Imperial College London

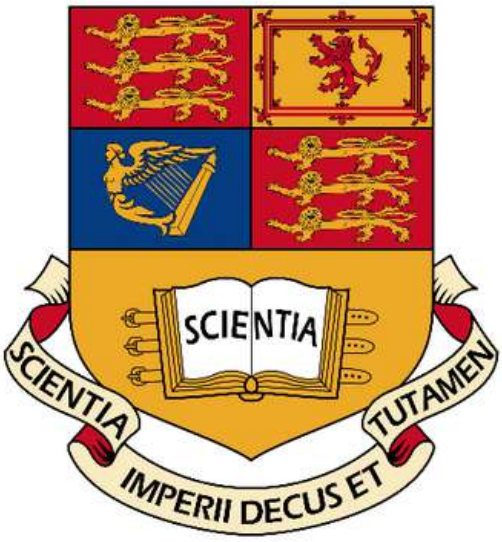
EE-08 Advanced Signal Processing

Advanced Signal Processing Coursework

Author:
Charalambos Hadjipanayi

CID:
01077219

Supervisor: Prof. Danilo P.Mandic
March 22, 2019



Contents

1 Random Signals and Stochastic Processes

1.1 Statistical estimation

1.1.1 - Uniformly distributed stochastic process

A 1000-sample vector is generated in MATLAB using the command `rand`, where each sample x[n] is a realisation of a Uniform random variable X$\sim \mathcal{U}(0,1)$ at time instant n. It can be observed that **x** exhibits a degree of uniformity due to time-invariant statistical properties.

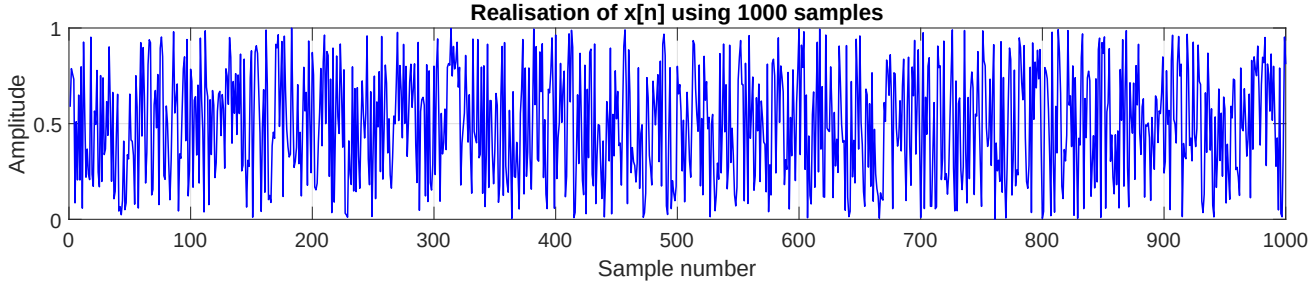


Figure 1.1: Plot of a 1000-sample realisation of stationary stochastic process $X_n \sim \mathcal{U}(0,1), \forall n$.

Since X$\sim \mathcal{U}(0,1)$, then its probability density function is $f_X(x) = \begin{cases} 1 & , \; for\ 0 \le x \le 1 \\ 0 & , \; otherwise \end{cases}$. The expected value, or theoretical mean, of X is defined as:

$$m = E\{X\} = \int_{-\infty}^{+\infty} x f_X(x)\, dx = \int_0^1 x(1) dx = \left[\frac{x^2}{2}\right]_0^1 = \frac{1}{2} \tag{1}$$

Using MATLAB, the sample mean of 1000 samples was computed using function `mean` and was found to be 0.5065. Sample mean ($\hat{m}$) can be used as an estimator for the theoretical mean of the random process, since it is an unbiased estimator, i.e. on average it produces the true value for mean.

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

Thus, $bias(\hat{m}) = E\{\hat{m}\} - m = 0$, meaning that estimator is unbiased and yields the true value on average. If an ensemble of large number of realisations is used, then the ensemble average will converge to the theoretical mean. Additionally, the standard error ($\sigma_{\hat{m}}$) of the estimator can be obtained from:

$$\begin{aligned} Var(\hat{m}) &= Var\left(\frac{1}{N}\sum_{n=1}^{N} x[n]\right) = \frac{1}{N^2} Var\left(\sum_{n=1}^{N} x[n]\right), \; \textit{and since all } x[n] \textit{ values are statistically independent}, \\ &= \frac{1}{N^2}\sum_{n=1}^{N} Var(x[n]) = \frac{\sigma^2}{N} = \frac{1}{12000}, \quad \textit{therefore } \sigma_{\hat{m}} = \sqrt{\frac{1}{12000}} \approx 0.00913 = 0.913\%. \end{aligned} \tag{3}$$

Furthermore, the absolute relative error in estimation is 0.783%, which is due to the fact that only 1000 samples were used. It is expected that as sample size increases, relative error decreases. Also, standard error of sample mean is $\propto \frac{1}{\sqrt{N}}$, so decreases with sample size. Bias, standard error and relative error are indicators of accuracy of estimator. The theoretical variance (σ^2) of the process is given by:

$$\sigma^2 = E\{(x[n] - E\{x[n]\})^2\} = \int_{-\infty}^{+\infty} (x-m) f_X(x)\, dx = \int_0^1 (x^2 - 2mx + m^2) dx = \left[\frac{x^3}{3} - x^2 m + m^2 x\right]_0^1 = \frac{1}{12} \tag{4}$$

Theoretical standard deviation, $\sigma = \sqrt{\sigma^2} = \frac{\sqrt{3}}{6} \approx 0.2887$.
Sample estimate of standard deviation from data **x** was found to be 0.2923. As a result, the absolute relative error in the estimation is 0.769%. According to Rao C.R. (1973), *Linear statistical inference and its applications*, 2^{nd} edition, New York, John Wiley & Sons, standard error of sample variance (se($\hat{\sigma}^2$)) is given by: $se(\hat{\sigma}^2) = \sqrt{\frac{1}{N}(\mu_4 - \frac{N-3}{N-1}\sigma^4)}$, where μ_4 is the fourth central moment of the distribution given by: $\mu_4 = E\{(X-\mu)^4\}$. For $X_n \sim \mathcal{U}(0,1)$, $\mu_4 = \frac{1}{80}$. In addition, according to Rao (1973), $std(g(\hat{\theta})) \approx \left|g'(\hat{\theta})\right| \times std(\hat{\theta})$, where $\hat{\theta}$ in our case is sample variance. For large number of samples N, this means $se(\hat{\sigma})) \approx \frac{1}{2\sigma} std(\hat{\sigma}^2)$. Therefore for our case, standard error for sample std is $\approx 0.00409 = 0.409\%$. It can be shown that sample variance is an unbiased estimator for theoretical variance:

$$\begin{aligned} E\{\hat{\sigma}^2\} &= E\{\frac{1}{N-1}\sum_{n=1}^{N}(x[n] - \hat{m})^2)\} = E\Big\{\frac{1}{N-1}\sum_{n=1}^{N}(x^2[n] - 2x[n]\hat{m} + \hat{m}^2\Big\} \\ &= \frac{1}{N-1}\sum_{n=1}^{N}(E\{x^2[n]\} - 2E\{x[n]\}E\{\hat{m}\} + E\{\hat{m}^2\}) \end{aligned} \tag{5}$$

Since we know that $E\{z^2\} = Var(z) + (E\{z\})^2$ and that $Var(\hat{m}) = \frac{\sigma^2}{N}$,

$$E\{\hat{\sigma}^2\} = \frac{1}{N-1}\sum_{n=1}^{N}(\sigma^2 + m^2 - 2m^2 + \frac{\sigma^2}{N} + m^2) = \frac{N}{N-1}\frac{N\sigma^2 - \sigma^2}{N} = \sigma^2 \frac{N-1}{N-1} = \sigma^2 \tag{6}$$

Therefore, it is an unbiased estimator. However, sample standard deviation is not an unbiased estimator since square root is not a linear operator which means $E\{\sqrt{\hat{\sigma}^2}\} \neq \sqrt{E\{\hat{\sigma}^2\}}$. Due to Bessel's correction, some bias is removed (but not all) in the estimation of population standard deviation. If an ensemble of large number of realisations is used, then the ensemble std will never converge to the theoretical std. It should be noted that as sample size increases for each realisation, bias of this estimator decreases, so it is an asymptotically unbiased estimator (Asymptotic consistency). This is illustrated in Figure 1.9.
An ensemble of ten 1000-sample realisations of X is generated, from which the sample means and standard deviations are obtained and then plotted below. The theoretical values are shown in the plot as red lines.

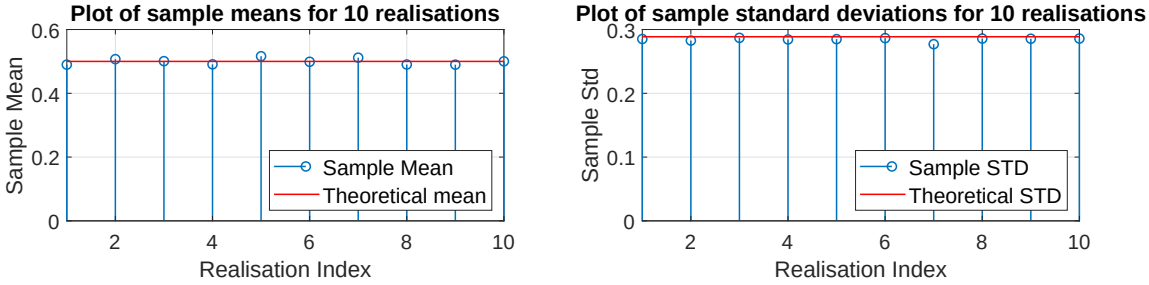


Figure 1.2: Plot of samples means and sample std of an ensemble of ten 1000-sample realisations of stationary stochastic process $X_n \sim \mathcal{U}(0,1), \forall n$.

The bias of the sample mean and standard deviation is defined by $B = E\{X\} - \hat{m}$. Using this equation the average bias for sample mean was found to be 0.00034 and for sample std 0.0029. This proves quantitatively that these estimates cluster about their theoretical values.
The pdf of X is estimated using the `histogram` command in MATLAB, which generated a histogram of the process normalised by the number of samples considered (Figures 1.3 and 1.4. The theoretical pdf was plotted in red colour.

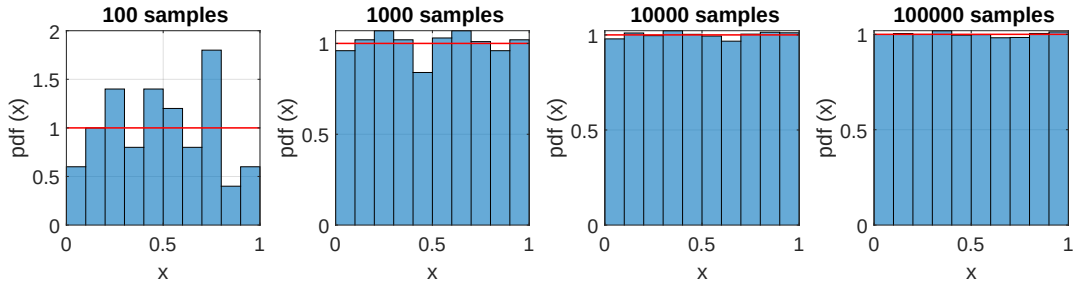


Figure 1.3: Effect of number of generated samples on the histogram pdf approximation for distribution of X. Estimated pdf is also compared against theoretical pdf of $X_n \sim \mathcal{U}(0,1)$ (Red Line). Number of bins used is 10.

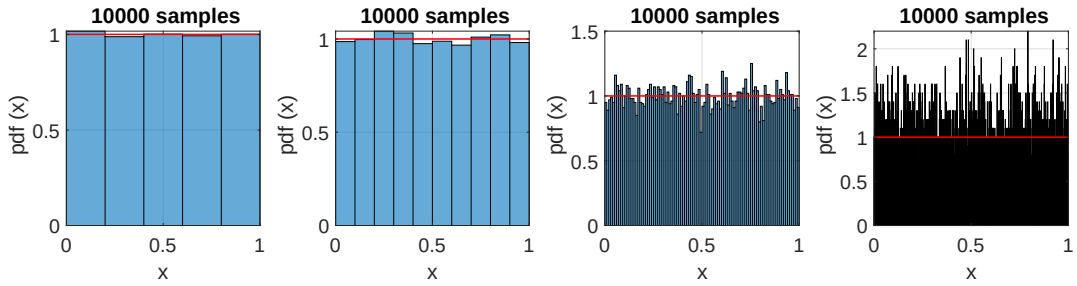


Figure 1.4: Effect of number of bins on the histogram pdf approximation for distribution of X. Estimated pdf is also compared against theoretical pdf of $X_n \sim \mathcal{U}(0,1)$ (Red Line). Number of samples used is 10000.

Figure 1.3 shows that the estimate converges to the theoretical pdf as the number of generated samples increases. If the number of bins is held steady, then as sample size increases, the standard error, i.e. the amount of chance fluctuation we can expect decreases, or gets more precise. Therefore the statistical precision of pdf estimation increases.
Figure 1.4 shows that as the number of bins considered increases, the estimated pdf starts to deviate from the theoretical pdf. It can be shown that the bias of the estimation is $\alpha\ \frac{1}{M}$,where M is the number of bins used. In addition, the variance of the estimation is $\alpha\ M$ .This means that as the number of bins increase, the lower the bias of estimation, thus the higher the resolution (accuracy) but simultaneously the higher the variance of the estimation. This effect is called undersmoothing. The opposite effect, i.e. when M decreases, is called oversmoothing.

1.1.2 - Normally (Gaussian) distributed stochastic process

The same procedure was repeated for $X \sim \mathcal{N}(0,1)$, where 1000-sample vector $\mathbf{x}$ was generated using MATLAB function `randn`, for zero-mean, unit standard deviation random variables. Sample mean and sample standard deviation were computed from data to be 0.0083 and 0.9939 respectively (their theoretical values are 0 and 1). Absolute error in the estimation of mean is 0.0165 and absolute relative error in estimation of std is 2.976%. The standard error for sample mean is 0.01. The central fourth moment (μ_4) for normal distribution is $3\sigma^4$. Using the same justification as for uniform distribution, standard error for sample std is $\approx 0.0224 = 2.24\%$.

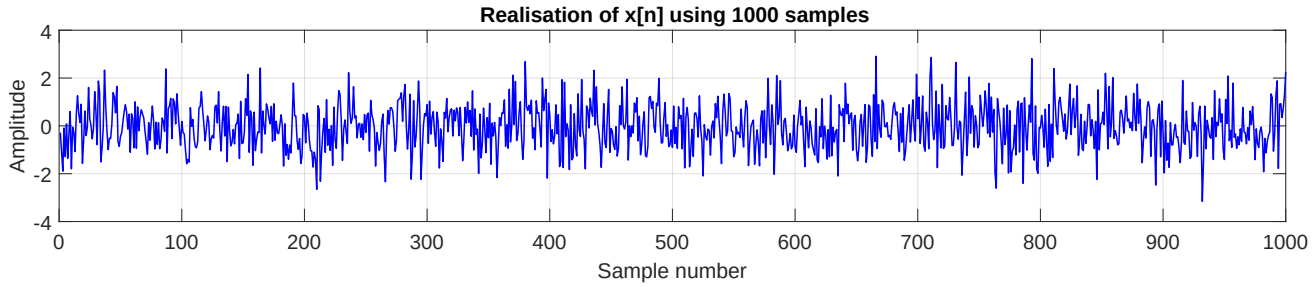


Figure 1.5: Plot of a 1000-sample realisation of stationary stochastic process $X_n \sim \mathcal{N}(0,1), \forall n$.

An ensemble of ten 1000-sample realisations of X is then generated, from which the sample means and standard deviations are obtained and plotted together with their theoretical values (red lines).

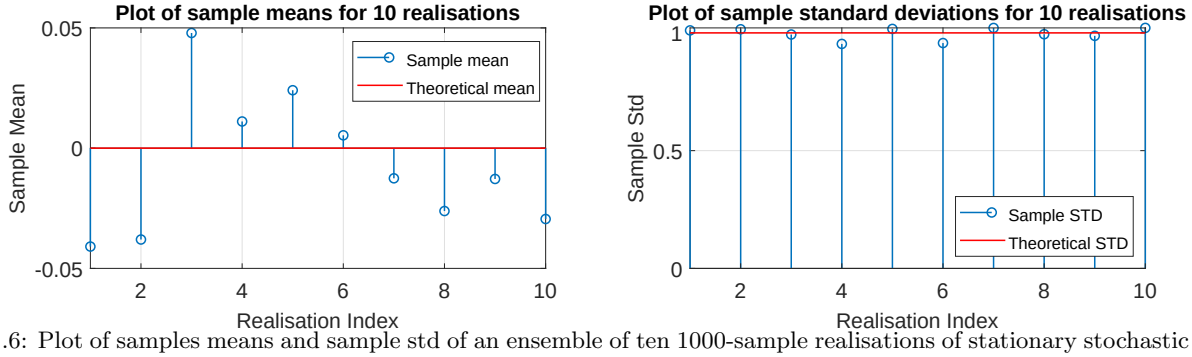


Figure 1.6: Plot of samples means and sample std of an ensemble of ten 1000-sample realisations of stationary stochastic process $X_n \sim \mathcal{N}(0,1), \forall n$.

The average bias for sample mean was found to be 0.278 and for sample std -0.719. This proves quantitatively that these estimates cluster about their theoretical values.
The pdf of X was estimated using the procedure described in part 1.1.1, generating a histogram normalised by the number of samples considered. This was then plotted together with the theoretical pdf, for different sample numbers and for different bin numbers.

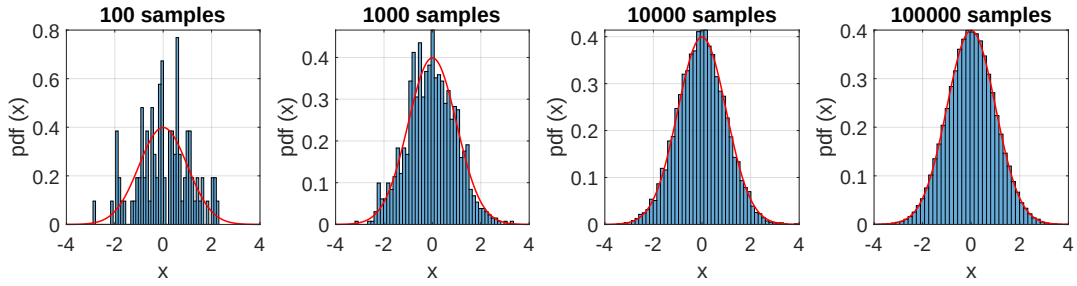


Figure 1.7: Effect of number of generated samples on the histogram pdf approximation for distribution of X. Estimated pdf is also compared against theoretical pdf of $X_n \sim \mathcal{N}(0,1)$ (Red Line). Number of bins used is 50.

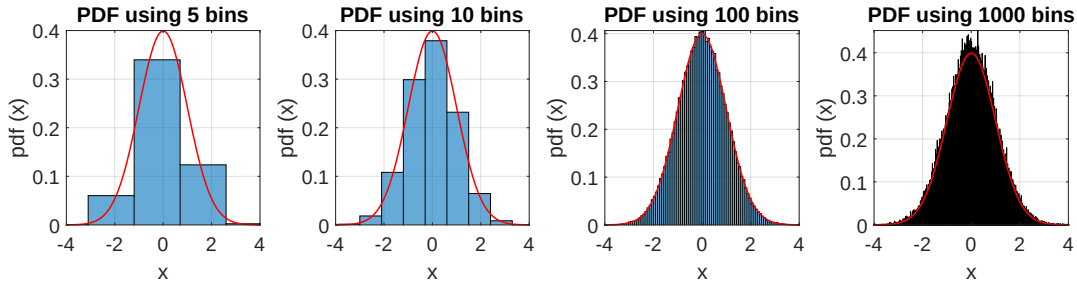


Figure 1.8: Effect of number of bins on the histogram pdf approximation for distribution of X. Estimated pdf is also compared against theoretical pdf of $X_n \sim \mathcal{N}(0,1)$ (Red Line). Number of samples used is 100000.

As for the case of uniformly distributed stochastic process, Figure 1.7 shows that the estimate converges to the theoretical pdf as the number of generated samples increases. Figure 1.8 shows that as the number of bins considered increases, the estimated pdf starts to deviate from the theoretical pdf, due to increase in variance of estimator, however resolution of estimation increases.
Note: In general, the mean squared error (MSE) error for an estimator is given by $MSE(\hat{\theta}) = Var(\hat{\theta}) + bias^2(\hat{\theta})$, where in our case $\hat{\theta} = \hat{f}_X(x)$, so given the desired MSE, one can determine the optimal number of equally sized bins to be used.
Note: Figure 1.9 shows that for both uniform and gaussian distribution, sample mean and sample std converge to theoretical values (Asymptotic Consistency).

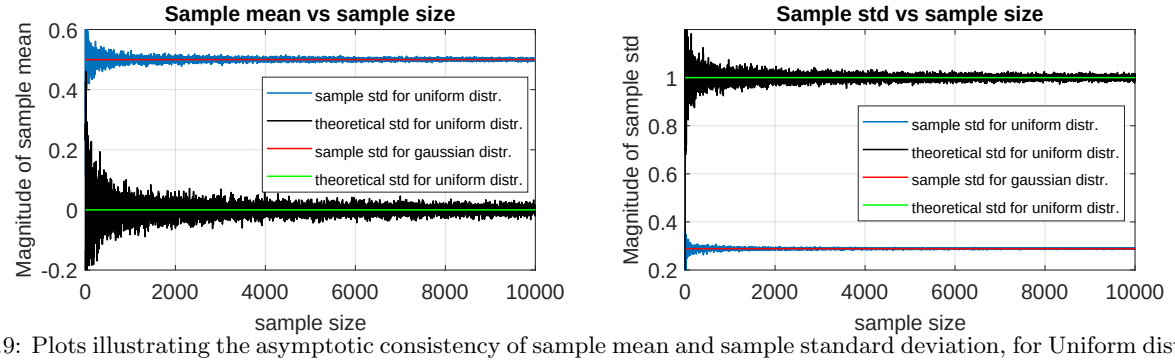


Figure 1.9: Plots illustrating the asymptotic consistency of sample mean and sample standard deviation, for Uniform distribution and Gaussian distribution. Theoretical values are plotted for comparison. Graphs show that as sample size increases, these estimators converge to the theoretical values.

1.2 Stochastic Processes

In this section, three stochastic processes are studied from which an ensemble of M N-sample realisations are obtained. Relation between time averages and ensemble averages is investigated to understand properties of the processes such as stationarity and ergodicity.

1.2.1 - Stationarity

Ensemble mean and standard deviation for each process were computed and then plotted as a function of time.

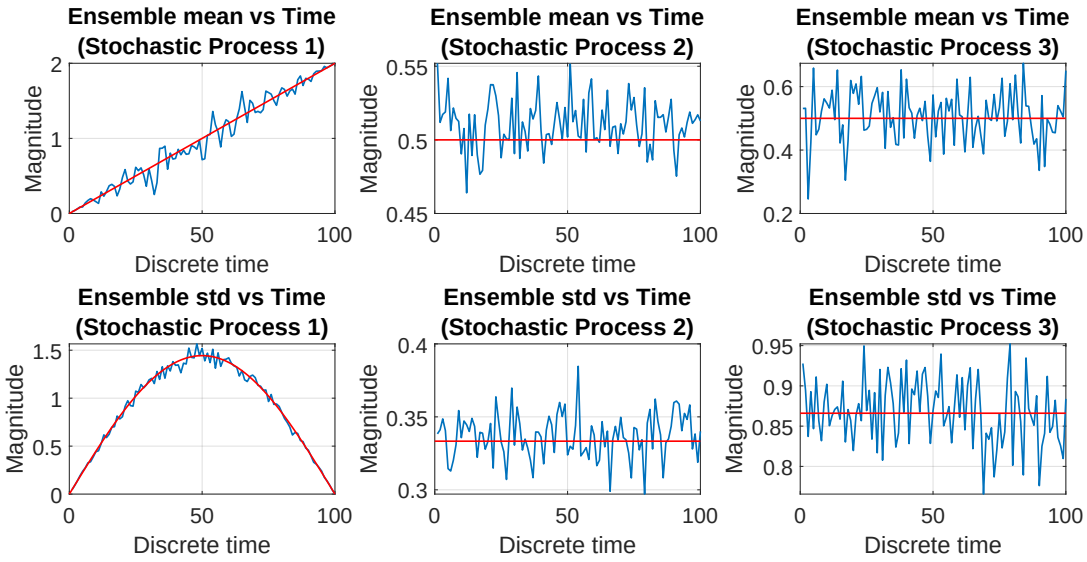


Figure 1.10: Plots of ensemble mean and standard deviation for each stochastic process as a function of time, using 100 realisations each of sample size 100. Blue lines show the ensemble mean/standard deviation and red lines the corresponding theoretical values.

A stationary process is a process whose unconditional joint probability distribution does not change when shifted in time. If we only consider first order statistics, such as mean and variance, and these are constant with time independent, then the process is said to be first order stationary. From the figure 1.10, it can be seen that for Stochastic processes 2 and 3, ensemble mean and standard deviation are relatively constant with time. However, for Stochastic Process, ensemble mean and std are functions of time. Therefore, Processes 2 and 3 are stationary whereas Process 1 is non-stationary.

1.2.2 - Ergodicity

An ensemble of four 1000-sample realisations were obtained for each process, for which time average and standard deviation for each process were computed.

Table 1: Table of mean and standard deviation for the three stochastic processes. For each process the average value of mean and std was calculated for the four realisations.

	Process 1		**Process 2**		**Process 3**	
Realisation Index	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
1	10.040	5.852	0.641	0.106	0.506	0.884
2	9.971	5.837	0.507	0.022	0.510	0.860
3	9.965	5.878	0.894	0.263	0.486	0.853
4	10.030	5.871	0.044	0.190	0.452	0.872
Average Value	**10.002**	**5.860**	**0.552**	**0.145**	**0.489**	**0.867**

A stochastic process is ergodic for the mean when the time average and standard deviation of a single realisation converge to the ensemble average and standard deviation. This means that given a single realisation, the ensemble average and standard deviation are estimated by the average and standard deviation of the process over time.

In this case the average value of temporal mean of Process 1 (10.002) does not converge to theoretical mean of 20 (at sample size of 1000). Therefore, this suggests that Process 1 is non-ergodic. This can also be seen from the fact that this process in non-stationary, therefore it cannot be ergodic. It should be noted that if a process is stationary, it is not necessarily ergodic. Finally, the theoretical variance of the process is a function of time.
For Process 2, although the average value of temporal mean is 0.552 which is very close to the theoretical ensemble mean, there is significant variation between the realisations concerning the temporal mean. Additionally the average value of temporal standard deviation of the process (0.145) does not converge to the theoretical value of $\frac{1}{3}$, and there is also significant variation on the values in each realisation. Therefore, this means that process 2 is non-ergodic.
For Process 3, the average value of temporal mean (0.489) converges to the theoretical ensemble average of 0.5 and the values in each realisation vary closely around the average value. Additionally the average value of temporal standard deviation of the process (0.867) converges to the theoretical value of $\frac{\sqrt{3}}{2} \approx 0.866$, with low variation between realisations. Therefore, this suggests that process 3 is ergodic.

Table 2: Summary on stationary and ergodicity of each stochastic process analysed in section 1.2.1 and 1.2.2 .

	Stationary	**Ergodic**
Process 1	NO	NO
Process 2	YES	NO
Process 3	YES	YES

Note: The theoretical values of ensemble mean and std of each process are derived in part 1.2.3 .

1.2.3 - Mathematical Description of the three Stochastic Processes

In this section a mathematical description for each process is provided and for each process the theoretical mean and variance are calculated.

Process 1

- Mathematical Description: $\mathbf{v[n]} = (\mathbf{X} - \mathbf{\frac{1}{2}})\mathbf{M_c} + \mathbf{A_c}$, where $X \sim \mathcal{U}(0,1)$, $M_c = bsin(\frac{n\pi}{N})$ and $A_c = an$.
  Additionally, a=0.02 , b=5 , $1 \leq n \leq N$ and N=sample size.
- Theoretical mean:

$$E\{v[n]\} = E\{(X - \frac{1}{2})bsin(\frac{n\pi}{N}) + an\} = E\{(X - \frac{1}{2})bsin(\frac{n\pi}{N})\} + E\{an\} \text{ due to linearity of expectation operator.}$$
$$= bsin(\frac{n\pi}{N})E\{(X - \frac{1}{2})\} + an = bsin(\frac{n\pi}{N})\left[E\{X\} - \frac{1}{2}\right] + an \text{ , where } E\{X\} = \frac{1}{2} \text{ for } X_n \sim \mathcal{U}(0,1)$$
$$= bsin(\frac{n\pi}{N})\left[\frac{1}{2} - \frac{1}{2}\right] + an = an = \mathbf{0.02n}. \text{ Therefore theoretical mean increases linearly with } n.$$

- Theoretical variance:

$$Var(v[n]) = E\{(v[n] - E\{v[n]\})^2\} = E\{v^2[n]\} - E\{v[n]\}^2 = E\Big\{[(X - \frac{1}{2})bsin(\frac{n\pi}{N})]^2\Big\} - a^2n^2$$
$$= E\Big\{(X - \frac{1}{2})^2b^2sin^2(\frac{n\pi}{N})\Big\} + E\Big\{2anbsin(\frac{n\pi}{N})(X - \frac{1}{2})\Big\} + E\{a^2n^2\} - a^2n^2$$
$$= b^2sin^2(\frac{n\pi}{N})E\{X^2 - X + \frac{1}{4}\} + 2anbsin(\frac{n\pi}{N})E\{(X - \frac{1}{2})\}, \text{ where we know that } E\{(X - \frac{1}{2})\} = 0$$
$$Var(v[n]) = b^2sin^2(\frac{n\pi}{N})\left[E\{X^2\} - E\{X\} + \frac{1}{4}\right] = b^2sin^2(\frac{n\pi}{N})\left[Var(X) + E\{X\}^2 - \frac{1}{2} + \frac{1}{4}\right]$$
$$= b^2sin^2(\frac{n\pi}{N})\left[\frac{1}{12} + \frac{1}{4} + \frac{1}{4} - \frac{1}{2}\right] = \frac{b^2}{12}sin^2(\frac{n\pi}{N}) = \mathbf{\frac{25}{12}sin^2(\frac{n\pi}{N})} \text{ , since } Var(X) = \frac{1}{12} \text{ for } X \sim \mathcal{U}(0,1)$$

  Therefore $\sigma_{(v[n])} = \sqrt{\frac{25}{12}sin^2(\frac{n\pi}{N})} = \frac{5}{\sqrt{12}}sin(\frac{n\pi}{N})$.

Process 2

- Mathematical Description: $\mathbf{v[n]} = (\mathbf{X} - \mathbf{\frac{1}{2}})\mathbf{M_r} + \mathbf{A_r}$, where $X_n \sim \mathcal{U}(0,1)$, $M_r \sim \mathcal{U}(0,1)$ and $A_r \sim \mathcal{U}(0,1)$.
- Theoretical mean:
  $E\{v[n]\} = E\{(X - \frac{1}{2})M_r + A_r\} = E\{(X - \frac{1}{2})\}E\{M_r\} + E\{A_r\} = E\{A_r\} = \mathbf{\frac{1}{2}}$, *since the random variables* X, M_r *and* A_r *are statistically independent.*
- Theoretical variance:

$$Var(v[n]) = E\{\left[(X - \frac{1}{2})M_r + A_r\right]^2\} - E\{v[n]\}^2 = E\{(X - \frac{1}{2})^2M_r^2 + 2M_rAr(X - \frac{1}{2}) + A_r^2\} - (\frac{1}{2})^2$$
$$= E\{(X - \frac{1}{2})^2M_r^2\} + 2E\{M_rA_r(X - \frac{1}{2})\} + E\{A_r^2\} - \frac{1}{4}$$
$$= E\{(X - \frac{1}{2})^2\}E\{M_r^2\} + 2E\{M_r\}E\{A_r\}E\{(X - \frac{1}{2})\} + Var(A_r) + E\{A_r\}^2 - \frac{1}{4}$$
$$= \left[E\{X^2\} - E\{X\} + \frac{1}{4}\right]\left[Var(M_r) + E\{M_r\}^2\right] + \frac{1}{12} + \frac{1}{4} - \frac{1}{4}$$
$$= \left[\frac{1}{12} + \frac{1}{4} - \frac{1}{2} + \frac{1}{4}\right](\frac{1}{12} + \frac{1}{4}) + \frac{1}{12} = \mathbf{\frac{1}{9}}$$

Therefore $\sigma_{(v[n])} = \sqrt{\frac{1}{9}} = \frac{1}{3}$.

Process 3

- Mathematical Description: $\mathbf{v[n]} = (\mathbf{X} - \mathbf{\frac{1}{2}})\mathbf{m} + \mathbf{a}$, where $X_n \sim \mathcal{U}(0,1)$, $m = 3$ and $a = 0.5$.
- Theoretical mean:
  $E\{v[n]\} = mE\{(X - \frac{1}{2})\} + a = a = \mathbf{\frac{1}{2}}$
- Theoretical variance:

$$Var(v[n]) = E\{v^2[n]\} - E\{v[n]\}^2 = E\{\left[(X - \frac{1}{2})m + a\right]^2\} - \frac{1}{4} = E\{(X - \frac{1}{2})^2 m^2 + 2am(X - \frac{1}{2}) + a^2\} - \frac{1}{4}$$

$$= m^2 E\{X^2 - X + \frac{1}{4}\} + \frac{1}{4} - \frac{1}{4} = m^2 E\{X^2\} - m^2 E\{X\} + \frac{m^2}{4} = 9\left[\frac{1}{12} + \frac{1}{4}\right] - \frac{9}{2} + \frac{9}{4} = \mathbf{\frac{3}{4}}$$

Therefore $\sigma_{(v[n])} = \sqrt{\frac{3}{4}} = \frac{\sqrt{3}}{2}$.

The values of theoretical means and standard deviations agree with the results obtained when computing in MATLAB the ensemble means and standard deviations for the three stochastic processes. The theoretical values are plotted in figure 1.10 in Part 1.2.2 as red lines, allowing direct comparison of the values. Additionally these values are also compared to temporal averages and standard deviations in Part 1.2.2 . It can be observed that for stochastic process 2 the values of ensemble standard deviation are deviated from the theoretical value (have an average value of ≈ 0.31 instead of $\frac{1}{3}$, however both mean and std remain constant with time.

1.3 Estimation of probability distributions

The purpose of this section is the design and implementation of a probability density function estimator, which will be testes on stochastic processes. This process is called *density estimation* and histogram estimator is a method of non-parametric density estimation. This method relies only on the assumptions that $f(x)$ is smooth and continuous and produces a non-smooth estimator $\hat{f}(x)$.

1.3.1 - Designing and testing pdf estimator for a stationary Gaussian pdf

The pdf estimator is based on the MATLAB function `hist` and is designed to give an estimate of the pdf of a collection of samples. The code for histogram estimator is given in next page.

```
function pdf(x)
    nbins = 100; %Default is 10
    [counts,centers] = hist(x,nbins);
    Tot_width=max(x)-min(x);
    bin_size=Tot_width/nbins;
    counts_tot=sum(counts);
    rel_freq=counts./(counts_tot*bin_size); % converts Counts to relative frequency
    bar(centers,rel_freq,'FaceColor','b','EdgeColor','k')
end
```

All commands used were based on MATLAB documentation for `hist` function. `hist` itself will give us the total number of counts (frequency) in each bin and will automatically plot the histogram ,however, this will not be normalised to be used as pdf estimator. For this reason, the method in the code above is used. Number of counts in each of the equally spaced bins are used for normalisation and plotting and bin centers(locations) are required by `bar` command, which allows for modification of the plot, unlike `hist`. Using this function in MATLAB, the estimator is tested for a stationary process with Gaussian pdf, for different data lengths that are at least 100. As sample size increases, the pdf estimation converges to theoretical pdf.

1.3.2 - Testing pdf estimator for a stationary and ergodic processes of Part 1.2

In Part 1.2, it was found that only Process 3 is both stationary and ergodic. The pdf estimator is tested on this process for different samples sizes. The theoretical pdf derived below is also plotted on the same plot allowing for direct comparisons.

Obtaining theoretical pdf of Process 3: Defined as $\mathbf{v[n]} = (\mathbf{X} - \mathbf{\frac{1}{2}})\mathbf{m} + \mathbf{a}$, where $X_n \sim \mathcal{U}(0,1)$, $m = 3$ and $a = 0.5$.

We know that $f_X(x) = \begin{cases} 1 & , \ for\ 0 \le x \le 1 \\ 0 & , \ otherwise \end{cases}$, and that $v[n] \in [-1, 2]$, since when X=0, $v[n] = -\frac{1}{2}(3) + \frac{1}{2} = -1$ and when X=1, $v[n] = (1 - \frac{1}{2})(3) + \frac{1}{2} = 2$.

Cumulative distribution function for v[n] is $F_{v[n]}(v) = P(v[n] \le v) = P((X - 0.5)m + a \le v) = P(X \le \frac{v-a}{m} + 0.5)$. Therefore, this shows that $F_{v[n]}(v) = F_X\left(\frac{v-a}{m} + 0.5\right)$.

$F_{v[n]}(v) = \int_0^{\frac{v-a}{m}+0.5} f_X(x)\,dx = [x]_0^{\frac{v-a}{m}+0.5} = \frac{v-a}{m} + 0.5$, so since $f_{v[n]}(v) = \frac{d}{dv}F_{v[n]}(v)$ then we know that the theoretical pdf of the process is:

$$f_{v[n]}(v) = \frac{1}{m} = \mathbf{\frac{1}{3}} \ , \ for \ -1 \le v[n] \le 2 \ . \tag{7}$$

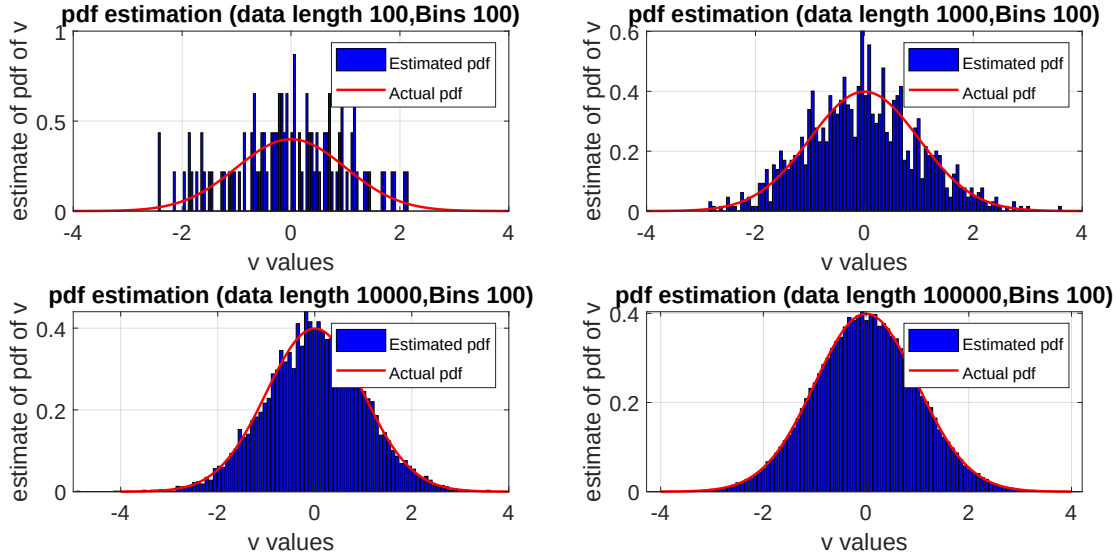


Figure 1.11: Testing pdf estimator for different data lengths($\geq$ 100) using 100 bins for a stationary Gaussian process. The

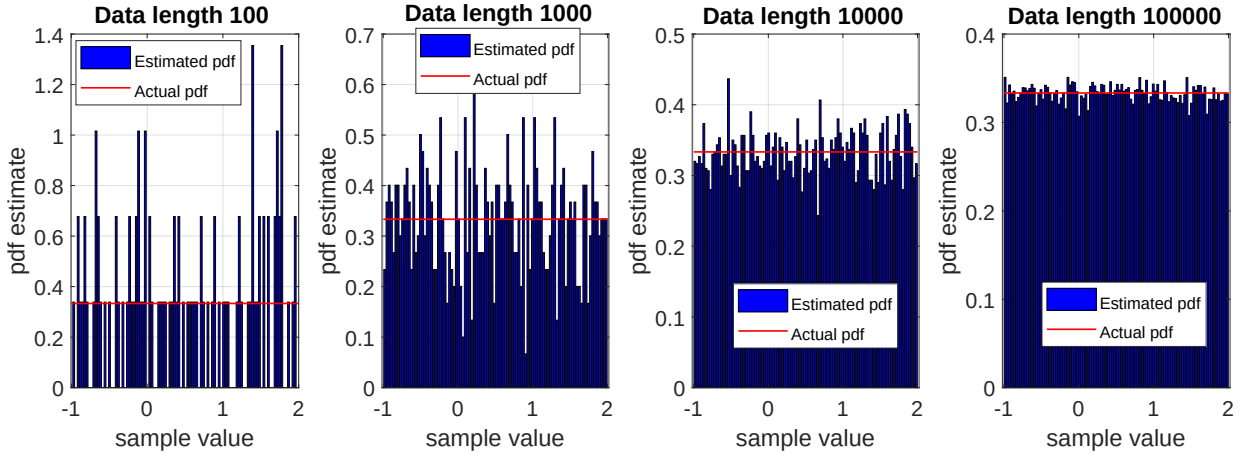


Figure 1.12: Testing pdf estimator for stationary and ergodic process 3 from Part 1.2 for $N \in \{100, 1000, 10000\}$, using 100 bins. The theoretical pdf is plotted in red for comparison.

Figure 1.12 shows that the pdf estimator converges to the true (theoretical) pdf when the number of samples increases, and variations around the theoretical value are decreasing.
Note: The fact that pdf of process is constant, i.e. independent of n, proves that process 3 is stationary.

1.3.3 - Estimating pdf for a non-stationary process

The designed pdf estimator cannot be used to estimate the pdf of a nonstationary process since the density function of the process is a function of time, so mean and standard deviation are not time-invariant. In the case that pdf estimator is used for a non-stationary process, then the estimator output can be thought of as a weighted average of the probability distributions at each value of n. Therefore, this means that the pdf estimation is not valid (not representing the true pdf of the process).
In the case of a 1000-sample-long realisation, of a non-stationary stochastic process, for which the mean of a signal changes from 0 to 1, after sample point N = 500, we cannot use the pdf estimator on the whole realisation. However, we could instead separate the realisation into two time windows (segments), one window between n=1 and n=500 and the other between n=501 and n=1000. The designed pdf estimator will then be used to approximate independently the pdf of each segment. This method will only give a correct pdf approximation under the following assumptions:

- The two pdf's are independent.
- Each point in each segment follows the same distribution, i.e. the process in each segment is stationary.
- The change in the mean is done by an instantaneous deterministic change in the process.

Under these assumptions, we can model the covariance-stationary stochastic process in each segment as the sum of a deterministic process and a white noise component (stochastic process), according to Wold's decomposition theorem. Furthermore the change in mean is assumed to be an instantaneous deterministic DC offset of magnitude 1. Due to restrictions in the use of pdf function listed above, estimation obtained is very rough, but still allows pdf estimator to be used. Interpretation of results, however, must be done with caution.

Model of the process v[n], using noise component X with mean zero and variance σ^2:

$$v[n] = \begin{cases} v_1[n] = X & , \; for \; 1 \leq n \leq 500 \\ v_2[n] = X + 1 & , \; for \; 501 \leq n \leq 1000 \end{cases} \text{, therefore } f_{v[n]}(v) = \begin{cases} f_{v_1[n]}(v) & , \; for \; 1 \leq n \leq 500 \\ f_{v_2[n]}(v) & , \; for \; 501 \leq n \leq 1000 \end{cases}$$

2 Linear stochastic modelling

2.1 ACF of uncorrelated and correlated sequences

2.1.1 -Unbiased Estimate of ACF for 1000-sample realisation of WGN

The autocorrelation function (ACF) of a stochastic process X_n is given by $R_X(n, n+\tau)) = E\{x[n]\overline{x[n+\tau]}\}$. If the process is stationary, then $R_X(n, n+\tau) = R_X(\tau)$. For an independent real stochastic process (assuming stationary) theoretical ACF is given by:

$$R_X(\tau) = \begin{cases} E\{x^2[n]\} = \sigma^2 + \mu^2 & , \; for\ \tau = 0 \\ E\{x[n]x[n+\tau]\} = E\{x[n]\}E\{x[n+\tau]\} = \mu^2 & , \; for\ \tau \neq 0 \end{cases}$$

For White Gaussian noise (WGN) which is a stationary and independent stochastic process with zero-mean and unit-variance, the theoretical ACF is expected to be:

$$R(\tau) = \sigma^2\delta(\tau) = \begin{cases} \sigma^2 & , \; for\ \tau = 0 \\ 0 & , \; for\ \tau \neq 0 \end{cases} , \; therefore \; , \; R_{WGN}(\tau) = \begin{cases} 1 & , \; for\ \tau = 0 \\ 0 & , \; for\ \tau \neq 0 \end{cases}$$

For an ergodic process, with only one finite-sample realisation ACF can be estimated using the unbiased ACF estimator ($\hat{R}_X(\tau)$) given by equation 8. The ACF estimate (or correlogram) is plotted in Fig. 2.1 for $\tau \in [-999 : 999]$.

$$\hat{R}_X(\tau) = \frac{1}{N - |\tau|} \sum_{n=0}^{N-|\tau|-1} x[n]x[n+|\tau|], \; for\ \tau = -N+1, ..., N-1 \; used\ by\ MATLAB\ command\ xcorr() \; . \tag{8}$$

Figures 2.1(a)(b) shows that the ACF estimate is similar to a discrete dirac delta function, with a value close to 1 for τ=0, and values varying around zero for other τ values. However, deviations of these values are increasing with increasing τ values. The reason for ACF estimate being different than theoretical ACF, is due to small (finite) sample size used. It is expected that as sample size increases, ACF estimate will converge to theoretical ACF.
Additionally, figure 2.1 (c) shows that ACF estimate is symmetric about zero lag (τ=0). This is due to the fact that the ACF is a Hermitian function, i.e. $R_X(-\tau) = \overline{R_X(\tau)}$. Since signal x[n] is real, this means that ACF is an even function and thus symmetric about zero lag.

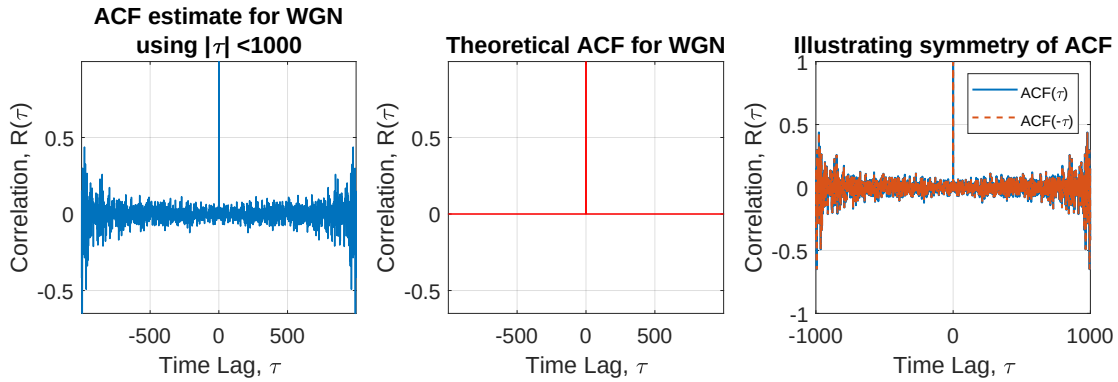


Figure 2.1: (a) Unbiased estimate of ACF for a single 1000-sample realisation of WGN and (b) Theoretical ACF . (c) Illustrating symmetry of ACF estimate through plot of $R(\tau)$ and $R(-\tau)$.

2.1.2 - Observing ACF estimate for different time-lag values

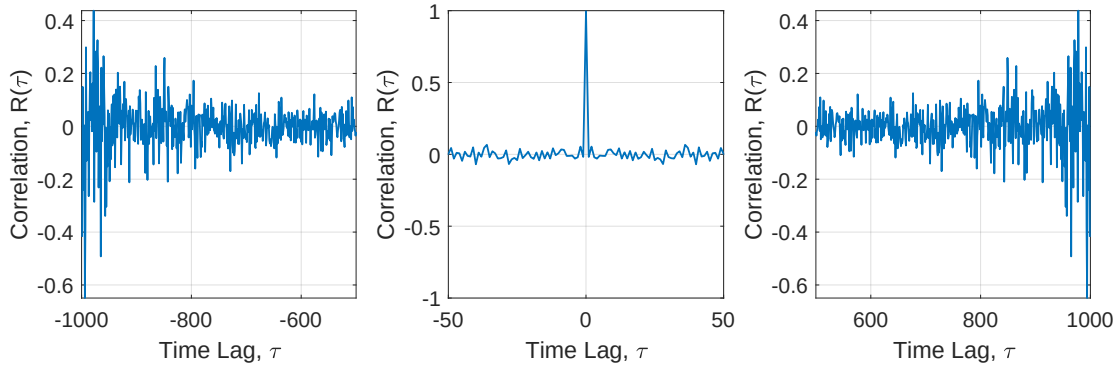


Figure 2.2: Unbiased estimate of ACF for a 1000-sample realisation of WGN ,(Left) using $\tau < -500$, (Middle) using $|\tau| < 50$ and (Right) using $\tau > 500$

Using the `zoom(20)` command the ACF is displayed in Fig 2.2 for the region $|\tau| < 50$. This shows that for small time lag values, the ACF estimate agrees with the theoretical ACF, with only small deviations from 0 for $\tau \neq 0$. This verifies that indeed the stochastic process has approximately uncorrelated samples, due to small correlation between samples (approximately zero). Therefore, the value of signal at a specific time instant cannot be predicted with any confidence based on past (or future) signal values.
On the other hand, when larger time lag values where considered, as shown in Fig. 2.2, the ACF estimate deviates more and more significantly as τ approaches N-1 or -N+1. As an effect, this shows that for large τ values, there is high correlation between samples.

This can be explained from the fact that as time lag increases, the number of samples used in the calculation of ACF estimate decreases. As a result, for larger time lag values, the ACF estimate will diverge more from the theoretical ACF and for small time lag values, ACF will converge more to the theoretical ACF.

2.1.3 - Effect of large autocorrelation lags $|\tau|$ on the accuracy of ACF estimates

The effects of large autocorrelation time lags on accuracy of ACF estimates can be explained according to equation 8. The value of $|\tau|$ affects the factor of $\frac{1}{N-|\tau|}$ in equation and the number of terms involved in the summation which are $N-|\tau|$. This means that as $|\tau|$ increases, both terms decrease. As a result, equation 8 diverges from the theoretical equation of ACF given by:

$$R_X(\tau) = \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} x[n]x[n+\tau], \ \ for\ \tau \in \mathbb{Z}. \tag{9}$$

This means that the number of samples that are involved in the ACF estimate calculation is reduced, causing the variance of estimator to increase, and thus its accuracy to decrease. Since variance of the ACF estimator is given by $\frac{1}{N-|\tau|}$, therefore as τ increases, variance of estimator increases. This means that for large lag values, the ACF estimate is not statistically reliable. Note: Another possible solution in general is the usage of biased ACF estimate, that has variance independent of τ and can be helpful for identification of shape of ACF (bias-variance tradeoff).

$$\begin{aligned} Var(\hat{R}_X(\tau)) &= \frac{1}{(N-|\tau|)^2} Var\left(\sum_{n=0}^{N-|\tau|-1} x[n]x[n+\tau]\right) = \frac{1}{(N-|\tau|)^2} \sum_{n=0}^{N-|\tau|-1} Var(x[n]x[n+\tau]) \\ &= \frac{(N-|\tau|)}{(N-|\tau|)^2}\left[\mu_{X_n}^2 Var(x[n+\tau]) + \mu_{X_{n+\tau}}^2 Var(x[n]) + Var(x[n])Var(x[n+\tau])\right] = \frac{1}{N-|\tau|} \end{aligned} \tag{10}$$

For large values of (N-$|\tau|$),i.e. when $|\tau|$ is much smaller than N, the ACF estimator follows approximately normal distribution with zero-mean and variance approximately equal to $\frac{1}{N}$. This is because ACF estimate is equivalent to the sample mean of the product of x[n] and x[n+τ]. The product of these signal values, which are independent and are separated by time lag τ, has mean equal to 0 and variance equal to 1. By assuming that $|\tau|$ is very small, the standard error of the estimate is equal to $\frac{1}{\sqrt{N}} = \frac{1}{\sqrt{1000}} \approx 0.0316$ which is also the minimum possible standard error for the given sample size. To find the bound for $|\tau|$ we will find the lag value for which ACF estimate exceeds 3 standard errors, i.e. when it exceeds a value of 0.0948, as shown in Fig. 2.3. The value of 3 standard errors, is equivalent to 99.5%-confidence interval, in other words within this interval we are 99.5% confident that this is a realisation of WGN noise and outside this interval, ACF estimate values are statistically unreliable. Due to random errors, outliers may occur, so the empirical bound for $|\tau|$ is obtained when the ACF estimate crosses the maximum allowable value for the second time. For our particular realisation this value is found to be $|\tau| = 271$.

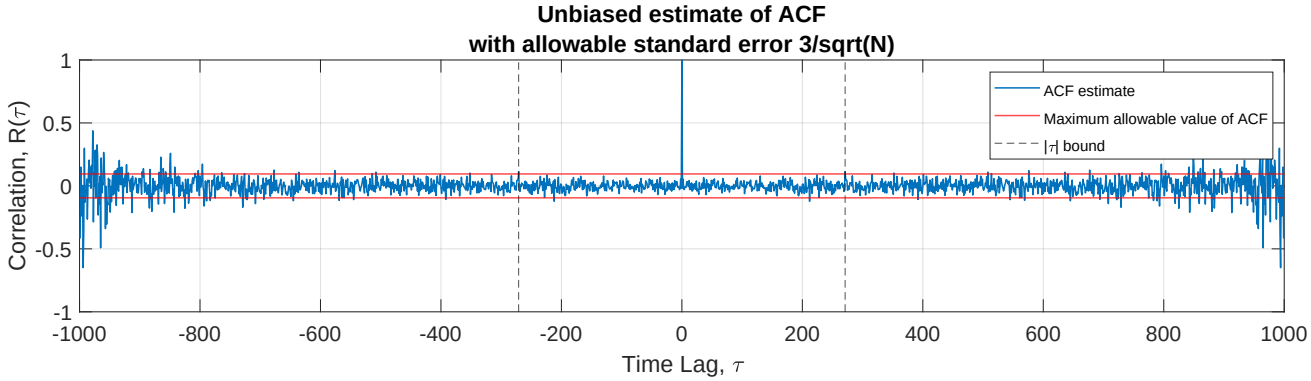


Figure 2.3: Illustrating maximum allowable value for ACF estimate and $|\tau|$ bound

2.1.4 - Filtering WGN with Moving Average (MA) filter

A 1000-sample WGN vector **x** was generated which was then filtered by a 9th-order MA filter. The ACF estimate of the output of this process is shown in Figure 2.4, for lags $|\tau| < 20$. This is equivalent to a Moving Average process of order 8 (MA(8)).

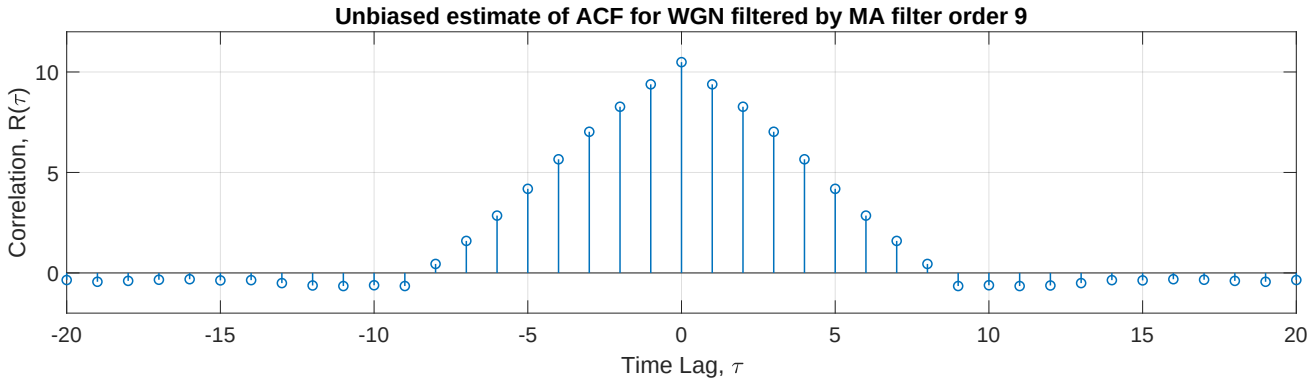


Figure 2.4: ACF estimate of the WGN filtered by MA filter for $|\tau| < 20$

Figure 2.4 shows that the shape of the ACF estimate is triangular for values of $|\tau| < m$, where m is the order of the unit coefficients of the filter, or the window length. The MA filter used, effectively increases the correlation between the current sample value with the previous 8 samples, meaning that these values within $|\tau| < 9$ are not uncorrelated as before.
For lags≥ 9, the ACF estimate values behave the same way as for the case we saw in parts 2.1.1 - 2.1.3 involving unfiltered WGN realisation (or for MA filter of order 1, i.e. y[n]=x[n]) with very low ACF values. The reason for the introduction of correlation between filter's output samples is because the MA filter effectively adds the current value to the previous 8 values, so this value is dependent on these 9 values. In other words, each output sample will contain the sum of the current value with the previous 8 samples of input.
Due to overlap of the input values used in the calculation of different output samples, the correlation is high for $\tau = 1$ and decreases to zero as τ approaches the value of 8, after which there is no overlap, which reduces correlation to zero.
The MATLAB command `y=filter(ones(order,1),[1],x)`, is equivalent to:

$$y[n] = (x[n] + x[n-1] + ... + x[n-(M-1)]) = \sum_{k=0}^{M-1} x[n-k] \ , \ where\ M\ is\ filter\ order. \tag{11}$$

This means that the impulse response of the MA filter is given by $h[n] = \sum_{k=0}^{M-1} \delta[n-k]$, where M is filter order (window length). Output y[n] is produced by the convolution of input WGN and filter's impulse response. In z-domain this is equivalent to $H(z) = \sum_{k=0}^{M-1} b_{M-1}(k) z^{-k}$, where $b_{M-1}(k) = 1$ for our case. This shows that the MA filter is a rectangular function (window) in the time-domain.
We know from theory that ACF of filter output is $R_Y(\tau) = g(\tau) * R_X(\tau)$, where $g(\tau) = h[\tau] * h[-\tau]$. The function $g(\tau)$ is a convolution of a rectangular function with its time-inverted copy, which produces an output with triangular shape (Fig. 2.5). Since autocorrelation of the input is a scaled dirac delta function, the output will be a scaled triangular function.

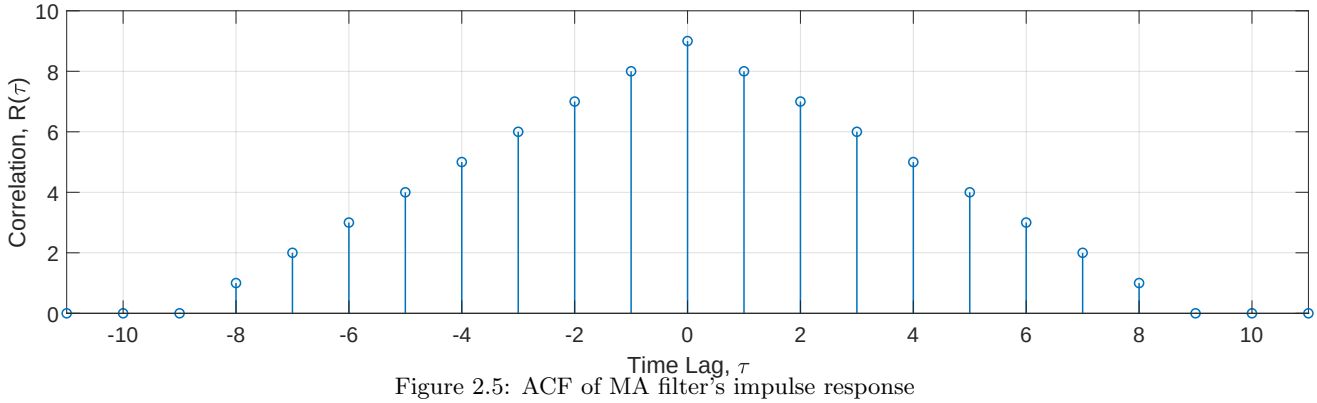


Figure 2.5: ACF of MA filter's impulse response

The higher the order of the filter, the longer the duration of its impulse response (which is equal to 2*order) and thus the wider the base of the triangular function in the output and effectively the higher the time lag required to decrease ACF to zero. This is because the degree of overlap between rectangular windows used takes longer to become zero. This is shown in Fig. 2.6. It should be noted that the peak value for ACF is equal to the filter's order, since the higher the order the more input values are added up to produce the output.

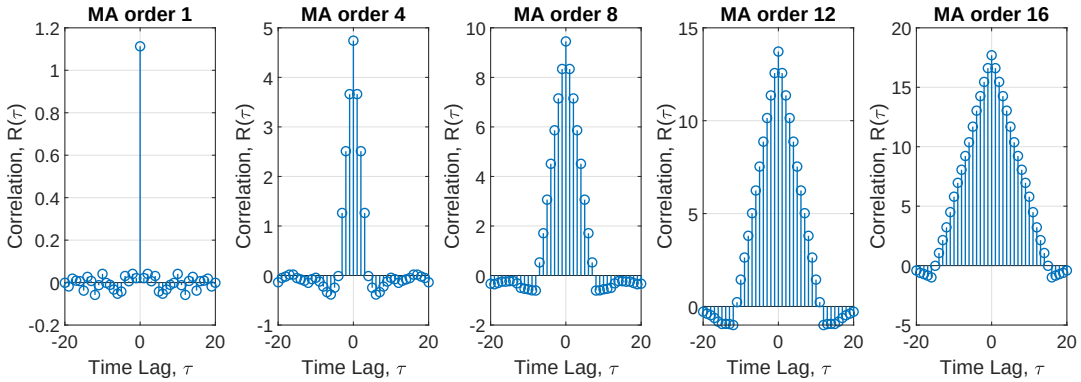


Figure 2.6: ACF estimate of the WGN filtered by MA filter of order 1, 4, 8, 12 and 16.

The (local) sample mean can be obtained by dividing the value of each output sample by (M+1) where M is the order of the filter. Then each output value will be equivalent to the local average value of the current value and previous M values. The only limitation in this approach is that for the first input values, there are not enough previous values for calculation of average, which is not consistent with the way the other local averages are calculated. Finally, this method allows for sample mean to be calculated, by using a 1000th order MA filter which will produce a single output that will be then divided by the filter's order. In other words:

$$y[n] = \frac{1}{1000} \sum_{k=0}^{999} x[n-k] = \bar{x} \tag{12}$$

2.1.5 - Mathematical description of ACF of filtered arbitrary stochastic process

The ACF of a realisation of a filtered stochastic process X_n , denoted by $R_Y(\tau)$ is given by the convolution of of ACF of filter's impulse response ($R_h(\tau)$) and ACF of the input($R_X(\tau)$):

$$R_Y(\tau) = R_X(\tau) * R_h(\tau) \tag{13}$$

If the input process X_n is uncorrelated, then we know that $R_X(\tau) = \sigma^2_{X_n}\delta(\tau) + \mu^2_X$. Due to dirac delta function property of behaving like the "identity" for convolution, and by linearity property of convolution:

$$R_Y(\tau) = \left(\sigma^2_{X_n}\delta(\tau) + \mu^2_X\right) * R_h(\tau) = \sigma^2_{X_n}\left[\delta(\tau) * R_h(\tau)\right] + \mu^2_X * R_h(\tau) = \sigma^2_{X_n}R_h(\tau) + \mu^2_X \sum_{\tau=-N}^{+N} R_h(\tau) \tag{14}$$

where N is the length of filter's impulse response. This means that the ACF of output($R_Y(\tau)$) is the scaled (amplified) version of filter's ACF, whose scale is equal to the variance of the stochastic process X_n, with an offset value, that is finite for an FIR filter.

2.2 Cross-correlation function (CCF)

2.2.1 - Unbiased CCF estimate for sequences x and y from section 2.1

The cross-correlation estimate of sequences x and y was calculated and then plotted in figure 2.7 for autocorrelation lags in the region $\tau < 20$. Sequence x is a WGN and y is the output of this WGN filtered by an MA filter with unit coefficients of order 9.

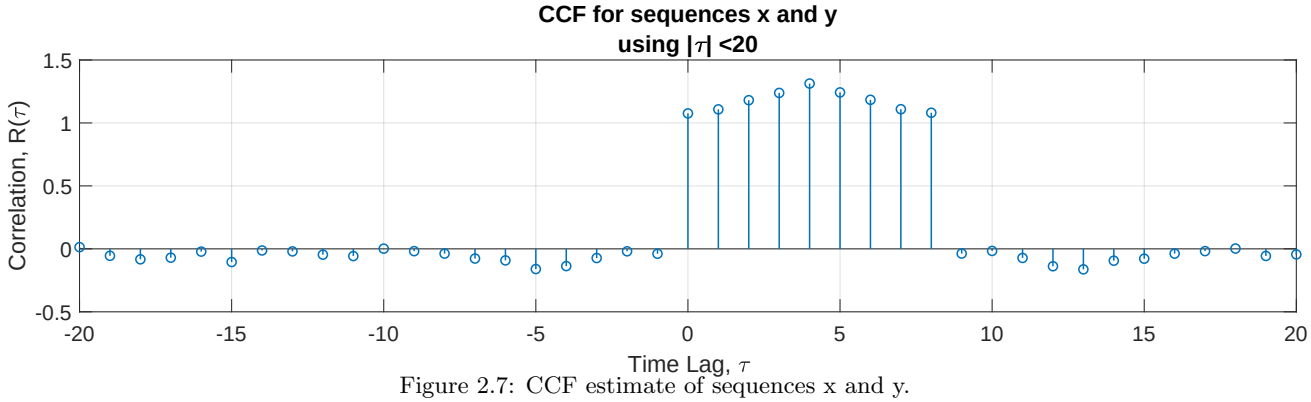


Figure 2.7: CCF estimate of sequences x and y.

Figure 2.7 shows that the CCF estimate has approximately the shape of a rectangular function, with a maximum value approximately equal to 1 for time lags of $0 \leq \tau \leq 8$ and a value close to zero outside this range. It can be seen that there are exactly nine peaks with value of approximately 1, the same as the filter's order.

This can be explained through mathematical analysis of the CCF between the two processes. We know that sequence y is related to sequence x in the following way: $y[n] = x[n] * h[n] = \sum_{k=-\infty}^{+\infty} h[k]x[n-k]$. Therefore the CCF ($R_{XY}(\tau)$) is computed in MATLAB according to the following equation:

$$\begin{aligned} R_{XY}(\tau) &= E\left\{x[n]y[n-\tau]\right\} = E\left\{x[n]\sum_{k=-\infty}^{+\infty} h[k]x[n-\tau-k]\right\} = \sum_{k=-\infty}^{+\infty} h[k]E\left\{x[n]x[n-\tau-k]\right\} \\ &= \sum_{k=-\infty}^{+\infty} h[k]R_X(-(\tau+k)) = \sum_{k=-\infty}^{+\infty} h[\tau-(-m)]R_X(m)h[\tau-(-k)] = h(\tau) * R_X(-\tau) = h(\tau) * R_X(\tau) \end{aligned} \tag{15}$$

Since process X_n is WGN, its ACF ($R_X(\tau)$) is a dirac delta function with magnitude 1 (since WGN has unit-variance) centered at $\tau = 0$. Therefore, convolution with filter's impulse response shows that $R_{XY}(\tau) = h(-\tau)$, so the CCF of x and y is expected to have the same shape as the filter's impulse response. We know that a MA filter with unit coefficients of order 9 has an impulse response given by: $h[n] = \sum_{k=0}^{M-1}\delta(n-k)$, where M is the filter's order. As a result, the CCF is expected to be a discrete delta train for values of τ between time lags of 0 and 8. Any deviations from value of 1 or 0 are due to the low number of samples used, which increase the variance of the estimator.

2.2.2 - System Identification and Order of filter

By considering the general case for which $h(\tau)$ is filter's impulse response and X_n is an uncorrelated stochastic process, the CCF of filter's input X_n and filter's output Y_n, computed using equation 15, is given by:

$$R_{XY}(\tau) = h(-\tau) * R_X(\tau) = h(-\tau) * \left(\sigma^2_X\delta(\tau) + \mu^2_X\right) = \sigma^2_X h(-\tau) + \mu^2_X \sum_{\tau=-N}^{+N} h(\tau) \tag{16}$$

which means that CCF is the same as the scaled impulse response of the filter with an offset value, whose scaling is the variance of the stochastic uncorrelated process. Therefore, CCF estimate shape can show the expected shape of the impulse response of the system (filter) that is used, if we use a zero-mean input (to remove offset). In the case of an MA filter with input WGN, number of peaks correspond to the order of filter, and the magnitude of each peak corresponds to the magnitude of the corresponding MA coefficient. A simple example is shown in third plot of Fig. 2.8 where the filter coefficients are designed to give

$$h[n] = \sum_{k=0}^{4} 4\delta(n-k) + \sum_{k=0}^{4} 3\delta(n-5-k) + \sum_{k=0}^{4} 2\delta(n-10-k) \tag{17}$$

The CCF will have the same shape as this impulse response. However, this is only possible if the system is an LTI system and input process is uncorrelated and zero-mean, although the distribution of random process can vary (e.g. same observation when we have uniformly distributed white noise).
The effect of filter's order CCF estimate is illustrated in the first 3 plots of figure 2.8. As filter order increases, the number of peaks and range for which values are not 0 increase. The value for each peak will remain approximately equal to 1 for any order of filter in the range $0 \leq \tau \leq (order - 1)$.

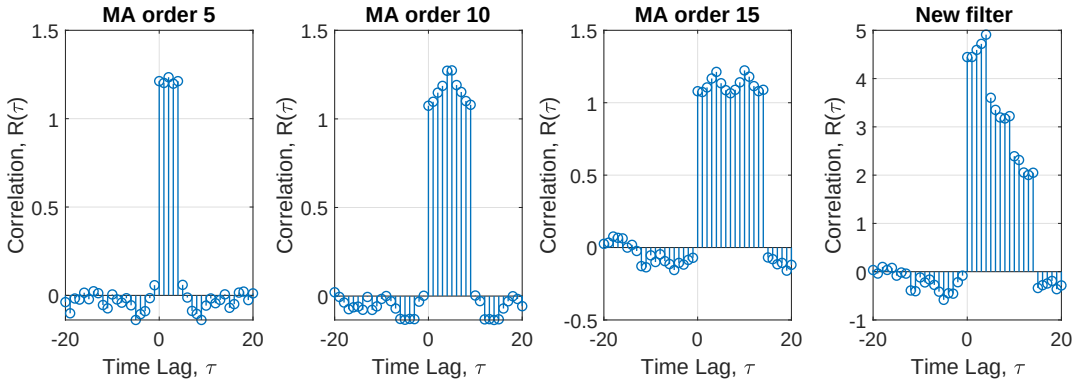


Figure 2.8: (First 3) CCF estimate of sequences x and y, with different MA unit coefficient order. (Last) CCF estimate of an arbitrary filter.

2.3 Autoregressive Modelling

2.3.1 - Stability of Autoregressive (AR) Process of order 2

An Autoregressive regressive process x[n] is generated by filtering white noise with an all-pole,causal, linear, and time-invariant IIR filter. For a process of order p, AR(p), the transfer function of filter is given by $H(z) = \frac{1}{1+\sum_{k=1}^{p} a_p(k)z^{-k}}$, therefore AR(p) can be expressed as:

$$x[n] = a_1 x[n-1] + a_2 x[n-2] + ... + a_p x[n-p] + w[n] \tag{18}$$

In our case, a second order AR process given by $x[n] = a_1 x[n-1] + a_2 x[n-2] + w[n]$ is produced with coefficients $a_1 \sim \mathcal{U}(-2.5, 2.5)$ and $a_2 \sim \mathcal{U}(-1.5, 1.5)$. The coefficient pairs (a_1, a_2) that preserve the stability of process, i.e. causing the signal x[n] to converge, were plotted in figure 2.9. The same procedure was done for those pairs that caused signal to diverge. For convergence of signal x[n] an empirical upper bound of amplitude of 1000 was used. If for any time instant value of x[n] goes above magnitude of 1000, then x[n] is assumed to be diverging.

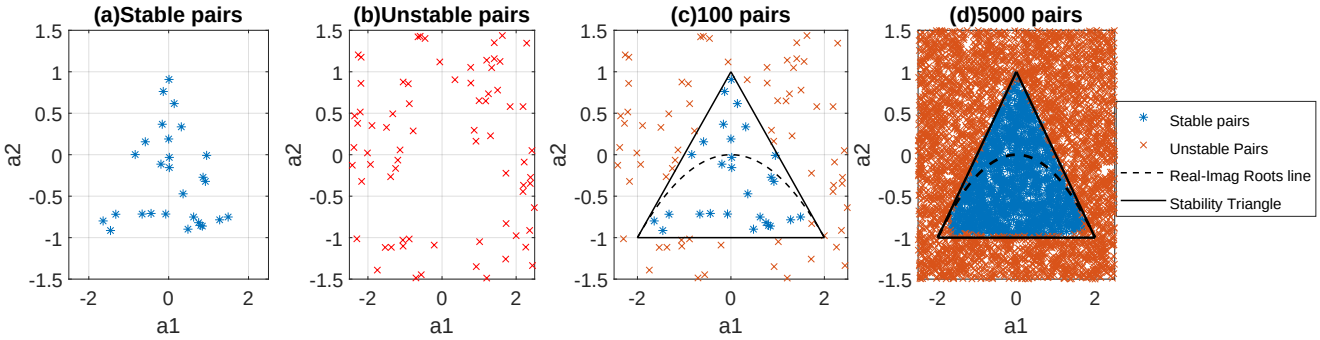


Figure 2.9: (a) Stable Pairs, (b) Unstable Pairs and (c) All pairs with stability triangle (100 pairs) (d) Using 5000 pairs

From theory, the region of convergence, or stability, of the coefficients a_1 and a_2, be obtained by setting the denominator of the LTI filter equal to zero, and obtaining the roots of this characteristic polynomial. In other words:

$$H(z) = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2}} \ , \ therefore \ we \ set \ 1 - a_1 z^{-1} - a_2 z^{-2} = 0 \tag{19}$$

The polynomial can also be expressed as $z^2 - a_1 z^1 - a_2 = 0$, whose roots are $z_{1,2} = \frac{a_1 \pm \sqrt{a_1^2 + 4a_2}}{2}$. For stability, in the z-domain, the roots of the polynomial must be within the unit circle, i.e. $|z_{1,2}| < 1$. Using this stability condition, it can be shown that three conditions must be satisfied by the coefficients of the AR(2) process by considering two cases:

- **Real roots**, i.e. $a_1^2 + 4a_2 \geq 0$
  Considering the two largest roots: $a_1 + \sqrt{a_1^2 + 4a_2} < 2 \quad and \quad a_1 - \sqrt{a_1^2 + 4a_2} > -2$, we obtain two of the three conditions from each one respectively, which specify that:
  1. $a_2 + a_1 < 1$
  2. $a_2 - a_1 < 1$
- **Complex roots**, i.e. $a_1^2 + 4a_2 < 0$:
  For this case coefficients satisfy: $\left| \frac{a_1}{2} \pm \frac{j\sqrt{-(a_1^2+4a_2)}}{2} \right| < 1$, which specifies that:
  3. $|a_2| < 1$

These three conditions, must be satisfied at the same time for stability (signal convergence) and form a triangle in the a_1-a_2 plane, with vertices located at (-2,-1),(0,1) and (2,-1). By plotting this stability triangle(Fig. 2.9 (c)) it can be shown that the stable pairs lie within this triangle and unstable pairs lie outside this triangular region. Additionally, in Fig. 2.9 (c) the dashed line drawn within the triangle shows that any real roots lie above this line and any complex roots below this line.

2.3.2 - ACF of real world sunspot time series

The ACF function of real world sunspot time series is obtained and then plotted in figure 2.10 (blue plot) for data lengths of N = 5, 20 and 250.

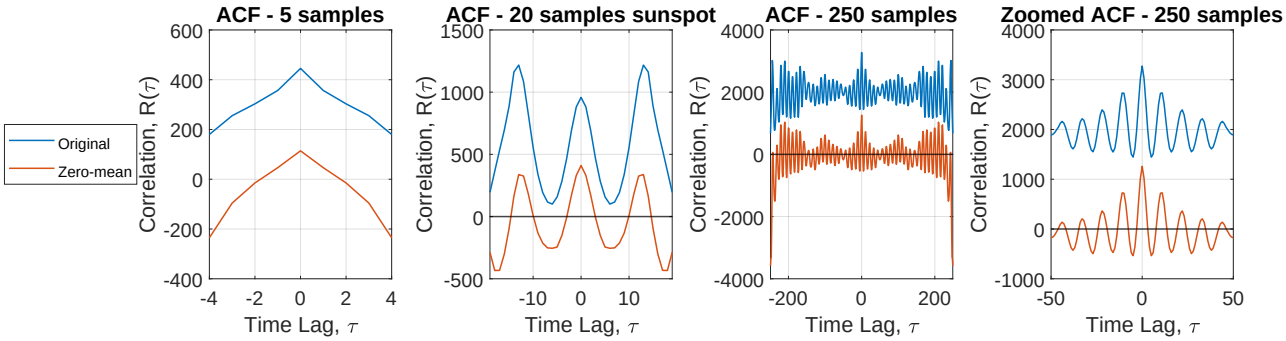


Figure 2.10: ACF estimate for sunspot time series. Plot shows the ACF both of zero-mean version and original version of series.

It is shown that for original series, the ACF estimate for N=5 has a triangular shape, which is manily due to the fact that the mean value of data is very large compared to variations in the series (high level of DC offset), which dominates in ACF estimate and "hides" any useful information for the series. However, as sample size increases the ACF function has a sinusoidal shape, indicating that the underlying time series also varies sinusoidally. This is because more samples are being considered. Additionally, for original time series, the ACF function remains constantly positive with very large value above zero. It should be noted that for large time lags ($|\tau| > 100$), the ACF estimate starts to deviate from its regular shape due to decreasing samples used in ACF estimate calculation (increasing variance) as described in previous sections.

From theory, the ACF of a periodic function is, itself, periodic with the same period. Since time lag is in years, it can be seen that the ACF has a period of approximately 13 years, in other words every 13 years the ACF reaches a peak value.

In general, for the three data lengths considered, the mean value is acting as DC offset in the signal, causing disturbances in the ACF measurements. Although the sinusoidal shape can be identified for N=20 and N=250, we can identify peaks for $\tau \neq 0$ which have a higher value than that at zero lag.

The disturbances caused by the mean-values of series can be removed by considering the zero-mean version of the series, plotted in Fig. 2.10 as red plot. For N=5, ACF has a very similar shape although it starts to deviate from the triangular shape. For higher sample-sizes, ACF still varies sinusoidally, but with no offset, therefore it alternates between positive and negative values, instead of being only positive. This technique therefore allows us to identify any underlying behavior of ACF, which is also present within the time series itself.

The zoomed ACF plot for N=250, restricted to $|\tau| < 50$, is a typical plot for a windowed sinusoidal series, which a triangular and sinusoidal shape, decaying to zero value as time lag increases.

2.3.3 - Yule-Walker equations for sunspot time series

The general form of AR model of order p is expressed as:

$$a_0 x[n] - \sum_{k=1}^{p} a_k x[n-k] = w[n] \; with \; restriction \; that \; a_0 = 1. \tag{20}$$

Yule Walker equations is a system of equations that relate the AR model parameters to the autocorrelation functions of the process. In this section we make use of Yule-Walker equations to find the AR parameters and thus calculate the partial correlation functions up until the model order p = 10 in order to obtain the most likely model order of the sunspot time series. The relation between the ACF and model parameters is given by:

$$r_{xx}(p) = \sum_{k=1}^{N} a_k r_{xx}(p-k) \; and \; in \; matrix \; form \; \mathbf{r}_{xx} = \mathbf{R}_{xx}\mathbf{a} \tag{21}$$

By performing the normalisation given by $\rho_k = \frac{r_{xx}(k)}{r_{xx}(0)}$, the normalised equations for coefficients determination are obtained (using normalised correlation coefficients). This results in the Yule-Walker equations for an AR model of order k:

$$\begin{aligned} \rho_1 &= a_{k1} + a_{k2}\rho_1 + ... + a_{kk}\rho_{k-1} \\ \rho_2 &= a_{k1}\rho_1 + a_{k2} + ... + a_{kk}\rho_{k-2} \\ &\vdots = \vdots \\ \rho_k &= a_{k1}\rho_{k-1} + a_{k2}\rho_{k-2} + ... + a_{kk} \end{aligned} \tag{22}$$

where a_{kj} is the j^{th} coefficient in an AR representation of order k. The equations can be expressed in compact matrix form as $\rho = \mathbf{R}\mathbf{a}$, where $\mathbf{R}$ is the normalised ACF matrix, which is a Toeplitz or diagonal-constant matrix, since it has unit diagonal elements. This guarantees matrix inversion. AR coefficients are found from $\mathbf{a} = \mathbf{R}^{-1}\rho$.

These operations are done by built-in MATLAB function `aryule(x,p)`, which returns the AR parameters corresponding to a model of order p for the input data x.
The partial autocorrelation functions (PACF) are the coefficients a_{kk}, which theoretically for an AR(p) process will be non-zero for $k \leq p$ and zero for $k > p$. This can be a good indication for the order of the AR process. For the original sunspot time series up to a model order 10, the PAC coefficients were obtained by the negative reflection coefficients returned by the command `aryule`. The PAC values were then plotted in figure 2.11 (a).

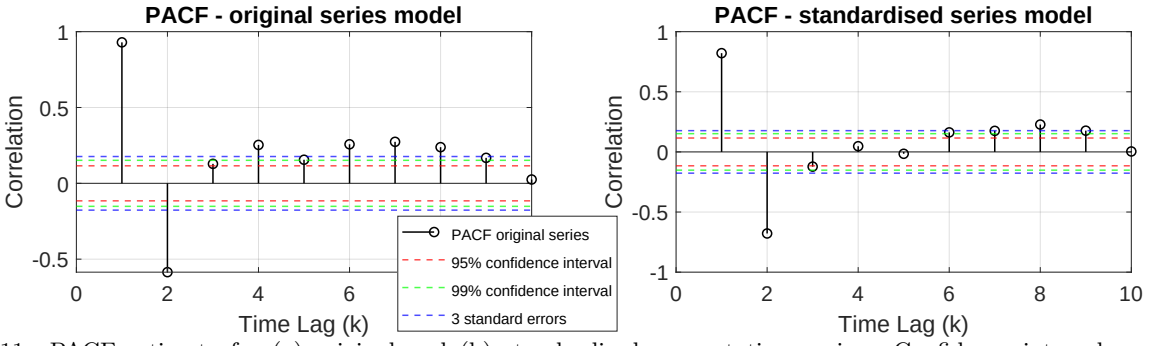


Figure 2.11: PACF estimate for (a) original and (b) standardised sunspot time series. Confidence intervals are used for determining the order of the AR model.

By plotting the Partial Autocorrelation coefficients a_{kk}, we can obtain the order of the AR(p) process by observing which coefficients are zero for $k > p$. In practise, for real world data it is difficult to guarantee that $a_{kk} = 0$ for $k > p$, so we will use 95% confidence interval with value $\pm 1.96/\sqrt{(N)}$ or 99% confidence interval with value $\pm 2.575\sqrt{(N)}$ for which PAC$\approx$ 0. The largest value of lag that does not fall within the confidence interval, is an indication of the model order. For original series, the order is found to be order 2, for 99% confidence interval (or within 3 standard errors), and order 9 for 95% confidence interval. If order 2 is used model coefficients are [-1.4740 , 0.5857].
The same procedure is repeated for standardised standardised sunspot series. This is obtained by subtracting the mean value from original series and then divide by its variance.
For both 99% confidence interval (or within 3 standard errors), the order is 2 and for 95% confidence interval, the order is 3. If order 2 is used, model coefficients are [-1.3783 , 0.6783].
The difference between the two models, is that for original sunspot series the data is not centered, which means the DC offset (deterministic component) is also modeled in addition to the stochastic component of series. This requires additional complexity in the model, thus a higher order. Therefore, the DC offset dominates the PACF estimation.
In general, the data must first be standardised and then modelled, which is exactly what the command `parcorr` does which obtains the sample PACF.
Additionally, we can obtain the estimated variance, σ^2, of the underlying white noise input. For original series of order 2, this is equal to 347.9 and for standardised series of order 2, this is equal to 0.1752.

2.3.4 - Determining the correct model order of standardised data

In general, the greater the order of AR model (over modelling), the greater its accuracy (lower model error for interpolation) but the higher its complexity. A trade-off between computational complexity and model accuracy is established by introducing a "penalty" for higher order models. Three criteria that were used on the standardised data for correct model order selection to determine the correct model order, were the minimum description length criterion (MDL), the Akaike information criterion (AIC) and the corrected Akaike information criterion (AIC_c). The definitions of these criteria are:

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

where p is the model order (number of estimated parameters) and N is data length. E_p is the loss function (cumulative squared error) or final prediction error. This is equivalent to the variance estimate of the white noise input to the AR model, obtained from the code `[a,e] = aryule(x,p)` command in MATLAB. Negative values are obtained since logarithms are used, with the loss function having values less than 1. The three criteria values, as well as the logarithm of loss function, were then plotted against model order in Figure 2.12.
The optimal order for the model is given by the location of the minima on criterion plots. It should be noted that all criteria have an increasing penalty term. The figure above shows that both MDL and AIC criteria have a minimum at order values of 9. However, corrected AIC, AIC_c, is better suited for small sample sizes by providing an additional penalty term. For MDL and AIC the slope of increasing penalty term is much lower for small sample sizes. Plot of AIC_c has a minimum at order equal to 2. This means that the optimal (correct) model order for the standardised data is 2, i.e. standardised sunspot series is optimally modeled as AR(2) process, with no overmodelling.

2.3.5 - Using AR model to predict the sunspot time series m steps ahead

The AR model generated can be used to predict the sunspot time series m steps, in the form:

$$\hat{x}[n+m] = a_1 x[n-1] + ... + a_p x[n-p] \text{ where } p \text{ is the model order.} \quad (23)$$

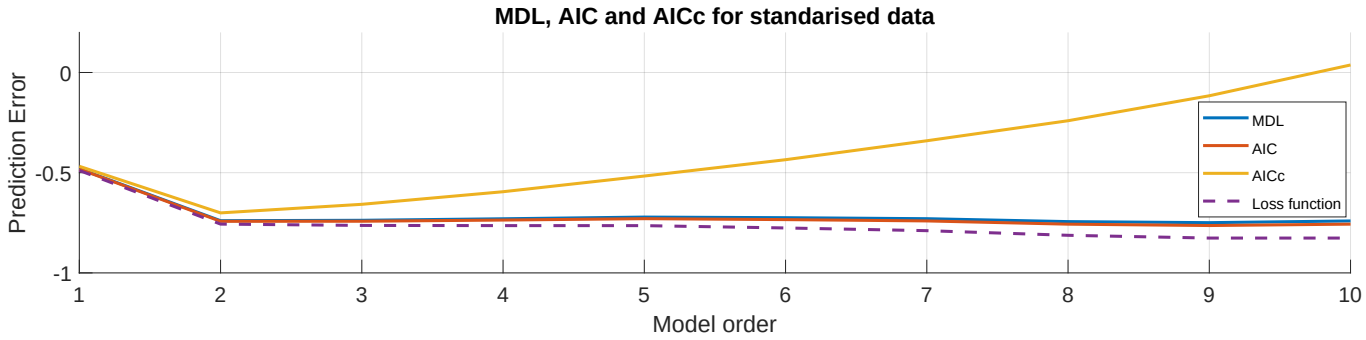


Figure 2.12: MDL, AIC and AICc criteria for model order determination

This can be implemented in MATLAB by using the command `predict(model, sunspot, prediction horizon)` by constructing the model according to Yule-Walker method. The predicted series for AR(1), AR(2) and AR(10) were plotted against the original sunspot series in Fig. 2.13 for different prediction horizons. The graphs were zoomed-in to make differences more visible. Additionally Fig.2.14 shows a plot of mean squared errors of prediction (MSE) versus prediction horizon values,i.e. $MSE = \frac{1}{N}\sum_{n=0}^{N-1}(\hat{x}[n] - x[n])^2$ where N is the sample size.

Figures 2.13 and 2.14 show that AR(1), AR(2) and AR(10) models have a low prediction error for a prediction horizon of 1 year. For m=1, AR(1) model has the highest error, while AR(2) and AR(10) have very similar error. However, for all three models, the greater the prediction horizon, the higher the prediction error(higher mean square error), due to amplitude and phase deviations from the original series. The model of order 10 has the lowest prediction error for all horizons and especially for m=10, it is still very close to the original series. All models however, for all prediction horizons, manage to capture the sinusoidal behavior of the series.

AR(1) model in general does not have sufficient degrees of freedom and does not adequately capture the full underlying structure of the data (underfitting), since coefficients that would appear in a correctly specified model are missing. This is why AR(1) tends to have a poor predictive performance.

We have seen in the previous parts that as model order increases above the optimal one, the model error increases due to deviations of autoregressive coefficients and variance estimate of input white noise from their true values (since it contains more parameters than can be justified by the data). As a consequence, over-modelling leads to spectral line splitting , i.e. creation of additional peaks in the frequency plot. On the other hand, Figure 2.12 shows that the prediction error function is a monotonically decreasing function with model order, thus the higher the order the lower the prediction error. These results indicate the trade-off between model error and prediction error.

We have also seen that AR(2) has the correct model order, therefore the lowest model error, while at the same time it has a reliable prediction performance for relatively small prediction horizons (m¡5). For larger prediction horizons a large model order should be used.

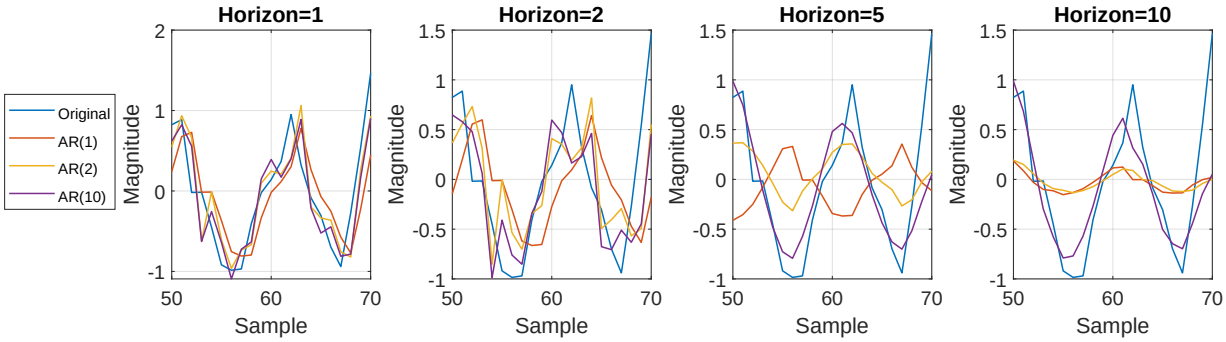


Figure 2.13: Predicted series using AR(1), AR(2) and AR(10) for different prediction horizons. Original series also plotted for comparison. To make differences clear, only an arbitrary segment of series is shown.

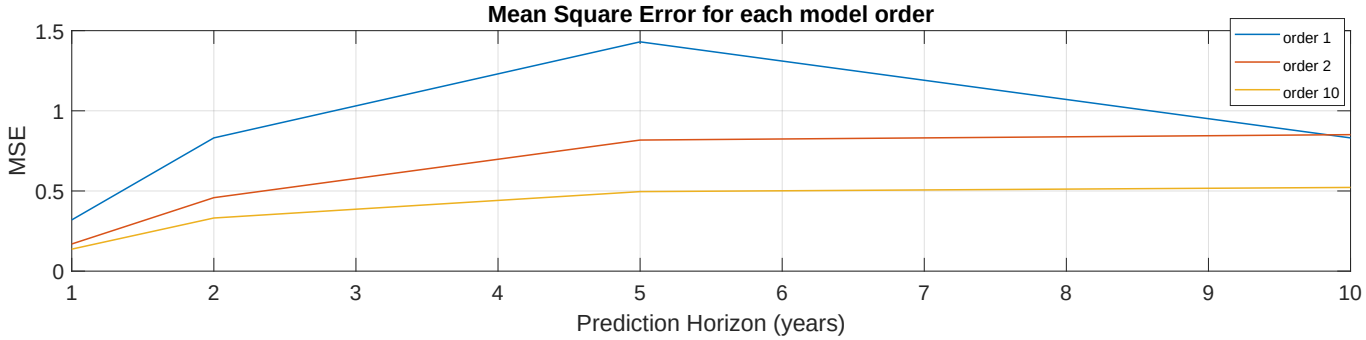


Figure 2.14: MSE error of predicted series against prediction horizons for different model orders.

2.4 Cramer-Rao Lower Bound (CRLB)

2.4.1 - Determining correct order of AR model

The correct model of AR model for NASDAQ financial index's end-of-day prices is determined by utilising both PACF and information theoretic criteria (AIC, MDL and AICc). Results are plotted in figure 2.15.

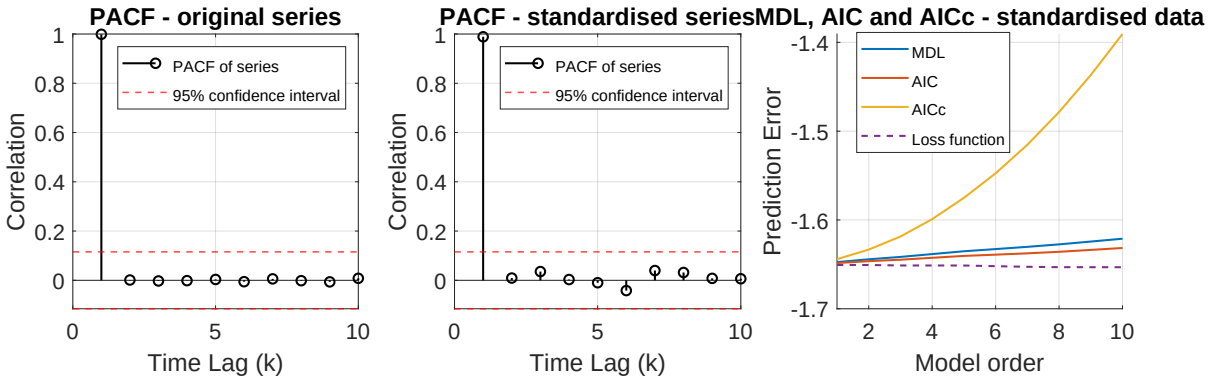


Figure 2.15: (a) PACF estimate for original series, (b) PACF estimate for standardised series and (c) MDL, AIC and AICc criteria for model order determination (using standardised data)

The partial autocorrelation (PAC) coefficients would theoretically be non-zero for $k \leq p$ and zero for $k > p$. Using 95% confidence interval, we see that after time lag of 1 the PACF falls within the confidence interval so it is assumed to be zero. This indicates that the model order is 1. Additionally, plots of MDL, AIC and AICc criteria have a minimum value at order value of 1, which indicate as well that the optimal order is 1.

2.4.2 - Obtaining elements of Fisher Information matrix $\mathbf{I}(\theta)$

The elements of the Fisher's Information matrix can be approximated as:

$$[\mathbf{I}(\boldsymbol{\theta})]_{ij} = \frac{N}{2}\int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{\partial ln\left[P_X(f;\boldsymbol{\theta})\right]}{\partial\theta_i}\frac{\partial ln\left[P_X(f;\boldsymbol{\theta})\right]}{\partial\theta_j}df \tag{24}$$

We are given that AR model is 1 (p=1), $\boldsymbol{\theta} = [\alpha_1, \sigma^2]$, $[\mathbf{I}(\boldsymbol{\theta})]_{11} = \frac{Nr_{xx}(0)}{\sigma^2}$ and $[\mathbf{I}(\boldsymbol{\theta})]_{12} = [\mathbf{I}(\boldsymbol{\theta})]_{21} = 0$.
The value of $[\mathbf{I}(\boldsymbol{\theta})]_{22}$ is given by:

$$[\mathbf{I}(\boldsymbol{\theta})]_{22} = \frac{N}{2}\int_{-\frac{1}{2}}^{\frac{1}{2}}\left[\frac{\partial ln\left[P_X(f;\boldsymbol{\theta})\right]}{\partial\sigma^2}\right]^2 df = \frac{N}{2}\int_{-\frac{1}{2}}^{\frac{1}{2}}\left[\frac{1}{\sigma^2}\right]^2 df = \frac{N}{2}\int_{-\frac{1}{2}}^{\frac{1}{2}}\frac{1}{\sigma^4}df = \frac{N}{2\sigma^4}\left[f\right]_{-\frac{1}{2}}^{+\frac{1}{2}} = \frac{N}{2\sigma^4}$$

Therefore, the Fisher's information matrix is equal to:

$$[\mathbf{I}(\boldsymbol{\theta})] = \begin{bmatrix} \frac{Nr_{xx}(0)}{\sigma^2} & 0 \\ 0 & \frac{N}{2\sigma^4} \end{bmatrix} \tag{25}$$

where $r_{xx}(0)$ is the autocorrelation of process at zero lag (Signal Power or signal variance) , σ^2 is the variance of input white noise and N the sample size.

2.4.3 - Computation of CRLB of $\hat{\sigma}^2$ and $\hat{\alpha_1}$ estimators.

The CRLB of an unbiased estimator of a process parameter, i.e. σ^2 and α_1 for this case, is found as the $(ii)^{th}$ element of the inverse of the Fisher's Information matrix. In other words:

$$Var(\hat{\theta}_i) \geq \left[\mathbf{I}^{-1}(\boldsymbol{\theta})\right]_{ii} \tag{26}$$

The inverse of Fisher Information matrix is given by:

$$\mathbf{I}^{-1}(\boldsymbol{\theta}) = \frac{1}{\frac{N^2 r_{xx}(0)}{2\sigma^6}}\begin{bmatrix} \frac{N}{2\sigma^4} & 0 \\ 0 & \frac{Nr_{xx}(0)}{\sigma^2} \end{bmatrix} = \begin{bmatrix} \frac{\sigma^2}{Nr_{xx}(0)} & 0 \\ 0 & \frac{2\sigma^4}{N} \end{bmatrix}$$

This gives the CRLB for each estimator:

- $Var(\hat{\sigma}^2) = [\mathbf{I}^{-1}(\boldsymbol{\theta})]_{22} \geq \frac{2\sigma^4}{N}$.
- $Var(\hat{\alpha}_1) = [\mathbf{I}^{-1}(\boldsymbol{\theta})]_{11} \geq \frac{\sigma^2}{Nr_{xx}(0)} = \frac{1}{N}(1-\alpha_1^2)$.

The fact that $\frac{\sigma^2}{Nr_{xx}(0)} = \frac{1}{N}(1-\alpha_1^2)$ can be derived in the following way. We know that an AR(1) process can be expressed as: $x[n] = \alpha_1 x[n-1] + w[n]$, for $w[n] \sim \mathcal{N}(0, \sigma^2)$. Multiplying both sides by x[n-l] and then taking the expectation operation (we assume stationary process x[n]):

$$\begin{aligned} x[n]x[n-l] &= \alpha_1 x[n-l]x[n-1] + w[n]x[n-l] \\ E\left\{x[n]x[n-l]\right\} &= \alpha_1 E\left\{x[n-l]x[n-1]\right\} + E\left\{w[n]x[n-l]\right\} \\ r_{xx}(l) &= \alpha_1 r_{xx}(l-1) + E\left\{w[n]x[n-l]\right\} \end{aligned} \tag{27}$$

where $E\left\{w[n]x[n-l]\right\} = \begin{cases} \sigma^2 & , \; for \; l = 0 \\ 0 & , \; for \; l > 0 \end{cases}$.

For l>0, and specifically l=1, we can obtain that $r_{xx}(1) = \alpha_1 r_{xx}(0)$. For l=0, we obtain that $r_{xx}(0) = \alpha_1 r_{xx}(-1) + \sigma^2 = \alpha_1 r_{xx}(1) + \sigma^2$. This gives:

$$
\begin{aligned}
r_{xx}(0) &= \alpha_1^2 r_{xx}(0) + \sigma^2 \\
r_{xx}(0)[1-\alpha_1^2] &= \sigma^2 \\
r_{xx}(0) &= \frac{\sigma^2}{[1-\alpha_1^2]}
\end{aligned}
\tag{28}
$$

Therefore, $\frac{\sigma^2}{N r_{xx}(0)} = \frac{\sigma^2}{\frac{N\sigma^2}{1-\alpha_1^2}} = \frac{1-\alpha_1^2}{N}$.

The CRLB for both $\hat{\sigma}^2$ and $\hat{a}_1$, for varying sample size, N, and true variance of input noise, σ^2, in the range [1:50:1001], was obtained and then plotted in Fig. 2.16 using MATLAB function `heatmap`. Standardised data was used, since assymptotic CRLB is based on the undelying assumption that the process x[n] has zero mean.

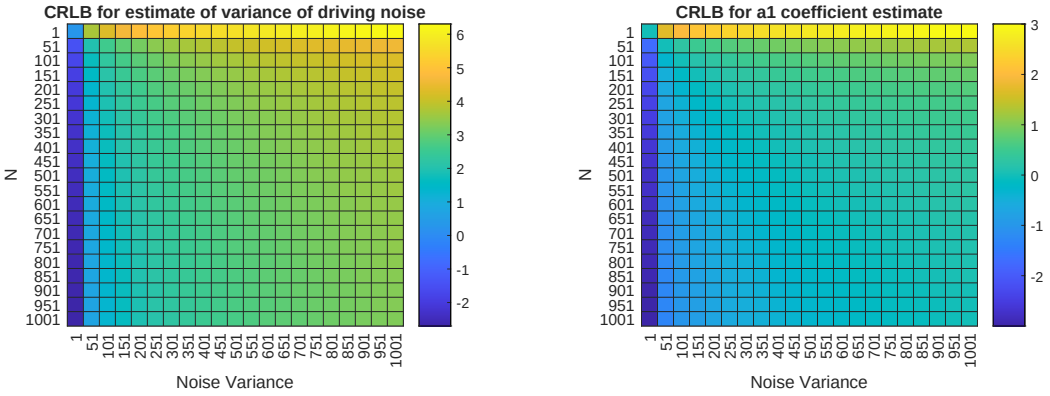


Figure 2.16: (a) Heatmap of CRLB for $\hat{\sigma}^2$ and (b) Heatmap of CRLB for $\hat{a}_1$. Note that the color-scale values of $log_{10}(CRLB)$, to allow us to better differentiate data.

The two plots of Fig.2.16 show that CRLB of both estimates have a minimum at N=1101 and σ^2=1, however CRLB for variance estimate is higher. Similarly, the maximum CRLB is at N=1 and σ^2=1001. As expected from the mathematical descriptions of the CRLB for both estimators, the higher the variance of the driving noise, the higher the CRLB value. Note that the $log_{10}(CRLB)$ was used for the plot, due to 10^6 order difference between minimum and maximum values. Displaying the log of the magnitude reduces the dynamic range, allowing CRLB values to be visualised more clearly.

By considering the AR(1) model of the financial dataset, we can obtain an estimate for $\hat{\alpha}_1 = -0.9989$, driving input noise variance estimate $\hat{\sigma}^2 = 9696.1$ and $r_{xx}(0) = 4.344x10^6$. Thus the minimum possible variance of α_1 estimator (CRLB) is: $min(Var(\hat{\alpha}_1)) = \frac{9696.1}{(924)(4.344x10^6)} \approx 2.415x10^{-6}$. If we use the equation involving α_1 coefficient from AR(1) model of original series, i.e. $\alpha_1 = 0.9989$, then $min(Var(\hat{\alpha}_1)) \approx 2.380x10^{-6}$. When standardised series is used the estimated noise variance is approximately 0.0224, so the the heatmaps cannot be used. Note that assymptotic CRLB assumes x[n] has zero mean, so standardised series is more accurate.

As AR(1) model coefficient α_1 approaches unity, then variance of $\hat{\alpha}_1$ approaches zero. Therefore, this means that if the estimator used has the optimal performance with variance equal to CRLB, then coefficient α_1 can be determined exactly with zero variance, and thus infinite accuracy. This shows that filter parameters can be more accurately estimated when magnitude of α_1 approaches unity rather than zero.

Since AR(1) model is given by $x[n] = \alpha_1 x[n-1] + w[n]$, then the filter's frequency response $H(f) = \frac{1}{1-\alpha_1 e^{-j2\pi fn}}$ for $0 \leq f \leq 1/2$ due to periodicity of frequency response and non-existence of negative frequencies. As a result, the pole of filter is equal to α_1. We know that the Power Spectral Density of the AR(1) model is given by:

$$
P_X(f;\boldsymbol{\theta}) = |H(f)|^2 P_{WGN}(f) = \frac{\sigma^2}{|1-\alpha_1 e^{-j2\pi fn}|} \tag{29}
$$

Therefore, as α_1 approaches unity, then the peak in PSD of process becomes sharper. This indicates that if the PSD has sharp peaks then process parameters are more accurately estimated. In other words, the higher the curvature or sharpness of the PSD, the higher accuracy of parameter estimators. Curvature is defined as the negative second derivative of the logarithm of the PSD.

2.4.4 - CRLB of Power Spectral Density's (PSD) estimate

From the CRLB it can be shown that

$$
Var(\hat{P}_X(f;\boldsymbol{\theta})) \geq \frac{\partial P_X(f;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}^T \boldsymbol{I}^{-1}(\boldsymbol{\theta}) \frac{\partial P_X(f;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \;, \quad where \quad \frac{\partial P_X(f;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \left[\frac{\partial P_X(f;\boldsymbol{\theta})}{\partial \alpha_1}, \frac{\partial P_X(f;\boldsymbol{\theta})}{\partial \sigma^2}\right]^T
$$

By letting $A(f) = 1 - \alpha_1 e^{-j2\pi f}$, we can write $P_X(f;\boldsymbol{\theta}) = \frac{\sigma^2}{|A(f)|^2}$. Therefore we compute first

$$
\frac{\partial P_X(f;\boldsymbol{\theta})}{\partial \alpha_1} = \frac{-\sigma^2}{(A(f)A^*(f))^{-2}} \left[A(f)\frac{\partial A^*(f)}{\partial \alpha_1} + A^*(f)\frac{\partial A(f)}{\partial \alpha_1}\right] = \frac{\sigma^2}{|A(f)|^4}\left[A(f)e^{j2\pi f} + A^*(f)e^{-j2\pi f}\right] \tag{30}
$$

and then

$$
\frac{\partial P_X(f;\boldsymbol{\theta})}{\partial \sigma^2} = \frac{1}{|A(f)|^2} \tag{31}
$$

Using these two results, we can compute the bound for Periodogram, using also the expression for inverse Fisher Information matrix ($\boldsymbol{I}^{-1}(\boldsymbol{\theta})$) from part 2.4.3.

$$\begin{aligned} min(Var(\hat{P}_X(f;\boldsymbol{\theta}))) &= \left[\frac{\partial P_X(f;\boldsymbol{\theta})}{\partial \alpha_1}, \frac{\partial P_X(f;\boldsymbol{\theta})}{\partial \sigma^2}\right] \begin{bmatrix} \frac{\sigma^2}{N r_{xx}(0)} & 0 \\ 0 & \frac{2\sigma^4}{N} \end{bmatrix} \begin{bmatrix} \frac{\partial P_X(f;\boldsymbol{\theta})}{\partial \alpha_1} \\ \frac{\partial P_X(f;\boldsymbol{\theta})}{\partial \sigma^2} \end{bmatrix} \\ &= \frac{\sigma^2}{N r_{xx}(0)} \left[\frac{\partial P_X(f;\boldsymbol{\theta})}{\partial \alpha_1}\right]^2 + \frac{2\sigma^4}{N} \left[\frac{\partial P_X(f;\boldsymbol{\theta})}{\partial \sigma^2}\right]^2 \\ &= \frac{\sigma^4}{N\left|A(f)\right|^4} \left[\frac{\sigma^2}{r_{xx}(0)\left|A(f)\right|^4}(A(f)e^{j2\pi f} + A^*(f)e^{-j2\pi f})^2 + 2\right]. \end{aligned} \tag{32}$$

2.5 Real world signals: ECG from iAmp experiment

2.5.1 - Heart rate probability density estimate (PDE)

The RRI signal rr[n] from unconstrained breathing is obtained using `iAmp_import_v40` function and then extracted by identifying the start and end times of this specific trial. Using this segment, the heart rate is calculated using

$$h[n] = \frac{60}{rr[n]} \tag{33}$$

For smoother estimates of heart rate, the following method is used, first using $\alpha = 1$ and then using $\alpha = 0.6$.

$$\hat{h}[1] = \frac{1}{10}\sum_{i=1}^{10} \alpha h[i], \hat{h}[2] = \frac{1}{10}\sum_{i=11}^{20} \alpha h[i], ... \tag{34}$$

The probability density estimates (PDE) of the original and averaged heart rates, for the two values of α, are computed using the MATLAB function `ksdensity`, which returns PDE based on a a normal kernel function. This allows the three PDE's to be plotted on the same figure (Fig. 2.17). Equivalently the `pdf` function designed in section 1.3, could be utilised for these estimates. It is assumed that the RRI signal is stationary.

The mean value of original heart rates and averaged heart rates, is obtained from the maximum values of the PDE plots. Original heart rate PDE has the approximately the same peak as the averaged heart rate with $\alpha = 1$, at 58.1 beats/min and 59.2 beats/min respectively. On the other hand, the averaged heart rate with $\alpha = 0.6$ has a peak at 35.5 beats/min. The original heart rates have a PDE that is positively skewed, however the averaged heart with $\alpha = 1$, is effectively a smoother version of this estimate, with a lower standard deviation (4.62 instead of 5.14).

The decrease in standard deviation can be explained by considering what averaging process does. Averaging is essentially truncation of signal into windows of length 10 for which the mean value is obtained. By assuming that heart rate values are independent and the variance of original heart rates (h[n]) is σ^2, then the variance of average heart rates will be $\frac{\alpha^2}{M}\sigma^2$, where M is the number of samples being averaged. Therefore, in the case that $alpha = 1$ and L=100, we expect the variance of averaged heart rates to be 10 times less than the variance of original heart rates. In reality, however, this factor is less than 10, possibly due to the fact that independence assumption is not valid and therefore variance of averaged heart rates has an additional term involving covariance of the heart rates.

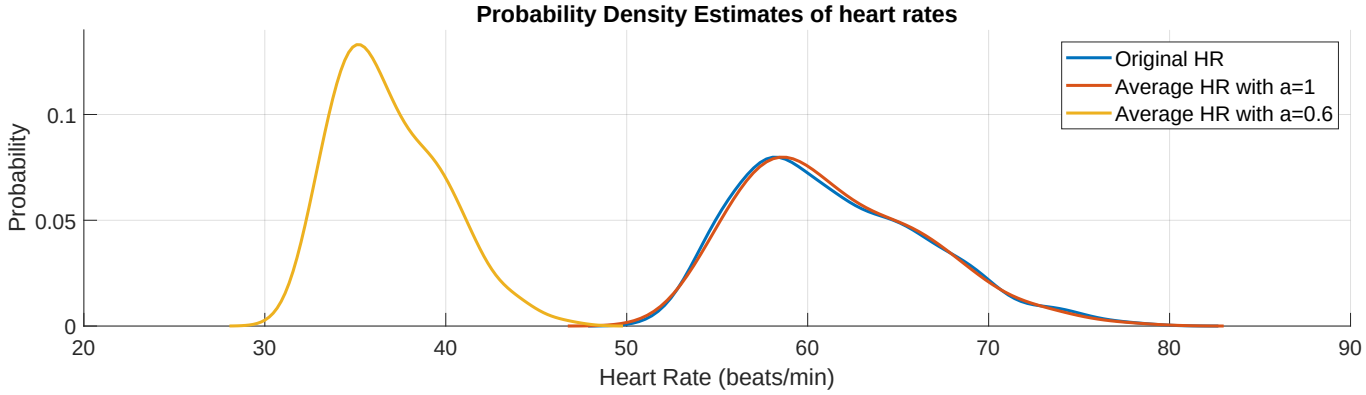


Figure 2.17: Probability density estimates (PDE) for original and averaged heart rates

Additionally, the value of constant α affects the shape (width and peak-magnitude) and position of PDE. Effectively, all original heart rate values are scaled by the value of the constant, and specifically: when $\alpha > 1$ the mean value of heart rates shifts to the right of the original series and when $\alpha < 1$ the mean shifts to the left. Moreover, we have seen that variance of heart rate estimates is proportional to α^2, therefore when $\alpha = 0.6$, the variance decreases even more causing the width of PDE to decrease. Amplitude of peak increases as well in order for probability to be conserved (Unity Area). Note that any deviations in the mean in the case of $\alpha = 1$ are due to the fact that different sample sizes are used for PDE computation.

2.5.2 - AR Modelling of heart rate

The RRI data for each trial was firstly centered (zero-mean) using command `detrend` and then used for computation of autocorrelation function estimate, shown in Figure 2.18. The shape of ACF can be used to determine if RRI data is an AR or MA process.

Table 3: Characteristics of AR(p) and MA(p) process

	ACF	PACF
AR(p)	Geometric Decay	Significant until p lags
MA(q)	Significant until q lags	Geometric Decay
ARMA(p,q)	Geometric Decay	Geometric Decay

From theory, the ACF for an AR process follows a geometric decay, whereas the ACF for an MA(p) process is non-zero for first p lags and zero for all lags¿p. Additionally, the exact opposite occurs when PACF is considered. This is summarised in Table 3. The characteristic for ACF of AR and MA processes can be explained by considering the following equations, obtained from the definitions of the two processes:

- For AR(p) process:
$$\sum_{k=0}^{p} \alpha_k r_{xx}(l-k) = \begin{cases} \sigma^2 & , \ for\ l=0 \\ 0 & , \ for\ l>0 \end{cases} \tag{35}$$
- For MA(q) process
$$r_{xx}(l) = \sum_{k=0}^{p} \alpha_k E\left\{x[n-l]w[n-k]\right\} \tag{36}$$

  which for l=0, $r_{xx}(0) = \sigma^2 \sum_{k=0}^{q} \alpha_k^2$ and for l>0, $r_{xx}(l) = \sum_{k=0}^{q} \alpha_k \sum_{m=0}^{q} \alpha_m E\left\{w[n-m]w[n-k]\right\}$, which is non-zero for $l+m=k$ (even for $l=q$) and zero for l>q.

Figure 2.18 shows that ACF of trial 1 is a decaying sinosoid and ACF of trial 2 is approximately an exponential decay. ACF of trial 3, is not exactly a decaying function but a sinusoidal function; however the values of ACF are not significant for any lag, i.e. much less than 95% confidence interval bound. In addition, Figure 2.19 shows that PACF is significant, i.e. non-zero, up to specific lag for all trials and then reduces to zero (within 95% confidence interval). Therefore, this shows that RRI data from the three trials is an AR process and not an MA process.

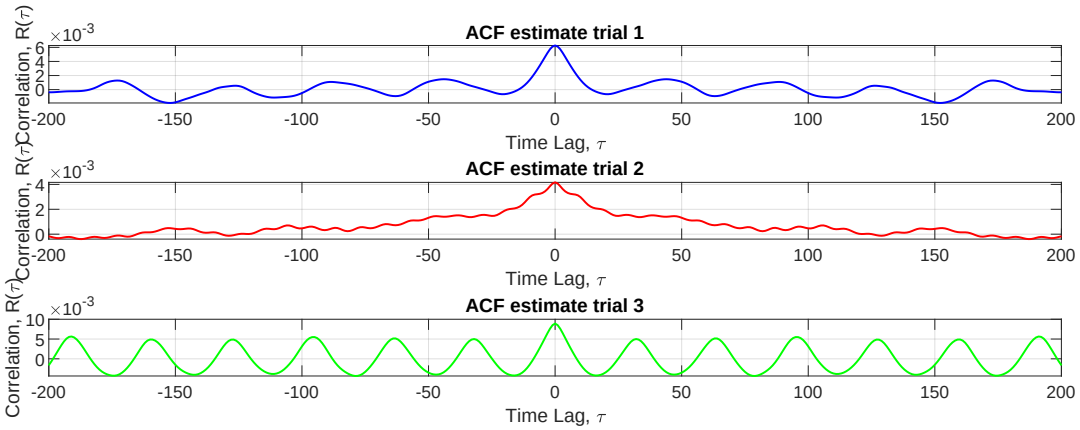


Figure 2.18: Autocorrelation sequence for the RRI data for the three trials

Using both PACF and information theoretic criteria (AIC, MDL and AICc), we can determine the optimal AR model order for each trial. The partial autocorrelation (PAC) coefficients would theoretically be non-zero for $k \leq p$ and zero for $k > p$. Using 95% confidence interval, as shown in Fig 2.19, we detect the time lag after which PACF falls within the confidence interval so it is assumed to be zero. Additionally, order is identified on the location of minimum values on plots of MDL, AIC and AICc criteria (Fig.2.20). Results are presented in Table 4.

Table 4: Predicted AR model orders according to PACF, MDL, AIC and AICc criteria for RRI data of each trial.

	PACF	MDL	AIC	AICc
Trial 1	3	5	5	3
Trial 2	3	9	10	3
Trial 3	2	2	2	2

It should be noted that due to small sample size, the AICc criterion gives a clearer indication of the model order. Moreover, trial 3 can be more accurately modelled as an AR(2) process, since all criteria agree. Finally, trials 1 and 2 can be modelled as AR(3) processes. If all trials should be modelled by the same AR process, then the order would be 3.

3 Spectral estimation and modelling

The Power Spectral Density of an ergodic (and thus stationary) stochastic process is given by the absolute value of the Fourier Transform of its ACF, given by

$$P_X(f) = \left| \sum_{r=-\infty}^{\infty} R_X(\tau) e^{-j2\pi f \tau} \right|, f \in [0,1] \tag{37}$$

where R_X is the ACF of the process X_n and f is the normalised frequency. However, in practice only a limited number of samples of the stochastic process is available therefore a PSD estimate is used, based on Fast Fourier Transform, known as periodogram, defined in terms of N dicrete normalised frequencies.

$$\hat{P}_X(f) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x[n] e^{-j2\pi f \frac{n}{N}} \right|^2 \ , \ for\ f = 0, 1/N, ..., (N-1)/N \tag{38}$$

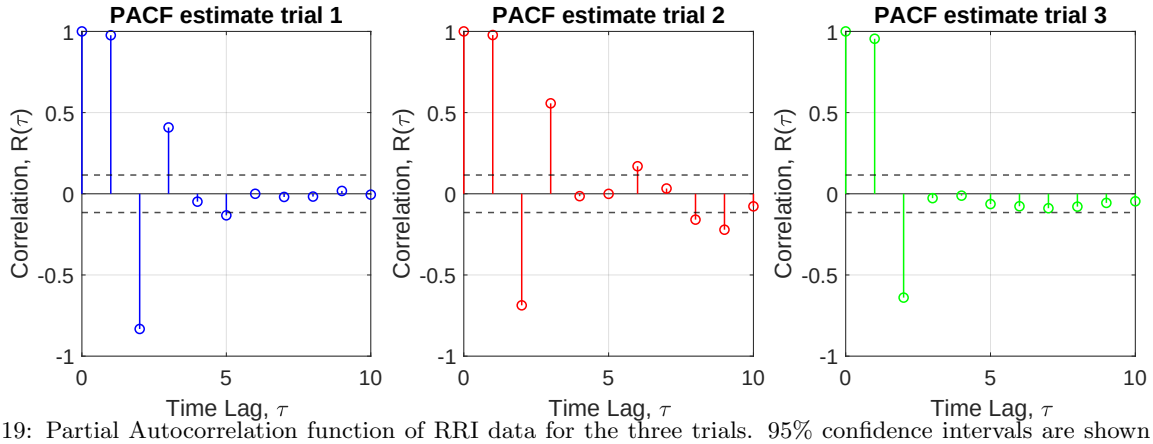


Figure 2.19: Partial Autocorrelation function of RRI data for the three trials. 95% confidence intervals are shown in dashed

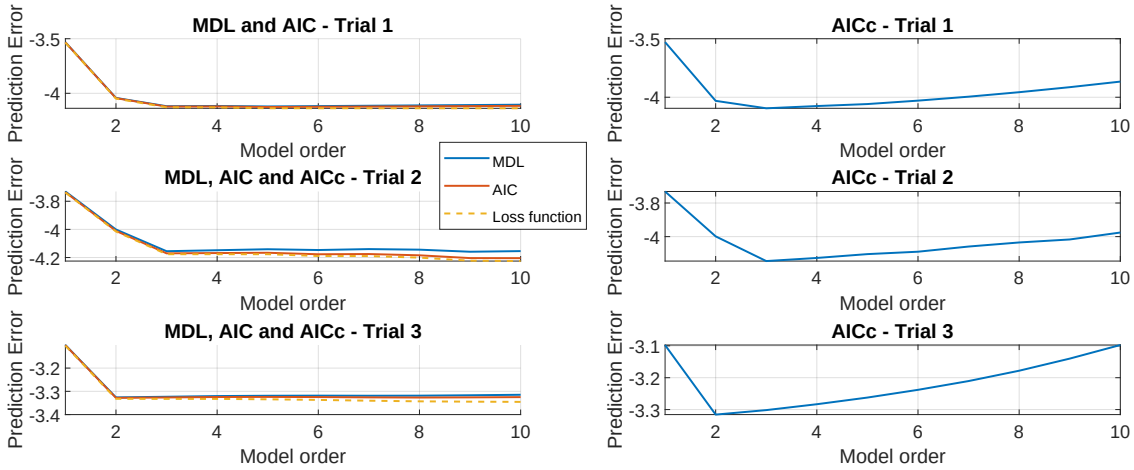


Figure 2.20: MDL, AIC and AICc criteria for each trial

Periodogram estimate given by equation 38 is implemented in MATLAB using created script `pgm`, which produces a periodogram with the same length as the input process. The created function was tested on an N-sample realisation of WGN, for different sample sizes, shown in figure 3.1.

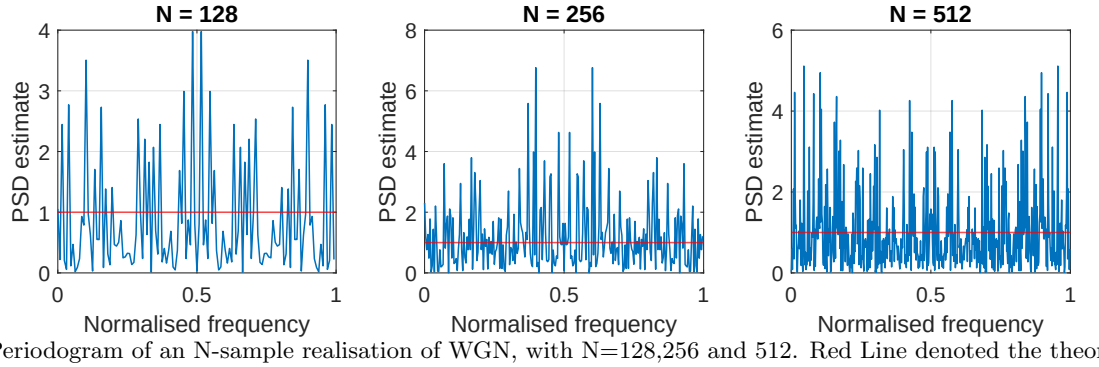


Figure 3.1: Periodogram of an N-sample realisation of WGN, with N=128,256 and 512. Red Line denoted the theoretical PSD value.

Figure 3.1 shows that the periodogram is a periodic function with period 1 unit of normalised frequency, corresponding to 2π radians. This is why the periodogram is symmetric about normalised frequency of 0.5.

The ACF for WGN ($w[n] \sim \mathcal{N}(0, \sigma^2)$, is equal to $R_{WGN}(\tau) = \sigma^2\delta(\tau)$. Since PSD is equivalent to the Fourier Transform of ACF, then $P_{WGN} = \sigma^2$. The WGN used for figure 3.1, has unit-variance therefore its PSD is expected to have a constant value of unity for all frequencies (constant power).

In practice, however, periodogram values are not equal to unity for all frequencies as expected but instead have different magnitudes for different frequencies as Fig. 3.1 shows. This is mainly due to the finite length realisations of WGN used. Ideally, for a signal having constant PSD across all frequencies, then its length must be infinite, and thus it must have infinite power. Figure 3.1 also shows that the average value of PSD estimate is very close to 1, without being affected by the sample size. Large deviations around the expected value of 1 are due to high variance of estimate (independent of sample size).

The estimate of the PSD at each frequency is a random variable, with an associated probability density function, as well as mean and variance. Periodogram in general is an asymptotically unbiased estimator, which means as $N \to \infty$ its expected value will be equal to the actual PSD value for that frequency. (for WGN, this value will be equal to noise variance σ^2). This can be explained mathematically by deriving the expression for the expected value of the periodogram. This is done by first considering the expected value of the corresponding ACF estimate which is the Inverse Fourier Transform of periodogram used, i.e. $\hat{R}_X(\tau) = \frac{1}{N}\sum_{\tau=0}^{N-1-|\tau|} E\{x(n)x(n+|\tau|)\}$. This shows that the ACF estimate is biased with a triangular (Bartlett) window, $w_B(\tau)$.

Using Fourier Transform of the ACF estimate, we can compute the expected value of periodogram. This can be found to be equal to:

$$E\left\{\hat{P}_X(f)\right\} = \frac{1}{2\pi}P_X(f) * W_B(f) = \frac{1}{2\pi}P_X(f) * \frac{1}{N}\left[\frac{sin(N\pi f)}{sin(\pi f)}\right]^2 \tag{39}$$

The larger the sample size N, the narrower the primary lobe of this function in frequency domain and thus it approaches more and more to a dirac delta function. At $N \to \infty$, convolution of true PSD value at frequency f with a dirac delta centered at f, will give the true PSD value. Therefore

$$\lim_{N\to\infty} Bias(\hat{P}_X) = E\left\{\hat{P}_X(f)\right\} - P_X(f) = P_X(f) - P_X(f) = 0. \tag{40}$$

However, it is not a consistent estimator because its variance does not decrease to zero with increasing sample length N. In fact, variance of the periodogram estimate at a particular frequency is proportional to the square of the true PSD value at that frequency. In other words, according to documentation *Performance of the Periodogram : Statistical Signal Processing (Signal Processing Toolbox) by MathWorks, Inc* :

$$Var(\hat{P}_X(f)) \approx P_X^2(f)\left[1 + \left(\frac{sin(\pi f N)}{N sin(\pi f)}\right)^2\right], \; so \; \lim_{N\to\infty} Var(\hat{P}_X(f)) \approx P_X^2(f) \tag{41}$$

Specifically, in the case of white noise, however, the PSD estimate has special properties. The expected value of periodogram of WGN is

$$E\left\{\hat{P}_X(f)\right\} = \frac{1}{2\pi N}P_X(f) * \left[\frac{sin(N\pi f)}{sin(\pi f)}\right]^2 = \frac{1}{2\pi N}\sum_{m=-\pi}^{\pi} P_X(m)\frac{sin^2(N\pi(f-m))}{sin^2(\pi(f-m))}$$

and since $P_X(f) = \sigma^2$ and Fourier Transform of Bartlett window is an even function, the result of the summation will be the same independently of frequency f, therefore

$$E\left\{\hat{P}_X(f)\right\} = \frac{\sigma^2}{2\pi N}P_X(f) * \left[\frac{sin(N\pi f)}{sin(\pi f)}\right]^2 = \frac{1}{2\pi N}\sum_{m=-\pi}^{\pi} \frac{sin^2(N\pi(f-m))}{sin^2(\pi(f-m))} \tag{42}$$

This was computed using MATLAB (analytical solution is not trivial) and the results for different values of N were plotted in Figure 3.2, which shows that the periodogram for WGN is always an unbiased estimator, for any sample size.

The variance of periodogram for a WGN can be obtained by considering the covariance and second-order moment of the periodogram, which simplify the expression for variance into

$$Var(\hat{P}_X(f)) = P_X^2(f) = \sigma^4 \; , \; for \; any \; N. \tag{43}$$

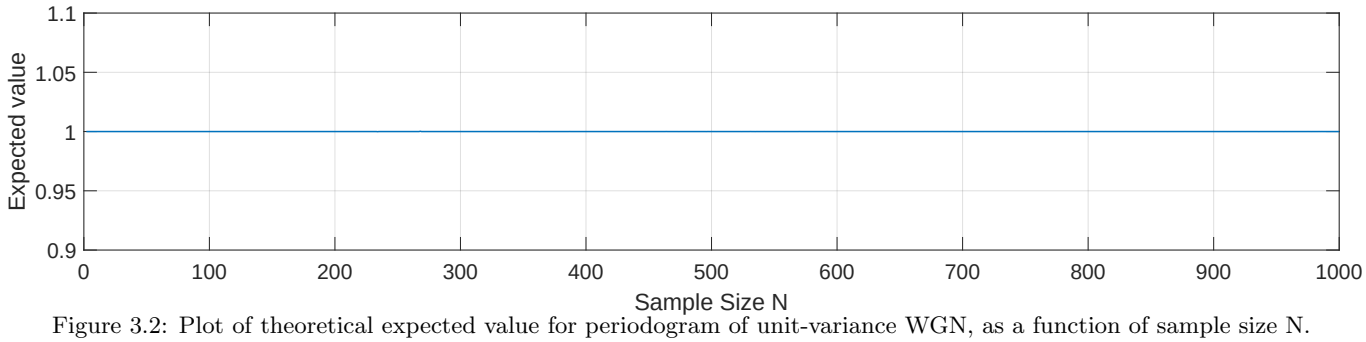


Figure 3.2: Plot of theoretical expected value for periodogram of unit-variance WGN, as a function of sample size N.

3.1 Averaged Periodogram estimates

3.1.1 - Smoothing the PSD estimate

The PSD estimate calculated in previous part can be smoothed by using a zero-phase FIR filter with impulse response given by $h[n] = \frac{1}{5}\sum_{k=0}^{5}\delta[n-k]$. The smoothed PSD estimates for N=128, 256 and 512 are shown in Fig. 3.3. In order to determine the effect of smoothing operation on the PSD estimate, variance of original and smoothed estimate where computed and then then plotted in Fig. 3.4. It should be noted that estimator variance was obtained by computing the sample variance of periodogram, which is only true for this specific case where the values are varying around the expected value of 1. If PSD was not constant for all frequencies, this metric would have been incorrectly used.

Figure 3.3 shows that the smoothed PSD estimate is still periodic with period equal to normalised fequency 1. It is clearly visible that amplitude of variations around the expected value have been reduced. This is confirmed using Fig. 3.4, which shows that the variance of the estimate is greatly reduced.

Specifically, smoothing the PSD estimate ($\tilde{P}_X(f)$) using FIR filter with window length equal to 5, caused the variance of estimate to decrease by a factor of 5. This can be explained by considering the smoothing operation mathematically, i.e. $\tilde{P}_X(f) = \frac{1}{5}\sum_{k=0}^{5}\hat{P}_X(f-k)$. Therefore, by identifying that Periodogram estimates are independent,

$$Var(\tilde{P}_X(f)) = \frac{1}{25}Var\left(\sum_{k=0}^{4}\hat{P}_X(f-\frac{k}{N})\right) = \frac{1}{25}\sum_{k=0}^{4}Var(\hat{P}_X(f-\frac{k}{N})) = \frac{1}{5}Var(\hat{P}_X(f)) \tag{44}$$

The smoothing operation does not introduce bias in the new estimator, which can be explained by computing the expected value of the smoothed PSD estimate:

$$E\left\{\tilde{P}_X(f)\right\} = \frac{1}{5}E\left\{\sum_{k=0}^{4}\hat{P}_X(f-\frac{k}{N})\right\} = \frac{1}{5}\sum_{k=0}^{4}E\left\{\hat{P}_X(f-\frac{k}{N})\right\} = P_X(f) \tag{45}$$

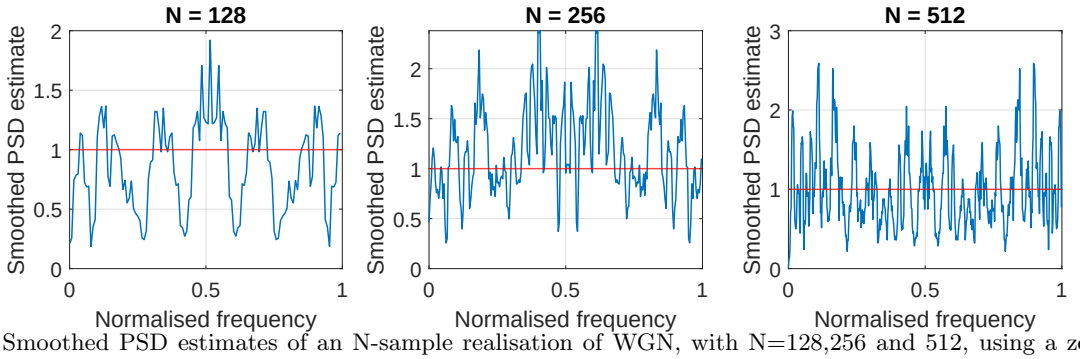


Figure 3.3: Smoothed PSD estimates of an N-sample realisation of WGN, with N=128,256 and 512, using a zero-phase FIR filter. Red Line denoted the theoretical PSD value.

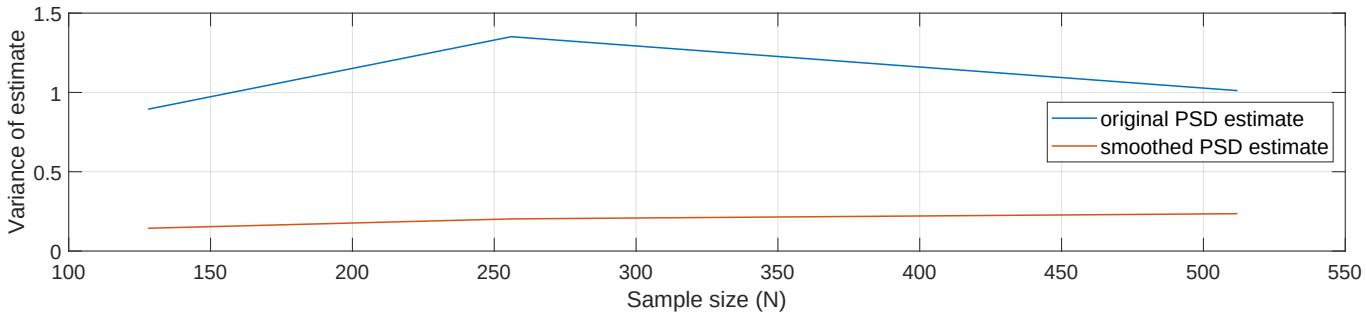


Figure 3.4: Sample variance comparison of original and smoothed PSD estimates.

This is only true when we consider white noise realisations, since the PSD at each frequency is constant (equal to σ^2) and periodogram of WGN is unbiased. Therefore, this improves the apparent PSD estimate. If PSD was not constant at each frequency, then we would expect that smoothing operation would increase bias of estimate.

3.1.2 - Dividing 1024-sample realisation of WGN into eight non-overlapping segments

A 1024-sample sequence of WGN was generated and then subdivided into eight non-overlapping 128-sample segments, for which the PSD estimate was obtained. PSD estimates for each segment are shown in in Figure 3.5.

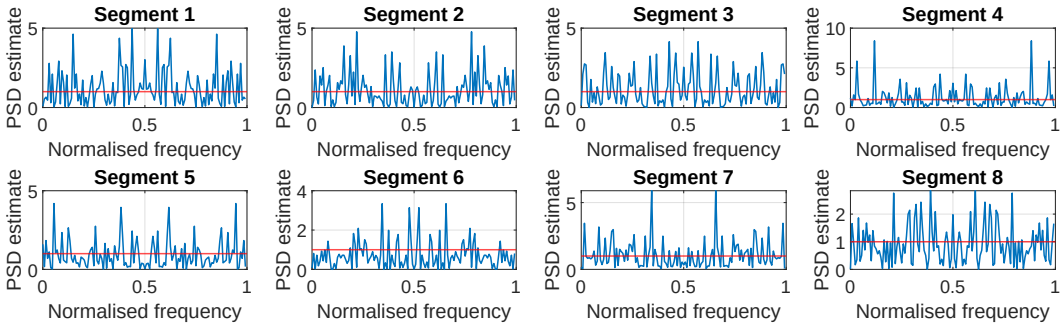


Figure 3.5: PSD estimates for each of the eight non-overlapping 128-sample segments. Red Line denoted the theoretical PSD value.

Variation of PSD estimates of each segment was analysed by computing their sample variance. The values obtained for each segment are tabulated in Table 5.

Table 5: Sample Variance for each 128-sample segment and for 1024-sample realisation.

Segment	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Average**	**1024-sample Realisation**
Variance	0.842	0.934	0.956	0.511	0.850	1.838	0.669	0.834	0.929	0.962
Sample Mean	1.233	1.107	1.162	1.221	0.929	0.713	1.083	0.915	1.045	1.045

Subdividing the 1024-sample realisation into eight segments does not change the variance of each segment, compared to the variance of the original 1024-sample series. Therefore, the average sample variance is the same as before.
Additionally, as it was explained in the previous sections, the expected value of the PSD estimate for each segment is expected to be the same with that of the larger sample-sized realisation.
Therefore, we expect zero bias for all sample sizes. This is verified by computing the sample means for each PSD estimate. The results observed for each segment are very similar to the ones obtained for the PSD estimate for N=128, shown in Fig. 3.1.

3.1.3 - Averaged Periodogram

The values of PSD estimates for the 8 segments were averaged to yield the averaged periodogram. The results of this is displayed in Fig. 3.6.
The method of dividing an divide the an N-sample signal in K intervals of length L=N/K samples, is called the Bartlett's method. This method, in general, aims to reduce the variance of the periodogram in exchange for a reduction of spectral resolution and increase in bias, compared to standard periodograms. This can be shown by analysing the mean (indication of bias) and variance values of the the 8 segments and the averaged periodogram,

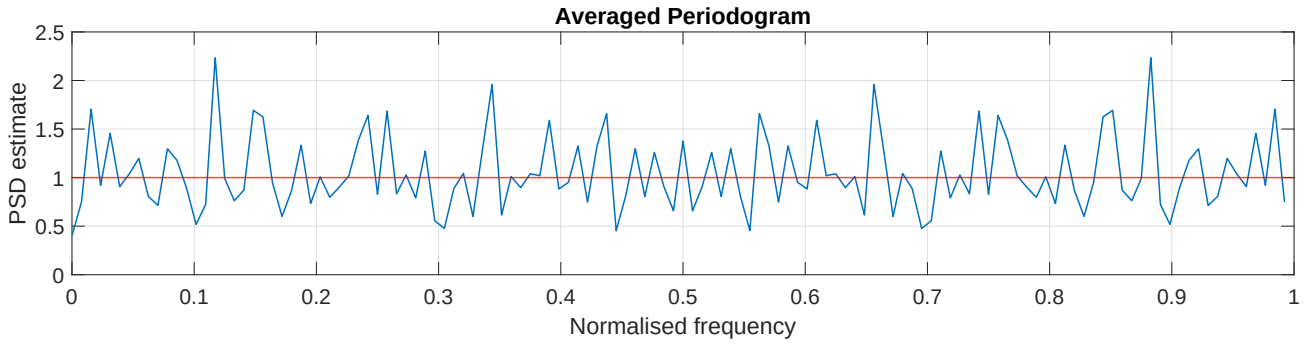


Figure 3.6: Averaged Periodogram, for the 8 segments of Fig. 3.5.

which are recorded in Table 6. It should be noted that sample variance is used for variance computation.

Table 6: Sample variance and mean for each 128-sample segment and for 1024-sample realisation.

	Individual Segments	**Averaged Periodogram**
Mean	1.045	1.046
Variance	0.929	0.122

As expected, the variance of averaged periodogram is lower than the average variance of the 8 segments, about 8 times less. However, sample mean of the average periodogram and thus bias is the same as before, i.e. sample mean is close to 1 and thus bias is close to 0.

We know that the mathematical description of averaged periodogram is

$$\bar{P}_X(f) = \frac{1}{K}\sum_{k=1}^{K}\hat{P}_k(f) = \frac{1}{N}\sum_{i=0}^{K-1}\left|\sum_{n=0}^{L-1}x(n+iL)e^{-j2\pi f\frac{n}{N}}\right|^2 \tag{46}$$

where K is the number of intervals, L is interval length, N is original data size, $\hat{P}_k(f)$ is the periodogram of k^{th} segment. Also, the data in each segment can be expressed as $x_i(n) = x(n+iL)$, for n=0,1,...,L-1 and i=0,1,...,K-1.

Therefore, the expected value of the averaged periodogram for the case of WGN, by assuming independent estimate values, is

$$E\left\{\bar{P}_X(f)\right\} = \frac{1}{K}\sum_{k=1}^{K}\left\{\hat{P}_k(f)\right\} = \frac{1}{K}KP_X(f) = P_X(f) = \sigma^2 = 1 \tag{47}$$

This means that the averaged periodogram is also an unbiased estimator. Additionally the variance of the averaged periodogram is given by:

$$Var(\bar{P}_X(f)) = \frac{1}{K^2}\sum_{k=1}^{K}Var(\hat{P}_k(f)) = \frac{K}{K^2}\sigma^4 = \frac{\sigma^4}{K} \tag{48}$$

This illustrates that variance of the averaged periodogram is inversely proportional to the number of segments (K) and thus as $N \to \infty$, $Var(\bar{P}_X(f)) = 0$, while its bias is always zero for WGN (or asymptotically unbiased in general) which makes it a consistent estimator.

In general, there is bias-variance trade-off for the averaged periodogram, since the higher the number of segments used, the shorter the length of each segment and thus the higher the bias, while at the same time its variance decreases. For the special case of WGN, however, there is no bias-variance trade-off since estimator is always unbiased. The price we pay is a trade-off between good spectral resolution and variance reduction (which also occurs for the general case). Spectral resolution for periodogram is the ability of the estimator to distinguish between closely located peaks in spectrum, and is given as a rule of thumb by $Resolution(\hat{P}_X) = \Delta(f) = 0.89\frac{2\pi}{M}$, where M is data length used (Guido Schuster (2010) *Spectrum estimation using Periodogram, Bartlett and Welch*) This means that the higher the data length (lower K), the higher the variance and the lower the resolution (the lower the resolution, the higher the precision).

3.2 Spectrum of autoregressive processes

A 1064-sample WGN sequence **x** was filtered using an AR filter, with coefficients $a_0 = 1$ and $a_1 = 0.9$, resulting in $y[n] = x[n] - y[n-1]$, where **y** is filter's output sequence. The first 40 output values were removed due to filter's transient effects. The two sequences were plotted in Fig. 3.7.

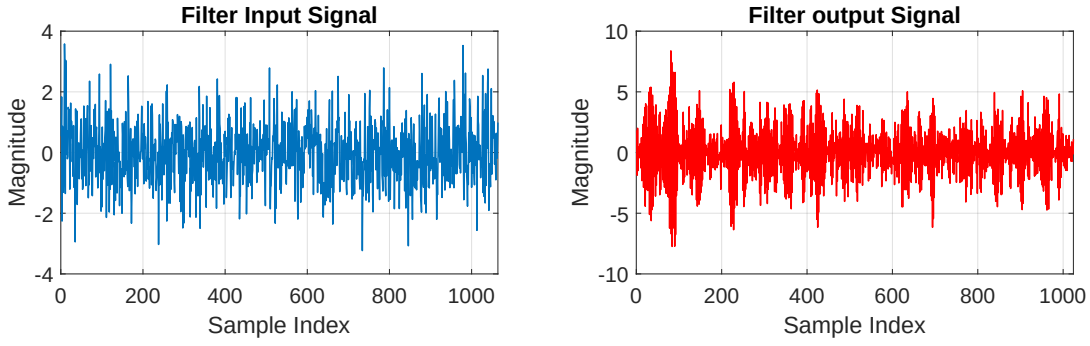


Figure 3.7: Plot of sequences **x** and **y** in the time domain.

The figure above shows that filter's output sequence **y** has higher amplitude variations compared to input WGN.

Output sequence has sample variance of 4.675, while input sequence has 1.022 (since WGN has unit-variance). However, sample means of both sequences is close to zero as expected. Therefore AR filter increases variance of sequence. The fact that output magnitude does not diverge but only varies around the mean value, indicates that the filter is stable. This can be explained by considering the equation of filter's transfer function magnitude:

$$|H(z)| = \frac{1}{|1+0.9z^{-1}|} \ , \ so \ |H(f)| = \frac{1}{|1+0.9e^{-j2\pi f}|} = \frac{1}{|1+0.9e^{-j2\pi f}|} = \frac{1}{\sqrt{1.81+1.8cos(2\pi f)}} \tag{49}$$

Since filter is periodic from $-\pi$ to π, then this means that it is a first-order high-pass filter which supresses low-frequency components of initial signal. As a result, only high-frequency components are presents, resulting in increase in variance. The pole of this filter is located at z=-0.9, so since $|z| < 1$, the filter is stable. The filter also has a zero at z=0, therefore we expect the magnitude of filter's response to be increasing until the pole frequency is reached that will reduce filter's response magnitude to a constant value.

3.2.1 - Filtering WGN by AR filter

The exact PSD of the considered filtered signal $\mathbf{y}$, AR(1) process, is given by:

$$P_Y(f) = \frac{1}{|1+0.9e^{-j2\pi f}|^2} \tag{50}$$

Figure 3.8 shows the plot of the exact PSD (left) of sequence $\mathbf{y}$ in normalised frequency in blue colour.

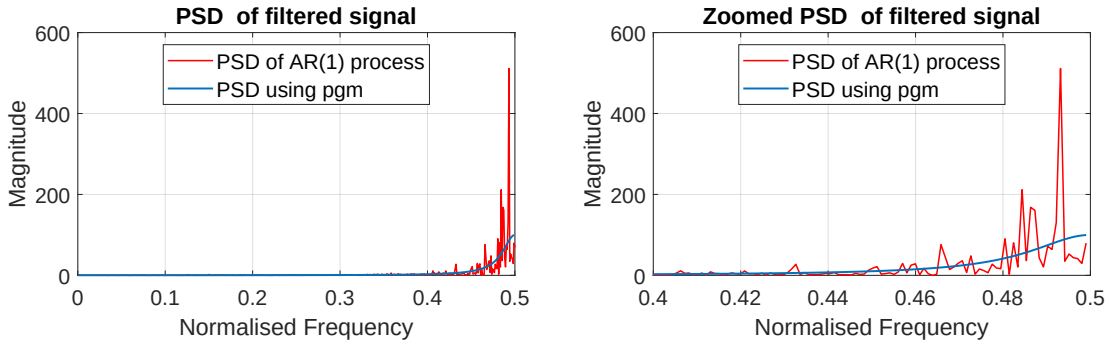


Figure 3.8: (Left) Plot of exact PSD in blue, and PSD estimate of sequence $\mathbf{y}$. (Right) Zoomed in version of left plot in the interval $f = [0.4, 0.5]$.

The PSD of the AR(1) model shows that the PSD magnitude is close to zero for low frequencies and very large for high frequencies, i.e. low frequencies have much less power than higher frequencies. This is the typical shape of a WGN process filtered by a high-pass filter, as shown mathematically in previous part.

For high-pass filters, the cutoff frequency is usually defined as the frequency where the magnitude of the transfer function H(f) is 3 dB the maximum value. The cut-off frequency of the spectral estimate is defined here as the frequency at which the amplitude falls -3dB below the passband gain. In this case, the pass-band gain is 100 (40dB) so we need to compute the frequency at 37dB(magnitude 70.7946). This means we find the value of f, for which $37 = -20log_{10}(|1+0.9e^{-j2\pi f}|)$, which is given by:

$$f = \frac{1}{2\pi} cos^{-1}\left(\frac{10^{-\frac{37}{20}} - 1.81}{1.8}\right) = 0.4892 \tag{51}$$

Since sampling frequency is 1000Hz by default in MATLAB, cutoff frequency in Hz is given by $f.f_s$ which is 489.2Hz.

3.2.2 - Finding PSD estimate of sequence y using `pgm` function

The periodogram for sequence $\mathbf{y}$ is calculated and then plotted in Fig. 3.8 (Left) in red, for comparisons with the exact PSD. The figure shows that the PSD estimate of filtered WGN, has very low amplitude variations around the theoretical value for low normalised frequencies and this variance is increasing with increasing frequency values. The deviations from the theoretical value of PSD, are partly due to the finite number of samples used. Since periodogram is an unbiased estimator of PSD, the lower the sample size the higher the bias of the estimate. When sample size increases to infinity, then the periodogram's expected value will converge to the theoretical value. However, as it was explained in first part of section 3, the variance of periodogram is approximately:

$$Var(\hat{P}_Y(f)) \approx P_Y^2(f)\left[1+\left(\frac{sin(\pi f N)}{N sin(\pi f)}\right)^2\right] \tag{52}$$

The equation above shows that variance of estimator is proportional to the square of the actual PSD value at that frequency. Therefore, as frequency increases, theoretical PSD increases due to high-pass filter effect, thus variance increases. Variance will only be zero if the PSD value is zero. It should be noted that if sample size goes to infinity, variance will be equal to $P_Y^2(f)$, so the variance will never converge to zero, but only its bias.

3.2.3 - Zooming PSD plot of sequence y in interval f=[0.4,0.5].

The plot of PSD values was zoomed for the interval of normalised frequencies between 0.4 and 0.5, as shown in Fig. 3.8 (Right). The results differ from each other due to the underlying rectangular windowing within the periodogram estimator. We can express our sample signal as the output of an infinite sample signal through a rectangular window

$(w_L[n])$ of length N (sample size).
This mathematically is expressed as:

$$y_L[n] = y[n]w_L[n],\ where\ w_L[n] = \begin{cases} 1 & ,\ for\ 0 \leq n \leq N-1 \\ 0 & ,\ otherwise \end{cases} \tag{53}$$

It can be shown that the expected value of periodogram estimate is given by:

$$E\left\{\hat{P}_Y(f)\right\} = \frac{1}{2\pi N} P_Y(f) * |W_L(f)|^2 \tag{54}$$

The squared magnitude of DFT of rectangular window in time domain is a function similar to a sinc function in frequency domain. Therefore, the expected value of periodogram can be expressed as:

$$E\left\{\hat{P}_Y(f)\right\} = \frac{1}{2\pi N} P_Y(f) * \frac{sin^2(\pi f N)}{sin^2(\pi f)} \tag{55}$$

The function given by $\frac{1}{2\pi N}\left(\frac{sin^2(\pi f N)}{sin^2(\pi f)}\right)$ is plotted in MATLAB (Fig.3.9) for continuous frequency domain, instead of discrete, for visualisation of function's shape, for different values of N. As sample size N increases, the function approaches more and more to a dirac delta function, which causes the expected value of periodogram to converge to the theoretical value.

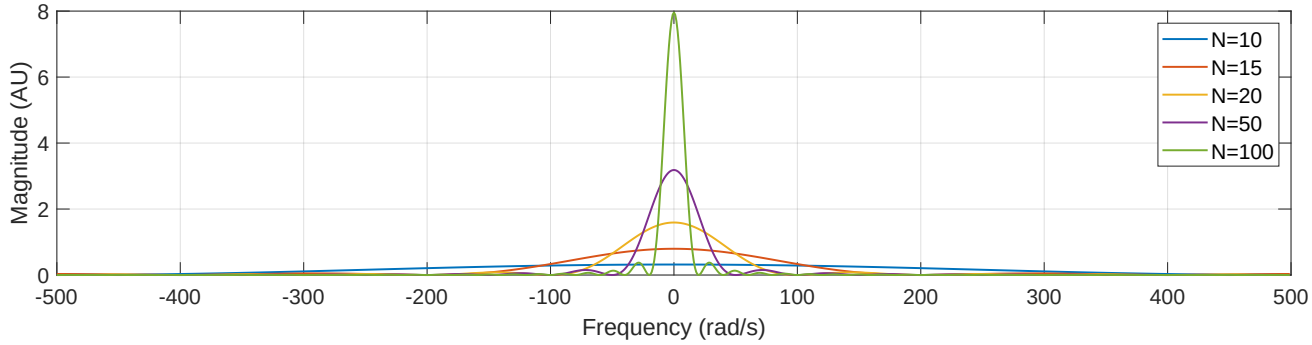


Figure 3.9: Plot of function $\frac{1}{2\pi N}\left(\frac{sin^2(\pi f N)}{sin^2(\pi f)}\right)$ for different N values.

This means that for our case, where N=1024, we expect bias of periodogram to be non-zero, therefore the estimator values are not varying exactly around the theoretical value of PSD.
The variance of the estimator is increasing with higher normalised frequencies. This was explained mathematically in part 3.2.2. Conceptually, however, this can be explained by considering that theoretical PSD values are very close to zero for low frequencies and very high for higher frequencies, due to convolution of theoretical value with the sinc function. It should be noted that the equation of the magnitude of Fourier Transform of rectangular window is involved in the equation of periodogram variance. Therefore, the higher the sample size, the closer the variance gets to the squared PSD value at each frequency.

3.2.4 - Model Based PSD estimate for sequence y

By assuming that the sequence **y** is generated by an AR(1) model, then the calculation of PSD requires the estimation of the two parameters of AR(1) process which are a_1 and input variance σ^2. The estimates can be computed from the ACF estimate $\hat{R}_Y$ using:

$$\hat{\alpha}_1 = -\hat{R}_Y(1)/\hat{R}_Y(0) \tag{56}$$

$$\hat{\sigma}_X^2 = \hat{R}_Y(0) + \hat{\alpha}_1 \hat{R}_Y(1) \tag{57}$$

The ACF estimates were computed using MATLAB command `xcorr` which were then used to give $\hat{\alpha}_1 = 0.913$ and $\hat{\sigma}_X^2 = 0.964$. These estimates are very close to the theoretical values of $\alpha_1 = 0.9$ and $\sigma_X^2 = 1$. The model based PSD according to these estimates was computed using:

$$\hat{P}_y(f) = \frac{\hat{\sigma}_X^2}{|1 + \hat{\alpha}_1 e^{-j2\pi f}|^2} \tag{58}$$

This estimate was then plotted in Fig. 3.10 with the periodogram estimate for comparison.

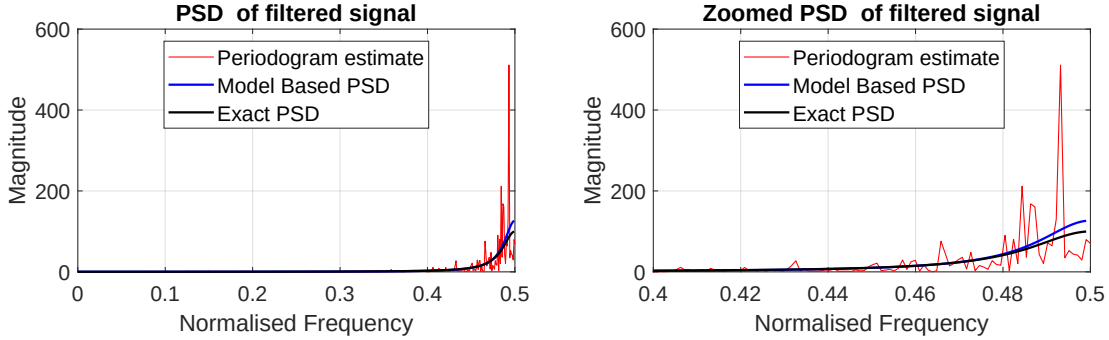


Figure 3.10: Comparison of Model-based PSD estimate with Periodogram estimate.

Fig. 3.10 shows that Model-based PSD estimate is a better estimate for signal's PSD compared to the periodogram, due to absence of spikes and to the fact that its values are close to the exact PSD even for high frequencies. The model-based PSD estimates are less biased and possess a lower variance than periodogram, only in the case that the assumed model order is correct. Deviations of model-based PSD from exact PSD is due to variance of the estimator. In general there are two types of PSD estimation methods:

- Parametric methods
  These are methods based on models of a signal, such as AR models, which requires estimation of the model parameters.

- Nonparametric methods
  These are methods, such as the periodogram, based on FFT (or DFT) of on either windowed signal or windowed ACF, which do not require prior knowledge about the process.

In this case, since the correct order of model is known, the model-based PSD estimator is more accurate than the periodogram estimate. However, if model order is not known and a wrong order is used, model-based estimates will not correctly reflect the behavior of the system that generates the time series, and thus it will not be a reliable estimate. In such case, where model order assumption is not valid, the periodogram estimate, despite the high variance, would be more reliable. The correct model order can be determined by criteria specified in section 2, which require additional computations.
Non-parametric methods such as the periodogram estimate have the limitation of being dependent on data windowing, resulting in distortion of the resulting PSDs due to window effects (sinc function convolution), leading to low resolution. The key benefit of nonparametric methods is the robustness—the estimated PSDs do not contain spurious frequency peaks. In contrast, model-based estimates (parametric methods) do not use data windowing but depend on the assumption that the sequence fits a particular model. However, parametric methods may introduce false spectral peaks due to incorrect model order, which does not occur int the case of periodogram, thus this makes periodogram estimate more robust.

3.2.5 - PSD estimates for sunspot series

The procedure above was repeated for both original sunspot time series and mean-center version, for which periodogram and model-based PSD were computed. The model-based PSD estimator was obtained for different AR model orders, specifically or AR(1), AR(2), AR(5), AR(10) , AR(50). The effects of the under/overmodelling on the PSD estimate were determined by plotting the periodogram of the original and mean-centered sunspot series with model-based PSD for several orders in Fig. 3.11 and 3.12.

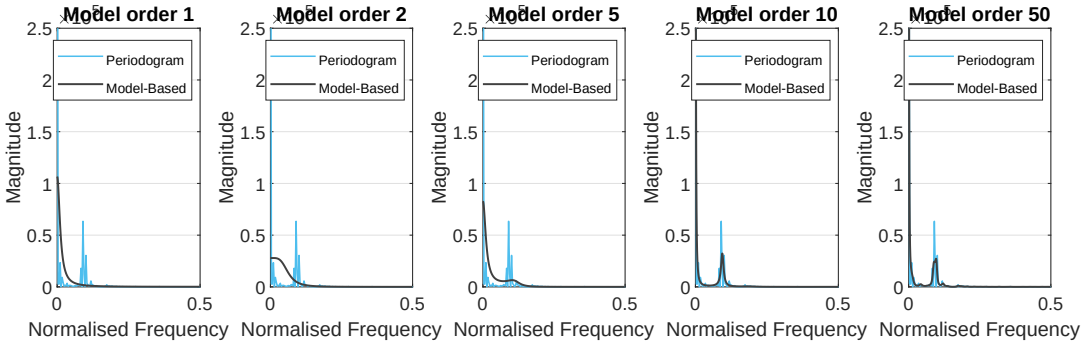


Figure 3.11: Comparison of Model-based PSD estimate with Periodogram estimate for original series, for different model orders.

In section 2.3, we obtained that the correct AR model order is 2, for both original and mean-centered sunspot series. This means that we expect the AR(2) PSD estimate to be the most accurate and reliable estimate, reflectling correctly the behavior of the system, compared to other models orders and periodogram estimates. AR(2) model-based estimate shows that the for original series, theoretical PSD would have a maximum value at zero frequency which will then decay to zero as frequency increases. This is due to the high DC offset (deterministic component) present in the series which is captured by the AR model and does not allow identification of peaks in spectrum. For this reason, the mean-centered version of series is used, shown in Fig. 3.12, which shows that according to AR(2) model-based PSD, the theoretical PSD would have a spectral peak at approximately normalised frequency 0.1, and a non-zero magnitude at 0 frequency. This indicates that the sunspot series is approximately a sinusoidal sequence with frequency 0.1.
Undermodelling (AR(1) model) does not have enough degrees of freedom (resulting in high variance and bias of model's parameters) to describe system's behaviour and thus fails to identify the spectral peak expected at 0.1 for mean-centered series. Overmodelling (AR models of order higher than 2) causes the model-based estimates to be closer to the periodogram estimates as model order increases, seen in both Fig.3.11 and 3.12. Although this manages to capture the spectral peak at 0.1 for mean-centered series and overall shape of spectrum, the magnitude of this peak is significantly higher compared to AR(2) model. Therefore both under/overmodelling provides unreliable results and these model-based estimates do not best represent the data.
It should be noted, however, that for original series, overmodelling manages to detect the spectral peak at frequency 0.1 (Fig. 3.11), therefore for non-centered data, higher order models can be used for spectral peak identification.

3.3 The Least Squares Estimation (LSE) of AR Coefficients

3.3.1 - Matrix form of Cost Function

An AR(p) process can be expressed as

$$x[n] = \sum_{i=1}^{p} \alpha_i x[n-i] + w[n] \ , \ for \ w[n] \sim \mathcal{N}(0,1) \tag{59}$$

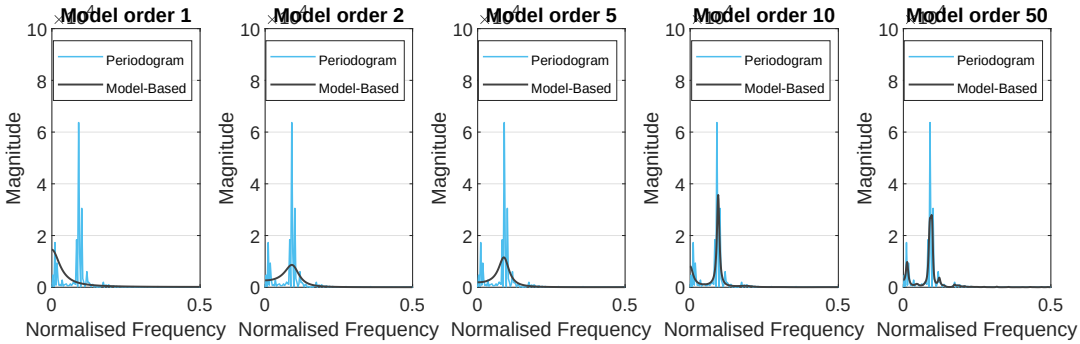


Figure 3.12: Comparison of Model-based PSD estimate with Periodogram estimate for mean-centered series, for different model orders.

The LS cost function for finding the unknown AR coefficients $\mathbf{a} = [\alpha_1, \alpha_2, ..., \alpha_p]$ is given by

$$J(\alpha) = \sum_{k=1}^{M} \left[x[k] - \sum_{i=1}^{p} \alpha_i x[k-i] \right]^2 \quad , \; for \; M \geq p \tag{60}$$

which for an AR(p) process this is equivalent to

$$J(\alpha) = \sum_{k=1}^{M} \left[\hat{r}_{xx}[k] - \sum_{i=1}^{p} \alpha_i \hat{r}_{xx}[k-i] \right]^2 \quad , \; for \; M \geq p \tag{61}$$

where M is data length and p is the model order. The cost function (J) can be expressed in compact (matrix) form. Firstly we define matrix **B** (Mx1 matrix) as

$$\boldsymbol{B} = \begin{bmatrix} x[1] - \sum_{i=1}^{p} \alpha_i x[1-i] \\ \vdots \\ x[M] - \sum_{i=1}^{p} \alpha_i x[M-i] \end{bmatrix} \tag{62}$$

This way we can express cost function J (which is a scalar quantity) as $J(\boldsymbol{\alpha}) = \boldsymbol{B}^T \boldsymbol{B}$. Matrix **B** can be re-expressed as

$$\boldsymbol{B} = \begin{bmatrix} x[1] \\ \vdots \\ x[M] \end{bmatrix} - \begin{bmatrix} \sum_{i=1}^{p} \alpha_i x[1-i] \\ \vdots \\ \sum_{i=1}^{p} \alpha_i x[M-i] \end{bmatrix} = \boldsymbol{x} - \boldsymbol{s} \tag{63}$$

where **x** is observed ACF estimate matrix (Mx1) and **s** is the model matrix (Mx1). We can express s[k] as

$$s[k] = [x[k-1], x[k-2], ..., x[k-p]] \, . \, [\alpha_1, \alpha_2, ..., \alpha_p]^T \tag{64}$$

Therefore, matrix **s** can be expressed as

$$\boldsymbol{s} = \begin{bmatrix} x[1-1] & x[1-2] & \dots & x[1-p] \\ x[2-1] & x[2-2] & \dots & x[2-p] \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ x[M-1] & x[M-2] & \dots & x[M-p] \end{bmatrix} . \boldsymbol{a} = \begin{bmatrix} x[0] & 0 & \dots & 0 \\ x[1] & x[0] & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ x[M-1] & x[M-2] & \dots & x[M-p] \end{bmatrix} . \boldsymbol{a} = \boldsymbol{H} . \boldsymbol{a} \tag{65}$$

where **H** is called the observation matrix and has dimensions of Mxp. It should be noted, that x[n] = 0 for n<0. Therefore, since **B**=**x**-**s** and **s**=**Ha**,

$$J(\boldsymbol{a}) = \boldsymbol{B}^T \boldsymbol{B} = (\boldsymbol{x} - \boldsymbol{H}\boldsymbol{a})^T (\boldsymbol{x} - \boldsymbol{H}\boldsymbol{a}) \tag{66}$$

The LSE for unknown coefficients $\boldsymbol{a}$ is given by expanding first the cost function

$$J(\boldsymbol{a}) = (\boldsymbol{x} - \boldsymbol{H}\boldsymbol{a})^T (\boldsymbol{x} - \boldsymbol{H}\boldsymbol{a}) = \boldsymbol{x}^T \boldsymbol{x} - \boldsymbol{x}^T \boldsymbol{H}\boldsymbol{a} - \boldsymbol{a}^T \boldsymbol{H}^T \boldsymbol{x} + \boldsymbol{a}^T \boldsymbol{H}^T \boldsymbol{H}\boldsymbol{a} = \boldsymbol{x}^T \boldsymbol{x} - 2\boldsymbol{x}^T \boldsymbol{H}\boldsymbol{a} + \boldsymbol{a}^T \boldsymbol{H}^T \boldsymbol{H}\boldsymbol{a} \tag{67}$$

Taking the derivative of the cost function with respect to the coefficients

$$\frac{\partial J(\boldsymbol{a})}{\partial \boldsymbol{a}} = -2\boldsymbol{H}^T \boldsymbol{x} + 2\boldsymbol{H}^T \boldsymbol{H}\boldsymbol{a} \tag{68}$$

Setting the gradient to zero, yield the LSE

$$\hat{\boldsymbol{a}} = (\boldsymbol{H}^T \boldsymbol{H})^{-1} \boldsymbol{H}^T \boldsymbol{x} \tag{69}$$

The relation of the LSE for the unknown Ar coefficients to the corresponding Yule-Walker estimates can be explained by considering that in LS method, the minimum of cost function occurs when gradient is zero, i.e.

$$\frac{\partial J(a)}{\partial a} = \sum_{n=1}^{M} -2 \left(x[n] - \sum_{i=1}^{p} \alpha_i x[n-i] \right) \left(\sum_{i=1}^{p} x[n-i] \right) \tag{70}$$

therefore,

$$x[n] = \sum_{i=1}^{p} \hat{\alpha}_i x[n-i] \tag{71}$$

which is equivalent to

$$\hat{r}_{xx}[k] = \sum_{i=1}^{p} \hat{\alpha}_i \hat{r}_{xx}[k-i] \tag{72}$$

In the case of Yule-Walker method, however, AR coefficients are estimated according to the equations

$$\hat{r}_{xx}[k] = \begin{cases} \sum_{i=1}^{p} \alpha_i r_{xx}[k-i] & , \; for \; k \geq 1 \\ \sum_{i=1}^{p} \alpha_i r_{xx}[k-i] + \sigma_w^2 & , \; for \; k = 0 \end{cases} \tag{73}$$

This means that the LSE method uses the Yule-Walker equations with restriction of $k \geq 1$, which are called Least-square modified Yule-Walker equations (LSMYWE).

3.3.2 - Examination of observation matrix H

Observation matrix **H** in this case is classified as a stochastic matrix, although in general this is considered a deterministic function, This is because elements of matrix **H** are ACF estimates, which are considered as stochastic functions, since computation of ACF estimates is dependent on random process x[n], which is a function of white noise. ACF estimates are based on a finite-size sample of x[n] and thus it should be noted that ACF estimate values are not statistically reliable for higher time lags, i.e. estimates such as $\hat{r}_{xx}(M-p)$, for which M$\gg p$.

3.3.3 - Applying LSE approach to real world sunspot time series

In this part, the AR coefficients for different model orders for standardised sunspot time series were obtained using the LSE method. These coefficients were also compared to the coefficients obtained from the Yule-Walker method, as shown in Figure 3.13.

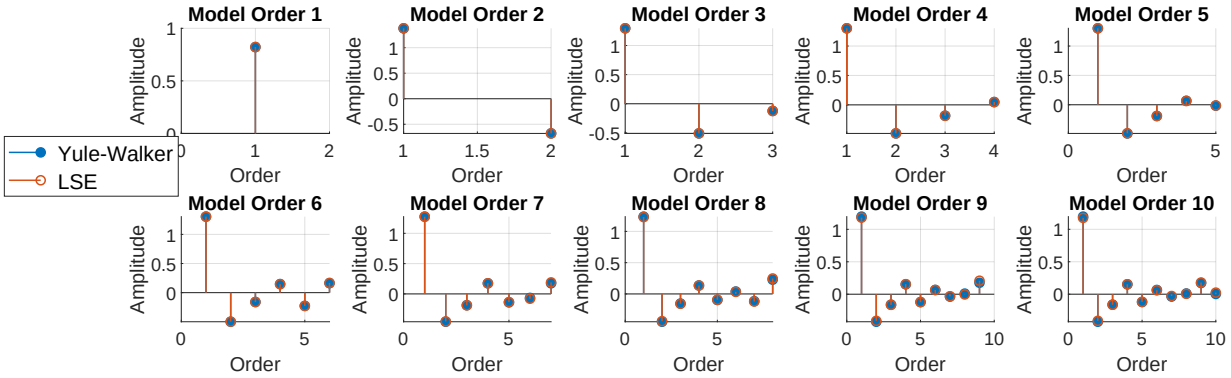


Figure 3.13: AR coefficients estimated using LSE method. Yule-Walker estimates included also for comparison

The minor differences in the two estimates arise due to reasons explained in part 3.3.1.

3.3.4 - Approximation error of obtained AR models

Approximation error measures the difference between the true data value and the approximation by the model.

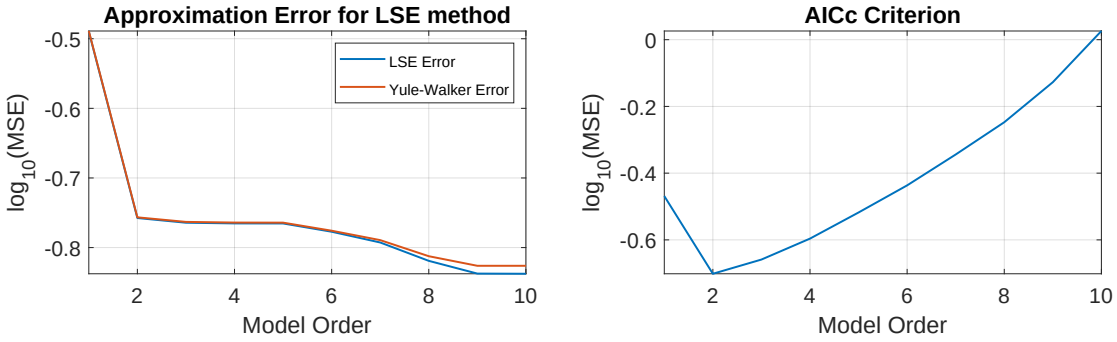


Figure 3.14: (Left)Plot of Approximation error for LSE method, together with Yule-Walker method error. (Right) AICc criterion using LSE error.

In our case, MSE was used for this measure. Approximation error was plotted in Fig. 3.14, together with the approximation error using Yule-Walker method, which are very similar. As we have seen before, the approximation error decreases as we increase the model order. In order for optimal order to be identified, we will introduce a penalty for overmodelling, to balance approximation error with cost of overmodelling. The corrected Akaike criterion is used, due to small sample size, which was also plotted in Fig. 3.14. The minimum of AICc plot is found at order model 2, which is the same model order chosen in Section 2.3. This is due to the strong relationship between the two methods used, expain in 3.3.1.

3.3.5 - Power Spectra associated with AR(p) models

The power spectra associated with constructed AR(p) models are plotted in Fig. 3.15. The PSD estimated using periodogram are also plotted for comparison.

The PSD estimates obtained are very similar to the ones obtained in section 3.2. using the Yule-Walker method. This is due to the strong relationship between the two methods. We can conclude for this method as well that overfitting (using model order greater than 2) increases the magnitude of the peak at normalised frequency of 0.1, and the PSD estimates follow closer the shape of periodogram, producing unreliable results.

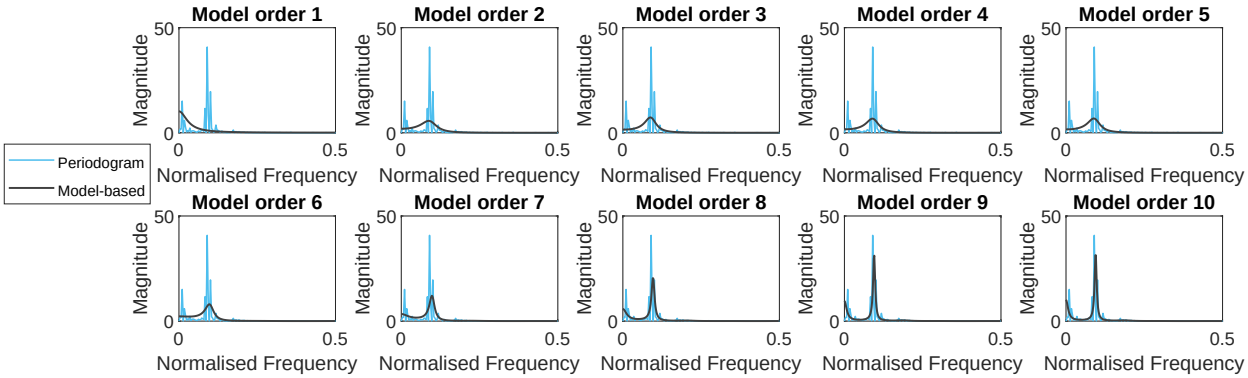


Figure 3.15: Model-based PSD of AR(p) for different model orders

3.3.6 - Behaviour of MSE vs Data Size (N)

Using model order 2, which was identified as the optimal model order, approximation error was computed for data lengths N $\in [10:5:250]$. This was then plotted in Figure 3.16 in order to identify the check the behaviour of MSE versus N, and identify an optimal data length for the AR modelling of the series.

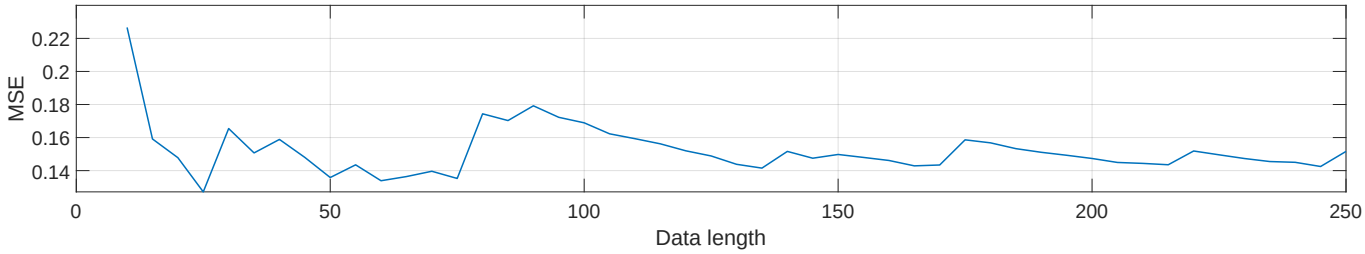


Figure 3.16: Mean-square error (MSE) versus Data length for AR(2) model of sunspot time series.

Figure 3.16 shows that the MSE error reaches a relatively constant value of 0.15 after data length 125. Therefore, data length of 125 is chosen as the optimal data length, since higher data lengths would have approximately the same error and would require higher number of computations.

3.4 Spectrogram for time-frequency analysis: dial tone pad

3.4.1 - Generating Random London Landline number

A random London Landline number, i.e. a sequence in the form 020 XXXX XXXX is generated, where the last eight digits are drawn from a uniformly distributed set 0,1,2,3,4,5,6,7,8,9. The discrete sequence **y** is computed assuming each digit is pressed for 0.25s followed by an idle time of 0.25s. According to the Dual Tone Multi-frequency (DTMF) system, a signal composed of two frequencies is assigned to each button of the keypad, i.e. $y[n] = sin(2\pi f_1 n) + sin(2\pi f_2 n)$, where f_1, f_2 are frequencies in Hz and n is the time index. The pairs of frequencies corresponding to each key are shown in Table 7 .

Table 7: Dial Pad frequencies

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

For generation of sequence **y**, the proposed sampling frequency of 32768 Hz is used. In general, sampling frequency (f_s) must be at least equal to the Nyquist Rate, i.e. $f_s \geq 2f_N$, where f_N is the signal's highest frequency (bandwidth), in order to avoid aliasing and to allow the signal in continuous-time domain to be uniquely determined by its samples. In our case, maximum frequency possible is 1477Hz, therefore the Nyquist rate is equal to 2954 Hz. Since $f_s \gg 2954Hz$, this ensures no aliasing. This is called is called oversampling, since sampling rate is significantly higher than the Nyquist Rate (by a factor of ≈ 11). A sampling frequency of 32768 Hz, means that each interval of 0.25s corresponds to 8192 samples. This improves resolution, creating a smoother graph to be analysed more accurately. Additionally, sampling frequency is a power of 2 (2^{15}). This is important since FFT is actually power-of-2 DFT and this increases efficiency and speed of computations. The signal **y** is shown in Fig.3.17 for two different keys (0 and 2) and the idle sequence.

3.4.2 - Analysing Spectral components using Spectrogram

Spectral components of sequence **y** were a analysed using `spectrogram`, using non-overlapping segments and Hanning window (due to low-amplitude sidelobes compared to other windows). The results together with the corresponding FFT segments are shown in Fig 3.18.

Spectrogram is a 2D frequency versus time plot, which uses colour to denote the amplitude of each frequency at each time. Peaks in spectrogram are shown in bright-yellow colour, while zero-magnitude values with dark blue colour. As a result, we can distinguish the peak frequency values that make up the signal in each 0.25s interval by the location of yellow bars on the y-axis, which occur at the frequencies in Table 7.

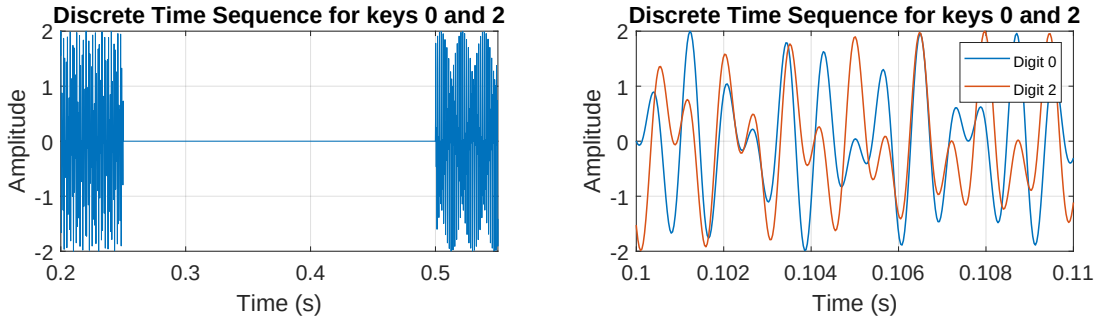


Figure 3.17: (Left) Discrete Time Sequence for keys 0 and 2. (Right) Differences between the tones.

It should be noted that for intervals where the key is pressed, the spectrogram does not have only two unique peaks due to Hanning window used, which has a shape similar to that of half a cycle of a cosine wave. Fig.3.8 also shows the FFT of each segment, where magnitude is in Decibels to make peaks more distinguishable. Also, as expected, each digit is produces two spectral peaks due to the two frequency components present. However, FFT plots can only be used to determine the numbers present in the landline number, but not their sequence.

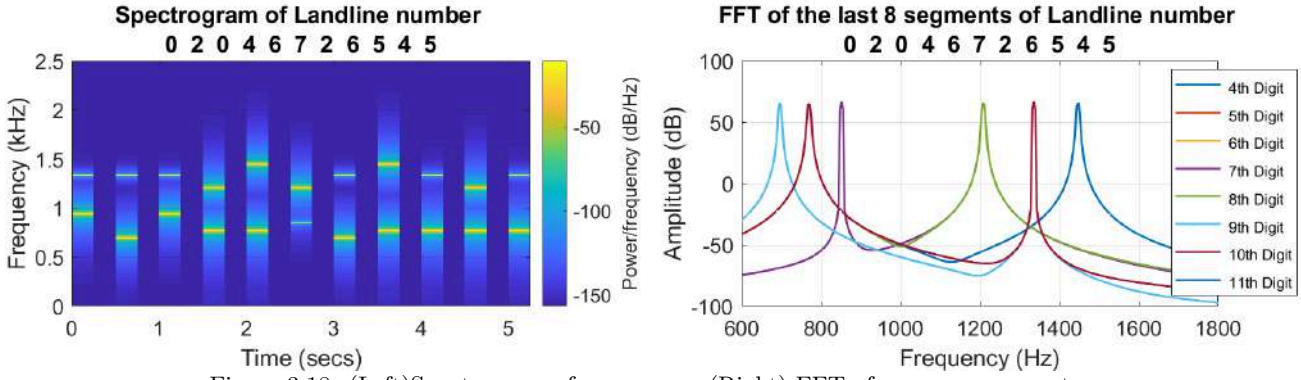


Figure 3.18: (Left)Spectrogram of sequence **y** (Right) FFT of sequence segments

3.4.3 - Method for identifying sequence generated by key press

Using spectrogram, we can identify the sequence of generated, by identifying the spectrum peaks at each time interval of 0.25s. To obtain the frequency peaks in each segment, we will obtain the sequence segment corresponding to a key press only, ignoring the idle time between the segments. Once the sequence of segments is determined, we can use spectral estimates for each segment to obtain the location of the two local maxima in the spectrum. This can be done in MATLAB using the command `[pks,locs] = findpeaks(data)`. The variable `locs` will give us the indices of the maxima value which can be used to obtain the corresponding normalized frequency values. These in turn can be converted to actual frequencies by multiplying them with the sampling frequency. Then these frequency values will be matched with frequencies of Table 7 to obtain the corresponding value of the digit, whose spectrum is analysed.

3.4.4 - Addition of channel noise to sequence y

Parts 3.4.1-3.4.3 were repeated while adding white noise to sequence **y**, specifically WGN. Three cases were considered: (1) Low-variance noise ($\sigma^2 = 0.001$), (2) Medium-variance noise($\sigma^2 = 0.1$) and (3) High-variance ($\sigma^2 = 100$) noise (immersing sequence completely in noise).

We can express the signal corrupted by channel noise as $y_c[n] = y[n] + w[n]$, where $w[n] \sim \mathcal{N}(0, \sigma^2)$. Effect of noise on signal can be explained by considering Signal-to-Noise ratio (SNR), of a signal with variance s^2 given by:

$$SNR = \frac{P_{signal}}{P_{noise}} = \frac{s^2}{\sigma_N^2} \tag{74}$$

Therefore, as noise variance (σ_N^2) increases, SNR drops which means that level of a desired signal is becoming smaller to the level of background noise. If signal power is estimated using sample variance, we obtain that $s^2 \approx 0.5233$. Therefore, when noise variance is higher than 0.5233, SNR< 1, so this indicates more noise than signal. In the case of high-variance, we expect SNR lower than 1, thus we expect key tones to be non-identifiable. This can be shown using simulations shown in Fig. 3.19-3.21.

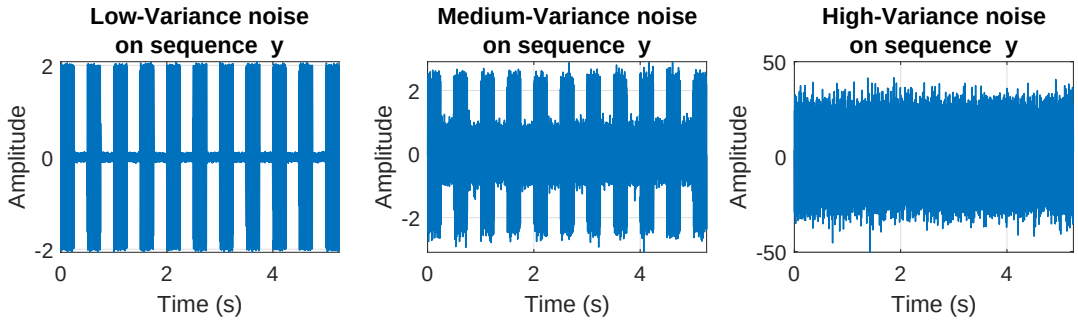


Figure 3.19: Plot of signal corrupted by channel noise, of different variances, in the time-domain.

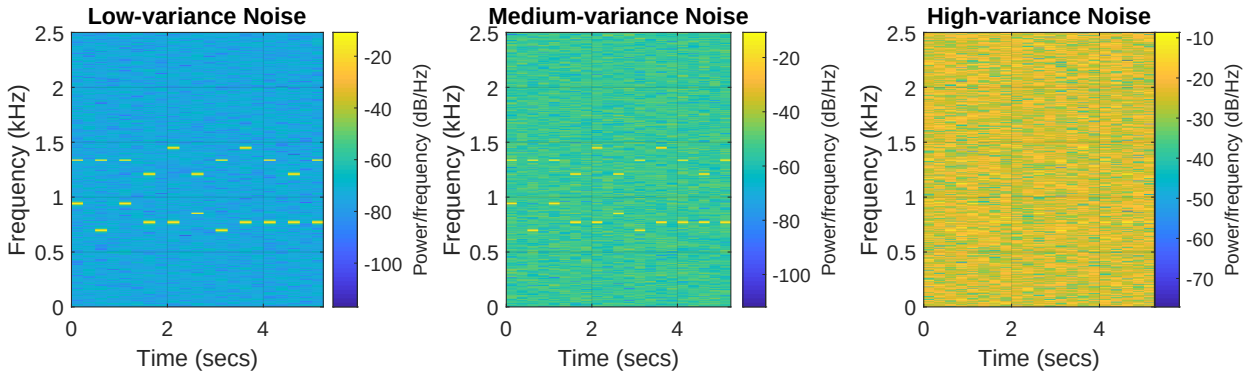


Figure 3.20: Spectrogram Plots for the three cases of noise variance.

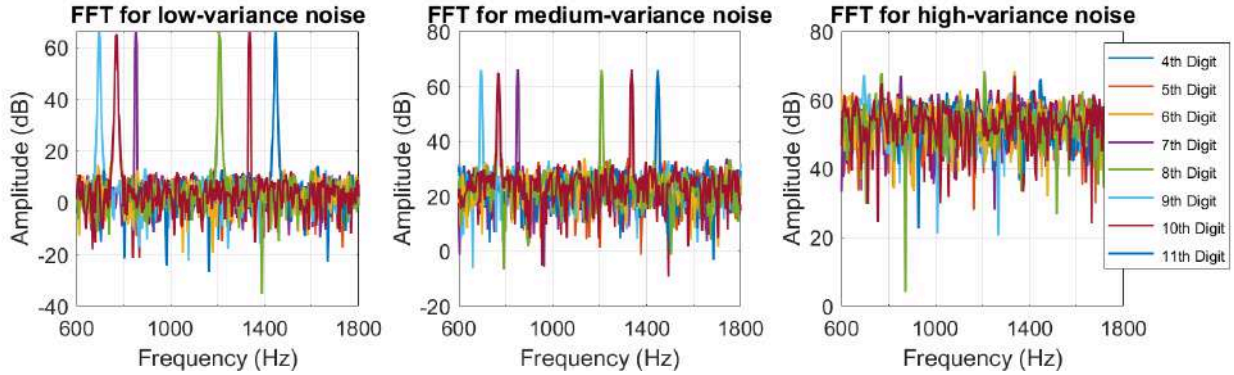


Figure 3.21: FFT of segments for the three cases of noise variance.

The three figures above, show that for the low and medium variance noise added to signal, the spectral peaks can still be identified from spectrogram and FFT of segments. As variance increases, noise causes more significant distortion to original signal which causes amplitude of all frequencies (due to white-noise) to increase. When maximum noise-variance is used, to completely immerse signal in noise, the signal tones can no longer be identified from spectrogram or FFT of segments since amplitude of all frequencies increases above the original spectral peaks magnitude.

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

Standard periodogram (using user-defined `pgm` function) and averaged periodogram with different window lengths are used to obtain the PSD of the RRI data from section 2.5. The PSD of the three trials using both methods is shown in Fig. 3.22 and 3.23.

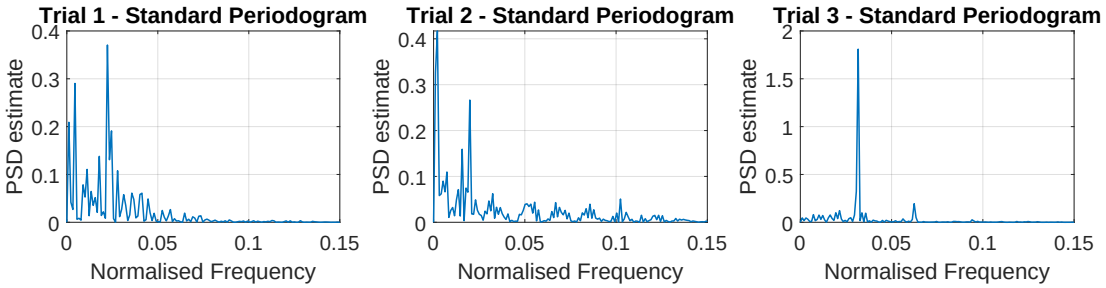


Figure 3.22: Standard Periodogram for the three trials

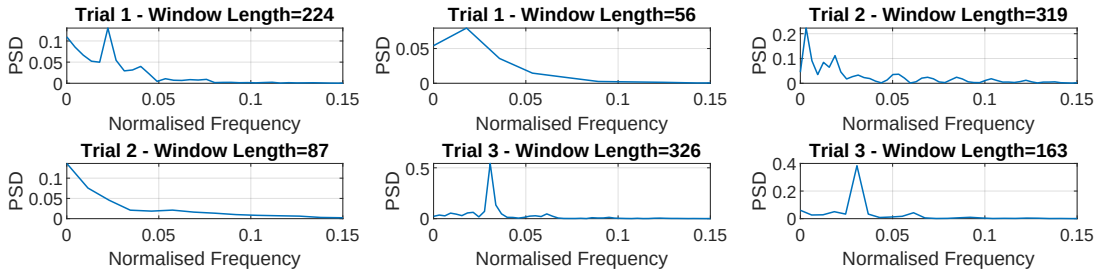


Figure 3.23: Averaged Periodogram for the three trials using different window lengths

The PSD estimates of RRI data from the three trials contain two spectral peaks corresponding to speeding up of heart rate during inspiration (higher frequency) and slowing down of heart rate during expiration (lower frequency), due to Respiratory Sinus Arrhythmia (RSA). However, each trial's spectral peaks occur at different frequency values, which are illustrated in the table below.
This is also shown in the averaged periodogram estimates as well. However, for lower window lengths variance decreases but spectral resolution decreases as well, which makes it more difficult to distinguish spectral peaks, as was shown for Trial 2 with window length 87.

Table 8: Spectral peaks using Periodogram estimates for the three trials (using normalised frequencies)

	Trial 1	**Trial 2**	**Trial 3**
Lower Frequency (Expiration)	0.00446	0.00209	0.03170
Higher Frequency (Inspiration)	0.02232	0.01985	0.06237

4 Optimal filtering - Fixed and Adaptive

4.1 Wiener Filter

A 1000-sample WGN sequence **x** is generated and presented as input to system described by a filter with coefficients **b**=[1,2,3,2,1] and **a**=[1]. The variance of system's output y[n] was normalised (mean-centered and divided by its standard deviation) so that it has unit variance. Sequence z[n] is generation by the introduction of additive 1000-sample WGN noise η[n] with standard deviation equal to 0.1 to sequence y[n]. The signal-to-noise (SNR) in dB for sequence z[n], which is composed of signal y[n] and noise η[n], is found using the equation:

$$SNR_{dB} = 10log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) = 10log_{10}\left(\frac{\sigma_y^2}{\sigma_\eta^2}\right) \tag{75}$$

We know that $\sigma_\eta^2 = (0.1)^2$ and $\sigma_y^2 = 1$. Therefore, the theoretical SNR value is expected to be 20dB. Using MATLAB function `snr` this was found to be 20.0388dB.

4.1.1 - Finding optimal coefficients for Wiener Filter

The autocorrelation matrix of sequence **x**, i.e. $\boldsymbol{R}_{xx}$, and cross correlation vector between sequence **z** and **x**, $\boldsymbol{p}_{zx}$, were computed in MATLAB using `xcorr` command. The optimal Wiener coefficients of the Wiener filter were found using:

$$\boldsymbol{w}_{opt} = \boldsymbol{R}_{xx}^{-1}\boldsymbol{p}_{zx} \tag{76}$$

In this case, the vector $\boldsymbol{w}_{opt}$ contains 5 elements tabulated below. It should be noted that the original coefficients obtained are normalised due to the normalisation operation on process y[n]. Therefore, the optimal coefficients estimates can be obtained by multiplying the normalised coefficients by the variance of the original process y[n]. These estimated are shown to be very close to the theoretical values.

Table 9: Wiener filter optimal coefficients

b	1	2	3	2	1
Normalised $\boldsymbol{w}_{opt}$	0.2366	0.4777	0.7181	0.4777	0.2301
$\boldsymbol{w}_{opt}$	0.9915	2.0014	3.0088	2.0015	0.9648

4.1.2 - Effects of additive noise variance and number of filter coefficients

The experiment was repeated by varying the variance of additive noise η[n] in the region $\sigma^2 \in [0.1, 10]$, and SNR and the optimum Wiener filter solution were re-calculated for each case, summarised in table below.

Table 10: Absolute Error for MLE for different f_0 values

σ_η^2	Theoretical SNR	Empirical SNR	Normalised $\boldsymbol{w}_{opt}$					$\boldsymbol{w}_{opt}$				
0.1	10	10.1528	0.2136	0.4580	0.7119	0.4556	0.2331	0.9186	1.9697	3.0614	1.9593	1.0025
0.5	3.0103	3.5622	0.1948	0.4433	0.7390	0.4703	0.2459	0.8376	1.9061	3.1779	2.0224	1.0574
1	0	0.3247	0.2349	0.4655	0.7171	0.5356	0.2377	1.0103	2.0018	3.0837	2.3031	1.0221
3	-4.7712	-4.4178	0.2894	0.4680	0.7122	0.5322	0.2950	1.2446	2.0126	3.0624	2.2887	1.2687
5	-6.9897	-7.0116	0.1591	0.4216	0.7174	0.5631	0.3474	0.6842	1.8130	3.0848	2.4214	1.4937
10	-10	-10.3911	0.2501	0.5103	0.8391	0.4162	0.2662	1.0755	2.1944	3.6085	1.7895	1.1448

As expected, the SNR for z[n] decreases as noise variance increases, and becomes negative when the additive noise variance becomes greater than 1, since variance of y[n] sequence is equal to 1. Additionally, the higher the noise variance, the greater the deviation of optimal coefficients of the Wiener filter from the theoretical values. This can be explained by considering the effect of noise variance on the cross-correlation function between x[n] and z[n]. It can be shown that the due to the fact that z[n]=y[n]+η[n], the variance of sample cross-correlation function is proportional to the noise variance (dependent on σ_N^2). Therefore, as noise variance increases, variance of CCF increases. Additionally, as noise variance increases, then filter's output becomes more immersed in noise, therefore z[n] converges to $\eta[n]$, meaning that any correlation introduced by filter is lost. As a result, system identification is becoming harder. Note that matrix $\boldsymbol{R}_{xx}$ remains unaffected, as it does not involve η[n].

The effects on Wiener solution of assuming $N_w > 4$ were also investigated. The optimum Wiener filter solution is shown in table below for $N_w = 5$ and $N_w = 6$, using additive noise standard deviation of 0.1.

In general, the number of coefficients obtained for each N_w value are equal to $N_w + 1$. However, only the first 5 coefficients are significant in value and are very close to the theoretical values, while any additional coefficients are varying closely to zero-value, and are non-zero due to additive noise effects on p_{zx}. This effect can also be seen by calculating the MSE between the sequence **y** and sequence $\hat{\boldsymbol{y}}$, which is the output of the optimum filter obtained using Wiener solution, for different N_w values. This additionally shows that MSE error does not decrease further after $N_w = 4$. This can be used to indicate the correct filter order.

Table 11: Effect of $N_w \geq 4$ on Wiener solution

N_w	$\boldsymbol{w}_{opt}$						
4	0.9940	1.9812	3.0230	2.0294	1.0005		
5	0.9940	1.9813	3.0231	2.0294	1.0003	-0.0049	
6	0.9939	1.9813	3.0232	2.0294	1.0003	-0.0050	-0.0045

Table 12: MSE between sequence $\mathbf{y}$ and sequence $\hat{\boldsymbol{y}}$, as a function of N_w.

N_w	0	1	2	3	4	5	6	7	8	9	10
MSE	19.5781	15.3310	5.5183	1.1266	0.0020	0.0020	0.0020	0.0023	0.0024	0.0024	0.0031

4.1.3 - Estimating computational complexity of Wiener solution

Algorithm complexity can be assessed by considering the approximate number of additions and multiplications involved in each step of computation $\boldsymbol{w}_{opt}$ solution, which contains $N_w + 1$ filter coefficients. The first steps include the computation of ACF and CCF estimates. By assuming that in general $N \gg N_w$, for each time-lag, there are approximately N additions and $N + 1 \approx N$ multiplications for both ACF and CCF estimates (N is the sample size). However, for ACF, $2N_w + 1$ time-lags are considered whereas for CCF $(N_w + 1)$ time-lags (since we consider CCF from $-N_w \leq m \leq 0$). Inversion of matrix $\boldsymbol{R}_{xx}$, which has $(N_w + 1)$ by $(N_w + 1)$ dimensions, requires roughly $(N_w + 1)^3$ operations. Finally, matrix multiplication between $\boldsymbol{R}_{xx}^{-1}$ and $\boldsymbol{p}_{zx}$ involves $(N_w + 1)$ multiplications and N_w additions for each element of $\boldsymbol{w}_{opt}$, which has a total of $(N_w + 1)$ elements.
Additing these operations up gives an estimate of:

$$Total\ Operations\ \approx (N_w + 1)^2(2N_w + 1) + N(N + 1)(3N_w + 2)$$

Therefore, Wiener solution has complexity $\mathcal{O}(N^2)$.

4.2 The least mean square (LMS) algorithm

4.2.1 - Implementing LMS algorithm in MATLAB

Using the least mean square (LMS) algorithm an adaptive filter is used to approximate, in a recursive fashion, the Wiener solution. The LMS algorithm is defined by:

$$\begin{aligned} \boldsymbol{w}(n+1) &= \boldsymbol{w}(n) + \mu e[n]\boldsymbol{x}(n)\ ,\ \boldsymbol{w}(0) = 0 \\ \hat{y} = \boldsymbol{w}^T(n)\boldsymbol{x}(n) &= \sum_{m=0}^{N_w} w_m(n)x(n-m) \\ e[n] &= z[n] - \hat{y}(n) \end{aligned} \tag{77}$$

where μ is the adaptation gain (learning rate) which controls the stability of the algorithm, $\boldsymbol{w}(n)$ is weight vector of the adaptive filter, and error e[n] is the difference between z[n] and the output of the adaptive filter $\hat{y}$[n]. This algorithm was implemented in MATLAB by writing a function called `lms`. This function has as inputs the N-sample vectors $\boldsymbol{x} = [x[1], ..., x[N]]^T$ and $\boldsymbol{z} = [z[1], ..., z[N]]^T$, the adaptation gain μ, and the order of the adaptive filter $(N_w + 1)$ and as outputs the LMS estimate $\hat{y}(n)$ for n=1,...,N , the error vector e[n] and the $(N_w + 1)xN$ matrix containing the time evolution of adaptive weights.
The function `lms` was then applied to **x** and **z** signals used in Section 4.1. The output of adaptive filter $\hat{y}(n)$ is plotted together with the measured output signal z[n] in Fig. 4.1, using a learning rate (μ) of 0.005.

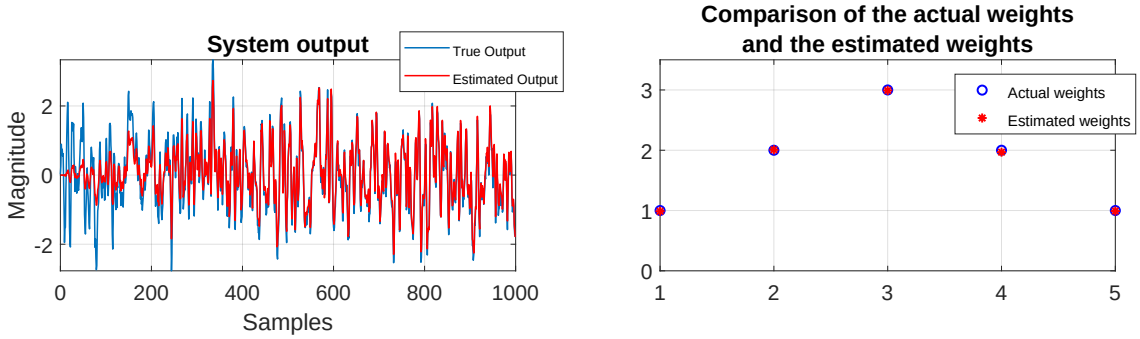


Figure 4.1: (Left)Plot of $haty(n)$ and z[n] (Right) Scatter Plot of true and estimated filter weights using LMS adaptive filter, in the final algorithm iteration.

Choice of μ must be done in order for LMS to converge to the optimal filter weights. Learning rate mu must ensure convergence in the mean and MSE. However, if mu ensures converge in the mean only, it enforces stability in the mean of LMS but the filter coefficients can still grow infinitely large. Convergence in MSE ensures that the LMS algorithm converges to real coefficients with low variance. Convergence in the mean requires $0 < \mu < \frac{2}{\lambda_{max}} = 1.8920$, where λ_{max} is the largest eigenvalue of autocorrelation matrix $\boldsymbol{R}_{xx}$. Convergence in the MSE requires $0 < \mu < \frac{2}{tr[\boldsymbol{R}_{xx}]} = \frac{2}{p\sigma_x^2} = 0.401$, where p is the filter order. Therefore, for testing μ was chosen to be 0.005.
Fig.4.1 shows that there is a higher estimation error during the initial iterations but this error decreases as number of iterations increases. As number of iterations increases, the LMS gets closer to the true filter weights and as a result the estimated output converges to the true output. This is due to finite speed of convergence and learning delay of LMS algorithm.

4.2.2 - Effects of learning rate μ

The adaptation gain is chosen to be 0.01 and the time evolution of filter coefficients and squared estimate error is shown in Fig.4.2.
In general, the algorithm uses approximations to the expectation operations, thus weights would never reach the optimal weights in the absolute sense, but a convergence is possible in mean.

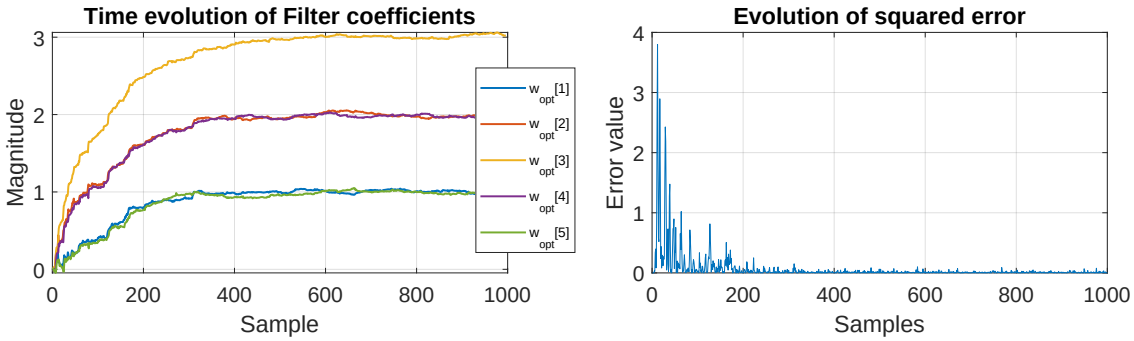


Figure 4.2: (Left)Plot of time evolution of estimated coefficients. (Right) Time evolution of squared error of LMS algorithm with $\mu = 0.01$.

This means that the weights, after convergence, will be varying about the optimal weights. However, if the deviations are large (large variance) then convergence in mean would be misleading and thus convergence in MSE must be ensured.

Fig. 4.2 shows that the estimated filter coefficients converge exponentially to their optimum values, with small variations around these optimal values. This shows that for $\mu = 0.01$ LMS algorithm both converges in the mean and MSE, since $\mu = 0.01 \ll \frac{2}{tr[\boldsymbol{R}_{xx}]} = 0.401$, thus bias and variance of estimation decreases with increasing number of iterations. Additionally, the squared error follows an exponential decay with time. It does not reduce to zero due to non-zero variance of coefficient estimates.

The effects of adaptation gain on LMS algorithm can be examined by plotting the evolution of the five coefficient estimates for different adaptation gains in the interval [0.002,0.5] (Fig. 4.3).

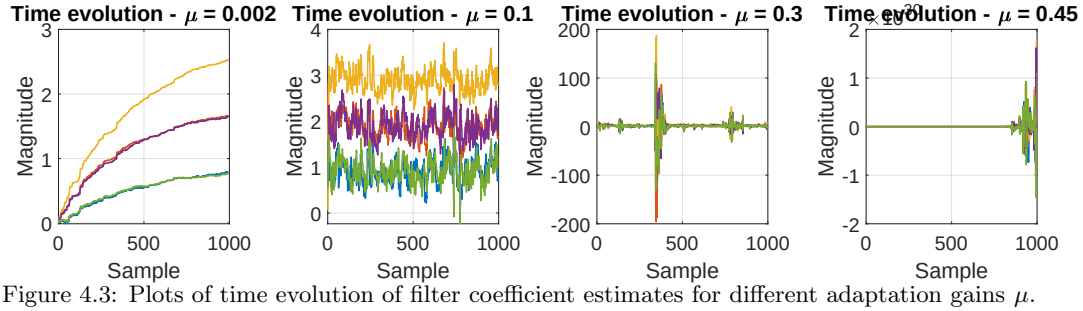


Figure 4.3: Plots of time evolution of filter coefficient estimates for different adaptation gains μ.

As it was discussed in Part 4.2.1, for adaptation gains that are lower than $\frac{2}{tr[\boldsymbol{R}_{xx}]} = 0.401$, the LMS algorithm converges both in mean and MSE and when μ is between 0.401 and $\frac{2}{\lambda_{max}} = 01.8920$ it converges only in the mean. Fig. 4.3 shows that for large μ values, i.e. values close to the MSE convergence bound , the algorithm converges faster to optimal values but oscillates with a larger variance about these optimal values. This is due to larger steps taken by the algorithm towards the optimum value, the optimal solution might be missed causing the estimates to vary largely and not converging towards the optimal values. Additionally, as adaptation gain decreases, then the algorithm convergence time increases (algorithm takes smaller steps), which explains the fact that for $\mu = 0.002$, the coefficients have not yet converged to true values. In general, there is a trade-off between convergence speed and convergence in MSE (estimate variance).

It should be noted that in Fig. 4.3 the coefficients do not converge in MSE for $\mu = 0.3$ which is within the theoretical MSE convergence bounds. This is due to the fact that autocorrelation matrix is constructed using sample ACF, which is an estimate of true ACF.

It can be shown that the time constant τ_i for convergence of the j^{th} coefficient to 1/e of its optimum value, for $0 < \mu \ll 2/\lambda_{max}$, given by

$$\tau_j = -\frac{1}{ln(1-\mu\lambda_j)} \approx \frac{1}{\mu\lambda_j} \ , \ where\ \lambda_j\ is\ the\ j^{th}\ eigenvalue\ of\ \boldsymbol{R}_{xx}. \tag{78}$$

The time index at which the the adaptive filter converges to the Wiener solution, can be estimated by considering the number of iterations required for coefficients to converge to their optimum value. The convergence time can be approximated as the maximum of 3τ, i.e. when all coefficients are at least at 95.02% of their optimum value. This was found to be at time index 339.

4.2.3 - Computational Complexity of LMS algorithm

Computational Complexity of algorithm is assessed by estimating the number of multiplications and additions required at each iteration. This is done by considering each computation involved separately.

Table 13: Number of Multiplications and Additions for each iteration of algorithm

Expression Computed	Additions	Multiplications
$\hat{y}[n] = \boldsymbol{w}^T(n)\boldsymbol{x}(n)$	$N_w + 1$	N_w
$e[n] = z[n] - \hat{y}[n]$	1	0
$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu e[n]\boldsymbol{x}(n)$	$N_w + 1$	$N_w + 2$
Total	$2N_w + 3$	$2N_w + 2$

This gives an overall total of $4N_w + 5$ computations and thus a complexity of $\mathcal{O}(N_w)$. Due to N iterations, the complexity of the whole algorithm is $\mathcal{O}(N)$, which is lower than the computational complexity of Wiener solution.

4.3 Gear shifting

In this section, a variable step LMS algorithm is implemented (gear shifting), in a function called `lms_gs`, in which adaptation gain μ is kept large for initial iterations and then gradually reduced in time. This makes use of the fact that large μ values increase the convergence speed while small μ values reduce the mean-square error in steady state. In the algorithm implemented the time-varying μ values are given by:

$$\mu(n+1) = \alpha\mu(n) + \beta e^2(n) \tag{79}$$

where α and β are in the range 0¡(a,b)¡1. According to Kwong, R. H., and Johnston, E. W. *Variable step size LMS algorithm.* IEEE Transactions on Signal Processing(1992), the parameter α must be close to value of 1 to ensure stability and convergence, and β must be close to 0 to minimize the degree by which e(n) affects μ(n+1). Using a=0.99 and b=0.001, the behavior of both the filter weights and error is computed and plotted in Fig. 4.4.

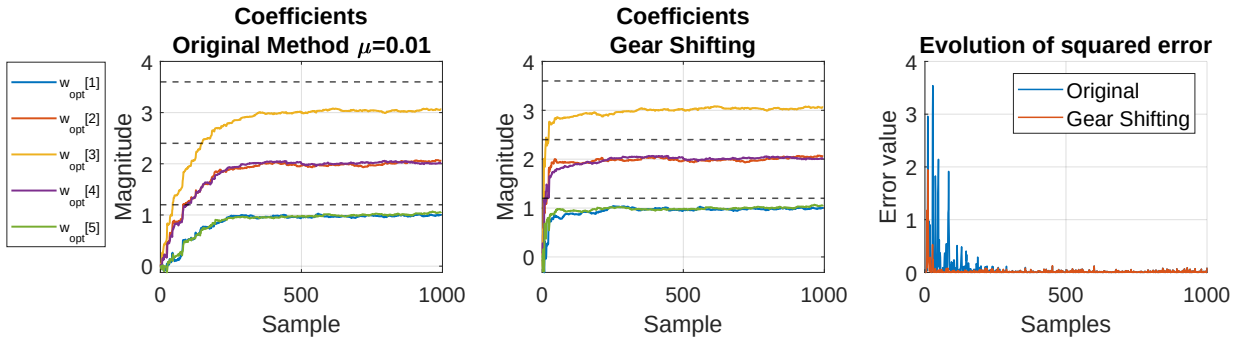


Figure 4.4: (Left) Plots of time evolution of filter coefficient estimates using the original LMS algorithm with $\mu = 0.01$, (Middle) Plots of time evolution of filter coefficient estimates using the variable step size LMS algorithm (Right) Comparison of time evolution of $error^2$ for the two methods

For the variable step-size LMS algorithm it was ensured that the maximum permissible overshoot for each coefficient is 20% of its true value. This was done by setting any coefficient values>1.2×true value, to be equal to 1.2×true value. It should be noted that the maximum permissible overshoot values for each coefficient are shown in Fig.4.4 as black broken lines. Fig.4.4 shows that coefficients converge to their optimal values in less iterations for the variable step-size LMS algorithm. Additionally, the squared error decays faster to zero value.

The two methods can be compared quantitatively by considering the rise time in each case. Rise time is defined as the time required for filter coefficients to rise from 10% to 90% of their steady values (true values). In each case the average rise time for the 5 filter coefficients was obtained. For LMS algorithm rise time was 185 and for the variable step-size LMS algorithm this was 52.4. Since the latter has a shorter rise time, this indicates that it is the best method of the two.

4.4 Identification of AR processes

4.4.1 - Implementing adaptive LMS algorithm

Linear adaptive filters have a wide range of applications. Speech recognition, for example, involves non-stationary signals, and therefore requires adaptive estimation. Application of linear adaptive filters is examined by first synthesizing a signal from WGN using an AR model, which is then analysed by an adaptive filter to find the AR model coefficients in order to recreate the original signal. Analysis stage is implemented in MATLAB using the function `adaptive_lms.m`, for the AR model with parameters **a**=[1,0.9,0.2].

Using adaptation gain (μ) equal to 0.01, as in section 4.2.2, the time evolution of coefficients α_1 and α_2 is shown in Fig.4.5.

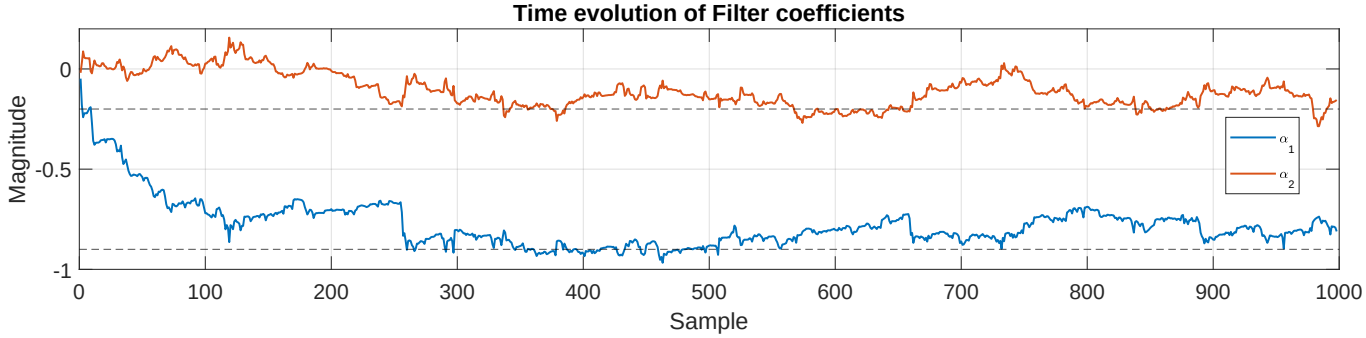


Figure 4.5: Time evolution of AR(2) model coefficients using adaptive LMS algorithm

The output of adaptation filter ($\hat{x}[n]$) is given by the following expression:

$$\hat{x}[n] = \alpha_1 x[n-1] + \alpha_2 x[n-2] \tag{80}$$

Therefore, this means that:

$$x[n] = \hat{x}[n] + e[n] = \alpha_1 x[n-1] + \alpha_2 x[n-2] - e[n] \tag{81}$$

However, an AR(2) model with coefficients **a**=$[a_0, a_1, a_2]$, amd input WGN ($\eta[n]$) is equivalent to

$$x[n] = -a_1 x[n-1] - a_2 x[n-2] + \eta[n] \tag{82}$$

Therefore, since in steady-state (optimal values of AR coefficient estimates), e[n] is minimised and is approximately equal to $\eta[n]$, the coefficients are expected to converge to the negative values of AR coefficients, in order for original signal to be reconstructed. In other words, the adaptive filter attempts to recover the original signal by cancelling the effects of AR model, by a negative feedback mechanism. As a result, α_1 will converge to -0.9 and α_2 to -0.2. The expected values of α_1 and α_2 are also plotted in Fig.4.5 as black broken lines. Variations around the true values arise due to the fact that $\eta[n]$ is not exactly equal to $e[n]$.

4.4.2 - Time evolution of adaptive filter coefficients for different adaptation gains

The MATLAB code `adaptive_lms.m` is used for analysis of evolution of coefficients α_1 and α_2 for four different adaptation gains, where $\mu = [0.01, 0.02, 0.05, 0.1]$. Results are shown in Fig.4.6.

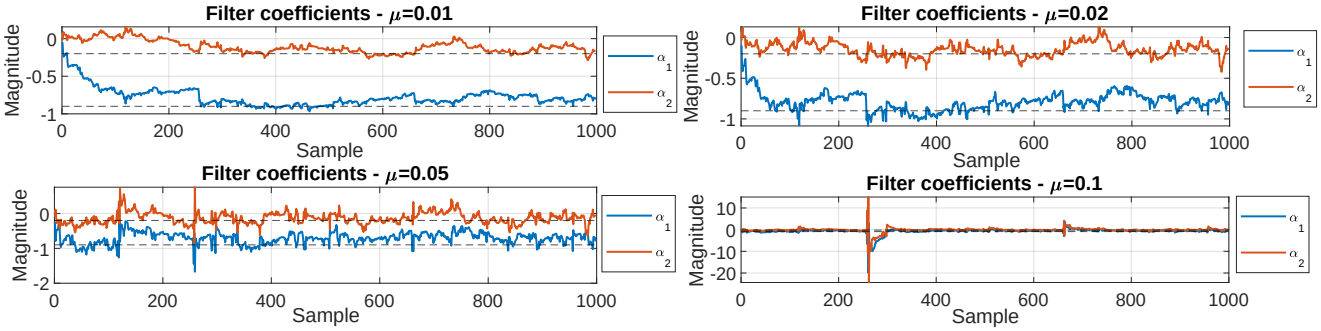


Figure 4.6: Time evolution of AR(2) model coefficients for $\mu = 0.01, 0.02, 0.05, 0.1$.

For this case, convergence of coefficients in the mean sense is ensured if $0 < \mu < \frac{2}{\lambda_{max}} = 0.5806$ and convergence in the MSE if $0 < \mu < \frac{2}{2*\sigma_x^2} = 0.4228$. Note that λ_{max} is the maximum eigenvalue of autocorrelation matrix of input x[n]. The figure shows that as adaptation gain increases the coefficients converge faster to to their expected values. However, as μ increases and becomes closer to the MSE convergence upper bound, the MSE error increases so deviations about the expected values are more significant. This again illustrates the trade-off between convergence speed and MSE of estimators, as was discussed in section 4.2. It should be noted that if gear shifting is used, both convergence time and MSE are reduced.

4.5 Speech recognition

4.5.1 - Performance of adaptive LMS algorithm on real-world audio signals

The adaptive LMS algorithm implemented in MATLAB, can be used in speech recognition. The algorithm is effectively a p^{th}-order predictor using samples delayed by 1,2,...,p time units to create its output. The performance of such predictor was tested on voice recordings of the sounds corresponding to "a","e","s","t" and "x" using sampling frequency of 44.1kHz and 1000-sample recording of audio signals. The performance of the predictor can be analysed by considering as a metric the Sum of Squared Errors (SSE) for each model order, which is also known as the Loss function for the model, for different adaptation gains. SSE values were plotted against model order in Fig. 4.7.

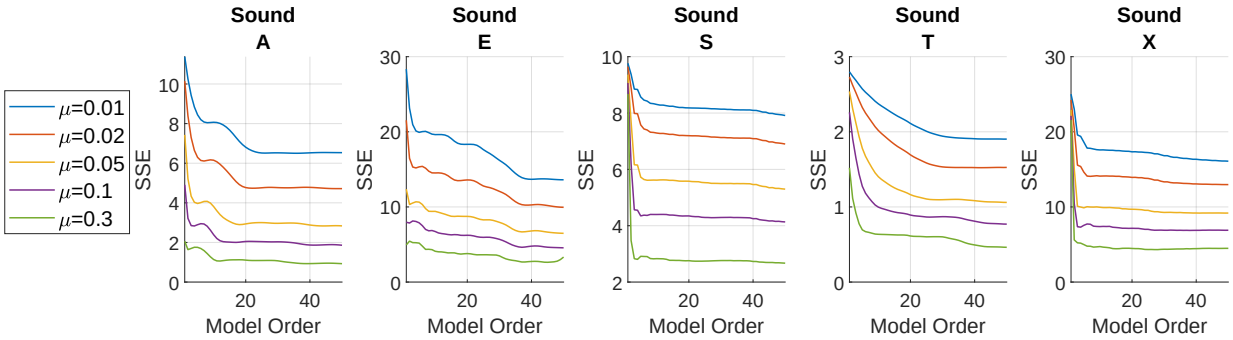


Figure 4.7: SSE against model order for the sounds "a","e","s","t" and "x", for $\mu = [0.01, 0.02, 0.05, 0.1, 0.3]$.

It should be noted that SSE is given by the summation of $e^2[n]$ for all n values. Fig.4.7 shows that the SSE is higher for all audio signals and model orders when the adaptation gain μ is small. This can be explained due to low convergence speed of algorithm for low step-size, which means that the AR coefficients have not yet converged to their optimal values. The fact that SSE is lower for higher μ values only applies here since the MSE converge bound has not yet been reached. Therefore, the optimal adaptation gain in this case is 0.3.

The figure also shows that SSE values for any considered μ value decrease as model order increases and then remain approximately constant. The model order at which SSE starts to remain constant is an indication of the optimal order of the predictor for the specific sound. It should be noted that SSE is in general a monotonically decreasing function, but after a specific model order decrease in SSE becomes very small compared to previous orders.

Moreover, gear shifting was also implemented for each audio signal and the SSE value was obtained for different model orders, as shown in Fig.4.8. This plot shows that each sound converges to a different SSE value, and converge to that value at different model order, thus indicating each sound is has different AR order.

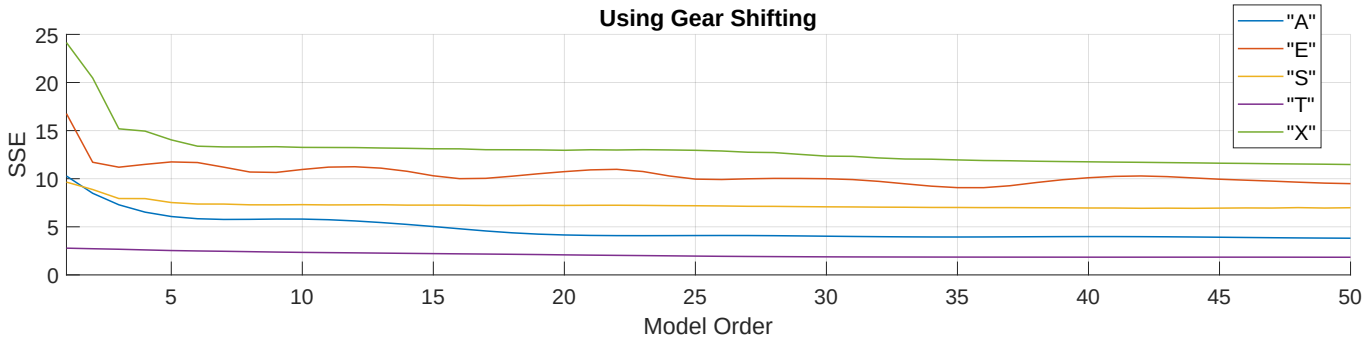


Figure 4.8: SSE against model order for the sounds "a","e","s","t" and "x", using gear shifting.

4.5.2 - Heuristic and Analytical approaches in determining optimal filter length

The optimal filter length can be determined firstly by following the method described in part 4.5.1, i.e. to find order after which SSE converges to a relatively constant value. Similarly, the time constant of the decaying error e[n] can be used (or convergence time constant of coefficients), which will remain approximately constant after a specific filter length. This can be explained from the fact that SSE decreases with model order, thus error e[n] decays faster making the total error less.

Another approach can be to observe the time evolution of AR coefficients for different model orders and identify the order after which any extra AR coefficients converge to a value very close to zero (assumed to not be part of model, similar to Yule Walker method). Additionally, since audio signals are involved, the optimal order is the one after which the reconstructed audio signal has no identifiable differences compares to the original signal (the order at which most of signal distortion vanishes). However, the last two proposed approached are time-consuming and require a large number of computations. Therefore, analytical approaches such as MDL, AIC and AICc criteria, discussed in section 2, could be used to determine optimal filter length. For non-stationary signals, Loss function may have multiple minimum values, therefore care must be taken to select filter order for which criteria are minimised. Another standard analytical approach is the prediction gain (R_p) of predictor given by

$$R_p = 10log_{10}\left(\frac{\sigma_x^2}{\sigma_e^2}\right) \tag{83}$$

where σ_x^2 and σ_e^2 denote the variance of input and error signals respectively. Optimal filter length is the order at which R_p remains relatively constant to its maximum value.

4.5.3 - Assessing Performance of Predictor using Prediction Gain

The performance of the predictor for each audio recording was assessed using prediction gain measures. The prediction gains of the corresponding predictors for a 1000-sample recording of speech signals, with sampling frequency f_s = 16kHz were calculated and plotted in Fig.4.9. The prediction gains for sampling frequency 44.1kHz was also plotted for comparison. The adaptation gain used in computations was 0.1.

Both cases show that prediction gain gradually converges to a maximum, approximately constant value. The order at which this constant value occurs is an indication of the optimal filter order as discussed in section 4.5.2. Prediction gain increases up to a maximum value because as filter order increases error signal power decreases. This is because the original signal is more accurately reconstructed up to the optimal filter length, where error signal has the minimum variance (power).

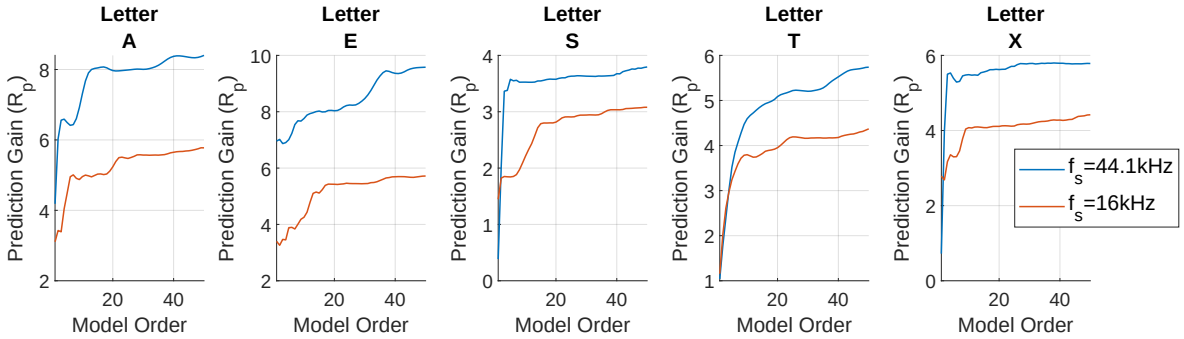


Figure 4.9: Prediction gains for the predictors for 1000-sample recordings of sounds "a","e","s","t" and "x", using $\mu = 0.1$.

Maximum prediction error for each sound, for the two sampling frequencies, is shown in table below.

Table 14: Maximum Prediction Gain (in dB) for each sound for a)f_s=16kHz and b)$f_s = 44.1kHz$

Sound	**"a"**	**"e"**	**"s"**	**"t"**	**"x"**
Prediction Gain for $f_s = 16kHz$	5.775	5.718	3.079	4.370	4.419
Prediction Gain for $f_s = 44.1kHz$	8.367	9.580	3.786	5.739	5.786

Sampling rate is in fact how many samples can be obtained per unit time. For sampling rate of 16kHz, the sampling period is equal to 62.5μs and for sampling rate of 44.1kHz, sampling period is 22.7μs. Since 1000 samples are taken from each recording, then for 16kHz the recording has a duration of 62.5ms while for 44.1kHz the duration is 22.7ms. This means that higher sampling frequency improves the resolution for prediction, which in turn reduces the error signal power and increases prediction gain (as shown in Fig.4.9).

As a result, prediction accuracy is improved. Increasing sample rate, means that for a fixed time period more samples are obtained, therefore in discrete-time domain sequence is longer.
Improvement in resolution reduces the error e[n] and since learning curves are logarithmic plots of the mean squared error (MSE), then this means that learning curves decay faster to zero.
Table 14 shows that for the same sample frequency, the prediction gain is higher for vowel sounds("a","e") compared to consonant sounds ("s","t","x"). Additionally, the higher the sampling rate the greater the differences in prediction gains between vowel and consonant sounds. Therefore, vowel sounds can be more easily identified and predicted.

4.6 Dealing with Computational Complexity: Sign Algorithms

A simplified version of the LMS algorithm is the class of the sign algorithms:

$$\begin{aligned} signed-error \quad & \boldsymbol{w}(n+1) = \boldsymbol{w}(n) = \mu sign(e[n])\boldsymbol{x}(n) \\ signed-error \quad & \boldsymbol{w}(n+1) = \boldsymbol{w}(n) = \mu e[n] sign(\boldsymbol{x}(n)) \\ sign-sign \quad & \boldsymbol{w}(n+1) = \boldsymbol{w}(n) = \mu sign(e[n]) sign(\boldsymbol{x}(n)) \end{aligned} \tag{84}$$

These algorithms are implemented as MATLAB functions and simulated in order to compare their performance with that of the basic LMS algorithm for AR parameter identification (using $\mu = 0.02$) in section 4.4 and for speech file of sound "a" in section 4.5. It should be noted that for the speech file an AR(10) model was used.
The time-evolution of filter coefficients of AR process of section 4.4 was plotted in Fig.4.10 for the four methods. This shows that basic LMS algorithm has the highest convergence speed, as expected, and sign-sign algorithm the lowest convergence rate. However, the sign-sign algorithm despite its low conversion rate, is less prone to divergence when higher adaptation gains are used and also has the lowest computational complexity.

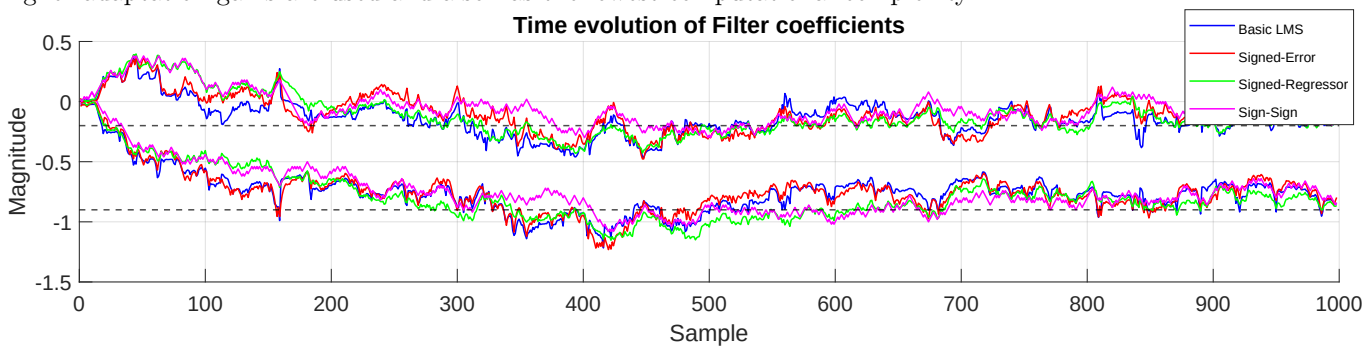


Figure 4.10: Time-Evolution of AR coefficients of process in section 4.4 using LMS, signed-error, signed-regressor and sign-sign algorithms. Broken lines show the expected values of the coefficients.

The same procedure was repeated for sound "a" signal and time-evolution of the first AR coefficient (Fig.4.11). This shows that the variance of the sign LMS algorithms is higher compared to basic LMS algorithm, with sign-sign algorithm having the greatest deviations and lower convergence speed.
The advantage of the sign methods is the low computational complexity compared to LMS algorithm and opposition to divergence for high adaptation gain values. However, since sign operation is used, it introduces a form of quantisation error which increases the steady state error and the convergence time constant.

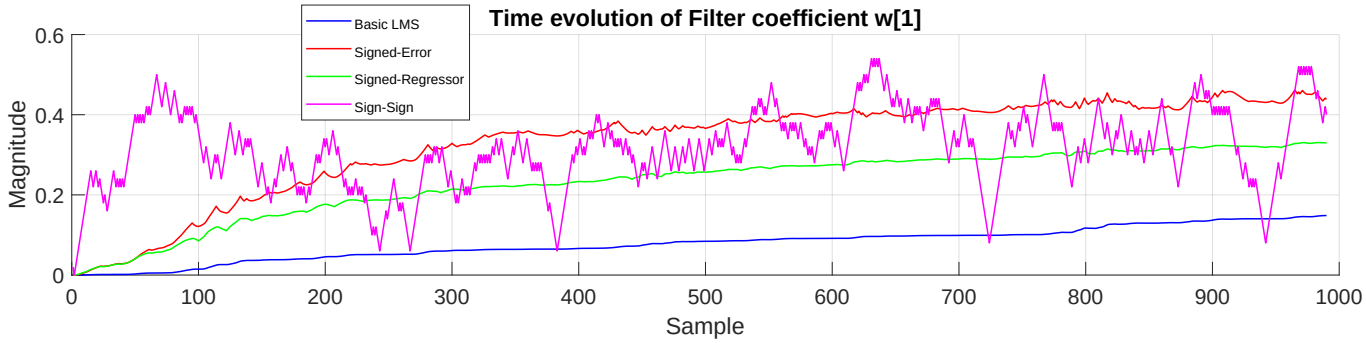


Figure 4.11: Time-Evolution of coefficient w[1] of sound "a" in section 4.5 using LMS, signed-error, signed-regressor and sign-sign algorithms.

5 MLE for the Frequency of a Signal

The pdf of a real signal $\mathbf{x}$, also called likelihood function, with expression $x[n] = Acos(2\pi f_0 n + \phi) + w[n]$, where w[n] is WGN $\sim \mathcal{N}(0, \sigma^2)$, parameterised by $\boldsymbol{\theta} = [A, f_0, \phi]^T$, can be expressed as:

$$p(\boldsymbol{x}; \boldsymbol{\theta}) = \frac{1}{(2\pi\sigma^2)^{N/2}} e^{-\frac{1}{2\sigma^2}\sum_{n=0}^{N-1}(x[n]-Acos(2\pi f_0 n+\phi))^2} = \frac{1}{(2\pi\sigma^2)^{N/2}} e^{-\frac{1}{2\sigma^2}J(\boldsymbol{\theta})} \tag{85}$$

where $A > 0$, $0 < f_0 < 0.5$ and $J(\boldsymbol{\theta})$ is the loss function. The maximum-likelihood estimate (MLE), $\hat{f}_0$, of frequency f_0 is the one that maximizes the pdf above, or equivalently minimizes the loss function.

5.1 - Using the trigonometric identity $cos(\alpha+\beta) = cos(\alpha)cos(\beta) - sin(\alpha)sin(\beta)$ we can express the loss function $J(\boldsymbol{\theta})$ as:

$$J(\boldsymbol{\theta}) = \sum_{n=0}^{N-1}(x[n] - (Acos(2\pi f_0 n)cos(\phi) - Asin(2\pi f_0 n)sin(\phi)))^2 \tag{86}$$

Using the substitutions $\alpha_1 = Acos(\phi)$ and $\alpha_2 = -Asin(\phi)$, $J(\boldsymbol{\theta})$ can be mapped into $J'(\alpha_1, \alpha_2, f_0)$:

$$J'(\alpha_1, \alpha_2, f_0) = \sum_{n=0}^{N-1}(x[n] - \alpha_1 cos(2\pi f_0 n) - \alpha_2 sin(2\pi f_0 n))^2 \tag{87}$$

We can express $J'(\alpha_1, \alpha_2, f_0)$ as $J'(\alpha_1, \alpha_2, f_0) = \boldsymbol{B}^T\boldsymbol{B}$, where $\mathbf{B}$ is defined as:

$$\begin{aligned}\boldsymbol{B} &= \begin{bmatrix} x[0] - \alpha_1 cos(0) - \alpha_2 sin(0) \\ \vdots \\ x[N-1] - \alpha_1 cos(2\pi f_0(N-1)) - \alpha_2 sin(2\pi f_0(N-1)) \end{bmatrix} \\ &= \begin{bmatrix} x[0] \\ \vdots \\ x[N-1] \end{bmatrix} - \alpha_1 \begin{bmatrix} 1 \\ \vdots \\ cos(2\pi f_0(N-1)) \end{bmatrix} - \alpha_2 \begin{bmatrix} 0 \\ \vdots \\ sin(2\pi f_0(N-1)) \end{bmatrix} = \boldsymbol{x} - \alpha_1\boldsymbol{c} - \alpha_2\boldsymbol{s}\end{aligned} \tag{88}$$

where $\mathbf{c}$ is the cosine vector, $\boldsymbol{c} = [1, cos(2\pi f_0), ..., cos(2\pi f_0(N-1))]^T$ and $\mathbf{s}$ is the sine vector, $\boldsymbol{s} = [0, sin(2\pi f_0), ..., sin(2\pi f_0(N-1))]^T$ and $\mathbf{s}$. Therefore, $J'(\alpha_1, \alpha_2, f_0)$ is given by:

$$J'(\alpha_1, \alpha_2, f_0) = (\boldsymbol{x} - \alpha_1\boldsymbol{c} - \alpha_2\boldsymbol{s})^T(\boldsymbol{x} - \alpha_1\boldsymbol{c} - \alpha_2\boldsymbol{s}) \tag{89}$$

Equivalently, we can express $\mathbf{B}$ in the form

$$\begin{aligned}\boldsymbol{B} &= \begin{bmatrix} x[0] \\ \vdots \\ x[N-1] \end{bmatrix} - \begin{bmatrix} \alpha_1 cos(0) + \alpha_2 sin(0) \\ \vdots \\ \alpha_1 cos(2\pi f_0(N-1)) + \alpha_2 sin(2\pi f_0(N-1)) \end{bmatrix} \\ &= \begin{bmatrix} x[0] \\ \vdots \\ x[N-1] \end{bmatrix} - \begin{bmatrix} cos(0) & sin(0) \\ \vdots & \vdots \\ cos(2\pi f_0(N-1)) & sin(2\pi f_0(N-1)) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \boldsymbol{x} - \boldsymbol{H}\boldsymbol{\alpha}\end{aligned} \tag{90}$$

where $\boldsymbol{H} = [\boldsymbol{c}, \boldsymbol{s}]$ and $\boldsymbol{\alpha} = [\alpha_1, \alpha_2]^T$. This means that

$$J'(\alpha_1, \alpha_2, f_0) = (\boldsymbol{x} - \boldsymbol{H}\boldsymbol{\alpha})^T(\boldsymbol{x} - \boldsymbol{H}\boldsymbol{\alpha}) = J'(\boldsymbol{\alpha}, f_0) \tag{91}$$

5.2 - The minimising solution $\hat{\boldsymbol{\alpha}}$ to $J'(\boldsymbol{\alpha}, f_0)$ can be found by first expanding the equation for loss function:

$$J'(\boldsymbol{\alpha}, f_0) = (\boldsymbol{x}^T - \boldsymbol{\alpha}^T\boldsymbol{H}^T)(\boldsymbol{x} - \boldsymbol{H}\boldsymbol{\alpha}) = \boldsymbol{x}^T\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{H}\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\boldsymbol{H}^T\boldsymbol{x} + \boldsymbol{\alpha}^T\boldsymbol{H}^T\boldsymbol{H}\boldsymbol{\alpha} = \boldsymbol{x}^T\boldsymbol{x} - 2\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\boldsymbol{H}^T\boldsymbol{H}\boldsymbol{\alpha} \tag{92}$$

where we used the fact that $\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{\alpha} = (\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{\alpha})^T = \boldsymbol{\alpha}^T\boldsymbol{H}^T\boldsymbol{x}$. By taking the derivative of this function with respect to vector $\boldsymbol{\alpha}$, we obtain:

$$\frac{\partial J'(\boldsymbol{\alpha}, f_0)}{\partial \boldsymbol{\alpha}} = -2\boldsymbol{H}^T\boldsymbol{x} + \left[(\boldsymbol{H}^T\boldsymbol{H}) + (\boldsymbol{H}^T\boldsymbol{H})^T\right]\boldsymbol{\alpha} = -2\boldsymbol{H}^T\boldsymbol{x} + 2\boldsymbol{H}^T\boldsymbol{H}\boldsymbol{\alpha} \tag{93}$$

since matrix $\boldsymbol{H}^T\boldsymbol{H}$ is symmetric. Setting the derivative to zero, yields:

$$\begin{aligned}2\boldsymbol{H}^T\boldsymbol{x} &= 2\boldsymbol{H}^T\boldsymbol{H}\hat{\boldsymbol{\alpha}} \\ \hat{\boldsymbol{\alpha}} &= (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{x}\end{aligned} \tag{94}$$

Substituting $\hat{\boldsymbol{\alpha}}$ back into the loss function, we obtain the minimum value

$$\begin{aligned}J'(\hat{\boldsymbol{\alpha}}, f_0) &= (\boldsymbol{x} - \boldsymbol{H}\hat{\boldsymbol{\alpha}})^T(\boldsymbol{x} - \boldsymbol{H}\hat{\boldsymbol{\alpha}}) = (\boldsymbol{I}\boldsymbol{x} - \boldsymbol{H}\hat{\boldsymbol{\alpha}})^T(\boldsymbol{I}\boldsymbol{x} - \boldsymbol{H}\hat{\boldsymbol{\alpha}}) = \left[\boldsymbol{I}\boldsymbol{x} - \boldsymbol{H}(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{x}\right]^T\left[\boldsymbol{I}\boldsymbol{x} - \boldsymbol{H}(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{x}\right] \\ &= \left[\boldsymbol{x}^T\boldsymbol{I} - \boldsymbol{x}^T(\boldsymbol{H}(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T)^T\right]\left[\boldsymbol{I}\boldsymbol{x} - \boldsymbol{H}(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{x}\right] = \boldsymbol{x}^T\left[\boldsymbol{I} - (\boldsymbol{H}(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T)^T\right]\left[\boldsymbol{I} - \boldsymbol{H}(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\right] \\ &= \boldsymbol{x}^T\left[\boldsymbol{I} - \boldsymbol{H}(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\right]^T\left[\boldsymbol{I} - \boldsymbol{H}(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\right]\boldsymbol{x}\end{aligned} \tag{95}$$

where $\boldsymbol{I}$ is the N-by-N identity matrix. It should be noted that the matrix $\boldsymbol{I} - \boldsymbol{H}(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T$ is a square (N-by-N), symmetric and idempotent matrix, i.e. the matrix is equal to its transpose and additionally when multiplied by itself, yields itself. Using this we can further express the transformed loss function as:

$$J'(\hat{\boldsymbol{\alpha}}, f_0) = \boldsymbol{x}^T\left[\boldsymbol{I} - \boldsymbol{H}(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\right]\boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{H}(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{x} - J_2'(\boldsymbol{\alpha}, f_0) \tag{96}$$

Therefore, in order to minimize the function $J'(\hat{\boldsymbol{\alpha}}, f_0)$, it means that we can equivalently maximize $J_2'(\hat{\boldsymbol{\alpha}}, f_0) = \boldsymbol{x}^T\boldsymbol{H}(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{x}$, since matrix $\boldsymbol{x}^T\boldsymbol{x}$ is independent of parameters.

5.3 - We have seen that we need to maximize the matrix $\boldsymbol{x}^T\boldsymbol{H}(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{x}$, or $(\boldsymbol{H}^T\boldsymbol{x})^T(\boldsymbol{H}^T\boldsymbol{H})^{-1}(\boldsymbol{H}^T\boldsymbol{x})$. So using the definition of $\mathbf{H}$, i.e. $\boldsymbol{H}[\boldsymbol{c}, \boldsymbol{s}]$, we can express:

- $\boldsymbol{H}^T\boldsymbol{x} = \begin{bmatrix} cos(0) \dots cos(2\pi f_0(N-1)) \\ sin(0) \dots sin(2\pi f_0(N-1)) \end{bmatrix} \begin{bmatrix} x[0] \\ \vdots \\ x[N-1] \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}^T \\ \boldsymbol{s}^T \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{c}^T\boldsymbol{x} \\ \boldsymbol{s}^T\boldsymbol{x} \end{bmatrix}$
- $\boldsymbol{H}^T\boldsymbol{H} = \begin{bmatrix} cos(0) \dots cos(2\pi f_0(N-1)) \\ sin(0) \dots sin(2\pi f_0(N-1)) \end{bmatrix} \begin{bmatrix} cos(0) & sin(0) \\ \vdots & \vdots \\ cos(2\pi f_0(N-1)) & sin(2\pi f_0(N-1)) \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}^T \\ \boldsymbol{s}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{c} & \boldsymbol{s} \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}^T\boldsymbol{c} & \boldsymbol{c}^T\boldsymbol{s} \\ \boldsymbol{s}^T\boldsymbol{c} & \boldsymbol{s}^T\boldsymbol{s} \end{bmatrix}$

Therefore, the MLE of the frequency f_0 is the value that maximizes

$$(\boldsymbol{H}^T\boldsymbol{x})^T(\boldsymbol{H}^T\boldsymbol{H})^{-1}(\boldsymbol{H}^T\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{c}^T\boldsymbol{x} \\ \boldsymbol{s}^T\boldsymbol{x} \end{bmatrix}^T \begin{bmatrix} \boldsymbol{c}^T\boldsymbol{c} & \boldsymbol{c}^T\boldsymbol{s} \\ \boldsymbol{s}^T\boldsymbol{c} & \boldsymbol{s}^T\boldsymbol{s} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{c}^T\boldsymbol{x} \\ \boldsymbol{s}^T\boldsymbol{x} \end{bmatrix} \tag{97}$$

5.4 - Under the assumption that f_0 is not close to 0 or 0.5, equation above can be approximated by

$$\begin{bmatrix} \boldsymbol{c}^T\boldsymbol{x} \\ \boldsymbol{s}^T\boldsymbol{x} \end{bmatrix}^T \begin{bmatrix} \frac{N}{2} & 0 \\ 0 & \frac{N}{2} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{c}^T\boldsymbol{x} \\ \boldsymbol{s}^T\boldsymbol{x} \end{bmatrix} \tag{98}$$

The reason for f_0 not being close to 0 or 0.5, can be explained by considering if the central matrix $\boldsymbol{H}^T\boldsymbol{H}$, is invertible. By expanding out the matrix $\boldsymbol{H}^T\boldsymbol{H}$ we can consider the case of $f_0 = 0$ and $f_0 = 0.5$.

$$\boldsymbol{H}^T\boldsymbol{H} = \begin{bmatrix} \boldsymbol{c}^T\boldsymbol{c} & \boldsymbol{c}^T\boldsymbol{s} \\ \boldsymbol{s}^T\boldsymbol{c} & \boldsymbol{s}^T\boldsymbol{s} \end{bmatrix}^{-1} \tag{99}$$

- $f_0 = 0$

$$\boldsymbol{H}^T\boldsymbol{H} = \begin{bmatrix} 1 & \dots & 1 \\ 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} 1 & \dots & 1 \\ 0 & \dots & 0 \end{bmatrix}^T = \begin{bmatrix} N & 0 \\ 0 & 0 \end{bmatrix} \tag{100}$$

- $f_0 = 0.5$

$$\boldsymbol{H}^T\boldsymbol{H} = \begin{bmatrix} 1 & -1 & \dots & -1 & 1 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & -1 & \dots & -1 & 1 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}^T = \begin{bmatrix} N & 0 \\ 0 & 0 \end{bmatrix} \tag{101}$$

In both cases, matrix $\boldsymbol{H}^T\boldsymbol{H}$ is singular (has determinant equal to zero), therefore it is not invertible (inverse is non-finite) and thus the MLE is not defined. Therefore, using the approximation of $f_0 \approx 0.25$:

$\boldsymbol{c}^T\boldsymbol{s} = \boldsymbol{s}^T\boldsymbol{c}$

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

$\boldsymbol{c}^T\boldsymbol{c} =$

$$= \sum_{n=0}^{N-1} cos^2(2\pi f_0 n)$$
$$= \frac{1}{2}\sum_{n=0}^{N-1} (1 + cos(4\pi f_0 n)) \approx \frac{N}{2}$$

$\boldsymbol{s}^T\boldsymbol{s} =$

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

As a result, expression (65) can be approximated by

$$\begin{bmatrix} \boldsymbol{c}^T\boldsymbol{x} & \boldsymbol{s}^T\boldsymbol{x} \end{bmatrix} \begin{bmatrix} \frac{2}{N} & 0 \\ 0 & \frac{2}{N} \end{bmatrix} \begin{bmatrix} \boldsymbol{c}^T\boldsymbol{x} \\ \boldsymbol{s}^T\boldsymbol{x} \end{bmatrix} = \frac{2}{N}\left[(\boldsymbol{c}^T\boldsymbol{x})^2 + (\boldsymbol{s}^T\boldsymbol{x})^2\right] = \frac{2}{N}\left[\left(\sum_{n=0}^{N-1} x[n]cos(2\pi f_0 n)\right)^2 + \left(\sum_{n=0}^{N-1} x[n]sin(2\pi f_0 n)\right)^2\right]$$

$$= \frac{2}{N}\left|\sum_{n=0}^{N-1} x[n]e^{-j2\pi f_0 n}\right|^2 = 2\hat{P}_X(f_0)\ ,\ \textit{whre}\ \hat{P}_X(f_0)\ \textit{is the periodogram.}$$

Therefore, the MLE $\hat{f}_0$ can be found by maximizing the periodogram. In other words,

$$\hat{f}_0 = \underset{f}{\text{argmax}}\, \hat{P}_X(f) = \underset{f}{\text{argmax}}\, \frac{1}{N}\left|\sum_{n=0}^{N-1} x[n]e^{-j2\pi f_0 n}\right|^2 \tag{102}$$

5.5 - Using the noiseless data $x[n] = cos(2\pi f_0 n)$, for n=0,1,...,N-1 , the periodogram and corresponding MLE estimate are plotted in figure below for varying $0 < f_0 < 0.5$.

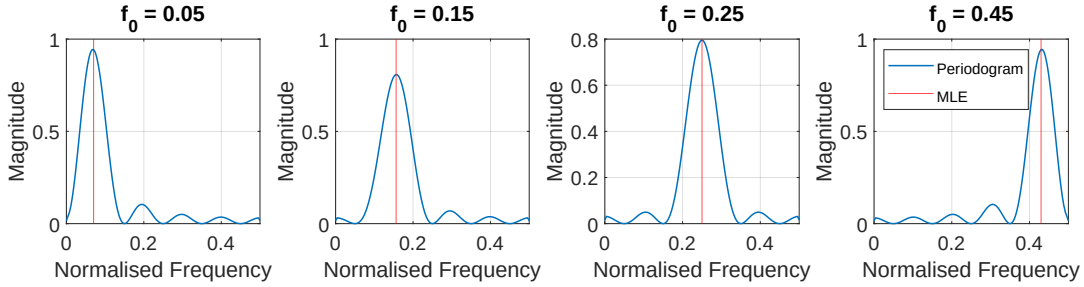


Figure 5.1: Periodogram and MLE estimates of noiseless data

MLE of frequency f_0 is given from the frequency location of the peak in the periodogram plot, plotted as a vertical red line. Fig. 5.1 shows that the closer f_0 is to 0 or 0.5, the MLE deviates further away from the true value. Only at frequency 0.25 the MLE is exactly the same as the actual value. This was verified by computing the absolute error between the MLE and actual f_0 value, summarised in table below.

Table 15: Absolute Error for MLE for different f_0 values

Actual (f_0)	0.05	0.1	0.15	0.25	0.35	0.45
MLE($\hat{f}_0$)	0.0703	0.1133	0.1563	0.2500	0.3438	0.4297
Absolute Error($\lvert f_0 - \hat{f}_0 \rvert$)	0.0203	0.0133	0.00625	0	0.00625	0.0203