Imperial College London

Imperial College London

Department of Electrical and Electronic Engineering

ELEC96002 Advanced Signal Processing

Prof. Danilo P. Mandic

Coursework Report

submitted by

Costanza Gulli

CID 01352733

Contents

1 Random signals and stochastic processes

1.1 Statistical estimation

1.1.1 Uniform distribution

Figure 1 is a plot of 1000 random realisations of the uniform random variable $X \sim \mathcal{U}(0,1)$. The pdf of the random variable is described in Equation 1. From the samples, the sample mean $\hat{\mu}_X$ and sample standard deviation $\hat{\sigma}_X$ are obtained. These are calculated with the matlab functions `mean` and `std`, according to the formulas in Equations 2 and 3.

$$f_X(x) \sim \mathcal{U}(0,1) = \begin{cases} 1 & 0 \leq x \leq 1 \\ 0 & otherwise \end{cases} \tag{1}$$

$$\hat{\mu}_X = \frac{1}{N}\sum_{n=1}^{N} x[n] \tag{2}$$

$$\hat{\sigma}_X = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(x[n]-\mu_x)^2} \tag{3}$$

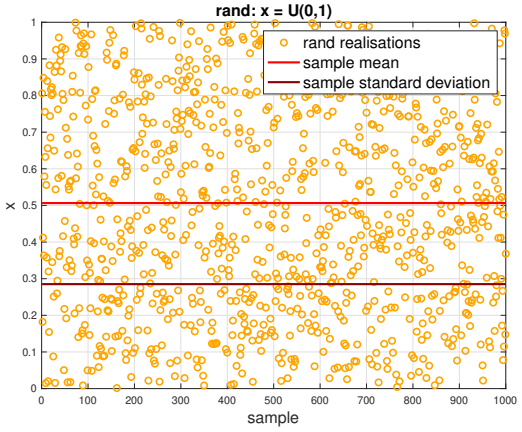

Figure 1: 1000 realisations of the random variable X;
their sample mean and standard deviation

After this, a set of 10 realisations, each of 1000 samples is obtained. For each realisation, the sample mean and sample standard deviation is calculated, and compared with the theoretical value. As it can be noticed from Figure 2, when considering the mean and standard deviation of different realisation, each comprising of a large enough number of samples (in this case N = 1000), the sample values are very close to the theoretical ones. In fact, it can be noticed from Figure 3 that the estimated values asymptotically converge to the theoretical ones as the sample size increases. In the plots, the error in sample mean estimation is calculated as $e = \mu_X - \hat{\mu}_X$, and the error in sample standard deviation estimation is calculated as $e = \sigma_X - \hat{\sigma}_X$.

The theoretical mean and standard deviation are calculated from the pdf, as follows in Equations 4 and 5.

$$\mu_X = \int_{-\infty}^{\infty} x\, f_X(x)\, \mathrm{d}x = \int_0^1 x\, \mathrm{d}x = \left[\frac{x^2}{2}\right]_0^1 = \frac{1}{2} \tag{4}$$

$$\sigma_X = \sqrt{\mathbb{E}\{X^2\} - \mathbb{E}\{X\}^2} \tag{5}$$

$$\mathbb{E}\{X^2\} = \int_{-\infty}^{\infty} x^2\, f_X(x)\, \mathrm{d}x = \int_0^1 x^2\, \mathrm{d}x = \left[\frac{x^3}{3}\right]_0^1 = \frac{1}{3}$$

$$\mathbb{E}\{X\}^2 = \mu_X^2$$

$$\sigma_X = \sqrt{\mathbb{E}\{X^2\} - \mathbb{E}\{X\}^2} = \sqrt{\frac{1}{3} - \frac{1}{2^2}} = \sqrt{\frac{1}{12}}$$

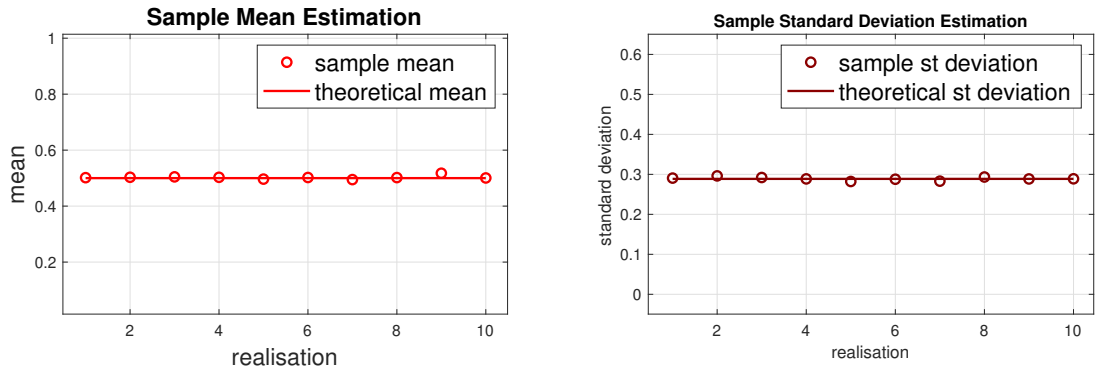

Figure 2: Sample mean and standard deviation estimation for 10 realisations of 1000 samples, compared with the theoretical value

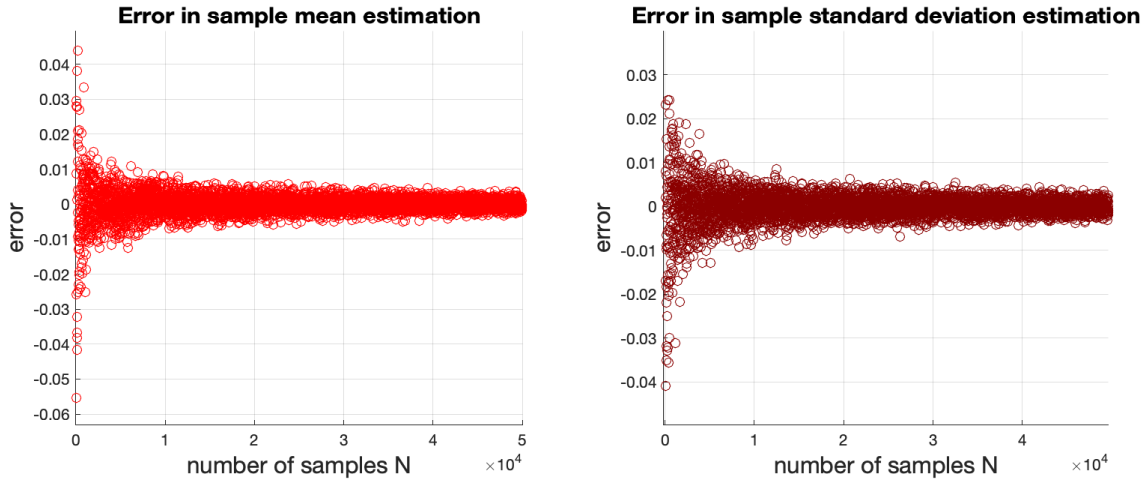

Figure 3: Error in mean and standard deviation estimation as a function of number of samples N

Afterwards, the samples are used to approximate the pdf from which the samples are drawn (Figure 4). This is done by using the `hist` matlab function. The result is plotted using `bins`. The pdf estimation is obtained varying both the number of bins and the number of samples. Two main phenomena can be noticed.

First of all, having more bins results in a better resolution in the pdf approximation. Secondly, the number of samples per bin has to be sufficient in order to obtain meaningful results. In subplot 2, there are 10 samples per bin. This leads to unexpected behaviour due to the random nature of the distribution from which the samples are drawn.

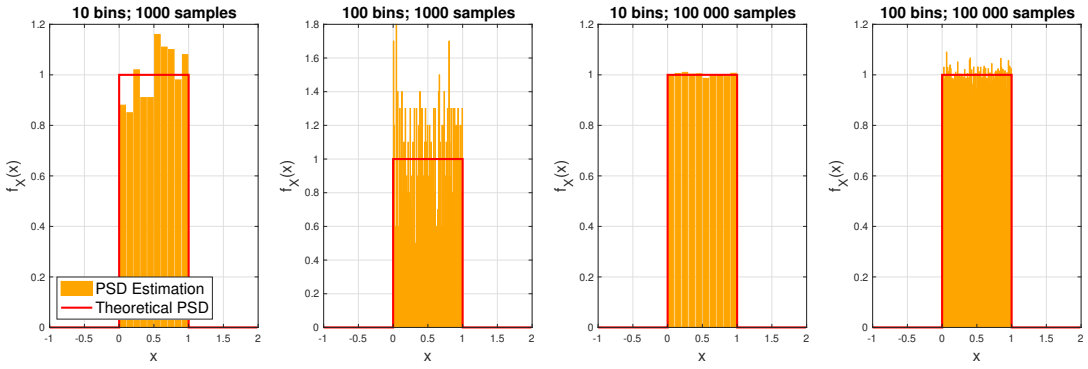

Figure 4: PDF estimation, varying the sample size and number of bins

1.1.2 Normal distribution

The same analysis is conducted with samples drawn from a standard normal distribution. The pdf is given in Equation 6. 1000 samples are drawn from this distribution. The plot is shown in Figure 5, together with the sample mean and

standard deviation. As above, these are obtained according to equations 2 and 3.

$$f_X(x) \sim \mathcal{N}(0,1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \tag{6}$$

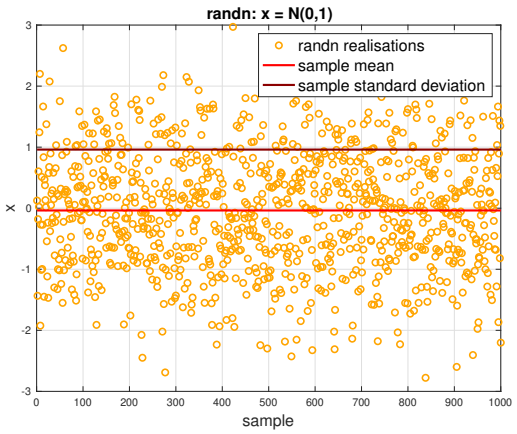

Figure 5: 1000 realisations of the random variable X;
their sample mean and standard deviation

Again, 10 realisations are obtained and their sample mean and standard deviations are compared with the theoretical values (Figure 6). The mean of a standard normal distribution is $\mu_X = 0$ and the standard deviation is $\sigma_X = 1$.

It can be noticed that the values obtained from the samples are less precise than those of the uniform distribution in section 1.1.1. This is because for the normal distribution the distribution is more spread around the mean, leading to higher uncertainty on the output. To obtain more accurate results, we would need to calculate the values from a higher number of samples N.

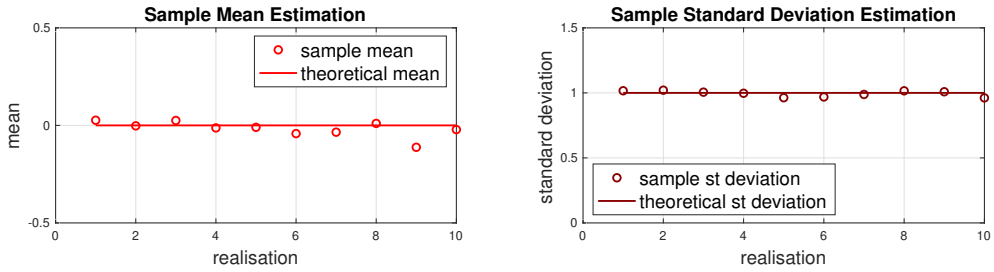

Figure 6: Sample mean and standard deviation estimation for 10 realisations of 1000 samples, compared with the theoretical value

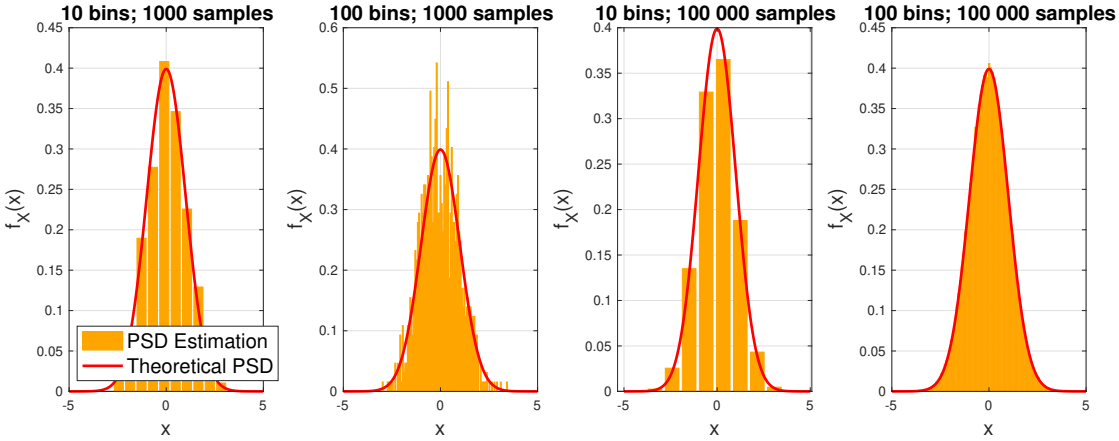

Figure 7: PDF estimation, varying the sample size and number of bins

When approximating the pdf of the distribution from which the samples are drawn, similar results to those obtained in section 1.1.1 are noticed (Figure 7). Subplot 4 shows that, more than in the uniform distribution, having more bins increases the resolution in approximating the pdf. However, subplot 2 shows that the number of samples per bin has to be higher than ~ 100 to avoid unpredictable behaviour. In fact, subplots 1 and 3 are really precise despite having very low resolution.

1.2 Stochastic processes

This section analyses the difference between ensemble and time means and standard deviations. These characteristics are than used o discuss the stationarity and ergodicity of random processes. In the following sections, signals drawn from 3 probability distributions are analysed.

1.2.1 Random process 1

The first signal is drawn from *random process 1*, defined by the following matlab function.

```
function v=rp1(M,N)
a=0.02;
b=5;
Mc=ones(M,1)*b*sin((1:N)*pi/N);
Ac=a*ones(M,1)*[1:N];
v=(rand(M,N)-0.5).*Mc+Ac;
```

The output is a MxN matrix, where M is the number of realisations and N is the number of samples per realisation.

Ensemble mean and standard deviation M = 100 realisations of N = 100 samples of the random process are are obtained. The ensemble mean and standard deviation are obtained by averaging the values for all realisation at a given time. The results are plotted as a function on time in Figure 8. More accurate values are obtained by averaging M = 100,000 realisations (dotted lines).

Both mean and standard deviation vary as a function of time. Therefore, the signal is **not stationary**.

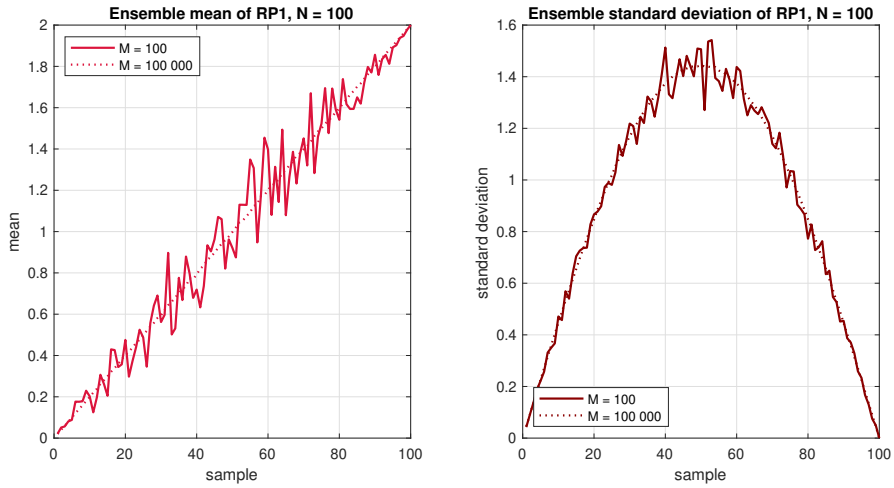

Figure 8: Ensemble mean and standard deviation of RP1

Mean and standard deviation for each realisation The mean and standard deviations are also obtained by averaging the values over time for each realisation. The time mean and standard deviation for each realisation are plotted in Figure 9. The values are constant over time, but the signal is non-stationary, therefore is also **non-ergodic**.

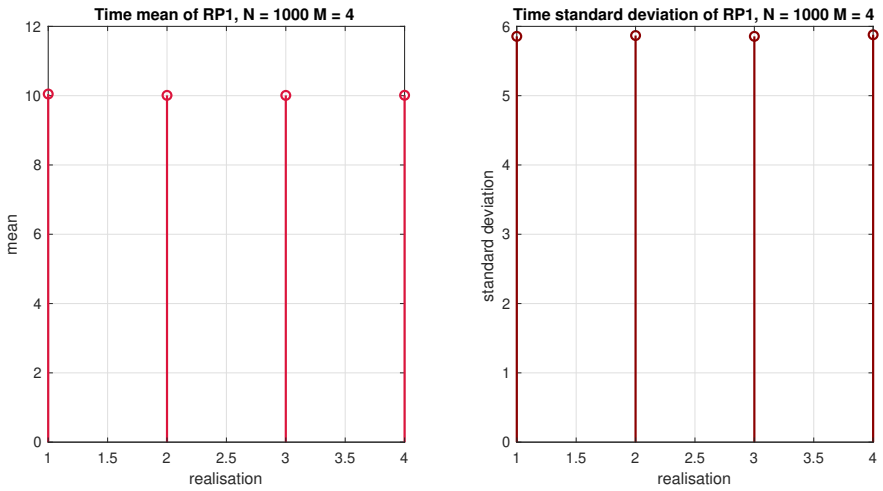

Figure 9: Time mean and standard deviation of RP1

Theoretical mean and variance The mathematical description for signal A, drawn from RP1, is given below.

$$f_A(n) = (u(n) - 0.5) * b * \sin(\frac{n\pi}{N}) + a * n = u_1(n) * \sin(\frac{n\pi}{N}) + 0.02n$$

$$u_1(n) \sim \mathcal{U}(-2.5, 2.5)$$

The expectation is calculated as follows:

$$\mu_A = \mathbb{E}\{u_1(n) * \sin(\frac{n\pi}{N})\} + \mathbb{E}\{0.02n\} = 0.02n$$

It can be noticed that the result matches the data in Figure 8. In fact, the mean is a straight line through the origin, with value of 2 at n = 100.

The variance is calculated from the signal's equation:

$$\sigma_A^2 = \mathbb{V}ar\{f_A(x)\} = \mathbb{E}\{(f_A(x) - \mu_A)^2\} = \mathbb{E}\{((u(n) - 0.5)5\sin(\frac{n\pi}{N}) + 0.02n - 0.02n)^2\} =$$

$$= \mathbb{E}\{(u(n) - 0.5)^2 25\sin^2(\frac{n\pi}{N})\} = 25\sin^2(\frac{n\pi}{N})\mathbb{E}\{(u(n) - 0.5)^2\} = 25\sin^2(\frac{n\pi}{N})\mathbb{V}ar\{u(n)\} = \frac{25}{12}\sin^2(\frac{n\pi}{N})$$

The standard deviation is therefore $\sigma_A = \frac{5}{\sqrt{12}}\sin(\frac{n\pi}{N})$. This matches the estimated standard deviation in Figure 8. In fact, this is a sinusoidal signal with period $\approx 2N$ and peak at ≈ 1.44, which is the approximate value of $\frac{5}{\sqrt{12}}$.

1.2.2 Random Process 2

The signal B is drawn from *random process 2*, defined by the following matlab function.

```
function v=rp2(M,N)
Ar=rand(M,1)*ones(1,N);
Mr=rand(M,1)*ones(1,N);
v=(rand(M,N)-0.5).*Mr+Ar;
```

Ensemble mean and standard deviation Figure 10 shows that the mean and standard deviation are constants over time. The signal is **stationary**.

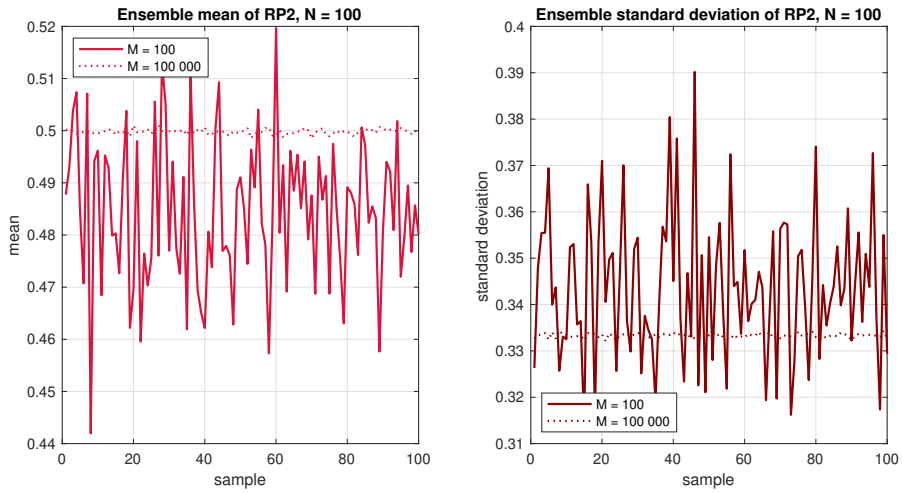

Figure 10: Ensemble mean and standard deviation of RP2

Mean and standard deviation for each realisation It can be noticed from Figure 11 that the time mean and standard deviation varies across realisations. Therefore the signal is **non-ergodic**.

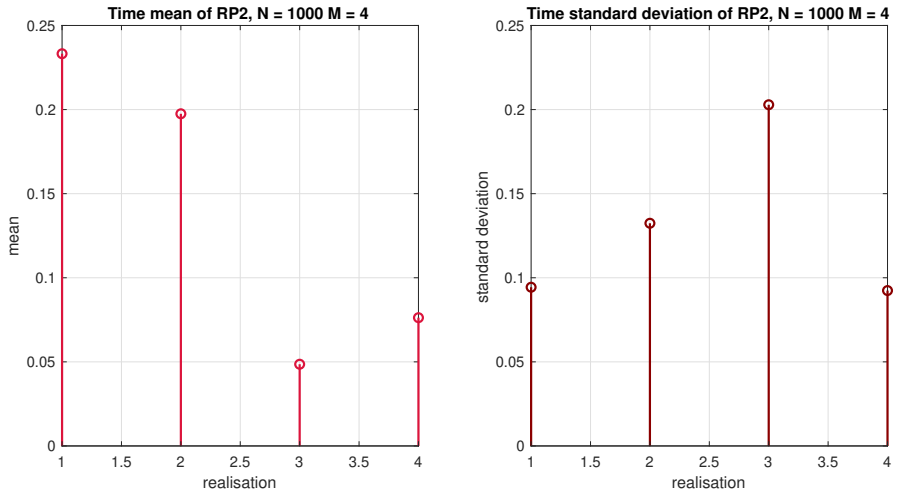

Figure 11: Time mean and standard deviation of RP2

Theoretical mean and variance The signal's equation is as follows:

$$f_B(n) = u(n) * (u(n) - 0.5) + u(n) = u(n) * u_2(n) + u(n)$$
$$u_2(n) \sim \mathcal{U}(-0.5, 0.5)$$

The theoretical mean and variance are obtained from the above equation. Note that the calculations exploit the fact that $u(n), u_2(n)$ and $u(n)$ are independent random variables.

$$\mu_B = \mathbb{E}\{u(n) * u_2(n) + u(n)\} = \mathbb{E}\{u(n)\} * \mathbb{E}\{u_2(n)\} + \mathbb{E}\{u(n)\} = 0.5 * 0 + 0.5 = 0.5$$
$$\sigma_B^2 = \mathbb{V}ar\{u(n) * u_2(n) + u(n)\} = \mathbb{V}ar\{u(n) * u_2(n)\} + \mathbb{V}ar\{u(n)\} =$$
$$= \mathbb{E}\{u^2(n)\}\mathbb{E}\{u_2^2(n)\} - \mathbb{E}\{u(n)\}^2\mathbb{E}\{u_2(n)\}^2 + \mathbb{E}\{u^2(n)\} - \mathbb{E}\{u(n)\}^2 =$$
$$= \int_0^1 x^2 \, \mathrm{d}x \; * \int_{-0.5}^{0.5} x^2 \, \mathrm{d}x - 0.5^2 * 0^2 + \int_0^1 x^2 \, \mathrm{d}x - 0.5^2 =$$
$$\left[\frac{x^3}{3}\right]_0^1 * \left[\frac{x^3}{3}\right]_{-0.5}^{0.5} + \left[\frac{x^3}{3}\right]_0^1 - 0.5^2 = \frac{1}{9}$$

The value of the mean is confirmed by the data in Figure 10. Note that the value asymptotically tends towards the theoretical as the number of realisations increases. The standard deviation is obtained as the square root of the variance, $\sigma_B = \sqrt{\sigma_b^2} = \frac{1}{3} \approx 0.33$. This is confirmed by the value obtained empirically in Figure 10.

1.2.3 Random Process 3

The last signal is drawn from *random process 3*, defined by the following matlab function.

```
function v=rp3(M,N)
a=0.5;
m=3;
v=(rand(M,N)-0.5)*m + a;
```

Ensemble mean and standard deviation The ensemble mean and standard deviation are both constant over time (Figure 12). The signal is therefore **stationary**.

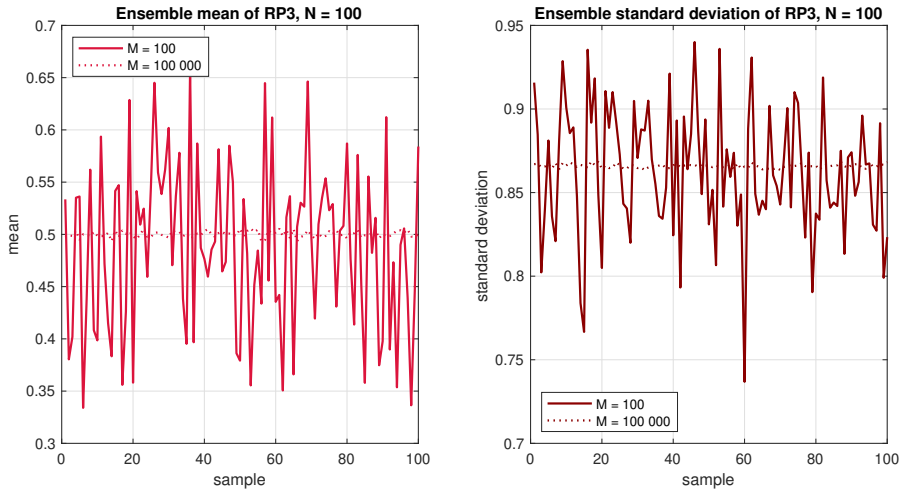

Figure 12: Ensemble mean and standard deviation of RP3

Mean and standard deviation for each realisation Figure 13 shows that the time mean and standard deviation do not vary across realisations. Therefore, the signal is **ergodic**.

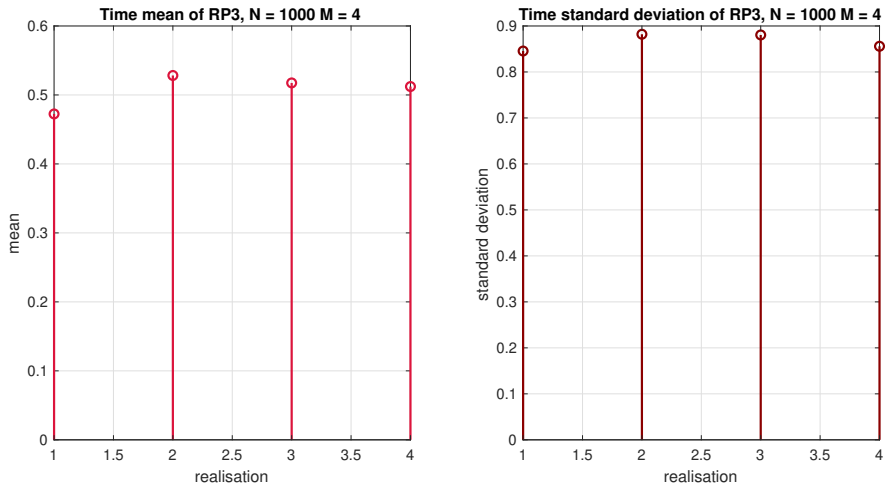

Figure 13: Time mean and standard deviation of RP3

Theoretical mean and variance Signal C is a random signal uniformly distributed $s_C(n) \sim \mathcal{U}(-1, 2)$.

$$f_C(n) = (u(n) - 0.5) * m + a = u_3(n) \sim \mathcal{U}(-1, 2)$$

Its mean and variance are calculated as follows:

$$\mu_C = \mathbb{E}\{u_3(n)\} = \int_{-\infty}^{\infty} x\, f_C(x)\, \mathrm{d}x = \int_{-1}^{2} x \frac{1}{3}\, \mathrm{d}x = \left[\frac{x^2}{6}\right]_{-1}^{2} = \frac{1}{2} = 0.5$$

$$\sigma_C^2 = \mathbb{V}ar\{(u(n) - 0.5) * m + a\} = m^2 \mathbb{V}ar\{u(n)\} = \frac{3^2}{12} = \frac{3}{4}$$

The standard deviation is therefore $\sigma_C = \sqrt{\frac{3}{4}} \approx 0.866$.

Notice that the theoretical predictions of mean and standard deviation match the data obtained both from the ensemble and the time data. This confirms that the signal is stationary and ergodic.

1.3 Estimation of probability distributions

1.3.1 Part 1: The function

The code shown below is a matlab function implemented to estimate PDFs. The input to the function is `x`: this is a vector containing the samples of the signal. The function will estimate the PDF from which these samples were drawn.

The outputs of the function are two vectors: `y` and `h_centers`. The pair of values at the same index in y and h_centers give the value and the relative probability of that value occurring.

```
function [y, h_centers] = pdf(x)
l = length(x);

if l ≥ 10000
    n_bins = 100;
elseif l ≤ 1000
    n_bins = 10;
else
    n_bins = round(l/100);
end

[h_counts, h_centers] = hist(x, n_bins);
y = h_counts./trapz(h_centers, h_counts);
end
```

The PDF estimation is done with the objective of finding a trade off between high resolution and high accuracy. Figure 14 shows why this is a relevant consideration to be taken into account:

- subplot 1 : if the number of bins is too low, the resolution is very poor;
- subplot 2: if the average number of samples per bin (10 in this case) is low, the accuracy of the estimate is very poor;
- subplot 3: if the number of bins is `n_bins` ≥ 100 and the average number of samples per bin is ≥ 100, the estimate is really close to the theoretical PDF;
- subplot 4: as the number of samples per bin increases, the estimate asymptotically converges towards the theoretical PDF.

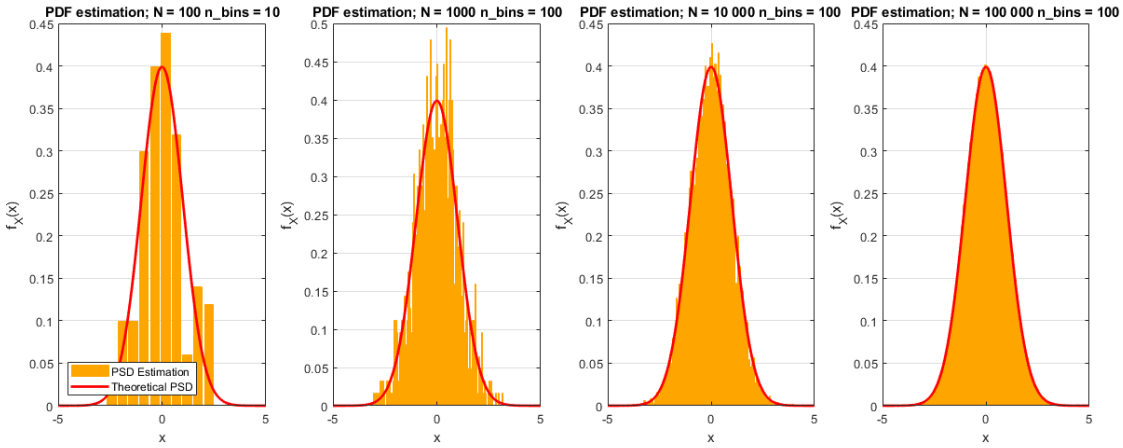

Figure 14: PDF estimation accuracy, varying N and n_bins

As a consequence, the code calculates the number of bins in the estimate `n_bins` based on the number of samples (`length(x)`): this is the desired resolution for the estimation.

Note that in the code (line 13), the number of samples counted in each interval is normalised so that the total area under the curve is equal to 1.

Figure 15 shows that the function `pdf` successfully estimates the PDF of a standard normal random variable for any number of samples, asymptotically providing a better estimate as N increases.

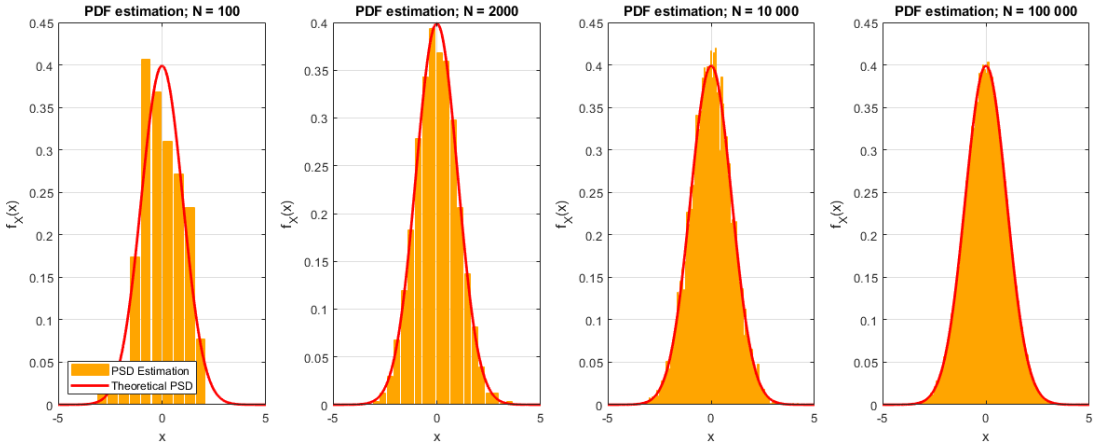

Figure 15: PDF estimation with the matlab function pdf, varying N

1.3.2 Part 2: Estimation of stationary ergodic signals

The function is tested by estimating the PDF of the *random process 3* from section 1.2.3. Figure 16 shows that the function is indeed able to asymptotically provide an estimate of the distribution as N increases. The estimated PDF converges towards the theoretical, whose mathematical expression follows:

$$f_C(x) \sim \mathcal{U}(-1,2) = \begin{cases} \frac{1}{3} & -1 \leq x \leq 2 \\ 0 & otherwise \end{cases}$$

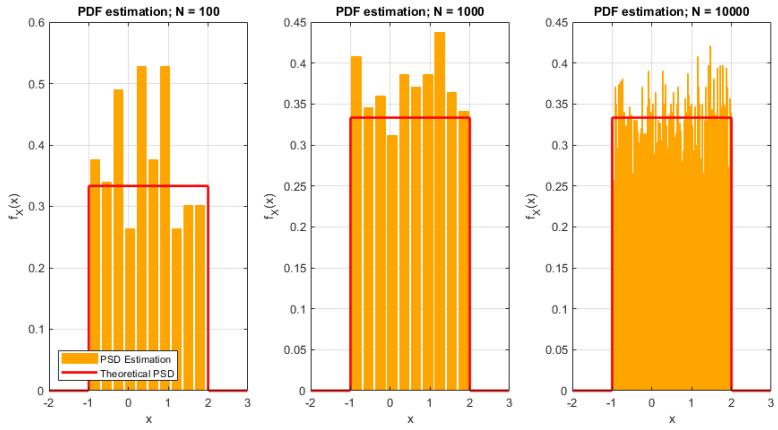

Figure 16: PDF estimation of RP3, varying N

1.3.3 Part 3: Estimation of non-stationary signals

The ability of the function to estimate the PDF of a non-stationary process is tested. The function is applied with an input vector `v`. `v(1:500)`$\sim \mathcal{U}(-0.5, 0.5)$, `v(501:1000)`$\sim \mathcal{U}(0.5, 1.5)$. As it can be noticed, the function fails to estimate the PDF. This is because the matlab function `hist` inherently uses a time average of the signal to produce an estimate. This compromises the ability of the function to discriminate between different distributions.

The correct PDF could be computed by separating the signal into v_1 = `v(1:500)` and v_2 = `v(501:1000)`, and separately estimating the PDFs. As it can be seen from Figure 18, this enables to correctly identify hte distributions. This is because the non-stationary signal `v` is separated into two stationary signals v_1 and v_2.

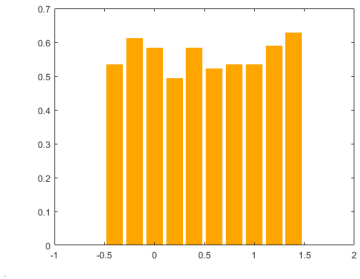

Figure 17: Wrong PDF estimation of the non-stationary signal

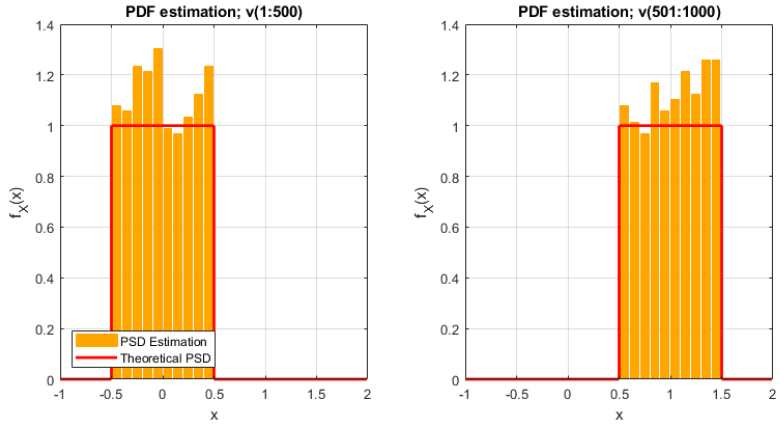

Figure 18: Correct PDF estimation of the non-stationary signal

2 Linear stochastic modelling

2.1 Autocorrelation function

The autocorrelation function (ACF) is a measure of the degree of similarity between a signal and its shifted version. In the general case, it is calculated according to Formula 7.

For ergodic signals, the correlation is calculated from the relative frequency perspective, as given in Formula 8.

$$R_X(m,n) = \mathbb{E}\{x[m]x[n]\} = \int_{-\infty}^{\infty} x[m]x[n]p(x[m],x[n])\,\mathrm{d}x[m]\mathrm{d}x[n] \tag{7}$$

$$R_X(\tau) = \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} x[n]x[n+\tau] \qquad \tau = -N+1, ..., N-1 \tag{8}$$

2.1.1 Parts 1,2,3: White Gaussian Noise

The autocorrelation of White Gaussian Noise (WGN) `x` is calculated with the matlab function `xcorr(x, 'unbiased')` according to Formula 9. `x` is a sequence of L samples drawn from a standard normal distribution.

$$\hat{R}_X(\tau) = \frac{1}{N-|\tau|} \sum_{n=0}^{N-|\tau|-1} x[n]x[n+\tau] \qquad \tau = -N+1, ..., N-1 \tag{9}$$

As it can be noticed from Figure 19, the autocorrelation has the following characteristics:

- a delta function of height 1 is present at zero. This is present because the unshifted signal is identical to itself;
- at $\tau \neq 0$, the autocorrelation is ≈ 0 because different realisations of WGN are independent, therefore uncorrelated.
- for input x[n] of length L, the length of the output $\hat{R}_X(\tau)$ is $(2L-1)$.

The analysed signal is a WSS (wide sense stationary) process, therefore the autocorrelation function $R(\tau)$ has the following properties:

1. shift invariant: $R(m,n) = R(m-n,0) = R(m-n)$;
2. symmetric: $R(-\tau) = R(\tau)$
3. maximum at 0: $R(0) \geq |R(\tau)|$

The comparison between the first and second plot in figure 19 shows that the estimate is much mote accurate for small $|\tau|$ (subplot 2). This is explained by looking at Formula 9. For large $|\tau|$ very few data points are considered to obtain the estimate, making it very inaccurate and variable over realisations. The estimate can be considered reliable over the interval $\pm\frac{N}{2}$.

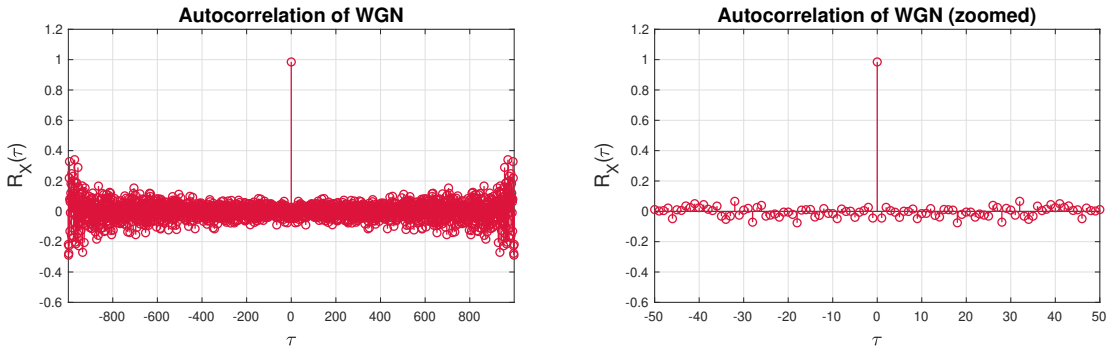

Figure 19: Autocorrelation function of white gaussian noise

2.1.2 Parts 4,5: Moving average filter

The signal `x` is filtered through a moving average digital filter of order 8 (finite impulse response filter). The filter's transfer function is $H(z)$. The output signal is $y[n]$:

$$H(z) = \sum_{i=0}^{8} b_i z^{-i} \qquad y[n] = \sum_{i=0}^{8} b_i x[n-i] \qquad b_i = 1,\ \forall i$$

The autocorrelation of the output `y` is obtained according to the following equation. The result is displayed in Figure 20.

$$\hat{R}_X(\tau) = \frac{1}{N-|\tau|} \sum_{n=0}^{N-|\tau|-1} y[n] * y[n+\tau] = \frac{1}{N-|\tau|} \sum_{n=0}^{N-|\tau|-1} \Big(\sum_{i=0}^{8} x[n-i] * \sum_{i=0}^{8} x[n-i+\tau] \Big)$$

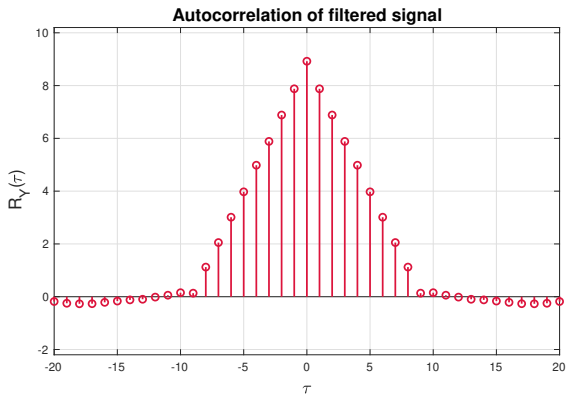

Figure 20: Autocorrelation of MA(8) filtered signal

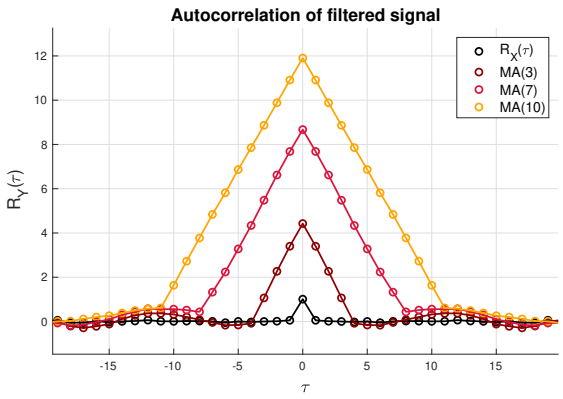

Figure 21: Autocorrelation of filtered signal, varying filter order

The result is easily explained by simplifying the problem to a MA(1) filter.

$$y[n] = \sum_{i=0}^{1} b_i x[n-i] = b_0 x[n] + b_1 x[n-1]$$

$$\hat{R}_Y(\tau) = \frac{1}{N-|\tau|} \sum_{n=0}^{N-|\tau|-1} y[n] * y[n+\tau] = \frac{1}{N-|\tau|} \sum_{n=0}^{N-|\tau|-1} \Big(b_0 x[n] + b_1 x[n-1]\Big) * \Big(b_0 x[n+\tau] + b_1 x[n-1+\tau]\Big) =$$

$$= \frac{1}{N-|\tau|} \sum_{n=0}^{N-|\tau|-1} \Big(b_0^2 x[n]x[n+\tau] + b_0 b_1 x[n]x[n-1+\tau] + b_0 b_1 x[n-1]x[n+\tau] + b_1^2 x[n-1]x[n-1+\tau]\Big)$$

$$\hat{R}_Y(0) = \frac{1}{N} \sum_{n=0}^{N-1} \Big(b_0^2 x[n]x[n] + b_0 b_1 x[n]x[n-1] + b_0 b_1 x[n-1]x[n] + b_1^2 x[n-1]x[n-1]\Big) = b_0^2 \hat{R}_X(0) + b_1^2 \hat{R}_X(0)$$

$$\hat{R}_Y(\pm 1) = \frac{1}{N-1} \sum_{n=0}^{N-2} \Big(b_0^2 x[n]x[n\pm 1] + b_0 b_1 x[n]x[n-1\pm 1] + b_0 b_1 x[n-1]x[n\pm 1] + b_1^2 x[n-1]x[n-1\pm 1]\Big) = b_0 b_1 \hat{R}_X(0)$$

If $b_0 = b_1 = 1$, the autocorrelation function is:

$$\hat{R}_Y(\tau) = \begin{cases} 2\hat{R}_X(\tau) & \tau = 0 \\ \hat{R}_X(\tau) & |\tau| = 1 \\ 0 & otherwise \end{cases}$$

This result is thus extended to a filter of order M, as obtained in the equation that follows. The result matches the empirical data obtained in Figure 21.

$$\hat{R}_Y(\tau) = \begin{cases} (M+1-|\tau|)\,\hat{R}_X(\tau) & |\tau| \leq M \\ 0 & otherwise \end{cases}$$

The autocorrelation of the ouptut of an LTI system $R_Y(\tau)$ is given in Formula 10. In the formula, $R_X(\tau)$ is the autocorrelation of the input signal, $R_H(\tau)$ is the autocorrelation of the impulse response of the filter and $\star$ denotes convolution.

If $R_X(\tau)$ is an uncorrelated process (autocorrelation is a δ-function), the autocorrelation of the output is simply the autocorrelation of the impulse response of the system $R_Y(\tau) = R_H(\tau)$. In fact, the convolution of a signal with a δ-function leaved the signal unchanged.

$$R_Y(\tau) = R_X(\tau) \star R_H(\tau) \tag{10}$$

2.2 Cross-correlation function

The cross-correlation function (CCF) is a measure of the similarity between a signal $x[n]$, and the shifted version of another signal $y[n+\tau]$. This function is defined as the expectation of the product of delayed samples of two stochastic processes (Equation 11).

$$R_{XY}(m,n) = \mathbb{E}\{x[m]y[n]\} = \int_{-\infty}^{\infty} x[m]y[n]p(x[m],y[n])\,\mathrm{d}x[m]\mathrm{d}y[n] \tag{11}$$

As for the autocorrelation function (Section 2.1), a simplification can be applied when dealing with ergodic data. In this case, the cross-correlation is obtained from the time samples rather than from different realisations at the same time instant. For finite length ergodic signals, the cross-correlation is obtained with the unbiased estimate given in Formula 12.

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

2.2.1 Part 1: Moving average filter

The cross-correlation between the input and output of a filter is given in Equation 13.

$$\hat{R}_{XY}(\tau) = h(\tau) \star \hat{R}_X(\tau) \tag{13}$$

If the input signal is uncorrelated, then the cross-correltion will be the impulse response of the filter: $\hat{R}_{XY}(\tau) = h(\tau)$.

This is tested with the signals obtained in the previous section (Section 2.1.2):

- the input $x[n]$ is WGN. This signal has been proven to be uncorrelated.
- the impulse response of the LTI system $h[n]$

In Figure 22, the red data points were obtained by calculating the cross-correlation with the formula for the general case given in Equation 12; the orange line is obtaied by convolution of the impluse response and the input autocorrelation. This shows that the relation of Formula 13 is indeed true.

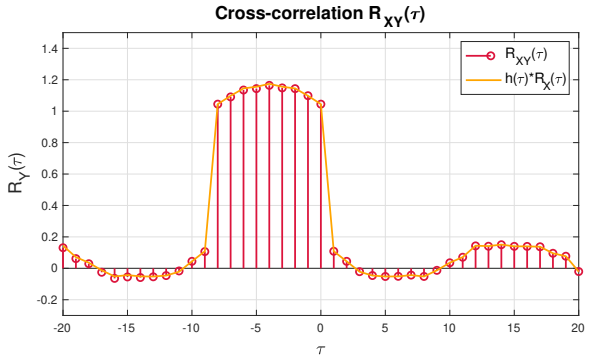

Figure 22: Cross-correlation of the input x(n) and output y(n) of a LTI system h(n)

2.2.2 Part 2: System identification

The relation explained in Formula 13 can be exploited to identify the impulse response of a system. One can obtain the impulse response of an unknown LTI system by feeding it with an input that is known to be uncorrelated, and calculating the cross-correlation of the input and the obtained output.

The number of non-zero samples of the cross-correlation function is the order of the system. As the signals are noisy, the values can not be exactly 0. Therefore, a threshold should be chosen to obtain an accurate enough description of the system.

The estimate asymptotically tends towards the real response of the system as the length of the input and output sequence increases, thus increasing the accuracy of the ACF and CCF estimates.

2.3 Autoregressive modelling

2.3.1 Part 1: AR(2) model

A general pole-only autoregressive process AR(p) is described by Equation 14, where $\mathbf{x}[n]$ is the vector of the input and its delayed samples, $\mathbf{a}$ is the vector of AR coefficients and $w[n]$ is WGN.

$$x[n] = \sum_{i=1}^{p} a_i x[n-i] + w[n] = \mathbf{a}^T \mathbf{x}[n] + w[n] \tag{14}$$

The input-output relationship of an AR(2) process is described by a difference equation (Formula 15) and a transfer function relation (Formula 16).

$$x[n] = a_1 x[n-1] + a_2 x[n-2] + w[n] = \begin{bmatrix} a_1 & a_2 \end{bmatrix} \begin{bmatrix} x[n-1] \\ x[n-2] \end{bmatrix} + w[n] \tag{15}$$

$$H(z) = \frac{X(z)}{W(z)} = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2}} = \frac{z^2}{z^2 - a_1 z - a_2} \tag{16}$$

To obtain a signal with finite variance (power), the vector $\mathbf{a}$ must belong to a domain called stability domain, to ensure that the roots of the transfer function's denominator $D(z) = z^2 - a_1 z - a_2$ are located inside the unit circle. The roots are given by

$$z_{1,2} = \frac{a_1 \pm \sqrt{a_1^2 + 4a_2}}{2}$$

To ensure that both lie inside the unit circle, the following three conditions need to be satisfied:

1. $\mathbf{a_1 + a_2 < 1}$: if the roots are real, the bigger root must be < 1:

$$\frac{a_1 + \sqrt{a_1^2 + 4a_2}}{2} < 1 \quad \rightarrow \quad a_1^2 + 4a_2 < (2 - a1)^2 \quad \rightarrow \quad a_2 < 1 - a1$$

2. $\mathbf{a_2 - a_1 < 1}$: if the roots are real, the smaller root must be > 1:

$$\frac{a_1 \sqrt{a_1^2 + 4a_2}}{2} > -1 \quad \rightarrow \quad a_1^2 + 4a_2 < (2 + a1)^2 \quad \rightarrow \quad a_2 < 1 + a1$$

3. $\mathbf{-1 < a_2 < 1}$: if the roots are real, the denominator can be written as $D(z) = (z - z_1)(z - z_2) = z^2 - (z_1 + z_2) + z_1 z_2$.
   Stationarity requires that $|z_i| < 1; \quad i = 1, 2$.
   Equating coefficients gives $a_1 = (z_1 + z_2)$ and $a_2 = -z_1 z_2$.
   If $|z_1| < 1$ and $|z_2| < 1$, then $|z_1 z_2| < 1$, therefore $|a_2| < 1$.

These results are confirmed by checking for convergence of an AR(2) process, where $a_1 \sim \mathcal{U}(-2.5, 2.5)$ and $a_2 \sim \mathcal{U}(-1.5, 1.5)$. 5000 realisations with different a_1-a_2 pairs are tested and the results are plotted in Figure 23. The orange '*' shows the pairs for which the output sequence $x[n]$ diverges, the dark red '*' shows the pairs for which the output sequence $x[n]$ converges. It can be noticed that the empirical data perfectly matches the theoretical prediction (plotted in red lines).

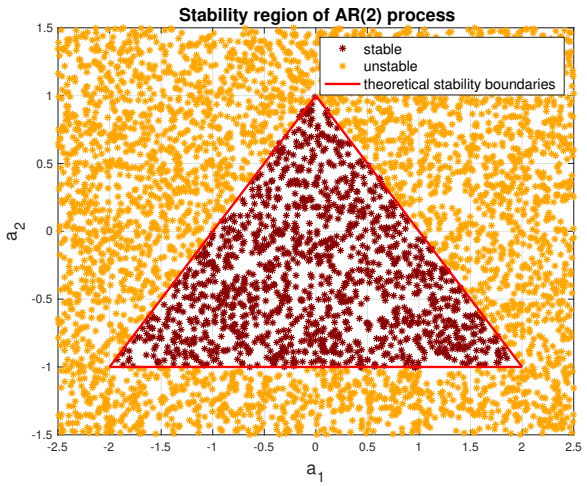

Figure 23: Stability region of AR(2) process

Note that the roots are real if $a_1^2+4a_2 \geq 0$, complex if $a_1^2+4a_2 < 0$. This gives the parabolic boundary $a_1^2+4a_2 = 0$.

2.3.2 Part 2: Sunspot time series

The sunspot time series is uploaded with the matlab command `load sunspot.dat`, where the sunspot data is contained in the second column of the variable.

The autocorrelation of the series is calculated under different conditions:

- for varying data lengths: $N = 5, N = 20$ and $N = 250$;
- in each case, for the data with non-zero mean and for the data normalised to have zero mean.

The results are plotted in Figure 24. It can be noticed that removing the DC offset of the data allows to reveal the real pattern of the ACF.

Other relevant considerations are that, as explained in the Autocorrelation section (2.1), the estimation becomes asymptotically more accurate ad the sample size N increases. Moreover, the estimation becomes very poor for high $|\tau|$, leading to unpredictable behaviour.

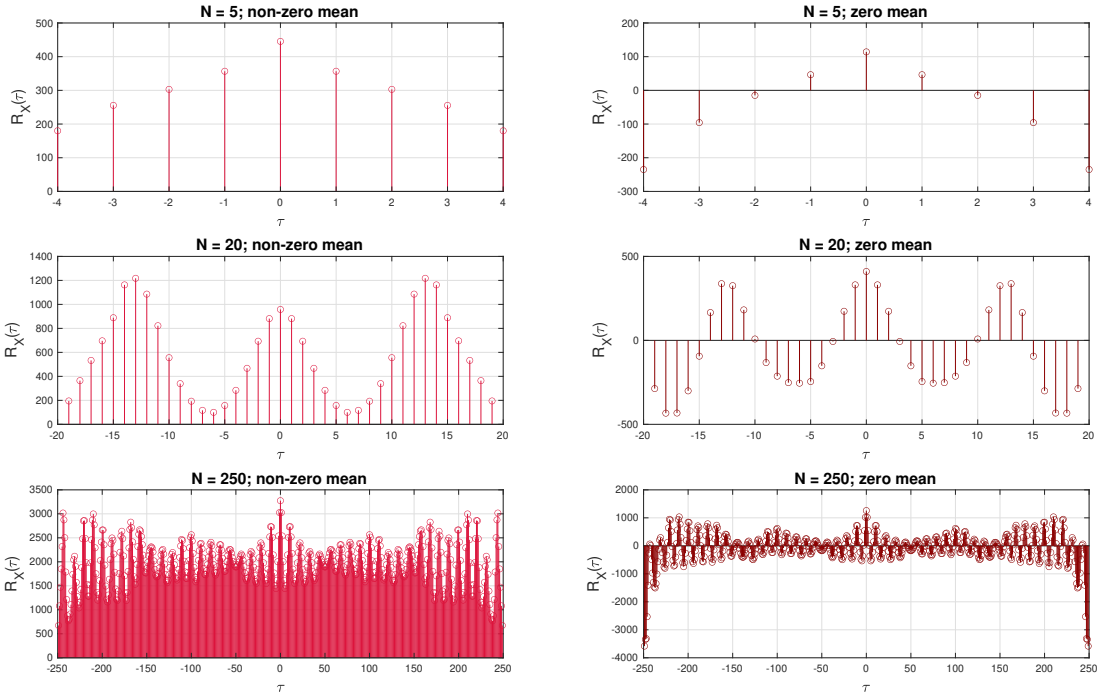

Figure 24: ACF of sunspot time series

2.3.3 Part 3: Model order selection with Yule-Walker equations

The Yule-Walker equations are used to calculate the AR coefficient following the steps:

1. start from data $x[n]$;
2. find the normalised autocorrelation coefficients $\rho_{XX}(\tau) = \frac{R_{XX}(\tau)}{R_{XX}(0)}$
3. solve the Yule-Walker equations to obtain the AR parameters

In practice, this is implemented with the matlab function `[a,e,rc] = aryule(x,p)`, which returns the normalised AR parameters a corresponding to a model of order p for the input array x. e is the estimated variance of the white noise input and rc are the reflection coefficients (partial autocorrelation coefficients scaled by –1).

The AR coefficients for the sunspot time series and the normalised sunspot time series (zero mean and unit variance) with model order 1 to 10 are plotted in Figure 25. It can be noticed that the both the raw and the normalised signals are better approximated by an AR(2) process, since higher coefficients have negligible values. Note that in the plot n is the number of coefficients, the model order $p = n - 1$.

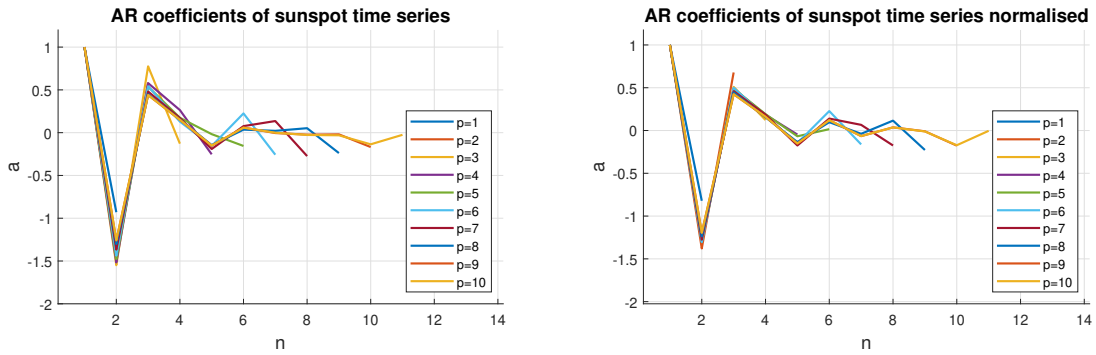

Figure 25: AR coefficients of sunspot time series

Figure 26 shows the partial autocorrelation coefficients for both versions of the signal (normalised and not). The partial autocorrelation function (PACF) of a signal $x[n]$, denoted by $\alpha[k]$ is the autocorrelation between $x[n]$ and $x[n+k]$ with the linear dependence of $x[n]$ on $x[n+1]$ through $x[n+k-1]$ removed. This function is used to identify the correct model order for AR order selection, by looking for the point on the plot where the partial autocorrelations for all higher lags are essentially zero.

It can be noticed that both the original signal and the normalised signal are well represented by an AR(2) model, although the approximation is better for the normalised sunspot time series data, since the values of the PACF for $k > 2$ are lower.

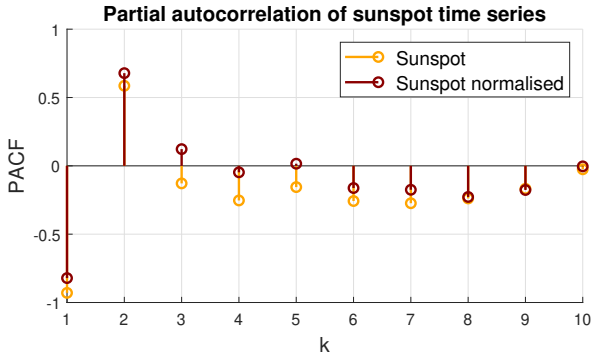

Figure 26: Partial autocorrelation of sunspot time series

2.3.4 Part 4: Model order selection with MDL, AIC and AIC$_c$

One normally does not know the model order to choose before trying out different possibilities, which would be computationally inefficient for large data sets. There are methods to predict the correct order in AR modelling, establishing a trade-off between computational complexity and accuracy, by introducing a "penalty" for higher order selection.

Figure 27 illustrates model order selection for the sunspot data used in the previous sections. The black line is a plot of $\log E_p$ (log of the cumulative squared error, i.e. the error in prediction). This is clearly a monotonically decreasing function. However, the error changes drastically when going form $p = 1$ to $p = 2$, while increasing the model order further does not lead to a significant improvement in the estimate.

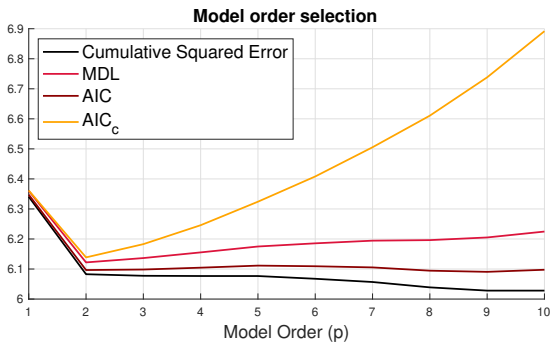

Figure 27: Model order selection with MDL, AIC and AIC(corrected)

The main criteria for model order estimation are defined as follows, and their performance when applied to the sunspot data is commented based on Figure 27:

- minimum description length

$$MDL = \log E_p + \frac{p \log N}{N}$$

For the sunspot data, MDL shows an absolute minimum at $p = 2$.

- Akaike information criterion

$$AIC = \log E_p + \frac{2p}{N}$$

For the sunspot data, AIC shows an absolute minimum at $p = 9$. However, the difference between the value of the function at 2 and the value of the function at 9 is only $AIC(2) - AIC(9) = 6.097 - 6.091 = 0.006$. In this case a model order 2 is indeed more appropriate, because order 9 provides a negligible error improvement but a significant additional computation effort. Therefore, model order selection should be done with the corrected version of the Akaike information criterion corrected version, which favours lower model order selection.

- corrected Akaike information criterion

$$AIC_c = AIC + \frac{2p(p+1)}{N-p-1}$$

For the sunspot data, AIC_c shows an absolute minimum at $p = 2$. This is because, compared to AIC, the corrected version favours lower order selection by adding an extra correction term.

2.3.5 Part 5: Sunspot time series predictions

AR modelling can be used to predict the values of a signal m steps ahead. This is tested with she sunspot time series, for a combination of model orders and number of steps. Model orders of 1, 2 and 10 are tested, each for 1, 2, 5 and 10 steps ahead in the prediction.

For the AR(1) modelling the results are plotted in Figure 28. It can be noticed that the prediction is not good for any number of steps. This is because a model order $p = 1$ is not sufficient for modelling the signal, as was illustrated in the section above. As one would expect, the estimate becomes worse as m increases.

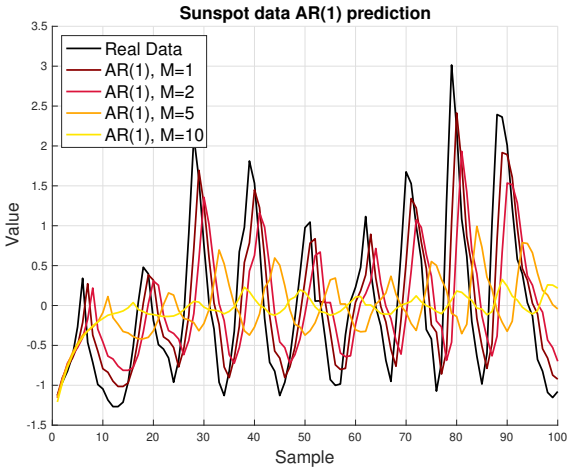

Figure 28: AR(1) modelling of sunspot data

With a model order of 2 (Figure 29) the signal is well approximated 1 and 2 steps ahead. The estimate for 5 and 10 samples ahead is not accurate because the AR(2) process does not capture the dependence of each point with the samples 5 and 10 steps before.

$p = 10$ gives a very good estimate for 1 and 2 steps ahead. In these cases, the performance is comparable to the AR(2) estimate. A model order of 10 does indeed give a better estimate, but the improvement does not justify the increased computational complexity. The estimate for 5 and 10 steps ahead follows the right trend, but can not be accurate as the signal only shows high correlation with samples 2 steps apart.

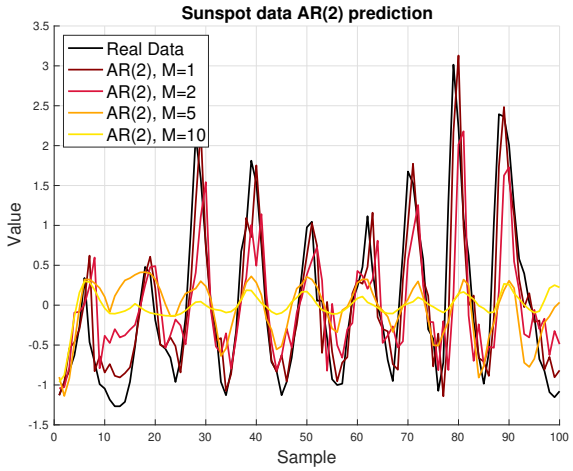

Figure 29: AR(2) modelling of sunspot data

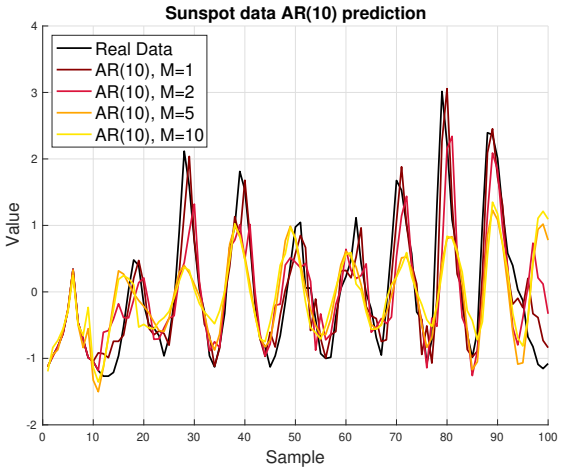

Figure 30: AR(10) modelling of sunspot data

2.4 Cramer-Rao Lower Bound

2.4.1 Part 1a: NASDAQ financial index

The data analysed in this section is the closing prices of the NASDAQ financial index between June 2003 and February 2007. The data as it stands is plotted in Figure 31.

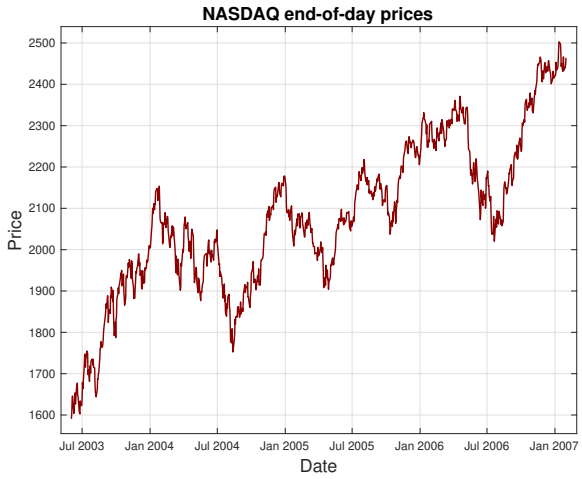

Figure 31: NASDAQ financial index between June 2003 and February 2007

A model order $p = 1$ is sufficient to represent this data. It can bee noticed from Figure 32 that all the methods for model order selection have an absolute minimum at 1, suggesting this is the best compromise between computational efficiency and accuracy.

Figure 33 shows the PACF of the data. It can be noticed that each data $x[n]$ is only strongly related to the one immediately preceding it $x[n-1]$, while having barely no correlation to the previous ones $x[n-k], k > 1$. Again, this suggests that a model of order 1 is sufficient to characterise the system.

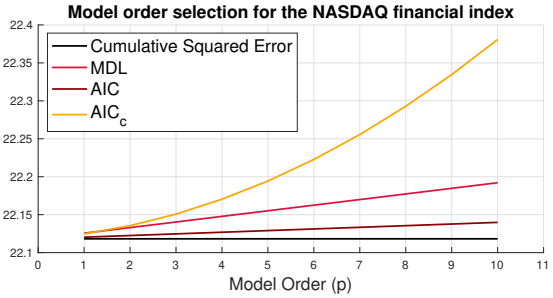

Figure 32: Model order selection with cumulative error

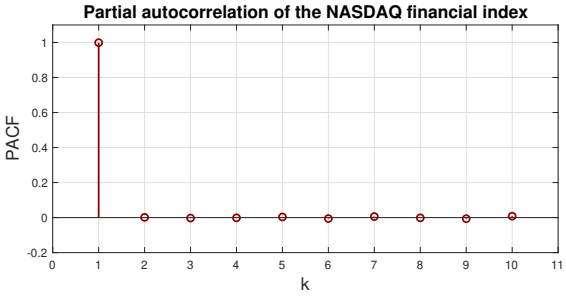

Figure 33: Model order selection with partial autocorrelation

2.4.2 Part 1b: Fisher information matrix

For an input signal $\mathbf{x} \in \mathbb{R}^N$, whose vector of unknown parameters to be estimated is $\boldsymbol{\theta} = \begin{bmatrix}\theta_1 & \theta_2 & \dots & \theta_p\end{bmatrix}^T$, the PDF is denoted as $p(\mathbf{x}; \boldsymbol{\theta})$.

The Cramer-Rao lower bound (CRLB) defines defines a lower bound on the variance of any unbiased estimator. For an unbiased estimator (i.e. $\mathbb{E}\{\hat{\theta}\} = \theta$), that CRLB is defined in Equations 17 and 18.

$$var(\hat{\theta}_i) \geq [\mathbf{I}^{-1}(\boldsymbol{\theta})]_{ii} \tag{17}$$

$$[\mathbf{I}^{-1}(\boldsymbol{\theta})]_{ij} = -\mathbb{E}\Big\{\frac{\partial^2 p(x;\boldsymbol{\theta})}{\partial\theta_i \partial\theta_j}\Big\} \tag{18}$$

In the equation, $i, j = 1, 2, ..., p$, $\mathbf{I}(\boldsymbol{\theta}) \in \mathbb{R}^{pxp}$ is the Fisher information matrix, and the equation is evaluated for the true values of the parameters in $\boldsymbol{\theta}$.

For an autoregressive model, defined in Equation 19, the vector of unknown parameter is defined in Equation 20.

$$x[n] = \sum_{i=1}^{p} a_i x[n-i] + w[n] \qquad w[n] \sim \mathcal{N}(0, \sigma^2) \tag{19}$$

$$\boldsymbol{\theta} = \begin{bmatrix} a_1 & a_2 & \dots & a_p & \sigma^2 \end{bmatrix}^T \tag{20}$$

In this case the task of finding the CRLB is hard, and the asymptotic CRLB is used instead. This is based on the power spectrum of an AR(p) process, defined in Equation 21. The log-likelihood function $\ln[p(\mathbf{x};\boldsymbol{\theta})]$ is replaced by $\ln[\hat{P}(f;\boldsymbol{\theta})]$, defined in Equation 23. The Fisher information matrix becomes as defined in Equation 25.

$$\hat{P}_X(f;\boldsymbol{\theta}) = \frac{\hat{\sigma}^2}{|1 - \sum_{m=1}^{p} \hat{a}_m e^{j2\pi f m}|^2} \tag{21}$$

$$\hat{P}_X(f;\boldsymbol{\theta}) = \frac{\hat{\sigma}^2}{|1 - \hat{a}_1 e^{j2\pi f}|^2} \tag{22}$$

$$\ln[\hat{P}(f;\boldsymbol{\theta})] = \ln[\hat{\sigma}^2] - \ln\left[1 - \sum_{m=1}^{p} \hat{a}_m e^{-j2\pi f m}\right] - \\ - \ln\left[1 - \sum_{m=1}^{p} \hat{a}_m e^{j2\pi f m}\right] \tag{23}$$

$$\ln[\hat{P}(f;\boldsymbol{\theta})] = \ln[\hat{\sigma}^2] - \ln\left[1 - \hat{a}_1 e^{-j2\pi f}\right] - \\ - \ln\left[1 - \hat{a}_1 e^{j2\pi f}\right] \tag{24}$$

$$[\mathbf{I}(\boldsymbol{\theta})]_{ij} = \frac{N}{2} \int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{\partial \ln[\hat{P}(f;\boldsymbol{\theta})]}{\partial \theta_i} \frac{\partial \ln[\hat{P}(f;\boldsymbol{\theta})]}{\partial \theta_j} \mathrm{d}f \tag{25}$$

In the case of an AR(1) model, $\boldsymbol{\theta} = \begin{bmatrix} a_1 & \sigma^2 \end{bmatrix}^T$. Equations 21 and 23 simplify to Equations 22 and 24. $\mathbf{I}(\boldsymbol{\theta})$ is computed according to the above equations to obtain

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

$[\mathbf{I}(\boldsymbol{\theta})]_{22}$ is calculated as follows, based on Equation 25:

$$[\mathbf{I}(\boldsymbol{\theta})]_{22} = \frac{N}{2} \int_{-\frac{1}{2}}^{\frac{1}{2}} \left(\frac{\partial \ln[\hat{P}(f;\boldsymbol{\theta})]}{\partial \theta_2}\right)^2 \mathrm{d}f = \frac{N}{2} \int_{-\frac{1}{2}}^{\frac{1}{2}} \left(\frac{\partial \ln[\hat{\sigma}^2] - \ln\left[1 - \hat{a}_1 e^{-j2\pi f}\right] - \ln\left[1 - \hat{a}_1 e^{j2\pi f}\right]}{\partial \sigma^2}\right)^2 \mathrm{d}f =$$

$$= \frac{N}{2} \int_{-\frac{1}{2}}^{\frac{1}{2}} \left(\frac{1}{\sigma^2}\right)^2 \mathrm{d}f = \frac{N}{2\sigma^4} \int_{-\frac{1}{2}}^{\frac{1}{2}} \mathrm{d}f = \frac{N}{2\sigma^4}$$

2.4.3 Part 1c: obtain the CRLB for the NASDAQ index

For the AR(1) process, therefore, the Cramer-Rao Lower Bounds for $\hat{\sigma}^2$ and $\hat{a}_1$ are obtained from Equation 17 as follows:

$$var(\hat{\theta}_i) \geq [\mathbf{I}^{-1}(\boldsymbol{\theta})]_{ii} \qquad \mathbf{I}^{-1}(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\sigma^2}{N r_{xx}(0)} & 0 \\ 0 & \frac{2\sigma^4}{N} \end{bmatrix} \quad \text{since the matrix is diagonal} \tag{26}$$

$$var(\hat{a}_1^2) \geq [\mathbf{I}^{-1}(\boldsymbol{\theta})]_{11} \quad \rightarrow \quad var(\hat{a}_1^2) \geq \frac{\sigma^2}{N r_{xx}(0)} = \frac{(1 - a_1^2)}{N}, \quad \text{since } r_{xx}(0) = \frac{\sigma^2}{(1 - a_1)^2} \tag{27}$$

$$var(\hat{\sigma}^2) \geq [\mathbf{I}^{-1}(\boldsymbol{\theta})]_{22} \quad \rightarrow \quad var(\hat{\sigma}^2) \geq \frac{2\sigma^4}{N} \tag{28}$$

The CRLB calculation is applied to the NASDAQ AR(1) data.
The results for $\hat{\sigma}^2$ are plotted in Figure 35. It can be noticed that the CRLB grows substantially as σ increases (note that the plot is logarithmic). At fixed sigma, the CRLB is lower for higher N. The relation is however weaker than that with σ. In fact, it can be noticed from Equation 28 that $var(\hat{\sigma}^2)$ is directly proportional to σ^4, while inversely proportional to N.

The results for $\hat{a}_1$ are plotted in Figure 34. The CRLB values are shown to be indeed directly proportional to σ^2 and inversely proportional to N.

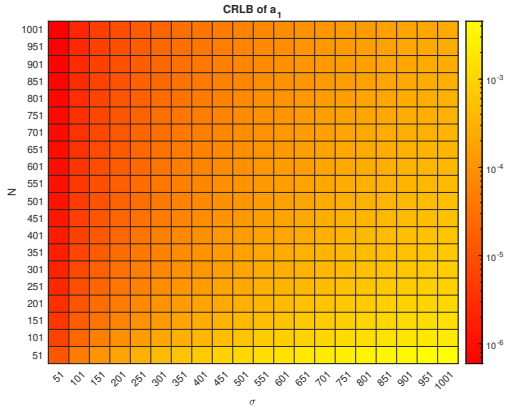

Figure 34: Cramer-Rao lower bound of a_1 (colours scale logarithmically)

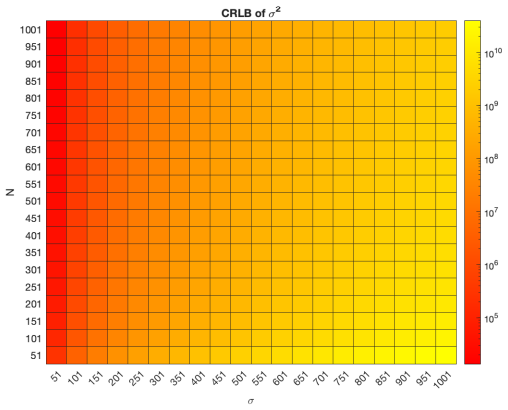

Figure 35: Cramer-Rao lower bound of σ^2 (colours scale logarithmically)

The NASDQ financial data is described by the following equation:

$$x[n] = a_1 x[n-1] + w[n] = -0.9989x[n-1] + w[n]$$

Applying CRLB analysis to this data shows that

$$var(\hat{a}_1) \geq \frac{1}{N}(1-a_1^2) = \frac{1}{924}(1-0.9989^2) = 2.4156e-06$$

It can be noticed from Figure 36 that, as $|a_1| \to 1$, meaning that the pole of the characteristic function approaches the unit circle, $var(\hat{a}_1)$ tends to 0.

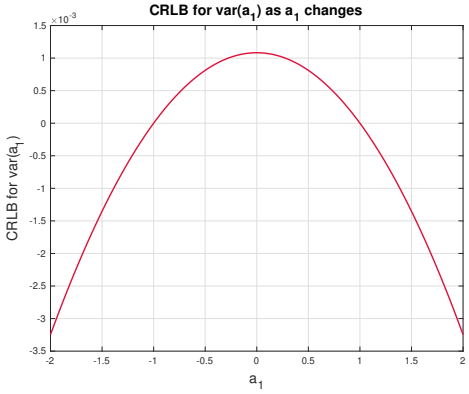

Figure 36: CRLB for $var(\hat{a}_1)$

2.4.4 Part 1d: variance of the PSD

The CRLB can be used to compute the maximum variance of the PSD of the NASDAQ signal, according to Equation 29.

$$var(\hat{P}_X(f;\boldsymbol{\theta})) \geq \frac{\partial \hat{P}_X(f;\boldsymbol{\theta})^T}{\partial \boldsymbol{\theta}} \mathbf{I}^{-1}(\boldsymbol{\theta}) \frac{\partial \hat{P}_X(f;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{29}$$

$$\text{where } \frac{\partial \hat{P}_X(f;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \begin{bmatrix} \frac{\partial \hat{P}_X(f;\boldsymbol{\theta})}{\partial \theta_1} \\ \frac{\partial \hat{P}_X(f;\boldsymbol{\theta})}{\partial \theta_2} \end{bmatrix}$$

Firstly, $\frac{\partial \hat{P}_X(f;\boldsymbol{\theta})^T}{\partial \boldsymbol{\theta}}$ is calculated:

$$\frac{\partial \hat{P}_X(f;\boldsymbol{\theta})}{\partial \theta_1} = \frac{\partial\left(\frac{\sigma^2}{|A(f)|^2}\right)}{\partial a_1} = \frac{-\sigma^2 \frac{\partial}{\partial a_1}|A(f)|^2}{|A(f)|^4} = \frac{-\sigma^2 \frac{\partial}{\partial a_1}\left(1-a_1 e^{-j2\pi f}\right)\left(1-a_1 e^{j2\pi f}\right)}{|A(f)|^4} =$$

$$\frac{\sigma^2\left(e^{-j2\pi f} + e^{j2\pi f} - 2a_1\right)}{|A(f)|^4} = \frac{2\sigma^2\left(\cos(2\pi f) - a_1\right)}{|A(f)|^4}$$

$$\frac{\partial \hat{P}_X(f;\boldsymbol{\theta})}{\partial \theta_2} = \frac{\partial\left(\frac{\sigma^2}{|A(f)|^2}\right)}{\partial \sigma^2} = \frac{|A(f)|^2}{|A(f)|^4} = \frac{1}{|A(f)|^2}$$

Then, the results are substituted in Equation 29:

$$var(\hat{P}_X(f;\boldsymbol{\theta})) \geq \begin{bmatrix} \frac{2\sigma^2\left(\cos(2\pi f)-a_1\right)}{|A(f)|^4} & \frac{1}{|A(f)|^2} \end{bmatrix} \begin{bmatrix} \frac{1-a_1^2}{N} & 0 \\ 0 & \frac{2\sigma^4}{N} \end{bmatrix} \begin{bmatrix} \frac{2\sigma^2\left(\cos(2\pi f)-a_1\right)}{|A(f)|^4} \\ \frac{1}{|A(f)|^2} \end{bmatrix} =$$

$$\frac{2\sigma}{N}\left[\frac{2(\cos(2\pi f))^2(1-a_1^2)}{|A(f)|^8} + \frac{1}{|A(f)|^4}\right]$$

2.5 ECG from iAmp experiment

The recorder ECG signal is converted to RR interval (RRI) data $rr[n]$. The heart rate is obtained from the RRI data as

$$h[n] = \frac{60}{rr[n]}$$

The data $h_1[n]$ and $h_2[n]$ are obtined from $h[n]$. Each value is obtained as the scaled average of 10 values from the original sequence, according to the following formulas:

$$h_1[n] = \frac{1}{10}\sum_{i=1}^{10} \alpha_1 h[10n+i] \qquad \alpha_1 = 1 \qquad\qquad h_2[n] = \frac{1}{10}\sum_{i=1}^{10} \alpha_2 h[10n+i] \qquad \alpha_2 = 0.6$$

2.5.1 Parts a,b: PDF estimate

The PDFs are estimated with the `pdf` function from Section 1.3.1. It can be noticed from subplots 1 and 2 that the PDF of $h[n]$ and $h_1[n]$ are comparable. In fact, when calculating $h_1[n]$ the values are scaled by 1, leaving the data effectively unvaried. This is confirmed by looking at the mean of the distribution (Table 1). It can be noticed that the distributions are centred around the same value. The 0.07 difference is only due to the fact that the last 8 values of $h[n]$ are neglected when calculating $h_1[n]$.

Subplot 3 shows that the PDF of $h_2[n]$ has the same shape, but is centred around a different value. This is because the values of $h[n]$ are scaled by 0.6 to obtain $h_2[n]$. In fact, the **mean** is scaled by 0.6, as from Equation 30. The **variance** is also changed, being smaller in the case of $h_2[n]$. This happens according to Equation 31. Again, the experimental results from Table 1 perfectly match the predictions.

$$\mathbb{E}(aX) = a\mathbb{E}(X) \rightarrow \mathbb{E}(h_2) = 0.6 * \mathbb{E}(h_1) = 0.6 * 84.76 = 50.86 \tag{30}$$

$$\mathbb{V}ar(aX) = a^2\mathbb{V}ar(X) \rightarrow \mathbb{V}ar(h_2) = 0.6^2 * \mathbb{V}ar(h_1) = 0.6^2 * 21.81 = 7.85 \tag{31}$$

signal	mean	variance
$h[n]$	84.83	26.72
$h_1[n]$	84.76	21.81
$h_2[n]$	50.86	7.85

Table 1: Mean and variance of the heart rate signals

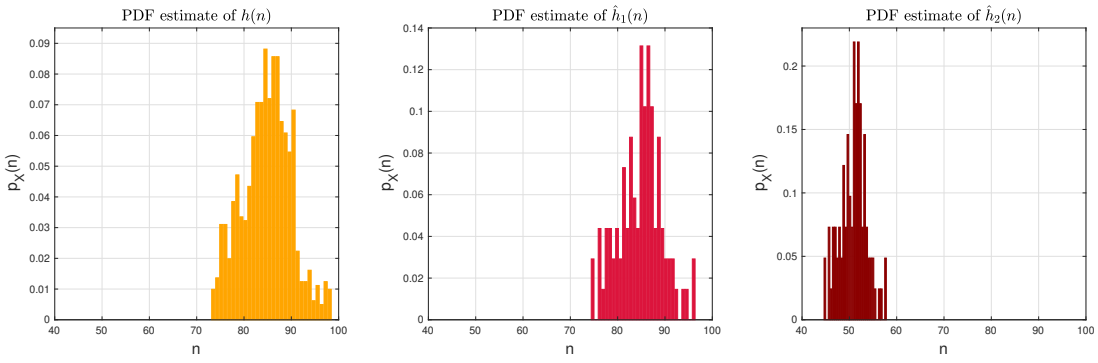

Figure 37: PDF estimate of the heart rate signals

2.5.2 Parts c,d: AR modelling

The autocorrelation functions of the signals $rr1[n]$, $rr2[n]$ and $rr3[n]$ is plotted in Figure38. This shows that the data is an **AR process**. In fact, for AR data the ACF is infinite in length and is a mixture of damped exponential and sinewaves, as happens in this case. If the signal was an MA process, the ACF would have been finite-length.

Figure 39 shows the model order selection methods and partial autocorrelation coefficients for all three RRI segments: $RRI1, RRI2, RRI3$.

- for signal $RRI1$, the most appropriate order is $p = 2$. This is the minimum of all three approximation methods. The partial autocorrelation function also shows negligible values for $k > 2$;

- for $RRI2$ and $RRI3$, the best order is $p = 3$. This is the minimum for MDL and AIC methods. AIC_c has a minimum at 2 because it favours lower model order selection. This indicates that, in case computational effort was a big problem, an AR(2) model would be sufficient to capture the essential features of the signal.
  The PACF confirms this: for $k \leq 2$, the values are significant; for $k = 3$ the vales is small but not negligible; for $k \geq 4$ the values are negligible.

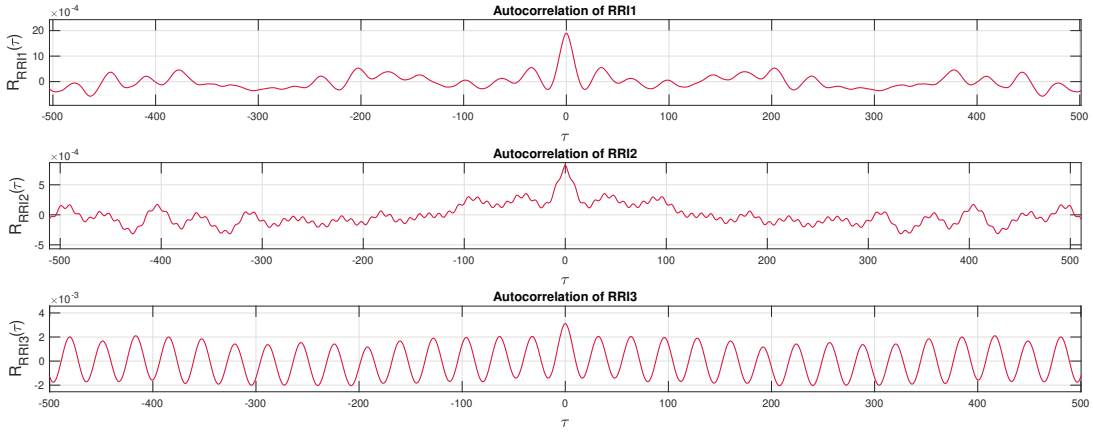

Figure 38: Autocorrelation of the three RRI signal segments

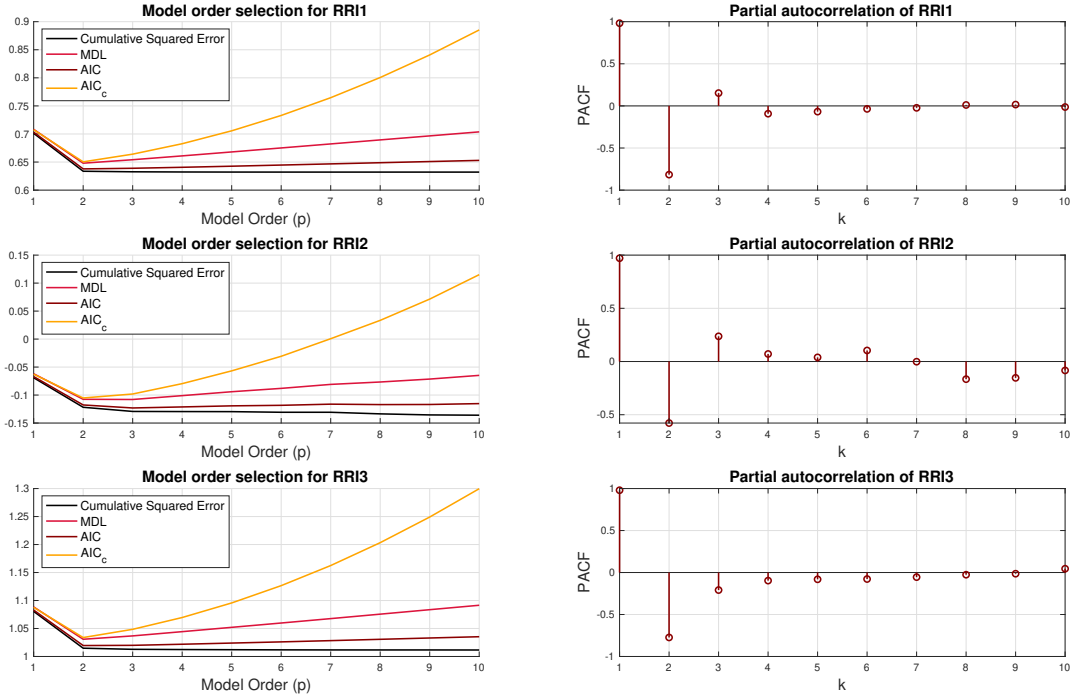

Figure 39: AR model order selection for the three RRI signals

3 Spectral estimation and modelling

The power spectral density (PSD) of a signal is obtained as the absolute value of the Fourier transform of the autocorrelation function, according to Equation 32. For finite length signals, this is approximated according to Equation 33.

$$P_X(f) = \Big| \sum_{\tau=-\infty}^{\infty} R_X(\tau) e^{-j2\pi f\tau} \Big| \qquad f \in [0,1] \qquad (32)$$

$$\hat{P}_X(f) = \Big| \frac{1}{N} \sum_{n=0}^{N} x[n] e^{-j2\pi f \frac{n}{N}} \Big|^2 \qquad (33)$$

Equation 33 is implemented in matlab with the function `p_x = pgm(x)`, which is used to approximate the PSD of an input sequence `x`.

This is tested on a WGN input signal of length N = 128, 256 and 512. The results are shown in Figure 40. It can be noticed that

- for an input signal of length N, the estimated PSD $P_X(f)$ has the same length N;
- for a real input signal (as is the WGN used for testing), $P_X(f)$ is symmetric with respect to π. This corresponds to 0.5 in the normalised frequency plotted in the graphs;
- the estimated PSD is far from the theoretical one, which is $P_X(f) = 1, \forall f$. This is due to the fact that the signal is finite-length and the characteristics of the distribution from which the samples were drawn are not fully captured.

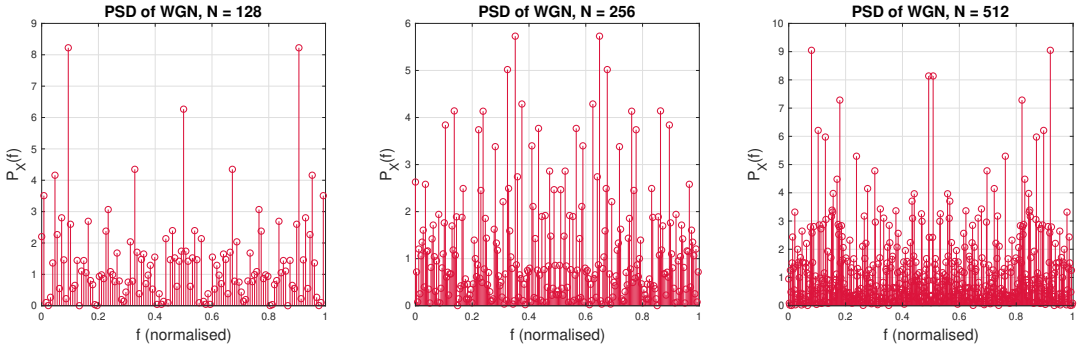

Figure 40: PSD of WGN, varying numer of samples N

3.1 Averaged periodogram estimates

A better PSD estimate for finite-length signals can be obtained by frequency domain smoothing. Two techniques are explored in the following sections: filtering and averaging the periodogram.

3.1.1 Part 1: Filtered periodogram

Firstly, the spectrum obtained with the function `pgm` is filtered with a zero-phase FIR filter with impulse response $h[n] = 0.2 * [1\ 1\ 1\ 1\ 1]$. At the output of this filter, the estimated power at each frequency is the average of the estimate at 5 nearby frequencies.

Figure 41 shows that this does improve the apparent PSD estimate. This however only happens in the particular case where the PSD is known to be a constant value, like in the case of white noise which is considered here. The improvement is only apparent, because the estimate for each frequency is now the average of the estimate for 5 nearby frequencies. If, for example, the PSD was a δ-function, this would be smoothed out to the surrounding frequencies, and the true nature of the PSD function would be hidden.

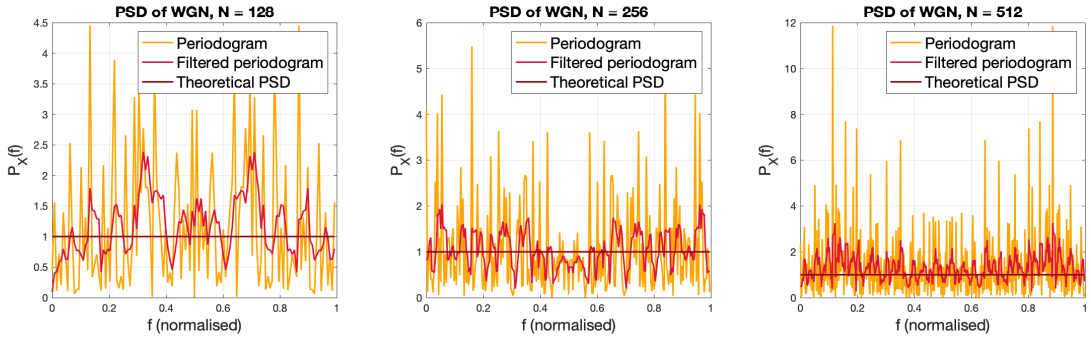

Figure 41: Filtered PSD of WGN

3.1.2 Parts 2,3: Averaged periodogram

The second option that is considered consists in separating the input sequence into 8 non-overlapping segments and calculating their PSD independently (Figure 42). The results are then averaged to obtain an enstimate for the overall sequence, as in Figure 43.

The input data used for this analysis is a WGN input of length $N = 1024$ samples. The plot clearly shows that the approximation obtained with this technique is significantly improved compared to the single-segment analysis.

The PSD approximated with this method is:

- less precise: if the input data had length N and the PSD was approximated in this way, the PSD estimate would have length $\frac{N}{8}$, therefore an estimate every $f = \frac{8}{N},\ 0 \leq f \leq 1$ would be produced.
  If the PSD estimate was obtained directly from the data, the estimate would have length N, with an estimate every $f = \frac{1}{N},\ 0 \leq f \leq 1$;
- correct: the approximation at each data point in only obtained by data at that frequency. No filter is used and the approximation is not distorted
- this method can only be applied if the signal is known to be ergodic, i.e. if its statistics do not vary with time.

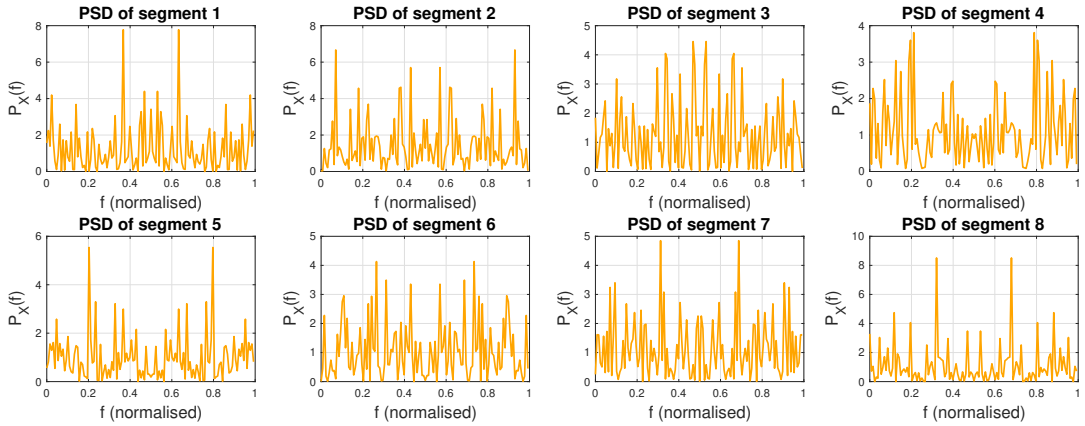

Figure 42: PSD of WGN segments

Figure 43: Averaged PSD of WGN

3.2 Spectrum of autoregressive process

3.2.1 Parts 1,2,3: PSD of AR(1) process

The power spectrum of an AR signal is given by Equation 34. For an AR(1) process, with input variance $\sigma_X^2 = 1$ and coefficient $a_1 = 0.9$, this simplifies to Equation 35.

$$P_Y(f) = \frac{\sigma_X^2}{|1 + \sum_{k=1}^{p} a_k e^{-j2\pi k f}|^2} \tag{34}$$

$$P_Y(f) = \frac{1}{|1 + 0.9e^{-j2\pi f}|^2} \tag{35}$$

$$y[n] = 0.9y[n-1] + w[n] \tag{36}$$

This result is tested with an AR(1) process, given in Equation 36. The result is plotted in Figure 44, and zoomed for higher frequencies in Figure 45. The periodogram is calculated from the data with the `pgm` function from section 3 (orange line), and compared with the theoretical prediction given in Equation 35 (dark red).

It can be noticed that at high frequencies, the PSD differs greatly from the theoretical prediction. This is because the experimental signal is finite in length. This is equivalent to having an infinite length signal windowed with a rectangular function. In the frequency domain, this corresponds to convolving the theoretical PSD with a sinc function. As a consequence, the distortion is much higher at higher frequency, whilst at lower frequency the approximation is valid.

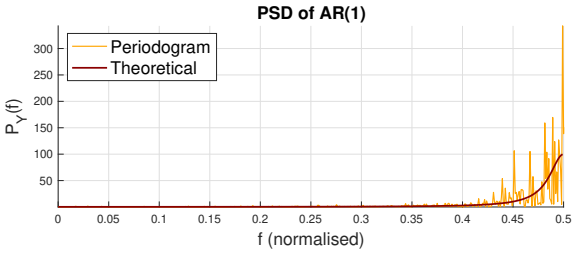

Figure 44: PSD of AR(1) process

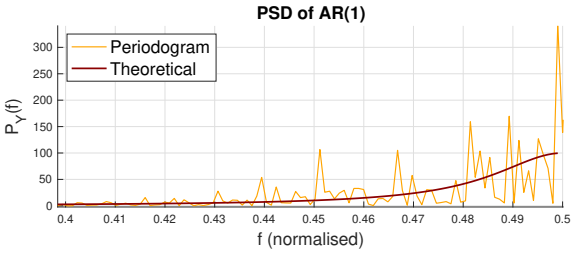

Figure 45: PSD of AR(1) process - zoomed

3.2.2 Part 4: PSD of AR(1) process from model

Figure 46 shows that a much better approximation can be obtained from the AR model of the signal, as in Equation 37. $\hat{\sigma}_X^2$ and $\hat{a}_1$ are estimated from the ACF as in Equation 38.

The estimated PSD is closer to the theoretical PSD because the rectangular window has no effect in the estimate: the error is introduced in estimating $\hat{\sigma}_X^2$ and $\hat{a}_1$, but the shape of the function is correct because the model is known to be AR(1).

In the example from Figure 46, the AR process is given in Equation 36. The estimated $\hat{a}_1$ and $\hat{\sigma}_X^2$ that are used to estimate the PSD are $\hat{a}_1 = 0.9004$ and $\hat{\sigma}_X^2 = 1.1716$. Notice that the estimated autocorrelation coefficient $\hat{a}_1$ is very close to the theoretical one, while the inaccuracy in the estimate of $\hat{\sigma}_X^2$ leads to the error in the PSD estimate.

$$\hat{P}_Y(f) = \frac{\hat{\sigma}_X^2}{|1 + \hat{a}_1 e^{-j2\pi f}|^2} \tag{37}$$

$$\hat{a}_1 = -\frac{\hat{R}_Y(1)}{\hat{R}_Y(0)}$$

$$\hat{\sigma}_X^2 = \hat{R}_Y(0) + \hat{a}_1 \hat{R}_Y(1) \tag{38}$$

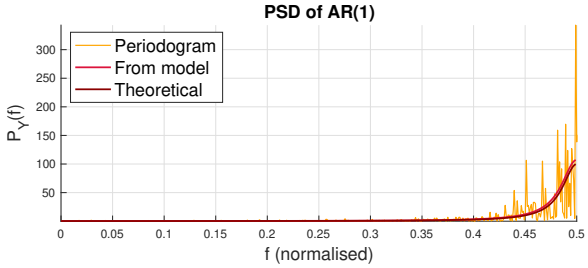

Figure 46: PSD of AR(1) process: from data, from AR model, theoretical

3.2.3 Part 5: PSD of sunspot time series

The same analysis is conducted to estimate the PSD of the original and normalised sunspot data. Again the estimate is obtained both with the periodogram and from the AR model. The theoretical prediction is not plotted because is not known and can only be estimated from the data.

Figure 47 shows the PSD estimate for the original and normalised sunspot data. It can be noticed that in the original data (subplot 1), much of the power is concentrated at DC. Low order estimates only result in modelling the high power components at DC.

Subplot 2 in Figure 47 shows that the model order choice of $p = 2$ is much mode appropriate for the zero-mean data. A model order 2 is sufficient to correctly identify the shape of the PSD. The estimate with model order $p = 10$ is more precise, and allows to capture also smaller peaks at lower frequencies. This however happens at the expense of a relevant increase in computational effort, which is unnecessary compared to the small increase in resolution.

Moreover, using a model order higher than required is very risky. This is well illustrated in Figure 48, which is a zoomed version of the previous plot. The prediction for an AR(60) model is added to the plot. It can be noticed

that **spectral peak splitting** occurs. This consists in wrongly splitting a single spectral peak due to over-modelling the signal.

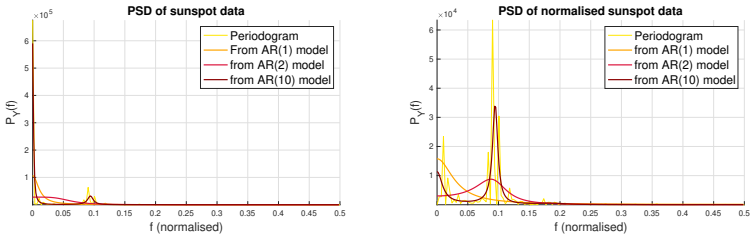

Figure 47: PSD of original and normalised sunspot data

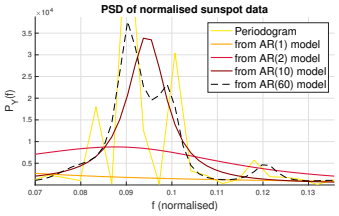

Figure 48: PSD of normalised sunspot data (zoomed)

3.3 Least squares estimation of AR coefficients

3.3.1 Parts 1,2: Theoretical background

Figure 49 shows the general form of a least squares estimation (LSE). In this method, no assumption is made about the data, only a signal model is assumed. The estimators obtained in this way have no optimality properties but ban be applied to a broader range of problems compared to the MVU estimators.

In the LS approach, the aim is to minimise the squared difference between the given data $x[n]$ and the assumed signal $s[n]$. The signal $s[n]$ is generated by some model which depends upon the unknown parameters $\boldsymbol{\theta}$, as in Equation 39.

$$\mathbf{s} = \mathbf{H}\boldsymbol{\theta} \tag{39}$$

The closeness of the estimate is measured by the LS error criterion, given in Equation 40.

$$J(\boldsymbol{\theta}) = \sum_{n=0}^{N-1} (\epsilon[n])^2 = \boldsymbol{\epsilon}^T\boldsymbol{\epsilon} = (\mathbf{x}-\mathbf{s})^T(\mathbf{x}-\mathbf{s}) =$$

$$= (\mathbf{x}-\mathbf{H}\boldsymbol{\theta})^T(\mathbf{x}-\mathbf{H}\boldsymbol{\theta}) \tag{40}$$

In the case of estimating coefficients $\mathbf{a} = [a_1\ a_2\ \dots\ a_p]$ of an AR(p) process, the LS cost function is given by the following equation, where $\hat{r}_{xx}[k]$ is the biased ACF estimate.

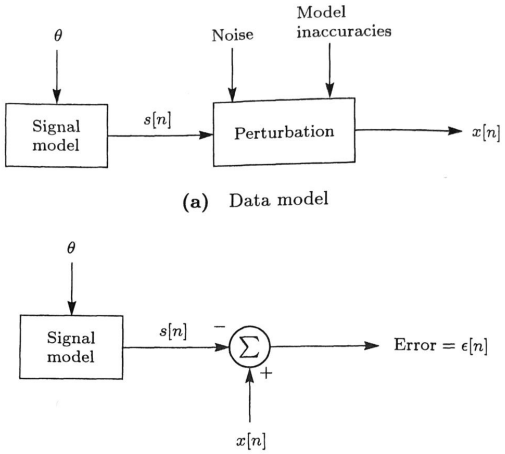

Figure 49: Least Square Estimate diagram

$$J(\boldsymbol{\theta}) = \sum_{k=1}^{M}\left[\hat{r}_{xx}[k] - \sum_{i=1}^{p} a_i\hat{r}_{xx}[k-i]\right]^2 \qquad \text{for } M \geq p$$

This can be written in the form given in Equation 40, where $\mathbf{x}$, $\boldsymbol{\theta}$ and $\mathbf{H}$ are as follows. Then, $\mathbf{s}$ is obtained. Note that $\mathbf{H}$ is normally a deterministic matrix, as it only depends on the input data. As it can be noticed from Figure 49(a), the signal $\mathbf{s} = \mathbf{H}\boldsymbol{\theta}$ is entirely deterministic, whilst the model inaccuracies and the probabilistic noise are accounted for by the error term $\boldsymbol{\epsilon} = \mathbf{x} - \mathbf{s}$. Here however, the matrix is stochastic since its entries are the estimates of the ACF.

$$\mathbf{x} = \underbrace{\begin{bmatrix}\hat{r}_{xx}[1]\\ \hat{r}_{xx}[2]\\ \vdots\\ \hat{r}_{xx}[M]\end{bmatrix}}_{M\times 1} \qquad \mathbf{H} = \underbrace{\begin{bmatrix}\hat{r}_{xx}[0] & \hat{r}_{xx}[-1] & \dots & \hat{r}_{xx}[1-p]\\ \hat{r}_{xx}[1] & \hat{r}_{xx}[0] & \dots & \hat{r}_{xx}[2-p]\\ \vdots & \vdots & \ddots & \vdots\\ \hat{r}_{xx}[M-1] & \hat{r}_{xx}[M-2] & \dots & \hat{r}_{xx}[M-p]\end{bmatrix}}_{M\times p} \qquad \boldsymbol{\theta} = \mathbf{a} = \underbrace{\begin{bmatrix}a_1\\ a_2\\ \vdots\\ a_p\end{bmatrix}}_{p\times 1}$$

$$\mathbf{s} = \mathbf{H}\boldsymbol{\theta} = \mathbf{H}\mathbf{a} = a_1\mathbf{h}_1 + a_2\mathbf{h}_2 + \dots + a_p\mathbf{h}_p = a_1\begin{bmatrix}\hat{r}_{xx}[0]\\ \hat{r}_{xx}[1]\\ \vdots\\ \hat{r}_{xx}[M-1]\end{bmatrix} + a_2\begin{bmatrix}\hat{r}_{xx}[-1]\\ \hat{r}_{xx}[0]\\ \vdots\\ \hat{r}_{xx}[M-2]\end{bmatrix} + \dots + a_p\begin{bmatrix}\hat{r}_{xx}[1-p]\\ \hat{r}_{xx}[2-p]\\ \vdots\\ \hat{r}_{xx}[M-p]\end{bmatrix}$$

In order to estimate the coefficients $\mathbf{a}$, the derivative of $J(\mathbf{a})$ w.r.t $\mathbf{a}$ is computed and set to 0, as follows in Equation 41. The estimate for $\mathbf{a}$ is derived and given in Equation 42.

$$J(\mathbf{a}) = (\mathbf{x}-\mathbf{H}\mathbf{a})^T(\mathbf{x}-\mathbf{H}\mathbf{a}) = (\mathbf{x}^T-\mathbf{a}^T\mathbf{H}^T)(\mathbf{x}-\mathbf{H}\mathbf{a}) = \mathbf{x}^T\mathbf{x}-\mathbf{x}^T\mathbf{H}\mathbf{a}-\mathbf{a}^T\mathbf{H}^T\mathbf{x}+\mathbf{a}^T\mathbf{H}^T\mathbf{H}\mathbf{a} = \mathbf{x}^T\mathbf{x}-2\mathbf{x}^T\mathbf{H}\mathbf{a}+\mathbf{a}^T\mathbf{H}^T\mathbf{H}\mathbf{a}$$

$$\frac{\partial J(\mathbf{a})}{\partial \mathbf{a}} = \frac{\partial(\mathbf{x}^T\mathbf{x} - 2\mathbf{x}^T\mathbf{H}\mathbf{a} + \mathbf{a}^T\mathbf{H}^T\mathbf{H}\mathbf{a})}{\partial \mathbf{a}} = -2\mathbf{x}^T\mathbf{H} + 2\mathbf{H}^T\mathbf{H}\mathbf{a} = 0 \tag{41}$$

$$\mathbf{a} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x} \tag{42}$$

3.3.2 Parts 3,4: Sunspot time series - approximation error varying P

The AR coefficients of the sunspot time series are obtained with the LSE approach, as it was done in section 2.3.3 with the Yuke-Walker equations. Figure 50 is a plot of the calculated coefficients for different model orders. A comparison with those obtained with Yuke-Walker equations in section 2.3.3 shows that these are really similar, and the mall differences are due to inaccuracies introduced by finite-length signals. For this data, 2 was shown to be the most appropriate model order. Therefore, the AR(2) coefficients obtained with the two methods are compared in Table 2. Indeed, the results are practically identical.

Figure 51 is a plot of the logarithm of the cumulative squared error, and the model order selection methods explored in Section 2.3.4 for the same data. Indeed, the results are practically identical, as they are calculated with the same coefficients, just obtained in a different way.

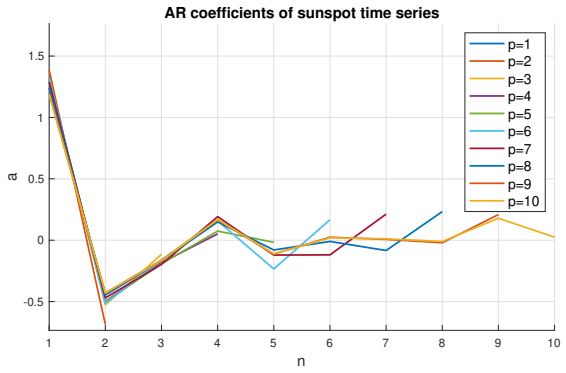

Figure 50: AR coefficient estimation for sunspot data with LSE approach

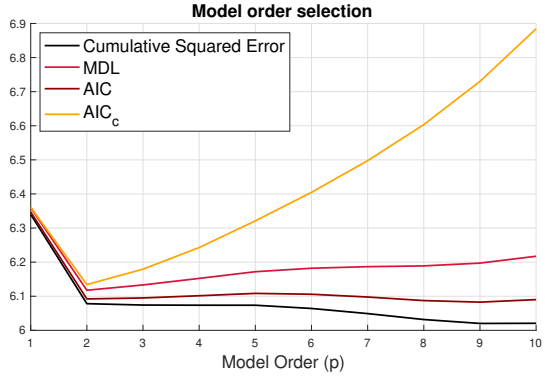

Figure 51: Model order selection for sunspot data with LSE approach

	a1	a2
Yuke-Walker	-1.3782	0.6783
LSE	-1.3863	0.6853

Table 2: Sunspot AR(2) coefficients obtained with Yuke-Walker equations and LSE approach

3.3.3 Part 5: Sunspot time series - PSD

In this section, the AR coefficients obtained with the LSE approach are used to obtain PSD estimates based on the AR models.

As in Section 3.2.3, the PSD is plotted for different model orders. The result is very similar to that obtained when the coefficients were calculated with Yule-Walker equations. In fact, once the same coefficients are calculated, the method for obtaining the PSD is the same and therefore the result is comparable.

As it was explained in Section 3.2.3, this plot confirms that a model order $p = 1$ is insufficient to represent the signal. A model order 2 is appropriate, while an AR(10) process models well the characteristics of the signal but adds useless extra computational complexity.

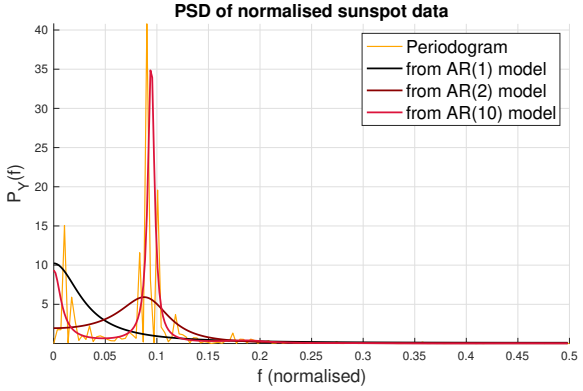

Figure 52: PSD of normalised sunspot data, obtained from LSE AR model

Note that in this section only the PSD for the normalised data is given, as this is more significant in describing the distribution of power across frequencies.

3.3.4 Part 6: Sunspot time series - approximation error varying M

Figure 53 is a plot of the mean squared error (MSE) of the AR(2) estimate for varying signal length $N \in [10 : 5 : 250]$.

It can be noticed that with low data length the AR prediction is very poor. In fact, with few data the coefficients can not be estimated precisely. As the number of data points increases, the estimate becomes better. The image would suggest that the optimal data length is $N = 50$ since the minimum for the error occurs at that point. However, this is just a local minimum, and the function is monotonically decreasing in the long run, so that the best performance of the estimate would be if $N = \infty$.

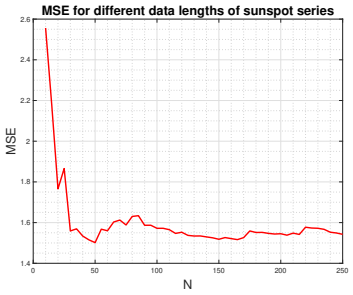

Figure 53: Sunspot MSE

3.4 Spectrogram for time-frequency analysis: dial tone pad

3.4.1 Part 1: Time domain analysis

Time-frequency analysis involves studying the evolution of the frequency content of a signal over time. This is relevant in the Dial Tone Multi-Frequency (DTMF) system, where each signal is composed of two sinusoids and corresponds to a button being pressed, according to Equation 43, with frequencies f_1 and f_2 given in Table 3.

$$y[n] = \sin(2\pi f_1 n) + \sin(2\pi f_2 n) \qquad (43)$$

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

Table 3: Dial pad frequencies f_1 and f_2

A London landline number (020 xxx xxx) is generated. Digits 'x' are randomly selected integers between 0 and 9. The resulting time domain signal is plotted in Figure 54. A zoomed version of the first two digits, with an interval where no digit is pressed, is given in Figure 55. Figure 56 is a zoom on the signals generated when the digits '0' and '2' are being pressed.

The frequency at which the signal is sampled is $f_s = 32768Hz$. Note that the Nyquist frequency required to avoid aliasing in the sampled signal is $f_N = 2f_{max} = 2 * 1477 = 2954Hz$. The sampling frequency is chosen to be ~ 11 times higher because this reduces the signal to noise ratio and allows to identify the frequencies better.

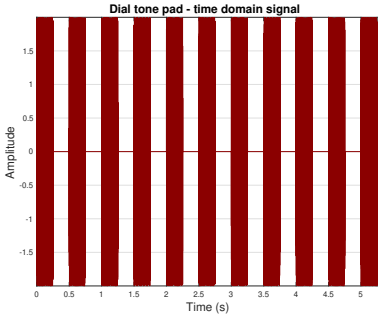

Figure 54: London landline number time domain signal

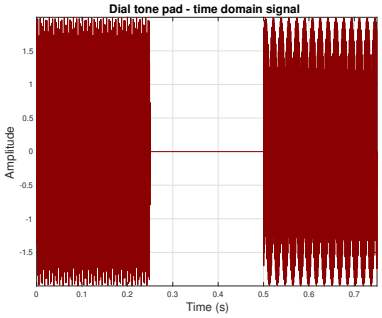

Figure 55: London landline number - Zoomed

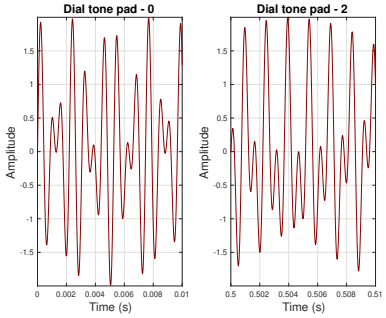

Figure 56: London landline number - Zoomed

3.4.2 Parts 2,3: Frequency domain analysis

The spectrogram for the time signal in FIgure 54 is generated and plotted in Figure 57. The matlab command `spectrogram(out, hann(8192), 0, 8192, 32768, 'yaxis');` is chosen to produce the spectrogram of the signal `out`, dividing it into 0.25s non-overlapping segments. The individual segments are filtered with a Hanning window before the estimate is calculated. This is done to avoid the sharp interruption of the signal, which would correspond

to a convolution with a sinc in the frequency domain. This would introduce unwanted frequency components in the signal.

By analysing the frequencies that are present in the signal at each interval, the numbers can be correctly identified. One could set a threshold at $-40dB/Hz$ to identify a frequency that is actually present in the signal. By identifying both f_1 and f_2, one can reconstruct the dialled sequence number by number by looking at Table 3.

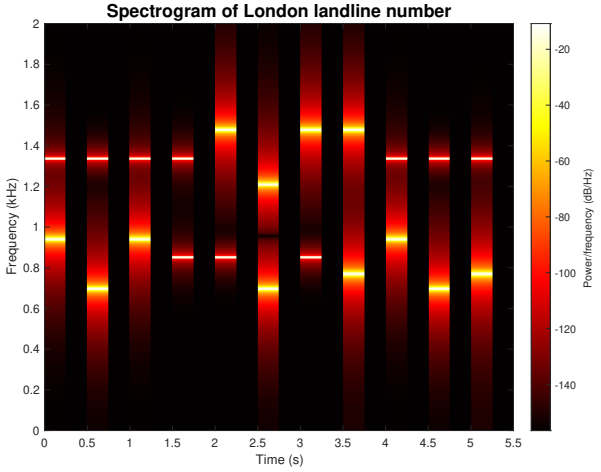

Figure 57: Spectrogram of London landline number

3.4.3 Part 4: Noise corrupted signal

In real applications, the signals would not be recognisable as well, because white noise would be added on top of the sinusoids, transforming Equation 43 into the following:

$$y[n] = \sin(2\pi f_1 n) + \sin(2\pi f_2 n) + w[n] \qquad w[n] \sim \mathcal{N}(0, \sigma_N^2)$$

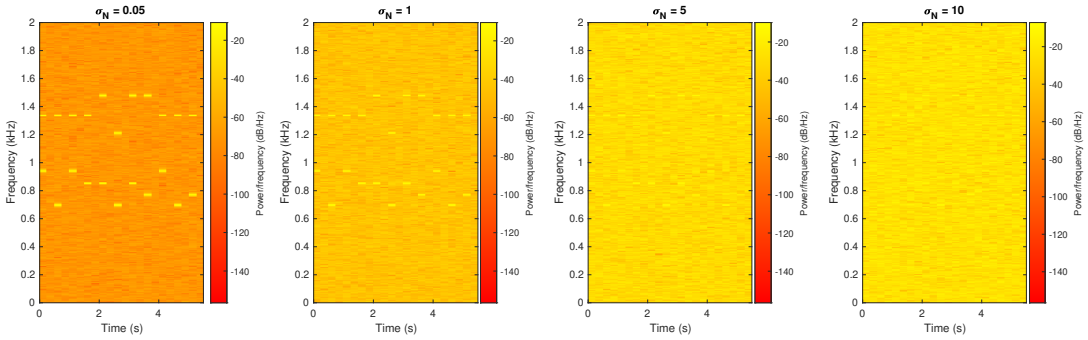

Figure 58: Spectrogram of London landline number with noise

By looking at Figure 58 we observe the spectrogram of the signal as noise with increasing standard deviation is introduced. With low-σ_N noise, it is still possible to recognise which digits are being pressed. With $\sigma_N = 5$, and even more in the case $\sigma_N = 10$, the threshold method proposed in Section 3.4.2 to identify the frequency of the sinusoids can no longer be applied. In fact, the sinusoids are hidden by the high power noise superimposed on the signal, which hides its pattern.

3.5 Respiratory sinus arrhythmia from RR-Intervals

The PSD of the RRI signals already analysed in Section 2.5 is obtained. The three signals are separately analysed:

1. $RRI1$: this signal is obtained with normal breathing. Under normal conditions, breathing normally happens around 16 times per minute, i.e. 0.26 times/second.
2. $RRI2$ 25 breaths per minute. This corresponds to a frequency of 0.417 breaths per second.
3. $RRI3$ 7.5 breaths per minute, or equivalently 0.125 breaths/second.

It can be noticed that in all three cases there is a component in the frequency spectrum that perfectly matches the theoretical data, this confirming that the breathing pattern can be distinguished from the heart signal in the frequency domain, whilst it would be impossible to isolate it in time domain.

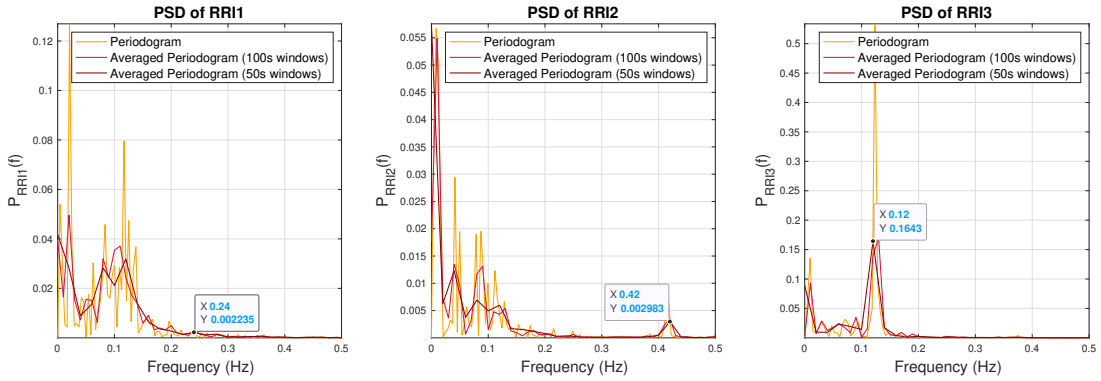

Figure 59: PSD of RRI signals

4 Optimal filtering - fixed and adaptive

4.1 Wiener filter

In this section, system identification is performed according to the diagram in Figure 60.

An optimum Wiener filter is used to obtain the closest estimate of an unknown system. The estimated filter with coefficients $\mathbf{w}_{opt}$ is assumed to be a moving-average (MA) filter of the same order as the unknown system.

The optimum Wiener filter estimate is obtained according to Equation 44, where $\mathbf{p}_{zx}$ and $\mathbf{R}_{xx}$ are as in Equation 45.

$$\mathbf{w}_{opt} = \mathbf{R}_{xx}^{-1}\mathbf{p}_{zx} \tag{44}$$

$$\mathbf{p}_{zx} = \begin{bmatrix} r_{zx}(0) \\ r_{zx}(-1) \\ \vdots \\ r_{zx}(-N_w) \end{bmatrix} \quad \mathbf{R}_{xx} = \begin{bmatrix} r_{xx}(0) & r_{xx}(-1) & \dots & r_{xx}(-N_w) \\ r_{xx}(1) & r_{xx}(0) & \dots & r_{xx}(-N_w+1) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(N_w) & r_{xx}(N_w-1) & \dots & r_{xx}(0) \end{bmatrix} \tag{45}$$

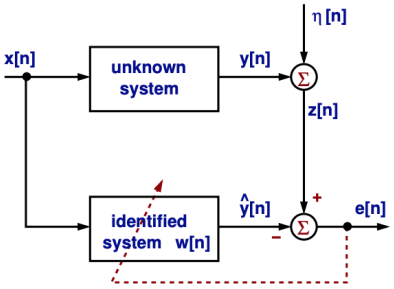

Figure 60: Diagram for system identification

The method described above is tested with 1000-sample WGN input $x[n]$ with power $\sigma_x^2 = 1$; an unknown system with coefficients $\mathbf{b} = [1\ 2\ 3\ 2\ 1]$, $\mathbf{a} = [1]$ output $\mathbf{y}$ normalised to have unit variance $\sigma_y^2 = 1$; disturbance η given by a 1000-sample WGN sequence with variance $\sigma_\eta^2 = 0.1^2$.

The expected signal-to-noise-ratio (SNR) is obtained in Equation 46.

$$SNR_z = \frac{P_y}{P_\eta} = \frac{\sigma_y^2}{\sigma_\eta^2} = \frac{1}{0.01} = 100 = 20dB \tag{46}$$

From obtained SNR_z for 5 different realisations is obtained in Table 4. As it can be seen, the values that are obtained match the theoretical prediction, with a small variability due to the limited number of samples (1000) that are used for the estimate.

Realisation	1st	2nd	3rd	4th	5th
$\mathbf{SNR}_\eta$	$20.09dB$	$19.76dB$	$19.81dB$	$19.92dB$	$20.19dB$

Table 4: SNR_η for 5 system realisations

4.1.1 Part 1: MA filter coefficient estimation

The described method is used to estimate the MA filter coefficients given above, i.e. $\mathbf{b} = [1\ 2\ 3\ 2\ 1]$. The estimated coefficients are as follows:

$$\hat{\mathbf{b}} = \begin{bmatrix} 1.0003 & 1.9994 & 2.9931 & 1.9934 & 1.0007 \end{bmatrix}$$

It can be noticed that the estimate is extremely precise when the analysed signal is analysed with no noise added.

4.1.2 Part 2: Varying noise variance and assumed model order

The same estimate is obtained varying the noise variance σ^2 and the assumed model order N_w.

Firstly, σ^2 in varied in an interval $\in [0.1, 10]$. The effect of varying the noise variance on the SNR is shown in Table 5. It can be noticed that increasing the noise power reduces the SNR, as expected from Equation 46.

The result on the coefficient estimation is plotted in Figure 61. Lower noise variance leads to a better coefficient estimation. This is because with increasing noise power, the actual output $y[n]$ is more and more hidden in noise, and the response of the MA filter is obscured by the stochasticity of WGN.

Noise Variance σ^2	Th. SNR	Exp. SNR
0.1	$10dB$	$10.46dB$
2.08	$-3.18dB$	$-3.15dB$
4.06	$-6.08dB$	$-5.94dB$
6.04	$-7.79dB$	$-7.74dB$
8.02	$-9.04dB$	$-9.12dB$
10	$-10dB$	$-9.95dB$

Table 5: Theoretical and experimental SNR_η varying noise variance

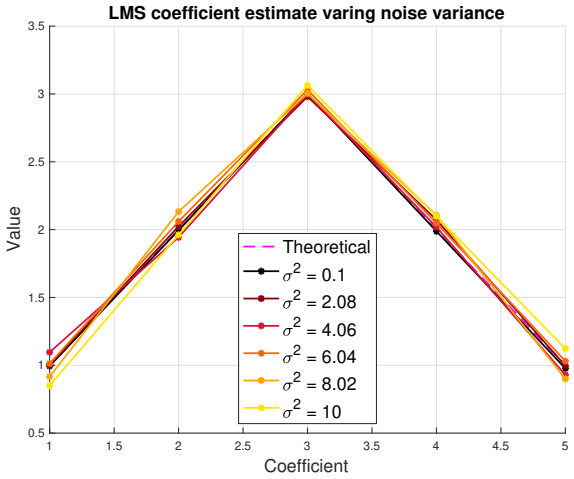

Figure 61: Wiener coefficient estimation varying noise power

Then, the assumed model order N_w is varied, although the filter order of the unknown system is left unchanged. For the higher coefficients, the estimate is really close to 0. This means that while choosing a lower order for the model makes it impossible to reconstruct the initial system, a higher order choice ensures a correct reconstruction of the system. This is an advantage if the order of the unknown system is unchanged, but is to be avoided to reduce the computational overhead.

4.1.3 Computational complexity

The Wiener algorithm involves:

- building $\mathbf{R}_{xx}$ and $\mathbf{p}_{zx}$. Both require $N(N_w+1)$ operations. This is a complexity of $\mathcal{O}(N * N_w)$;
- finding the inverse of $\mathbf{R}_{xx}$, which is a $(N_w+1)\times(N_w+1)$ matrix. This operation has complexity $\mathcal{O}((N_w+1)^3) = \mathcal{O}(N_w^3)$;
- multiplying $\mathbf{R}_{xx}^{-1}\mathbf{p}_{zx}$. The dimensions of the matrices are $[(N_w+1)\times(N_w+1)] * [(N_w+1)\times 1]$. Every entry of the result is a scalar product of two (N_w+1) vectors, which requires (N_w+1) multiplications and $(N_w+1)-1$ additions. This is done $(N_w+1) * 1$ times for every entry of the result. The total number of operations is therefore $(N_w+1) * 1 * (N_w+1+N_w) = 2N_w^2 + 3N_w + 1 = \mathcal{O}(N_w^2)$

The resulting complexity is $\mathcal{O}(N * N_w) + \mathcal{O}(N_w^3) + \mathcal{O}(N_w^2)$. This is equivalent to $\mathcal{O}(N * N_w)$ or $\mathcal{O}(N_w^3)$, whichever is highest.

4.2 The least mean square algorithm

While the Wiener algorithm assumes stationarity of the unknown system, the least mean square (LMS) algorithm uses adaptive filter approximation that recursively estimate the filter coefficients. This method is characterised by Equation 47, where $\mathbf{w}$ is the vector of estimated parameters and $\mathbf{w}(0) = \mathbf{0}$; μ is the adaptation gain; the error $e[n]$ is the difference between the actual output $z[n]$ and the estimated output of the unknown adaptive filter $\hat{y}[n]$, as from Equation 49.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e[n]\mathbf{x}(n) \quad n = 0, 1, ... \tag{47}$$

$$e[n] = z[n] - \hat{y}[n] \tag{48}$$

$$\hat{y}[n] = \mathbf{w}^T(n)\mathbf{x}(n) = \sum_{m=0}^{N_w} w_n(m)x(n-m) \tag{49}$$

4.2.1 Part 1: Varying noise variance

The method is applied to the 1000-sample signals **x** and **z** from Section 4.1, and tested with different SNRs, as was done for the Wiener algorithm in Section 4.1.2. The result is plotted in Figure 62.

What can be observed is very similar to what was noticed with the Wiener algorithm. A higher noise power reduces drastically the quality of the estimate. It can be clearly observed that for darker lines (higher SNR), the results are closer to the actual filter coefficients, shown with the magenta line.

A direct comparison with the performance of the Wiener method (see Figure 61) shows that for any value of noise variance the Wiener algorithm gives a better estimate, due to the fact that the estimate is given based on the whole sequences, while in the LSE case it is constantly updated, leading to a higher variability over time.

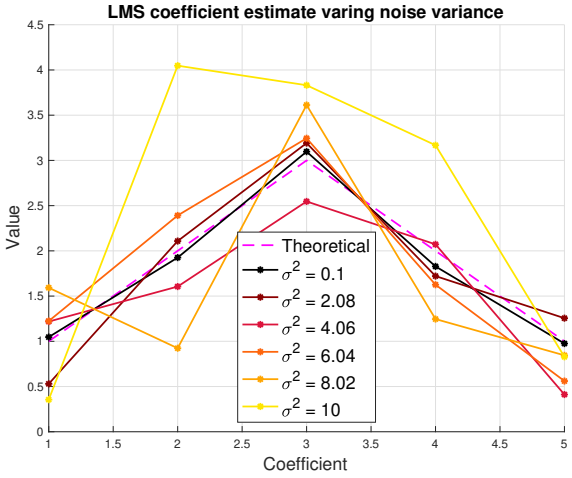

Figure 62: LMS coefficient estimation varying noise power

4.2.2 Part 2: Varying adaptation gain μ

In this section, the effect of varying the adaptation gain μ is studied: the results are clear from Figure 63.

With low adaptation gain, such as $\mu = 0.002$, the coefficients converge slowly to the actual values (after around 2500 samples), but have very little oscillation in steady state. This is confirmed by looking at the squared estimate error plot. The error clearly decreases with an exponential trend until the steady state is reached. After many samples, the error is still present but is very low: the peaks never exceed 0.1 in amplitude.

A gain $\mu = 0.1$ gives faster convergence: the final is reached after around 60 samples. However, this is at the expense of relatively high steady state oscillation around the actual values. Again, the behaviour is confirmed by the error plot: the error decreases fast but the peaks after many samples are high: the peaks reach 0.3 in amplitude, which is more than 3 times than for $\mu = 0.02$.

A higher gain, such as $\mu = 0.5$ in this case, leads to unstable overshooting estimate. This happens because the error is corrected "too quickly" leading to oscillating exponential behaviour of the estimate. Looking at the squared estimate error, we notice that it is only plotted until the 3600$^{\text{th}}$ sample, before its exponential increase is not tolerated by the software, again confirming the previous statement. This behaviour occurs for $\mu >\sim 0.25$.

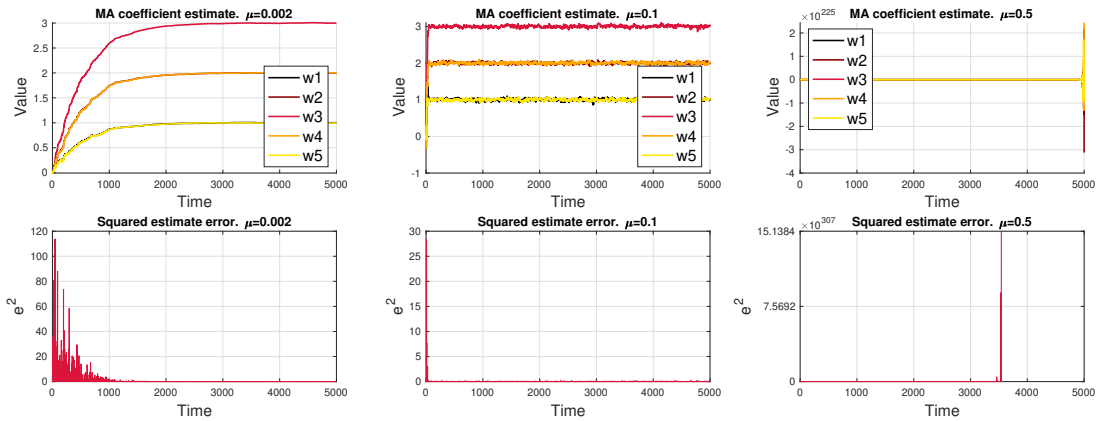

Figure 63: Time evolution of MA coefficient estimate and squared estimate error, varying the adaptation gain

4.2.3 Part 3: Computational complexity

- building $\hat{y}[n]$ involves multiplying $\mathbf{w}^T(n)\mathbf{x}(n)$, which have dimensions $[1 \times (N_w + 1)] * [(N_w + 1) \times 1]$. This has complexity $\mathcal{O}(N_w^2)$;
- building $e[n]$ and $w(n + 1)$, which are $\mathcal{O}(1)$;
- the steps are repeated N times, once for each data point

The resulting complexity is $\mathcal{O}(N * N_w^2)$.

4.3 Gear shifting

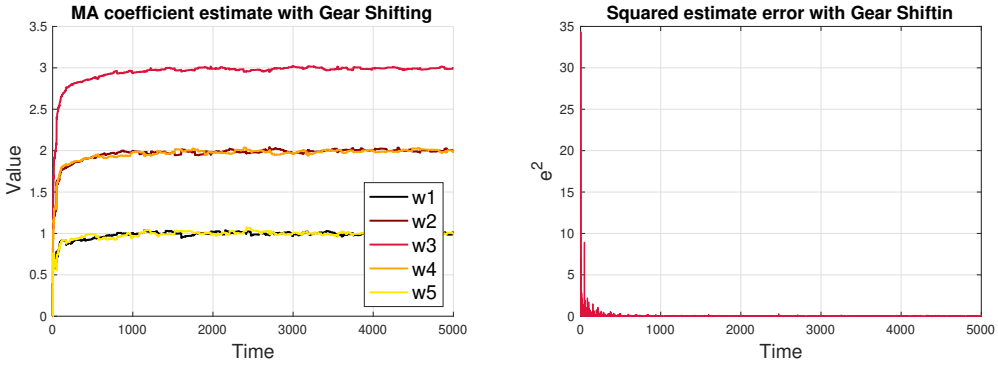

Figure 64: Time evolution of MA coefficient estimate and squared estimate error with gear shifting

Section 4.2.2 analysed the relation between the value of μ and the time evolution of the MA coefficient approximation. It was noticed that low adaptation gain was giving a slow non-oscillatory response. As the values become higher, the response becomes increasingly fast and oscillatory, until this results in overshooting behaviour for μ above ~ 0.25.

Clearly, one would ideally want to achieve a fast and non-oscillatory behaviour. This can be done by changing the adaptation gain as the estimation error decreases.

Instead of using the absolute error value for adaptation gain selection, the percentage error is calculated as $\%Error(n) = 100 * \frac{e(n)}{z(n)}$. The value of the error relative to the actual output is more relevant than its absolute value because it gives a better indication on how wrong the estimation is.

Figure 65 show the exact relation between the percentage error and the chosen μ. When the estimation error is high (above 90%), μ is set to 0.1, which is high but ensures to avoid overshooting behaviour and to meet the requirement of maintaining the maximum overshoot for each coefficient to 20% of its true value. As the error decreases, μ is progressively reduced until it reaches 0.02 when the relative error is below 10%.

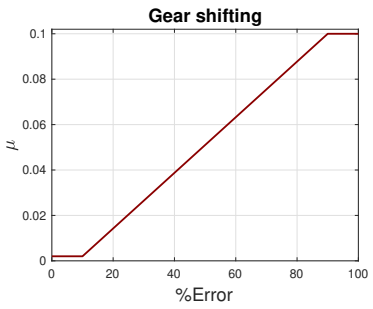

Figure 65: Selection of adaptation gain μ based on $\%Error$

Figure 64 shows the time evolution of the MA coefficient estimate and squared estimate error with gear shifting, tested on the same data as that in Section 4.1.2. It can be noticed that this method indeed finds a balance between speed of the estimate and steady state oscillation. The error is barely 0 after 500 samples and the steady state oscillation is negligible. Moreover, no overshoot is observed.

4.4 Identification of AR process

An AR process is synthesised from white noise using an autoregressive model. The same method used in the sections above can be used for the analysis stage. Adaptation is used to identify the AR coefficients of a process to recreate the original signal.

Filtering WGN through a filter with transfer function $F(z) = \frac{1}{1+0.9z^{-2}+0.2z^{-2}}$ is equivalent to building an AR(2) process with parameters $\mathbf{a} = [-0.9 \ \ -0.2]$. In fact, Figure 66 shows that the estimates for a_1 and a_2 tend towards -0.9 and -0.2 respectively.

For varying adaptation gain μ, the same considerations from Section 4.2.2 apply. Low adaptation gain, such as $\mu = 0.001$ leads to a very stable non-oscillatory estimate. However, the correct value for a_1 and a_2 is only reached after around 5000 samples. Increasing μ to 0.01 and then 0.05 leads to a faster estimation that is however increasingly oscillatory. This happens because higher μ leads to a bigger correction term that acts in the opposite direction of the error to counteract it. If the correction however is greater than the error itself, oscillatory behaviour is experienced. This can be clearly noticed in subplots 2 and 3. Increasing the adaptation gain even further (such as $\mu = 0.2$ in subplot 4) leads to overshooting behaviour and a stable estimate for $\mathbf{a}$ is not reached.

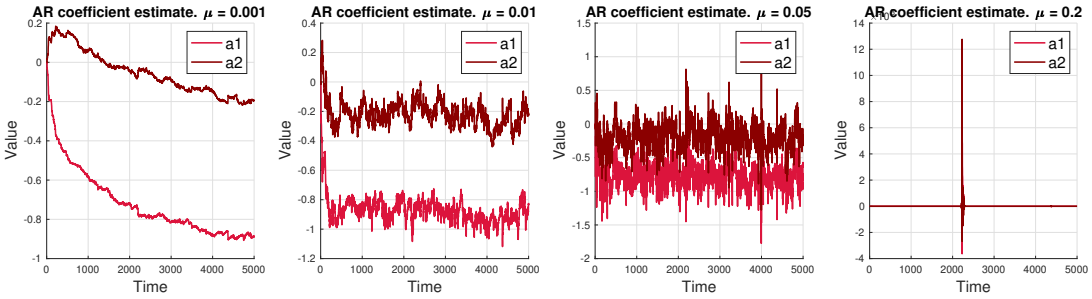

Figure 66: Time evolution of AR coefficient estimate, varying the adaptation gain

4.5 Speech recognition

The same method explained in Section 4.4 is used for speech recognition. A sound is used as input; the filter finds the coefficients $\mathbf{a}$ that minimise the squared estimate error; when the input is a delayed version of the sound, the estimator will output the sound itself.

The method is tested on 1000-samples recordings of sounds *E*, *A*, *S*, *T*, and *X*, sampled at 44100Hz and then at 16000Hz.

4.5.1 Part 1,2: Sampling at 44100 Hz

Order of the predictor Finding the order of the predictor means finding the order of AR process that successfully approximates the signal. This can be done in two ways:

1. Analytical approach: using model order selection methods, such as MDL, AIC and AIC_c from Section 2.3.4
2. Heuristic approach: by trying out a high model order, and selecting as model order p the last value for which $\mathbf{a}(p)$ is well above 0.

The model order selection was done using method one, and the selected model order for each sound is given in Table 6.

Sound	Order MDL	Order AIC	Order AICc	Selected Order
e	6	6	3	6
a	2	2	2	2
s	6	6	4	6
t	10	20	4	10
x	10	19	3	10

Table 6: Model order selection for each sound at $fs = 44100Hz$

Adaptation gain The adaptation gain was chosen experimentally by trying out values in the interval $\mu \in [0.001, 1]$. $\mu = 0.1$ was chosen as for *E*, *A* and *S* as the highest value that was ensuring no overshoot was occurring.

A low adaptation gain is not suitable for non-stationary signals as the AR coefficients need to be constantly updated to match the varying statistics of the signal. The response of the estimate needs to be fast and therefore lower μ is not a suitable choice. For *T* and *X* $\mu = 0.05$ is selected for the same reason as above: it is the highest adaptation gain that ensures avoiding overshooting behaviour.

It can be noticed from Figure 67 that $\mu = 0.1$ is indeed an appropriate choice in these cases, as the estimated output signal converges to the true value after around 20 samples and it matches well the real signal for the entire time period. The signals are plotted for the sounds *E*, *A* and *S*, but the same considerations apply for *T* and *X*.

Gear shifting Gear shifting was shown not to be useful for this application. It can be noticed from the plots of the AR coefficients in Figure 67 that the audio signals are non-stationary, as their statistics vary with time. Gear shifting is a technique designed to obtain a stable estimate of the coefficients when these are known to be invariant. In this case, gear shifting would cause slowing down the responsiveness of the algorithm to changes in the statistics of the signal, degrading the ability to match the output.

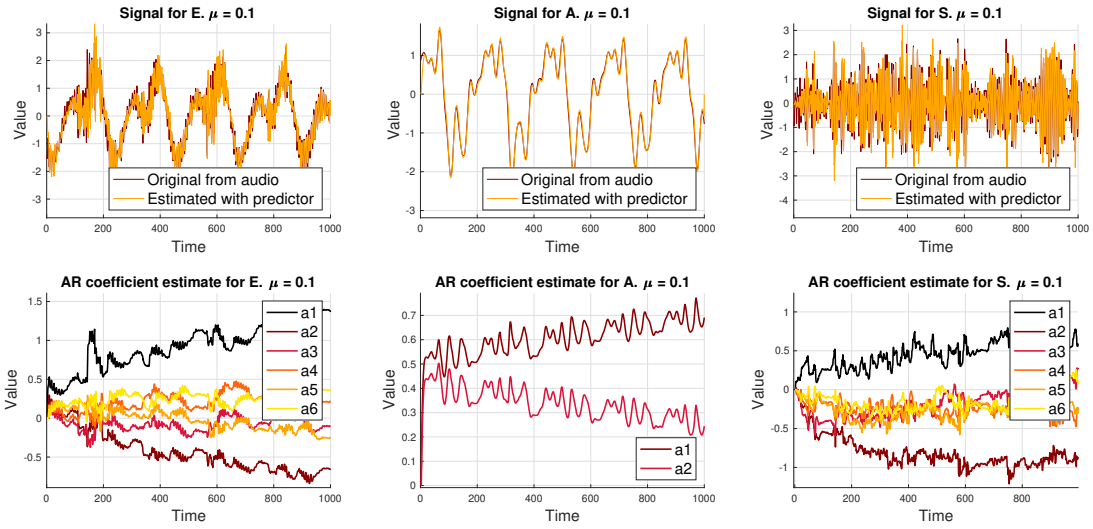

Figure 67: Time evolution of signal and AR coefficient estimate for different audio sounds

4.5.2 Part 3: Sampling at 16000 Hz

Order of the predictor The model order selected for each sound at sampling of 16000Hz is shown in Table 7. It can be noticed that the model order requirement for the lower sampling frequency is slightly lower.

Sound	Order MDL	Order AIC	Order AICc	Selected Order
e	6	11	6	6
a	3	3	3	3
s	8	12	6	8
t	10	20	4	10
x	8	15	8	8

Table 7: Model order selection for each sound at $fs = 16000Hz$

Adaptation gain $\mu = 0.1$ is used for *A* and *E*, $\mu = 0.07$ for *S* and $\mu = 0.05$ for *T* and *X*. It can be seen that these are all the same as those obtained in Section 4.5.1 for a sampling frequency of 44100Hz. The only difference is in *S*, where μ was adjusted from 0.1 to 0.07 to avoid overshoot.

4.5.3 Part 2: Performance of the predictor

The performance of the predictor can be assessed by the prediction gain, whose formula is given in Equation 50, where σ_x^2 is the input signal variance and σ_e^2 is the error variance.

$$R_p = 10 \log_{10} \left(\frac{\sigma_x^2}{\sigma_e^2} \right) \tag{50}$$

This is used to evaluate the performance of all the predictors that were generated, both with the sampling at 44100Hz and at 16000Hz, as shown in Table 8.

At lower sampling frequency, it was reasonable to expect a degradation of the performance in the prediction due to the fact that the samples are temporally spaced further apart. Therefore the previous samples are less informative towards the prediction. However, it can be noticed that with a much lower sampling frequency the performance is worse but the deterioration is very small if the model order of the prediction and the adaptation gain are varied appropriately, as explained in Section 4.5.2.

Sound	Rp at 44100Hz	Rp at 16000Hz
e	11.82 dB	10.05 dB
a	20.26 dB	14.38 dB
s	10.18 dB	7.75 dB
t	5.60 dB	5.60 dB
x	5.65 dB	5.15 dB

Table 8: Prediction gain for each sound at different sampling frequencies

4.6 Sign algorithms

The LMS algorithms exist in simplified versions where only the sign of the error or the input is used. In these algorithms, the LMS equation for updating the estimated output (given in Equation 49), is updated to one of the following Equations (51 to 53), depending on the chosen algorithm.

Equation 51 refers to the signed algorithm that uses signed error, Equation 52 uses the signed regressor, while Equation 53 uses the sign of both the error and the regressor.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu * \text{sign}(e[n]) * x(n) \tag{51}$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu * e[n] * \text{sign}(x(n)) \tag{52}$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu * \text{sign}(e[n]) * \text{sign}(x(n)) \tag{53}$$

These methods are applied to the AR coefficient estimation from Section 4.4. It can be noticed that all three algorithms provide a very good alternative to the standard LMS algorithm, both in the transient and in the long run estimate.

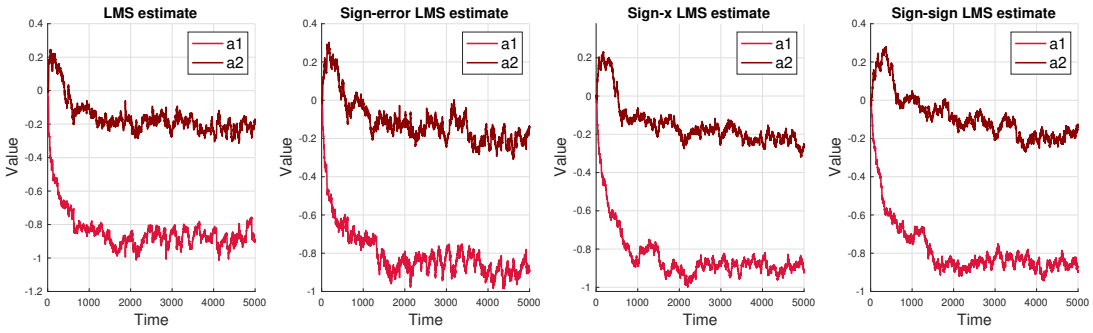

Figure 68: LMS sign algorithms for AR coefficient estimation, $\mu = 0.05$

The same algorithms are tested on the audio recording of the sound *A* from Section 4.5. Figure 69 shows the comparison between the standard LMS algorithm (subplot 1) and the orhter three methods. All the plots show the evolution of the coefficients over 1000 samples and for $\mu = 0.1$. It can be noticed that, while the *Signed-x algorithm* gives an estimate that matches the LMS well, the other two methods do not follow the evolution of the coefficients well.

This however was shown to be only a problem of parameter selection. In fact, by choosing $\mu = 0.01$ for the *Sign-error* and *Sign-sign algorithm* the coefficients match the standard LMS evolution well, even over a longer time period (here, 5000 samples). The behaviour is shown in Figure 70.

The difference in behaviour compared to the AR coefficients estimation given above is due to the fact that the signal analysed here is non-ergodic.

To sum up, all three algorithms provide a good alternative to the LMS method where the computational effort is reduced. The *Sign-error* and *Sign-x* algorithms barely show any difference with respect to the original method. On the other hand, the *Sign-sign* algorithm only provides correction is the right direction, always of the same magnitude μ. Therefore, the resulting curve is less smooth, but still valid as an estimate with careful μ selection.

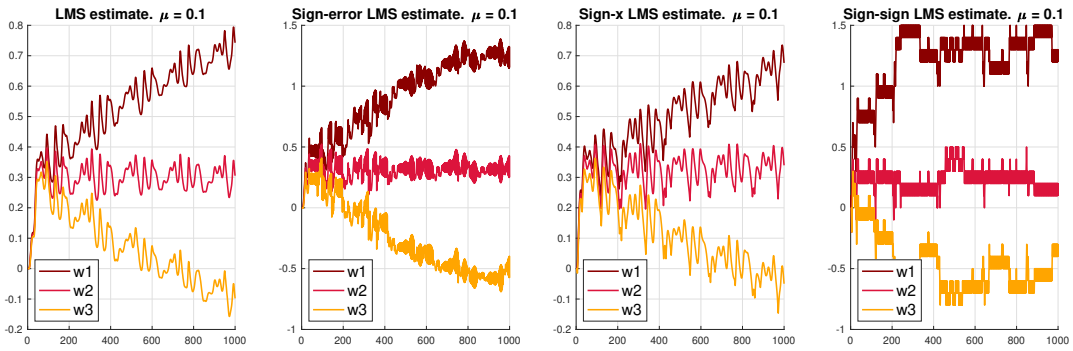

Figure 69: LMS sign algorithms for coefficient estimation in speech recognition, plotted for 1000 samples

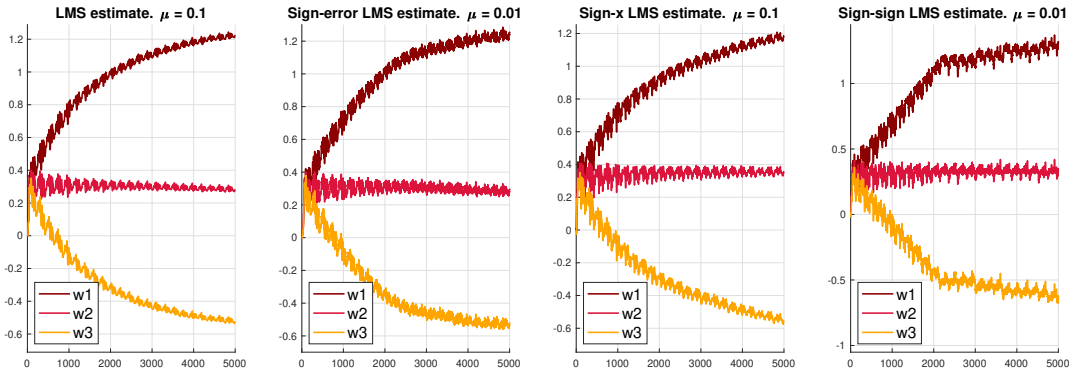

Figure 70: LMS sign algorithms for coefficient estimation in speech recognition, with adapted μ and plotted for 5000 samples

5 MLE for the frequency of a signal

The pdf of a signal $\mathbf{x} = \begin{bmatrix} x[0] & x[1] & \dots & x[N-1] \end{bmatrix}^T$, parametrised by $\theta = \begin{bmatrix} A & f_0 & \phi \end{bmatrix}^T$ is given in Equation 54, where $A > 0$ and $0 < f_0 < \frac{1}{2}$.

$$p(\mathbf{x};\boldsymbol{\theta}) = \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\Big\{ -\frac{1}{2\sigma^2} \sum_{n=0}^{N-1} (x[n] - A\cos(\pi f_0 n + \phi))^2 \Big\} \tag{54}$$

5.1 Part 1: Cost function

From the given pdf, the maximim-likelihood estimate (MLE), $\hat{f}_0$of the frequency is found by minimising $J(\boldsymbol{\theta})$, given in Equation 55

$$J(\boldsymbol{\theta}) = \sum_{n=0}^{N-1} \big(x[n] - A\cos(2\pi f_0 n + \phi)\big)^2 \tag{55}$$

This can be rewritten by expanding the cosine and then substituting $\alpha_1 = A\cos(\phi)$ and $\alpha_2 = -A\sin(\phi)$:

$$J(\boldsymbol{\theta}) = \sum_{n=0}^{N-1} \big(x[n] - A\cos(\phi)cos(2\pi f_0 n) + A\sin(\phi)\sin(2\pi f_0 n)\big)^2 =$$

$$= \sum_{n=0}^{N-1} \big(x[n] - \alpha_1\cos(2\pi f_0 n) - \alpha_2\sin(2\pi f_0 n)\big)^2 \tag{56}$$

Using $\mathbf{c}$ and $\mathbf{s}$ as defined in Equation 57, $J(\boldsymbol{\theta})$ is rewritten as $J'(\boldsymbol{\theta}')$ as in Equation 59, where $\boldsymbol{\theta}' = \begin{bmatrix}\alpha_1 & \alpha_2 & f_0\end{bmatrix}^T$, and $\mathbf{H}$ and $\boldsymbol{\alpha}$ are defined in Equation 58.

$$\mathbf{c} = \begin{bmatrix} 1 \\ \cos(2\pi f_0) \\ \cos(4\pi f_0) \\ \vdots \\ \cos(2\pi f_0(N-1)) \end{bmatrix} \qquad \mathbf{s} = \begin{bmatrix} 0 \\ \sin(2\pi f_0) \\ \sin(4\pi f_0) \\ \vdots \\ \sin(2\pi f_0(N-1)) \end{bmatrix} \tag{57}$$

$$\mathbf{H} = [\mathbf{c}\ \mathbf{s}] = \begin{bmatrix} 1 & 0 \\ \cos(2\pi f_0) & \sin(2\pi f_0) \\ \cos(4\pi f_0) & \sin(4\pi f_0) \\ \vdots & \vdots \\ \cos(2\pi f_0(N-1)) & \sin(2\pi f_0(N-1)) \end{bmatrix} \qquad \boldsymbol{\alpha} = \begin{bmatrix}\alpha_1 \\ \alpha_2\end{bmatrix} \tag{58}$$

$$J'(\boldsymbol{\theta}') = (\mathbf{x} - \alpha_1\mathbf{c} - \alpha_2\mathbf{s})^T(\mathbf{x} - \alpha_1\mathbf{c} - \alpha_2\mathbf{s}) = (\mathbf{x} - \mathbf{H}\boldsymbol{\alpha})^T(\mathbf{x} - \mathbf{H}\boldsymbol{\alpha}) \tag{59}$$

5.2 Part 2: Minimising solution

The MLE of $\boldsymbol{\alpha}$ is found by minimising $J'(\boldsymbol{\alpha}, f_0)$. This is done by differentiating, setting the derivative to 0 and solving for $\boldsymbol{\alpha}$.

Start by finding the partial derivative with respect to $\boldsymbol{\alpha}$:

$$\frac{\partial J'(\boldsymbol{\alpha}, f_0)}{\partial \boldsymbol{\alpha}} = \frac{(\mathbf{x} - \mathbf{H}\boldsymbol{\alpha})^T(\mathbf{x} - \mathbf{H}\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \frac{\partial}{\partial \boldsymbol{\alpha}}\Big\{ \big(\mathbf{x}^T - \boldsymbol{\alpha}^T\mathbf{H}^T\big)\big(\mathbf{x} - \mathbf{H}\boldsymbol{\alpha}\big) \Big\} = \frac{\partial}{\partial \boldsymbol{\alpha}}\Big\{ \mathbf{x}^T\mathbf{x} - \boldsymbol{\alpha}^T\mathbf{H}^T\mathbf{x} - \mathbf{x}^T\mathbf{H}\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\mathbf{H}^T\mathbf{H}\boldsymbol{\alpha} \Big\} =$$

$$= \frac{\partial}{\partial \boldsymbol{\alpha}}\Big\{ -2\boldsymbol{\alpha}^T\mathbf{H}^T\mathbf{x} + \boldsymbol{\alpha}^T\mathbf{H}^T\mathbf{H}\boldsymbol{\alpha} \Big\} = -2\mathbf{H}^T\mathbf{x} + 2\mathbf{H}^T\mathbf{H}\boldsymbol{\alpha}$$

Then, setting it to 0:

$$-2\mathbf{H}^T\mathbf{x} + 2\mathbf{H}^T\mathbf{H}\boldsymbol{\alpha} = 0$$

Solving for $\boldsymbol{\alpha}$ leads to Equation 60

$$\hat{\boldsymbol{\alpha}} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x} \tag{60}$$

Substituting back Equation 60 into Equation 59 leads to:

$$J'(\boldsymbol{\alpha}, f_0) = (\mathbf{x} - \mathbf{H}\hat{\boldsymbol{\alpha}})^T(\mathbf{x} - \mathbf{H}\hat{\boldsymbol{\alpha}}) = (\mathbf{x} - \mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x})^T(\mathbf{x} - \mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x}) = \mathbf{x}^T(\mathbf{I} - \mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T)\mathbf{x}$$

Minimising $J'(\boldsymbol{\alpha}, f_0)$ is equivalent to maximising $-J'(\boldsymbol{\alpha}, f_0)$.

$$-J'(\boldsymbol{\alpha}, f_0) = -\mathbf{x}^T(\mathbf{I} - \mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T)\mathbf{x} = -\mathbf{x}^T\mathbf{x} + \mathbf{x}^T\mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x}$$

Since the term $-J'(\boldsymbol{\alpha}, f_0)$ is maximised $-\mathbf{x}^T\mathbf{x}$ is effectively a constant, maximising the expression is equivalent to maximising the second term, i.e. $\mathbf{x}^T\mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x}$.

It was therefore shown that minimising $J'(\boldsymbol{\alpha}, f_0)$ is equivalent to maximising the term given in Equation 61.

$$\mathbf{x}^T\mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x} \tag{61}$$

5.3 Parts 3,4: Approximated solution

By substituting $\mathbf{H} = [\mathbf{c}\ \mathbf{s}]$, the term in Equation 61 is shown to be equivalent to:

$$\mathbf{x}^T\mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x} = (\mathbf{H}^T\mathbf{x})^T(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x} =$$

$$= \Big(\begin{bmatrix}\mathbf{c}^T \\ \mathbf{s}^T\end{bmatrix}\mathbf{x} \Big)^T \Big(\begin{bmatrix}\mathbf{c}^T \\ \mathbf{s}^T\end{bmatrix} \begin{bmatrix}\mathbf{c} & \mathbf{s}\end{bmatrix} \Big)^{-1} \begin{bmatrix}\mathbf{c}^T \\ \mathbf{s}^T\end{bmatrix}\mathbf{x} = \begin{bmatrix}\mathbf{c}^T\mathbf{x} \\ \mathbf{s}^T\mathbf{x}\end{bmatrix}^T \begin{bmatrix}\mathbf{c}^T\mathbf{c} & \mathbf{c}^T\mathbf{s} \\ \mathbf{s}^T\mathbf{c} & \mathbf{s}^T\mathbf{s}\end{bmatrix}^{-1} \begin{bmatrix}\mathbf{c}^T\mathbf{x} \\ \mathbf{s}^T\mathbf{x}\end{bmatrix} \tag{62}$$

The terms $\mathbf{c}^T\mathbf{c}$, $\mathbf{c}^T\mathbf{s}$, $\mathbf{s}^T\mathbf{c}$ and $\mathbf{s}^T\mathbf{s}$ can be approximated as follows. To ensure that these relations hold, require f_0 not to be close to 0 or $\frac{1}{2}$.

$$\mathbf{c}^T\mathbf{c} = \sum_{n=0}^{N-1} \cos^2(2\pi f_0 n) \approx \frac{N}{2}$$

$$\mathbf{c}^T\mathbf{s} = \mathbf{s}^T\mathbf{c} = \sum_{n=0}^{N-1} \cos(2\pi f_0 n)\sin(2\pi f_0 n) = \frac{1}{2}\sum_{n=0}^{N-1} \sin(4\pi f_0 n) \approx 0$$

$$\mathbf{s}^T\mathbf{s} = \sum_{n=0}^{N-1} \sin^2(2\pi f_0 n) \approx \frac{N}{2}$$

Then, Equation 62 can be rewritten as:

$$\begin{bmatrix}\mathbf{c}^T\mathbf{x}\\ \mathbf{s}^T\mathbf{x}\end{bmatrix}^T \begin{bmatrix}\frac{N}{2} & 0\\ 0 & \frac{N}{2}\end{bmatrix}^{-1} \begin{bmatrix}\mathbf{c}^T\mathbf{x}\\ \mathbf{s}^T\mathbf{x}\end{bmatrix} = \begin{bmatrix}\mathbf{x}^T\mathbf{c} & \mathbf{x}^T\mathbf{s}\end{bmatrix} \begin{bmatrix}\frac{2}{N} & 0\\ 0 & \frac{2}{N}\end{bmatrix} \begin{bmatrix}\mathbf{c}^T\mathbf{x}\\ \mathbf{s}^T\mathbf{x}\end{bmatrix} = \begin{bmatrix}\frac{2}{N}\mathbf{x}^T\mathbf{c} & \frac{2}{N}\mathbf{x}^T\mathbf{s}\end{bmatrix} \begin{bmatrix}\mathbf{c}^T\mathbf{x}\\ \mathbf{s}^T\mathbf{x}\end{bmatrix} =$$

$$= \frac{2}{N}\left\{\mathbf{x}^T\mathbf{c}\mathbf{c}^T\mathbf{x} + \mathbf{x}^T\mathbf{s}\mathbf{s}^T\mathbf{x}\right\} = \frac{2}{N}\left\{(\mathbf{c}^T\mathbf{x})^2 + (\mathbf{s}^T\mathbf{x})^2\right\} =$$

$$= \frac{2}{N}\left\{\Big(\sum_{n=0}^{N-1} x(n)\cos(2\pi f_0 n)\Big)^2 + \Big(\sum_{n=0}^{N-1} x(n)\sin(2\pi f_0 n)\Big)^2\right\} = \frac{2}{N}\left|\sum_{n=0}^{N-1} x(n)\exp(-j2\pi f_0 n)\right|^2$$

5.4 Part 5: Experimental results

This method is tested on on a the noiseless data given in Equation 63.

$$x[n] = \cos(2\pi f_0 n) \quad n = 0, 1, ..., N-1 \tag{63}$$

Figure 71 is a plot of the periodogram of the signal for different f_0. It can be clearly noticed that the maxima occur at f_0, which is the frequency more present in the signal.

As from the MLE approach, the estimate of f_0 is the peak of the periodogram. Figure 72 shows that this produces a correct estimate of the oscillation frequency of the signal.

The inaccuracy that are noticed on the plot are due to the fact that the signal is finite in length. For an input sequence of N samples, the output of the periodogram will have N values, of which only $\frac{N}{2}$ are unique, due to symmetry (if the input sequence is a real signal). Then, with $N = 10$, the estimated frequency $\hat{f}_0$ can only have one of the 5 unique values that correspond to those present in the periodogram. In fact, it is noticed that the estimate is entirely correct if the number of samples is increased to $N = 1000$.

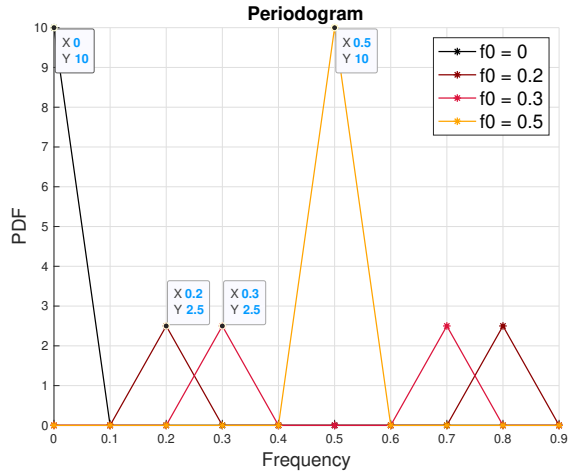

Figure 71: Periodogram of the input signal varying f_0

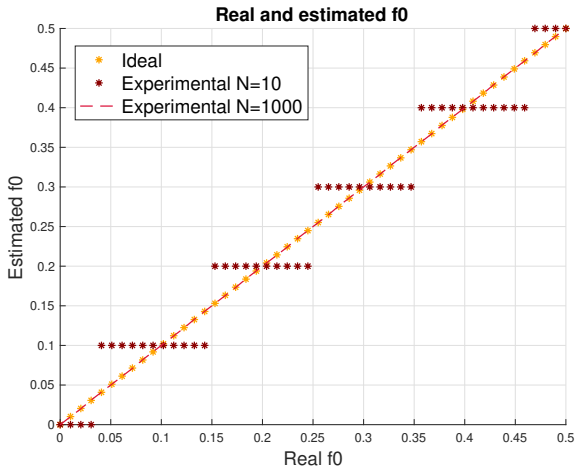

Figure 72: Real and estimated f_0 varying signal length N