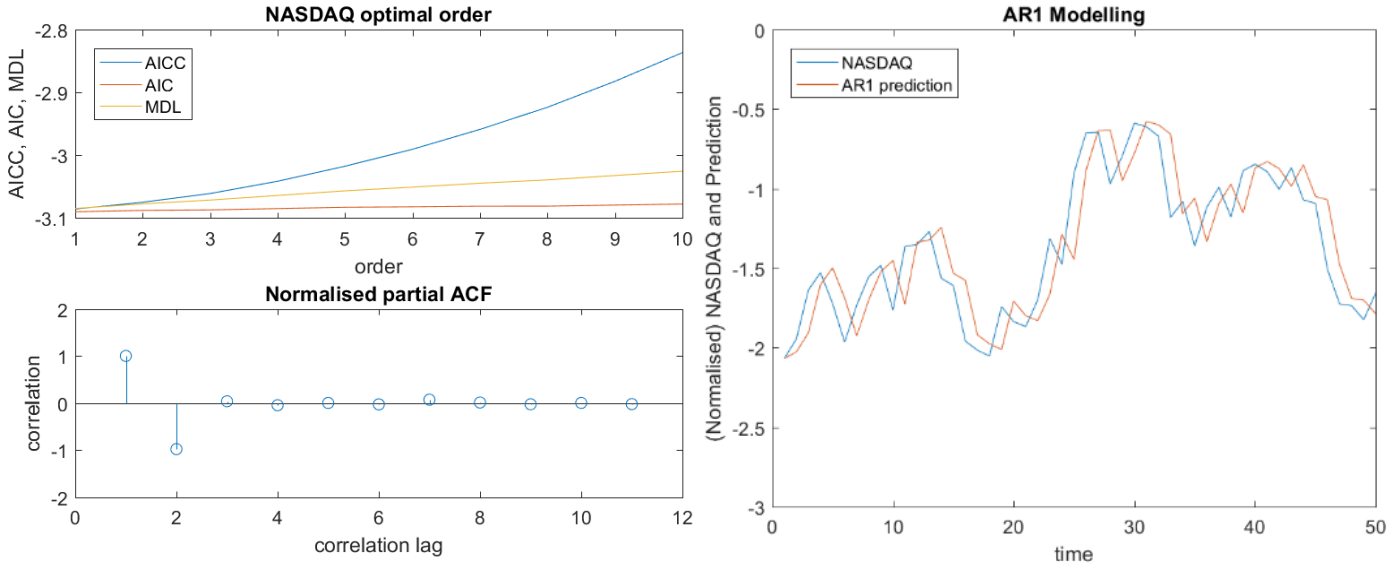


Figure 21 – NASDAQ AR modelling

2.4.1.b Fisher Information Matrix

For p = 1, $\ln\left[\hat{P}_X(f;\boldsymbol{\theta})\right]$ can be simplified as:

$$\ln\left[\hat{P}_X(f;\boldsymbol{\theta})\right] = \ln[\hat{\sigma}^2] - \ln\left[1 - \hat{a}_1 e^{-j2\pi f}\right] - \ln\left[1 - \hat{a}_1 e^{j2\pi f}\right]$$

In this specific case, $\boldsymbol{\theta} = [a_1, \sigma^2]$, where $\theta_1 = a_1$ and $\theta_2 = \sigma^2$. From this follows:

$$\left(\frac{\partial \ln\left[\hat{P}_X(f;\boldsymbol{\theta})\right]}{\partial \sigma^2}\right) = \frac{\partial}{\partial \sigma^2}\left(\ln[\hat{\sigma}^2] - \ln\left[1 - \hat{a}_1 e^{-j2\pi f}\right] - \ln\left[1 - \hat{a}_1 e^{j2\pi f}\right]\right) = \frac{1}{\sigma^2}$$

It follows that the equation $[\boldsymbol{I}(\boldsymbol{\theta})]_{ij} = \frac{N}{2}\int_{-0.5}^{0.5} \frac{\partial \ln[\hat{P}_X(f;\boldsymbol{\theta})]}{\partial \theta_i} \frac{\partial \ln[\hat{P}_X(f;\boldsymbol{\theta})]}{\partial \theta_j} df$ becomes:

$$[\boldsymbol{I}(\boldsymbol{\theta})]_{22} = \frac{N}{2}\int_{-0.5}^{0.5} \frac{\partial \ln[\hat{P}_X(f;\boldsymbol{\theta})]}{\partial\theta_2}\frac{\partial \ln[\hat{P}_X(f;\boldsymbol{\theta})]}{\partial\theta_2} df = \frac{N}{2}\int_{-0.5}^{0.5}\left(\frac{\partial \ln[\hat{P}_X(f;\boldsymbol{\theta})]}{\partial\sigma^2}\right)^2 df = \frac{N}{2\sigma^4}\int_{-0.5}^{0.5} 1\, df = \frac{N}{2\sigma^4}$$

Therefore, given $[\boldsymbol{I}(\boldsymbol{\theta})]_{11} = \frac{N\boldsymbol{r}_{xx}(0)}{\sigma^2}$, $[\boldsymbol{I}(\boldsymbol{\theta})]_{12} = [\boldsymbol{I}(\boldsymbol{\theta})]_{21} = 0$, the Fisher information matrix is approximately:

$$\boldsymbol{I}(\boldsymbol{\theta}) = \begin{bmatrix} \frac{N\boldsymbol{r}_{xx}(0)}{\sigma^2} & 0 \\ 0 & \frac{N}{2\sigma^4} \end{bmatrix}$$

2.4.1.c.i Variance inequalities

Given:

$$var(\hat{\theta}_i) \geq [\boldsymbol{I}^{-1}(\boldsymbol{\theta})]_{ii} \quad => \quad var(\hat{\theta}_2) = var(\hat{\sigma}^2) \geq [\boldsymbol{I}^{-1}(\boldsymbol{\theta})]_{22} = \frac{1}{\frac{N}{2\sigma^4}} \quad \therefore \quad var(\hat{\sigma}^2) \geq \frac{2\sigma^4}{N}$$

Similarly:

$$var(\hat{\theta}_1) = var(\hat{a}_1) \geq [\boldsymbol{I}^{-1}(\boldsymbol{\theta})]_{11} = \frac{1}{\frac{N\boldsymbol{r}_{xx}(0)}{\sigma^2}} = \frac{1}{N}\frac{\sigma^2}{\boldsymbol{r}_{xx}(0)} = \frac{1}{N}\frac{E[X^2[n]] - E^2[X[n]]}{E[X^2[n]]} = \frac{1}{N}\left(1 - \frac{E^2[X[n]]}{E[X^2[n]]}\right)$$

$$\therefore \quad var(\hat{a}_1) \geq \frac{1}{N}(1 - a_1^2)$$

Figure 22 shows the CRLB for both cases mentioned above (note that in this example, $\boldsymbol{r}_{xx}(0)$ is computed using normalised NASDAQ series). As it is expected, the bound increase for smaller number of data points as it is hard to estimate the true characteristics of a signal if there is a small dataset. The CRLB is therefore proportional to $\frac{1}{N}$ for both cases. The variance of the driving noise however affects $var(\hat{\sigma}^2)$ much more than it affects the $var(\hat{a}_1)$. This is also expected from the above equations since $var(\hat{\sigma}^2)$ is proportional to the square of the driving noise variance, while it is only proportional to it for $\hat{a}_1$

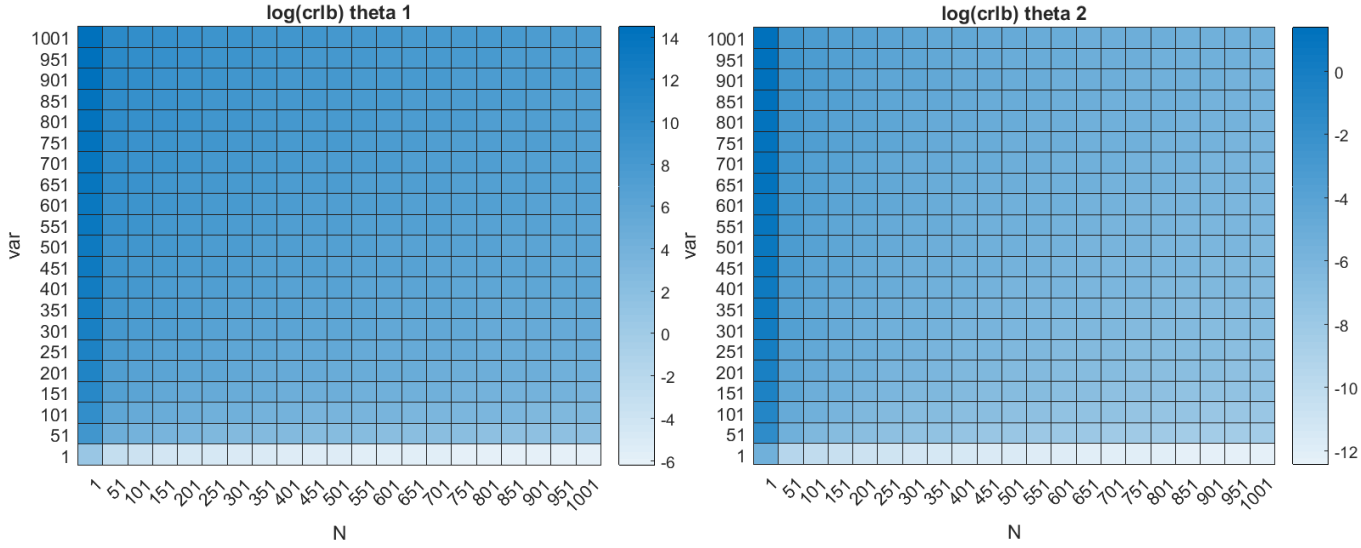


Figure 22 – CRLB

2.4.1.c.ii Financial dataset parameter a1

By following the equations derived previously $\frac{1}{N}(1 - a_1^2) = \frac{1}{N}\left(1 - \frac{E^2[X[n]]}{E[X^2[n]]}\right)$, it follows that $a_1 = \sqrt{\frac{E^2[X[n]]}{E[X^2[n]]}}$. Using MATLAB to calculate all the terms in the equation, it follows that $a_1 = 0.9958$ for the original NASDAQ series, while $a_1 =$ 1.39e-15 for the normalised version. As a_1 approaches unity, the bound for $var(\hat{a}_1)$ will approach to zero as $(1 - a_1^2) \to 0$ for $a_1 \to 1$. In terms of transfer function, as the pole of the characteristic function approaches the unit

circle, it becomes increasingly oscillatory and unstable. For instance, if the poles were outside of the unit circle, the impulse response would grow exponentially. On a physical point of view, this can be understood simply by looking at a^t, which is stable and converging as t grows to infinity only if |a|<1. This explains the much smaller value of a_1 for the normalised series since after de-trending the NASDAQ, there is no growth over time.

2.4.1.d CRLB proof

Given $A(f) = 1 - \hat{a}_1 e^{-j2\pi f}$ and the following power spectrum equation for an AR(p) model:

$$\hat{P}_X(f;\boldsymbol{\theta}) = \frac{\hat{\sigma}^2}{\left|1 - \sum_{m=1}^{p} \hat{a}_m e^{-j2\pi f m}\right|^2}, \quad p = 1 \rightarrow \hat{P}_X(f;\boldsymbol{\theta}) = \frac{\hat{\sigma}^2}{|1 - \hat{a}_1 e^{-j2\pi f}|^2} = \frac{\hat{\sigma}^2}{(1 - \hat{a}_1 e^{-j2\pi f})(1 - \hat{a}_1 e^{j2\pi f})}$$

It follows that:

$$\left(\frac{\partial \hat{P}_X(f;\boldsymbol{\theta})}{\partial a_1}\right) = \frac{\partial}{\partial a_1}\left(\frac{\hat{\sigma}^2}{|1 - \hat{a}_1 e^{-j2\pi f}|^2}\right) = (\hat{\sigma}^2)\frac{\partial}{\partial a_1}\left(\frac{1}{(1 - \hat{a}_1 e^{-j2\pi f})}\frac{1}{(1 - \hat{a}_1 e^{j2\pi f})}\right)$$

$$= (\hat{\sigma}^2)\left(\left(\frac{e^{-j2\pi f}}{(1 - \hat{a}_1 e^{-j2\pi f})^2}\right)\left(\frac{1}{(1 - \hat{a}_1 e^{j2\pi f})}\right) + \left(\frac{1}{(1 - \hat{a}_1 e^{-j2\pi f})}\right)\left(\frac{e^{j2\pi f}}{(1 - \hat{a}_1 e^{j2\pi f})^2}\right)\right)$$

$$= (\hat{\sigma}^2)\left(\frac{e^{-j2\pi f}}{(A(f))^2 A(-f)} + \frac{e^{j2\pi f}}{(A(-f))^2 A(f)}\right) = (\hat{\sigma}^2)\left(\frac{1 - A(f)}{\hat{a}_1 (A(f))^2 A(-f)} + \frac{1 - A(-f)}{\hat{a}_1 (A(-f))^2 A(f)}\right)$$

$$= (\hat{\sigma}^2)\left(\frac{A(-f) - A(f)A(-f) + A(f) - A(-f)A(f)}{\hat{a}_1 (A(f)A(-f))^2}\right) = (\hat{\sigma}^2)\left(\frac{A(f) + A(-f) - 2|A(f)|^2}{\hat{a}_1 |A(f)|^4}\right) = t_1$$

Similarly:

$$\left(\frac{\partial \hat{P}_X(f;\boldsymbol{\theta})}{\partial \sigma^2}\right) = \frac{\partial}{\partial \sigma^2}\left(\frac{\hat{\sigma}^2}{|1 - \hat{a}_1 e^{-j2\pi f}|^2}\right) = \frac{1}{|1 - \hat{a}_1 e^{-j2\pi f}|^2} = \frac{1}{A(f)A(-f)} = \frac{1}{|A(f)|^2} = t_2$$

Leading to:

$$\frac{\partial \hat{P}_X(f;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \left[\frac{\partial \hat{P}_X(f;\boldsymbol{\theta})}{\partial \theta_1}, \frac{\partial \hat{P}_X(f;\boldsymbol{\theta})}{\partial \theta_2}\right]^T = [t_1, t_2]^T$$

Therefore, from the equations given in the ASP coursework, the bound can be derived in the following way:

$$var\left(\hat{P}_X(f;\boldsymbol{\theta})\right) \geq \frac{\partial \hat{P}_X(f;\boldsymbol{\theta})^T}{\partial \boldsymbol{\theta}} \boldsymbol{I}^{-1}(\boldsymbol{\theta}) \frac{\partial \hat{P}_X(f;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = [t_1 \;\; t_2] \begin{bmatrix} \frac{N\boldsymbol{r}_{xx}(0)}{\sigma^2} & 0 \\ 0 & \frac{N}{2\sigma^4} \end{bmatrix}^{-1} [t_1 \;\; t_2]^T$$

$$= [t_1 \;\; t_2] \begin{bmatrix} \frac{\sigma^2}{N\boldsymbol{r}_{xx}(0)} & 0 \\ 0 & \frac{2\sigma^4}{N} \end{bmatrix} [t_1 \;\; t_2]^T = \begin{bmatrix} \frac{t_1 \sigma^2}{N\boldsymbol{r}_{xx}(0)} & \frac{t_2 2\sigma^4}{N} \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \frac{{t_1}^2 \sigma^2}{N\boldsymbol{r}_{xx}(0)} + \frac{2{t_2}^2 \sigma^4}{N} \end{bmatrix}$$

$$\therefore var\left(\hat{P}_X(f;\boldsymbol{\theta})\right) \geq \frac{(A(f) + A(-f) - 2|A(f)|^2)^2 \sigma^6}{{\hat{a}_1}^2 |A(f)|^8 N\boldsymbol{r}_{xx}(0)} + \frac{2\sigma^4}{|A(f)|^4 N}$$

2.5 Real world signals: ECG from iAmp experiment

2.5.a Heart rate PDE

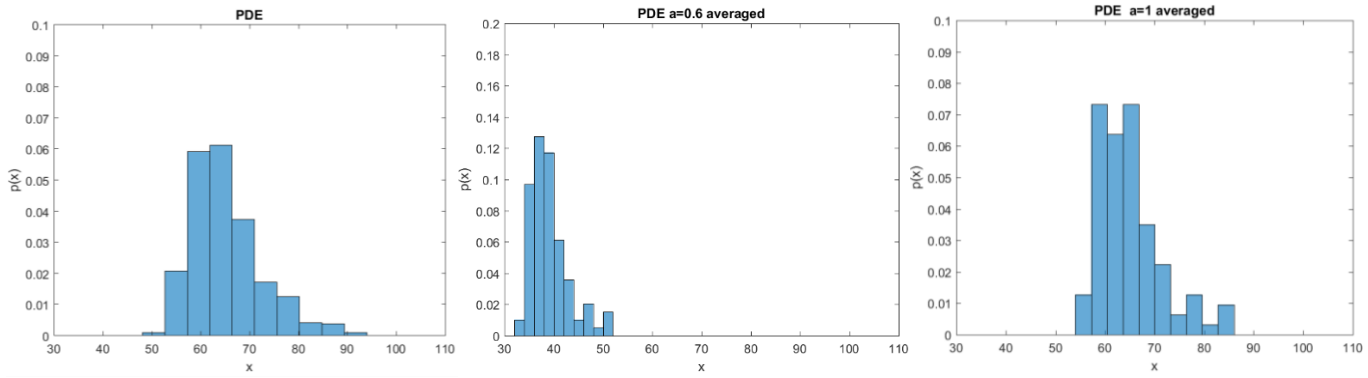


Figure 23 – heart rate PDE

The above figures show the probability density estimate during 2nd trial where the breathing rate is constrained at 50 beats per minute.

2.5.b The effect of the constant *a*

The left figure roughly follows a skewed Gaussian distribution with a peak around 65. However, when taking the average of the breathing rate according to

$$\hat{h}[1] = \frac{1}{10}\sum_{t=1}^{10} ah[i], \quad \hat{h}[2] = \frac{1}{10}\sum_{t=11}^{20} ah[i], \quad \ldots$$

The PDE graph is shifted by a factor of a, which is expected as we multiply the sample realisations of the heart rate by the same factor. However, another effect of the constant is that it changes the variance of the random variable, which can be noticed visually since the width of the PDE is reduced as well for $a = 0.6$. This is expected as when multiplying a random variable with standard deviation σ by a factor k, the variance of the new factor is $k^2\sigma^2$. Hence when multiplying by the factor of 0.6, it scales the random variable's variance by 0.36.

2.5.c AR modelling of heart rate

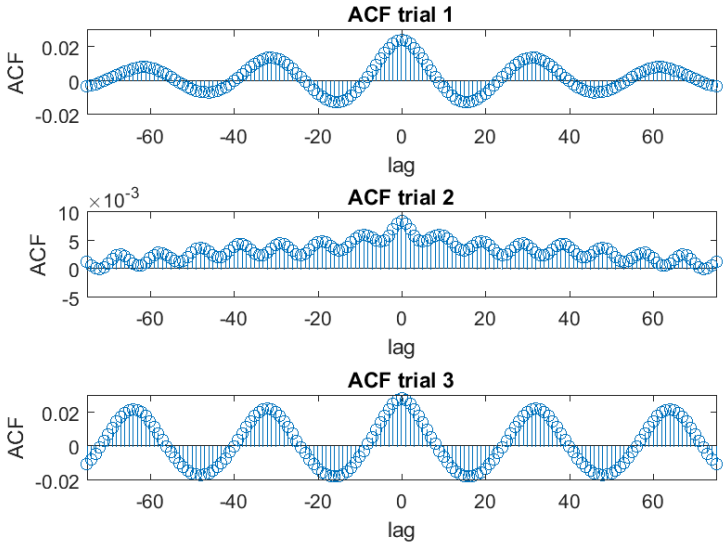


Figure 24 – heart rate ACF

The figure on the left compares the ACF of the heart rate for all three trials, which captures their periodic properties.

For trial 1, where the breathing is unconstrained, the peak at the origin reflects the random nature of the heat beats which is modulated by the random breathing rate. However, its cyclical nature due to the increasing/decreasing heart rate when breathing in/out is still notable as the ACF is periodic.

For the second trial, the breathing is constrained at an unnaturally high rate, which is reflected in the ACF with much smaller amplitude compared to the other two. This is in direct contrast with trial 3 where the breathing is prolonged and at a constant rate, making the ACF is almost perfectly periodic.

From the figure 24 it is clear that this is an AR process rather than moving average, since the ACF of MA functions behaves like figure 14, where the ACF goes quickly to zero after some lags. More specifically, for trial 1 and trial 3, the process is likely to be ar2 with positive a1 and negative a2 given the similarity in behaviour in the region IV of the stability triangle in figure 18. However, trial 2 exhibits a slightly different behaviour which in unlikely to be modelled successfully through an AR2 process.

2.5.d AR order of the heart rate

Using the same tools employed in the previous sections, we can confirm the previous conclusion that the best way to model the heart rate in trial 1 and 3 is through an order 2 model. For instance, for trial 1 and 3, the order 2 minimises AIC and MDL, as well as the corrected AIC which adds the overfitting penalty that comes with small sample number. The partial ACF also shows that the correlation is negligibly small after a lag of 3.

The only exception in this case is trial 2, where the irregularity due to the unnatural constrained breathing modulation seems to be better modelled with an order 4 AR process.

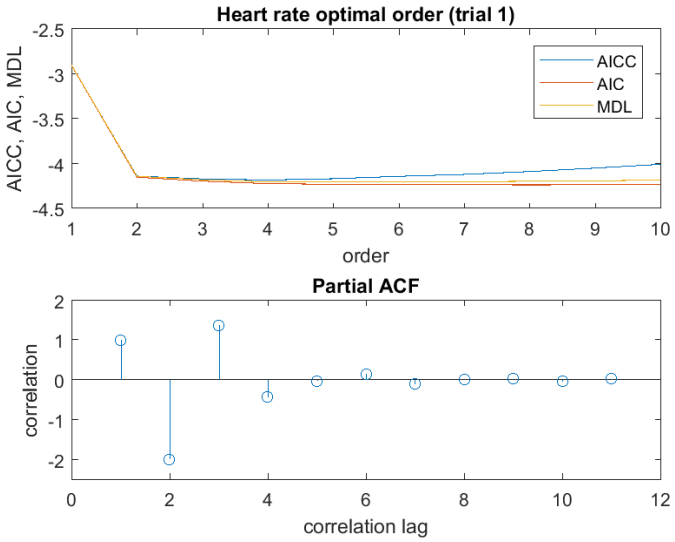


Figure 25 – t1 heart rate order

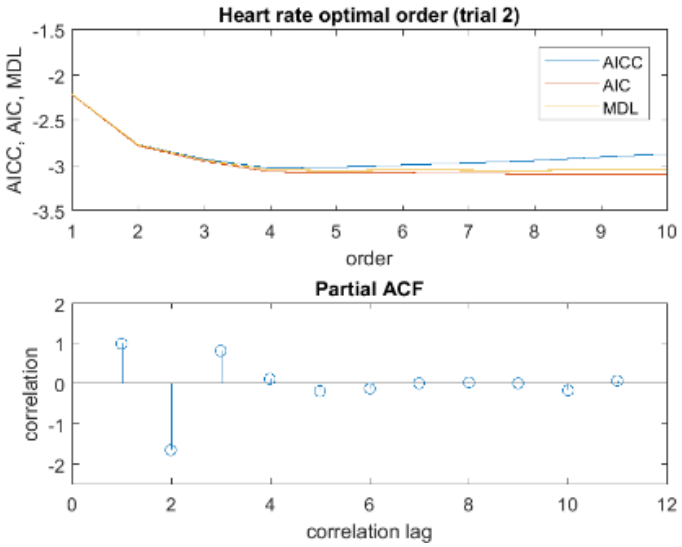


Figure 26 – t2 heart rate order

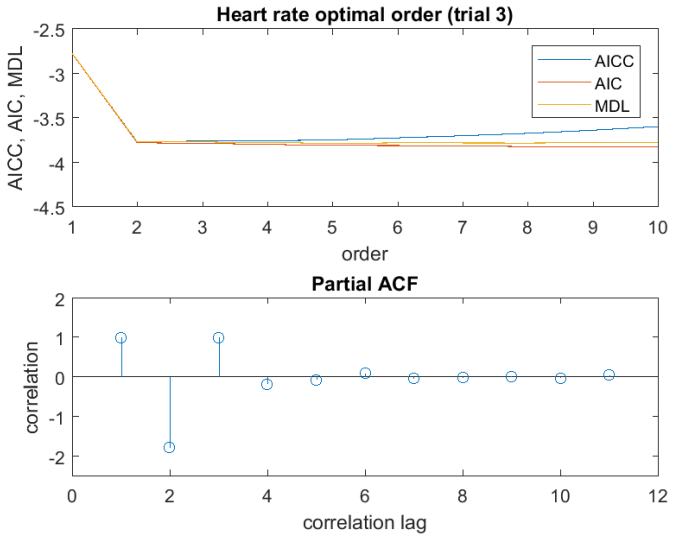


Figure 27 – t3 heart rate order

(3) Spectral estimation and modelling

PGM function

The periodogram function $\hat{P}_x(f) = \frac{1}{N}\left|\sum_{n=0}^{N-1} x[n]e^{-\frac{j2\pi fn}{N}}\right|^2$ estimates the PSD at N discrete frequencies using FFT, with normalised frequencies corresponding to f = 0, $\frac{1}{N}, \frac{2}{N}, \ldots, \frac{N-1}{N}$. The MATLAB code for the above pgm function is as follows:

```
function y = pgm(x)
    FTransf = fft(x);
    % computes the PSD
    y(:,2) = ((1/length(x)).*(abs(FTransf).^2));
    % computes the normalised f
    y(:,1) = (0:(1/length(x)):((length(x)-1)/length(x)));
    stem (y(:,1), y(:,2));
end
```

The following figure shows the pgm function applied to white Gaussian noise for 3 different values of N. Note that instead of normalised frequency in the range [0, 1] it is capped at 0.5 due to how MATLAB handles the negative frequencies using FFT. In addition, unless otherwise specified, all the f axes in this section are normalised frequencies ($x\ 2\pi\ rad/sample$), which is not in Hz.

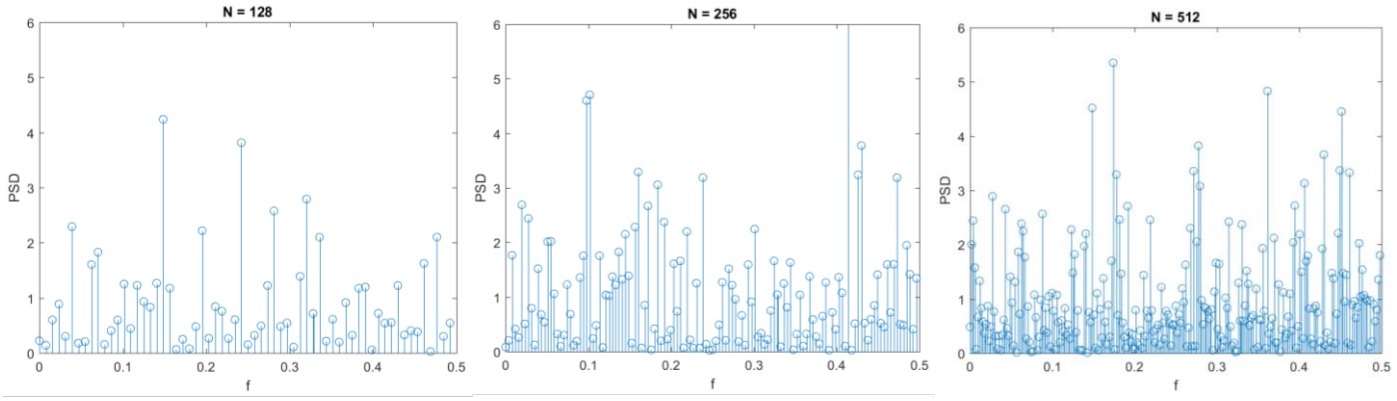


Figure 28 – wgn periodogram

The input sequence in figure 28 is the normal Gaussian, which should result in the autocorrelation function being a unit impulse. This means that its Fourier transform (hence the PSD) should be unity for all frequencies. However, the above figures do not display these characteristics, even if the number of samples increases. However, when looking at the statistics of the above results it is possible to note a few things:

	N = 128	**N = 256**	**N = 512**
mean	0.879	0.886	0.980
variance	0.966	0.931	1.181

Table 3 – pgm statistics

Firstly, the mean approaches 1 as the N increases, which is good since the theoretical PSD of the normal distribution is unity across all frequencies. This means that the above method forms an asymptotic estimate that will describe the PSD well as N tends to infinity. However, we do not observe the same behaviour in variance, which seems to fluctuate randomly for a given N.

A problem that is central to the non-ideal shape of the PSD when estimating through periodograms is that we are effectively applying a square window to the signal, which means that it will produce some frequency components that are not representative of the original signal (note that the FFT of a rectangular window produces a sinc function).

3.1 Averaged Periodogram Estimates

3.1.1 Smoothed PSD estimate

The figure on the right compares the original and frequency domain smoothed periodogram for N = 1024. It is clear that the filtered version produces a PSD much closer to the theoretical one, which oscillates about unity with less variance. This is not surprising since when filtering with a FIR filter with impulse response of 0.2*[1 1 1 1 1], the standard deviation is scaled by $0.2 * \sqrt{5}\sigma$, hence obtaining a PSD that oscillates much less (smaller variance) about unity compared to the original periodogram.

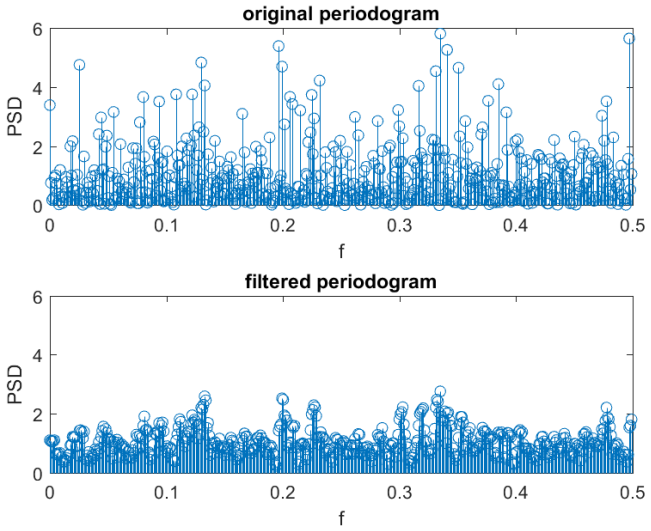


Figure 29 – filtered periodogram

3.1.2 Multiple segments

When dividing the 1024 sample sequence into non-overlapping 128 sample sequences, 8 completely random PSD were generated. This is expected since the original sequence is created using the normal distribution, by dividing it in 8 non-overlapping parts it is the equivalent of generating 8 random sequences of smaller N due to its independence property among samples.

3.1.3 Averaged periodogram

The figure on the right compares one of the 8 segments with the average of all 8 segments. Performance wise, the variance (fluctuations) is significantly reduced. This is expected as each frequency component taken into consideration is a random variable with an associated probability density function. Hence, by averaging it across 8 realisations, we produced an ensemble mean that reflects the properties of the ideal PSD better.

The more segments there are the better will the approximation be due to the law of large numbers. The main drawback however is that there is a reduction in frequency resolution, since there are less samples being considered each time given a finite sequence.

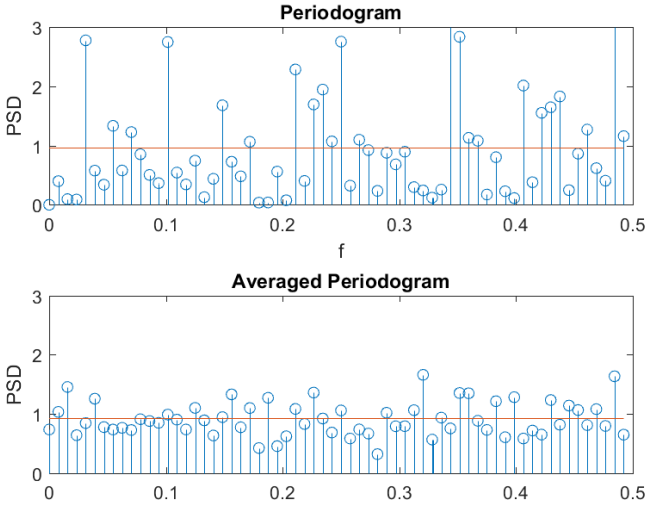


Figure 30 – averaged periodogram

3.2 Spectrum of autoregressive processes

3.2.1 Theoretical PSD

The figure on the right shows the PSD of an AR1 process after filtering the Gaussian noise using a filter with coefficients $b = 1$ and $a = [1, 0.9]$. Given that the PSD $P_Y(f) = |H(f)|^2 P_x(f) = \frac{1}{|1+0.9e^{-j2\pi f}|^2}$, the theoretical PSD has the shape that matches the plot on the right, which is a high pass filter with a cut-off frequency (normalised) around 0.45.

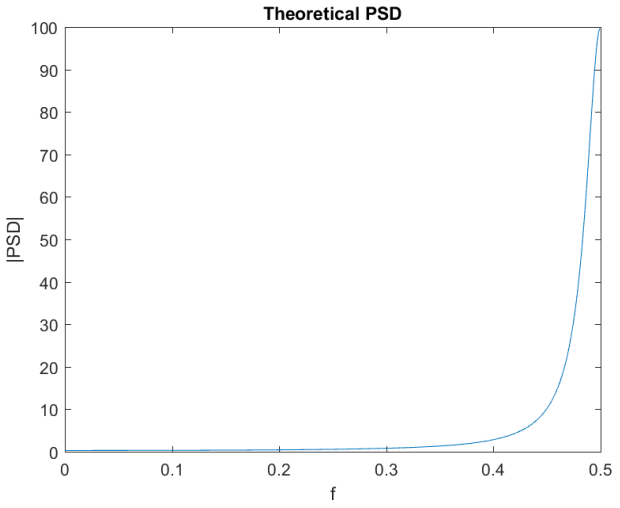


Figure 31 – theoretical PSD

3.2.2 Theoretical PSD vs pgm

The figure on the left compares the theoretical PSD and the pgm. The trend is similar in both, but the pgm has the same issues present in the previous section. For instance, the pgm estimates the PSD from a given realisation of the random variable at each frequency value, so this will create a large variance around the true value of the PSD as shown in the figure. As before, it is possible to improve the shape using FIR filters or averaging methods.

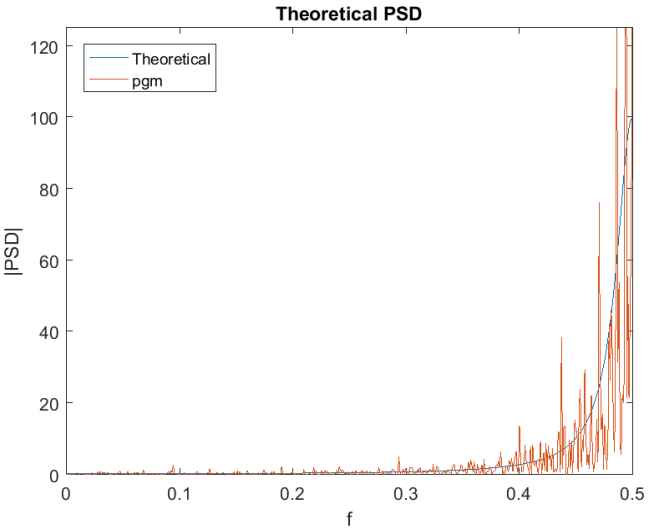


Figure 32 – theoretical PSD vs pgm

3.2.3 Theoretical PSD vs pgm – zoomed in

As mentioned in the previous section, when taking the FFT of a finite function it is as if it was passed through a rectangular filter, and the Fourier transform of which is a sinc function $sinc(x) = \frac{\sin(x)}{x}$. Hence, in addition to the variance that is present due to the pgm method, an additive error due to the sinc wave is introduced. This can be seen in the figure on the right when considering the envelop of the periodogram.

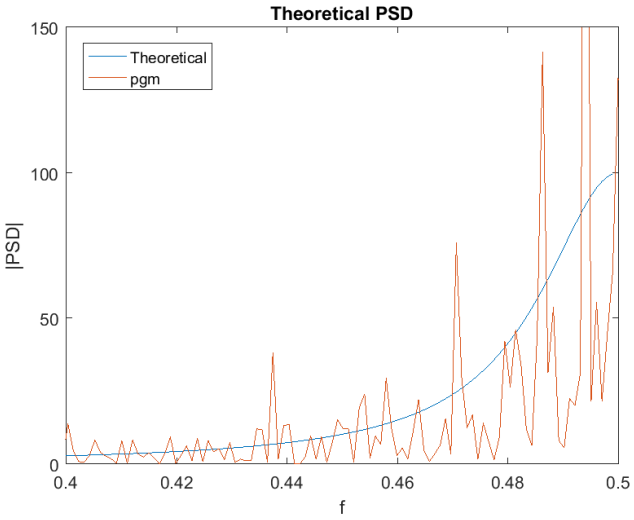


Figure 33 – theoretical PSD vs pgm

3.2.4 Model based PSD

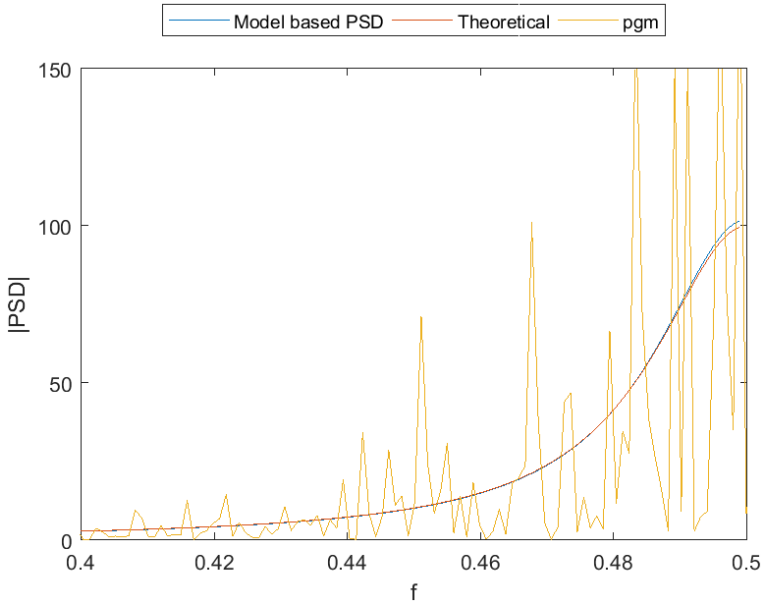


Figure 34 – model based PSD vs other methods

The figure on the left compares all the methods used so far. The model based PSD is a much better approximation of the theoretical PSD compared to the pgm, which overlaps almost entirely with the theoretical shape. This is because the model-based estimate assumes that the given data follows a certain model in the first place, hence working with the characteristics of that model (autocorrelation function) rather than the individual sample realisations like the pgm, making it a more accurate estimate. This means that usually, provided the model is correct, the model-based estimate will outperform the pgm method.

3.2.5 Model based PSD – Sunspot series

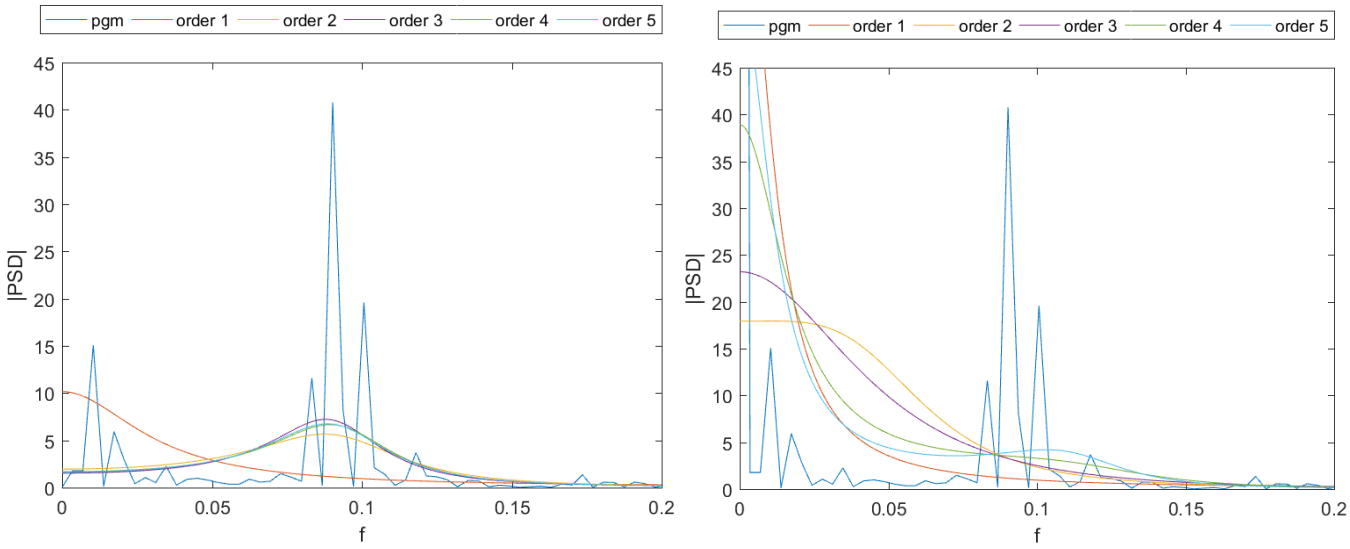


Figure 35 – sunspot PSD estimation

The above figure shows the previous methods applied to the zero-mean (left) and the original (right) sunspot series. For the zero-mean version, when under-modelling, the model based PSD is not able to capture the characteristics of the signal, hence having a peak around f = 0 unlike other models. For higher order models, the effect of over-modelling is relatively small since we produced the coefficients using the Yule Walker equations, and it is known that the coefficients produced are small in amplitude for orders greater than the actual model order, hence having almost no degrading effects on the PSD model. Finally, the pgm method's performance has the same problems discussed in the previous sections, hence performing relatively worse compared to the model based estimates.

For the figure on the right, it is important to notice that for the original sunspot series, the pgm is the only method that picks up the peak around f = 0.09 while model based estimates all have peaks around the origin. This is expected since the DC value (f=0) is not removed from the series.

3.3 Least square estimation of AR coefficients

3.3.1 Cost function and signal model

The LSE minimises the cost function defined as $J = \sum_{k=1}^{M} e^2[k] = \sum_{k=1}^{M} (x[k] - s[k])^2$, where x[n] represents the observed data and s[n] the model output. Given that:

$$J = \sum_{k=1}^{M} \left[\hat{r}_{xx}[k] - \sum_{i=1}^{p} a_i \hat{r}_{xx}[k-i] \right]^2$$

We can infer that

$$s[k] = \sum_{i=1}^{p} a_i \hat{r}_{xx}[k-i] = \sum_{i=1}^{p} a_i \boldsymbol{h}_i = \boldsymbol{Ha}$$

Therefore, the cost function is $J = (\boldsymbol{x} - \boldsymbol{s})^T (\boldsymbol{x} - \boldsymbol{s}) = (\boldsymbol{x} - \boldsymbol{Ha})^T (\boldsymbol{x} - \boldsymbol{Ha})$, where $\boldsymbol{a} = [a_1, a_2, \dots, a_p]^T$ and

$$\boldsymbol{s} = \begin{bmatrix} s[1] \\ s[2] \\ \vdots \\ s[M] \end{bmatrix} = \boldsymbol{Ha} = \begin{bmatrix} \hat{r}_{xx}[0] & \hat{r}_{xx}[-1] & \dots & \hat{r}_{xx}[1-p] \\ \hat{r}_{xx}[1] & \hat{r}_{xx}[0] & \dots & \hat{r}_{xx}[2-p] \\ \vdots & \vdots & & \vdots \\ \hat{r}_{xx}[M-1] & \hat{r}_{xx}[M-2] & \dots & \hat{r}_{xx}[M-p] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix}$$

Hence, to find the unknown coefficients we use the equation $\hat{\boldsymbol{a}}_{ls} = (\boldsymbol{H}^T \boldsymbol{H})^{-1} \boldsymbol{H}^T \boldsymbol{x} = \boldsymbol{H}^+ \boldsymbol{x}$, leading to the model output $\hat{\boldsymbol{s}} = \boldsymbol{H}\hat{\boldsymbol{a}}_{ls} = \boldsymbol{H}(\boldsymbol{H}^T \boldsymbol{H})^{-1} \boldsymbol{H}^T \boldsymbol{x} = \boldsymbol{P}\boldsymbol{x}$. This is very similar to the Yule Walker equations as defined in the lecture notes:

$$\boldsymbol{r}_{xx} = \boldsymbol{R}_{xx}\boldsymbol{a} \rightarrow \quad \boldsymbol{a} = \boldsymbol{R}_{xx}{}^{-1}\boldsymbol{r}_{xx}, \quad \boldsymbol{R}_{xx} = E\{\boldsymbol{x}^T[n]\boldsymbol{x}[n]\},$$

In Yule Walker however, $\boldsymbol{R}_{xx}$ is a positive definite (Toeplitz) matrix, which guarantees matrix conversion, which is not the case for $\boldsymbol{H}$. However, if M = p, then as defined in the equation 38 of the ASP coursework $\boldsymbol{R}_{xx} = \boldsymbol{H}$ due to the symmetric nature of the autocorrelation function, which implies full equivalence and $\boldsymbol{r}_{xx} = \boldsymbol{s} = \boldsymbol{R}_{xx}\boldsymbol{a} = \boldsymbol{Ha}$.

$$\boldsymbol{H} = \begin{bmatrix} \hat{r}_{xx}[0] & \hat{r}_{xx}[1] & \dots & \hat{r}_{xx}[p-1] \\ \hat{r}_{xx}[1] & \hat{r}_{xx}[0] & \dots & \hat{r}_{xx}[p-2] \\ \vdots & \vdots & & \vdots \\ \hat{r}_{xx}[M-1] & \hat{r}_{xx}[M-2] & \ddots & \hat{r}_{xx}[p-M] \\ & & \dots & \end{bmatrix}$$

3.3.2 Observation matrix H

The least squares approach assumes that the observed data are composed of a deterministic signal and zero mean noise, which leads to a purely deterministic s[n] that depends on unknown parameters $\boldsymbol{a}$ (however, by making this assumption, it means that the estimation can be biased by the presence of non-zero mean noises). Therefore, considering this assumption and that the output of deterministic models is fully determined by the parameter values, the observation matrix $\boldsymbol{H}$ is deterministic. For instance, the observation matrix is fully defined by the autocorrelation function of x[n], (x[n] = s[n] + w[n], where w[n] is the noise) which obeys the assumption.

3.3.3 LSE of Sunspot series

	AR1	AR2	AR3	AR4	AR5	AR6	AR7	AR8	AR9	AR10
a1	-1.631	-0.932	-4.160	-5.085	-5.581	-5.548	-4.919	-4.685	-3.079	-5.022
a2		-0.822	4.648	7.699	7.270	7.256	6.456	6.400	3.473	10.800
a3			-3.264	-6.728	-2.652	-2.535	-1.977	-2.115	0.203	-8.930
a4				1.530	-3.895	-4.426	-2.753	-2.756	-2.992	2.776
a5					2.519	3.205	-3.671	-1.222	-2.515	-4.129
a6						-0.322	8.204	0.822	6.004	5.123
a7							-3.943	4.406	-7.536	2.135
a8								-3.691	8.905	-15.410
a9									-5.505	20.194
a10										-11.031

Table 4 – LSE AR coefficients

The table above shows the AR coefficients estimated using LSE for various orders by using the equation $\hat{\boldsymbol{a}}_{ls} = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{x}$. Note that the parameters here need to change sign to adapt to the MATLAB notation.

3.3.4 Sunspot LSE error

The figure on the right shows the average approximation error for different orders, where the approximation error is defined as the magnitude in difference between the actual and estimated values $|x[n] - s[n]|$, and s[n] is computed using $\hat{\boldsymbol{s}} = \boldsymbol{H}\hat{\boldsymbol{a}}_{ls}$.

In the previous section, it was determined that the optimal orders for modelling the sunspot series are 2 and 9, with order 2 minimising the additional overfitting penalty. The figure on the right however does not follow this conclusion, where the approximation error tends to decrease as a function of the order. This can be explained by how the higher the model order the better it fits the training data, but this comes at the price of low generalisation performance outside of the training set.

However, the biggest change in performance happens for order 2 and 10 as indicated by the lower graph of the figure, which computes the difference in performance from the previous order, hence suggesting some similarity with the previous analysis.

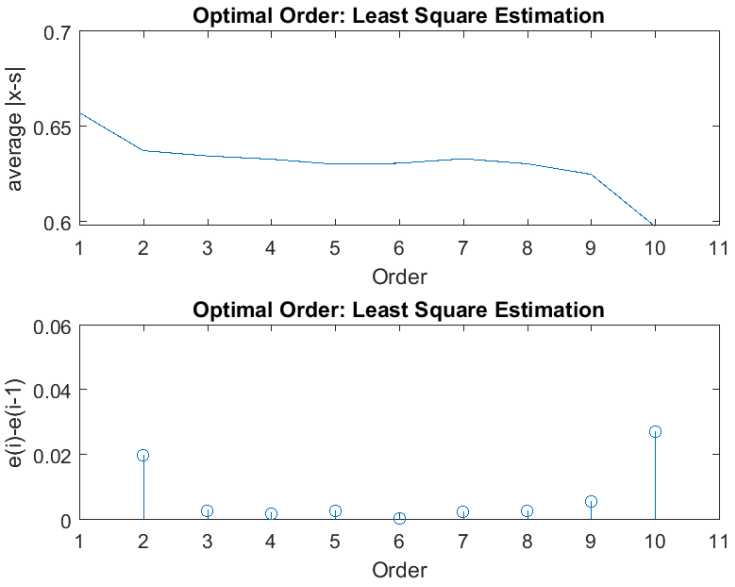


Figure 36 – sunspot LSE error

3.3.5 LSE Power Spectra

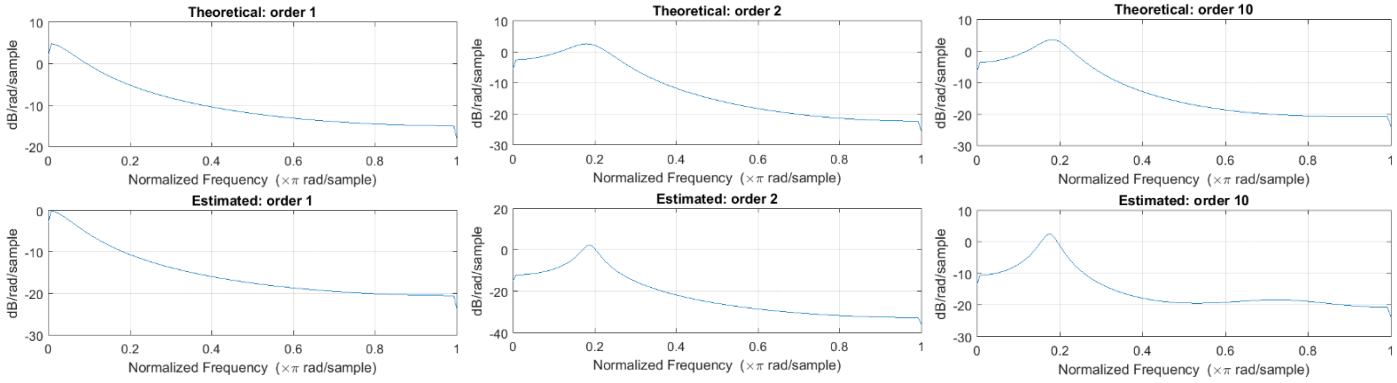


Figure 37 – LSE power spectra

The graphs above show the actual and estimated sunspot PSD for various orders. The graphs are generated using the function `pyulear` for the given order, passing the sunspot series for the theoretical graph and the estimated series $\hat{\boldsymbol{s}} = \boldsymbol{H}\hat{\boldsymbol{a}}_{ls}$ for the estimated graph. Despite the differences in the time domain, the PSD of the estimated series is almost identical to the original one, with peaks happening around the same normalised frequency values. All the PSD graphs generated for orders ≥ 2 have peaks around f = 0.2, with slight increase in difference as the order increases (effects of over-modelling). For order 1 however, the peak is at the f = 0. This shows how under modelling produces inaccurate results as it cannot fully capture the characteristics of the series.

3.3.6 MSE and data lengths

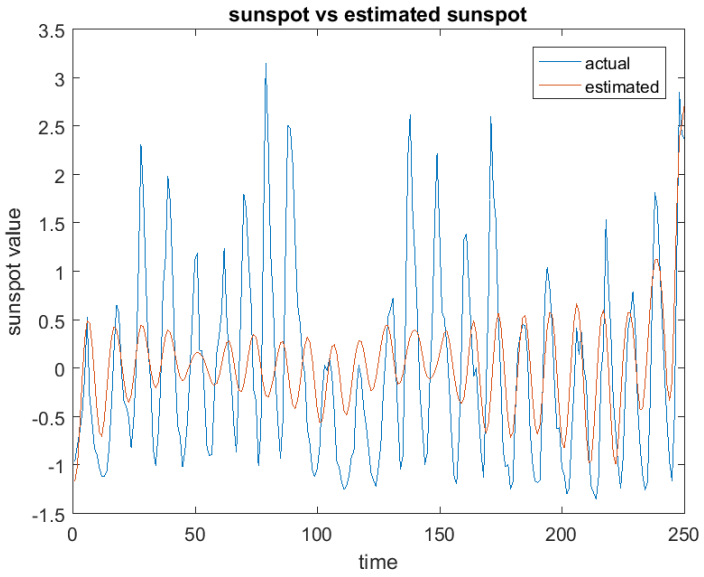


Figure 38 – actual and estimated sunspot

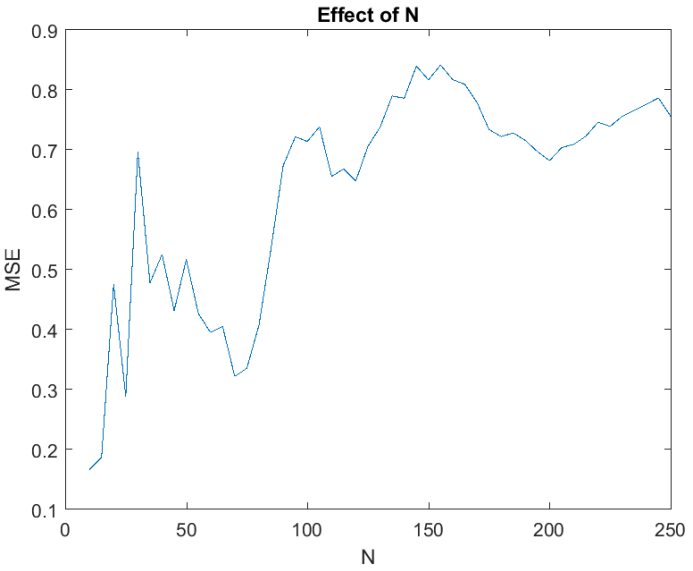


Figure 39 – approximation error vs N

Figure 38 shows the estimated sunspot series for 250 data points, while figure 39 shows the mean squared error for different values of data points. The graph on the right clearly shows that the mean squared error increase and eventually saturates for increasing number of N. This is expected given the approach used for computing the MSE in this case. For instance, for small number of data points (e.g. first 10 data points), the full characteristic of the sunspot series is not entirely captured, resulting in an apparent sine wave. The LSE of AR2 process is therefore able to recreate this simple sinewave using $\hat{\boldsymbol{s}} = \boldsymbol{H}\hat{\boldsymbol{a}}_{ls}$, producing 10 data points that matches well the first 10 data points of sunspot series x[n], resulting in small MSE. However, as the number of data points increases, the full characteristics of the sunspot series emerge and the least square estimation loses its accuracy, resulting in higher MSE.

However, if we were to change the approach and make the estimated $\hat{\boldsymbol{s}}$ periodic for the full 250 data points and then compute the difference, the MSE is likely to decrease as function of N.

3.4 Spectrogram for time-frequency analysis: dial tone pad

3.4.1 Random London landline number

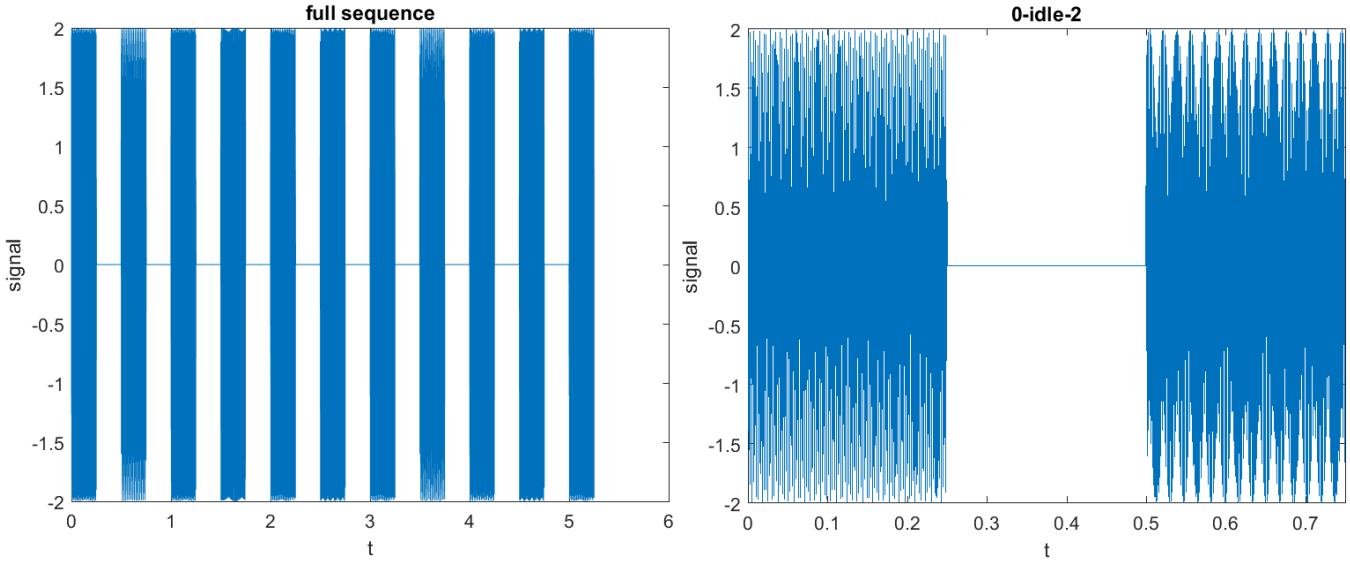


Figure 40 – landline waveform

The above figure shows the complete discrete time sequence y and the first 0.75 seconds of the same sequence. There are 11 numbers total with 10 idle time in between, all of which have the pressed time and idle time of 0.25s, which leads to a total of 0.25*21 = 5.25 seconds. As the figure above shows, there are intertwined intervals of waveforms and zero values, which is expected due to the alternating of pressed time and idle time. The sampling rate of 32768Hz (fs) makes sense since by Nyquist theorem, we can work with frequencies up to fs/2 = 16384Hz, while the highest frequency present is only 1477 Hz. In addition, since we have 0.25 seconds (T), it can generate fs*T = 8192 samples, which is perfect for digital processing since $8192 = 2^{13}$ is a power of 2.

The figures below zoom in further to show the waveforms for key 0 and key 2.

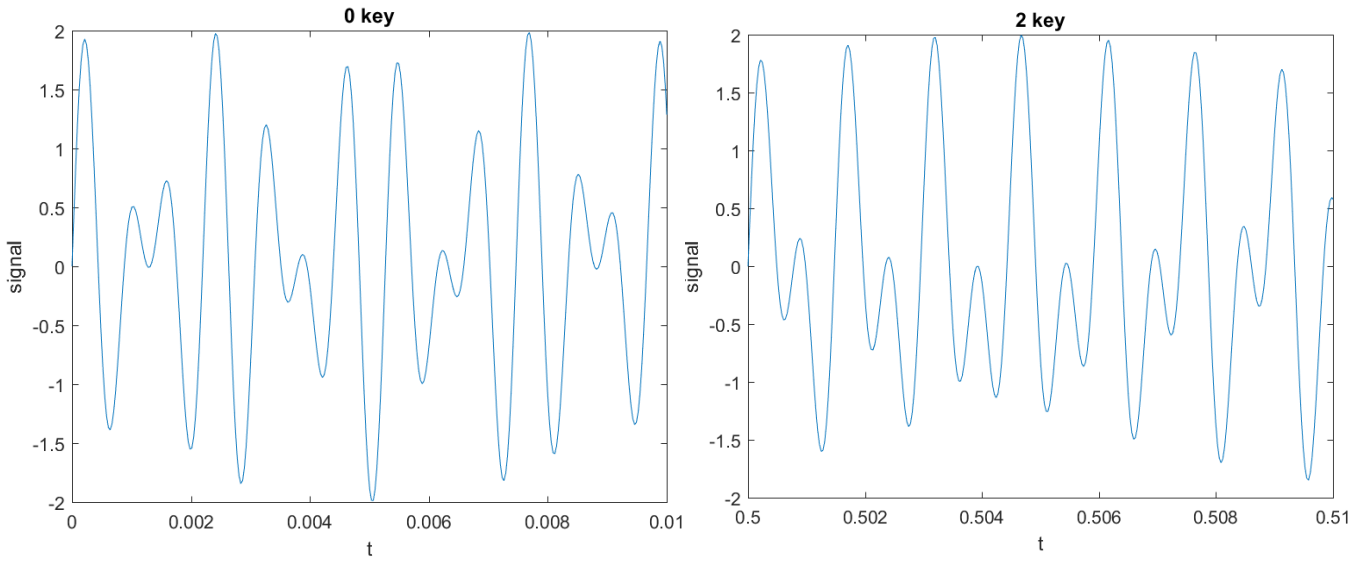


Figure 41 – key waveforms

3.4.2 Spectrogram

The figure below shows the spectrogram for the generated landline number (020 4931 5529), which was created using the MATLAB function `spectrogram(y, hann(8192), 0, 8192, 32768, 'yaxis');`

This function plots the spectrogram for the sequence y with frequency displayed on the y-axis using the Hanning window of length 8192 (same as the number of samples in the idle/pressed time of 0.25s). The above function takes into account zero overlapping samples, cyclical frequency of 8192 (this essentially divides the signal into sections of length 8192, which is the number of samples in 0.25s) and the sampling rate of 32768Hz.

The alternating of deep blue and of lighter colours are the alternating pressed and idle time that spans for 5.25 seconds. In addition, during the pressed time, the two frequencies that make up the particular key are highlighted in yellow (higher power dB/Hz) which allows us to recognize what keys are being pressed. However, the light blue colour surrounding the yellow peaks are present because of the leakage effect of the window (Hanning) which sets non zero values for frequencies around the key tones that fades away from the main frequencies.

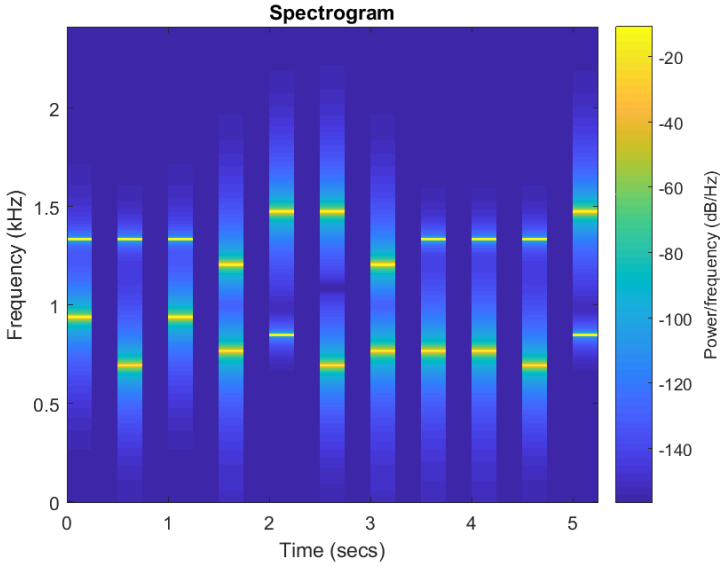


Figure 42 – noiseless spectrogram

3.4.3 Key classification

From the above figure, it is clear that in a noise free environment, the identification of the key is straightforward since there is a significant peak in power for the associated frequencies of a certain key. Hence, it is possible to set a threshold in power above which you take into consideration the frequencies and compare them to the table of frequencies to perform key classification (see figure 46 for the plot).

3.4.4 Noisy sequence

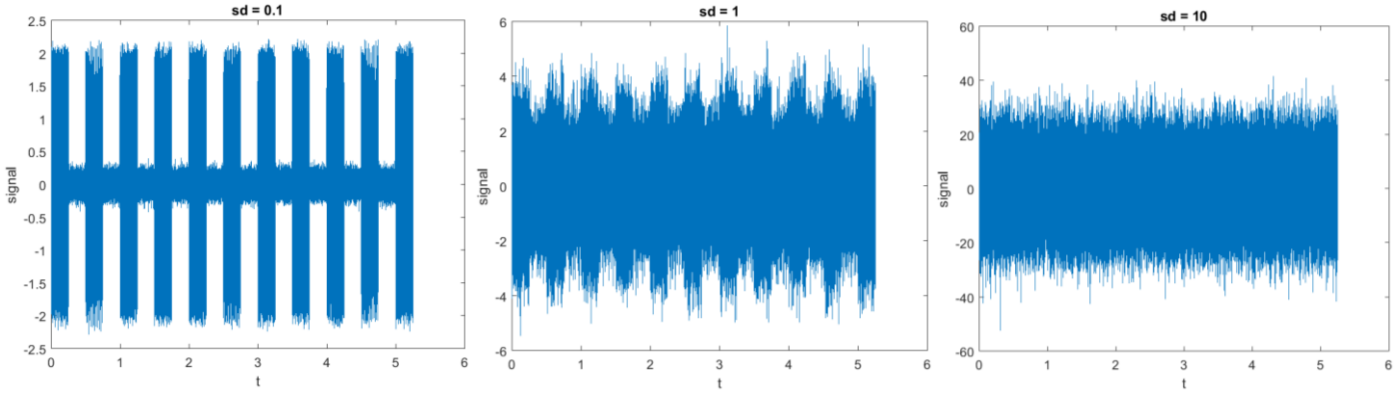


Figure 43 – noisy landline waveforms

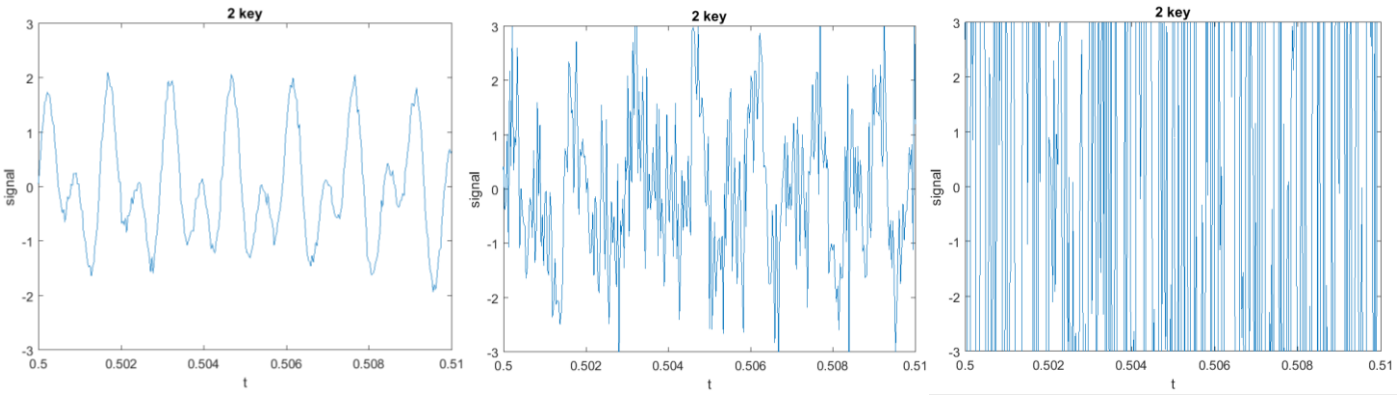


Figure 44 – noisy key waveforms

Figure 43 shows the full sequence of the landline, with an additive Gaussian noise with standard deviation of 0.1, 1 and 10 respectively from left to right. Visually, the smaller the variance the smaller is the effect of the additive noise. For the standard deviation of 10 the noise disrupts the signal completely in the time domain. Figure 44 instead shows the waveform for key = 2 in time domain for the same noises.

However, even for highest values of variance taken into consideration (variance = 100), the characteristic frequencies of various keys are still identifiable given their strong magnitudes. Although it is increasingly more difficult to notice the yellow peaks in the spectrogram, the key classification using the method mentioned in section 3.4.3 is still possible. For instance, if we look closely to the pressing time for the key 2 in the magnitude vs frequency domain, the characteristic frequencies are still noticeable. Figure 46 below compares the third 0.25s period (the pressing period for key 2) for the noiseless case and the standard deviation = 10 case. Even for the worst scenario taken into the consideration, the peaks around the frequency 697Hz and 1336Hz are still noticeable. Hence, the key classification by amplitude screening is still a viable method.

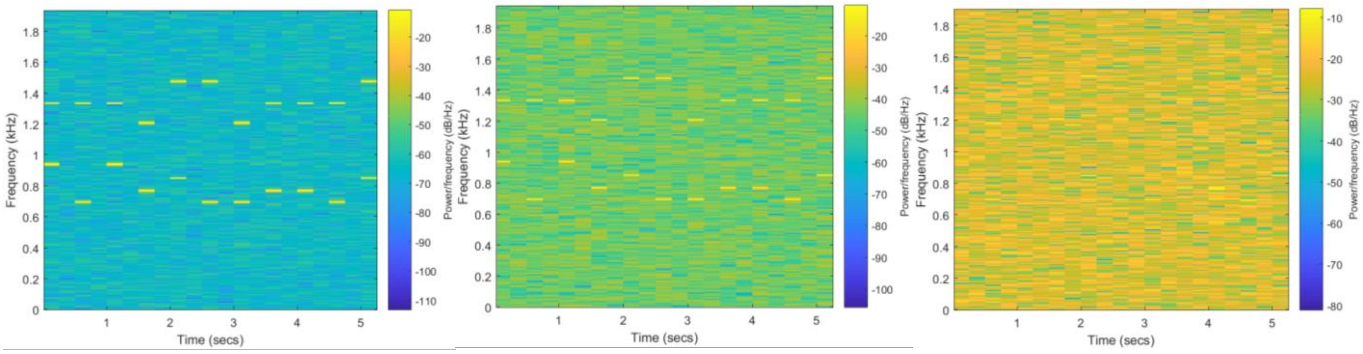


Figure 45 – noisy spectrograms

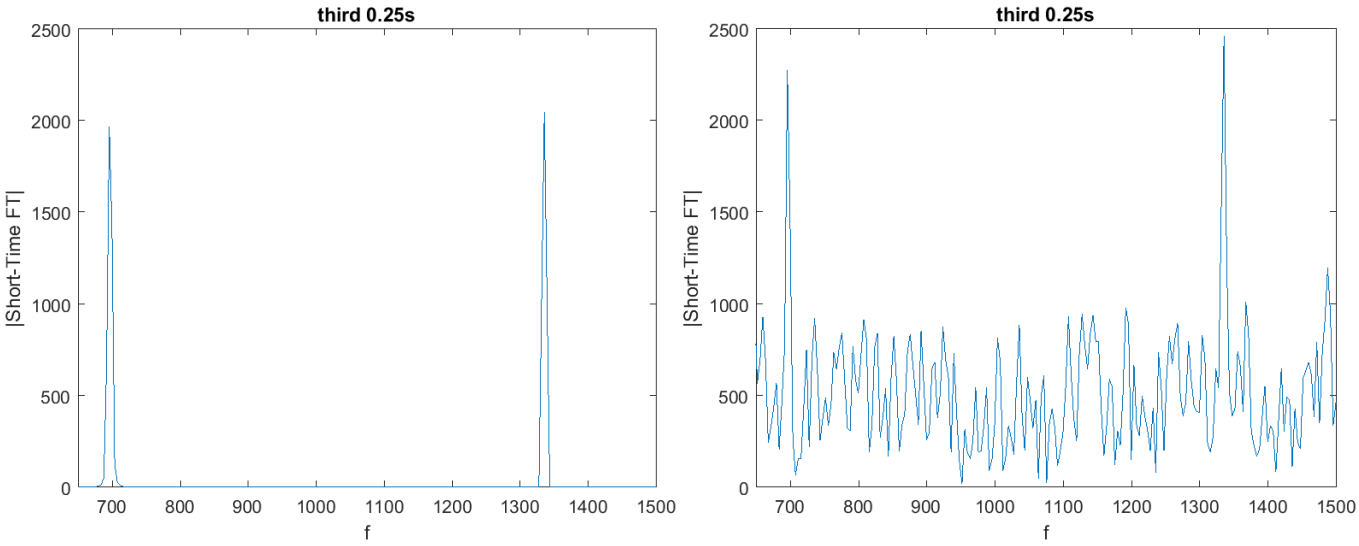


Figure 46 – key classification

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

3.5.a Standard periodogram

The figures below show the periodogram of the RRI data from the 3 trials, where figure 47 shows the implementation of the classical periodogram method, while figure 48 shows the averaged periodogram method with a window length of 50 and 150 respectively.

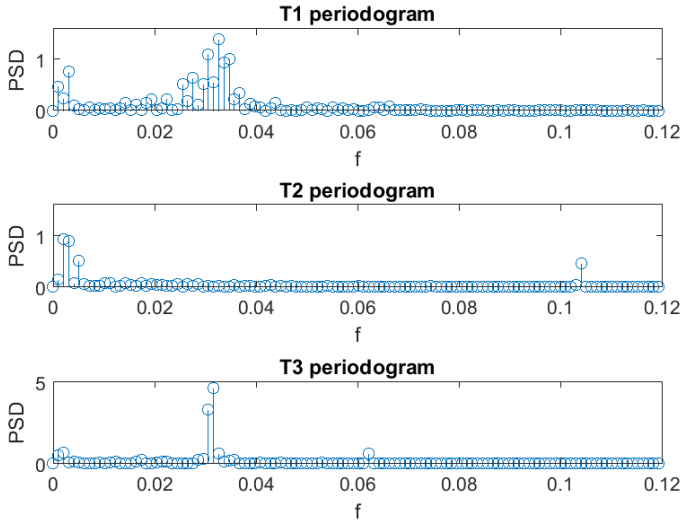


Figure 47 – heart periodograms

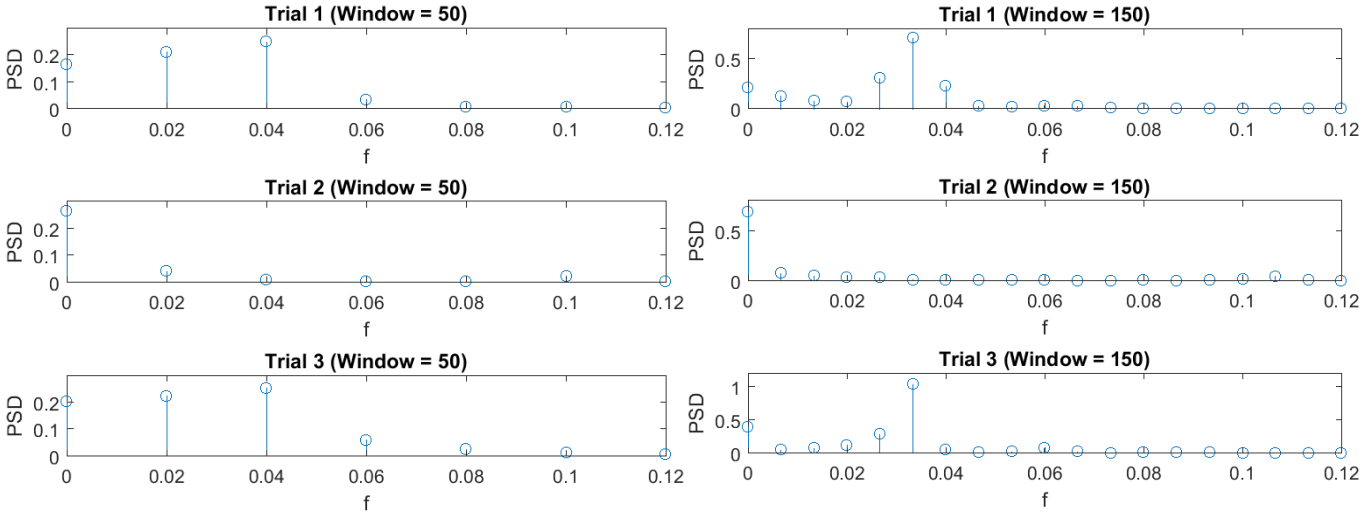


Figure 48 – averaged heart periodograms

3.5.b PSD analysis

As mentioned in previous sections, the averaged periodogram needs to balance the trade-off for which it loses the frequency resolution at the cost of better estimating the PSD values. For instance, for the window length = 50 scenario, due to the bad frequency resolution at low frequencies, there is a strong peak for the DC component. This drawback is much less present for the window length = 150 case, where the DC component is reduced by a greater magnitude.

From the periodogram with window length = 150 graphs, the peak for trial 3 is easily identifiable around the normalised frequency of 0.03, whereas the rest of the frequency components have negligibly small values. This is expected as the breathing for this trial was constrained at a slow and steady rate of 15 beats per minute.

Trial 1 also has the same peak as trial 3, but with some additional stronger frequency components around it. This is also expected as the breathing was not constrained and random, leading to random peaks in frequency.

Trial 2 (50 beats per minute) instead shows unexpected characteristics with a strong DC component. It is possible that due to unnaturally fast inhaling and exhaling, the modulation of the cardiac function was not carried-out fully, leading to an almost non-cyclical behaviour in the RRI data which was mistaken as DC. The actual frequency for trial 2 should be $\frac{50}{15}$ times faster than trial 3, which leads to a peak around the frequency $\frac{50}{15}0.03 = 0.1$Hz. This peak however can be identified in the regular periodogram plot, which means that it was potentially not picked up by the averaged pgm due to its frequency resolution issues.

(4) Optimal filtering: fixed and adaptive

4.1 Wiener filter

4.1.1 Optimal Wiener filter

The optimal Wiener filter is method is carried out in MATLAB following the procedure below:

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

```
x_n_xcorr = xcorr(x, z);              % cross correlation x z
tmp = x_n_xcorr(1000-4:1000);         % note: Nw=4, and sample number 1000 = r_zx(0)
Pzx = flip(tmp);
```

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

```
x_acorr = xcorr(x);                       % auto correlation
RxxColumn = x_acorr(1000:1000+4);    % sample number 1000 = r_xx(0)
Rxx = toeplitz(RxxColumn);                % column = row since ACF is symmetric
```

$$\boldsymbol{w}_{opt} = \boldsymbol{R}_{xx}^{-1}\boldsymbol{p}_{zx}$$

```
Wopt = inv(Rxx)*Pzx;
```

The returned value for Wopt is $[0.9990, 1.9995, 2.9963, 2.0005, 0.9973]$, which is very close to the coefficients b = [1,2,3,2,1]. The signal to noise ratio is in this case $SNR = 10log_{10}\frac{\sigma_y^2}{\sigma_{wgn}^2} = 10log_{10}\frac{17.7}{0.01} = 32.5dB$.

4.1.2 Optimal Wiener filter with additive noise

Wopt\Variance	0.1	0.2	0.5	1	5	10
w1	1.002	1.002	1.004	1.005	1.012	1.016
w2	1.995	1.993	1.989	1.985	1.967	1.953
w3	2.994	2.992	2.988	2.983	2.964	2.950
w4	1.995	1.993	1.989	1.986	1.969	1.956
w5	1.021	1.030	1.047	1.066	1.148	1.209

Table 5 – estimated coefficients (Nw=4)

From the above table (Nw = 4), it can be noticed that as the noise increases in variance, the five values are increasingly deviating more from b=[1,2,3,2,1] while becoming less symmetric about w3. In addition, the value of w3 seems to be decrease while the other coefficients increase in value. Therefore, the greater the variance for the noise, the more it affects the performance of the filter. This is expected since the signal with additive noise does not display the characteristics of the original filter as it is corrupted by the noise.

From the table below (the same procedure repeated for Nw = 6 and Nw = 8), the first five coefficients are a great estimate of b regardless of the value of Nw. For the coefficients that are not needed (w6 to w9) the values are almost zero as expected, since they will not contribute much to the prediction of the output (they are non-zero only because of the added noise).

	Nw = 6						Nw = 8					
w/var	0.1	0.2	0.5	1	5	10	0.1	0.2	0.5	1	5	10
w1	1.000	1.000	1.000	1.000	1.000	1.000	1.023	1.033	1.051	1.073	1.162	1.229
w2	1.982	1.976	1.965	1.952	1.898	1.857	1.989	1.985	1.976	1.967	1.926	1.896
w3	2.986	2.981	2.973	2.963	2.922	2.892	2.986	2.981	2.970	2.957	2.904	2.864
w4	1.983	1.977	1.965	1.952	1.896	1.854	2.002	2.005	2.009	2.014	2.035	2.051
w5	0.986	0.982	0.975	0.967	0.932	0.906	0.997	0.997	0.995	0.994	0.987	0.982
w6	-0.016	-0.020	-0.029	-0.039	-0.080	-0.110	0.016	0.021	0.031	0.042	0.090	0.125
w7	0.013	0.015	0.020	0.024	0.045	0.060	0.008	0.010	0.013	0.016	0.031	0.041
w8							-0.001	-0.002	-0.005	-0.008	-0.022	-0.032
w9							0.016	0.021	0.031	0.043	0.091	0.127

Table 5 – estimated coefficients (Nw=6,8)

4.1.3 Optimal Wiener filter computational complexity

In order to determine the computational complexity of the Wiener filter, we must find the computational complexity for computing: $\boldsymbol{p}_{zx}$, $\boldsymbol{R}_{xx}$, and $\boldsymbol{w}_{opt} = \boldsymbol{R}_{xx}{}^{-1}\boldsymbol{p}_{zx}$.

Given that the correlation function needs to perform multiplication of all N samples and add them together, it needs $O(2N) \sim O(N)$ operations. This is then repeated for a lag that is equivalent to $(N_w + 1)$, which means that the correlation function alone takes $O(NN_w)$, which is carried out for both $\boldsymbol{p}_{zx}$ and $\boldsymbol{R}_{xx}$. To compute the inverse of $\boldsymbol{R}_{xx}$ it takes $O((N_w + 1)^3) \sim O(N_w^3)$ and multiplying $\boldsymbol{R}_{xx}{}^{-1}\boldsymbol{p}_{zx}$ with a naïve algorithm takes an additional $O(N_w^3)$ operations. This leads to a complexity of $O(NN_w + N_w^3)$.

Note that the above complexity calculation takes into account an efficient algorithm that computes the correlation function only for the lags needed. The implemented code however computes for all possible lags in order to facilitate the analysis for different values of Nw.

4.2 The least mean square algorithm

4.2.1 The LMS function

Below is the MATLAB code for implementing the adaptive system from figure 4 of the ASP coursework:

```
function [y, e, w] = lms(x, z, u, order)
    w = zeros (order, length(x));
    w(:, 1) = zeros(order, 1);      % initialize w to zero
    for i = order:length(x)
        y(i) = (w(:, i-order+1)') * flip(x(i-order+1:i));
        e(i) = z(i) - y(i);
        w(:, i-order+2) = w(:, i-order+1) + u*e(i)*flip(x(i-order+1:i));
    end
end
```

The above function implements the formulae (39), (40) and (41) outlined in the ASP coursework. The function takes as input the N-sample vectors **x** and **z**, the adaptation gain u, and the order of the adaptive filter. The output is the (N_w+1)xN sample matrix **w** (evolution of weights over time), the N-sample vectors LMS estimate **y** and the error **e**. Note that we flip the vector `x(i-order+1:i)` because it is defined as $\sum_{m=0}^{Nw} w_m(n)x(n-m)$.

When applying the above function to the previous dataset, we get the characteristics shown in figure49. Due to the adaptive nature of the LMS algorithm, the error (left figure) decreases as n increases as it learns from more data. Unlike the wiener filter where the solution is given by solving the Lagrangian to follow the negative gradient directly to minimize error, here are we are taking a zig-zag approach that on average will lead to a local error minimum. Indeed, the error decreases almost exponentially in magnitude while bouncing from positive to negative values. From the evolution of the coefficients on the right, it is clear that 5 parameter eventually reach the value of b.

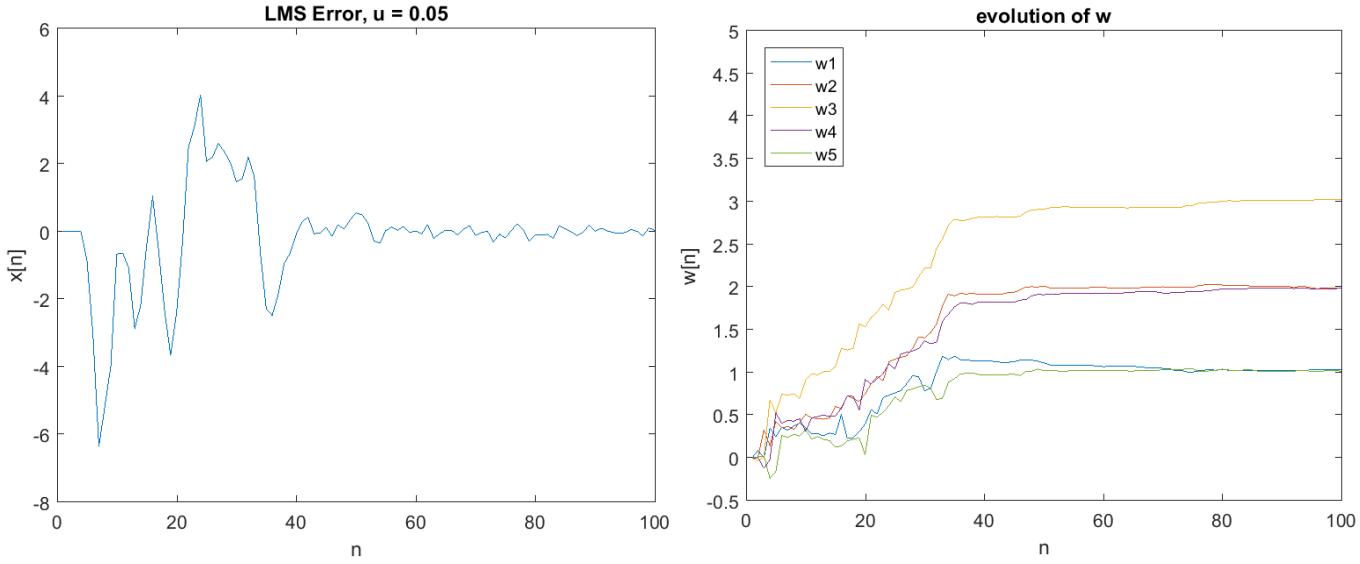


Figure 49 – LMS estimation

4.2.2 The effects of the adaptation gain

When the adaptation gain is small, the squared error decrease faster as we increase the learning rate. From the plot of **w**[n], it is clear that **w** eventually converge to b. In the case of u =0.01, these values are reached at small values of n (n=170) and they remain stable, whereas for the smaller u=0.002 this happens at a smother but slower rate.

As the adaptation gain increases however, the squared error does not always decrease as a function of n. This can be explained in terms of the stability, since when the learning rate is too big, it starts to overshoot the correction term and change **w** by an excessive amount. This overcompensation in **w** leads to an overshoot in the estimate of y, which causes an error much larger than it was trying to compensate in the first place. Hence, it is clear that the learning rate must be adjusted proportional to the error.

Therefore, as indicated in the ASP notes, there exists an u_{crit} such that the system is convergent only if $u < u_{crit}$. Another look at the mean square convergence requires $0 < u < \frac{2}{tr[\boldsymbol{R}_{xx}]}$.

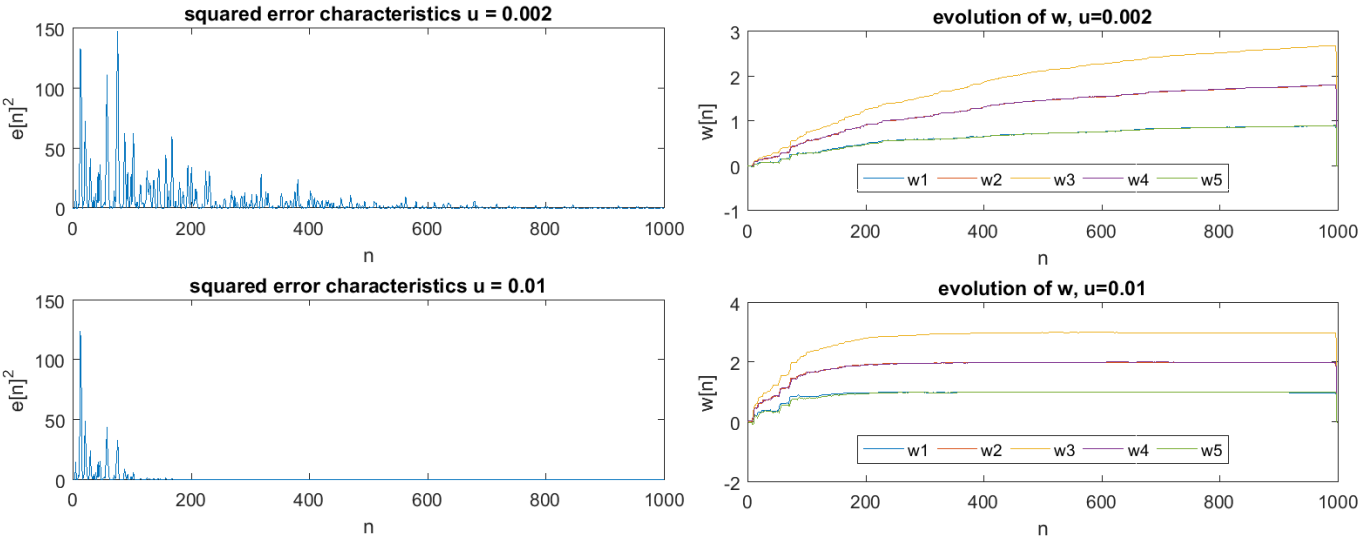


Figure 50 – effects of the adaptation gain (1)

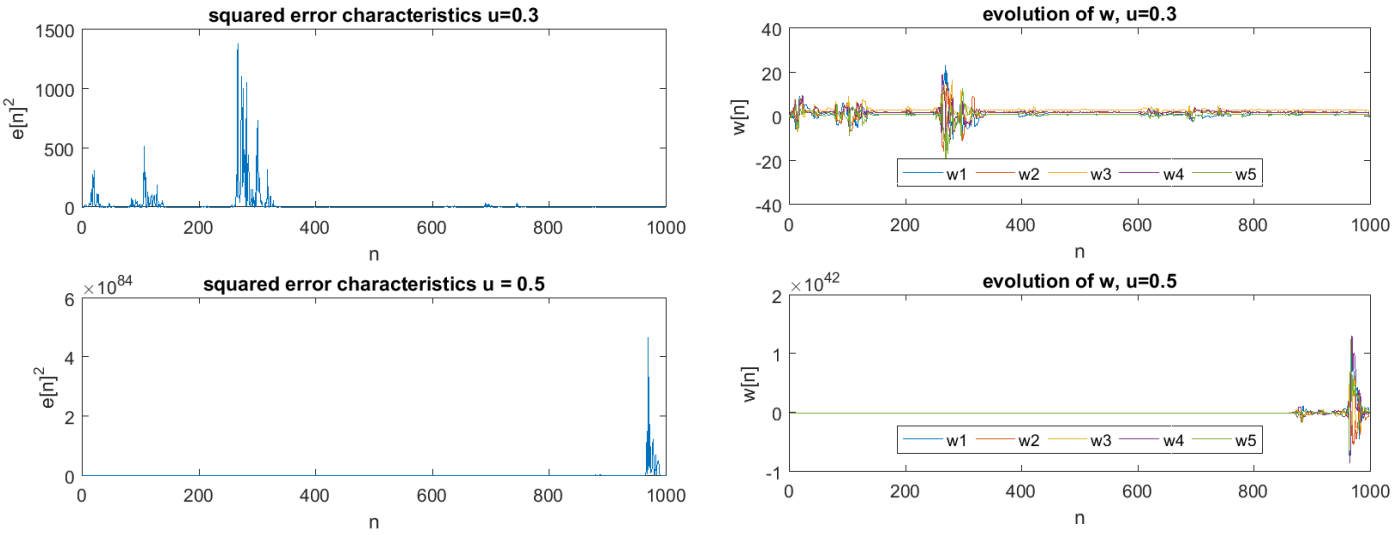


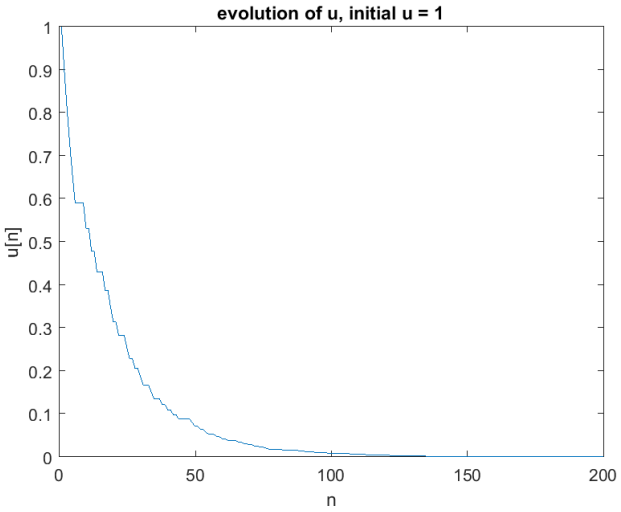
Figure 51 – effects of the adaptation gain (2)

4.2.3 Computational complexity of LMS algorithm

At each iteration, $\hat{y}[n] = \sum_{m=0}^{Nw} w_m(n)x(n-m)$ takes (N_w+1) multiplications and N_w additions, hence of complexity $O(N_w)$. The error $e[n] = z[n] - \hat{y}[n]$ on the other hand takes only 1 operation. $\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + ue[n]\boldsymbol{x}(n)$ takes 2 multiplications per (Nw+1) values of **x**, hence 2(N_w+1) operations. Finally, this last value is added to the **w** from the last iteration which takes Nw+1 sums, leading to a total of $O(N_w)$ in complexity. This is significantly faster than the wiener filter.

4.3 Gear shifting

```
function uo = gearshift (ui, e, i)
    % if the squared error is getting bigger it means that the current
    % value for u is too big, hence decrease it by 10%
    if (e(i)^2 > e(i-1)^2)
        uo = 0.9*ui;
    end
    if (e(i)^2 <= e(i-1)^2)
        uo = ui;
    end
end
```



For gear shifting, as the LMS learns more from the data, it needs smaller step size in order to improve the tracking behaviour in the MSE sense at steady state. This is the idea behind the above algorithm, which compares the error of successive iteration, and decreases the step size only if there is increasing error, otherwise, the same learning rate is kept to allow faster convergence. This leads to a learning that only decreases as time moves on, which is plotted in the figure on the right.

In terms of performance, we consider the learning rate u = 1. For this value of step size, which is much bigger than the previously considered static learning rates that led to instability, the performance is much better as the figure 53 indicates.

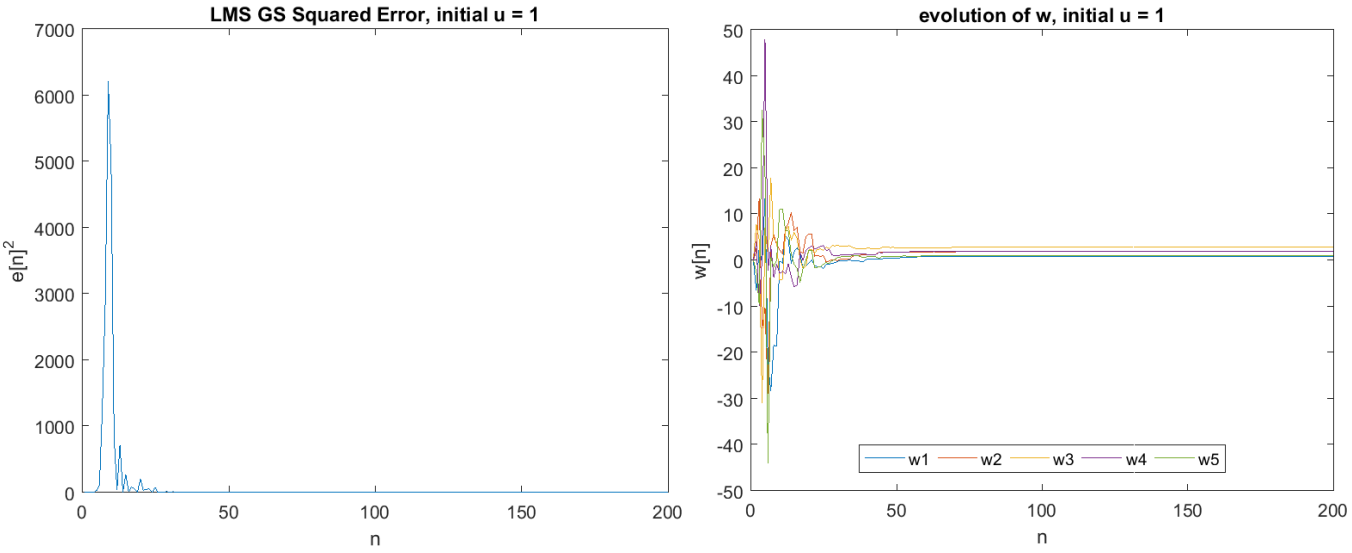


Figure 53 – gear shifting LMS estimation

4.4 Identification of AR processes

4.4.1 AR parameters estimation

By using the LMS algorithm to identify the parameters of the AR processes, the asymptotic estimates converge around values of -0.9 and -0.2. These are the negative values of the vector a = [1 0.9 0.2] used to implement the MATLAB function, but this is simply because of MATLAB notation, which considers:

$$x[n] + a_1 x[n-1] + \cdots + a_p x[n-p] = 0$$

Instead the notation taught in lectures:

$$x[n] = a_1 x[n-1] + \cdots + a_p x[n-p]$$

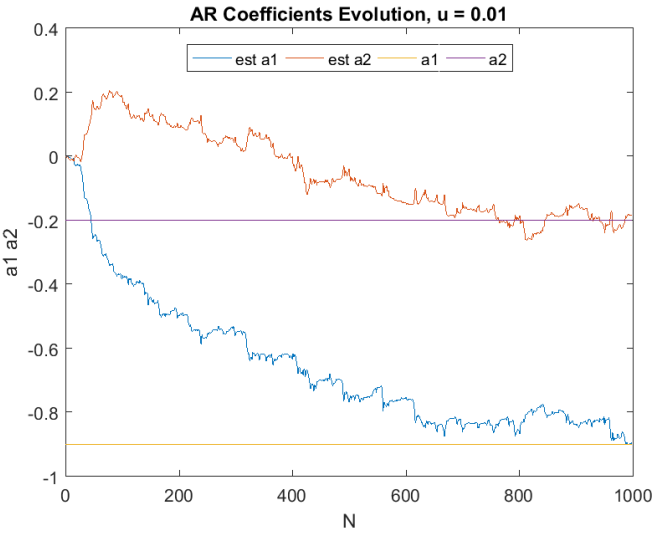


Figure 54 – AR estimation (u=0.01)

4.4.2 AR parameters estimation, varying u

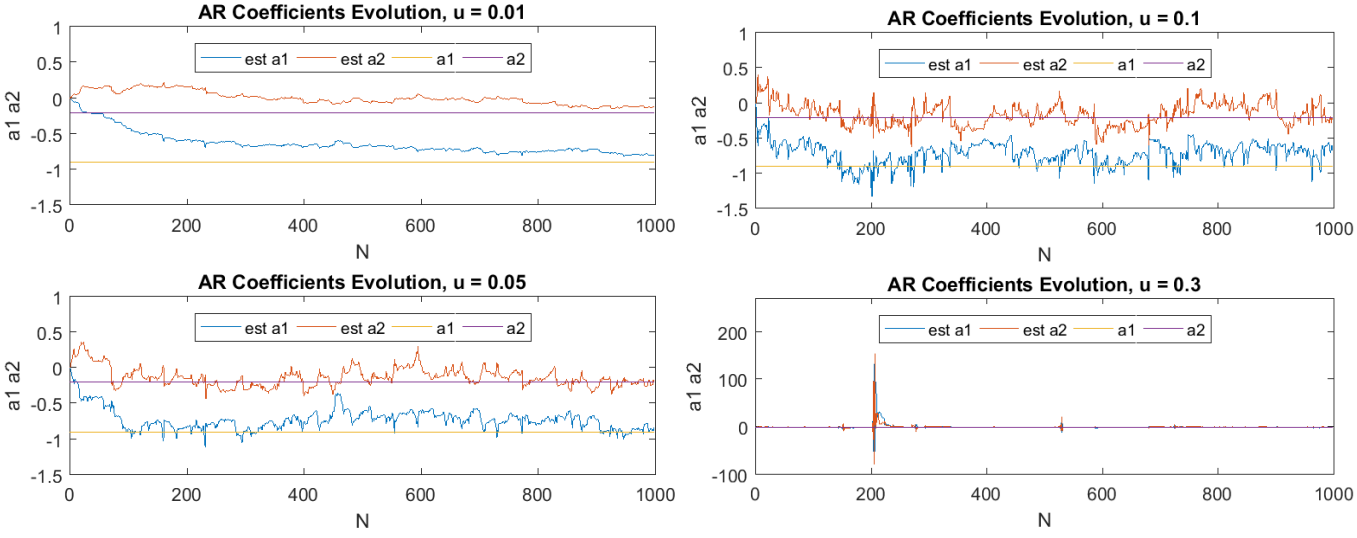


Figure 55 – AR estimation (varying u)

The effect of different values of learning rate is similar to the previous exercises. For instance, before a critical value of u, increasing the learning rate allows for the estimation of the parameters to converge faster, although this comes at the cost of higher oscillation. When the learning rate is too big (for e.g. in u = 0.3), the convergence is extremely fast, although this comes at the cost of big sudden errors for the same reasons mentioned previously.

4.5 Speech recognition

4.5.1 Effects of adaptation gain and predictor order

The figure on the right shows the sound waves recorded when pronouncing different letters, while the figures on the bottom shows the error characteristics of the estimation as a function of N, order and learning rate for the sound e specifically.

As expected, as the learning rate increases, the estimation error converges readily to zero. It is also interesting to notice that in this particular case, there is no penalty in error for higher learning rates up until 1, so for this case in particular there is no need to apply gear shifting.

A similar trend can be noticed when varying the order, where the smaller orders tend to perform worse than the other ones. This is also expected, as models of smaller orders cannot capture the full characteristics of the sound wave. This behaviour is generally true, but it is clear from the graph that the error is not always a decreasing function of the order, as it has some local minimum along the way. This suggests that there is an optimal order that not only minimises the error but also reduces the complexity.

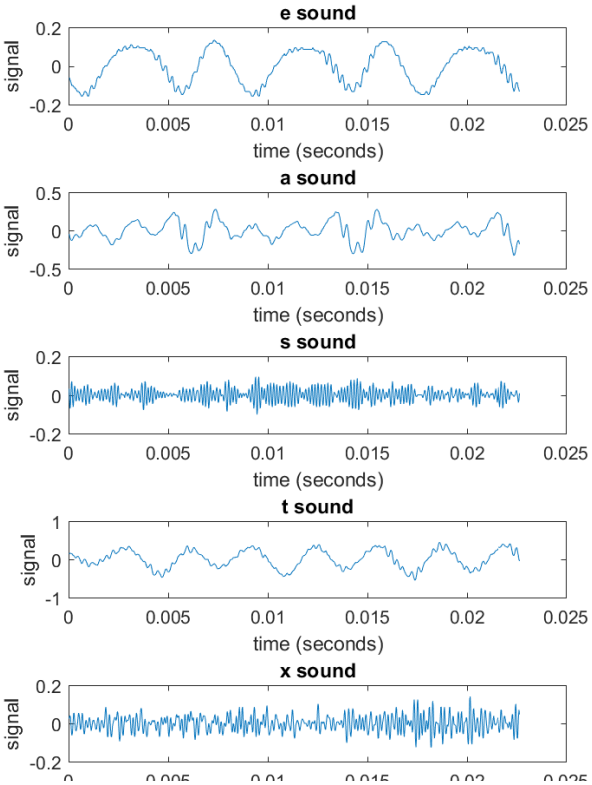


Figure 56 – recorded sounds

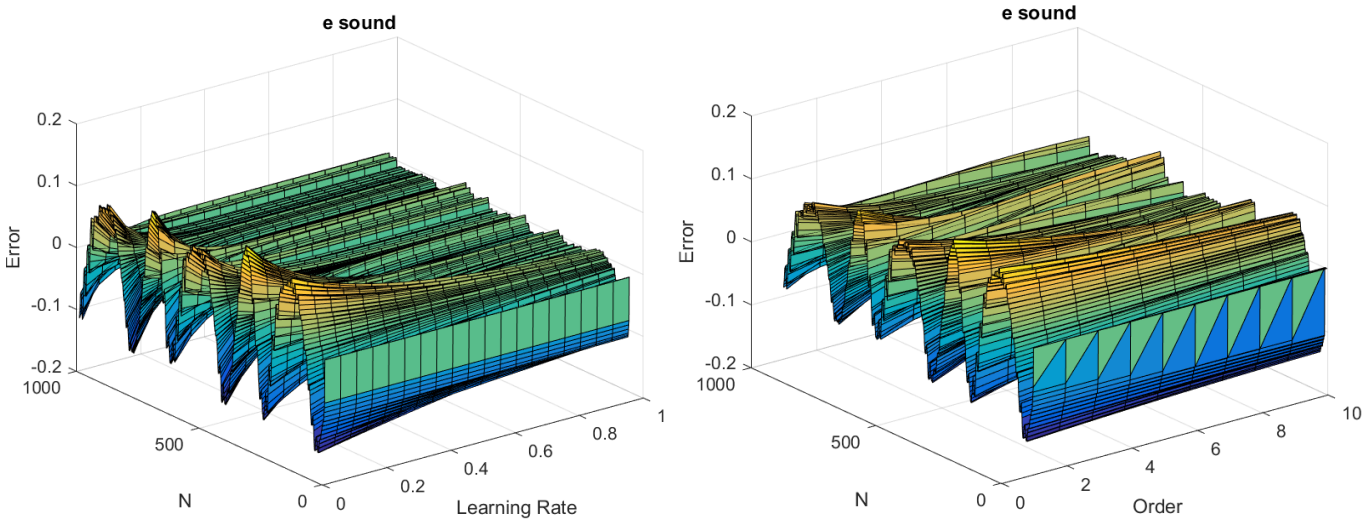


Figure 57 – effects of order and learning rate

4.5.2 Optimal Filter Length

The error decreases as a function of order reaching almost a plateau where increasing the order further will not increase the performance by much anymore. Therefore, a suitable strategy for finding the optimal filter length could be by quantifying the error through statistical methods (such as mean or variance) and differentiating against the length. Ideally, the optimal length is the one for which the derivative reaches 0 (hence the error does not decrease as the length

increases anymore), but this is rarely the case. Hence, it would be advisable to set a margin of tolerance (e.g. 0.2) and choose the order for which this margin is reached.

The figure above plot the prediction gain $R_p = 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_e^2}\right)$ for the sound 'e', which can be used as a way to quantify the error as discussed in the previous paragraph in order to find the optimal filter length.

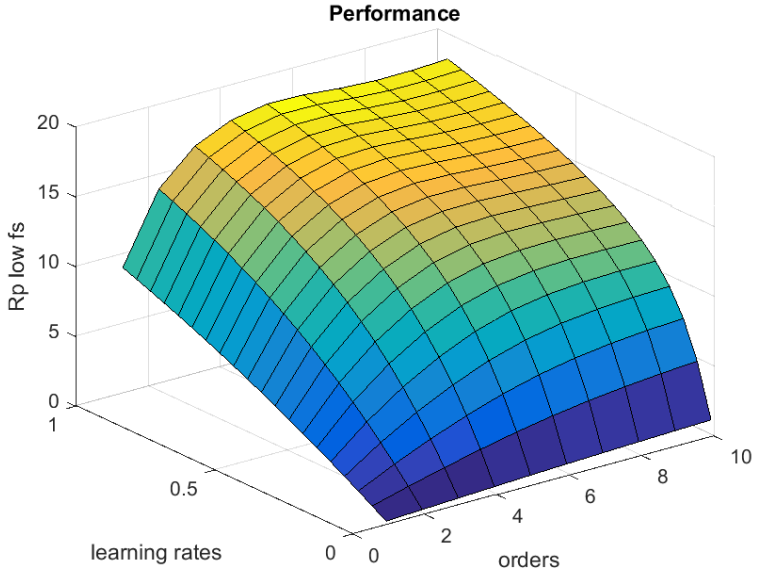


Figure 58 – Rp

4.5.3 Predictor performance for various sounds

The figures below shows Rp for a fixed learning rate (0.06) comparing the performance at $f_s = 44100Hz$ and $f_s = 16000Hz$ for the quasi-stationary vowel sounds. It is evident that R_p for high sampling frequency not only performs better reaching higher values, it also tends to converge faster to a fixed value as well. This can be shown by the figure on the right that shows that its derivative reaching 0 at smaller order. The better performance is likely due to higher frequency allows for a more smooth reconstruction of the function, which is more easily estimated.

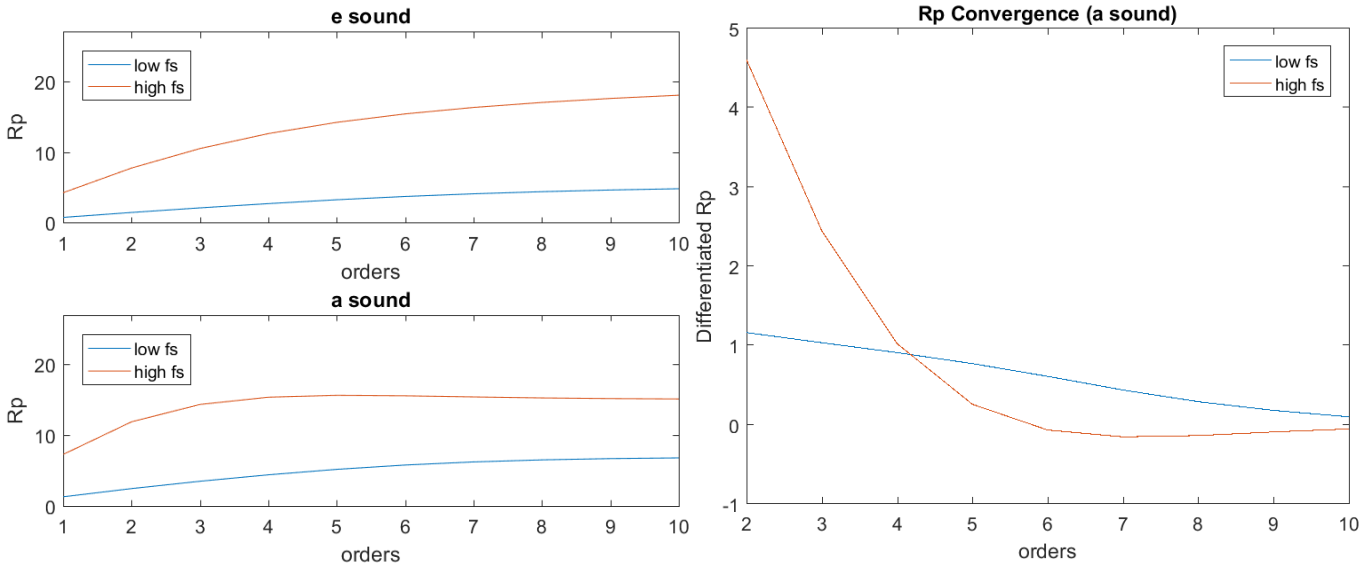


Figure 58 – Rp for vowels

The following table compares the predictor performance for all the sounds recorded for this exercise at $f_s = 44100Hz$ by fixing the learning rate = 0.06 and order = 6, the value around which Rp converges. As expected, the waveform with lower frequency and more periodic behaviour (see figure 56) obtained the highest Rp, as they are the easiest to model. Among the best performing ones, all are vowels ('e' and 'a') with the exception of the sound 't'. However, the waveform of 't' is extremely similar to the one for 'e' since it is pronounced 't-eee', which might explain its high performance as despite being a consonant.

sound e	sound a	sound s	sound t	sound x
6.676	6.751	0.591	10.919	0.537

Table 6 – Rp of various sounds

4.6 Dealing with computational complexity: sign algorithms

AR processes

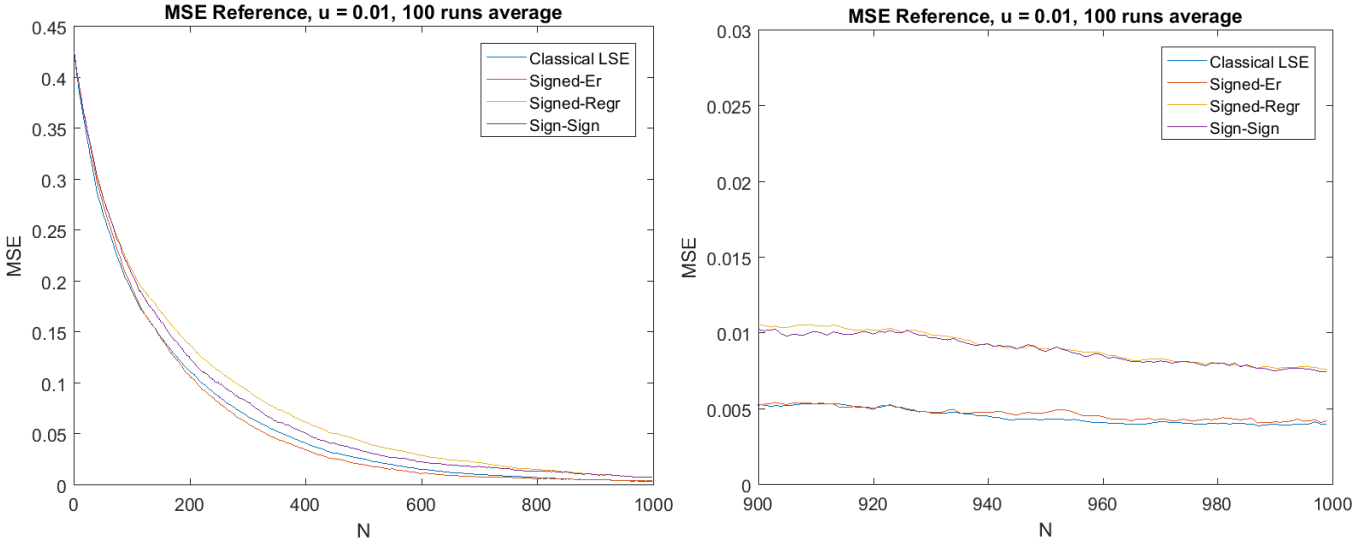


Figure 59 – performance of sign algorithms

The above figure show the performance of different signed algorithms in terms of mean squared error after averaging them over 100 runs. The mean squared error is calculated using the following method:

$$e_1[n] = \left(a_1 - a_{1_{estimate}}[n]\right)^2, \quad e_2[n] = \left(a_2 - a_{2_{estimate}}[n]\right)^2, \quad MSE[n] = \frac{e_1[n] + e_2[n]}{2}$$

As the figure indicates, the only result that improves upon the classical LSE method is the signed-error algorithm, which proves to converge at a faster rate than all the methods considered while obtaining similar accuracy to the classical LSE. This is not surprising given that once normalising the data, the error term will be likely to be less than unity, which will affect the correction factor by reducing it. Hence, by taking only the sign (+1 or -1), this value is larger in magnitude than the error term as it is unity, which increases that correction factor compared to the classical LSE method, and accelerates the convergence. Hence, to assess this effect, one must consider all the implications of convergence as the learning rate changes as discussed previously.

The signed-regressor and sign-sign algorithm instead are the worse performing in terms of both convergence and accuracy. This is expected since the optimal correction value that is dependent on the x[n] is being diluted by only taking the sign, which means that the direction of the correction is correct but the size is less than optimal. The slightly faster convergence characteristics of the sign-sign algorithm compared to the signed regressor is due to the same reasons mentioned above, as the magnitude of the correction term is affected.