EE3-08 Advanced Signal Processing
Coursework Report

Nicolas Mohnblatt
CID: 00860901

March 25, 2016

Contents

Chapter 1

Random Signals and Stochastic Processes

1.1 Statistical estimation

1. The random variable X follows a uniform distribution over the interval $[0, 1]$. Therefore its probability density function (pdf) is described by equation 1.1:

$$pdf_X(x) = \begin{cases} 1, & \text{for } x \in [0,1] \\ 0, & \text{otherwise} \end{cases} \tag{1.1}$$

We can calculate the theoretical mean m to be 0.5 by applying the definition as follows:

$$m = E(X) = \int_{-\infty}^{+\infty} x pdf_X(x) dx = 0.5 \tag{1.2}$$

The sample mean found using MATLAB is $\hat{m} = 0.502215306352569$. In this case the error is smaller than 1% and can be reduced by using more samples for the computation.

2. Let us repeat this process for the standard deviation σ:

$$\sigma = \sqrt{Var(X)} = \sqrt{E(X^2) - E(X)^2} \tag{1.3}$$

$$E(X^2) = \int_{-\infty}^{+\infty} x^2 pdf_X(x) dx = \frac{1}{3} \tag{1.4}$$

Therefore, substituting 1.4 and 1.2 into 1.3:

$$\sigma = \sqrt{\frac{1}{3} - \frac{1}{4}} = \frac{\sqrt{3}}{6} \simeq 0.28868 \tag{1.5}$$

Using the MATLAB command `std` we find that the sample standard deviation for one run of the script evaluated to $\hat{\sigma} = 0.288760597317148$. Just as in the previous case, the sample value is within one percent of the theoretical value. Once again, increasing the number of samples reduces the error.

3. We will now investigate an ensemble of ten realisations of the two experiments above, which we will denote $\mathbf{x}_{1:10}$. For each realisation we will compute the sample mean and sample standard deviation and plot them in Figure 1.1 against the theoretical values:

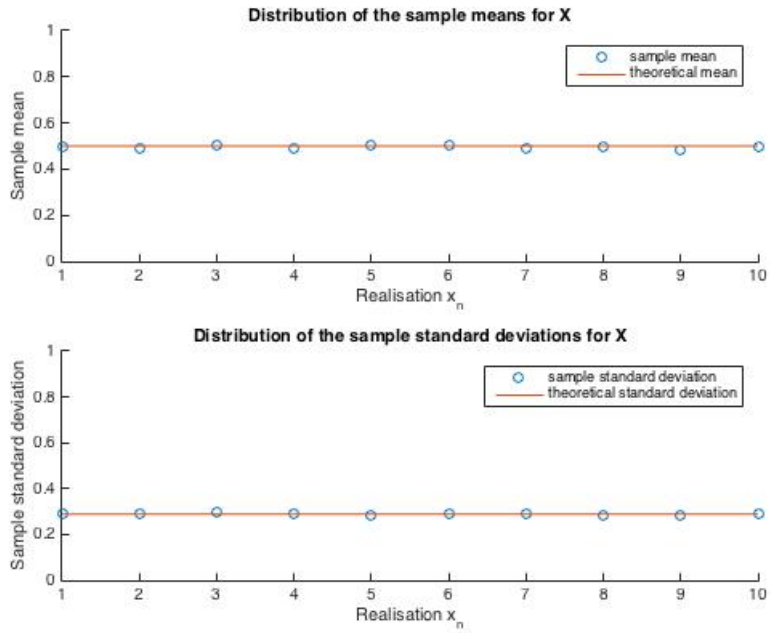
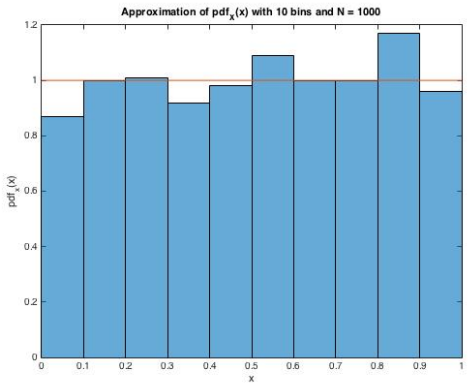


Figure 1.1: Comparison of the theoretical and sample statistical properties of the ensemble $\mathbf{x}_{1:10}$

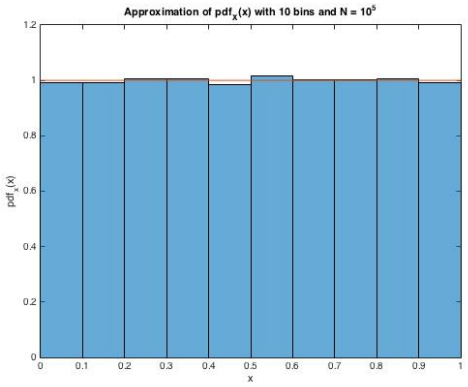
As we can see, the sample statistics are almost on the line of the theoretical statistics. We can therefore conclude that both of our estimators are unbiased.

4. Finally, we will attempt to approximate the probability density function of X. All the following figures are normalised histograms of $\mathbf{x}$ in which we vary the number of bins and the number of samples (see Figure 1.2).

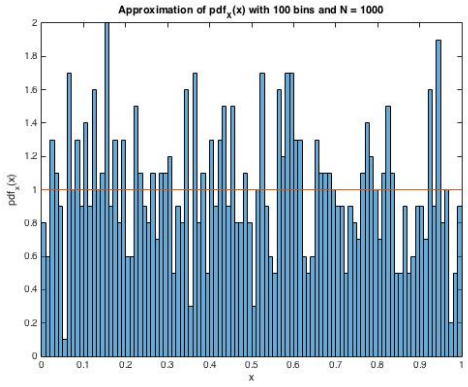
   In order to yield a better approximation of the pdf, we need to have a high sample per bin ratio. Having a smaller number of bins allows for a more loose grouping of the realisations, effectively reducing the approximation error. To obtain a plot of the exact theoretical pdf, we would need an infinite number of samples, arranged into a infinite number of bins.



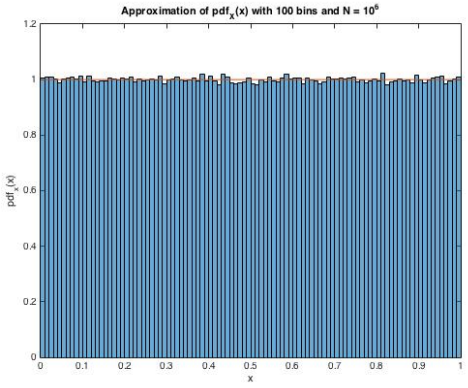
(a) 10 bins and $N = 10^3$



(b) 10 bins and $N = 10^5$



(c) 100 bins and $N = 10^3$



(d) 100 bins and $N = 10^6$

Figure 1.2: Approximation of the pdf of X with varying numbers of samples and bins

5. Let us now apply the same process to a random variable Z that follows a normal distribution. Inherently, the theoretical values for its means and standard deviation are $m = 0$ and $\sigma = 1$. By applying the MATLAB functions `mean` and `std` to one thousand sample realisations of Z, we obtain the following sample statistics:

$$\hat{m} = 0.003822319627605 \quad \text{and} \quad \hat{\sigma} = 0.983481496944549 \tag{1.6}$$

Computing the same statistics over an ensemble of 10 realisations ($\mathbf{z}_{1:10}$), we notice once again that our estimators are unbiased, as can be seen in Figure 1.3 below.

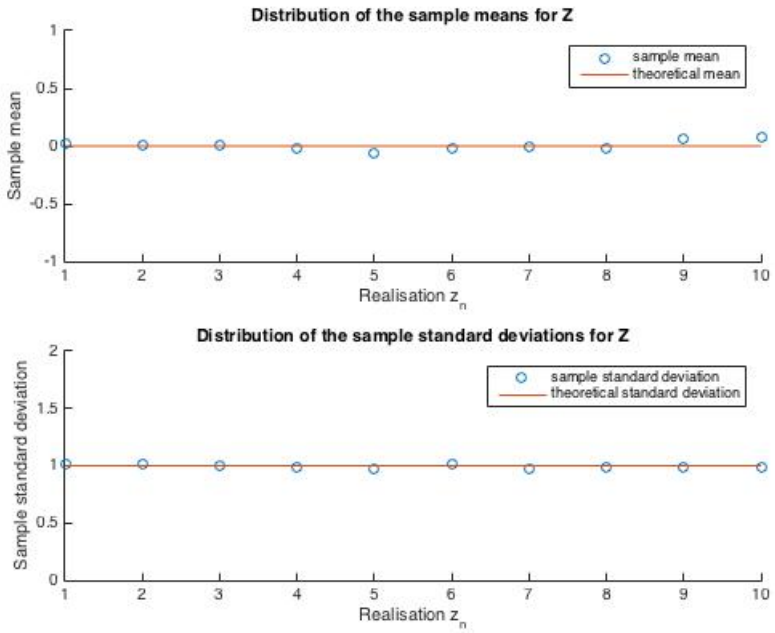
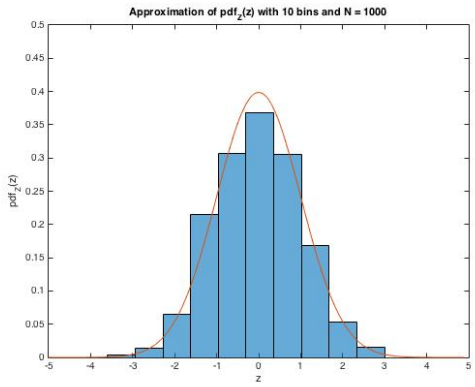


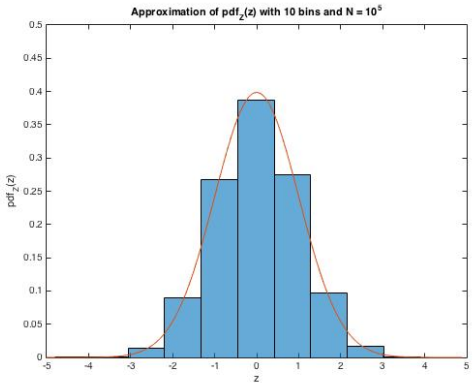
Figure 1.3: Comparison of the theoretical and sample statistical properties of the ensemble $\mathbf{z}_{1:10}$

Finally, let us approximate the pdf of Z. Once again, Figure 1.4 shows that as we increase the samples-per-bin ratio, our approximation improves. Indeed, as we increase the number of bins but keep as many samples in each bin, our approximation becomes closer to a continuously distributed function.

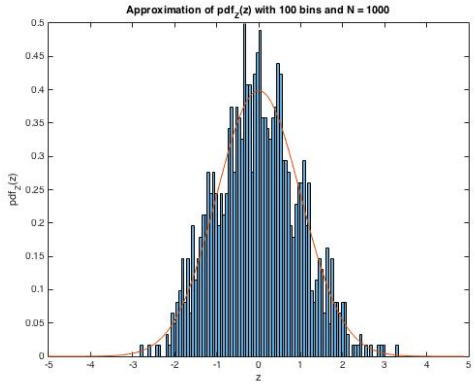
In this case, it seems that using a greater number of samples for a low number of bins reduces the quality of our approximation (see Figures 1.4a and 1.4b). This is due to the fact that the pdf of Z is not bounded. With a greater number of samples, there are higher chances that one realisation will have a very large magnitude. In order to account for this realisation, we need an extra bin. Effectively, we are creating a bin for only a few elements, leaving one less bin to approximate the rest of the function.



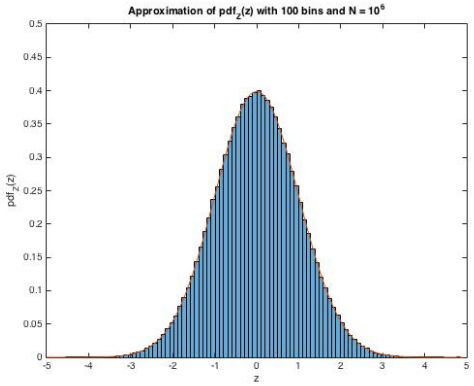
(a) 10 bins and $N = 10^3$



(b) 10 bins and $N = 10^5$



(c) 100 bins and $N = 10^3$



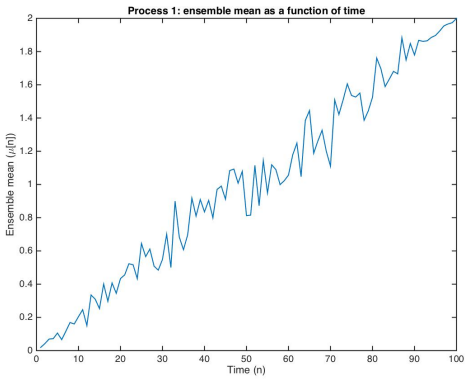
(d) 100 bins and $N = 10^6$

Figure 1.4: Approximation of the pdf of X with varying numbers of samples and bins

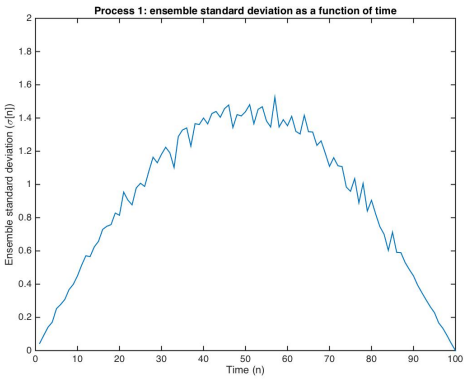
1.2 Stochastic processes

In this section, we are interested in studying three random processes that have been described using MATLAB code (see assignment). We will discuss their stationarity and ergodicity before studying them theoretically.

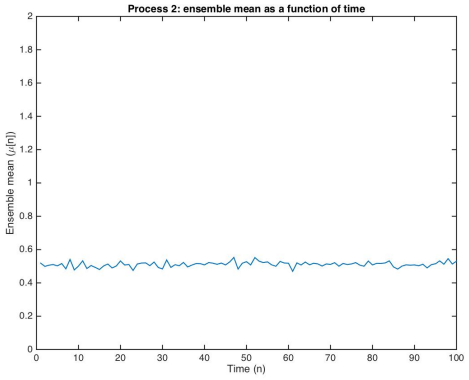
1. Figure 1.5 shows plots of the ensemble means and standard deviations as a function of discrete time for each of the three processes of interest. Process 1 has a time-varying mean, therefore it is clearly not stationary. Processes 2 and 3 seem to have a constant mean and a constant standard deviation. They are both likely to be wide-sense stationary (at the least).



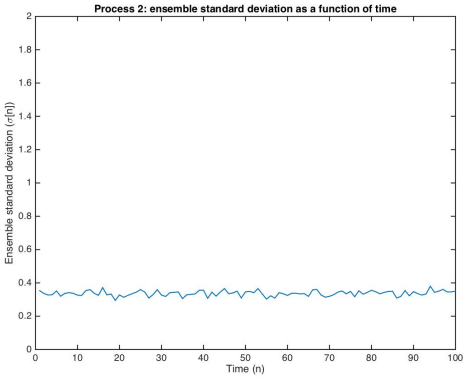
(a) Process 1 - Ensemble mean



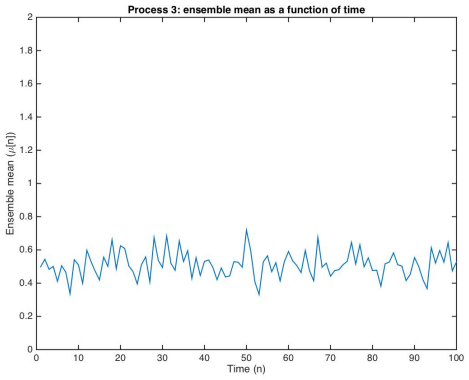
(b) Process 1 - Ensemble standard deviation



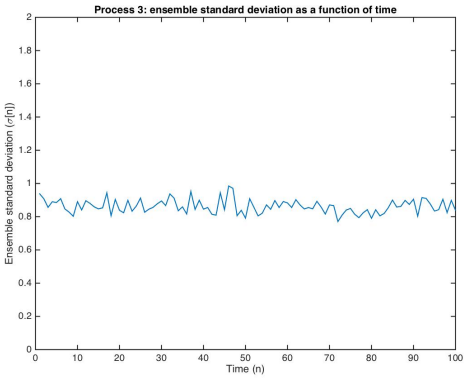
(c) Process 2 - Ensemble mean



(d) Process 2 - Ensemble standard deviation



(e) Process 1 - Ensemble mean



(f) Process 3 - Ensemble standard deviation

Figure 1.5: Ensemble mean and standard deviations for each of the three processes

2. The two tables below present each process's mean and standard deviation computed over each realisation. Comparing these results to the ensemble statistics allows us to comment on ergodicity. Firstly, Process 2's mean varies with each realisation which clearly states that it is not ergodic. Then, although Process 1 has a relatively constant mean from one realisation to another, this mean is very far from the time-varying ensemble mean seen in Figure 1.5a. Therefore, Process 1 is not ergodic either. Finally, Process 3 has the same mean over each of the four realisations and this value corresponds to what we have observed in Figure 1.5e: it is ergodic.

Realisation	Process 1	Process 2	Process 3
1	10.0432	0.3034	0.5319
2	10.0407	0.0865	0.4822
3	10.0765	0.1151	0.5073
4	10.0049	0.6304	0.5396

Table 1.1: Mean for each of the four realisations of each process

Realisation	Process 1	Process 2	Process 3
1	5.9082	0.1252	0.8697
2	5.8407	0.2875	0.8608
3	5.8756	0.0453	0.8481
4	5.8550	0.0385	0.8566

Table 1.2: Standard deviation for each of the four realisations of each process

3. Let us now write a mathematical description of the three processes in order to calculate their theoretical means and variances. The computations associated to each process output an $M \times N$ matrix in which each row represents one of its realisations. Let $Y[n]$ be a random process such that at any time instant n_0, $Y[n_0]$ is a uniformly distributed random variable within the range $[-0.5, 0.5]$, and A and B both be uniformly distributed random variables over the range $[0, 1]$, the three processes of interest can be expressed as:

$$RP_1[n] = 5\sin\left[\frac{\pi n}{N}\right] Y[n] + \frac{2n}{100} \tag{1.7}$$
$$RP_2[n] = AY[n] + B \tag{1.8}$$
$$RP_3[n] = 3Y[n] + 0.5 \tag{1.9}$$

Given that $Y[n]$ is evenly distributed over an interval centred around 0, it is a zero-mean process. Furthermore, the process $Y[n]$ is independent from the variable A and B. Therefore when computing the means of Processes 1-3, any term factored by $Y[n]$ will evaluate to 0. Consequently, the means of each of these processes are:

$$E(RP_1[n]) = \frac{2n}{100} \tag{1.10}$$
$$E(RP_2[n]) = E(B) = 0.5 \tag{1.11}$$
$$E(RP_3[n]) = 0.5 \tag{1.12}$$

Finally let us compute their variances using the identity $Var(aX + b) = a^2 Var(X)$. The computation of 1.14 involves slightly more complicated algebraic manipulations that take advantage of the fact that all three random variables are independent.

$$Var(RP_1[n]) = 25\sin^2\left[\frac{\pi n}{N}\right] Var(Y[n]) = \frac{25}{12}\sin^2\left[\frac{\pi n}{N}\right] \tag{1.13}$$
$$Var(RP_2[n]) = E(A^2)E(Y[n]^2) + Var(B) = \frac{1}{9} \tag{1.14}$$
$$Var(RP_3[n]) = 9\,Var(Y[n]) = \frac{9}{12} \tag{1.15}$$

Comparing these descriptions to the results obtained in 1.5, it seems that the plots could indeed come from our models. From the models we can confirm the guesses we have made

earlier about the processes' ergodicity and stationarity. These conclusions are summarised in table 1.3 below.

Process	Ergodicity	Stationarity
RP_1	No	No
RP_2	No	Yes
RP_3	Yes	Yes

Table 1.3: Standard deviation for each of the four realisations of each process

1.3 Estimation of probability distributions

1. In this task we will attempt to estimate the probability density function of a given set of input samples. The following figure shows the output of the MATLAB script for an input consisting of 500 samples of a Gaussian random variable.

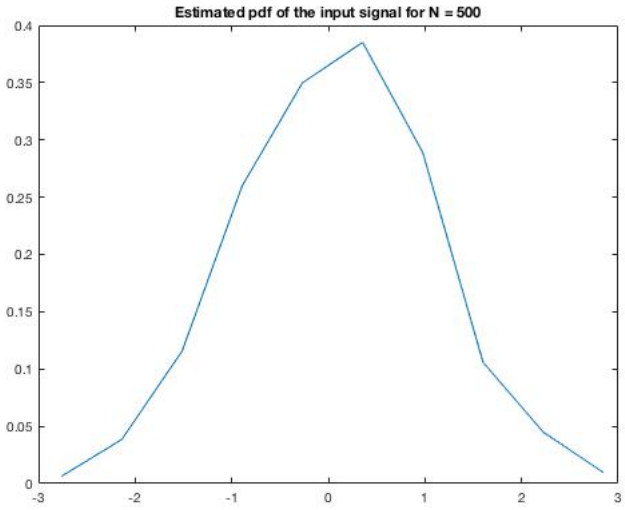
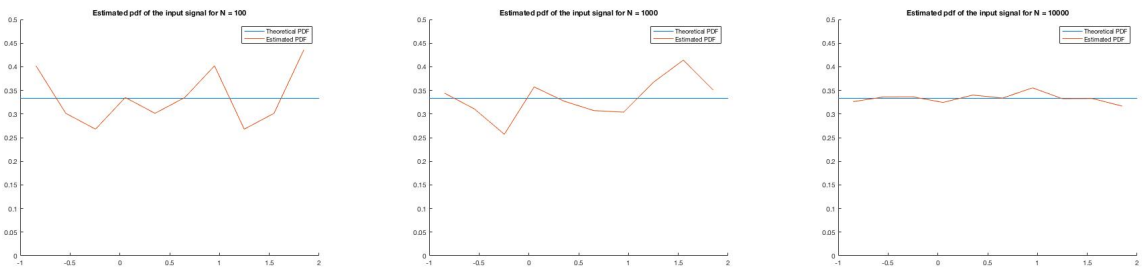


Figure 1.6: Estimation of a Gaussian pdf

2. Let us now apply this script to the only process from the previous part that was ergodic and stationary, namely Process 3. Each plot in Figure 1.7 shows the estimated pdf as well as the theoretical value as shown in the legend. As one would expect, the quality of the estimation improves with the size of the of the input signal.



(a) $N = 100$ (b) $N = 1000$ (c) $N = 10000$

Figure 1.7: Estimation of the pdf of Process 3

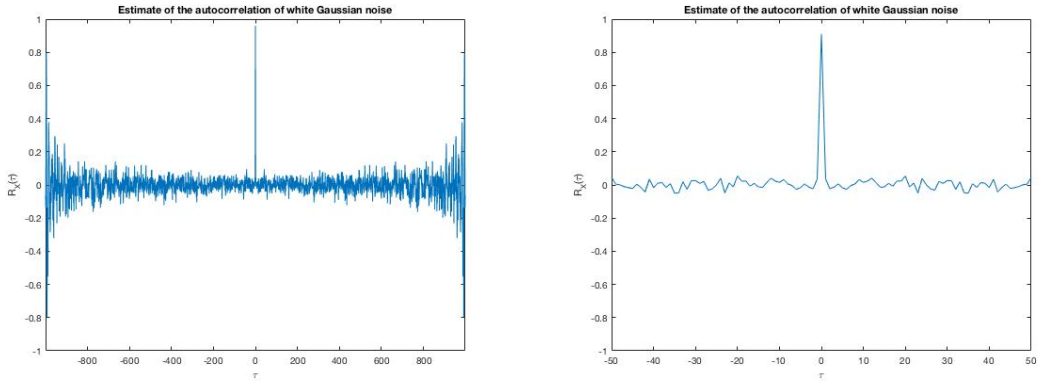
3. This method relies on the fact that all the samples of the input signal follow the same pdf. For a non-stationary process, the function `pdf` would not yield an appropriate approximation. An example of such a process would be a 1000-sample-long signal of which the mean changes from 0 to 1 after the point N = 500. The first 500 samples will be centred around zero, while the next 500 will be centred around 1. Allocating samples into bins as we do in the `pdf` function is therefore irrelevant. To estimate a non-stationary process, we would need to produce multiple estimates for each period during which the process is stationary: in the example discussed we would need one estimate for the first 500 samples and another for the rest of the signal.

Chapter 2

Linear Stochastic Modelling

2.1 ACF of uncorrelated sequences

1. In this first exercise we will plot an estimate of the autocorrelation function of a 1000-sample realisation of white Gaussian noise (WGN). Figure 2.1a shows the full estimation for all possible lag values whereas Figure 2.1b shown a zommed-in version of the same plot. The resulting plot shows that for a time delay of 0, we estimate that two samples of the input signal are almost perfectly correlated. However, for any time delay their correlation is almost 0 (except for high lags see point 3 of this section). This is what we would expect since samples of WGN are independent and therefore uncorrelated. The symmetry comes from the fact that for an ACF, τ represents the time delay between a signal and a copy of itself. Whether we delay one copy or the other makes no difference. Therefore the ACF is inherently symmetric with respect to the y-axis.



(a) Estimate for $\tau \in [-999, 999]$ (b) Estimate for $\tau \in [-50, 50]$

Figure 2.1: Estimate of the ACF of a 1000-sample realisation of WGN

2. Zooming in on the ACF (see Figure 2.1b), we see what looks like an impulse at 0 and noise otherwise. This is very close to what we would expect to see in theory as explained above. Notice however that as we zoom out, the amplitude of the ACF for large lags becomes non-negligible as compared to the impulse at 0.

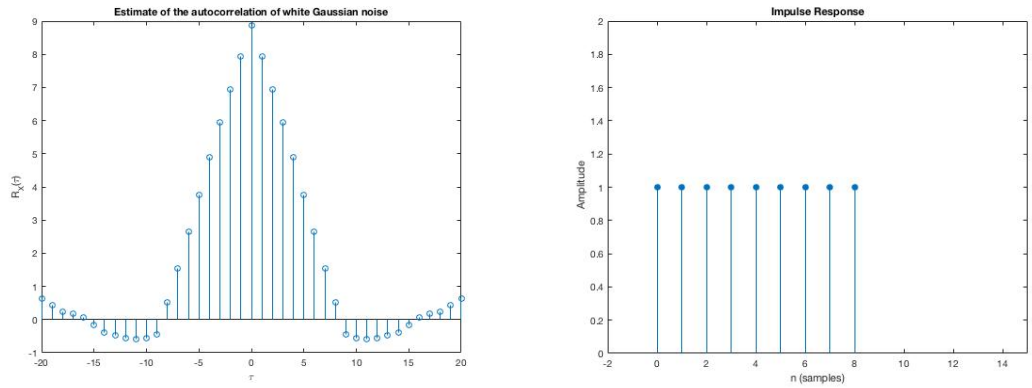
3. This effect can be explained by taking a closer look at the equation we use to estimate the autocorrelation function:

$$\hat{R}_X(\tau) = \frac{1}{N - |\tau|} \sum_{n=0}^{N-|\tau|-1} x[n]x[n+\tau], \quad \tau = -N+1, \ldots, N-1 \tag{2.1}$$

As can be seen above, for large $|\tau|$, the factor $\frac{1}{N-|\tau|}$ approaches its maximum value which is 1. For other values of $|\tau|$, this factor will provide strong attenuation. Because we are removing the attenuation for large lag values, any small discrepancy at the input will be picked up in our calculations and will appear in the ACF. In order to have a reliable estimate of the ACF, we will us the empirical bound $|\tau| < \frac{N}{2}$.

2.2 ACF of correlated sequences

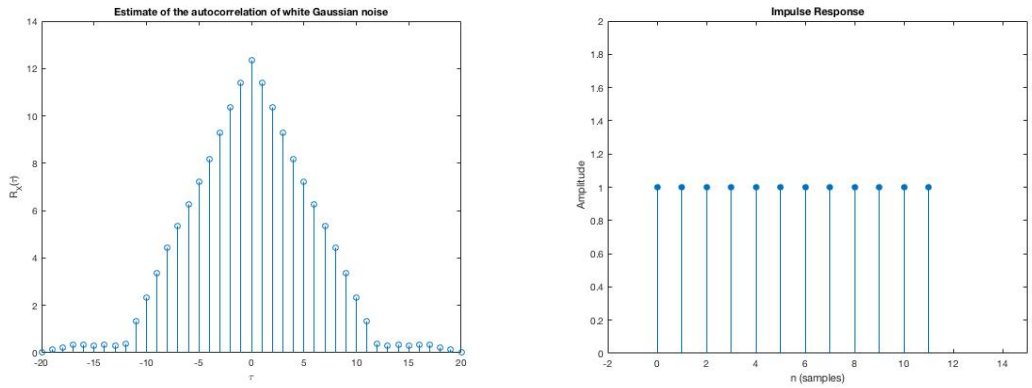
1. Let us now proceed with a similar analysis for a correlated sequence: in this case, WGN passed through a moving average (MA) filter. The following figure shows the autocorrelation of such a signal as well as the impulse response of the filter. The ACF for low lags ($|\tau| < 20$) resembles a triangle with a maximum at 0 and intercepts with the x-axis when the time delay is equal to the number of filter coefficients. This is the kind of signal we would expect when performing the autocorrelation of the filter's impulse response.



(a) ACF of WGN through MA filter (b) Impulse response of the MA filter

Figure 2.2: ACF of a correlated sequence: WGN through an 8^{th} order FIR

As we can see from figure 2.3, it is indeed the filter order that determines the shape of the output's ACF. Furthermore, by the nature of the filter itself, we are computing a moving average over the past few samples. In this case all our filter coefficients are 1 therefore - assuming there are n coefficients - we are adding together the past n samples. Computing the local mean would simply require to divide the filter output by n.



(a) ACF of WGN through MA filter (b) Impulse response of the MA filter

Figure 2.3: ACF of a correlated sequence: WGN through an 11^{th} order MA

2. The autocorrelation of the stochastic process $Y[n]$ can be expressed as:

$$R_Y(\tau) = R_X(\tau) * R_h(\tau) \tag{2.2}$$

If $X[n]$ is an uncorrelated process, its autocorrelation can be expressed as a Dirac-delta function[1] at zero. Therefore the ACF of $Y[n]$ is equal to the ACF of the filter's impulse response as shown in equation 2.3.

$$R_Y(\tau) = \int_{-\infty}^{+\infty} R_h(t)\delta(t-\tau)dt = R_h(-\tau) = R_h(\tau) \tag{2.3}$$

2.3 Cross-correlation function

1. Let us now plot the cross-correlation function (CCF) of the WGN with the output of the MA filter from the previous section (see Figure 2.4).

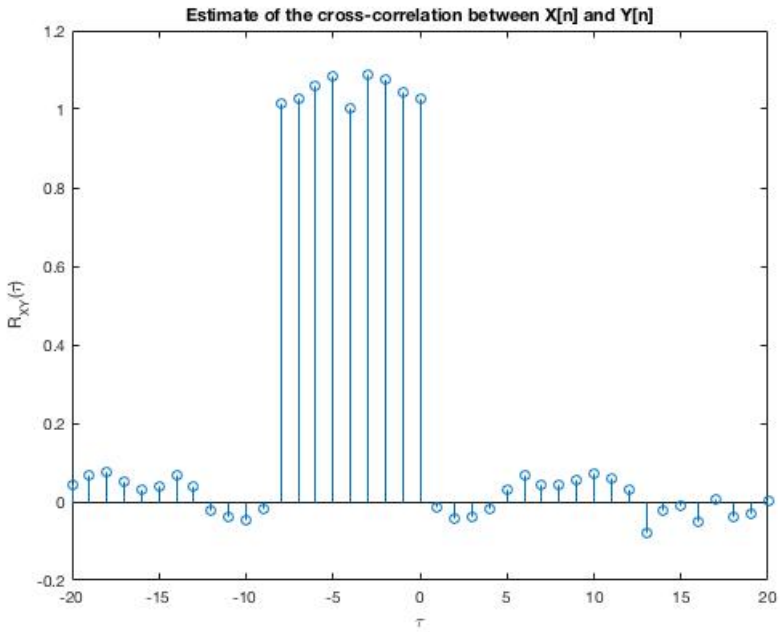


Figure 2.4: Cross-correlation function of WGN with the output of the $8th$ order MA filter

This CCF function closely resembles the filter impulse response. Indeed in theory we would expect the CCF R_{XY} to be a time-reversed version of the impulse response if $X[n]$ is an uncorrelated process as demonstrated by the equation below:

$$R_{XY}(\tau) = \int_{-\infty}^{+\infty} \delta(t)h(t-\tau)dt = h(-\tau) \tag{2.4}$$

2. Changing the order of the filter will change the impulse response $h(\tau)$ and should therefore also change the cross-correlation between $X[n]$ and $Y[n]$. We can see the expected effect of the change in Figure 2.5. Therefore, inputting an uncorrelated signal into a filter and taking the CCF of the input with the output is a means by which we can determine the impulse response of an unknown system.

[1] Which we will denote by $\delta(.)$

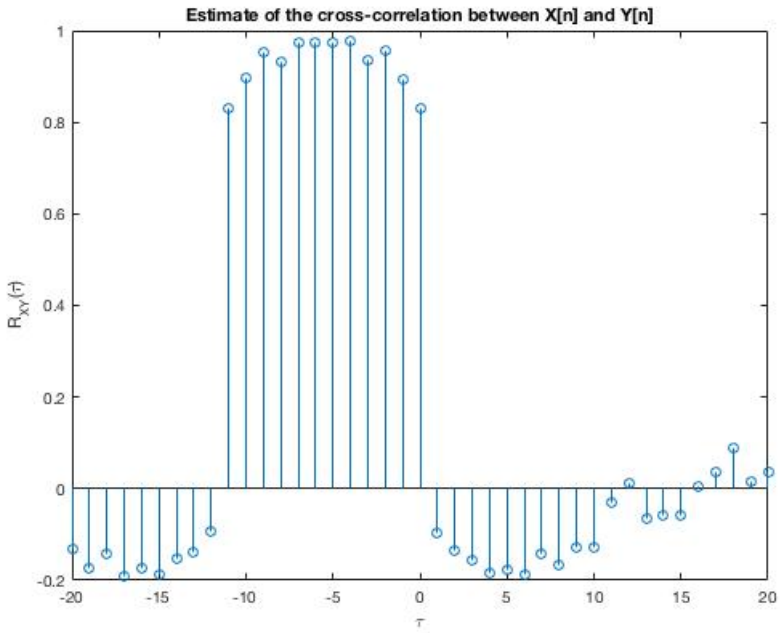


Figure 2.5: Cross-correlation function of WGN with the output of the $11th$ order MA filter

2.4 Autoregressive modelling

In this section we will investigate autoregressive processes (AR) by using the example of the sunspot data embedded in MATLAB

1. Let us first investigate the stability of an AR(2) model. We will do so by randomly generating 100 sets of coefficients a_1 and a_2 using the following distributions:

$$a_1 \sim U(-2.5, 2.5) \quad \text{and} \quad a_2 \sim U(-1.5, 1.5) \tag{2.5}$$

The AR(2) process that will be studied is a 1000-sample long realisation of:

$$x[n] = a_1 x[n-1] + a_2 x[n-2] + w[n], \qquad w[n] \sim N(0, 1) \tag{2.6}$$

By plotting a point for each pair of coefficients that yield a converging process we obtain the following graph in Figure 2.6. As is explicitly shown by the lines drawn onto the plot, the coefficients converge into a triangular region defined by the points $(1, 1)$, $(2, -1)$ and $(-2, -1)$.

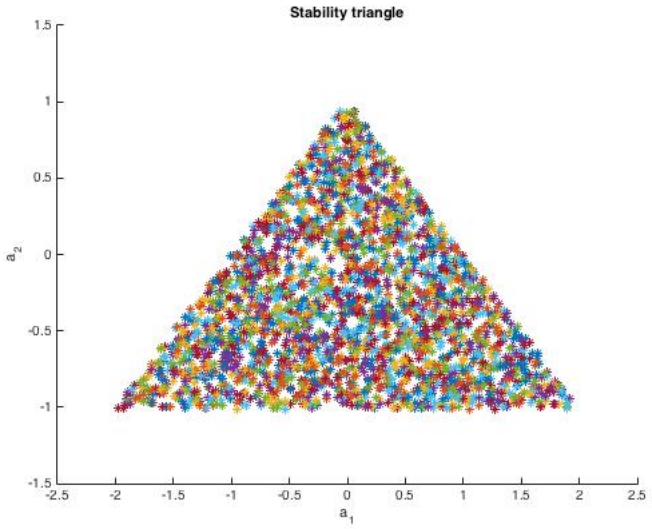


Figure 2.6: Plot of the coeffecient pairs that yield a stable process

This "stability triangle" can be shown to exist by applying the **Jury stability criterion**[2] which states that there are three necessary conditions to the stability of a discrete-time system. Our system's transfer function can be written as:

$$H(z) = \frac{X(z)}{W(z)} = \frac{1}{z^2 - a_1 z - a_2} \tag{2.7}$$

Its characteristic polynomial is therefore $P(z) = z^2 - a_1 z - a_2$. In order for the system to be stable, we need to verify that:

$$P(1) > 0 \tag{2.8}$$
$$(-1)^2 P(-1) > 0 \tag{2.9}$$
$$|-a_2| < 1 \tag{2.10}$$

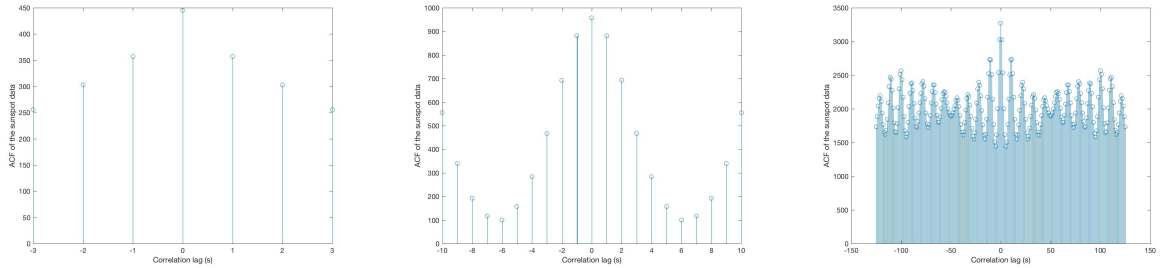
After replacing with our system's values we get the equivalent set of conditions:

$$a_2 < 1 - a_1 \tag{2.11}$$
$$a_2 < 1 + a_1 \tag{2.12}$$
$$|a_2| < 1 \tag{2.13}$$

Which are the three equations that define the triangle seen in 2.6. Given that our system is of the second order we do not need to build the Jury array and there are no other conditions for stability than the ones enunciated above.

2. Let us now investigate the real world sunspot series. The figure below shows its ACF plotted for different signal lengths for time lags within the empirical bound discussed in this chapter. For short signals, the ACF is a triangle. As we extend the signal, the ACF exhibits some pseudo-periodic properties while being raised by a constant.

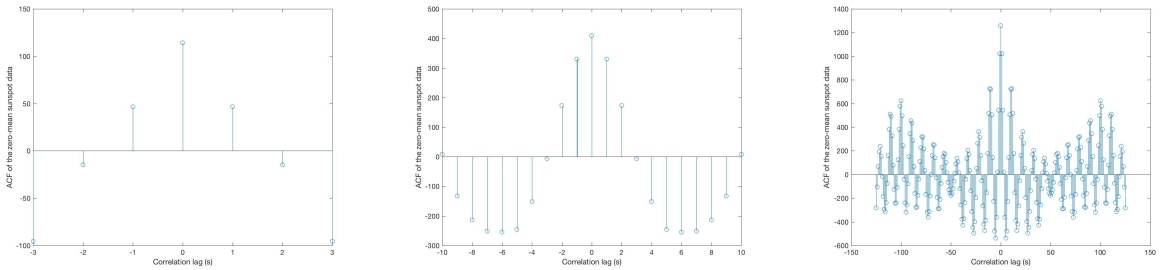
[2] `https://en.wikibooks.org/wiki/Control_Systems/Jurys_Test`



(a) $N = 5$ (b) $N = 20$ (c) $N = 250$

Figure 2.7: ACF of the sunspot series

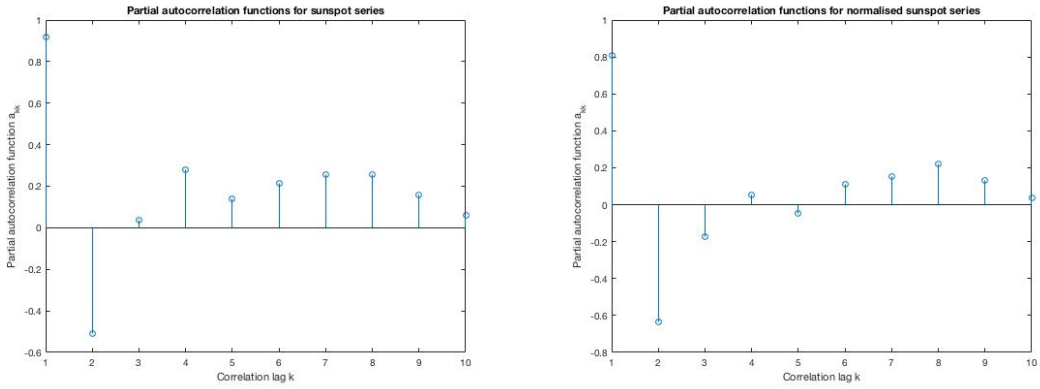
To further investigate the effect of the mean, we will plot the ACF of the sunspot series from which we have removed the mean (see Figure 2.8). Removing the mean has centred the ACF around 0. The effect we see is due to the fact that computing the ACF of a non-zero-mean process, is equivalent to adding the ACF of a zero-mean process with the ACF of a constant. The ACF of a constant is a triangle, however the longer the data, the flatter the triangle will look as we can observe in Figure 2.7.



(a) $N = 5$ (b) $N = 20$ (c) $N = 250$

Figure 2.8: ACF of the sunspot series

3. Using the Yule-Walker equations we can estimate the coefficients of the AR model for different possible orders. Plotting the partial autocorrelation functions allows us to determine the correct order to use to estimate the system. Figure 2.9 shows that for lags above $k = 2$, the PAC are close to zero. This indicates that the sunspot series arises from an AR(2) process. Notice that we obtain different plots for normalised and non-normalised data. However, in both cases the coefficients under the maximum lag are almost equal - the main differences arising from higher lag values which in theory should all be zero.



(a) Sunspot series (b) Normalised sunsport series

Figure 2.9: Partial autocorrelation function of the time series

4. Let us now determine the optimal filter order using two different mathematical descriptions: the minimum description length (MDL) and the Akaike criterion (AIC). Both processes are defined as,

$$\textbf{MDL} \quad p_{opt} = min_p \left[\log(E) + \frac{p \log N}{N} \right] \tag{2.14}$$

$$\textbf{AIC} \quad p_{opt} = min_p \left[\log(E) + \frac{2p}{N} \right] \tag{2.15}$$

Applying these criteria to our model estimations we obtain the following plot (see Figure 2.10).

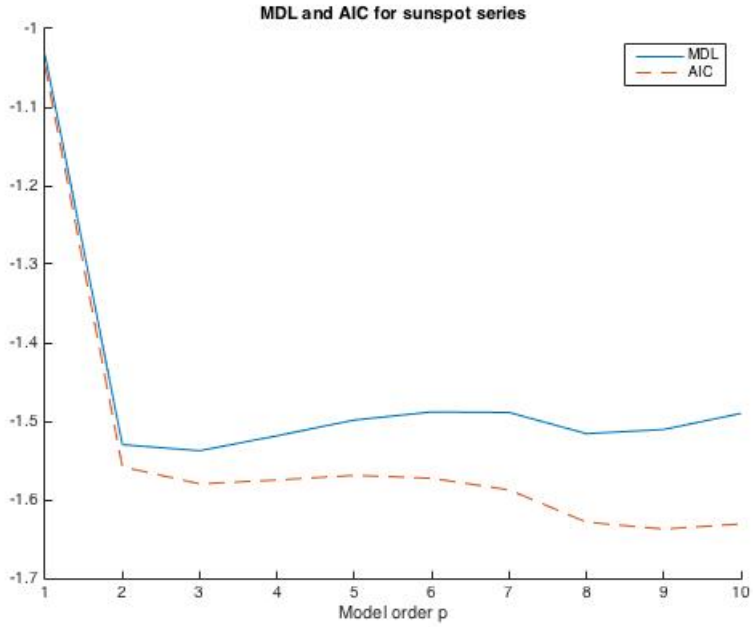


Figure 2.10: MDL and AIC for sunspot series

Using the MATLAB `min` function the minimums of these function can be computed to be $p = 3$ for the MDL and $p = 9$ for the AIC. In both cases, the mathematical criteria chooses

a higher order than the process' original order. Although this we be more computationally complex, the approximation will be closer to the data.

5. Here we will try to predict future values of the process using our AR model using the following equation:

$$\hat{x}[n+m] = a_1 x[n-1] + \ldots + a_p x[n-p] \tag{2.16}$$

where m is the prediction horizon. This equations shows that our prediction will only rely on the coefficients and past inputs. Furthermore, the prediction horizon does not affect the right-hand side of 2.16, therefore the values computed will always be the same with a time shift equal to m. Figure 2.11 shows the prediction for a horizon of $m = 1$ using models AR(1), AR(2) and AR(10).

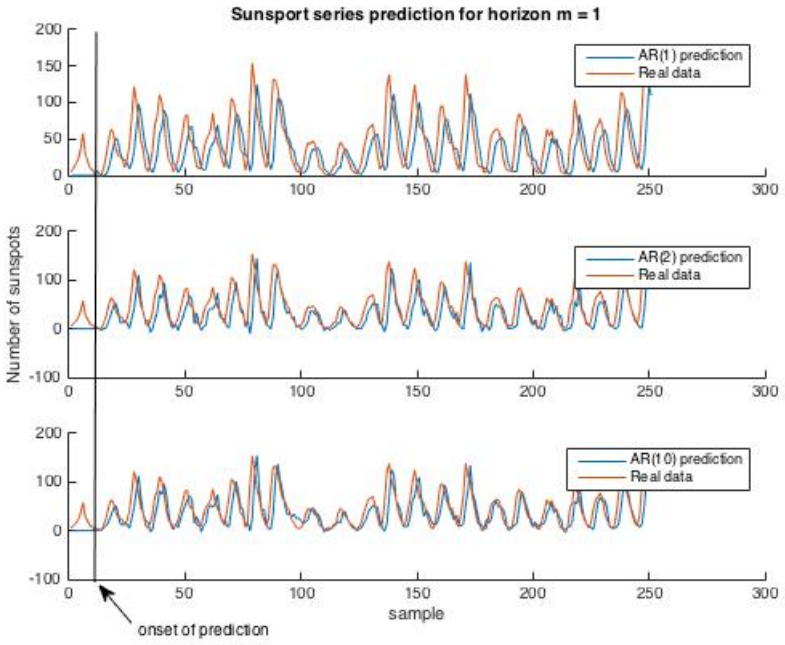


Figure 2.11: Prediction of sunspot series for $m = 1$

As explained above, predicting for greater horizons will only result in shifting the generated series to the right. This will introduce prediction error that will rise as the horizon becomes close to half the data's period, and then diminish again when the prediction is back in phase with the real data. When the prediction is very out of phase, the interpolation error will become negligible when compared to the extrapolation error.

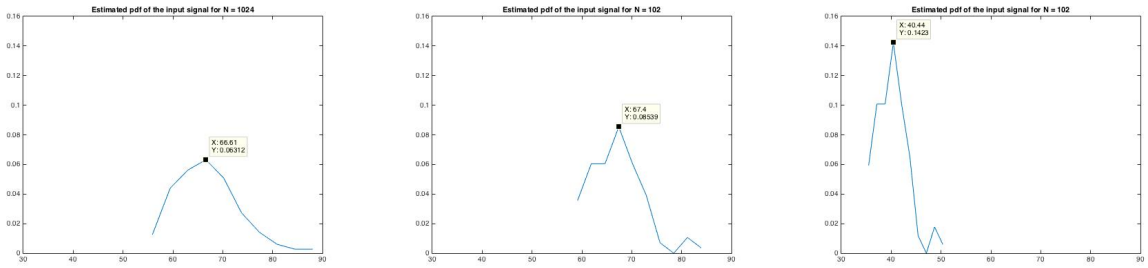
2.5 Real world signals: ECG from iAmp experiment

In this section we will be interested in estimating the pdf of a measured heart rate and investigate an AR model for this timeseries.

Heart rate probability density estimate (PDE)

a) Firstly, we will plot an estimate of the pdf of the heart rate as well as an estimate for two sequences built using averages over 10 samples of the heart rate (see equation 2.17)

$$\hat{h}[1] = \frac{1}{10}\sum_{i=1}^{10} \alpha h[i],\ \hat{h}[2] = \frac{1}{10}\sum_{i=11}^{20} \alpha h[i], \ldots \qquad \alpha = 0.6, 1 \tag{2.17}$$



(a) Original series (b) Averaged ($\alpha = 1$) (c) Averaged ($\alpha = 0.6$)

Figure 2.12: PDE of a unconstrained heart rate

b) While averaging every ten samples was supposed to yield a smoother estimate, we have reduced the number of collected points by a factor of 10. Therefore as we can see above, the estimate with no averaging is smoother and seems to follow a Gaussian distribution with a mean at $66.61bpm$.

The constant α acts a a scaling factor on our data which will therefore shift the mean and scale the standard deviation as shown by the identities

$$E(\alpha H[n]) = \alpha E(H[n]) \tag{2.18}$$
$$Var(\alpha H[n]) = \alpha^2 Var(H[n]) \tag{2.19}$$

where $H[n]$ is the random process that characterises the heart rate distribution. In this case when $\alpha < 1$, the mean is shifted to the left and the overall shape becomes thinner and higher.

AR modelling of heart rate

c) For the three trials of the iAmp experiment, we will plot the ACF of the obtained data (see Figure 2.13). The first two autocorrelation seem to have a peak centred around $\tau = 0$ and then cut-off after a certain lag value. Therefore we can assume that this data has been produced by a MA process. The third trial shows some periodicity in the ACF which is a characteristic of AR processes.

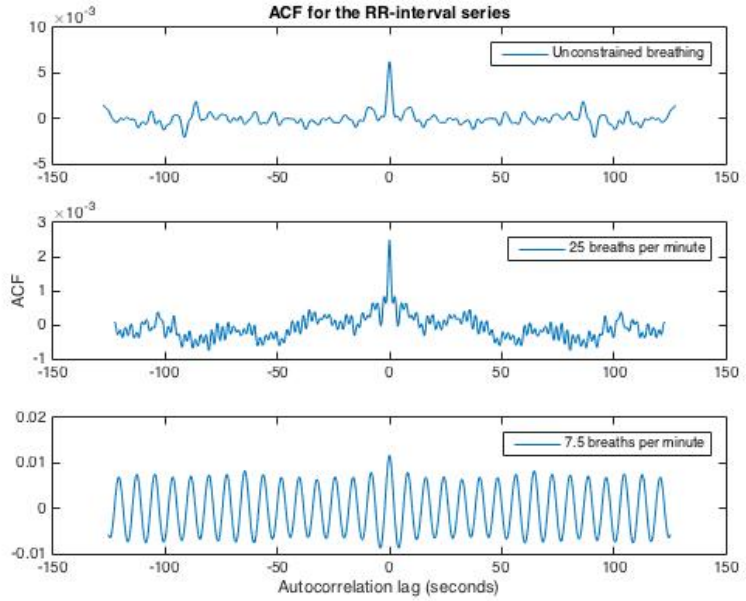
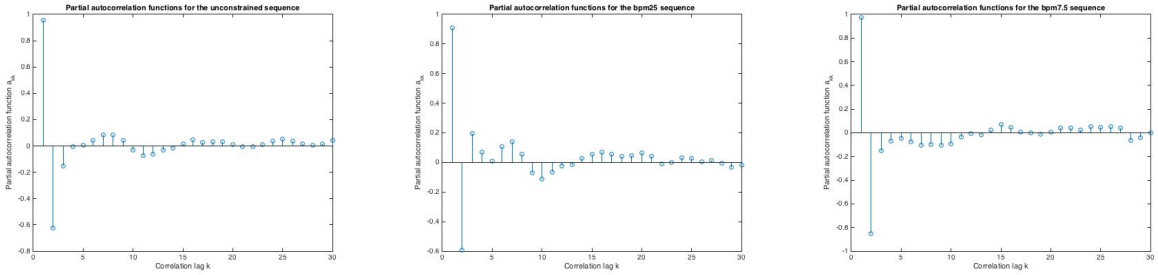


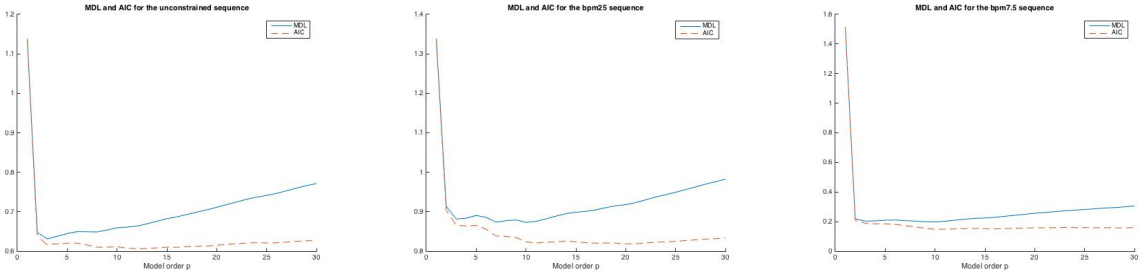
Figure 2.13: Autocorrelation functions for each of the measured heart rate series

d) Given the duality between FIR and IIR filters, we can produce an estimate of this process regardless of its nature using an AR(p) model. We will plot the PCF as well as the MDL and the AIC for each of the three sequences



(a) Original series (b) 25 breaths per minute (c) 7.5 breaths per minute

Figure 2.14: PCF of the heart rate series



(a) Original series (b) 25 breaths per minute (c) 7.5 breaths per minute

Figure 2.15: MDL and AIC of the heart rate series

From these plots, it seems that an AR(3) model would be optimal, yielding a close estimation without being too computationally intensive.

Chapter 3

Spectral Estimation and Modelling

1. The following plot was produced using the function `pgm` and shows the periodogram of an N-sample long realisation of WGN. The periodogram provides an estimate of the power spectral density (PSD) of a signal. Therefore in this case we should, in theory, observe a straight line of amplitude 1. Although this is not the case, the average of the periodogram seems to converge to 1 as we increase the number of samples.

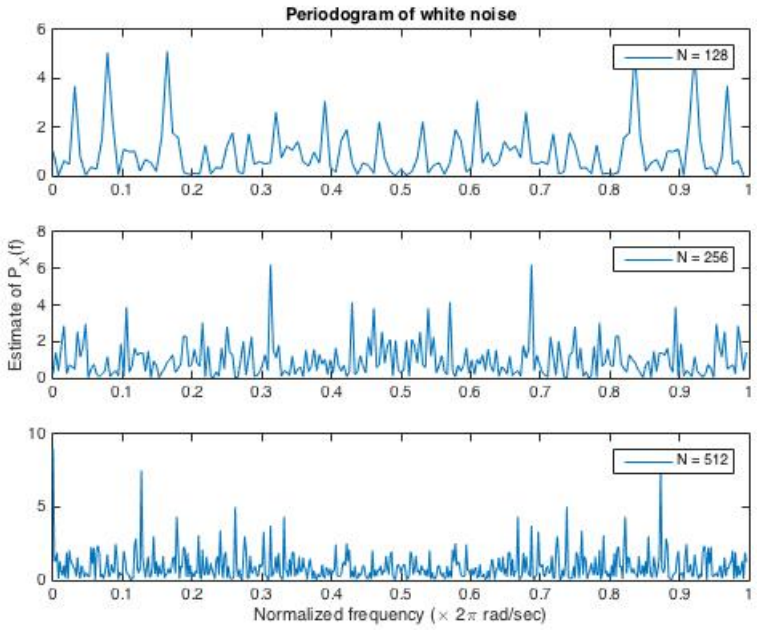


Figure 3.1: Periodogram of WGN for varying signal lengths

Notice that the periodogram is symmetric with respect to 0.5 since its computation is based on the discrete Fourier transform (DFT). Furthermore, making the input sequence longer does not seem to improve the estimate.

3.1 Averaged periodogram estimates

1. Here, we will try to filter the PSD in order to smooth it out as is shown in Figure 3.2. As we can see, the amplitudes of the peaks have decreased, yielding an estimate that is closer to the theoretical value discussed above. Notice also that the filtering has removed the symmetry observed in Figure 3.1.

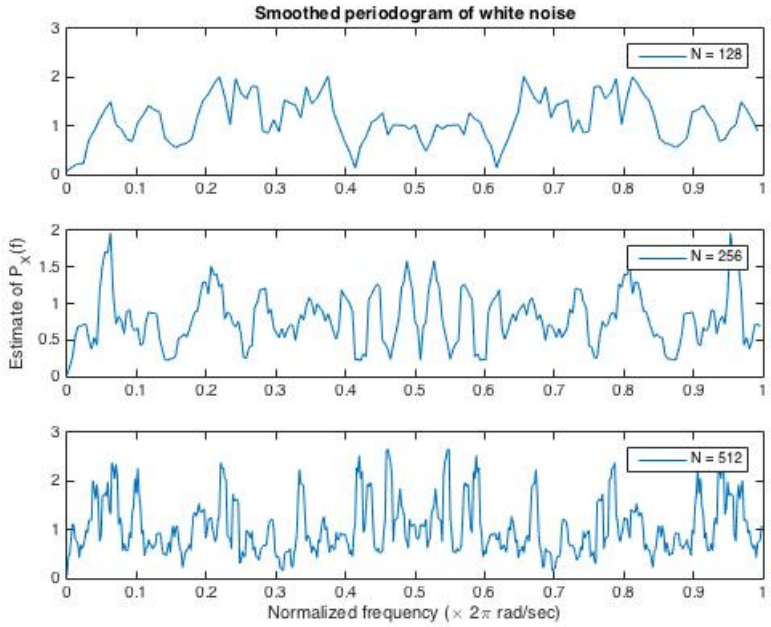


Figure 3.2: Filtered version of the PSD estimate of WGN

2. Repeating the non-smoothed experiment for 8 realisations of 128-sample long WGN segments, we notice that the locations of the spikes in the periodogram seem completely random - apart for the symmetry.

3. If we now average these eight realisations, we will be able to plot the averaged periodogram of WGN (see Figure 3.3).

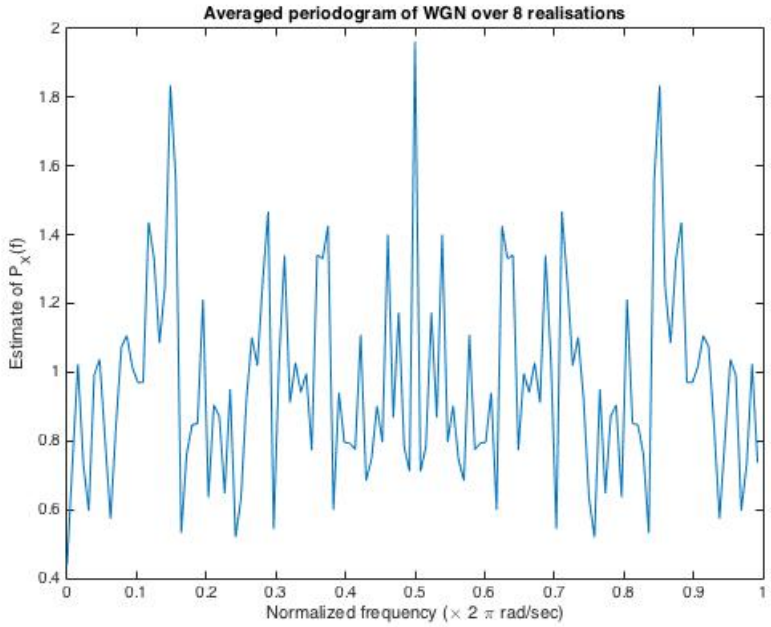
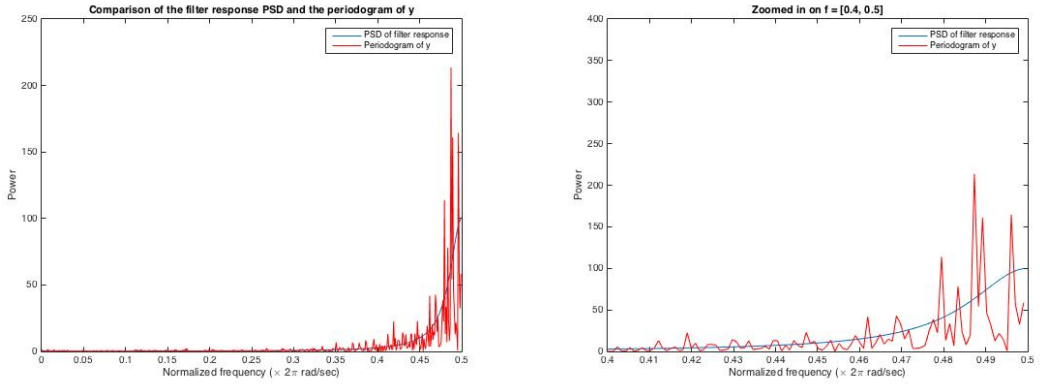


Figure 3.3: Averaged periodogram of WGN over 8 128-sample long realisations

Once again, we have reduced the amplitudes of the peaks and yielded an estimate that is closer to the theoretical value.

3.2 Spectrum of autoregressive processes

1. We will now investigate the spectra of autoregressive processes, first by plotting the expected PSD of the filter's frequency response as well as a estimate obtained with a periodogram (see Figure 3.4a).



(a) Theoretical PSD and periodogram

(b) Zoomed on $f = [0.4, 0.5]$

Figure 3.4: PSD of AR(1) model and periodogram-based estimate

The system we are studying is an infinite impulse response filter with a pole at $z = -0.9$ and a zero at $z = 0$. Therefore as the digital frequency approaches $\pi\, rad/sec$ we would expect the PSD to go towards infinity. That is indeed what we see in the above figure.

2. Although the periodogram follows the correct trend, there are substantial differences with the theoretical value. These can be explained by studying our estimator

$$\hat{P}_Y(f) = |H(f)|^2 \hat{P}_X(f) \tag{3.1}$$

As we have seen in the previous section, the output of the periodogram for a WGN input is not exactly 1 over the whole range of frequencies (i.e. if X is WGN, $\hat{P}_X(f) \neq P_X(f)$). Therefore the periodogram we obtain corresponds to the filter's PSD, modulated by variations similar to those observed in Figure 3.1

3. Figure 3.4b shows in more detail what happens over the interval $[0.4, 0.5]$, in which the periodogram estimate seems particularly wrong. Indeed, the input to the filter is not exactly WGN but has been multiplied by a rectangular window. In the frequency domain this operation corresponds to convolving the PSD with a $sinc^2$ function - which explains the 'bounces" that can be observed.

4. Let us now estimate the PSD using a specific model order (in this case $p = 1$). Using the autocorrelation function, we can estimate the filter coefficients as well as the standard deviation and finally compute an estimate of the PSD. This estimate is shown in Figure 3.5.

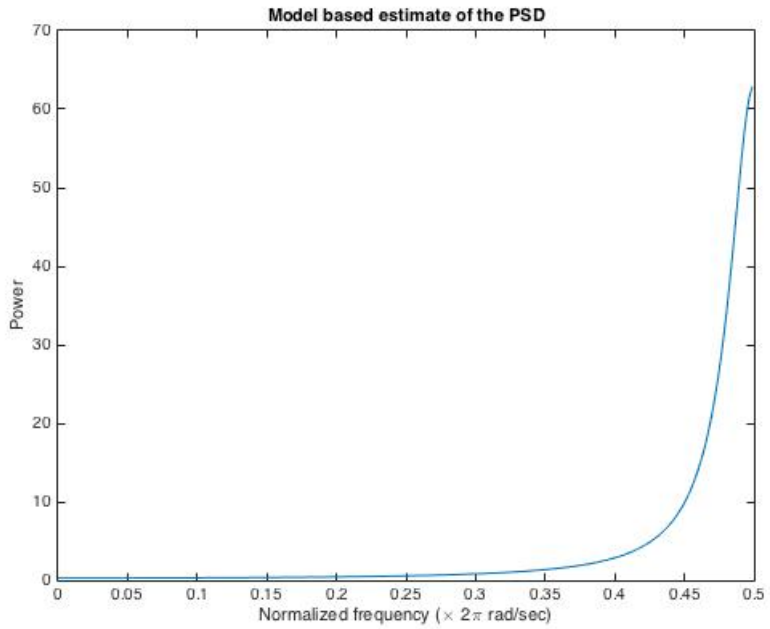
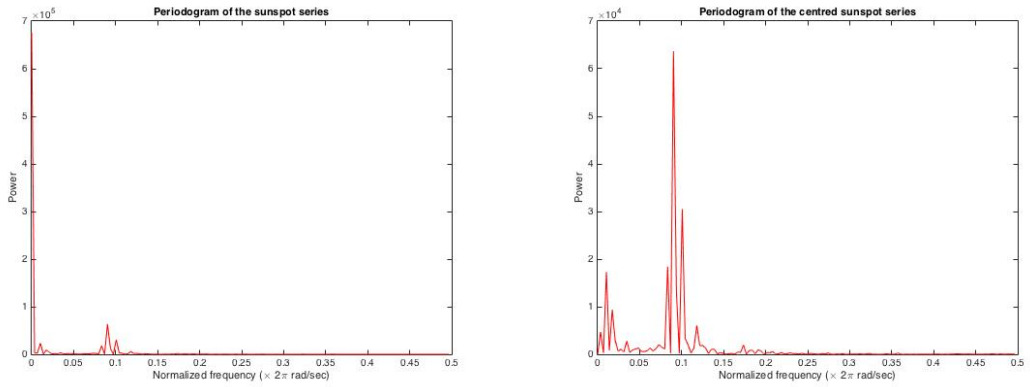


Figure 3.5: Model-based estimate of the PSD

This estimate is much smoother than the one obtained using the periodogram. This is partly due to the fact that we are no longer relying on the input to be a perfect realisation of WGN. Furthermore, making an assumption about the model means that we already have more information on the process than when plotting a periodogram. It is therefore logical that an estimate based on a more complete set of information would be closer to the actual PSD.

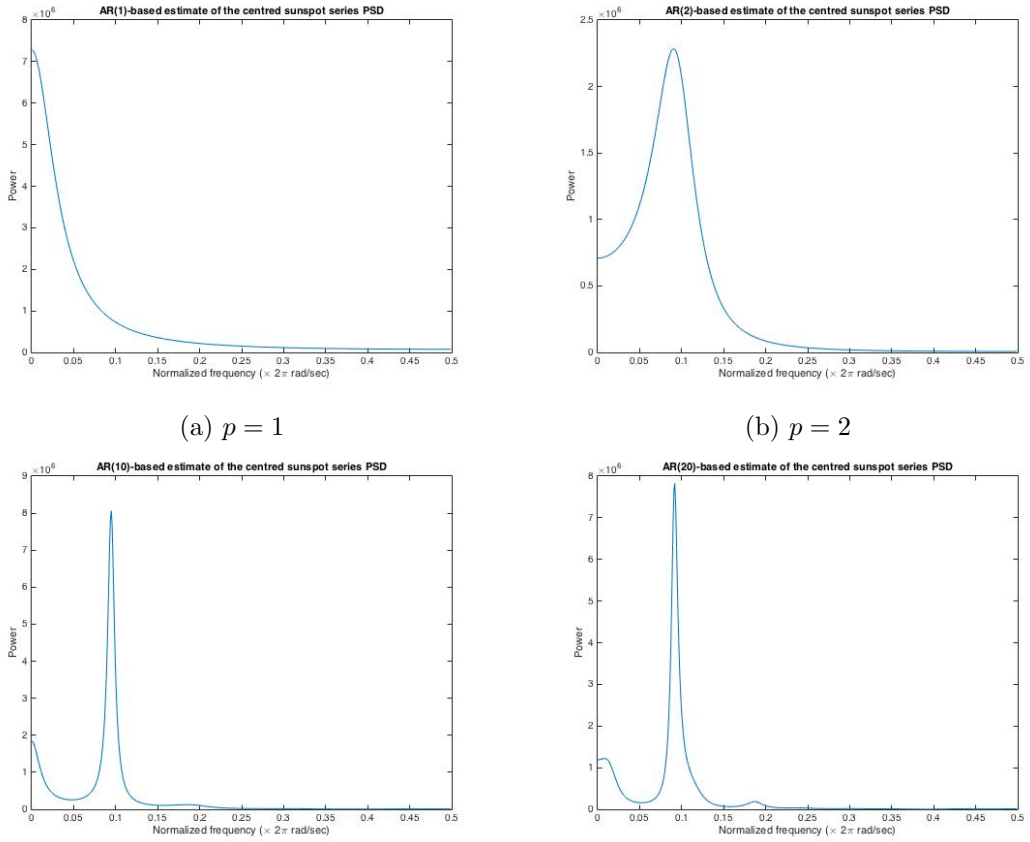
5. We will now repeat this process for the sunspot series. The first thing we notice is that the mean of the data has a very strong effect: it produces a peak in power at 0 Hz that renders the rest of the data hardly readable without zooming (see Figure 3.6a). Therefore, we will focus the mean-centred data.



(a) Original series (b) Mean-centred series

Figure 3.6: Periodogram of the sunspot series

Assuming different filter orders, we will plot an AR(p)-based estimate of the PSD (see Figure 3.7).



(a) $p = 1$

(b) $p = 2$

(c) $p = 10$

(d) $p = 20$

Figure 3.7: AR(p)-based PSD estimates

Figure 3.7 shows a plot of the PSD estimates for different model orders. We can clearly see that undermodelling yields a very bad estimate since the model cannot produce fast changes that are present in the data: effectively, the PSD estimate only contains low-frequency components. Overmodelling, on the other hand, allows high frequency changes but may introduce high frequency "noise" such as the peak at $f_{normalised} = 1.8$ in Figure 3.7d.

3.3 Spectrogram for time-frequency analysis: dial tone pad

1. We will now randomly generate a London landline number and from there compute the discrete-time sequence $\mathbf{y}$ according to the Dual Tone Multi-Frequency (DTMF) system. In this case we will use a sampling rate of 32768 Hz which is much higher than twice the maximum frequency component of the analysed signals. However, we would like to analyse the signal over short periods of time in order to observe a quasi-stationary signal. To obtain an acceptable time-domain resolution, we need a very high sampling rate. The following plot shows the signal $\mathbf{y}$ for the first two keys, as well as the idle time between them.

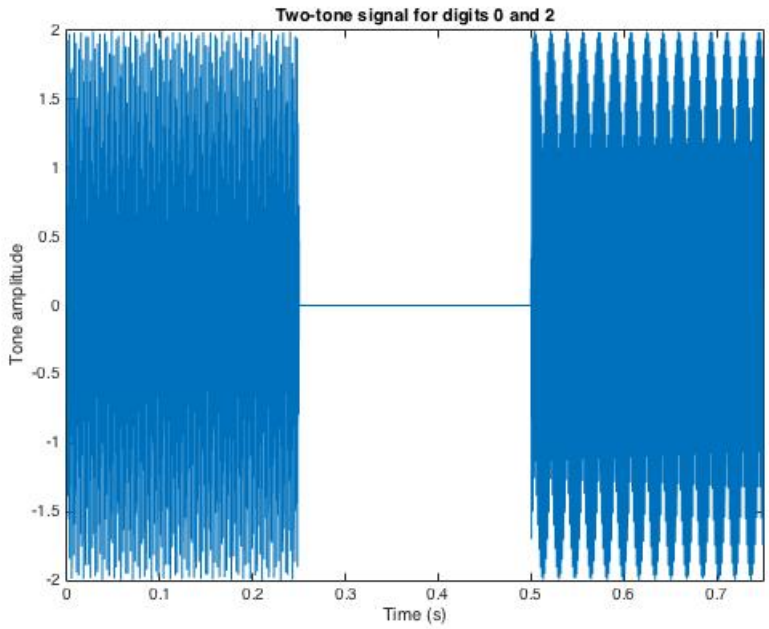


Figure 3.8: Two-tone signal using the DTMF system for the keys 0-idle-2

2. The following figure is the spectrogram obtained when dialling the randomly generated number 020 3229 8345.

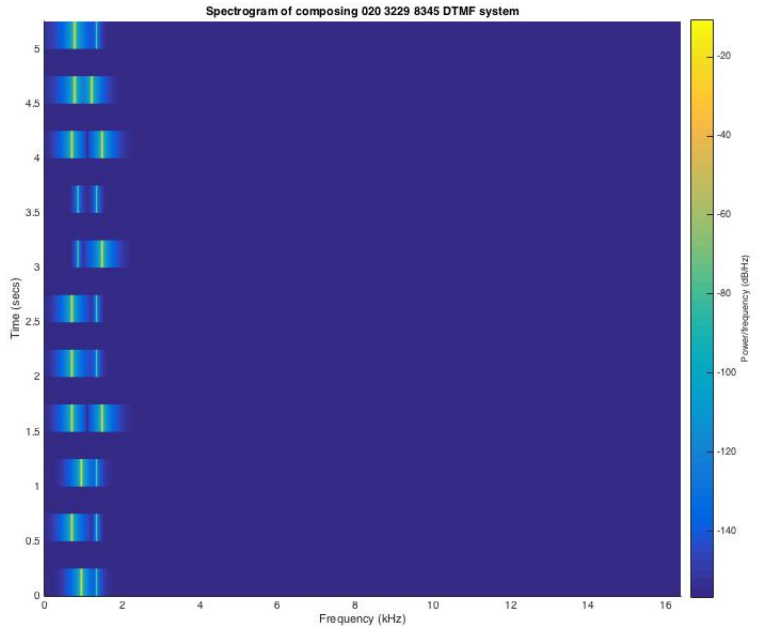
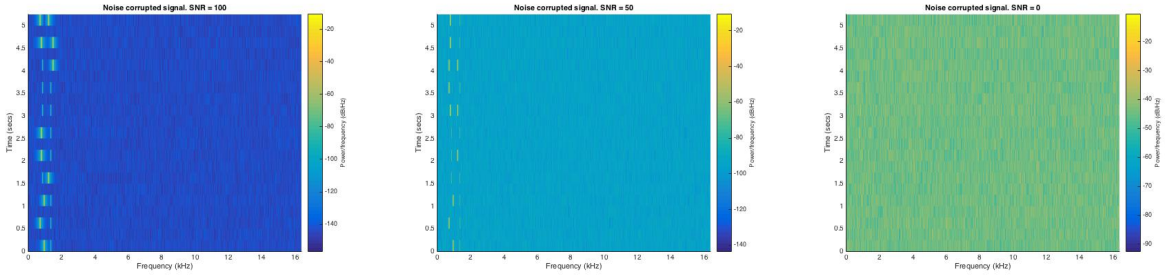


Figure 3.9: Spectrogram obtained by dialling 020 3229 8345

From this spectrogram we can clearly identify the moments at which keys were pressed and the frequencies that correspond to the pressed key. We also notice the windowing effect: rather that having amplitude spikes at the tone's frequencies, we observe gradual changes in amplitude.

3. Using the spectrogram we can identify two specific frequency peaks during every interval over which a key has been pressed. This can be done by sweeping over the DTMF frequency range and checking the amplitude values against a threshold. Given that each key is associated to a unique combination of frequencies, we can read through the DTMF table to recover the digit.

4. In order to simulate real-world conditions, we will corrupt the signal with WGN of varying power. Figure 3.10 shows the spectrogram of the corrupted signal for different signal-to-noise ratios (SNR).



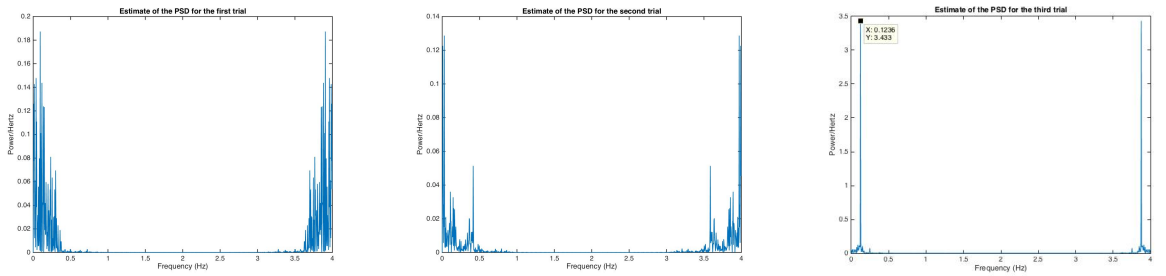
(a) $SNR = 100$ (b) $SNR = 50$ (c) $SNR = 0$

Figure 3.10: Spectrogram of the corrupted signals

As we can see, sweeping through the frequency range with a threshold will longer be an effective method since high variance noise may create artefacts that can be interpreted as one of the DTMF signals. Conversely, the noise could also reduce one of the peaks that corresponds to the pressed key, making it unreadable.

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

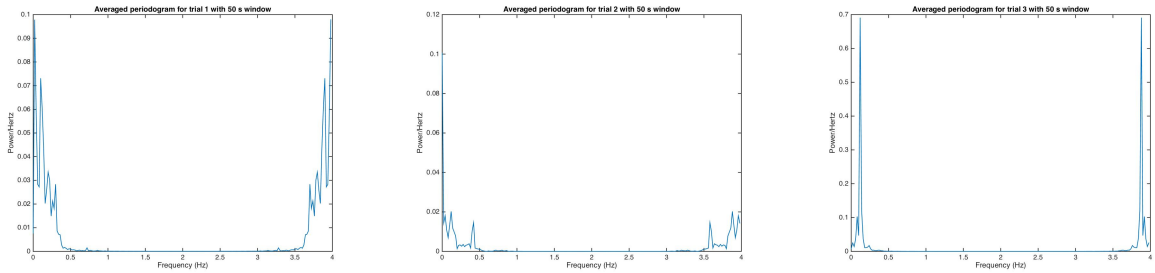
a) We will now plot the periodogram for each of the trials of the iAmp experiment in Figure 3.11. While the first two trials produce frequency components at low amplitudes, the third trial has a sharp peak at 0.1236 Hz.



(a) Trial 1 (b) Trial 2 (c) Trial 3

Figure 3.11: Periodogram of the RRI data

Alternatively, we can also plot the averaged periodogram as shown in Figure 3.12.



(a) Trial 1 (b) Trial 2 (c) Trial 3

Figure 3.12: Averaged periodograms of the RRI data using $50s$ windows

b) Comparing the PSDs of the three trials we can see the effect of the forced breathing on the beating of the heart. In trial 2, a short spike at $f = 0.4Hz$ is starting to appear. This corresponds to 25 beats per minute, which is identical to the forced breathing frequency. However the low amplitude would indicate that the modulation of the heart beat caused by breathing is very low.

In trial 3, the modulation is much stronger. There is a very apparent peak at $f = 0.1236Hz$, which corresponds to 7.41 beats per minute. The heart beat is now governed by the respiratory effort: it is a clear case of respiratory sinus arrhythmia.

Chapter 4

Optimal Filtering - Fixed and Adaptive

4.1 Wiener filter

1. In this section we are interested in estimating the parameters of an unknown system using an optimum filter. Here we will compute the coefficient of a Wiener filter for an SNR of $20dB$. The calculated coefficients are shown in table 4.1 below as well as the values of the so-called *unknown system*.

	$\mathbf{w}_{opt}(1)$	$\mathbf{w}_{opt}(2)$	$\mathbf{w}_{opt}(3)$	$\mathbf{w}_{opt}(4)$	$\mathbf{w}_{opt}(5)$
Unknown system coefficients	1	2	3	2	1
Wiener coefficients	0.2295	0.4553	0.6804	0.4512	0.2194
"De-normalised" coefficients	1.0031	1.9901	2.9736	1.9719	0.9589

Table 4.1: Table of the optimal coefficients

Notice that the coefficients follow the right trend but seem to be scaled by a constant. This is due to the fact that we normalised the filter output $\mathbf{y}$ to have unity variance. We can multiply the obtained coefficients by the standard deviation of $\mathbf{y}$ to recover a correct estimate of the filter coefficients.

2. Repeating the experiment for different noise powers we obtain the following table. Notice that we are now plotting the de-normalised coefficients only.

SNR	$\mathbf{w}_{opt}(1)$	$\mathbf{w}_{opt}(2)$	$\mathbf{w}_{opt}(3)$	$\mathbf{w}_{opt}(4)$	$\mathbf{w}_{opt}(5)$
20	0.9891	1.9929	2.9897	2.0200	1.0011
6.0206	0.9648	2.0494	2.9422	2.1317	1.0840
0	1.0100	1.9704	2.8114	1.9769	0.9632
-13.9794	1.7885	2.118	3.1759	1.5958	2.5931
-16.9020	2.4010	2.2939	2.5086	1.2797	1.3618
-20	1.3879	1.1993	3.0772	2.9060	-0.2909

Table 4.2: Table of the optimal coefficients

As long as the SNR is greater than $0dB$, the calculated coefficient yield a plausible approximation of the unknown system. However, when the noise power becomes greater than the signal power, coefficients are no longer similar to the original parameters. Even the relative magnitudes of the coefficients are lost with multiple cases of $\mathbf{w}_{opt}(1) > \mathbf{w}_{opt}(2)$ (for example).

Overestimating N_w will return additional coefficients that will all be very close to 0.

3. The Wiener method can be broken down as follows:

   - Autocorrelation of N-sample signal for N_w+1 lags - requires $\mathcal{O}(N(N_w+1))$ operations
   - Cross-correlation of N-sample signals for N_w+1 lags - requires $\mathcal{O}(N(N_w+1))$ operations
   - Inversion of $(N_w+1)\times(N_w+1)$ matrix - requires $\mathcal{O}((N_w+1)^3)$ operations
   - Product of $(N_w+1)\times(N_w+1)$ matrix with $(N_w+1)\times 1$ matrix - requires $\mathcal{O}((N_w+1)^2)$ operations

   Adding all four, we obtain the following two-input complexity $T(N, N_w)$:

$$T(N, N_w) = 2 \times N(N_w+1) + (N_w+1)^3 + (N_w+1)^2 = \mathcal{O}(N_w N + N_w^3) \tag{4.1}$$

4.2 The least mean square (LMS) algorithm

1. The MATLAB script `lms` implements an adaptive filter in which the coefficients $\mathbf{w}$ are recursively computed. The output is an error vector as well as a matrix that traces the evolution of the $\mathbf{w}$ coefficients. We can clearly see that as we go through multiple iterations, the coefficients seem to converge to the coefficients of the unknown system. Equally, the error seems to converge towards 0 as shown in the following plot. However, because the step size does not reduce to zero as the recursion goes on, the coefficients are constantly adjusted and are never exactly equal to the unknown parameters.

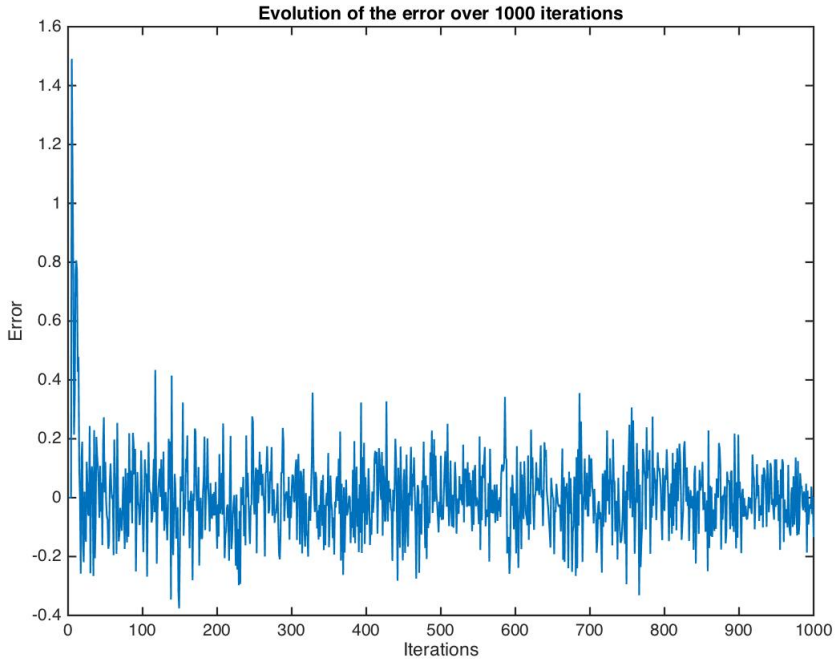
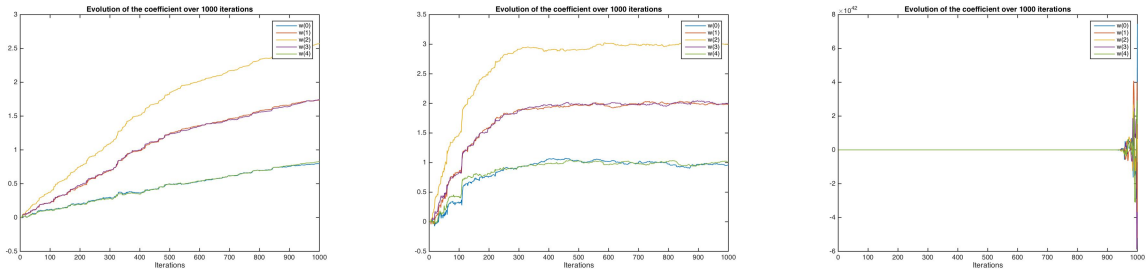


Figure 4.1: Evolution of the error over 1000 iterations

2. Let us now examine the evolution of the coefficients more closely for step sizes $\mu \in \{0.002, 0.01, 0.5\}$ (see Figure 4.2).



(a) $\mu = 0.002$ (b) $\mu = 0.01$ (c) $\mu = 0.5$

Figure 4.2: Evolution of the coefficients over 1000 computations

As we can see, if the gain is too low, the coefficients converge very slowly. On the other hand, a too big gain will yield diverging values. In the case $\mu = 0.01$, it takes approximately 300 iterations for the system to settle.

3. Each iteration of this adaptive method can be broken down as follows:

   - Dot product of two $N_w + 1 \times 1$ vectors - requires $\mathcal{O}(N_w)$ operations
   - One scalar subtraction - requires 1 addition and 1 multiplication
   - Sum of a $N_w+1\times1$ vector with a scaled $N_w+1\times1$ vector - requires $\mathcal{O}(N_w)$ operations

   Overall, the complexity of this algorithm is $T(N_w) = \mathcal{O}(N_w)$. This is much smaller than for the Wiener method investigated earlier.

4.3 Gear shifting

We would like to change the gain as the coefficients are converging towards the real values using a function f such that $\mu = f(e)$. The two border cases are:

- if the error is 0, the coefficients must not change - therefore $f(0) = 0$
- if the error is infinite, the gain must be as large as possible without falling into instability. From experiments we noticed that 0.15 is a suitable value - therefore $f(\infty) = 0.15$

Linking the two with a negative exponential we obtain an equation of the following form

$$\mu = 0.15(1 - e^{\frac{-|error|}{\tau}}) \tag{4.2}$$

The time constant τ can be chose to obtain the desired transient response. Tuning it empirically has allowed to find a value that settles quickly while limiting the overshoot to 20% for each coefficient: $\tau = 2$. The figure below shows the improved transient response. Notice that we have focused on shortening the settling time rather than improving the steady-state value. It now only takes approximately 100 iterations to reach steady-state.

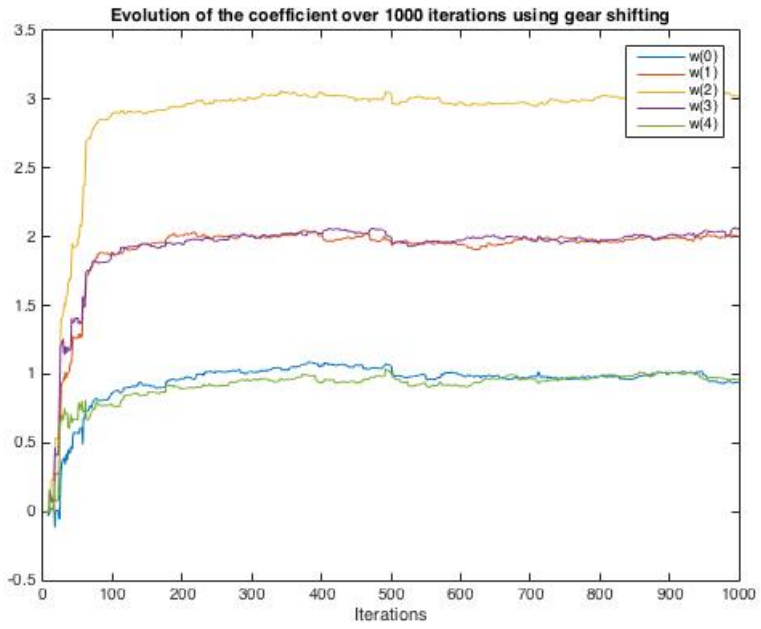


Figure 4.3: Improved transient response using gear shifting

4.4 Identification of AR processes

1. Giving the MATLAB specification $\mathbf{a} = [1\ 0.9\ 0.2]$ for the AR model will generate the following transfer function

$$\frac{X(z)}{\eta(z)} = \frac{1}{1 + 0.9z^{-1} + 0.2z^{-2}} \tag{4.3}$$

   Converting into the time domain yields

$$x(n) = \eta(n) - 0.9x(n-1) - 0.2x(n-2) \tag{4.4}$$

   The parameters we are estimating are therefore -0.9 and -0.2, to which a_1 and a_2 should respectively converge.

2. As shown above we are trying to estimate two parameters but have no information on the system's input ($\eta(n)$). Estimation in this case will be more complicated than it was in the case of an MA model. Figure 4.4 shows the evolutions of the coefficients for a gain of $\mu = 0.0001$.

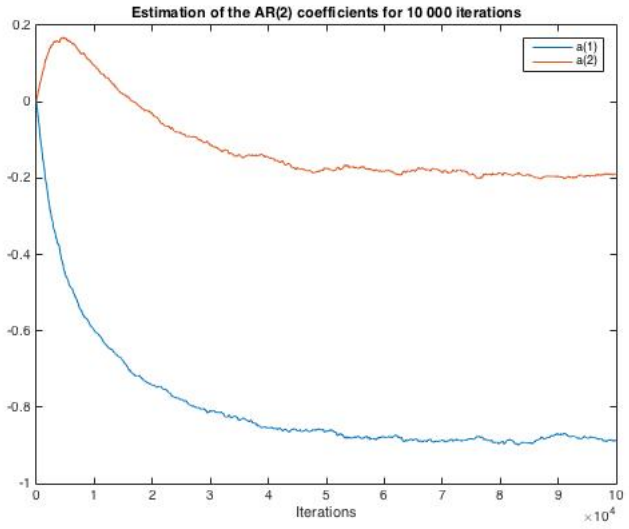


Figure 4.4: Evolution of the estimated coefficients for $\mu = 0.0001$

In this case, it takes a very long time for the system to settle. Changing the gain to greater values makes the system faster but the steady-state deteriorates very rapidly. Indeed for $\mu = 0.01$ or $\mu = 0.001$ there are very strong oscillations contained within ± 0.07 and ± 0.26 around the nominal values respectively. Yet a smaller gain value than the one plotted yields an even slower response.

4.5 Speech recognition

1. The system we have just studied is a predictor. We will now use it on a recorded voice signal and plot its output on the same graph as the real signal (see Figure 4.5)

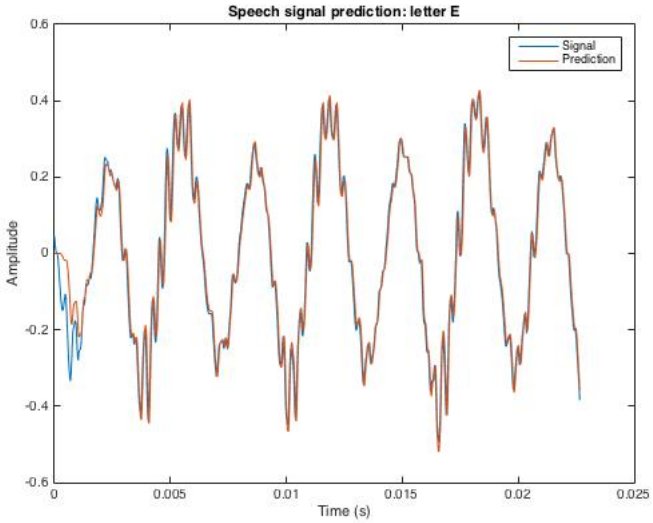


Figure 4.5: Prediction of the signal for the letter "e" using AR(2) predictor and $\mu = 0.1$

Playing with the two parameters at hand we notice that:

- using higher gains will make the response faster

- using higher order models yield similar estimates in shorter times until optimality is reached

2. To determine the ideal order we could use the same techniques employed for the sunspot series in section 2.4, namely using the partial correlation functions, the MDL and/or the AIC.

3. Keeping the same number of samples for a lower sampling frequency means that we are investigating a greater portion of time of the original signal. This in turns means that we may no longer be predicting a stationary process. The error will rise and therefore the prediction gain will diminish. Alternatively, we can use a higher order model to yield a closer approximation.

4.6 Dealing with computational complexity: sign algorithms

Let us now try three different algorithm based on the LMS, namely the sign algorithm, the signed-regressor and the sign-sign. The following plot shows the estimated coefficients using these three algorithms as well as the regular LMS for the signal studied in section 4.2.

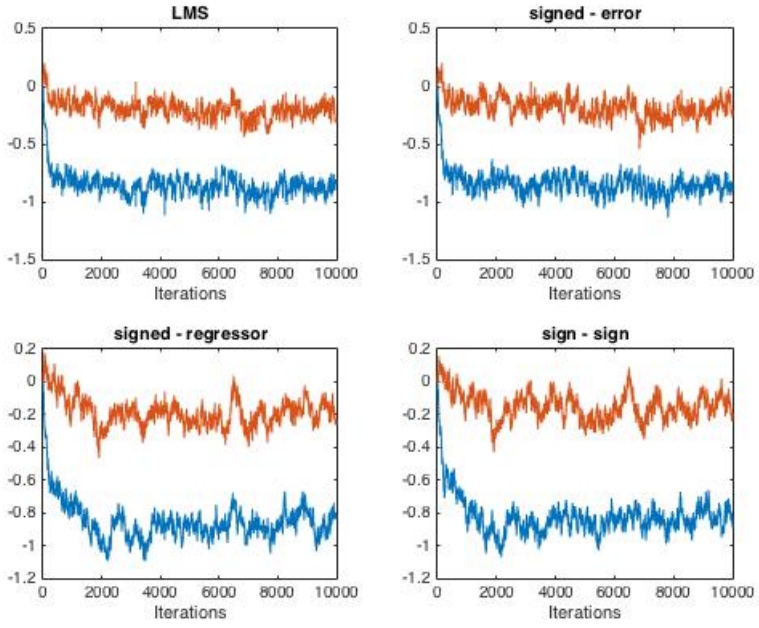


Figure 4.6: Comparison of the LMS-derived algorithms for an AR(2) process

The idea behind these algorithms is reduce computational complexity by using the sign of certain variables rather than the values themselves in order to compute the next coefficients. We notice that when the operation is simplified to the maximum (sign-sign algorithm), the coefficients do not converge exactly to their nominal values. The algorithms that use only one sign function yield a better estimate, however their steady-state error is greater than the regular LMS algorithm.

Depending on the application, this trade-off between error and complexity may be desirable. Indeed, if we were to use the algorithms as real-time predictors, we can see that although it yields different coefficients, the sign-sign algorithm produces an acceptable estimate in far less

time. Figure 4.7 shows predictions obtained using the four algorithms for the speech signal that corresponds to the letter “s” (sampled at 44100 Hz).

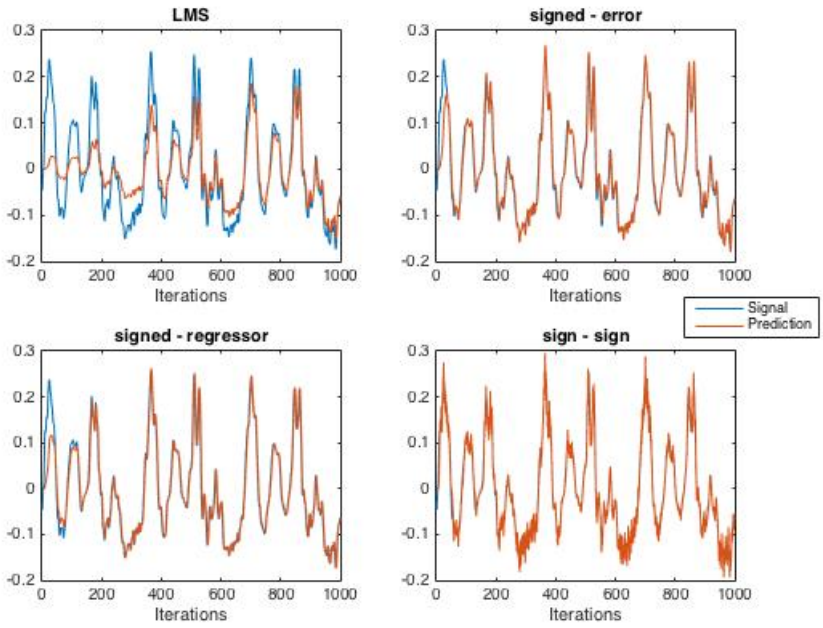


Figure 4.7: Comparison of the prediction obtained with the LMS-derived algorithms for letter “s”

Chapter 5

A Real World Case Study: Vinyl Denoising

1. In this section we will try to denoise part of the song Stairway to Heaven. First, we will plot the periodogram of the corrupted signal (see Figure 5.1). Although we cannot identify the spectral components of the ticks yet, there seems to be an anomaly in the power spectrum at $1500Hz$.

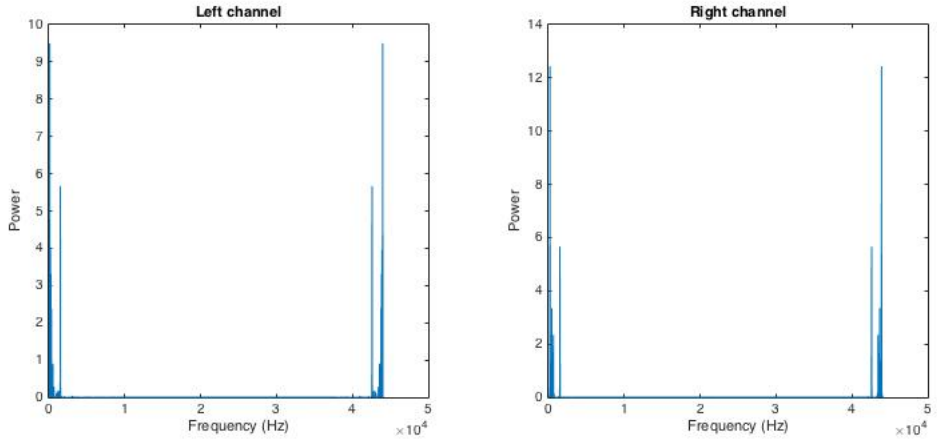


Figure 5.1: Periodogram of the corrupted signal

2. We now have access to a clean copy of the song of which we will plot the periodogram in Figure 5.2. We can now clearly see that the spikes mentioned above are indeed linked to the clicks.

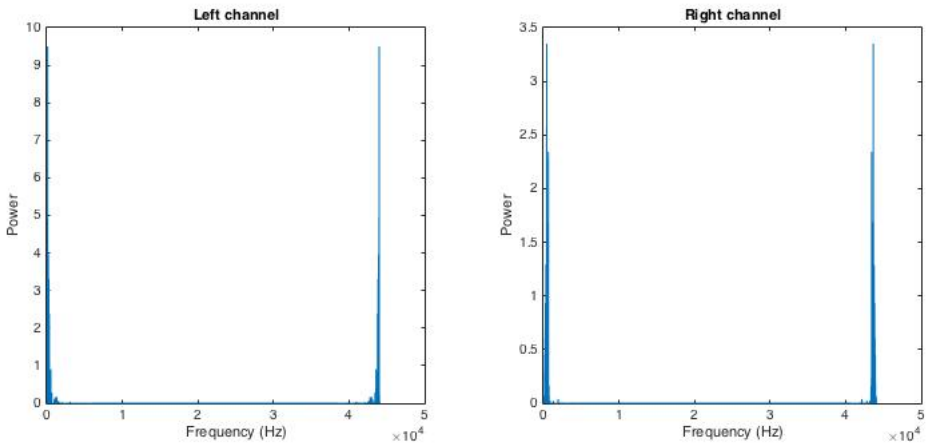


Figure 5.2: Periodogram of the clean signal

In order to plot an estimate of the PSD of the ticks, we can subtract the signals from each other to be left only with the ticks. The following periodogram is the one we obtain after performing the subtraction. We now know the frequency distribution of the noise corruption: spikes at $1.5kHz$ in the left channel, and spikes at $1.5kHz$ and $200Hz$ for the right channel.

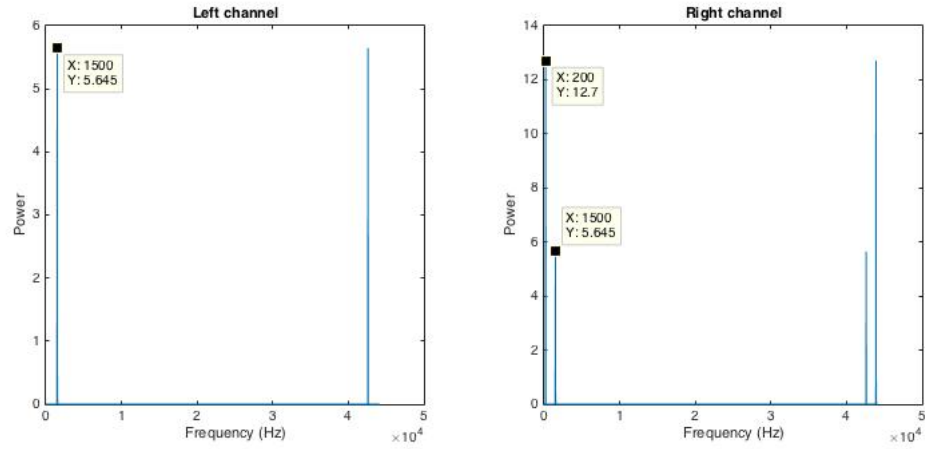


Figure 5.3: Periodogram of the ticks

3. Given the distribution of the noise spectrum we are interested in performing notch filtering. We will do so by using a Butterworth filter since it is the form of filter that has the flattest passband. A 4th-order filter should yield a sharp enough roll-off. The left channel will go through two notch filters at $1500Hz$ while the right channel will go through the same two filters in addition to having two notch filters at $200Hz$.

   As expected, the filtering has greatly reduced the noise, however, during the first few seconds no music is played in the right channel and some of the ticks can still be heard.

4. We will now plot the periodogram of the resulting signal along with those of the original and corrupted recordings.

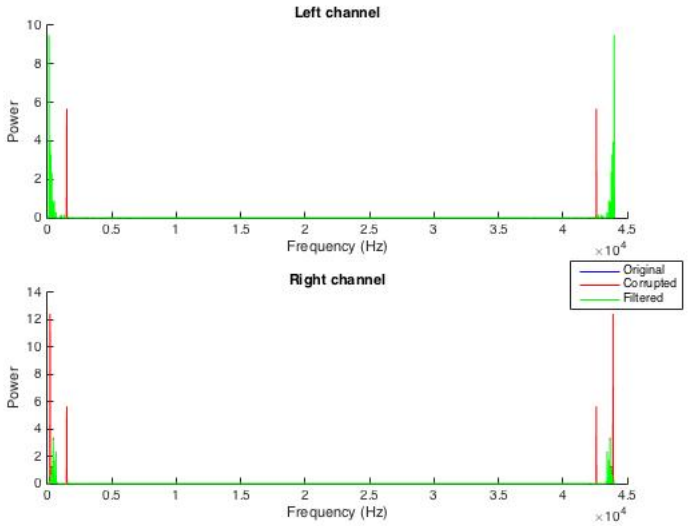


Figure 5.4: Comparison of the filtered, original and corrupted signal periodograms

The relative errors can be computed for both the left and right channels:

$$e_{left} = 0.0068 \quad \text{and} \quad e_{right} = 0.1733 \tag{5.1}$$

It is greater for the right channel since we had to filter out the $200Hz$ frequency band which also contains some of the music's spectral content.

5. In order to avoid the over simplistic filtering we were performing, we will implement an adaptive filter. Indeed we would like to suppress the frequencies of interest only when the ticks kick-in rather than over the whole time extract. Applying a normalised LMS (NLMS) algorithm allows us to do so. Varying the different parameters (gain μ and order p) produces the relative errors shown in Figure 5.5 for the left and right channels

Gain\Order	5		10		20	
0,01	0,0651	0,1283	0,0291	0,1296	0,0168	0,1216
0,1	0,038	0,0835	0,0141	0,0859	0,0115	0,0768
1	0,0214	0,0487	0,0169	0,0437	0,0059	0,0286

Figure 5.5: Table of the left and right relative errors while varying μ and p

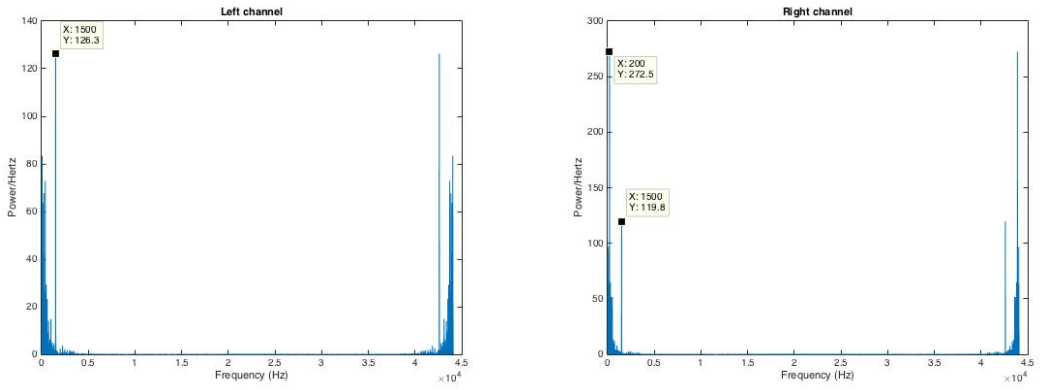
As we can see, a gain of 1 and increasing filter orders will reduce the error. However, although the error is much smaller than for the Butterworth filter, the quality of the outputted sound is not nearly as good. Most of the error in the Butterworth filter's right channel comes from over-suppression of the extract's spectrum. This was necessary because some of the ticks were located within the song's bandwidth. The adaptive filter does not perform this over-suppression and the human ear will very quickly pick up on it.

The normalised LMS performs much better than the regular LMS. Indeed for $\mu = 1$ and $p = 10$, the relative errors obtained with the latter were

$$e_{left,LMS} = 0.1388 \quad \text{and} \quad e_{right,LMS} = 0.6141 \tag{5.2}$$

6. Applying the same adaptive algorithm as we did for Stairway to Heaven, we now wish to denoise Liquid Tension Experiment's song Universal Mind. As we can see from its periodogram, this song has a much more complex frequency content due to distorted guitars

and analogue synths. We also notice that the ticks are present at the same frequencies as before.



(a) Left channel (b) Right channel

Figure 5.6: Periodogram of the corrupted copy of Universal Mind

Applying the `nlms` with parameters $\mu = 1$ and $p = 20$ seems to remove most of the noise. Just as for Stairway to Heaven, low frequency ticks are still audible in the right channel, however in this case they get covered by the music after a few seconds. Let us evaluate the algorithm by plotting the periodograms of the original, corrupted and filtered signals as in part 5.4 (see Figure 5.7). We will also compute the relative error for both channels.

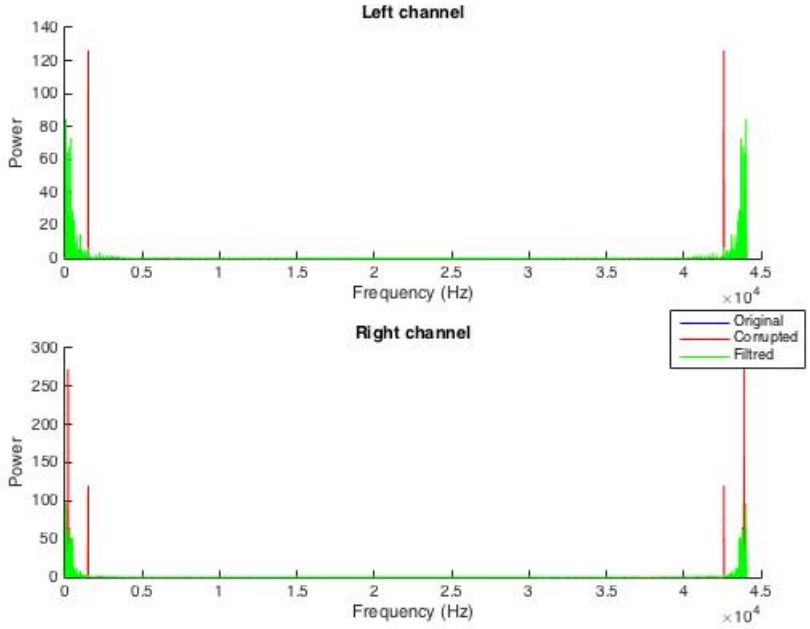


Figure 5.7: Comparison of the filtered, original and corrupted signal periodograms

The relative errors are

$$e_{left} = 0.0197 \quad \text{and} \quad e_{right} = 0.0478 \tag{5.3}$$

These are higher than for Stairway to Heaven due to the fact that the filter is removing part of the signal at the same time as it removes the noise. Although it can be observed

for both signals, this effect is amplified here because Universal Mind has a richer spectral content at the same frequencies as the ticks. However, this spectral loss is not as audible due to the density of the signal's spectral content.