Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

Coursework of Adaptive Signal Processing & Machine Intelligence

---

REPORT

MSc Future Power Networks

Author:
M. Jawid Salimi
CID 01427514

Professor:
Danilo P. Mandic

April 10, 2019

Table of Contents

1 Classical and Modern Spectral Estimation

1.1 Properties of Power Spectral Density (PSD)

The definition of PSD is as shown in eq. 1.1.1, given in equation 9 in the coursework guideline:

$$P(\omega) = \lim_{n\to\infty} E\left\{\frac{1}{N}\left|\sum_{n=0}^{N-1} x(n)e^{-jn\omega}\right|^2\right\} \qquad 1.1.1$$

$$= \lim_{n\to\infty} E\left\{\frac{1}{N}\sum_{m=0}^{N-1} x(m)e^{-jm\omega} \sum_{n=0}^{N-1} x^*(n)e^{jk\omega}\right\}$$

$$= \lim_{n\to\infty} \frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} E\{x(m)x^*(n)e^{-jm\omega}e^{jn\omega}\}$$

$$= \lim_{n\to\infty} \frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} E\{x(m)x^*(n)\}\, e^{-j\omega(m-n)}$$

$$= \lim_{n\to\infty} \frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} r_{xx}(m-n)\, e^{-j\omega(m-n)} \qquad 1.1.2$$

Where $\tau = m - n$ and the double summation written as one summation:

$$P(\omega) = \lim_{n\to\infty} \frac{1}{N}\sum_{\tau=-N+1}^{N-1} (N - |\tau|) r_{xx}(\tau)\, e^{-j\omega\tau}$$

$$= \sum_{\tau=-\infty}^{\infty} r_{xx}(\tau)e^{-j\omega\tau} - \lim_{n\to\infty} \frac{1}{N}\sum_{\tau=-N+1}^{N-1} |\tau|\, r_{xx}(\tau)\, e^{-j\omega\tau} \qquad 1.1.3$$

Equation 10 given in the coursework guideline which is the same as part 2 of eq. 1.1.3, decays to zero under mild assumptions. Hence:

$$P(\omega) = \sum_{\tau=-\infty}^{\infty} r_{xx}(\tau)e^{-j\omega\tau} \qquad 1.1.4$$

Which is the DFDT of autocovariance function (ACF) given in equation 7 in the coursework guideline. Therefore, it is proved that both definitions of PSD are valued and equivalent.

1.2 Periodogram-based Methods Applied to Real-World Data

a) Figure 1.2.1 shows different realization of the Sunspot time series. Figure 1.2.2 shows the power spectral density (PSD) of this data using hamming window periodogram-based spectral estimation.
The black line shows the original data while the blue line is after removing the mean and detrending it whereas the red line is the plot in logarithmic scale of pre-processed data.
Removing the mean eliminates the effects of zero frequency while detrending the data eliminates lower frequencies ($f \leq 0.02Hz$). It can be confirmed from Figure 1.2.2 where the PSD is the same for the raw data and the removed mean and detrended data for frequencies $f > 0.02Hz$.

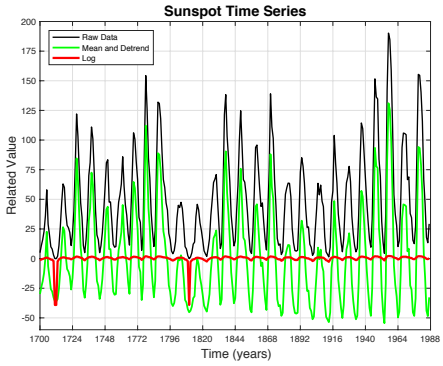


Figure 1.2.1: *Sunspot time series of raw and pre-processed data*

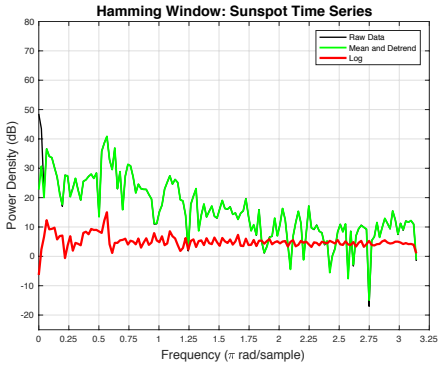


Figure 1.2.2: *Periodogram of the raw and pre-processed series using hamming window*

b) Figure 1.2.3 shows the PSD using the *periodogram* with rectangular windowing '*rectwin'*. This mean the rectangular window method is used to calculate the standard periodogram. Figure 1.2.4 shows the periodogram using '*Bartlet'* window for different window lengths $\Delta t = \{10s, 5s, 1s\}$.

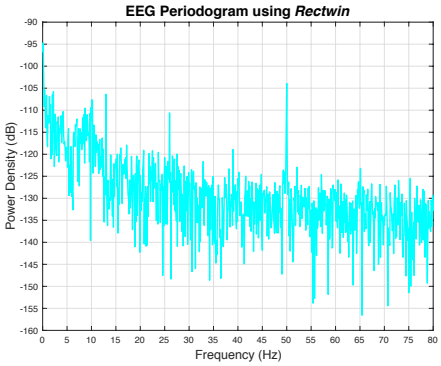


Figure 1.2.3: *EEG Periodogram using rectangular window*

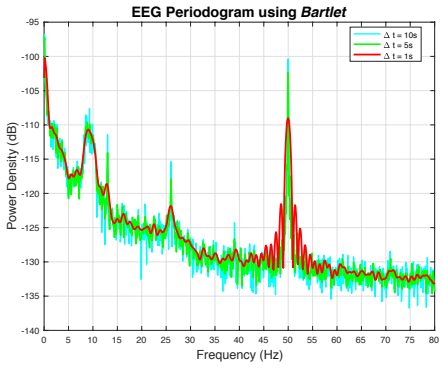


Figure 1.2.4: *EEG Periodograms using different length size bartlet windows*

Ignoring the peak in frequencies *[8-10]Hz* and *50Hz* which are due to external disturbances, one can see that the steady state visual evoked potential (SSVEP) at frequencies $13\text{Hz}, 26\text{Hz}, 39\text{Hz}, 52\text{Hz} \ldots (13n\ \text{Hz})\ n = 1,2,3, \ldots$

Figures 1.2.5 and 1.2.6 show how the variance and resolution affect each other. The standard periodogram results in greatest variance but less resolution. Whereas the windowed periodogram of length $\Delta t = 10s$ has less variance but higher resolution and this is even

more reinforced with $\Delta t = 1s$ where the variance decreases further and the peaks of frequencies 39Hz, 52Hz ... (13n Hz) $n \geq 3$ is hardly observable.

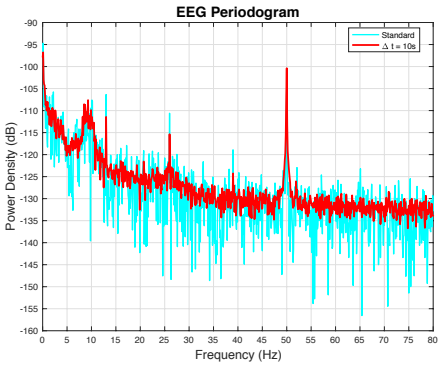


Figure 1.2.5: *EEG periodogram averaging window* $\Delta t = 10s$

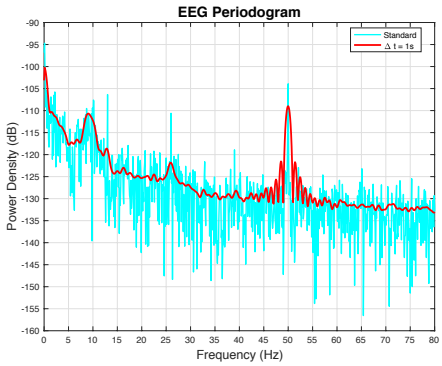


Figure 1.2.6: *EEG periodogram averaging window* $\Delta t = 1s$

1.3 Correlation Estimation

a) Figures 1.3.1 - 1.3.3 show the autocorrelation function (ACF), r_k in both biased (red) and unbiased (cyan) colors for white gaussian noise (WGN), noisy sine wave and filtered WGN, respectively.
Figures 1.3.4 - 1.3.6 show the correlogram for these signals in both biased (red) and unbiased (cyan) cases.

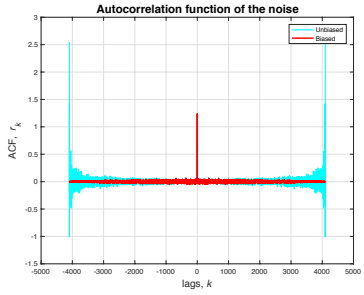


Figure 1.3.1: *ACF of the white gaussian noise (WGN)*

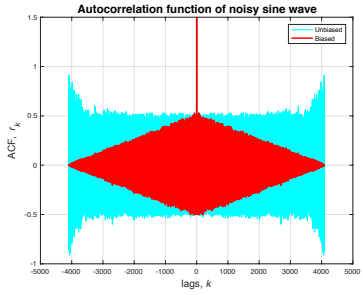


Figure 1.3.2: *ACF of sine wave with normal distributed noise*

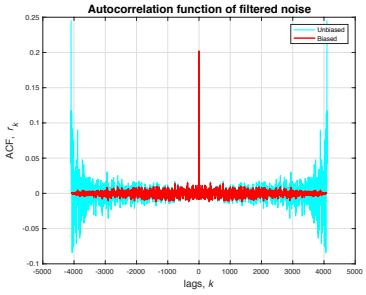


Figure 1.3.3: *ACF of filtered WGN*

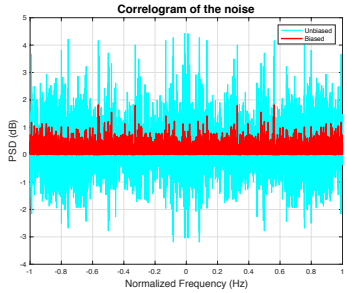


Figure 1.3.4: *Correlogram of WGN*

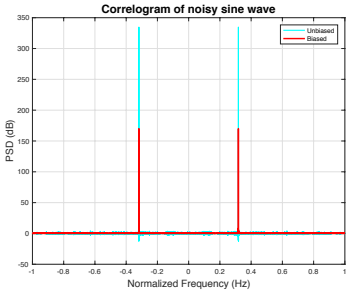


Figure 1.3.5: *Correlogram of noisy sine wave*

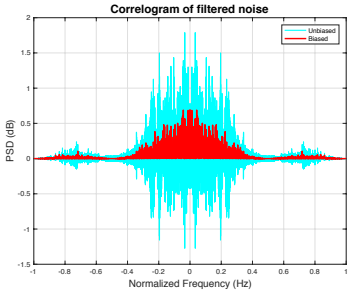


Figure 1.3.6: *Correlogram of filtered WGN*

The figures showing ACF are identical for lower values of the lags while their difference becomes more evident as k increases. By zooming in, one can see that the difference starts to show when $k > 200$. The biased estimate fades out to zero while the unbiased estimate values increase as k increases. This is due to the use of the *Bartlet* window for the biased estimation and *rectangular* window for the unbiased estimation. The DFT of rectangular window being a '*sinc'* function, the correlogram shows both positive and negative power which is not possible. Whereas the biased estimation shows only positive values as shown in the correlogram of the signals considered for all three cases.

b) Figure 1.3.7 shows 200 different iterations of the signal given below $x(n)$ composed of three different sinusoids with frequencies $0.4Hz, 0.8Hz$ & $1.2Hz$ with WGN of zero mean and variance 1. The mean of these different realization is shown in the same figure in red. The standard deviation is shown in figure 1.3.8 for the different iterations of this signal.

$$x(n) = \sin(2\pi(\mathbf{0.4})n) + 2\sin(2\pi(\mathbf{0.8})n) + \sin(2\pi(\mathbf{1.2})n) + w(n) \quad w \sim \mathcal{N}(0,1)$$

The correlogram becomes an inconsistent estimator when the biased ACF is used. As a result, the asymptotic property is violated and the variance of the estimate diverges as the number of iterations increases. This is also evident from the standard deviation in figure 1.3.8.

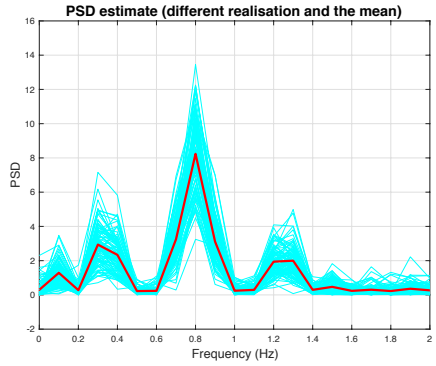


Figure 1.3.7: *Correlogram and mean of different realizations*

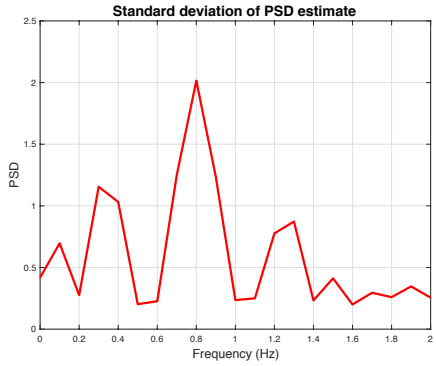


Figure 1.3.8: *standard deviation of different realizations*

c) Figure 1.3.9 shows the same signal $x(n)$ in a logarithmic scale. Notice that the interpretation of the frequencies of interest $0.4Hz, 0.8Hz$ & $1.2Hz$ is now much easier. However, if there are any other disturbances, they will be also amplified as for the frequencies closer to zero.

d) Figure 1.3.11 and figure 1.3.12 show considering different data samples for the signal $x(n)$.

$$x(n) = e^{j2\pi(0.3)n} + e^{j2\pi(0.32)n} + noise$$

Where the frequencies are $f_1 = 0.3Hz$ and $f_2 = 0.32Hz$, respectively. The different data samples considered results in different resolution. Therefore, the peak can not be visible until we reach a certain number of samples. In figure 1.3.12, it can be seen that the peaks for f_1 and f_2 start to show when the samples are larger than 40. In theory, we have the resolution of the periodogram as $\Delta f = \frac{0.89}{N}$ where $\Delta f = f_2 - f_1 = 0.32 - 0.3 = 0.02Hz$. Hence the number of samples requires will be:

$$N = \frac{0.89}{\Delta f} = \frac{0.89}{0.02} = 44.5$$

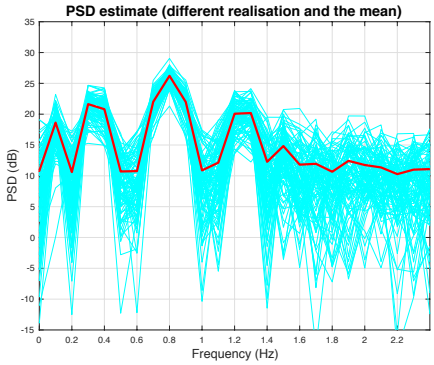


Figure 1.3.9: *Correlogram and mean of different realizations, dB*

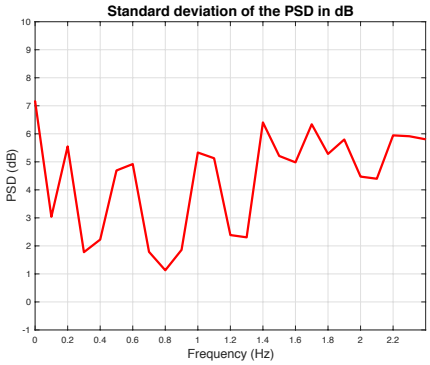


Figure 1.3.10: *standard deviation of different realizations, dB*

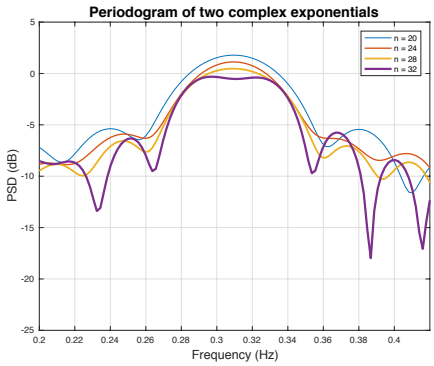


Figure 1.3.11: *peak observation of two complex exponentials - $n \in \{20, 24, 28, 32\}$*

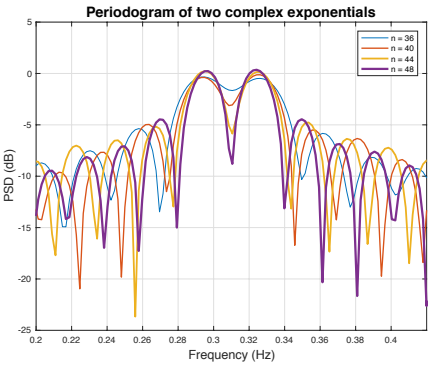


Figure 1.3.12: *peak observation of two complex exponentials - $n \in \{36, 40, 44, 48\}$*

e) The multiple signal classification (MUSIC) algorithm is one of the modern spectral estimations using the subspace method for spectral estimation. This method uses a modified autocorrelation matrix $\boldsymbol{R_{xx}}$, where:

$$x(n) = A_1 e^{jn\omega_1} + w(n)$$
$$\boldsymbol{x} = A_1\boldsymbol{e_1} + \boldsymbol{w}$$
$$\boldsymbol{R_{xx}} = E(\boldsymbol{x}\boldsymbol{x}^H) = |A_1|^2\boldsymbol{e_1}\boldsymbol{e_1^H} + \sigma_w^2\boldsymbol{I}$$

Where, $\boldsymbol{R_s} = |A_1|^2\boldsymbol{e_1}\boldsymbol{e_1^H}$ is Hermitian and the remaining $M-1$ eigenvectors are orthogonal, hence $e_1^H v_i = 0, i = 2, \dots, M$. And $\boldsymbol{R_n} = \sigma_w^2\boldsymbol{I}$.
Extending to 'p' sinusoids, $\boldsymbol{R_{xx}} = \boldsymbol{E}\boldsymbol{P}\boldsymbol{E}^H + \sigma_w^2\boldsymbol{I}$ where $E = \left[e_1, \dots, e_p\right]$, $P = diag\left(|A_1|^2, \dots, |A_p|^2\right)$. Using this, the MUSIC algorithm is then formulated as:

$$\hat{P}_{MU}(\omega) = \frac{1}{\sum_{i=p+1}^{M} |e^H v_i|^2} \qquad 1.3.1$$

All other terms except the ones with frequencies $\omega_1, \omega_2, \dots, \omega_p$ will become zero because of the orthogonality and hence showing the desired peaks in the spectrum.
The first line of the code given shows the modified $\boldsymbol{R_{xx}}$ used by the MUSIC method with $M = 14$. The term 'mod' lets the command know to return this modified matrix. The second line of the code is the MUSIC algorithm with inputs $\boldsymbol{R_{xx}}$, the modified autocorrelation matrix and p, which is the hyperparameter that is equal to 2 in this case.

Figure 1.3.13 shows the overlay of 100 realizations of a two complex exponential signal with frequencies $f_1 = 0.3Hz$ and $f_2 = 0.32Hz$. Figure 1.3.14 shows the standard deviation of these estimates of the MUSIC algorithm.

Advantage of using MUSIC algorithm is that despite having smaller number of samples, in this case $n = 30$, the algorithm clearly identifies the peaks of the signal at $f_1 = 0.3Hz$ and $f_2 = 0.32Hz$ when $p = 2$, whereas the standard periodogram was not able to identify the peaks with this number of sample and needed at least 44 samples to identify the peaks. On the other hand, the need for the prior of knowledge of p is a disadvantage of the MUSIC algorithm and in case of wrong number of p as seen in figure 1.3.13, the estimates are not reliable.

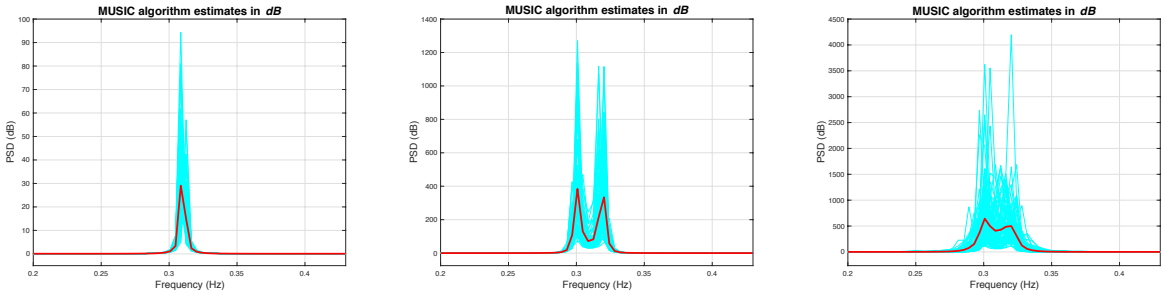


Figure 1.3.13: *MUSIC spectral estimate of two complex exponentials*

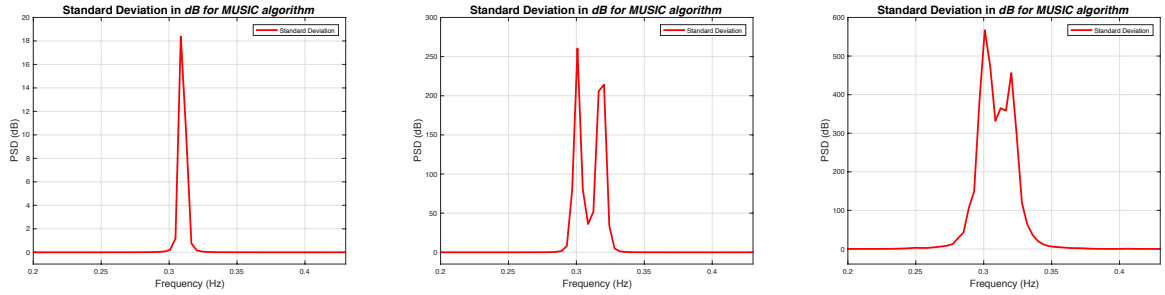


Figure 1.3.14: *standard deviation of the relevant MUSIC estimate*

1.4 Spectrum of Autoregressive Processes

a) AR coefficients are found using the Yule-Walker equations shown in the lecture notes and also in the coursework guideline. Hence:

$$\boldsymbol{r}_x = \boldsymbol{R}_x * \boldsymbol{a} \implies \boldsymbol{a} = \boldsymbol{R}_x^{-1} * \boldsymbol{r}_x$$

This clearly shows that $\boldsymbol{R}_x$ should be always a positive definite matrix which guaranties matrix inversion. This property is governed by the biased estimate as depicted in figures 1.3.2, 1.3.4 and 1.3.6. Whereas the unbiased estimate lacks this property and we can see that $\boldsymbol{R}_x$ loses its positive definiteness in the process.

b) Before we look at the results in this part or the next, an AR process of order n can be estimated with an AR model of order n Aor above. Number of samples used can also affect this number. The process given by:

$$x(n) = 2.76x(n-1) - 3.8x(n-2) + 2.56x(n-3) - 0.92x(n-4) + w(n)$$
$$w \sim \mathcal{N}(0,1)$$

is of order 4. Therefore, our models of interest should be higher than order 3.
For 500 samples, the models AR(4), AR(7) and AR(14) against the true model, are shown in figure 1.4.1. AR models of order 2-4 cannot capture the true model $x(n)$. Theoretically, AR(4) should be able to capture the true model. However, due to insufficient number of samples, this becomes impossible and it can be seen that the estimate shows only one peak while the true model has two peaks. With the same number of samples, higher AR models start to capture the true model.

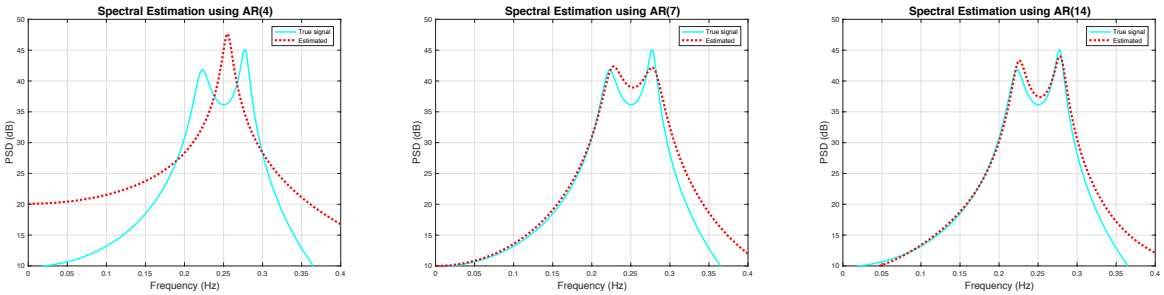


Figure 1.4.1: *Spectrum of AR(4), AR(7) and AR(11) – N = 500*

c) The arguments made in previous section can be readily confirmed by observing the results in this part. All models lower than order 4 can't capture the true model no matter how much the number of samples is increased. However, performance of AR(4) can become satisfactory with larger number of samples. With 10,000 samples, the estimates remain the same for AR of order 5 and above. Figure 1.4.2 shows the results and reflect the comments made above.

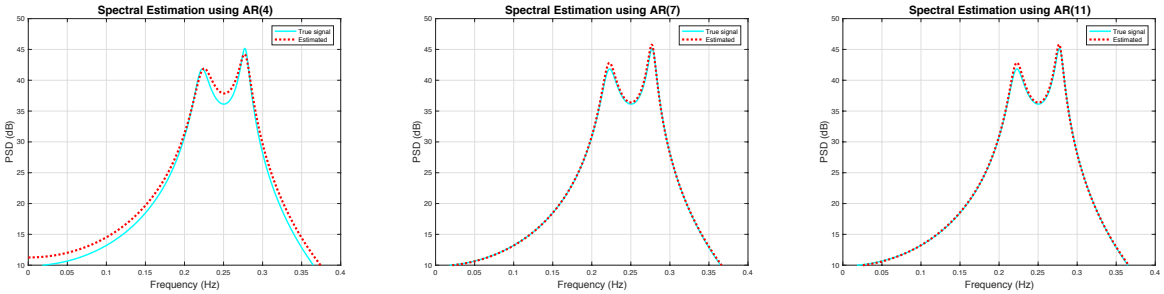


Figure 1.4.2: *Spectrum of AR(4), AR(7) and AR(11) – N = 9,500*

1.5 Real World Signals: Respiratory Sinus Arrhythmia from RR-Intervals

a) Figure 1.5.1 shows the standard and averaged periodograms using different window lengths of (e.g. 50s, 100s, 150s and 200s) for trial-1, trial-2 and trial-3 of the RRI data obtained from the three trials.

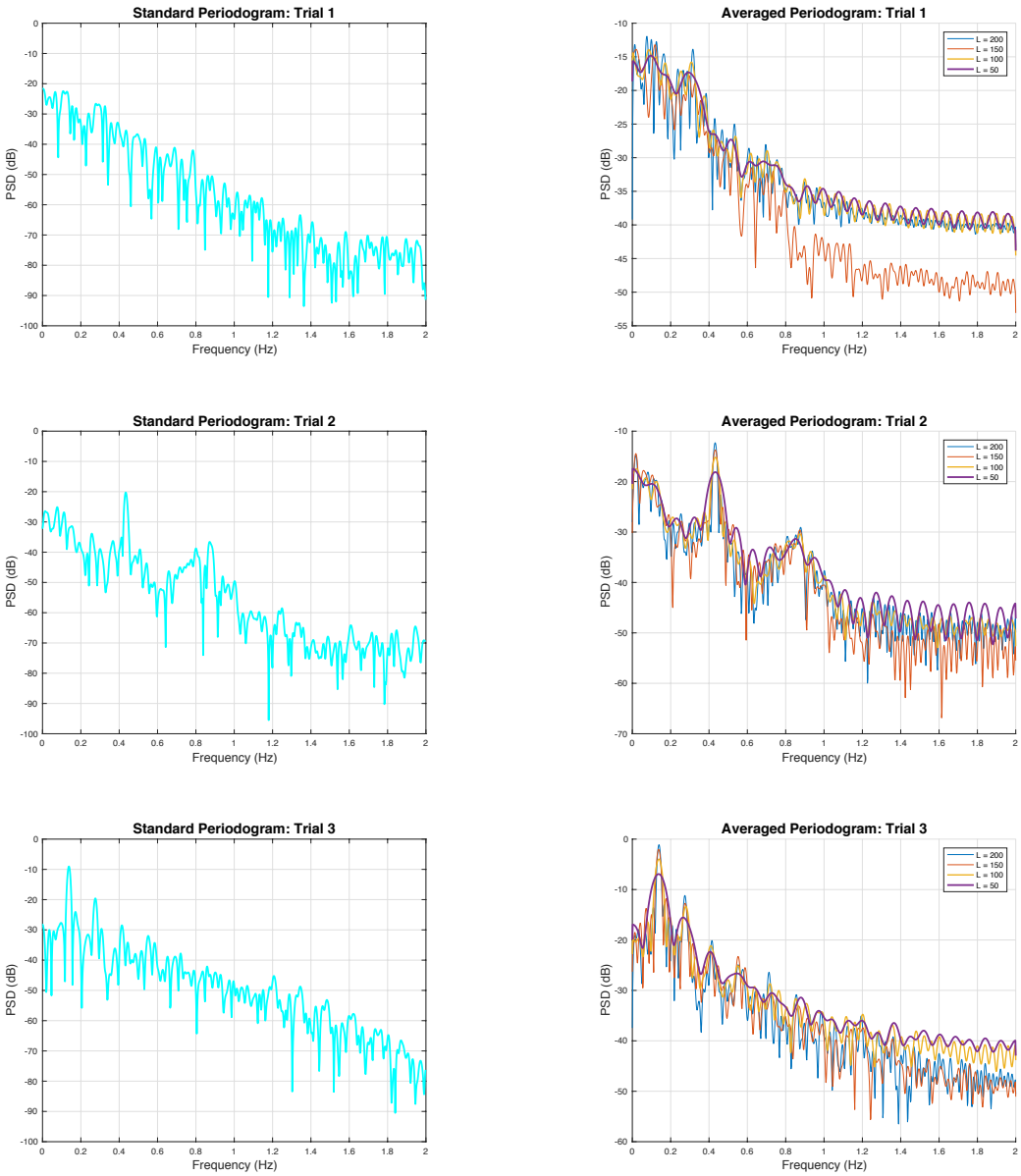


Figure 1.5.1: *Standard and averaged periodograms*

b) The three trials were conducted under different conditions:

1. Trial 1: unconstrained breathing – respiration rate 12-20 breaths per minute (BPM)
2. Trial 2: fast breathing – respiration rate 25 BPM.
3. Trial 3: slow breathing – respiration rate 7.5 BPM.

The periodograms in figure 1.5.1 can be studied to confirm the above results:

1. Trial 1: peak at $0.325Hz$, resulting in respiration rate $60 \times 0.325 = 19.5 \approx 20BPM$
2. Trial 2: peak at $0.416Hz$, resulting in respiration rate $60 \times 0.416 = 24.96 \approx 25BPM$
3. Trial 3: peak at $0.125Hz$, resulting in respiration rate $60 \times 0.125 = 7.5BPM$

The averaged periodograms shows the peaks clearer than the standard periodogram because of having less variance but higher resolution. This is achieved by averaging different numbers of the periodograms means. The peaks and its harmonics are visible for the values noted above. These values are also readable from the standard periodograms, however, its

2nd and 3rd harmonics are hardly readable from the standard periodogram whereas the averaged periodograms show the 2nd and 3rd harmonics clearly.

c) The AR spectrum estimate is performed for the three trials and the results are shown for trail 1 in figure 1.5.2, trial 2 in figure 1.5.3 and trial 3 in figure 1.5.4.
For the first trial of RRI data, the fundamental and second harmonic can be observed with AR of order 13 and above. The lower orders can't identify fundamental or second harmonics accurately.

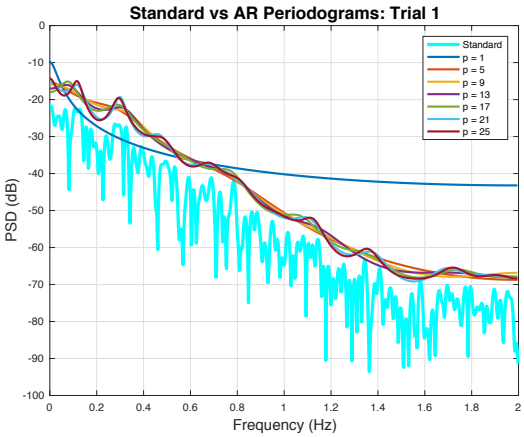


Figure 1.5.2: *AR spectrum estimate – trial 1*

For the second trial of RRI data, the fundamental and second, third and fourth harmonics can be observed with AR of order 13 and above. AR of order 9 can only identify the fundamental and second harmonic.

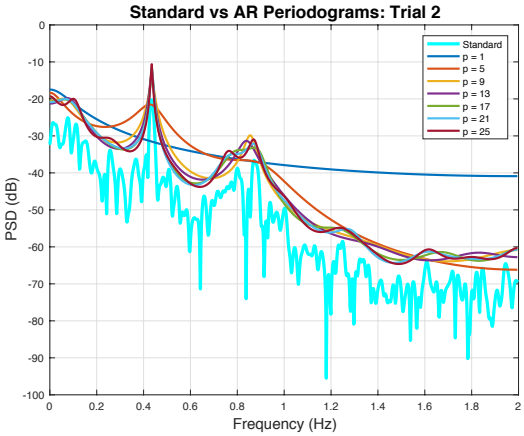


Figure 1.5.3: *AR spectrum estimate – trial 2*

For the third trial of RRI data, the fundamental and all of the first six harmonics are identified by AR of order 17 and above.

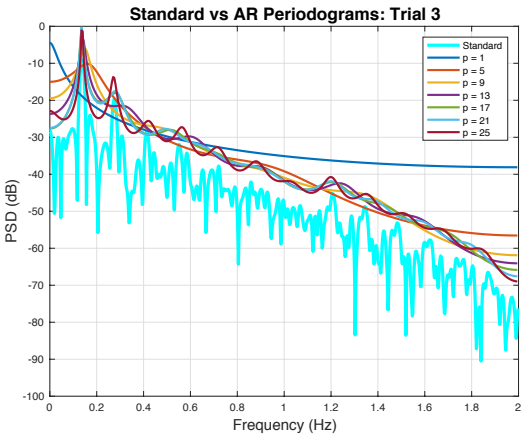


Figure 1.5.4: *AR spectrum estimate – trial 3*

For AR spectrum estimate, the results are reliable and accurate for the optimal order numbers or around those numbers. In case of under modelling, the AR estimate can't estimate the frequency peaks while in case of over modelling, the AR estimate may introduce new peaks which are due to the presence of noise. However, the problem of variance and periodogram resolution do not exist in AR estimate.

1.6 Robust Regression

a) Figure 1.6.1 shows the singular values of $\mathbf{X}$ and $\mathbf{X_{noise}}$. We can see from the input data $\mathbf{X}$, the rank to be 3 because of the existence of 3 non-zero singular values. On the same plot, the noise data is plotted and shown in red. The three first singular values in $\mathbf{X_{noise}}$ is due to the signal subspace whereas the rest of the indices for $\mathbf{X_{noise}}$ are roughly half the value of the first three values and span around noise subspace. Figure 1.6.2 shows the $error\ squared\ (\boldsymbol{X} - \boldsymbol{X_{noise}})^2$ for each singular value. Notice that the three first values can be set apart by a large difference due to the signal first three values being two times the values of the noise. If this difference is lower or the difference between the first values is larger, identifying the rank of $\boldsymbol{X_{noise}}$ will become troublesome.

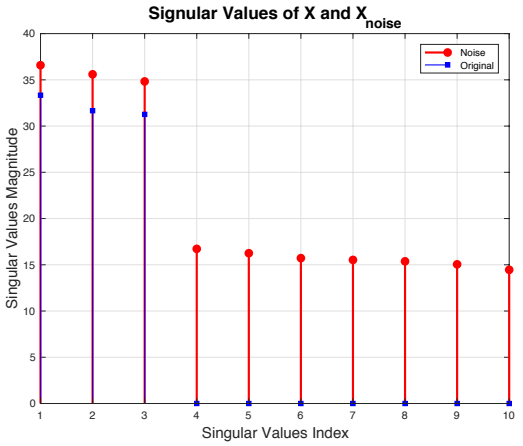


Figure 1.6.1: *Singular values of matrices* ***X*** *and* $\boldsymbol{X_{noise}}$

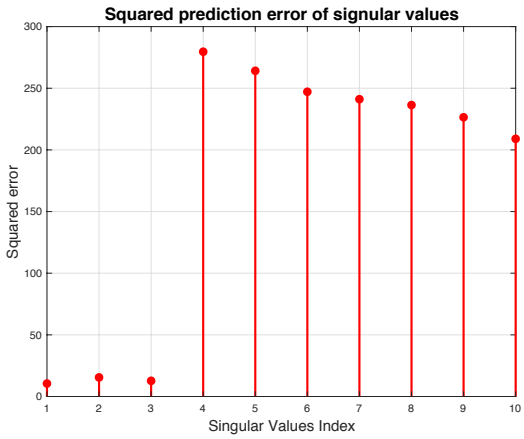


Figure 1.6.2: *Squared error between each value of matrices* ***X*** *and* $\boldsymbol{X_{noise}}$

b) Low rank approximation can be formulated as an optimization problem as follows:

$$\min \left\| \boldsymbol{X}_{noise} - \widetilde{\boldsymbol{X}}_{noise} \right\|$$

$$subject\ to:\ \ rank\left(\tilde{\boldsymbol{X}}_{noise}\right) \leq r$$

Low rank approximation is based on the fact that *r* most significant components are related with the data signal and the rest of $\tilde{X}_{noise}$ is noise. Observing the results in figure 1.6.3, it can be seen that the error is minimized at *r=3* which was identified as the rank of the matrix $\boldsymbol{X}_{noise}$.

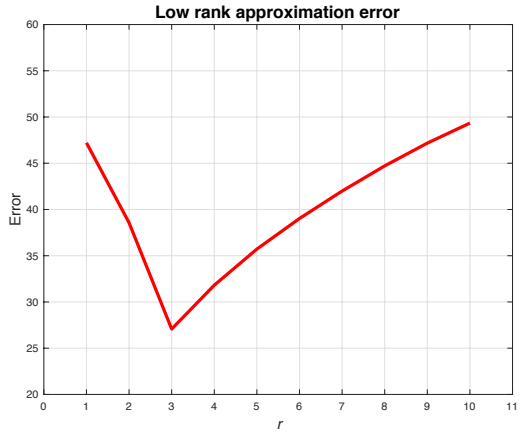


Figure 1.6.3: *Square prediction error of* $\boldsymbol{X}_{noise}$ *using low rank approximation*

c) $\hat{B}_{OLS}$ and $\hat{B}_{PCR}$ are calculated using:

$$\hat{B}_{OLS} = (X_{noise}^T X_{noise})^{-1} X_{noise}^T Y \qquad 1.6.1$$

$$\hat{B}_{PCR} = V_{1:r}(\Sigma_{1:r})^{-1} U_{1:r}^T Y \qquad 1.6.2$$

Using eq. 1.6.1 and eq. 1.6.2, the estimation of Y is calculated:

$$\hat{Y}_{OLS} = X_{noise}\hat{B}_{OLS} \qquad 1.6.3$$

$$\hat{Y}_{PCR} = X_{noise}\hat{B}_{PCR} \qquad 1.6.4$$

The Euclidean norm square is then calculated for the training data and presented in table 1.6-1. Based on this table, one may conclude that the OLS performed better than PCR. However, table 1.6-2 shows the results for test data and PCR has performed slightly better than OLS. These results can be verified by figure 1.6.4.

Table 1.6-1: training data OLS and PCR performance

	Euclidean norm squared
$\left\|\boldsymbol{Y} - \hat{\boldsymbol{Y}}_{\boldsymbol{OLS}}\right\|^2$	1714
$\left\|\boldsymbol{Y} - \hat{\boldsymbol{Y}}_{\boldsymbol{PCR}}\right\|^2$	2260

Table 1.6-2: test data OLS and PCR performance

	Euclidean norm squared
$\left\|\boldsymbol{Y} - \hat{\boldsymbol{Y}}_{\boldsymbol{OLS}}\right\|^2$	1361
$\left\|\boldsymbol{Y} - \hat{\boldsymbol{Y}}_{\boldsymbol{PCR}}\right\|^2$	1357

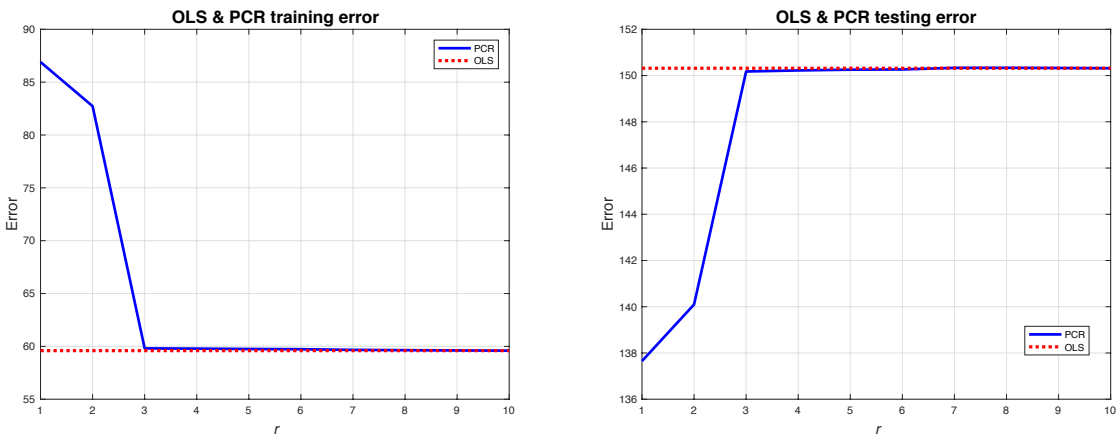


Figure 1.6.4: *OLS vs PCR performance for training and testing data*

d) Table 1.6-2 confirm the fact that PCR performs better in approximating Y than OLS, by averaging 200 realizations of the test data.
The results are also visualized for 200 iteration in figure 1.6.5.

Table 1.6-2: test data OLS and PCR performance

	Euclidean norm squared
$\|\boldsymbol{Y} - \widehat{\boldsymbol{Y}}_{\boldsymbol{OLS}}\|^2$	7115
$\|\boldsymbol{Y} - \widehat{\boldsymbol{Y}}_{\boldsymbol{PCR}}\|^2$	7094

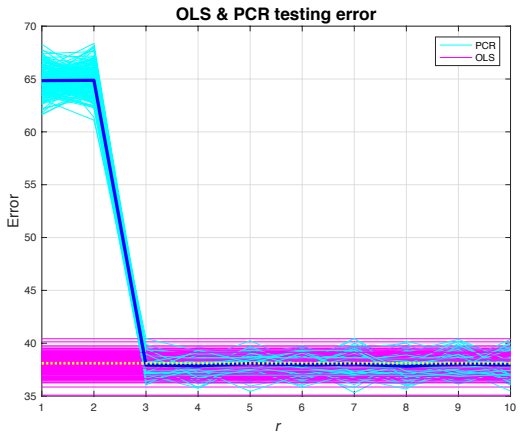


Figure 1.6.5: *OLS vs PCR test data error – 200 realizations*

2 Adaptive Signal Processing

2.1 The least Mean Square (LMS) Algorithm

a) The AR process of order 2 is defined as below:

$$x(n) = a_1 x(n-1) + a_2 x(n-2) + \eta(n) \quad \eta(n) \sim \mathcal{N}(0, \sigma_\eta^2) \tag{2.1.1}$$

Using the LMS, the weights of autoregressive process $x(n)$, a_1 and a_2, as well as the output $y(n) = \hat{x}(n)$ are estimated. The correlation matrix $\boldsymbol{R_{xx}}$ is computed for the input vector $x(n) = [x(n-1), x(n-2)]^T$ as below:

$$\boldsymbol{R_{xx}} = E[x(n)x^T(n)] = \begin{bmatrix} x(n-1) \\ x(n-2) \end{bmatrix} [x(n-1) \;\; x(n-2)] \tag{2.1.2}$$

$$\boldsymbol{R_{xx}} = E \begin{bmatrix} x(n-1)x(n-1) & x(n-1)x(n-2) \\ x(n-1)x(n-2) & x(n-1)x(n-2) \end{bmatrix} = \begin{bmatrix} \gamma(0) & \gamma(1) \\ \gamma(1) & \gamma(0) \end{bmatrix} \tag{2.1.3}$$

$\gamma(k)$ is also termed as $r_{xx}(k)$ and it is the autocorrelation function (ACF). In order to determine the entries of matrix $\boldsymbol{R_{xx}}$ we need to obtain at least two equations of $\gamma(k)$ and can be done by multiplying $x(n)$ by $x(n-1)$ and $x(n-2)$. Or in general cases, with $x(n-k)$ where k should be the number of unknowns.

$$\gamma(k) = E\{[x(n)x(n-k)]\} = \\ E\{[a_1 x(n-1)x(n-k) + a_2 x(n-2)x(n-k) + \eta(n)x(n-k)]\} \tag{2.1.4}$$

Expecting value being a linear operator, one can write:

$$\gamma(k) = a_1 E\{[x(n-1)x(n-k)]\} + a_2 E\{[x(n-2)x(n-k)]\} + E\{[\eta(n)x(n-k)]\}$$

For different values of $k = 1,2 \ldots$ and $a_1 = 0.1, a_2 = 0.8$ and $\sigma_\eta^2 = 0.25$ the following equations are obtained:

$$\gamma(0) = a_1\gamma(1) + a_2\gamma(2) + \sigma_\eta^2$$
$$\gamma(1) = a_1\gamma(0) + a_2\gamma(1)$$
$$\gamma(2) = a_1\gamma(1) + a_2\gamma(0)$$

Solving the above equations, one can get $\gamma(0) = \frac{25}{27}$ and $\gamma(1) = \frac{25}{54}$ from which the correlation matrix can be calculated:

$$\boldsymbol{R_{xx}} = \begin{bmatrix} \gamma(0) & \gamma(1) \\ \gamma(1) & \gamma(0) \end{bmatrix} = \begin{bmatrix} \frac{25}{27} & \frac{25}{54} \\ \frac{25}{54} & \frac{25}{27} \end{bmatrix} = \frac{25}{54} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

Convergence of the LMS should satisfy the condition:

$$0 < \mu < \frac{2}{\lambda_{max}} \qquad 2.1.5$$

Where μ is the step size of LMS and λ_{max} is the maximum eigenvalue of the correlation matrix $\boldsymbol{R_{xx}}$. Eigenvalues for this matrix are $\lambda_1 = 0.463$ and $\lambda_2 = \lambda_{max} = 1.3889$ resulting in:

$$0 < \mu < 1.44$$

Therefore, for the above limits, the convergence of LMS is guaranteed to converge.

b) Figure 2.1.1 shows the prediction error squared for one realization both for $\mu = 0.01$ and $\mu = 0.05$.
Figure 2.1.2 shows the prediction error squared for 100 realizations of both $\mu = 0.01$ and $\mu = 0.05$. It can be seen that $\mu = 0.05$ results in faster convergence to the steady state and is due to the fact the larger step size allows steeper descent of error surface.

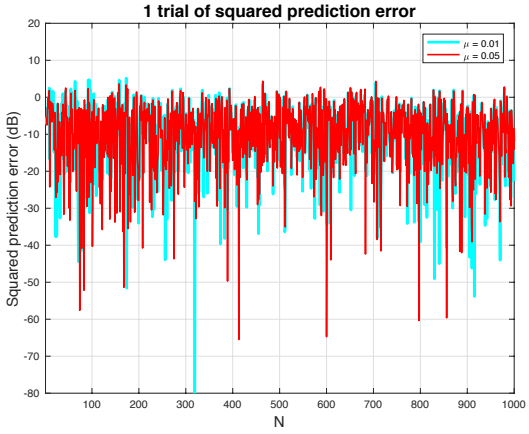


***Figure 2.1.1**: Squared prediction error - 1 trial*

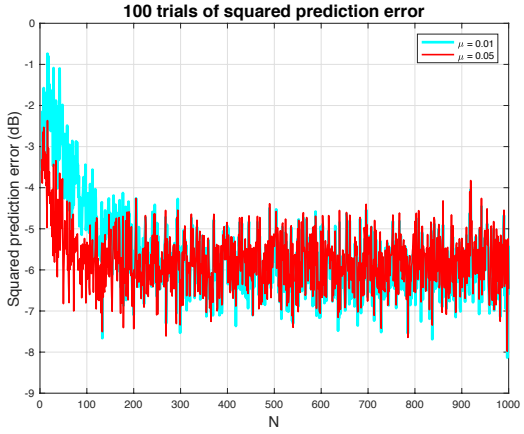


***Figure 2.1.2:** Squared prediction error & convergence speed seen over 100 trials.*

c) The theoretical Misadjustment , $\mathcal{M}_{LMS} = \frac{1}{2}\mu\, Tr(\boldsymbol{R_{xx}})$ where $Tr(\boldsymbol{R_{xx}}) = \frac{25}{27} + \frac{25}{27} = 1.8519$ for the $\boldsymbol{R_{xx}}$ obtained earlier in a. For different values of μ, $\mathcal{M}_{LMS}$ is shown in table 2.1-1. The empirical Misadjustment is calculated $\mathcal{M}_{emp} = \frac{MSE - \sigma_\eta^2}{\sigma_\eta^2}$ and the values are shown in table 2.1-1 for 100 realizations of the steady-state.

***Table 2.1-1:** Misadjustment comparison of $\mathcal{M}_{LMS}$ and $\mathcal{M}_{emp}$*

$\boldsymbol{\mu}$	$\boldsymbol{\mathcal{M}_{LMS}}$	$\boldsymbol{\mathcal{M}_{emp}}$
0.01	0.0093	0.01
0.05	0.0463	0.0533

As it can be seen in figures 2.1.3 and 2.1.4, smaller step sizes result in slower speed of convergence.

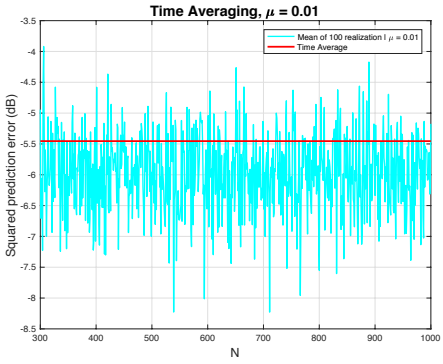


Figure 2.1.3: *Time average of steady state error | $\mu = 0.01$*

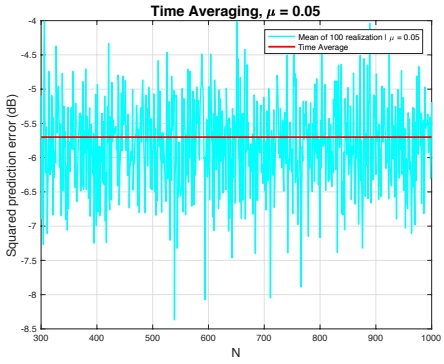


Figure 2.1.4: *Time average of steady state error | $\mu = 0.05$*

d) Figure 2.1.5 shows the evolution of the AR process parameters a_1 and a_2 for $\mu = 0.01$ whereas figure 2.1.6 shows these parameters for $\mu = 0.05$. For this case, 100 realizations are considered.
It can be seen that the convergence with larger step size, μ, is faster to the true values. However, it can also be noticed that $\hat{a}_1$ converges to its true value in both cases with negligible error. Whereas, $\hat{a}_2$ is converging to different values, with negative offset in both cases, 3.7% lower than the actual value for $\mu = 0.01$ and 10% lower than the actual value for $\mu = 0.05$.

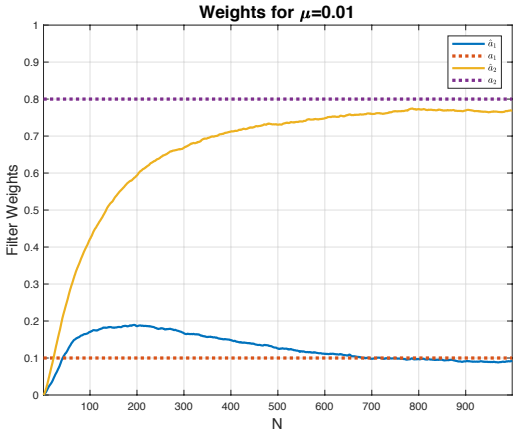


Figure 2.1.5: *Evolution of adaptive filters coefficients for LMS - $\mu = 0.01$*

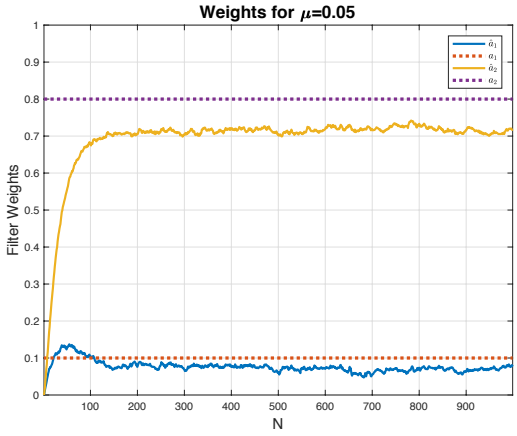


Figure 2.1.6: *Evolution of adaptive filters coefficients for LMS - $\mu = 0.05$*

e) Minimizing the objective function with respect to $\boldsymbol{w}(n)$ is to differentiate it with respect to $\boldsymbol{w}$ and equate it to zero:

$$J_2(n) = \frac{1}{2}(e^2(n) + \gamma\|\boldsymbol{w}(n)\|_2^2) \qquad 2.1.6$$

$$\frac{\partial J_2}{\partial \boldsymbol{w}} = e(n)\frac{\partial e}{\partial \boldsymbol{w}} + \gamma \boldsymbol{w}(n) = 0 \qquad 2.1.7$$

Where $e(n) = x(n) - w^T(n)x(n)$. Substituting this equation in the eq. 2.1.7, one can get:

$$\frac{\partial e}{\partial \boldsymbol{w}} = \frac{\partial\big(x(n) - w^T(n)x(n)\big)}{\partial \boldsymbol{w}} = -x(n) \qquad 2.1.8$$

$$\frac{\partial J_2}{\partial \boldsymbol{w}} = -e(n)x(n) + \gamma \boldsymbol{w}(n) = 0 \qquad 2.1.9$$

Using the gradient descent method update:

$$w(n) = w(n) - \mu\nabla_w J(n) \qquad 2.1.10$$

Substituting eq. 2.1.9 into eq. 2.1.10, will result in:

$$w(n+1) = w(n) - \mu\big(-e(n)x(n) + \gamma w(n)\big)$$
$$\boldsymbol{w}(n+1) = (1-\mu\gamma)\boldsymbol{w}(n) - \mu e(n)\boldsymbol{x}(n) \qquad 2.1.11$$

Eq. 2.1.11 is the same as equation (21) shown in the coursework used for Leaky LMS, hence proved.

f) From the plots of Leaky LMS, one can confirm that when $\gamma = 0$, the coefficients evolution is the same as in figures 2.1.5 and 2.1.6, presented in section d.
For the rest of the cases where $\gamma = 0.25, 0.5, 0.75, 1$, the results are presented in figures 2.1.7 - 2.1.14.

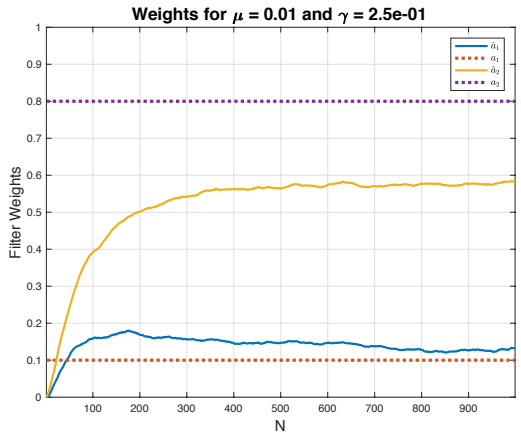


Figure 2.1.7: *filter coefficient evolution | $\mu = 0.01$ & $\gamma = 0.25$*

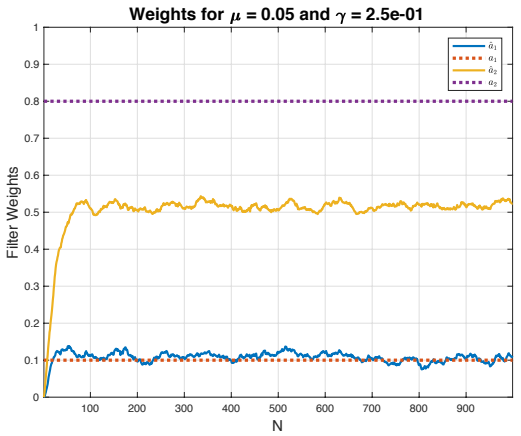


Figure 2.1.8: *filter coefficient evolution | $\mu = 0.05$ & $\gamma = 0.25$*

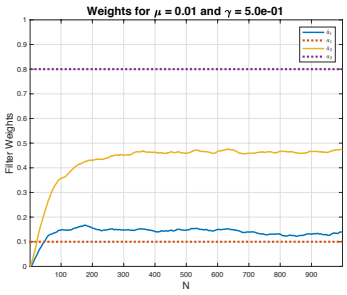


Figure 2.1.9: *filter coefficient evolution | $\mu = 0.01$ & $\gamma = 0.5$*

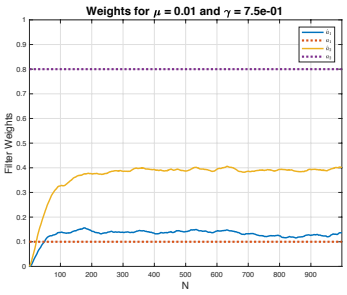


Figure 2.1.10: *filter coefficient evolution | $\mu = 0.01$ & $\gamma = 0.75$*

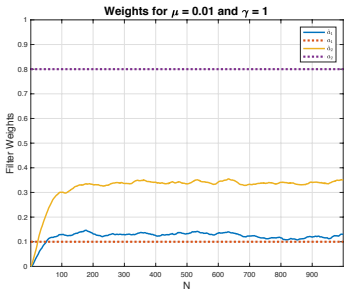


Figure 2.1.11: *filter coefficient evolution | $\mu = 0.01$ & $\gamma = 1$*

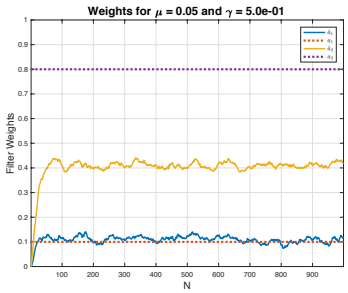


Figure 2.1.12: *filter coefficient evolution | $\mu = 0.05$ & $\gamma = 0.5$*

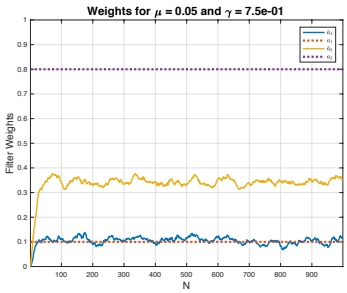


Figure 2.1.13: *filter coefficient evolution | $\mu = 0.05$ & $\gamma = 0.75$*

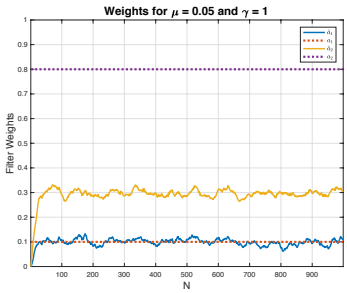


Figure 2.1.14: *filter coefficient evolution | $\mu = 0.05$ & $\gamma = 1$*

As it can be seen, $\hat{a}_2$ is not converging to its true value and with different values of γ, $\hat{a}_2$ converges to different values. The larger the γ the larger the bias of $\hat{a}_2$.
The Weiner-Hopf solution is used to get the optimal $\boldsymbol{w}$:

$$\boldsymbol{w}_{opt} = \boldsymbol{R}^{-1}\boldsymbol{p} \qquad 2.1.12$$

Where, $\boldsymbol{R}$ is the autocorrelation matrix of the input data and $\boldsymbol{p}$ is the cross-correlation vector. The LMS algorithm arrives to this optimal value for $\boldsymbol{w}$, under the boundaries set for μ. However, the Leaky LMS does not minimizes the same function due to presence of γ. The Leaky LMS as a result of this change converges to the following solution:

$$\lim_{k\to\infty} E[\boldsymbol{w}_k] = (\boldsymbol{R} + \gamma \boldsymbol{I})^{-1}\boldsymbol{p} \qquad 2.1.13$$

As a result, the optimal solutions are different for different values of γ.

2.2 Adaptive Step Size

a) In previous section, it was shown that LMS assumes the step size μ to be constant. The effects were studied such as the speed of convergence in case of increasing the step size and the circumstances on steady state error and converging to different values were observed.

In the adaptive step size (GASS) method, the step size μ is considered to be time varying, hence $\mu(n)$. Several algorithms for updating $\mu(n)$ are derived and the step size is updated according to the following equation:

$$\mu(n+1) = \mu(n) + \rho e(n) x^T(n) \psi(n)$$

Where $\psi(n)$ varies with time while ρ is constant. Several forms are presented in the coursework to find $\psi(n)$:

Benveniste:

$$\psi(n) = [\mathbf{I} - \mu(n-1)x(n-1)x^T(n-1)]\psi(n-1) + e(n-1)x(n-1)$$

Ang & ***Farhang***:

$$\psi(n) = \alpha\psi(n-1) + e(n-1)x(n-1), \quad 0 < \alpha < 1$$

Matthews & ***Xie***:

$$\psi(n) = e(n-1)x(n-1)$$

These updates for step size are implemented using GASS algorithm and the results are shown in figures 2.2.1 and 2.2.2 for 1 trial and figures 2.2.3 and 2.2.4 for 100 trials.

It can be seen that using Benveniste step size update results in smaller errors and converges faster to the true values in contrast to Ang-Farhang and Matthews-Xie, but all converges to the true values of the weights after 100 iterations. Whereas the LMS with step sizes $\mu = 0.1$ and $\mu = 0.01$ oscillates around the true values but never converges to the true values and has a bias as seen in figure 2.2.3.

Overall, the complexity and number of calculations needed may be the case in one application while the accuracy needed might be the case in another application and based on that a decision can be made to use which one of these algorithms.

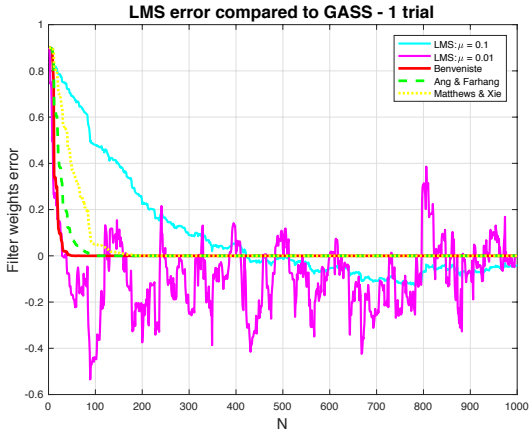


Figure 2.2.1: *Comparing the convergence time for LMS vs GASS – 1 trial*

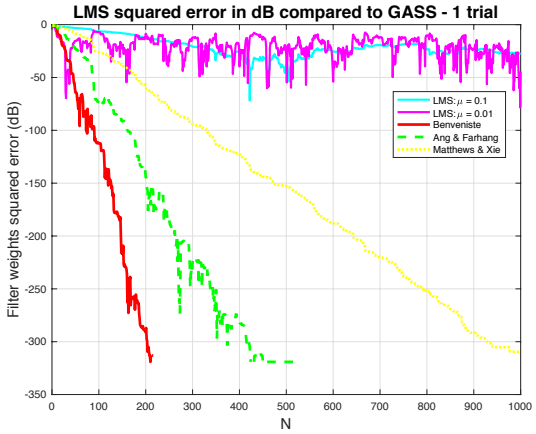


Figure 2.2.2: *Comparing the convergence time for LMS vs GASS, error squared in dB – 1 trial*

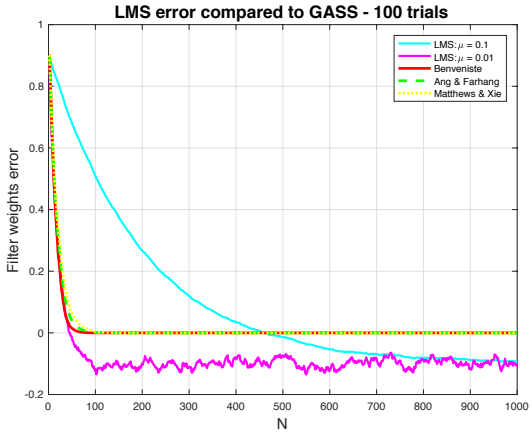


Figure 2.2.3: *Comparing the convergence time for LMS vs GASS – averaging 100 trials*

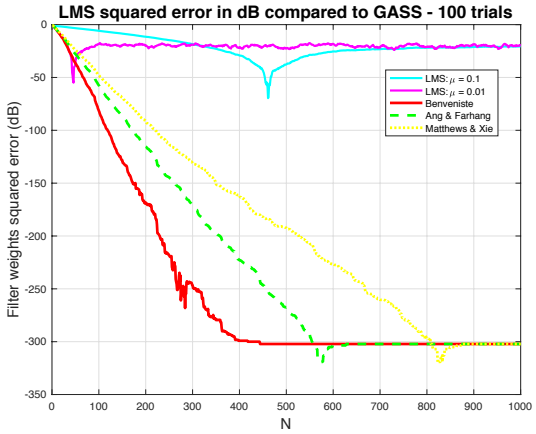


Figure 2.2.4: *Comparing the convergence time for LMS vs GASS, error squared in dB – averaging 100 trials*

b) Weights of the normalized least mean square (NLMS) are updated according to the following equation:

$$w(n+1) = w(n) + \frac{\beta}{\epsilon + \|x(n)\|^2} e(n)x(n) = w(n) + \frac{\beta}{\epsilon + x^T(n)x(n0} e(n)x(n) \qquad 2.2.1$$

To prove the NLMS equation's (eq. 2.2.1) similarity with the equation 2.2.2 based on *a posteriori error*:

$$w(n+1) = w(n) + \mu e_p(n)x(n) \qquad 2.2.2$$

Multiplying both sides by $x^T(n)$:

$$x^T(n)w(n+1) = x^T(n)w(n) + x^T(n)\mu e_p(n)x(n) \qquad 2.2.3$$

Adding $d(n)$ to both sides:

$$\begin{aligned} d(n) + x^T(n)w(n+1) &= d(n) + x^T(n)w(n) + x^T(n)\mu e_p(n)x(n) \\ d(n) - x^T(n)w(n) &= d(n) - x^T(n)w(n+1) + x^T(n)\mu e_p(n)x(n) \end{aligned} \qquad 2.2.4$$

With a posteriori error $e_p(n) = d(n) - x^T(n)w(n+1)$ and $e(n) = d(n) - x^T(n)w(n)$, eq. 2.2.4 can be written as:

$$e(n) = e_p(n) + x^T(n)\mu e_p(n)x(n)$$

Rearranging terms:

$$e_p(n) = \frac{e(n)}{1 + \mu\|x\|^2} \qquad 2.2.5$$

Substituting eq. 2.2.5 into eq. 2.2.2:

$$w(n+1) = w(n) + \mu \frac{e(n)}{1 + \mu\|x\|^2} x(n)$$

$$w(n+1) = w(n) + \frac{1}{\frac{1}{\mu} + \|x\|^2} e(n)x(n) \qquad 2.2.6$$

Comparing eq. 2.2.6 and eq. 2.2.1 used for NLMS, it can be observed that $\beta = 1$, $\epsilon = 1/\mu$ and the term $\epsilon = 1/\mu$ guarantees stability of *a posteriori* update algorithm even when $\|x\|^2$ becomes very small or zero.

c) Figure 2.2.5 shows the weights estimates by Benveniste's algorithm (cyan) and by the GNGD (red). It can be readily observed that GNGD has a faster speed of convergence while maintaining the same accuracy. The main difference is between the linearity and non-linearity of the two algorithms, giving GNGD an upper hand.

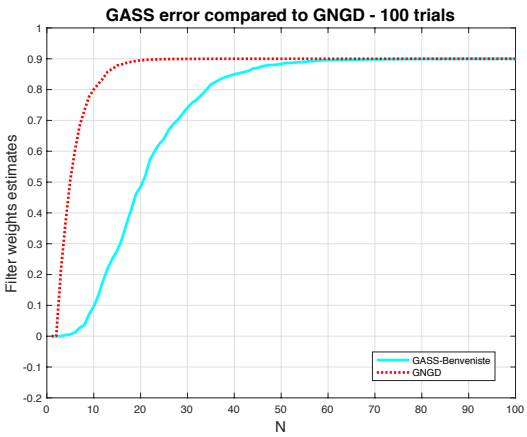


Figure 2.2.5: *Weights estimates by Benveniste's and GNGD algorithms*

To understand the computational complexity of the two algorithms, its necessary to express it in terms of the order of the process, the order is termed as M.
Each update of GNGD algorithm involves calculating inner products, additions/subtractions of vectors in M and other scalar operations hence the complexity is M, $\mathcal{O}(\mathrm{M})$. However, the Benveniste algorithm involved the out product of $x(n)$, $x(n-1)x^T(n-1)$, and $[\mathbf{I}-\mu(n-1)x(n-1)x^T(n-1)]\psi(n-1)+e(n-1)x(n-1)$ which both have complexity M^2, making the computational complexity $\mathcal{O}(\mathrm{M}^2)$ for Benveniste algorithm.

As it can be seen in figure 2.2.5 and also discussed above, the speed of convergence of GNGD is higher than that of Benveniste's while maintaining less computational complexity and same accuracy levels.

2.3 Adaptive Noise Cancellation

a) It is important to find the required value of the delay Δ for the delay operator. With the help of this delay, the adaptive linear enhancer (ALE) will then be able to separate the correlated signals noise, $\eta(n)$, and predictor input, $u(n)$, and get a denoised signal $x(n)$.
For finding the optimal value(s) of Δ, the mean squared error should be seen for ALE:

$$MSE = E\left\{\left(s(n)-\hat{x}(n)\right)^2\right\} = E\left\{\left(x(n)+\eta(n)-\hat{x}(n)\right)^2\right\}$$

$$= E\{\eta^2(n)\} + E\left\{\left(x(n)-\hat{x}(n)\right)^2\right\} + 2E\{\eta(n)\left(x(n)-\hat{x}(n)\right)\} \qquad 2.3.1$$

Out of the terms above, the term that can be manipulated and played with is $2E\{\eta(n)\left(x(n)-\hat{x}(n)\right)\}$ since the other two terms do not depend on Δ. To be more precise, this will narrow down to $E\{\eta(n)\hat{x}(n)\}$ since the other term is zero because of no correlation. Substituting $\eta(n) = v(n) + 0.5v(n-2)$ to $E\{\eta(n)\hat{x}(n)\}$ and expanding:

$$E\{\eta(n)\hat{x}(n)\} = E\{\left(v(n)+0.5v(n-2)\right)w^T u(n)\}$$

$$= E\left\{\left(v(n)+0.5v(n-2)\right)\left(\sum_{i=0}^{M} w_i s(n-\Delta-i)\right)\right\}$$

$$= E\left\{\left(v(n)+0.5v(n-2)\right)\left(\sum_{i=0}^{M} w_i\ (x(n-\Delta-i)+\eta(n-\Delta-i))\right)\right\}$$

Expected value of all the terms containing multiplication of $x(n)$ and $v(n)$ will be zero because these two are uncorrelated, leaving with:

$$= E\left\{\left(v(n)+0.5v(n-2)\right)\left(\sum_{i=0}^{M} w_i\,\eta\left(x(n-\Delta-i)\right)\right)\right\}$$

$$= E\left\{\left(v(n)+0.5v(n-2)\right)\left(\sum_{i=0}^{M} w_i\,\eta(n-\Delta-i)\right)\right\}$$

$$= E\left\{\left(v(n) + 0.5v(n-2)\right)\left(\sum_{i=0}^{M} w_i\left(v(n-\Delta-i) + 0.5v(n-\Delta-i-2)\right)\right)\right\} \qquad 2.3.2$$

Eq. 2.3.2 is equal to zero for values of $\Delta > 2$. Therefore, the optimal values of Δ should be greater than 2.

To justify this result by visualizing, refer to figures 2.3.1 to 2.3.8. With increasing the delay Δ, the noise in the signal is further suppressed. Additionally, when $\Delta > 2$, the mean square prediction error (MSPE) reduces from 0.44 to 0.32 which is a major drop down.

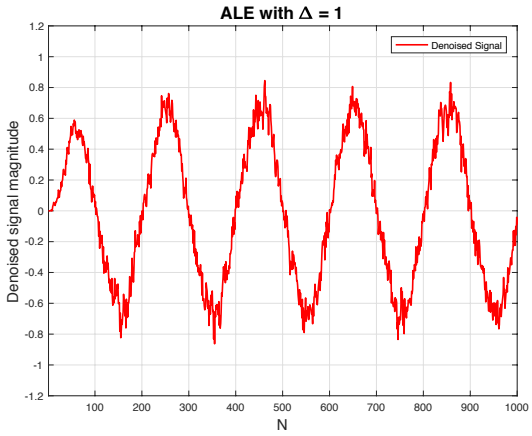


***Figure 2.3.1**: ALE performance with Δ | Δ= 1*

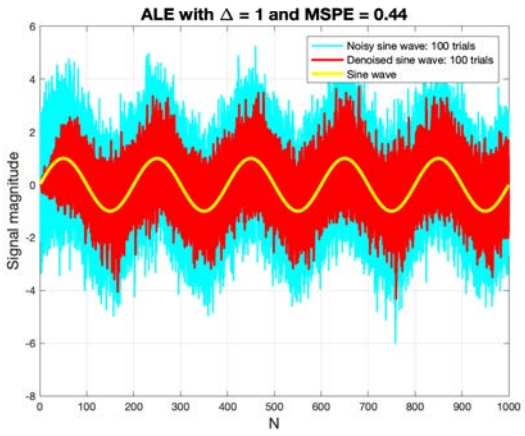


***Figure 2.3.2:** ALE performance with Δ | Δ= 1*

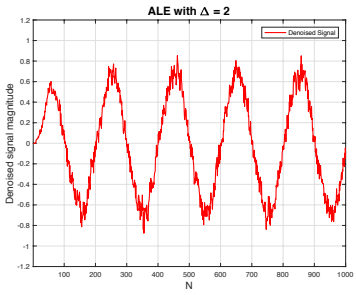


***Figure 2.3.3:** ALE performance with Δ | Δ= 2*

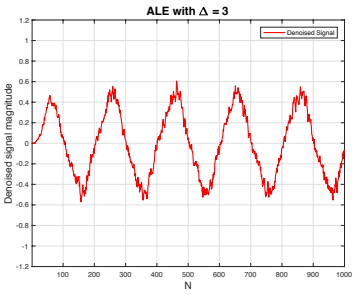


***Figure 2.3.4:** ALE performance with Δ | Δ= 3*

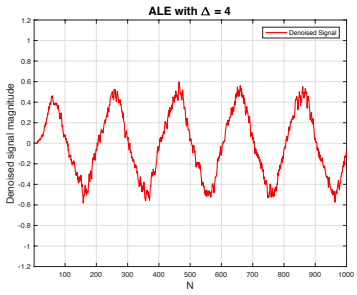


***Figure 2.3.5:** ALE performance with Δ | Δ= 4*

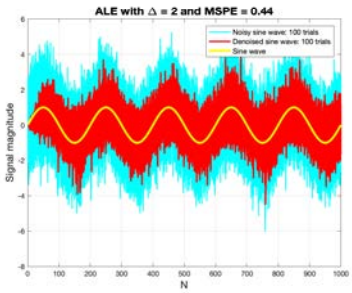


***Figure 2.3.6:** ALE performance with Δ | Δ= 2*

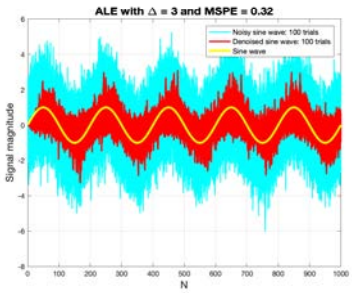


***Figure 2.3.7:** ALE performance with Δ | Δ= 3*

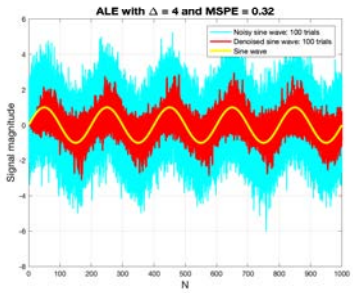


***Figure 2.3.8:** ALE performance with Δ | Δ= 4*

b) Figure 2.3.9 shows the effects of Δ on MSPE. It further supports the theoretical results that increasing Δ from 2 to 3 results in major drop down from 0.44 to 0.32 for filters of order $M = 5$. It further reduces by a small number when the delay is around 5 and filter $M = 5$ is used.

Figure 2.3.10 shows the relation between filter order, M, and MSPE. With $\mu = 0.01$, the minimum MSPE is acquired when $M = 5$. Varying μ may result in other solutions but for this value of μ a pragmatic solution would be using filter order $M = 5$ with delay $\Delta = 5$.

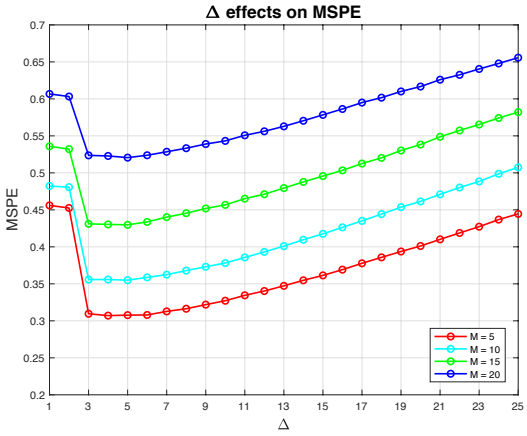


Figure 2.3.9: *Δ vs MSPE*

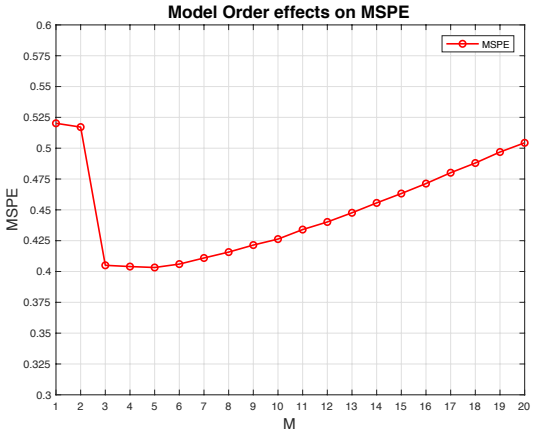


Figure 2.3.10: *filter order, M, vs MSPE*

c) Figure 2.3.11 shows the denoised signal using ALE (red), the denoised signal using ANC (blue) and the original sinusoid (yellow) for one iteration. It can be seen the ANC and the original sinusoid fit more precisely.
Figure 2.3.12 and figure 2.3.13 shows the denoised signal over 100 realizations by ALE and ANC, respectively. It can be seen that using ANC, one can get better approximation of the signal and MSPE becomes very small, in this case, 0.084 in contrast to 0.307 that of ALE.

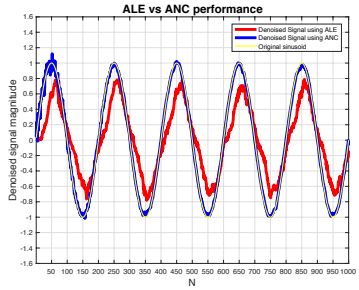


Figure 2.3.11: *denoised ALE, ANC, and original signal*

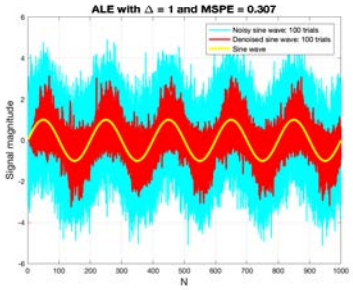


Figure 2.3.12: *Noisy signal denoised ALE, and original signal – 100 trials*

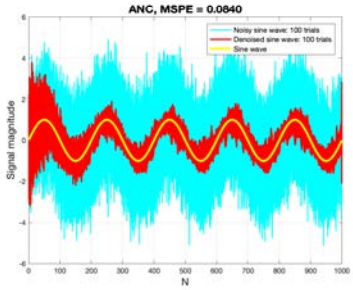


Figure 2.3.13: *Noisy signal denoised ANC, and original signal – 100 trials*

d) Figure 2.3.14 is the periodogram for the original signal with a strong frequency induced by the power line at $f = 50Hz$.

Figure 2.2.15 shows different configuration used for the ANC algorithm. With $\mu \in \{0.025, 0.01, 0.005, 0.001\}$ and $M \in \{5,15,25\}$, different results can be observed. Its evident that decreasing μ results in decreasing the number of frequencies over which the

algorithm is more useful. However, the cost for this is more time to converge and cancel the induced noise.

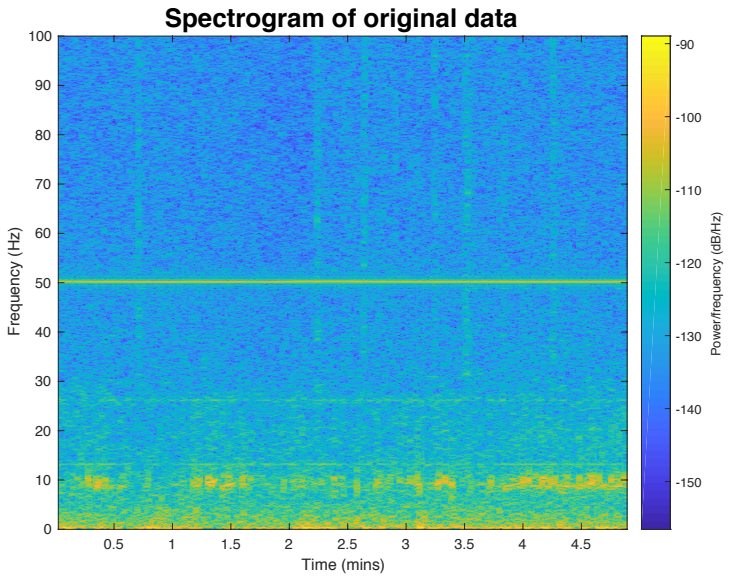
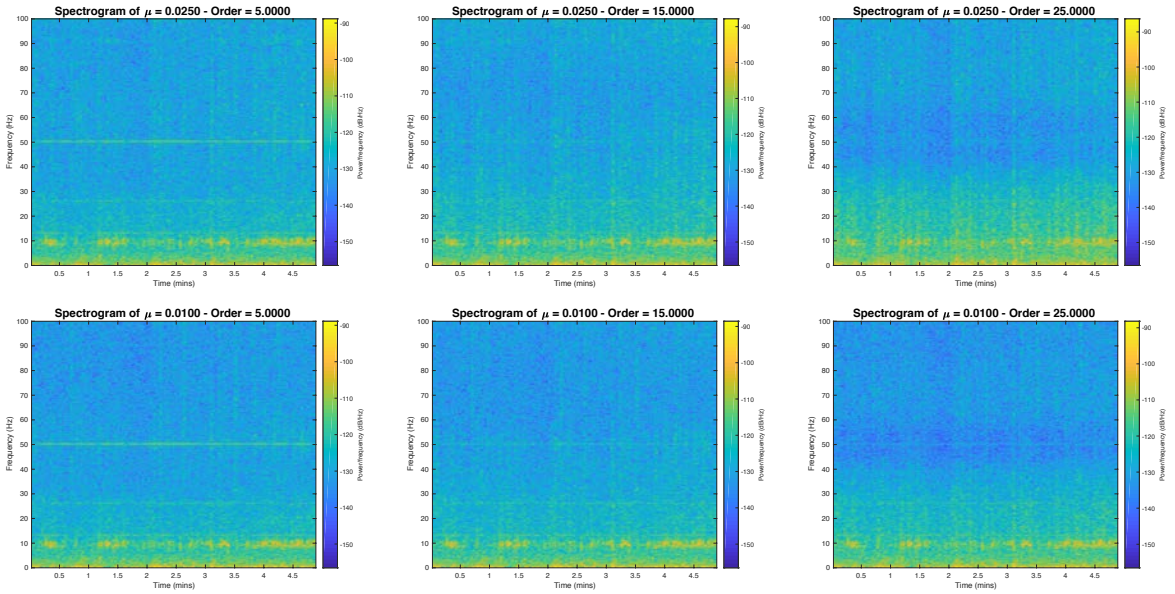


Figure 2.3.14: *Spectrogram of the original POz of EEG – 50Hz component*

On the other hand, increasing *M* also further cancels the frequency at 50Hz. However, with larger step sizes, this order filter also suppresses adjacent frequencies at 50Hz and may increase the error further.



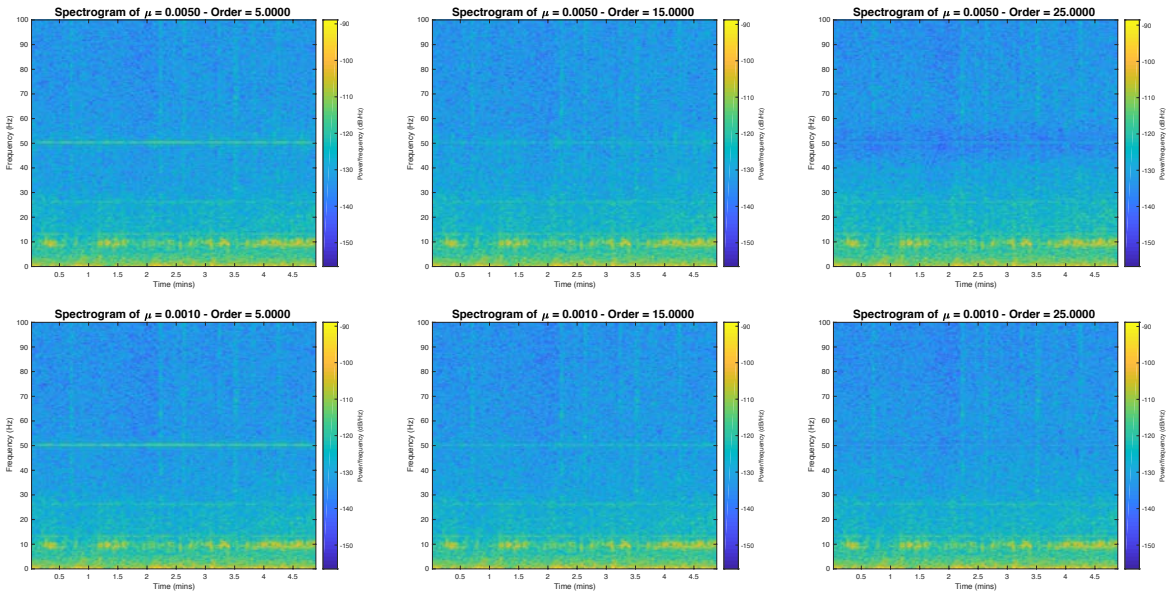


Figure 2.3.15: *Noise cancellation using ANC with different orders and step sizes*

Figure 2.3.16 shows the periodogram for the original data and the denoised signal with filter order 15 and 25 and $\mu = 0.001$. It can be seen that performance of filter order 15 is better than order 25 at lower frequency and the strong induced component at 50Hz.

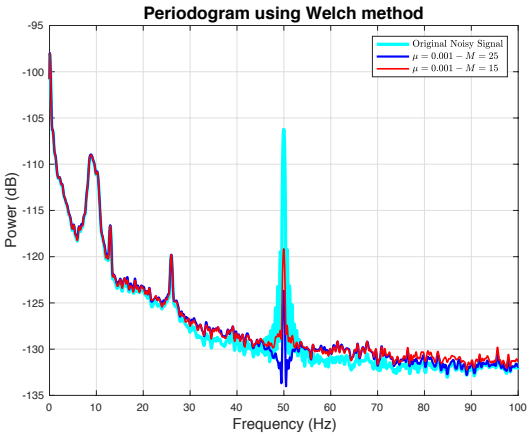


Figure 2.3.16: *Periodogram of original and denoised signal*

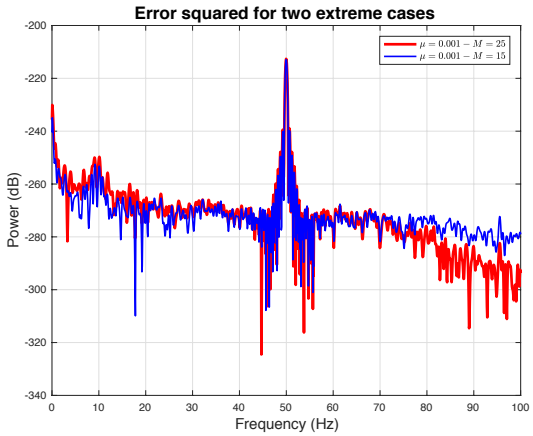


Figure 2.3.17: *Error squared (dB) for two different order filters*

3 Widely Linear Filtering and Adaptive Spectrum Estimation

3.1 Complex LMS and Widely Linear Modelling

a) Figure 3.1.1 shows the widely linear moving-average process of order 1, $WLMA(1) = y(n)$, (red) driven by the white gaussian noise (cyan).

$$y(n) = x(n) + (1.5 + 1j)x(n-1) + (2.5 - 0.5j)x^*(n-1) \quad x \sim \mathcal{N}(0,1)$$

From the data it can be seen that it is not circular. Figure 3.1.2 shows the error for each algorithm, the complex LMS (CLMS) in (red) and the augmented complex LMS (ACLMS) in (blue). As expected, the steady state error for CLMS is $8.77dB$ while it is $-307.2dB$ which is due to the non-circularity of the data.

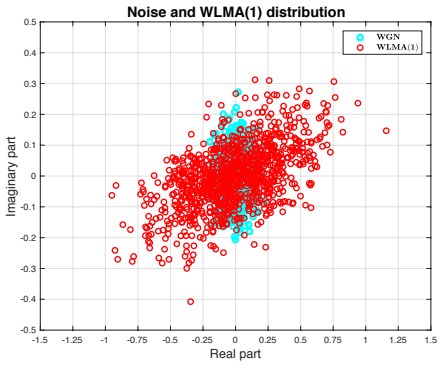


Figure 3.1.1: *WGN and non-circular WLMA(1) distribution*

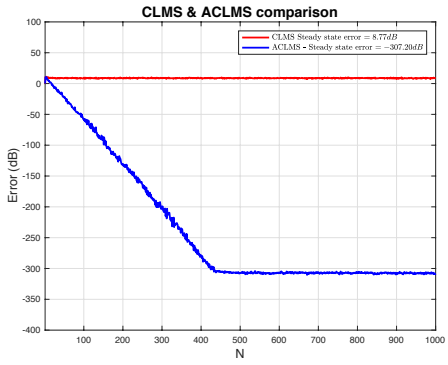


Figure 3.1.2: *Squared error comparison in dB for CLMS and ACLMS of WLMA(1)*

b) Figure 3.1.3 shows wind speed scatter plots for different wind speeds, low (yellow), medium (magenta), and high (cyan). Circularity for each of these data is calculated and as it can be observed from the data and circularity of each data, low wind is highly circular invariant, $\rho = 0.159$, and the result can be confirmed by the mean of data in real and imaginary parts. The axes plotted in black are for reference and the axes plotted in red are the mean of data for real and imaginary parts. The less the offset the more the data is rotationally invariant and the more the offset the less the data is rotationally invariant. Circularity coefficients for medium speed and high-speed wind are $\rho = 0.454$ and $\rho = 0.624$, making the high-speed wind data less circular. It should be noted that the shapes of data is not important and does not affect circularity.

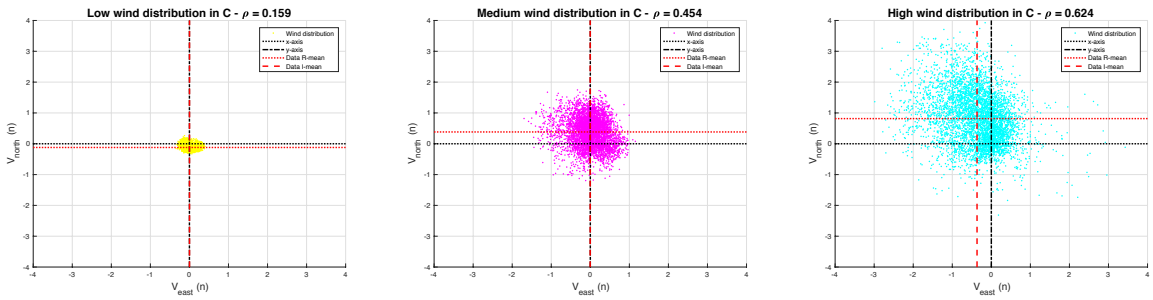


Figure 3.1.3: *Wind speed scatter plots and circularity of data*

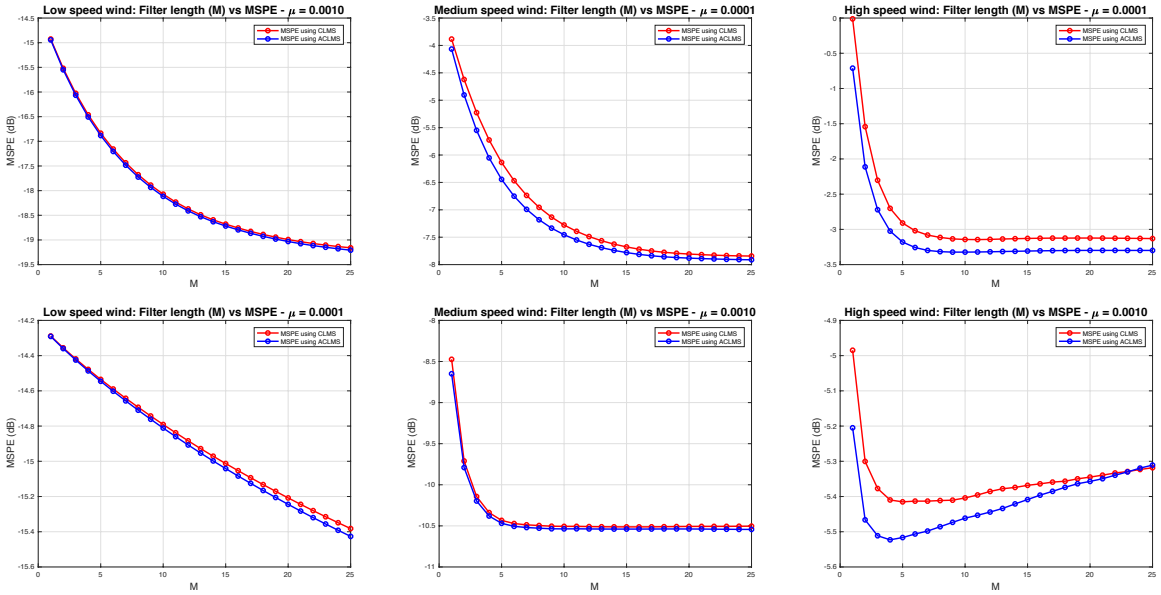


Figure 3.1.4: *Filter orders with different step sizes vs MSPE*

Filter orders of 1 to 25 are considered both for CLMS and ACLMS one-step ahead prediction. As it can be observed, ACLMS performs better in all cases for low filter lengths, but this performance is more evident with high speed data as expected because of the higher circularity coefficient for high speed wind. And the MSPE decreases with filter order for all cases except the high wind speed wind where MSPE has a minimum but as the filter length is increased beyond 25, the MSPE for ACLMS can perform worse than CLMS. This error is due to overfitting of data and ACLMS having more parameters than CLMS to tune.

c) Figure 3.1.5 shows the balanced network circularity diagram using the Clark transform to transform the balanced 3-phase voltages $V_a = V_b = V_c = constant$ with a phase shift of 120° between each phase and phase distortion of $\Delta_a = \Delta_b = 0$, to calculate the complex $\alpha - \beta$ voltages. These voltages are then plotted for the balanced network in figure 3.1.5 and for the unbalanced network with phase and amplitude distortions in figure 3.1.6.

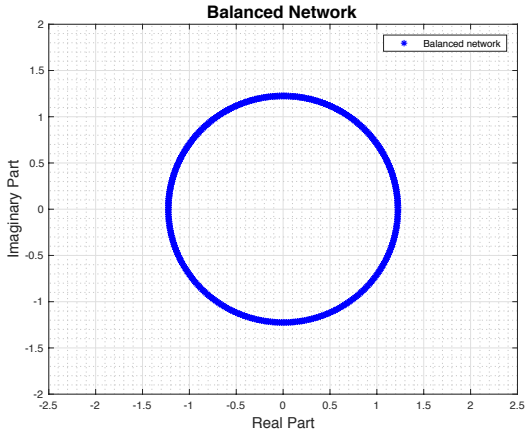


Figure 3.1.5: *Balanced network circularity diagram*

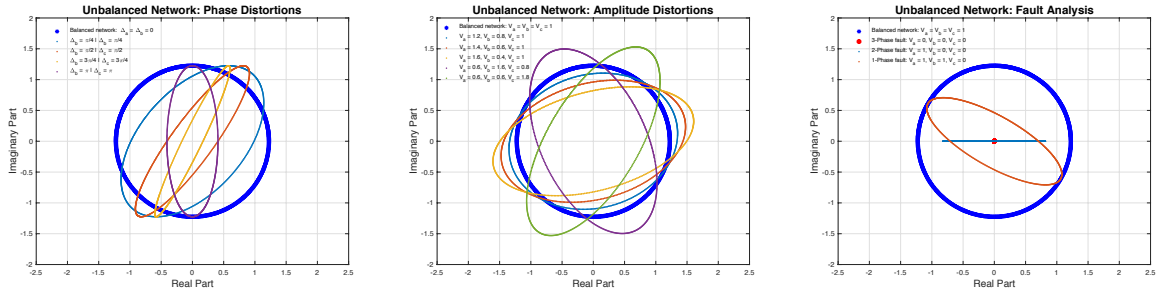


Figure 3.1.6: *Unbalanced network (phase and amplitude distortions) circularity diagram & fault analysis*

The unbalanced network has elliptical diagrams and vary with changing with phase distortions, Δ_a and Δ_b, and magnitude changes, V_a, V_b, V_c.
The fault conditions, 3-phase fault, 2-phase fault and 1-phase fault are considered in 3rd part of figure 3.1.6 and the relevant diagrams are shown for these cases.

d) The two autoregressive models are given in eq. 3.1.1 and eq. 3.1.2

$$\textbf{Strictly Linear}: \; v(n+1) = h^*(n)v(n) \qquad 3.1.1$$

$$\textbf{\textit{Widely Linear}}: \; v(n+1) = h^*(n)v(n) + g^*(n)v(n) \qquad 3.1.2$$

The voltage of a balanced system is given by:

$$v(n) = \sqrt{\frac{3}{2}}\, V\, e^{j\left(2\pi\frac{f_o}{f_s}n+\phi\right)} \qquad 3.1.3$$

Substituting 3.1.3 in 3.1.1 for $v(n)$ and $v(n+1)$ gives:

$$h^*(n) = \frac{\sqrt{\frac{3}{2}}\, V\, e^{j\left(2\pi\frac{f_o}{f_s}(n+1)+\phi\right)}}{\sqrt{\frac{3}{2}}\, V\, e^{j\left(2\pi\frac{f_o}{f_s}n+\phi\right)}} = e^{j2\pi\frac{f_0}{f_s}}$$

Since $h^*(n) = \Re\{h(n)\} - \Im\{h(n)\}$:

$$h^*(n) = \Re\{h(n)\} - \Im\{h(n)\} = |h(n)|\, e^{-j\theta} = |h(n)|\, e^{-j\left(arctan\left(\frac{\Re\{h(n)\}}{\Im\{h(n)\}}\right)\right)}$$

$$e^{j2\pi\frac{f_0}{f_s}} = |h(n)|\, e^{-j\left(arctan\left(\frac{\Re\{h(n)\}}{\Im\{h(n)\}}\right)\right)} \qquad 3.1.4$$

Equating the phases on both sides:

$$2\pi\frac{f_0}{f_s} = -arctan\left(\frac{\Re\{h(n)\}}{\Im\{h(n)\}}\right)$$

Rearranging terms and taking the negative sign inside arctan:

$$f_0(n) = \frac{f_s}{2\pi} arctan\left(\frac{\Im\{h(n)\}}{\Re\{h(n)\}}\right)$$

Hence, proved the equation 45 in the coursework used for the frequency of the balanced network.

For the unbalanced system, the widely linear model shown in eq. 3.1.2 is used. In order to find the frequency component, one should derive the exponential form of the voltages, which are approximated by Clarke transform:

$$v(n) = A(n)e^{j\left(2\pi\frac{f_0}{f_s}n+\phi\right)} + B(n)e^{-j\left(2\pi\frac{f_0}{f_s}n+\phi\right)} \quad 3.1.5$$

Substituting eq. 3.1.6 and its conjugate to the widely linear equation $v(n+1) = h^*(n)v(n) + g^*(n)v(n)$:

$$v(n+1) = h^*(n)\left(A(n)e^{j\left(2\pi\frac{f_0}{f_s}n+\phi\right)} + B(n)e^{-j\left(2\pi\frac{f_0}{f_s}n+\phi\right)}\right)$$
$$+g^*(n)\left(A^*(n)e^{-j\left(2\pi\frac{f_0}{f_s}n+\phi\right)} + B^*(n)e^{j\left(2\pi\frac{f_0}{f_s}n+\phi\right)}\right) \quad 3.1.6$$

Where $v(n+1)$:

$$v(n+1) = A(n+1)e^{j\left(2\pi\frac{f_0}{f_s}(n+1)+\phi\right)} + B(n+1)e^{-j\left(2\pi\frac{f_0}{f_s}(n+1)+\phi\right)} \quad 3.1.7$$

Equating eq. 3.1.6 and eq. 3.1.7:

$$A(n+1)e^{j\left(2\pi\frac{f_0}{f_s}(n+1)+\phi\right)} = \left(A(n)h^*(n) + B^*(n)g^*(n)\right)e^{j\left(2\pi\frac{f_0}{f_s}n+\phi\right)}$$

$$\Rightarrow e^{j2\pi\frac{f_0}{f_s}} = \frac{A(n)h^*(n) + B^*(n)g^*(n)}{A(n+1)} \quad 3.1.8$$

$$B(n+1)e^{-j\left(2\pi\frac{f_0}{f_s}(n+1)+\phi\right)} = \left(B(n)h^*(n) + A^*(n)g^*(n)\right)e^{-j\left(2\pi\frac{f_0}{f_s}n+\phi\right)}$$

$$\Rightarrow e^{-j2\pi\frac{f_0}{f_s}} = \frac{B(n)h^*(n) + A^*(n)g^*(n)}{B(n+1)} \quad 3.1.9$$

Assuming $A(n) \approx A(n+1)$ and $B(n) \approx B(n+1)$, and since eq. 3.1.8 is the conjugate of eq. 3.1.9, above terms can be written as:

$$\Rightarrow e^{j2\pi\frac{f_0}{f_s}} = \frac{A(n)h^*(n) + B^*(n)g^*(n)}{A(n+1)} \approx h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} \quad 3.1.10$$

$$\Rightarrow e^{-j2\pi\frac{f_0}{f_s}} = \frac{B(n)h^*(n) + A^*(n)g^*(n)}{B(n+1)} \approx h^*(n) + g^*(n)\frac{A^*(n)}{B(n)} \quad 3.1.11$$

$$h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} = h(n) + g(n)\frac{A(n)}{B^*(n)} \quad 3.1.12$$

Letting $\alpha(k) = \frac{B^*(n)}{A(n)} = \left(\frac{B(n)}{A(n)}\right)^*$, the eq. 3.1.10 can be written and solved as below:

$$h^*(n) + g^*(n)\alpha(k) = h(n) + g(n)\frac{1}{\alpha(k)}$$

$$g^*(n)\alpha^2(k) + \big(h^*(n) - h(n)\big)\alpha(k) - g(n) = 0 \qquad 3.1.13$$

Solving for $\alpha(k)$:

$$\alpha(k) = \frac{\left(-\big(h^*(n) - h(n)\big) \pm \sqrt{\big(h^*(n) - h(n)\big)^2 + 4g^*(n)g(n)}\right)}{2g^*(n)} \qquad 3.1.14$$

$$= \frac{\left(2j\Im\{h(n)\} \pm \sqrt{-4\Im\{h(n)\}^2 + 4|g(n)|^2}\right)}{2g^*(n)}$$

$$= \frac{\left(j\Im\{h(n)\} \pm j\sqrt{\Im\{h(n)\}^2 - |g(n)|^2}\right)}{g^*(n)} \qquad 3.1.15$$

Substituting the solution in eq. 3.1.15 to eq. 3.1.10 or 3.1.11 instead of 3.1.13 is simpler. Hence:

$$e^{j2\pi\frac{f_0}{f_s}} = h^*(n) + g^*(n)\frac{\left(j\Im\{h(n)\} \pm j\sqrt{\Im^2\{h(n)\} - |g(n)|^2}\right)}{g^*(n)}$$

$$e^{j2\pi\frac{f_0}{f_s}} = \Re\{h(n)\} \pm j\sqrt{\Im^2\{h(n)\} - |g(n)|^2} \qquad 3.1.16$$

Since $f_s > f_0 > 0$ (resulting in $e^{j2\pi\frac{f_0}{f_s}}$ imaginary part to be positive), the first solution is used only. Thus:

$$e^{j2\pi\frac{f_0}{f_s}} = \Re\{h(n)\} + j\sqrt{\Im^2\{h(n)\} - |g(n)|^2}$$

$$e^{j2\pi\frac{f_0}{f_s}} = |S|e^{j\left(\arctan\left(\frac{\sqrt{\Im^2\{h(n)\} - |g(n)|^2}}{\Re\{h(n)\}}\right)\right)}$$

With S, being any number, the phases can be equated:

$$2\pi\frac{f_0}{f_s} = \arctan\left(\frac{\sqrt{\Im^2\{h(n)\} - |g(n)|^2}}{\Re\{h(n)\}}\right)$$

$$f_0(n) = \frac{f_s}{2\pi}\arctan\left(\frac{\sqrt{\Im^2\{h(n)\} - |g(n)|^2}}{\Re\{h(n)\}}\right) \qquad 3.1.17$$

Hence proved as required in the equation 46 in the coursework.

e) For system frequency of $f_0 = 50Hz$, CLMS and ACLMS algorithms performances are shown in figure 3.1.7 both for balanced and unbalanced systems. For balanced system, the CLMS converges much faster and is the preferred algorithm. ACLMS also converges after an overshoot. The reason of CLMS converging faster is due to the nature of data being circular.

For the two unbalanced cases of phase and amplitude distortions, the results are shown in figure 3.1.7. In both cases, the ACLMS converges to the true value of $f_0 = 50Hz$ after an

overshoot but CLMS, does not converge and oscillates around lower means. This mean is lower in case of phase distortion.

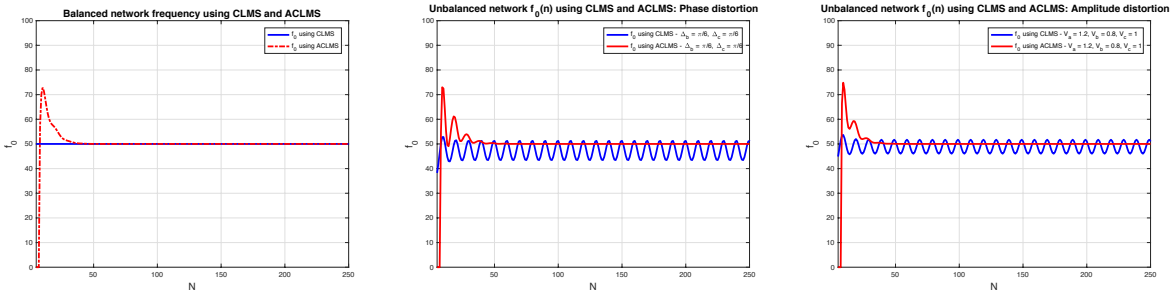


Figure 3.1.7: *CLMS and ACLMS perfoormance for balanced and unbalanced systems*

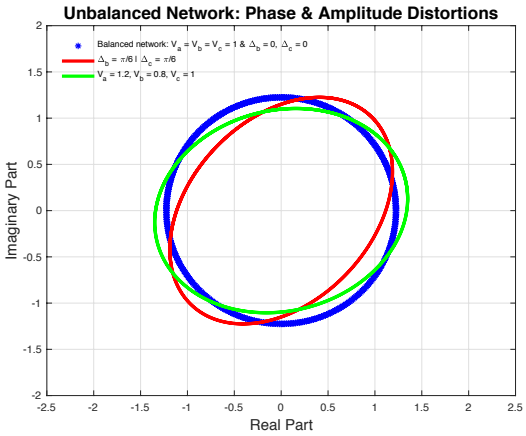


Figure 3.1.8: *Circularity comparison of the two unbalance cases considered in figures 1.3.7*

As explained above, the phase distortion results in even lower performance of CLMS that amplitude distortion in contrast to ACLMS. The reason for that can be observed in figure 3.1.8 as the data for amplitude distortion is more circular than data for the phase distortion and CLMS is designed for circular data whereas ACLMS can estimate both circular and non-circular.

3.2 Adaptive AR Model Based Time-Frequency Estimation

a) The frequency and the power of the frequency-modulated signal are shown in figure 3.2.1.

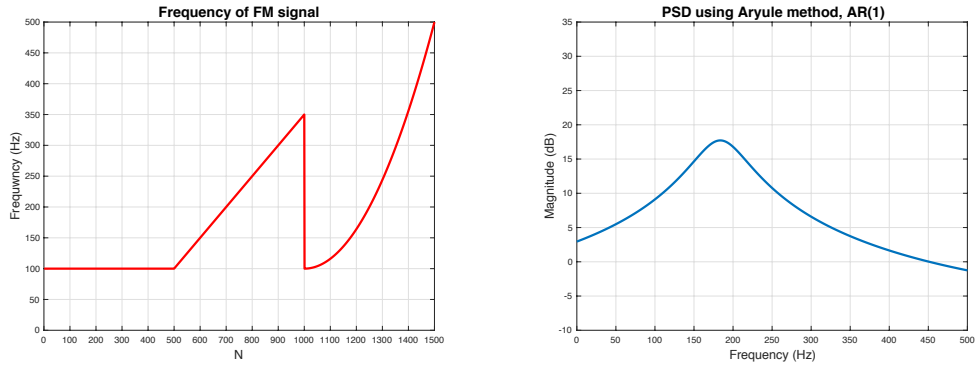
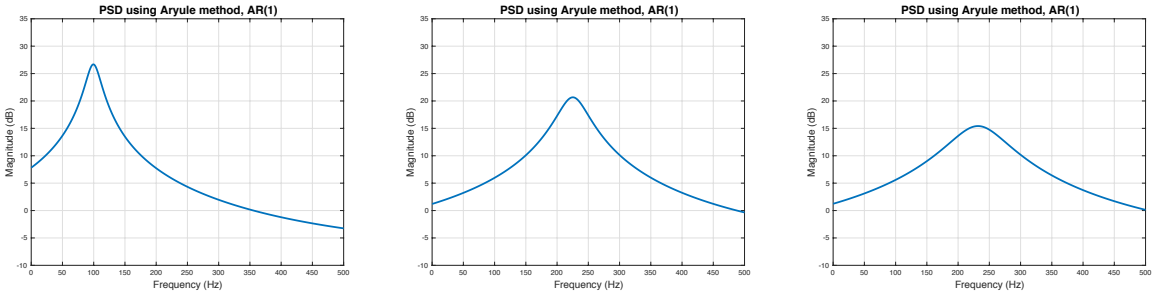


Figure 3.2.1: *Frequency and power of the FM signal*

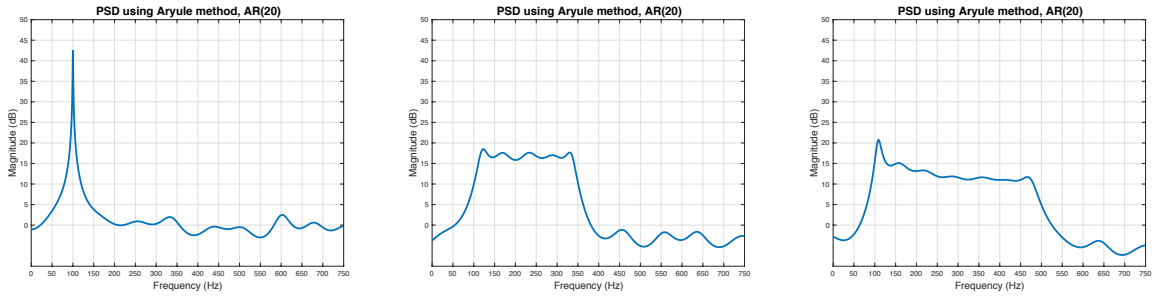
The frequency in figure 3.2.1 shows that the process is non-stationary and is constant in the interval $1-500$, linear in $501-1000$ and quadratic in $1001-1500$, as expected. The power spectrum is approximated using 'Aryule' method and shown in figure 2.3.1 using AR(1). Aryule assumes the process to be stationary and next it will be shown that why this graph is shifted to different frequencies when using different time intervals.

A block based estimation approach can be used to estimate different intervals of the process and the results can be observed as in figure 3.2.2 for AR(1) and figure 3.2.3 for AR(20). It was mentioned that using 'Aryule' one can capture the true estimation of the process in case of stationarity, hence constant frequency. The results can be verified in figures 2.3.2 and 3.2.3. The constant frequency block $(1-500)$ is captured both by AR(1) and AR(20). However, the two other blocks, linear increasing frequency, block $(501-1000)$, and quadratic increasing frequency, block $(1001-1500)$, are not approximated correctly either by AR(1) or AR(20) and the reason is again the 'Aryule' function only used for stationary processes.



Constant Frequency **Linear Frequency** **Quadratic Frequency**

Figure 3.2.2: *Block based estimation of frequency – AR(1)*



Constant Frequency **Linear Frequency** **Quadratic Frequency**

Figure 3.2.3: *Block based estimation of frequency – AR(20)*

b) Figure 3.2.3 shows how using CLMS can capture the non-stationarity of the signal. The coefficients of the process are now approximated adaptively. Changing the step size changes the speed of convergence and variance. Larger values of step size, μ, results in faster convergence with larger variance while smaller values of step size, μ, results in slower convergence with smaller variance. This can be verified by figure 2.2.3.

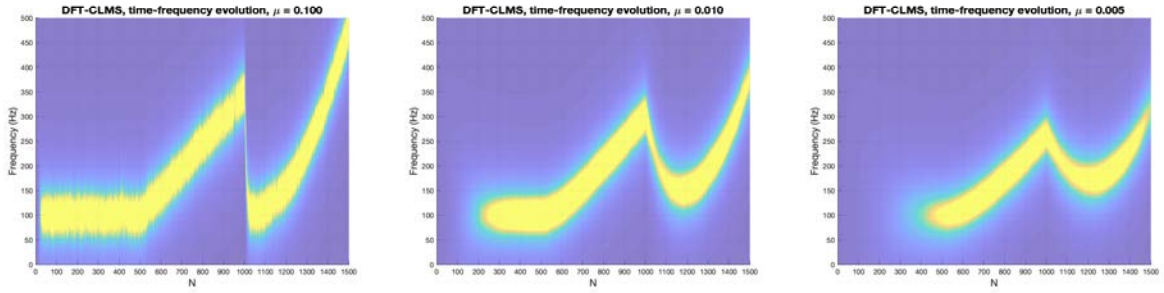


Figure 3.2.3: *time frequency evolution using CLMS with different step sizes.*

3.3 A Real Time Spectrum Analyzer Using Least Mean Square

a) Solving the least squares (LS) problem of equation 48 in the coursework guideline:

$$\min_{w} \|y - \hat{y}\|^2 = \min_{w} \sum_{n=0}^{N-1} |y(n) - \hat{y}(n)|^2 \qquad 3.3.1$$

Involves differentiating with respect to the weights w and equating to zero:

$$\frac{\partial}{\partial w} \|y - \hat{y}\|^2 = \frac{\partial}{\partial w} (y - \hat{y})^H (y - \hat{y})$$

$$\frac{\partial}{\partial w} (y - Fw)^H (y - Fw)$$

$$\frac{\partial}{\partial w} \left((y^H y) - (w^H F^H y) - (y^H F w) + (w^H (F^H F) w) \right)$$

$$-(y^H F) - (y^H F) + 2w(F^H F) = 0$$

$$2w^H (F^H F) = 2y^H F^H$$

$$(F^H F) w = Fy$$

Where w_* can be obtained and the equation proved:

$$w = (F^H F)^{-1} F y \qquad 3.3.2$$

Knowing the Inverse Discrete Fourier Transform (IDFT) for a signal $\hat{y}(n)$:

$$\hat{y}(n) = \frac{1}{N} \sum_{k=0}^{N-1} Y(k) e^{\frac{j2\pi nk}{N}} \qquad 3.3.3$$

With $W_{n,k} = e^{\frac{j2\pi nk}{N}}$, eq. 3.3.3 can be written in the matrix form as:

$$\begin{bmatrix} \hat{y}(0) \\ \vdots \\ \hat{y}(N-1) \end{bmatrix} = \begin{bmatrix} W_{0,0} & \cdots & W_{0,N-1} \\ \vdots & \ddots & \vdots \\ W_{N-1,0} & \cdots & W_{N-1,N-1} \end{bmatrix} \begin{bmatrix} Y(0) \\ \vdots \\ Y(N-1) \end{bmatrix}$$

Further reducing to:

$$\hat{y} = WY \qquad 3.3.4$$

Using 3.3.2 to get the Fourier Coefficients, Y, it can be written:

$$Y = (W^H W)^{-1} W y \qquad 3.3.5$$

Eq. 3.3.5 should give the signal $\hat{y}$ that minimizes the squared error between y and $\hat{y}$.

b) Eq. 3.3.4, $\hat{y} = WY$ which is the matrix form of IDFT, is the approximation of the true signal y. This approximation delivers the fact that Y is formed by projecting y into the weights, W. The approximated signal should be periodic such that $T = N$, whereas columns of W with basis as complex exponentials should be orthonormal.

c) Figure 3.3.1 shows the FM signal approximated with normal DFT-CLMS and leaky DFT-CLMS, $\gamma = 0.05$. The normal CLMS approximates the frequency nature of constant, linear and quadratic but the frequency at each iteration remains in the memory and is not cleared with the next update. The reason for the this can be the adaptability of the algorithm. The leaky CLMS, however, updates the weights with time rather than adapting them which leads to better results as shown in figure 3.3.1.

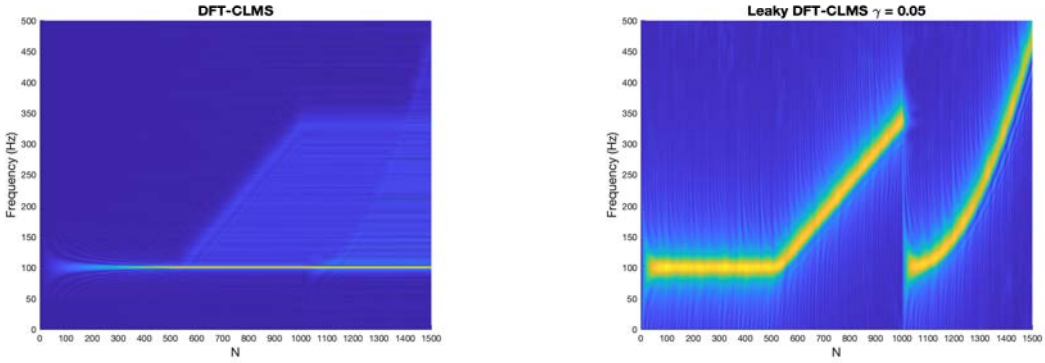


Figure 3.3.1: *DFT-CLMS, normal and leaky, time-frequency estimation*

d) Once again, the DFT-CLMS and leaky DFT-CLMS are used to estimate the spectrum of the EEG using the POz component. The strong 50Hz induced frequency by the power line is identified by both CLMS and leaky CLMS in figure 3.3.2. The SSEVP and low frequency alpha rhythms are also identifiable.
It was confirmed in the previous section that the leaky CLMS performed better with non-stationary processes while the non-leaky CLMS approximates the stationary processes better. This conclusion is readily approved by the results shown in figure 3.3.2 where the CLMS shows the peaks in the spectrum very clearly whereas the leaky CLMS fails to show these peaks to a great extent.

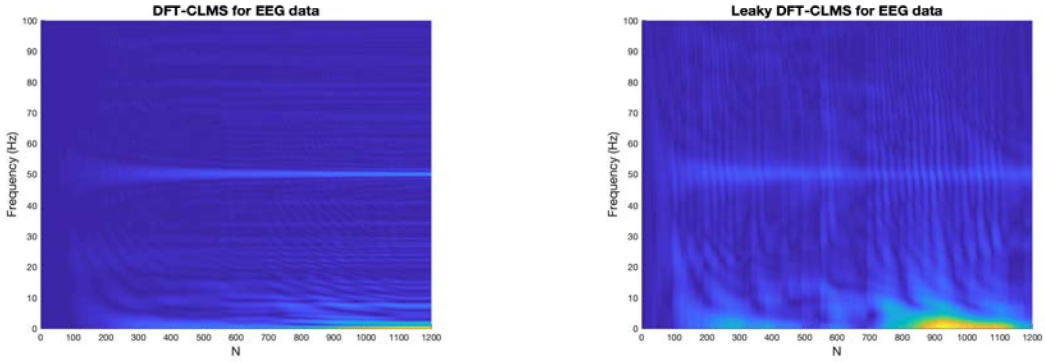


Figure 3.3.2: *time frequency approximation of EEG data using DFT-CLSM & Leaky DFT-CLMS*

4 From LMS to Deep Learning

1. Figure 4.1 shows the zero mean original time series (cyan) and the one step ahead prediction using LMS and an AR process of order 4, (red), with a learning rate of $\mu = 1 \times 10^{-5}$.
   The zoomed in figure shows how the one step ahead prediction adapts to the original signal. The error is larger in initialization and becomes smaller as the weights converge to their true values.
   The mean square error (MSE) is calculated and its value is 0.0401 for the one step ahead prediction and the prediction gain, $R_p = 5.1966$.

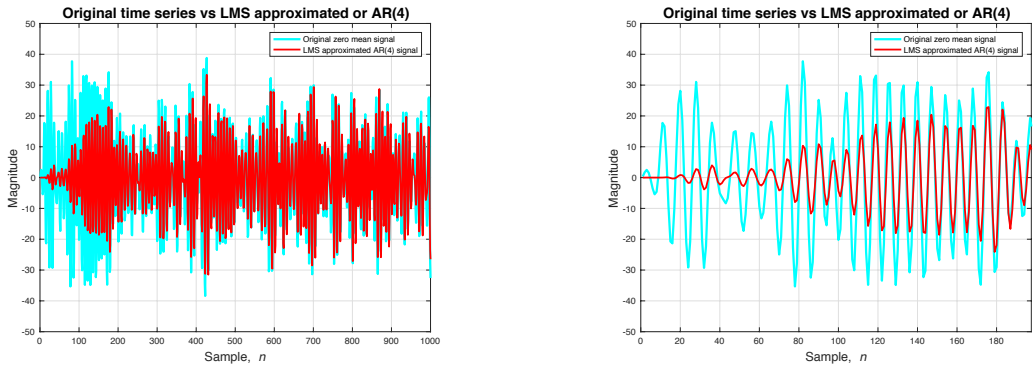


Figure 3.3.1: Original zero mean time series plotted against the one step ahead predicted signal using LMS

Figure 4.2 shows the error squared and its mean for the one step ahead prediction and its mean.

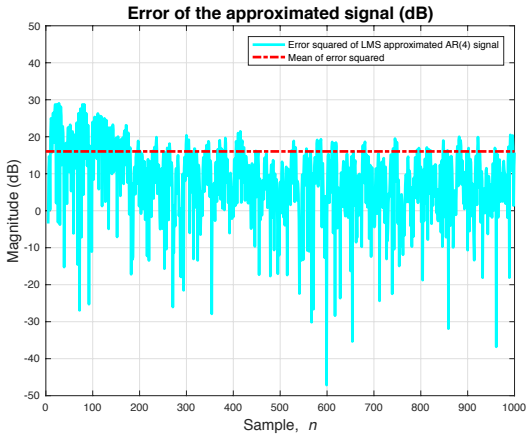


Figure 3.3.2: Error squared and its mean of $y(n)$ and $\hat{y}(n)$

2. Figure 4.3 shows the zero mean original time series (cyan) and the one step ahead prediction using LMS and an AR process of order 4 with activation function *tanh*, (red), with a learning rate of $\mu = 1 \times 10^{-5}$.
   The zoomed in figure shows how the one step ahead prediction adapts to the original signal. The error is larger in initialization and converges to a wrong value in later steps.
   The mean square error (MSE) is calculated and its value is now 0.1967 for the one step ahead prediction and the prediction gain, $R_p = -23.336$. These values show that the dynamical perceptron using *tanh* as the activation function is not appropriate and it should be adjusted to perform better and result in smaller values for MSE.

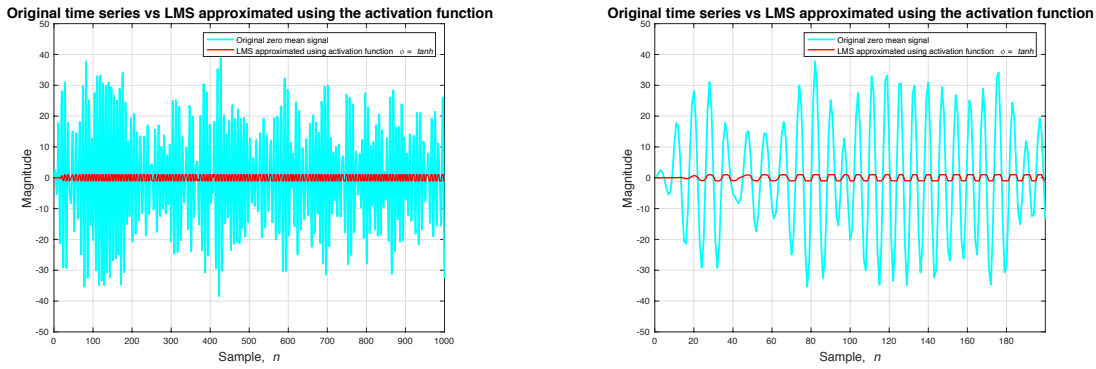


Figure 4.3: *Original zero mean time series plotted against the one step ahead predicted signal using LMS*

Figure 4.4 shows the error squared and its mean for the one step ahead prediction and its mean. As expected, the mean of error had increased.

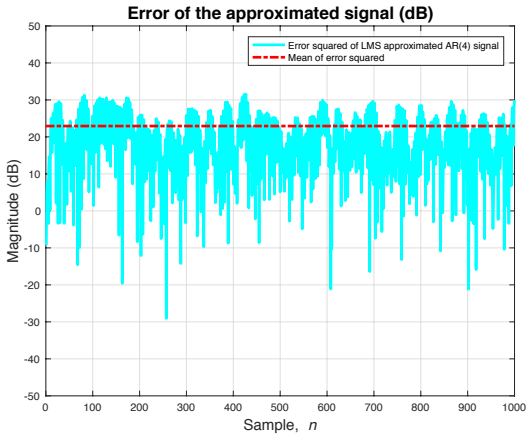


Figure 4.4: *Error squared and its mean of* $y(n)$ *and* $tanh(\hat{y}(n))$

3. To find the minimum MSE, one should find the optimal value of the scale for activation function, $a \times tanh$. Figure 4.5 shows the value of a giving the minimum MSE, which is equal to 13 with learning rate, $\mu = 1 \times 10^{-5}$.

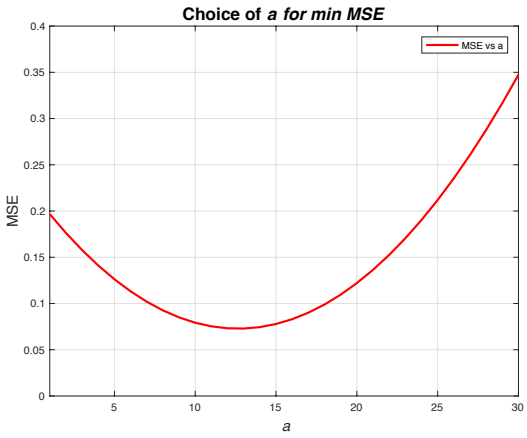


Figure 4.5: *activation function amplitude 'a' vs MSE*

Figure 4.6 shows the zero mean original time series (cyan) and the one step ahead prediction using LMS and an AR process of order 4 with modified activation function $a \times tanh$, (red), with a learning rate of $\mu = 1 \times 10^{-5}$.
The zoomed in figure shows how the one step ahead prediction adapts to the original signal. The error is larger in initialization and converges to closer values of the signal in later steps.

The mean square error (MSE) is calculated and its value is now 0.0729 for the one step ahead prediction and the prediction gain, $R_p = 3.252$. These values show that the dynamical perceptron using $a \times tanh$ as the scaled activation function result in better results than $tanh$ where $a = 1$, whereas in the latter case $a = 13$.

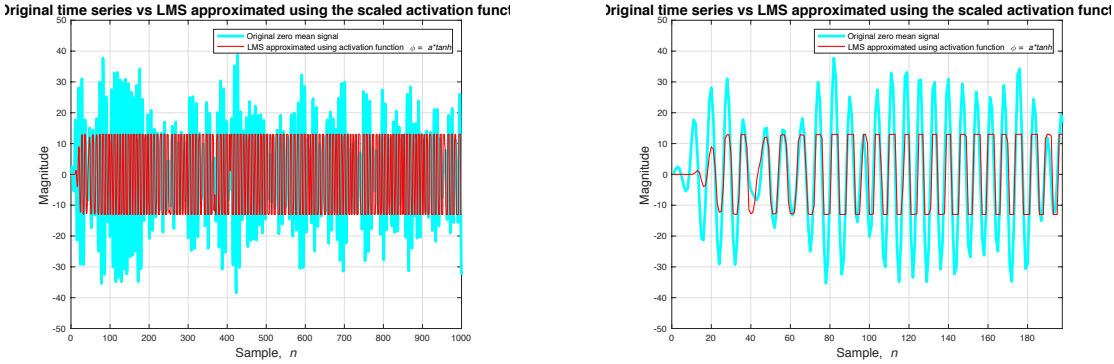


Figure 4.6: *Original zero mean time series plotted against the one step ahead predicted signal using LMS with activation fucntion* $a \times tanh(\hat{y}(n))$

Figure 4.7 shows the error squared and its mean for the one step ahead prediction and its mean. As expected, the mean of error is now lower than the case using activation function with unit scaling.

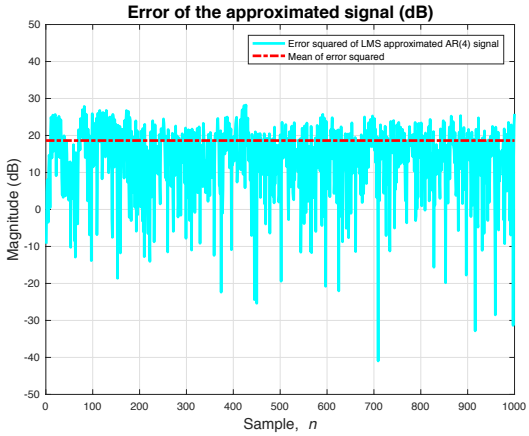


Figure 4.7: *Error squared and its mean of* $y(n)$ *and* $a \times tanh(\hat{y}(n))$

It can be observed that by changing the learning rate, μ, from 1×10^{-5} to 1×10^{-7} results in better approximation and different scale value for the activation function. With this change, the new values of mean square error (MSE) becomes 0.0631 with the activation function $a \times tanh$ which is closer to the MSE of original one step ahead prediction with $\mu = 1 \times 10^{-5}$.
Figure 4.8 shows the value of a giving the minimum MSE, which is equal to 13 with learning rate, $\mu = 1 \times 10^{-7}$.

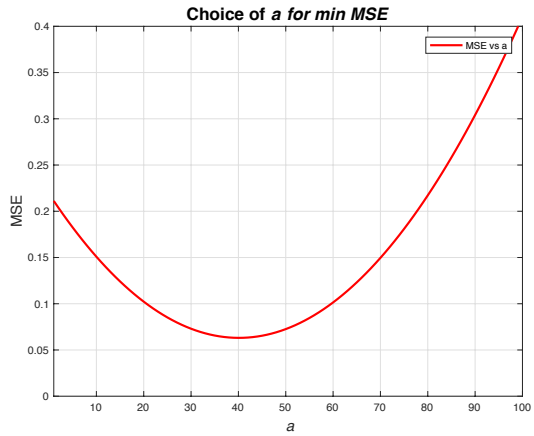


Figure 4.8: *activation function amplitude 'a' vs MSE*

Figure 4.9 shows the zero mean original time series (cyan) and the one step ahead prediction using LMS and an AR process of order 4 with modified activation function $a \times tanh$, (red), with a learning rate of $\mu = 1 \times 10^{-7}$.
The zoomed in figure shows how the one step ahead prediction adapts to the original signal. The error is larger in initialization and converges to closer values of the signal in later steps. The mean square error (MSE) is calculated and its value is now 0.0631 for the one step ahead prediction and the prediction gain, $R_p = 3.9$. These values show that the dynamical perceptron using $a \times tanh$ as the scaled activation function result in better results than $tanh$ where $a = 1$ and $a = 13$, whereas in this case $a = 40$.

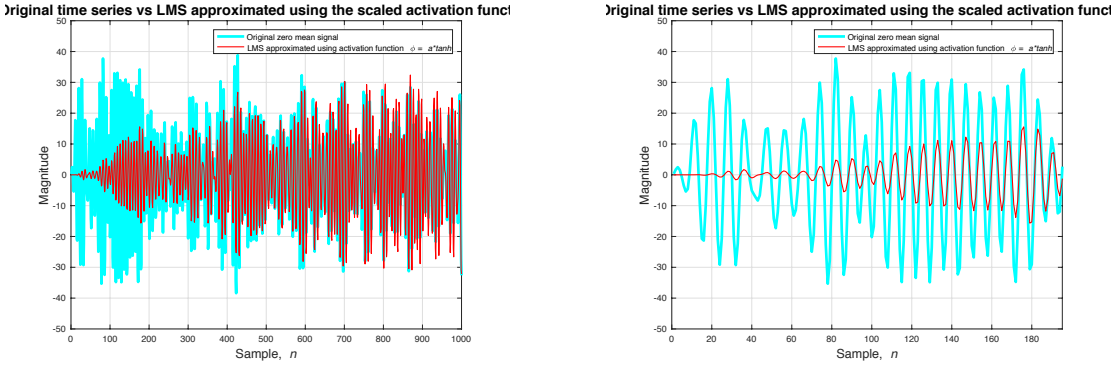


Figure 4.9: *Original zero mean time series plotted against the one step ahead predicted signal using LMS with activation fucntion* $a \times tanh(\hat{y}(n))$

4. To account for the mean, a bias is introduced to the model input as $\phi(w^T x + b)$. Figure 4.10 shows the minimum value of MSE for a range of a, where minimum MSE is achieved when $a = 13$.
   Figure 4.11 shows the zero mean original time series (cyan) and the one step ahead prediction using the biased LMS and an AR process of order 4 with modified activation function $a \times tanh$, (red), and a learning rate of $\mu = 1 \times 10^{-5}$.

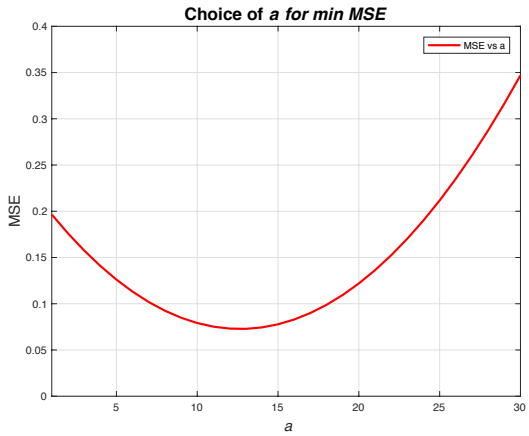


Figure 4.10: *activation function amplitude 'a' vs MSE*

The zoomed in figure shows how the one step ahead prediction deviates from the original signal. The error is larger in initialization and converges to closer values of the signal in later steps.

The mean square error (MSE) is calculated and its value is now 0.1967 for the one step ahead prediction and the prediction gain, $R_p = -0.4594$. Changing the step size from $\mu = 1 \times 10^{-5}$ to $\mu = 1 \times 10^{-7}$ results in enhanced results as shown in figure 4.12.

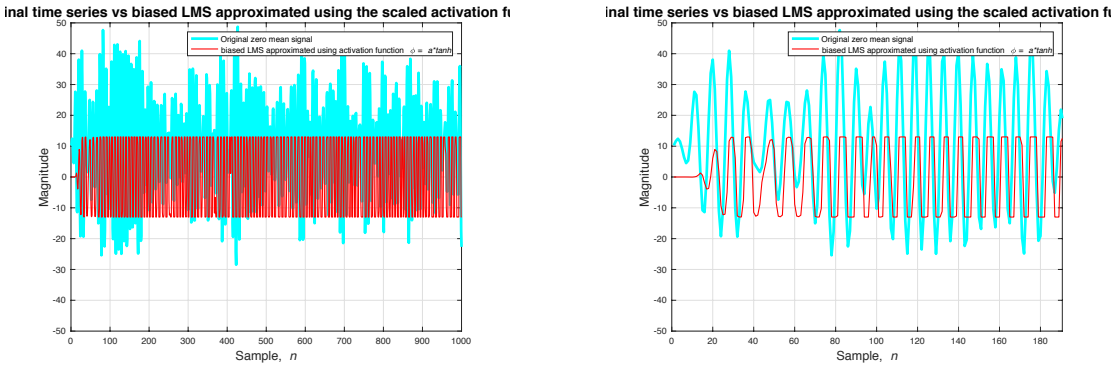


Figure 4.11: *Original zero mean time series plotted against the one step ahead biased predicted signal using LMS with activation fucntion* $a \times tanh(\hat{y}(n))$

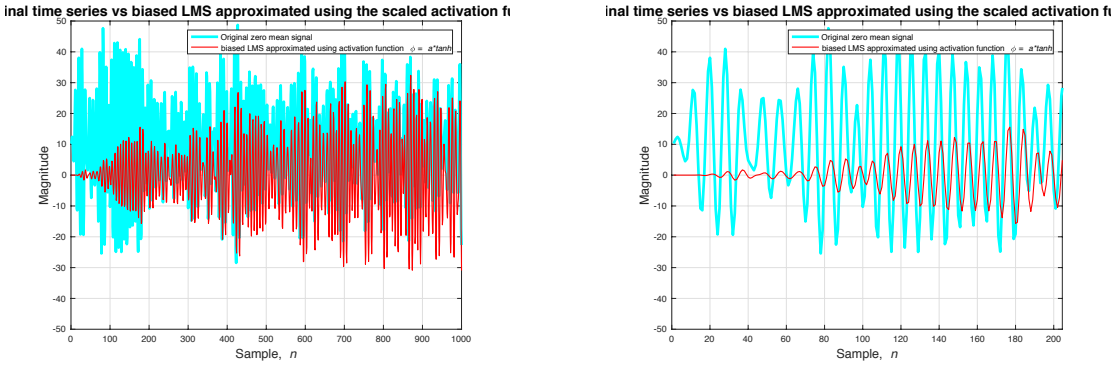


Figure 4.12: *Original zero mean time series plotted against the one step ahead biased predicted signal using LMS with activation fucntion* $a \times tanh(\hat{y}(n))$

5. Figure 4.13 shows the zero mean original time series (cyan) and the one step ahead prediction using LMS and an AR process of order 4 with pretrained weights, (red), with a learning rate of $\mu = 1 \times 10^{-5}$.
   The pre-trained weights were used from the overfitting of 20 first samples of the signal overall 100 times averaging of the weights.
   The zoomed in figure shows how the one step ahead prediction adapts to the original signal from the beginning rather than the other cases where it would take hundreds of steps to adapt to a reasonable value. The error is the same in initialization and later steps.
   The mean square error (MSE) is calculated and its value is now smaller than all other previous cases at 0.0168 for the one step ahead prediction and the prediction gain, $R_p = 11.292$. These values show that pre-training the weights hugely affect the overall performance of the algorithm.

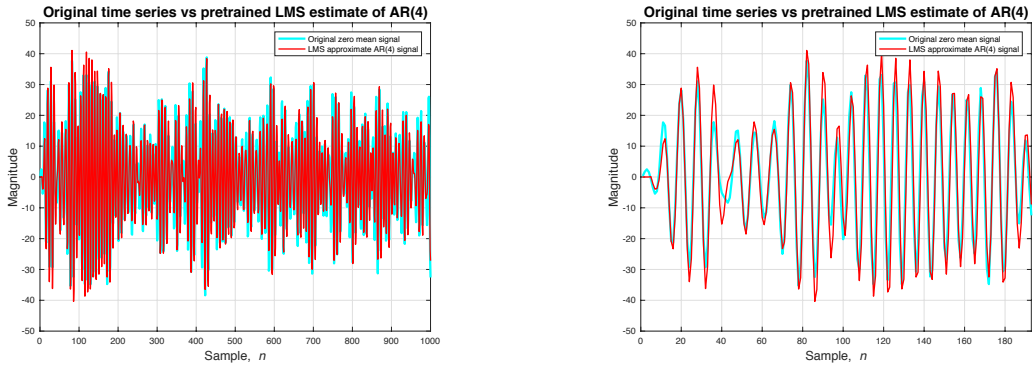


Figure 4.13: *Original zero mean time series plotted against the one step ahead predicted signal using LMS with pretrained weights*

Figure 4.14 shows the error squared and its mean for the one step ahead prediction and its mean. As expected, the mean of error is now lower than all other cases above.

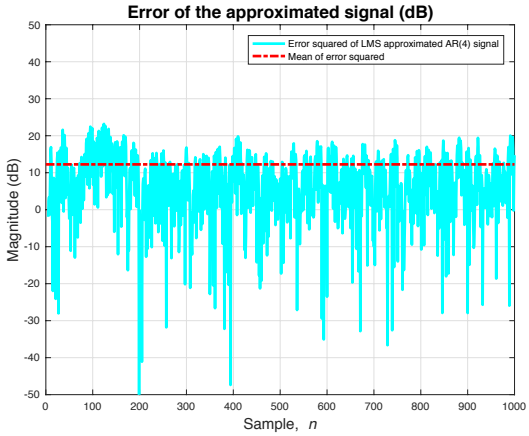


Figure 4.14: *Error squared and its mean of $\hat{y}(n)$*

5 Tensor Decomposition for Big Data Applications