Imperial College London

Adaptive Signal Processing and Machine Intelligence Coursework

Author

Haoxiang Huang

CID: 02470313

Taught by

Prof. Danilo P. Mandic

A coursework submitted in fulfillment of requirements for the module of
ELEC97003 - Adaptive Signal Processing and Machine Intelligence

Department of Electrical and Electronic Engineering
Imperial College London

2024

Contents

1

Classical and Modern Spectrum Estimation

1.1 Properties of Power Spectral Density (PSD)

1.1.1 Task a: Theoretical Proof

The Power Spectral Density (PSD) of an random sequences considered here is defined as the DTFT of the autocovarinace function (ACF) $r(k)$ as following:

$$P(\omega) = \sum_{k=-\infty}^{\infty} r(k) e^{-j\omega k}. \tag{1.1}$$

Under a mild assumption that the ACF $r(k)$ decays rapidly, that is

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=-(N-1)}^{N-1} |k||r(k)| = 0, \tag{1.2}$$

the definition in equation (1.1) can be rewritten as following:

$$P(\omega) = \lim_{N\to\infty} \mathbb{E}\left\{ \left| \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-jn\omega} \right|^2 \right\}. \tag{1.3}$$

Proof. Alternatively, the DFT can also be formulated as the inner product between the signal $\mathbf{x}$

and a complex sinusoidal basis $\mathbf{f}$:

$$X(\omega_m) = \mathbf{f}^H \mathbf{x} = \sum_{k=0}^{N-1} x[k] e^{-j\omega k} \in \mathbb{C} \tag{1.4}$$

where

$$\mathbf{f} = \left[1, e^{j\omega}, e^{j2\omega}, e^{j3\omega} \cdots e^{j(N-1)\omega}\right]^T \in \mathbb{C}^N \tag{1.5}$$

Thus, the Equation (1.3) can be transformed into vector representation:

$$P(\omega) = \lim_{N\to\infty} \mathbb{E}\left\{\frac{1}{N}\mathbf{f}^H \mathbf{x}\mathbf{x}^H \mathbf{f}\right\} \tag{1.6a}$$

$$= \lim_{N\to\infty} \frac{1}{N}\mathbf{f}^H \begin{bmatrix} r(0) & r(-1) & \cdots & r(-(N-1)) \\ \vdots & \vdots & \ddots & \vdots \\ r(N-1) & r(N-2) & \cdots & r(0) \end{bmatrix} \mathbf{f} \tag{1.6b}$$

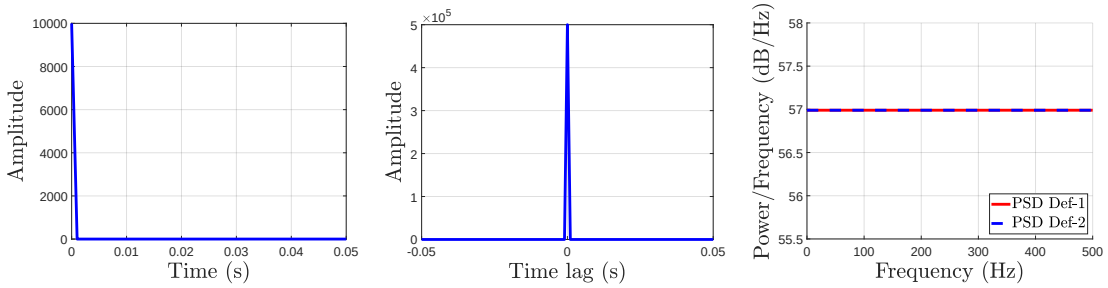
$$= \lim_{N\to\infty} \sum_{k=-(N-1)}^{N-1} \frac{N-|k|}{N} r(k) e^{-j\omega k} \tag{1.6c}$$

$$= \sum_{k=-\infty}^{\infty} r(k) e^{-j\omega k} - \lim_{N\to\infty} \sum_{k=-(N-1)}^{N-1} |k| r(k) e^{-j\omega k} \tag{1.6d}$$

Obviously, the first term in equation (1.6d) is the first definition of PSD in equation (1.1). When the second term satisfies the condition in equation (1.1), the two definition of PSD is equal. □

1.1.2 Task b: Simulation Validation

Consider given impulse signal $x_1(t) = \delta(t)$ and a composite sinusoidal signal $x_2(t)$ with White Gaussian Noise (WGN) added, where the Signal-to-Noise Ratio (SNR) is set to 0 dB.

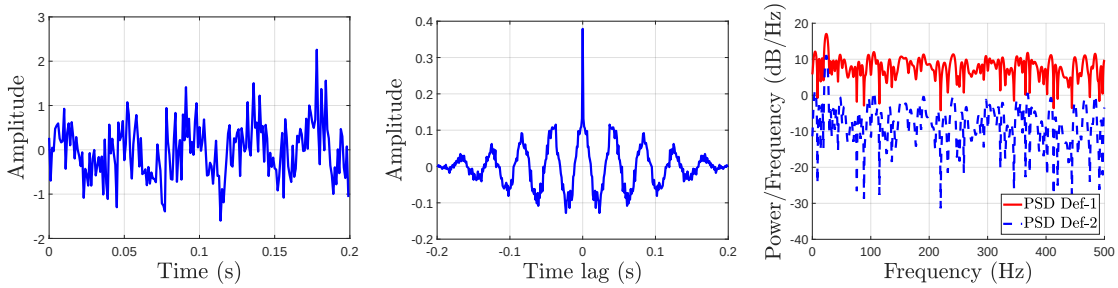


(a) Impulse Signal Sequence $x_1(t)$ (b) ACF of Impulse Signal $x_1(t)$ (c) PSD of Impulse Signal $x_1(t)$

Figure 1.1: ACF and PSD of the Impulse Signal

The simulation results for the impulse signal $x_1(t)$, as illustrated in Figure 1.1, show that its

ACF decays rapidly. Consequently, its PSD keeps consistent across both definitions. While the ACF of sinusoidal signal plus WGN decays slowly as shown in 1.2, its PSD is different between each definitions.



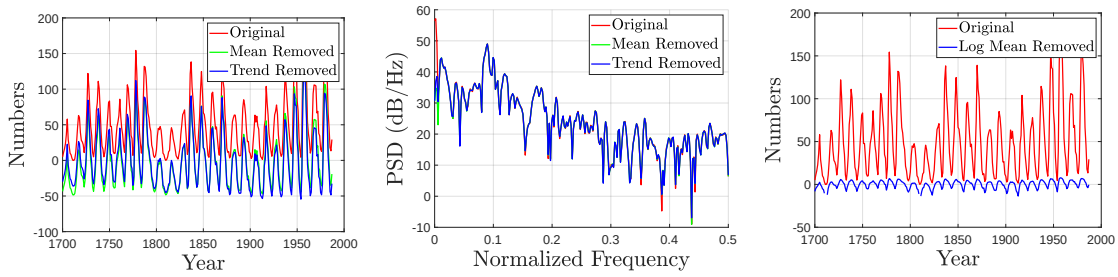
(a) Sinusoidal Signal Sequence $x_2(t)$ (b) ACF of Sinusoidal Signal $x_2(t)$ (c) PSD of Sinusoidal Signal $x_2(t)$

Figure 1.2: ACF and PSD of the Sinusoidal Signal

1.2 Periodogram-based Methods Applied to Real–World Data

1.2.1 Task a: Sunspot Time Series Dataset

Appling Mean removed and Tread removed to Sunspot Time Series Dataset, the time series results are illustrated in Figure 1.3(a) and PSD in Figure 1.3(b). In Figure 1.3(b), removing the mean from the signal eliminates the DC component (0 Hz), whereas the trend removed operation does not have significant removes at DC component. These operations similarly affect other frequency components, resulting in nearly same spectral plots for the high frequency spectrum. Furthermore, Figure 1.3(c) compares original to mean-removed logarithmic sunspot data. The logarithmic transformation compresses the data, attenuating magnitudes greater than 1. This results in a smoother spectrum, enhancing the distinguishability of key peaks.

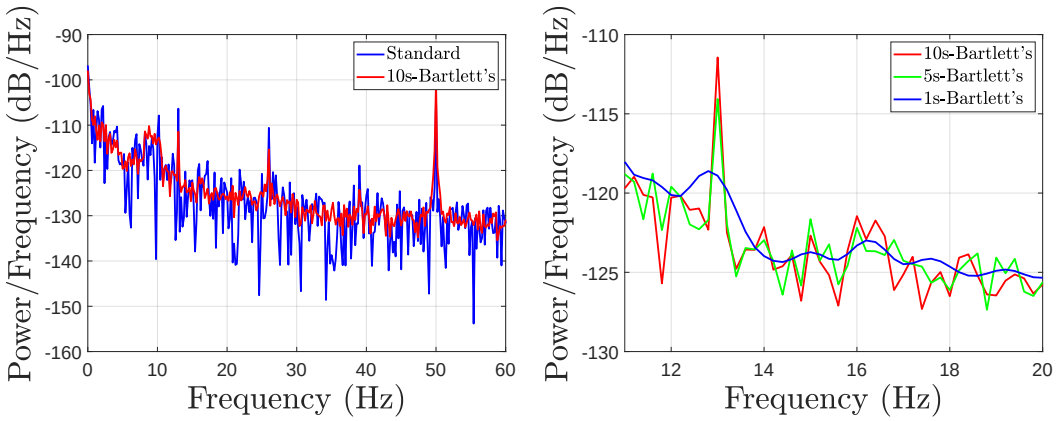


(a) The original, mean-removed, trend-removed sunspot data (b) The original, mean-removed and trend-removed PSD (c) The original and logarithmic sunspot time series data

Figure 1.3: Processing of sunspot series dataset

1.2.2 Task b: EEG Signal

This stimulus generates a response in the EEG known as the steady state visual evoked potential (SSVEP), which matches the stimulus frequency. The frequency can be clearly identified, as shown in Figure 1.4, at 13Hz. Figure 1.4(a) shows long frequency range of PSD spectrum compared standard spectrum periodogram with 10's window length Bartlett's method. Notably, strong response are observed at 8-10Hz due to the factor that subject was tired and at 50Hz due to electrical interference from power lines. Apart from the fundamental frequency response peak at 13Hz, the harmonics of the response also can be found in 26Hz, 39Hz in Figure 1.4(a). Figure 1.4(b) shows that the more longer windows of Bartlett's method reveals more distinct peaks for easier identification (higher the resolution of PSD).



(a) Comparison of Standard Spectrum Periodogram and Bartlett's method

(b) Different Window length of Bartlett's method

Figure 1.4: Processing of EEG Signal

1.3 Correlation Estimation

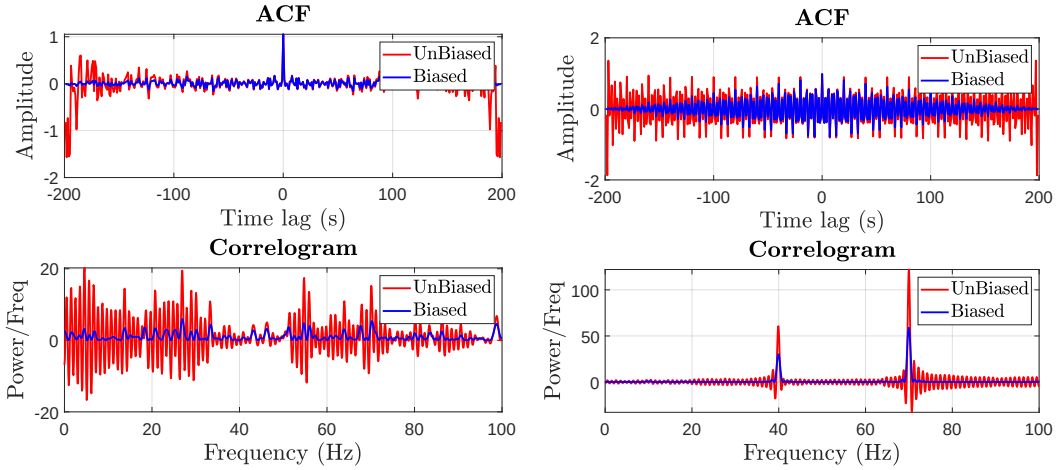
1.3.1 Task a: Biased and Unbiased Estimators

Consider the given WGN signal and a composite sinusoidal signal with White Gaussian Noise (WGN) for the analysis of biased and unbiased estimators

$$x_1(t) = w\,(t) \sim \mathcal{N}(0,1) \tag{1.7a}$$

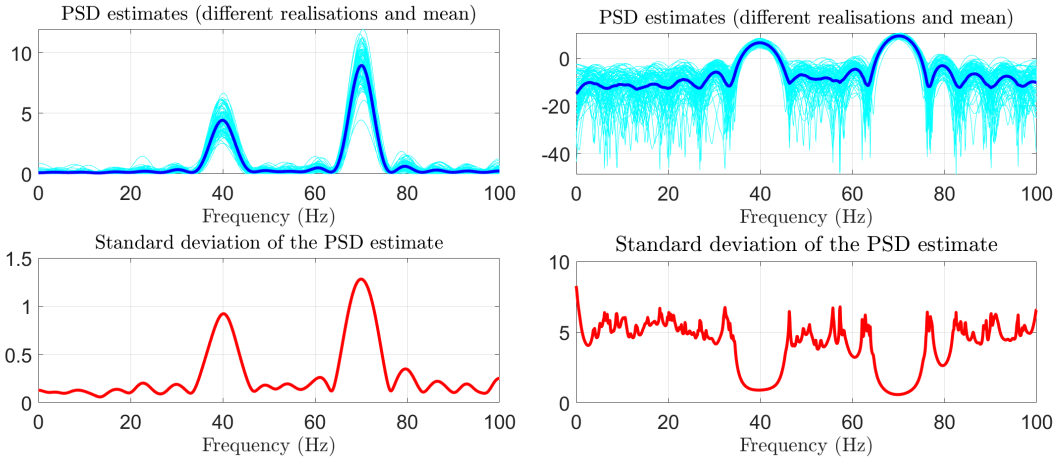
$$x_2\,(t) = 0.8\sin\,(40\pi t) + 1.1\sin\,(70\pi t) + 0.3w\,(t) \tag{1.7b}$$

Figure 1.5 illustrates the ACF and the Correlogram PSD of both biased and unbiased estimates for the noisy sinusoidal signal defined in equation (1.7b) and WGN signal defined in equation (1.7a). We can find that the unbiased and biased ACF are overlapping in the lower lag. While the unbiased ACF shows unstable higher oscillation in large lag than biased ACF. This stability also can be evidenced by less magnitude in the biased Correlogram PSD of both signal.



(a) WGN signal (b) Sinusoidal signal

Figure 1.5: ACF (top) and Correlogram PSD (bottom) of sampled signal



(a) Plotting in Decimal (b) Plotting in dB

Figure 1.6: Different realisation (top) and Standard deviation (bottom) of the PSD Estimation

1.3.2 Task b & c: Plotting the PSD in dB

In this task, the composite sinusoidal signal with White Gaussian Noise (WGN) defined in equation 1.7b is applied to generate the PSD estimate of 100 realisations. The Figure 1.6(a) plots different

realisation and standard deviation of the PSD estimation in decimal. It's observed that the spectral estimates vary more widely at peaks. This is also evidenced by the higher standard deviation at 40 and 70Hz. However, the peak is important factor in spectral estimation. The problem of high uncertainty at peak can be tackled by plotting the PSD in dB as shown in Figure 1.6(b), where standard deviation at peak is the lowest point.

1.3.3 Task d: Generation of Complex Exponential Signals

In this task, following complex exponential signal is generated:

$$x_5(t) = e^{j2\pi\times 0.3t} + e^{j2\pi\times 0.32t} + 0.2n(t) \tag{1.8}$$

The resolution of the periodogram is inversely proportional to the number of data points N, meaning that a larger N leads to more precise spectrum estimates. Figure 1.7 illustrates how spectrum estimates become more precise from $N = 30$ to $N = 60$. When the length of the signal is low, specifically $N = 30$, the two frequency components of the signal (0.3Hz and 0.32Hz) cannot be distinguished in the periodogram, allowing only an estimation of their general range. As the the number of data points N increase, these two frequency components can be accurately distinguished, and the periodogram begins to reveal the correct line spectra. Additionally, the PSD value of side-lobe components diminishes, further facilitating the identification of the main-lobe frequency components.

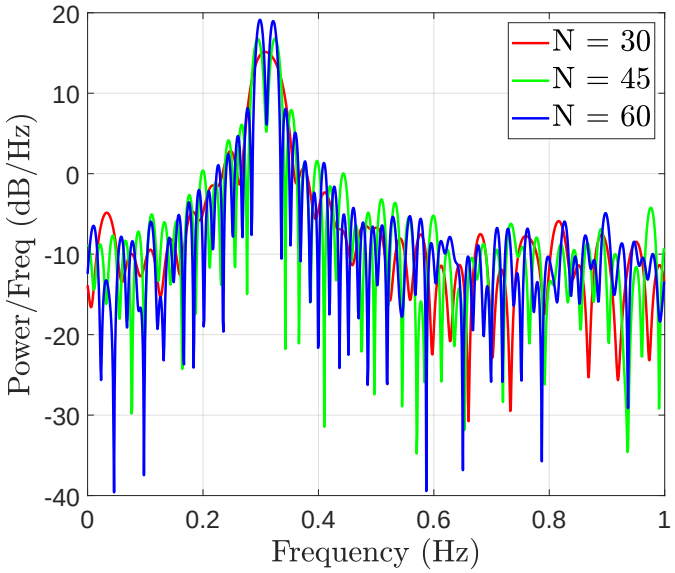


Figure 1.7: Periodogram of different data points N

1.3.4 Task e: Frequency Estimation by MUSIC

In this task, the MUSIC method is utilized to identify the desired line spectra. The original signal is formulated as Equation (1.8). The following Matlab code snippet demonstrates the implementation of the MUSIC method:

```
[X,R] = corrmtx(x,14,'mod');
[S,F] = pmusic(R,2,[ ],1,'corr');
plot(F,S,'linewidth',2); set(gca,'xlim',[0.25 0.40]);
grid on; xlabel('Hz'); ylabel('Pseudospectrum');
```

The first code line `[X,R] = corrmtx(x,14,'mod');` generate a 14th-order modified covariance matrix X and its correlation matrix R from signal x using the corrmtx function. The second line `[S,F] = pmusic(R,2,[ ],1,'corr');` aims to get the pseudospectrum estimates via the MUSIC algorithm. Here, the first argument is R, the signal's autocorrelation matrix; the second argument specifies the signal subspace dimension as 2; the third argument utilizes default FFT sizes; the fourth argument denotes the sampling frequency, and the final argument confirms the first argument as the correlation matrix. The output S yields the pseudospectrum estimates, and F represents the associated frequency values. The remaining code part plots the etiamated spectrum displayed in Figure 1.8. It is evident that the two frequency peaks at 0.3 Hz and 0.32 Hz are easily distinguishable, with the spectrum magnitude at other frequencies being suppressed to mostly zero. Although the signal length N is 30, MUSIC shows superior resolution to the periodogram. However, a limitation of the MUSIC are its requirement for prior knowledge or estimation of the number of signal sources. Besides, its performance is sensitive to the noise and correlated signal.

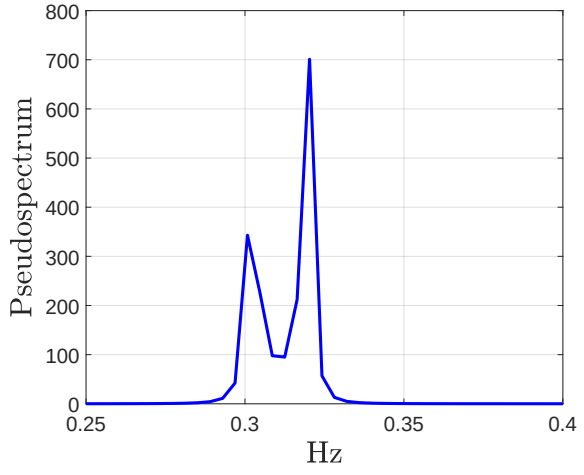


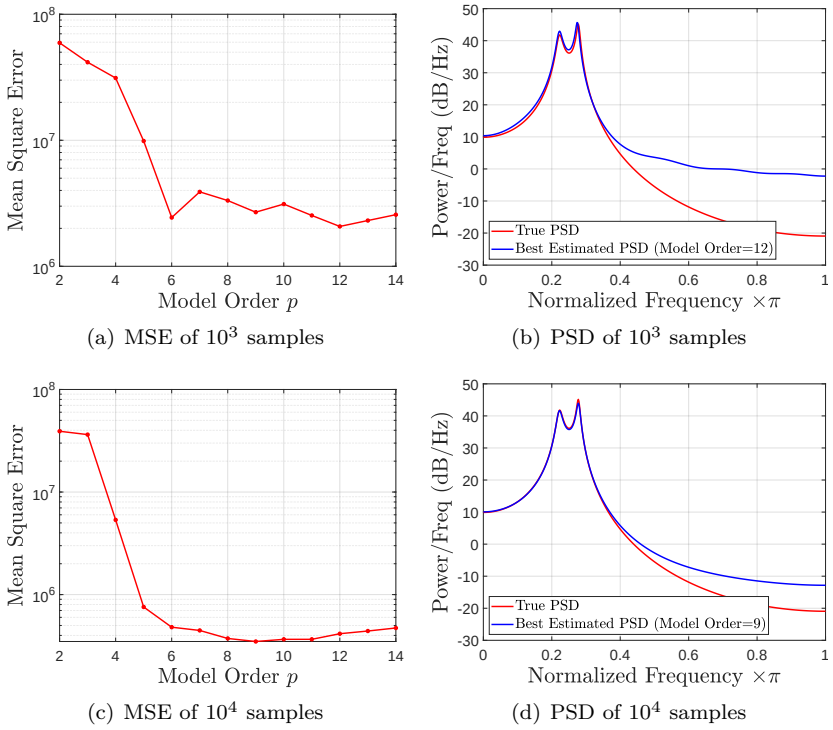
Figure 1.8: Spectrum estimated by MUSIC (N=30)

1.4 Spectrum of Autoregressive Processes

To estimate the coefficients of an Autoregressive Process, we use the formula from the Yule-Walker equation given by:

$$R_{xx}a = r_x \Rightarrow a = R_{xx}^{-1}r_x \tag{1.9}$$

where Rxx is the autocorrelation matrix of the signal. Employing the biased estimator of the ACF ensures that the matrix is positive definite and thus invertible. However, the unbiased ACF will not decay to zero rapidly, which may lead to uncertainty and does not guarantee the matrix will be positive definite.This potentially making it impossible to compute the inverse.



(a) MSE of 10^3 samples (b) PSD of 10^3 samples

(c) MSE of 10^4 samples (d) PSD of 10^4 samples

Figure 1.9: Mean Square Error(left) and PSD estimation (right) of the AR Process

Here, the AR coefficient is considered as following:

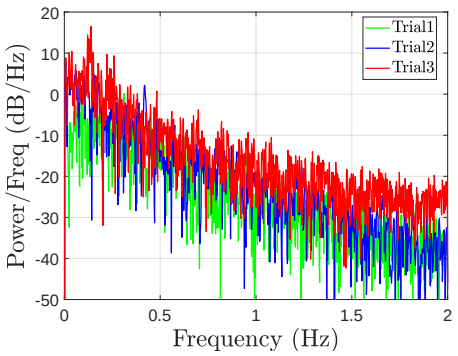
$$a = \begin{bmatrix} 2.76 & -3.81 & 2.65 & -0.92 \end{bmatrix}^T \tag{1.10}$$

The Figure 1.9 shows the true PSD and optimal order of AR estimated PSD with 10^3 and 10^4 samples. When the model order is set lower than the true order AR(4) (under-modeling), there is a significant decrease in error as the model order increases. Conversely, in the case of over-modeling, increasing order cannot gain significant error decrease.

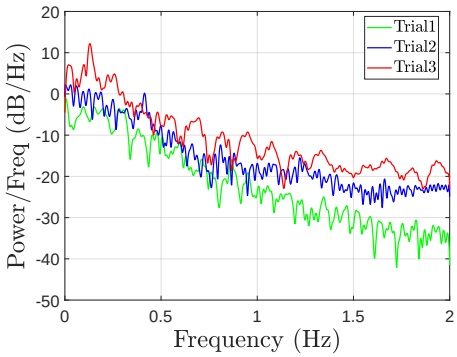
1.5 Real World Signals: Respiratory Sinus Arrhythmia from RR-Intervals

1.5.1 Task a & b: Periodogram Estimates of PSD

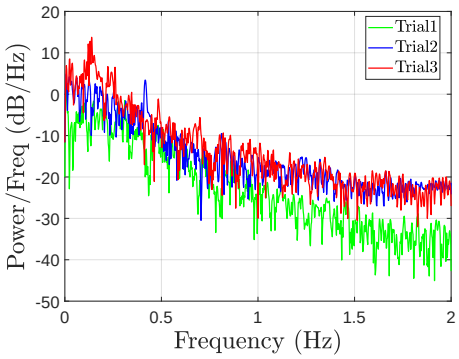
Figure 1.10(a) presents the standard Periodogram spectrum for three RRI trial datasets. Figures 1.10(b) and 1.10(c) display the Bartlett Periodogram spectra for window lengths of 50s and 100s, respectively, demonstrating that a larger window length yields higher resolution. For the trials, only trial 1's PSD estimates lack clear peaks. Conversely, trial 2 exhibits a peak at approximately 0.41 Hz, corresponding to around 25 breaths per minute, and trial 3 identifies a peak at approximately 0.13 Hz, corresponding to around 7.5 breaths per minute.



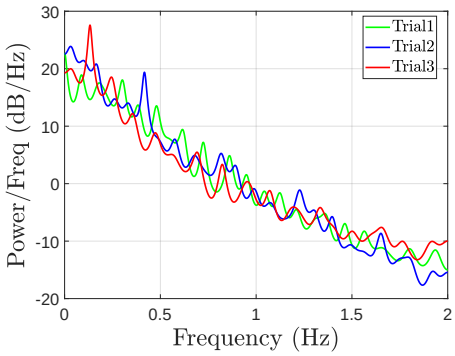
(a) Standard Periodogram



(b) Bartlett Periodogram (windows length is 50s)



(c) Bartlett Periodogram (windows length is 150s)



(d) AR spectrum estimation (P=50).

Figure 1.10: Periodogram and AR spectrum estimation for the RRI data

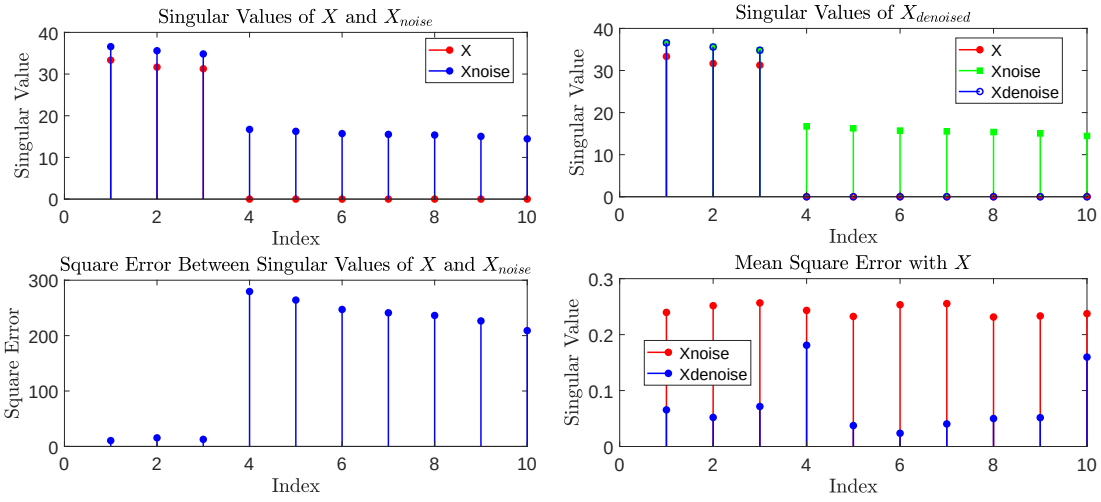
1.5.2 Task c: AR Spectrum Estimates

Figure 1.10(d) illustrates the AR spectrum estimation with 50 model order. It is evident that the AR spectrum estimates successfully identify frequency peaks. Comparing with the periodogram spectrum, the AR spectrum shows greater smoothness and stability with reduced variance, facilitating easier peak distinction.

1.6 Robust Regression

1.6.1 Task a & b: SVD Denoise

Figure 1.11(a) illustrates Singular values and the square errors between X and Xnoise. we can identify that the rank of X is 3, evidenced by the first three singular values significantly surpassing subsequent ones in magnitude. Examination of the square error plot at the bottom of Figure 1.11(a) highlights the impact of noise on these singular values, notably increasing those that were zero in the original dataset.



(a) Singular values (top) and the square errors(bottom) between X and Xnoise

(b) Singular values (top) and the square errors (bottom) among X, Xnoise and Xdenoise

Figure 1.11: SVD Analysis among X, Xnoise and Xdenoise

Denoising of $\mathbf{X}_{\text{noise}}$ is achievable by using only the r most significant principal components, where r is the matrix's rank. The denoised low-rank matrix, $\mathbf{X}_{\text{denoise}}$, is expressed as

$$\mathbf{X}_{\text{denoise}} = \mathbf{U}(:, 1:r)\mathbf{S}(1:r, 1:r)\mathbf{V}^H(:, 1:r), \tag{1.11}$$

where $\mathbf{U}$, $\mathbf{S}$, and $\mathbf{V}$ derived from the singular value decomposition (SVD) of $\mathbf{X}_{\text{noise}} = \mathbf{USV}^H$. The

comparison among $\mathbf{X}$, $\mathbf{X}_{\text{denoise}}$ and $\mathbf{X}_{\text{noise}}$ is shown in Figure 1.11(b). We can find $\mathbf{X}_{\text{denoise}}$ keep the same value at 3 lower rank and zero out other components as shown in the top of Figure 1.11(b). In the bottom of Figure 1.11(b), for the error between ground-truth $\mathbf{X}$, $\mathbf{X}_{\text{denoise}}$ is less than $\mathbf{X}_{\text{noise}}$.

1.6.2 Task c & d: OLS and PCR

The Performance of OLS and PCR method on training set, test set and extra 10,000 testing set is summarized as Table 1.1. For the training set, the mean square error (MSE) for Ordinary Least Squares (OLS) is 0.7103, whereas for Principal Component Regression (PCR) it is 0.7154, indicating a slightly better performance by OLS. However, within the testing set, OLS achieves an MSE of 0.4755 compared to PCR's MSE of 0.4703, suggesting superior performance by PCR. Moreover, across extra 10,000 testing set , the MSE values observed for OLS and PCR are 0.4713 and 0.4714.

Table 1.1: Performance of OLS and PCR

	OLS	PCR
Training set	0.7103	0.7154
Testing set	0.4755	0.4703
10,000 testing set	0.4713	0.4714

2

Adaptive signal processing

2.1 The Least Mean Square (LMS) Algorithm

2.1.1 Task a: Convergence of LMS in the Mean

Consider the following second-order AR process:

$$x(n) = a_1 x(n-1) + a_2 x(n-2) + \eta(n) \tag{2.1}$$

where $\eta(n) \sim \mathcal{N}(0, \sigma_\eta^2)$ is the Gaussian noise. In this task, we aim to implement a LMS estimator for AR coefficients a_1 and a_2 using signal x(n).

The correlation matrix of input signal $\mathbf{x}(n) = [x(n-1), x(n-2)]^T$ is defined as:

$$\mathbf{R} = \mathbb{E}\left\{\mathbf{x}(n)\mathbf{x}^T(n)\right\} = \begin{bmatrix} \mathbb{E}\left\{x(n-1)x(n-1)\right\} & \mathbb{E}\left\{x(n-1)x(n-2)\right\} \\ \mathbb{E}\left\{x(n-2)x(n-1)\right\} & \mathbb{E}\left\{x(n-2)x(n-2)\right\} \end{bmatrix} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(1) & r_{xx}(0) \end{bmatrix} \tag{2.2}$$

The elements of the correlation matrix can be determined by Yule-Walker equation as follows:

$$\begin{aligned} r_{xx}(0) &= a_1 r_{xx}(1) + a_2 r_{xx}(2) + \sigma_n^2 \\ r_{xx}(1) &= a_1 r_{xx}(0) + a_2 r_{xx}(1) \\ r_{xx}(2) &= a_1 r_{xx}(1) + a_2 r_{xx}(0) \end{aligned} \tag{2.3}$$

Given the original parameters $a_1 = 0.1$, $a_2 = 0.8$, and $\sigma_\eta^2 = 0.25$, the solution of Equation (2.2)

and (2.3) is:

$$\mathbf{R} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(1) & r_{xx}(0) \end{bmatrix} = \begin{bmatrix} 0.9259 & 0.4630 \\ 0.4630 & 0.9259 \end{bmatrix} \tag{2.4}$$

The eigenvalue matrix Λ for the correlation matrix $\mathbf{R}$ is derived as follows:

$$\Lambda = \begin{bmatrix} 1.3889 & 0 \\ 0 & 0.4629 \end{bmatrix} \tag{2.5}$$

In order to guarantee the LMS to converge to the Wiener optimal solution's mean, the step size μ must satisfy:

$$0 < \mu < \frac{2}{\lambda_{max}} = \frac{2}{1.3899} \approx 1.439 \tag{2.6}$$

where λ_{max} is the largest eigenvalue of the correlation matrix $\mathbf{R}$.

2.1.2 Task b: Learning Curve

Due to the unpredictability of learning, learning curves for a single realization are very noisy and have a large variance. Hence, averaging across various realizations (e.g., 100 realizations) obtained through Monte Carlo simulation is necessary to reveal a clearer trend. Figure 2.1 shows the learning curve of squared error (over a single realization) and mean squared error (over 100 realizations) with step sizes of 0.05 and 0.01, respectively. The learning curve demonstrates that the LMS algorithm begins to converge at approximately 100 steps for a step size (μ) of 0.05. In contrast, for a step size of 0.01, convergence starts around 200 steps. This confirms that a larger step size enables the LMS algorithm to converge more quickly.

2.1.3 Task c & d: Excess Error

Here, we estimate the LMS misadjustment $\mathcal{M}$ by averaging the steady-state of learning curves over 100 realizations, comparing it to the theoretical value from Equation (2.8). This misadjustment $\mathcal{M}$, defined as the ratio of the additional error power from the adaptive filter (EMSE) to the minimum MSE of a Wiener filter, is detailed in Equation (2.7).

$$\mathcal{M} = \lim_{n\to\infty} \frac{EMSE}{\sigma_\eta^2} = \lim_{n\to\infty} \frac{\mathbb{E}\{\epsilon^2(n)\} - \sigma_\eta^2}{\sigma_\eta^2} \tag{2.7}$$

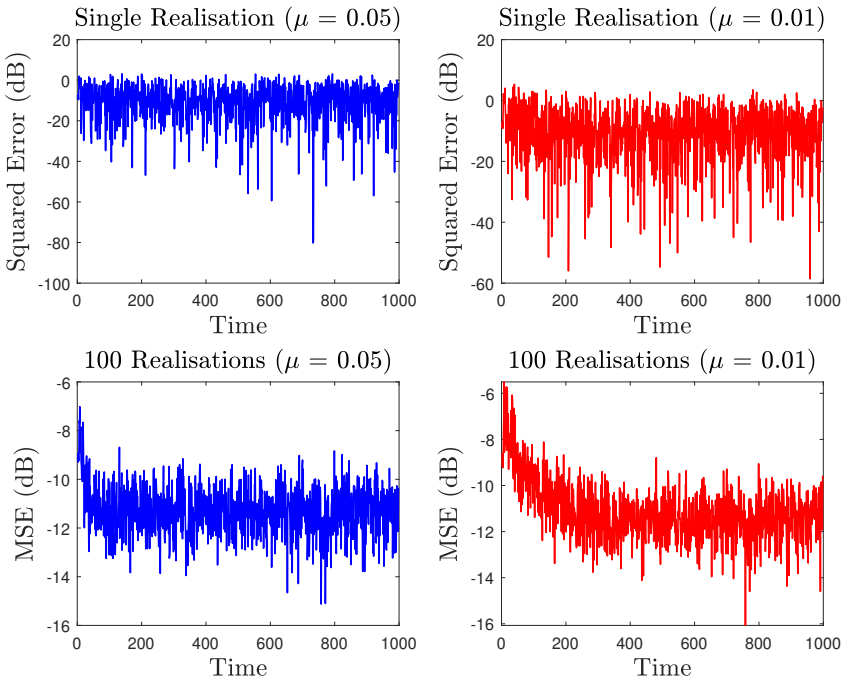


Figure 2.1: The learning curve of LMS

The theoretical misadjustment $\mathcal{M}_{\text{LMS}}$ for small step sizes is defined as follows:

$$\mathcal{M}_{\text{LMS}} \approx \frac{\mu}{2}\text{Tr}\{R\} \tag{2.8}$$

Based on Equations (2.7) and (2.8), the comparison between theoretical and estimated misadjustments is presented in Table 2.1. It is observed that the estimated misadjustment consistently surpasses the theoretical misadjustment. Nonetheless, with the adoption of a smaller step size (e.g., $\mu = 0.01$), the estimated results approach much nearer to the theoretical values.

Table 2.1: Comparison of theoretical and estimated misadjustments.

μ	$\mathcal{M}$	$\mathcal{M}_{\text{LMS}}$	$\lvert\mathcal{M}_{\text{LMS}} - \mathcal{M}\rvert$
0.05	0.0486	0.0463	0.0023
0.01	0.0099	0.0093	0.0006

Additionally, the autoregressive (AR) coefficients, $\hat{a}_1$ and $\hat{a}_2$, can also be estimated by calculating the average of the steady-state values from the Monte Carlo simulation over 100 realizations, as presented in Table 2.2. It is also evident that employing a smaller step size (e.g. $\mu = 0.01$) results in reduced error in the estimated coefficients.

Table 2.2: Comparison of theoretical and estimated misadjustments.

μ	$\hat{a}_1$	$\hat{a}_2$	$\lvert\hat{a}_1 - a_1\rvert$	$\lvert\hat{a}_2 - a_2\rvert$
0.05	0.071	0.715	0.029	0.085
0.01	0.107	0.752	0.007	0.048

2.1.4 Task e & f: The Leaky LMS Algorithm

In this task, we aim to formulate the update rule for the LMS coefficient, $\mathbf{w}(n)$, utilizing the Steepest Descent method. This method minimizes the cost function presented in Equation (2.9) to its optimal value.

$$\begin{aligned}
J_2(n) &= \mathbb{E}\{J_2(n)\} \\
&= \frac{1}{2}\mathbb{E}\left\{e^2(n) + \gamma\|\mathbf{w}(n)\|_2^2\right\} \\
&= \frac{1}{2}\mathbb{E}\left\{e^2(n)\right\} + \frac{1}{2}\gamma\|\mathbf{w}(n)\|_2^2 \\
&= \frac{1}{2}\mathbb{E}\left\{\left(x(n) - \mathbf{w}^T(n)\mathbf{x}(n)\right)\left(x(n) - \mathbf{w}^T(n)\mathbf{x}(n)\right)^T\right\} + \frac{1}{2}\gamma\mathbf{w}^T(n)\mathbf{w}(n) \\
&= \frac{1}{2}\left(\underbrace{\mathbb{E}\{x^2(n)\}}_{\sigma_x^2} - 2\mathbf{w}^T(n)\underbrace{\mathbb{E}\{\mathbf{x}(n)x(n)\}}_{\mathbf{p}} + \mathbf{w}^T(n)\underbrace{\mathbb{E}\{\mathbf{x}(n)\mathbf{x}^T(n)\}}_{\mathbf{R}}\mathbf{w}(n)\}\right) + \frac{1}{2}\gamma\mathbf{w}^T(n)\mathbf{w}(n) \\
&= \frac{1}{2}\sigma_x^2 - \mathbf{w}^T(n)\mathbf{p} + \frac{1}{2}\mathbf{w}^T(n)\mathbf{R}\mathbf{w}(n) + \frac{1}{2}\gamma\mathbf{w}^T(n)\mathbf{w}(n)
\end{aligned} \tag{2.9}$$

Here, the gradient of the cost function $J_2(n)$ can be derived as follows:

$$\begin{aligned}
\nabla_{\mathbf{w}} J_2(n) &= -\mathbf{p} + \frac{1}{2}\left(\mathbf{R} + \mathbf{R}^T\right)\mathbf{w}(n) + \gamma\mathbf{w}(n) \\
&= -\mathbf{p} + \mathbf{R}\mathbf{w}(n) + \gamma\mathbf{w}(n)
\end{aligned} \tag{2.10}$$

The update of the coefficients in the Steepest Descent method represents the most direct path towards the minimization of the cost function $J_2(n)$, as delineated as follows:

$$\begin{aligned}
\mathbf{w}(n+1) &= \mathbf{w}(n) + \mu\left(-\nabla_{\mathbf{w}} J_2(n)\right) \\
&= \mathbf{w}(n) + \mu\left(\mathbf{p} - \mathbf{R}\mathbf{w}(n) - \gamma\mathbf{w}(n)\right) \\
&= (1 - \mu\gamma)\mathbf{w}(n) + \mu(\mathbf{p} - \mathbf{R}\mathbf{w}(n))
\end{aligned} \tag{2.11}$$

In the LMS, instantaneous estimates are used to substitute $\mathbf{p}$ and $\mathbf{R}$, specifically, $\hat{\mathbf{p}} = \mathbf{x}(n)x(n)$

and $\hat{\mathbf{R}} = \mathbf{x}(n)\mathbf{x}^T(n)$. Therefore, the leaky LMS equation is obtained as follows:

$$\begin{aligned}\mathbf{w}(n+1) &= (1-\mu\gamma)\mathbf{w}(n) + \mu(\hat{\mathbf{p}} - \hat{\mathbf{R}}\mathbf{w}(n)) \\ &= (1-\mu\gamma)\mathbf{w}(n) + \mu\mathbf{x}(n)\underbrace{\big(x(n) - \mathbf{x}^T(n)\mathbf{w}(n)\big)}_{e(n)} \\ &= (1-\mu\gamma)\mathbf{w}(n) + \mu e(n)\mathbf{x}(n) \end{aligned} \tag{2.12}$$

In the implementation of the leaky LMS algorithm to estimat AR coefficients of Equation (2.1), various values of μ and γ are employed. The outcomes are presented in Table 2.3.

Table 2.3: AR coefficients estimation using leaky LMS algorithm

	γ	$\hat{a}_1$	$\hat{a}_2$
	0.01	0.075	0.708
$\mu = 0.05$	0.5	0.108	0.409
	5	0.051	0.105
	0.01	0.107	0.747
$\mu = 0.01$	0.5	0.141	0.468
	5	0.058	0.116

For a given value of γ, a smaller μ leads to better estimates. Conversely, holding μ constant, an increment in γ causes the leaky LMS algorithm to converge to inaccurate coefficient values, trending towards zero. This observation can be attributed to the optimal Wiener-Hopf solution $\mathbf{w}_{\text{opt}} = \mathbf{R}^{-1}\mathbf{p}$, whereas the Leaky-LMS weights follow Equation $\mathbf{w}_{\text{optLeaky}} = (\mathbf{R} + \gamma\mathbf{I})^{-1}\mathbf{p}$. This implies that as γ approaches zero, the optimal Leaky-LMS weights approach the Wiener-Hopf solution. Therefore, as γ increases, the deviation from the optimal Wiener-Hopf solution increases.

2.2 Adaptive Step Sizes

2.2.1 Task a: GASS Algorithm

In this task, we aims to investigate different gradient adaptive step size (GASS) algorithms including the Benveniste, Ang & Farhang, and the Mathews & Xie algorithms. Employing these, the

weight update is revised as:

$$w(n+1) = w(n) + \mu(n)e(n)x(n) \tag{2.13}$$

Subsequently, the adaptive step size rule is formulated as:

$$\mu(n+1) = \mu(n) + \rho e(n)x^T(n)\psi(n) \tag{2.14}$$

The performance of three GASS algorithms was evaluated and compared against the MA(1) model described in Equation (2.15):

$$x(n) = 0.9\eta(n-1) + \eta(n), \quad \eta \sim \mathcal{N}(0, 0.5) \tag{2.15}$$

The standard LMS algorithm is implemented with step sizes of $\mu = 0.01$ and $\mu = 0.1$. For GASS algorithms, the initial step size for weight updates is uniformly set to $\mu(0) = 0.01$. Additionally, the step size for updating $\mu(n)$ is fixed at $\rho = 0.0003$. The comparative performance of various algorithms against the standard LMS is illustrated in Figure 2.2, generated by averaging results from 100 trials.

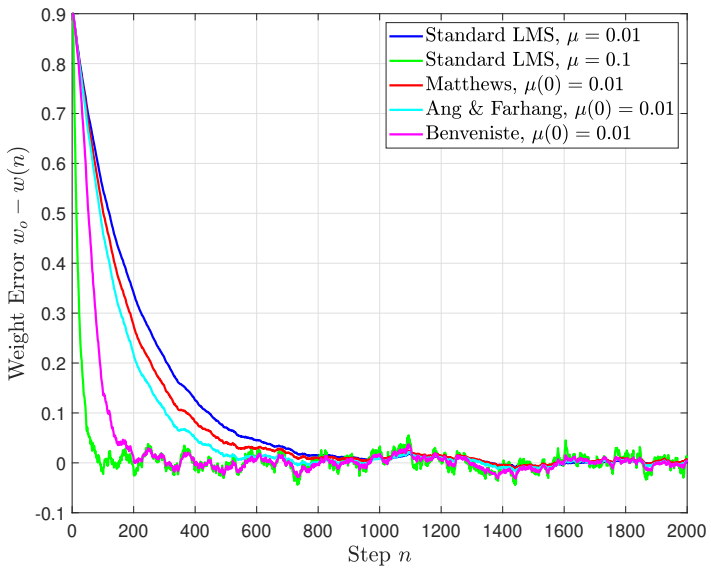


Figure 2.2: Comparison of LMS Algorithm with GASS Variants

In general, while GASS algorithms generally offer better convergence speed over LMS, this advantage may come at the expense of increased steady-state error. For details, the standard LMS algorithm, with $\mu = 0.1$, achieves the fastest convergence at the cost of the highest steady-state error. Benveniste's method closely follows in convergence speed, starting from a lower μ value,

yet it yields a lower steady-state error compared to the standard LMS with $\mu = 0.1$. Although Farhang's and Matthews's methods converge more slowly than Benveniste's, they demonstrate superior steady-state error performance. Conversely, the standard LMS with $\mu = 0.01$ exhibits the slowest convergence but a steady-state error smaller than that of Benveniste's method.

2.2.2 Task b: NLMS Algorithm

In this task, we aim to verify the update equation defined as Equation (2.16) is equivalent to weight update equation for the NLMS algorithm.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e_p(n)\mathbf{x}(n) \tag{2.16}$$

where $e_p(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n+1)$.

First, let $\Delta\mathbf{w}(n) = \mathbf{w}(n+1) - \mathbf{w}(n)$. Consequently, we obtain:

$$\begin{aligned}
\Delta\mathbf{w}(n) &= \mathbf{w}(n+1) - \mathbf{w}(n) \\
&= \mu e_p(n)\mathbf{x}(n) \qquad (2.17) \\
&= \mu \left(d(n) - \mathbf{x}^T(n)\left(\mathbf{w}(n) + \Delta\mathbf{w}(n)\right)\right)\mathbf{x}(n) \\
&= \mu(\underbrace{d(n) - \mathbf{x}^T(n)\mathbf{w}(n)}_{e(n)} - \mathbf{x}^T(n)\Delta\mathbf{w}(n))\mathbf{x}(n) \\
&= \mu e(n)\mathbf{x}(n) - \mu\mathbf{x}^T(n)\mathbf{x}(n)\Delta\mathbf{w}(n) \qquad (2.18)
\end{aligned}$$

Rearrange this expression:

$$(1 + \mu\mathbf{x}^T(n)\mathbf{x}(n))\Delta\mathbf{w}(n) = \mu e(n)\mathbf{x}(n) \tag{2.19}$$

$$\Delta\mathbf{w}(n) = \frac{1}{\frac{1}{\mu} + \|\mathbf{x}(n)\|^2} e(n)\mathbf{x}(n) \tag{2.20}$$

By assigning $\beta = 1$ and $\epsilon = 1/\mu$:

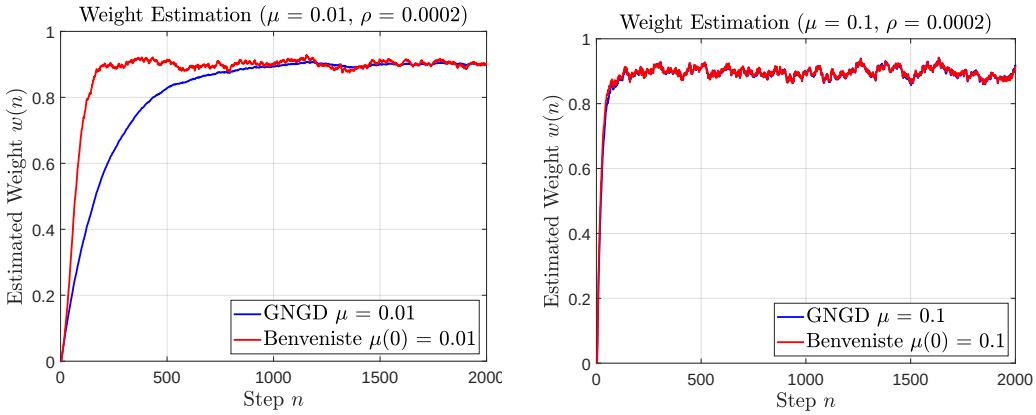
$$\mathbf{w}(n+1) = \mathbf{w}(n) + \Delta\mathbf{w}(n) = \mathbf{w}(n) + \frac{\beta}{\epsilon + \|\mathbf{x}(n)\|^2} e(n)\mathbf{x}(n) \tag{2.21}$$

where Equation (2.21) is the definition of weight update equation for normalized LMS (NLMS) algorithm. Therefore, Equation (2.16) is equivalent to the NLMS algorithm.

2.2.3 Task c: GNGD Algorithm

The gain of the Normalized Least Mean Squares (NLMS) algorithm can be rendered adaptive by the introduction of a time-varying regularization factor, $\epsilon(n)$. This adaptation is achieved in the Generalized Normalized Gradient Descent (GNGD) algorithm through a gradient method, as given by:

$$\epsilon(n+1) = \epsilon(n) - \rho\mu \frac{e(n)e(n-1)\mathbf{x}^T(n)\mathbf{x}(n-1)}{\left(\epsilon(n-1) + \|\mathbf{x}(n-1)\|^2\right)^2} \tag{2.22}$$



(a) $\mu = 0.01, \rho = 0.0002$ (b) $\mu = 0.1, \rho = 0.0002$

Figure 2.3: Comparison of GNGD and Benveniste's algorithms

Here, $\beta = 1$ and $\epsilon = 1/\mu$ are set as we obtained in the section 2.2.2. Figure 2.3 compares the GNGD algorithm and Benveniste's algorithm in identifying the MA(1) system defined in Equation (2.15). For a small step size of $\mu = 0.01$, the GNGD algorithm demonstrates slower convergence compared to Benveniste's algorithm, yet it achieves superior performance in steady-state error. As the step size increases to $\mu = 0.1$, both algorithms exhibit nearly identical convergence rates.

In computational complexity, Benveniste's algorithm requires a total of $2M^2 + 5M + 3$ multiplications and $2M^2 + 5M - 1$ additions per step, including operations to update $\psi(n)$, $e(n)$, $\mu(n)$ and $w(n)$. On the contrary, the GNGD algorithm requires $5M + 8$ multiplications and $5M$ additions per step, for updates to $\epsilon(n)$, $e(n)$ and $w(n)$. Thus, Benveniste's algorithm exhibits $O(M^2)$ complexity, while GNGD's complexity is $O(M)$, making GNGD have lower computational burden.

2.3 Adaptive Noise Cancellation

2.3.1 Task a & b: Adaptive Line Enhancer (ALE)

In this task, we investigate Adaptive Line Enhancer (ALE) applied to the signal corrupted by noise as delineated as follows:

$$s(n) = x(n) + \eta(n) \tag{2.23}$$

where x(n) represents a sinusoid with unit amplitude and an angular frequency of $\omega_0 = 0.01\pi$ and $\eta(n)$ is the coloured noise which is given as

$$\eta(n) = v(n) + 0.5v(n-2), \quad v(n) \sim \mathcal{N}(0,1) \tag{2.24}$$

First, we consider the Mean Square Error (MSE) as follows:

$$\begin{aligned}
\mathbb{E}\{(\hat{s}(n) - \hat{x}(n))^2\} &= \mathbb{E}\{(x(n) + \eta(n) - \hat{x}(n))^2 \\
&= \mathbb{E}\{(x(n) - \hat{x}(n))^2\} + 2\mathbb{E}\{(x(n) - \hat{x}(n))\eta(n)\} + \mathbb{E}\{\eta(n)^2\} \\
&= \mathbb{E}\{(x(n) - \hat{x}(n))^2\} + +\mathbb{E}\{\eta(n)^2\} + 2\mathbb{E}\{(x(n)\eta(n)\} - 2\mathbb{E}\{\hat{x}(n)\eta(n)\}
\end{aligned} \tag{2.25}$$

Examining the equation (2.25), the first term $E\left[(x(n) - \hat{x}(n))^2\right]$ is independent of noise. The second term $E\left[\eta^2(n)\right]$ is a DC component. The third term $2E\left[x(n)\eta(n)\right]$ vanishes, since the signal and noise are uncorrelated. The final term, being dependent on Δ via $\hat{x}(n)$, is the primary subject of further analysis. Therefore we have:

$$\begin{aligned}
\mathbb{E}\{\hat{x}(n)\eta(n)\} &= \mathbb{E}\{\mathbf{w}^T(n)\mathbf{u}(n)\eta(n)\} \\
&= \mathbb{E}\left\{\sum_{k=0}^{M-1} w_k s(n-\Delta-k)\eta(n)\right\} \\
&= \mathbb{E}\left\{\sum_{k=0}^{M-1} w_k \left[x(n-\Delta-k) + \eta(n-\Delta-k)\right]\eta(n)\right\} \\
&= \mathbb{E}\left\{\sum_{k=0}^{M-1} w_k \eta(n-\Delta-k)\eta(n)\right\} \\
&= \mathbb{E}\left\{\sum_{k=0}^{M-1} w_k \left[v(n-\Delta-k) + 0.5v(n-\Delta-k-2)\right]\eta(n)\right\}
\end{aligned} \tag{2.26}$$

Given the white noise properties, where $\mathbb{E}\{v(n)v(n-m)\} = 0$ for $m \geq 1$, the minimum value of Δ to minimize the MSE is determined to be 3, as this setting causes Equation (2.26) to equal zero. Figure 2.4 illustrates significant enhancement in the signal realizations for $\Delta > 2$.

In the Part b, LMS adaptive filters with orders 5, 10, 15, and 20 are evaluated. the delay Δ ranging from 3 to 25 is considered, and the step size is set to $\mu = 0.01$. The outcomes averaged over 100 trials are depicted in Figure 2.5(a). For filters of varying orders, the mean square prediction

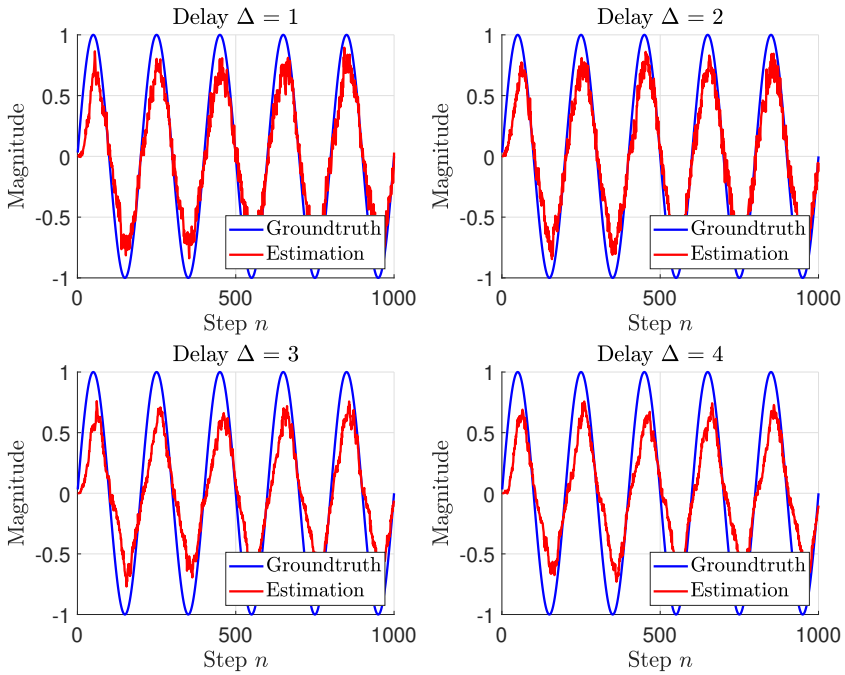
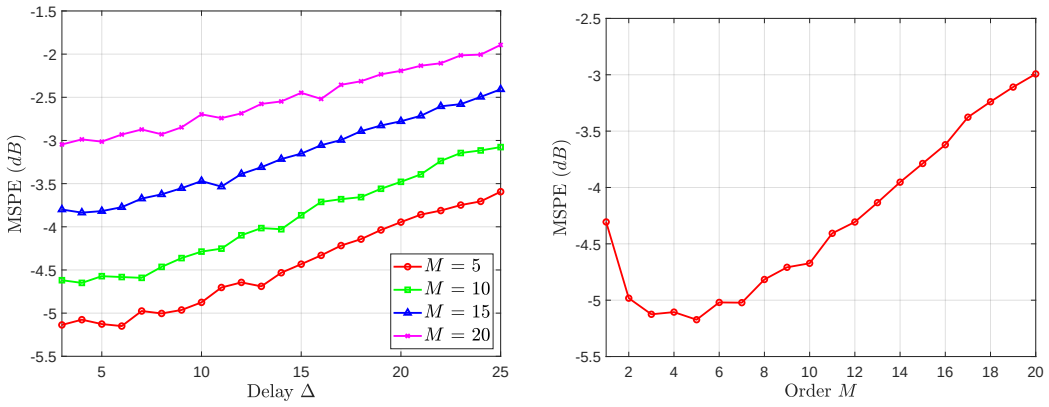


Figure 2.4: Comparison of different delay

error (MPSE) demonstrates a dependence on the delay parameter Δ: it initially levels off, and begins to increase from $\Delta = 6$. This deterioration in performance is because of an additional phase shift introduced by the ALE filter output. This causes deviations of the denoised signal $\hat{x}(n)$ from the ground-truth signal $x(n)$. Specifically, the MPSE increases as the filter order M increases, while Δ keep same. This also can be confirmed from Figure 2.5(b). This is attributed to the algorithm overfitting the noise present in the signal. This overfitting, a consequence of increasing complexity, leads to increased error and degraded performance. Therefore, $\Delta = 3$ and $M = 5$ is the optimal parameters, balancing best performance with minimal complexity.



(a) MPSE of specific filter orders against delays

(b) MPSE of different filter orders ($\Delta = 3$)

Figure 2.5: MPSE of different filter orders and delays

2.3.2 Task c: Adaptive Noise Cancellation (ANC)

In this task, Adaptive Noise Cancellation (ANC) was implemented with the parameters set as follows: step size $\mu = 0.01$, delay $\Delta = 3$, and filter order $M = 10$. The secondary noise input $\epsilon(n)$ is considered as follows:

$$\epsilon(n) = 1.27\eta(n) + 0.09 \tag{2.27}$$

where $\eta(n)$ is the coloured noise defined in Equation 2.24.

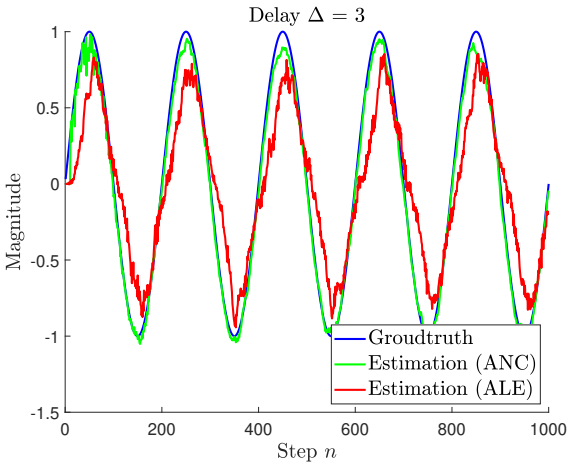


Figure 2.6: Comparison of ground-truth signal and estimated signal by ANC and ALE

Figure 2.6 shows the comparison of ground-truth signal and estimated signal by ANC and ALE. It is evident that although the ANC-estimated signal may lack smoothness and show instability at lower sample steps, it still outperforms the ALE-estimated signal. This also can be quantified by the MSPE. The MSPE for ALE is 0.3491, while for ANC it is 0.1110, indicating superior performance of ANC over ALE.

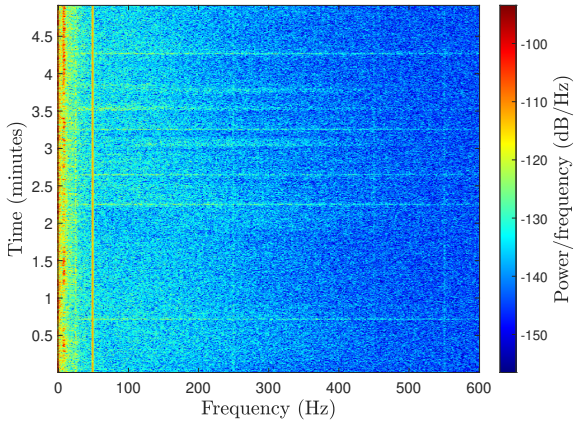


Figure 2.7: Spectrum of original EEG data (POz)

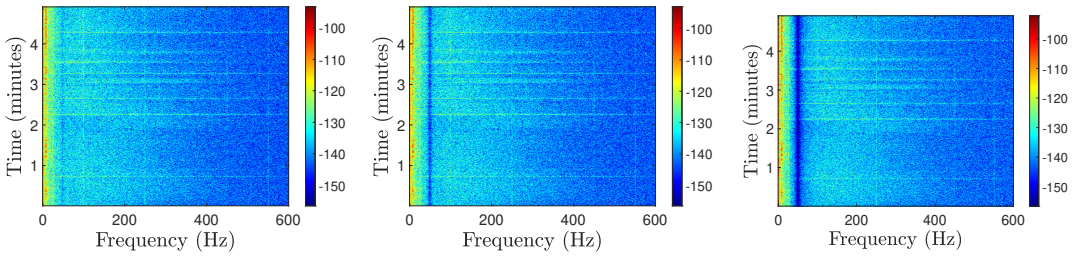
2.3.3 Task d: ANC on EEG Data

The original EEG data (POz) has been plotted using the `spectrogram` function with a window length of 1000 and an overlap of 500, as illustrated in Figure 2.7. It is evident that the bright orange vertical line at 50Hz in the Figure 2.7 signifies noise. The ANC algorithm is implemented here to eliminate this 50Hz noise component. Its parameters set as follows: step size $\mu = 0.01$ and filter order $M = 10$. The synthetic reference signal is generated as a 50 Hz sinusoid corrupted by white Gaussian noise, as follows:

$$s(n) = \sin(2\pi \cdot 50 \cdot n) + w(n) \tag{2.28}$$

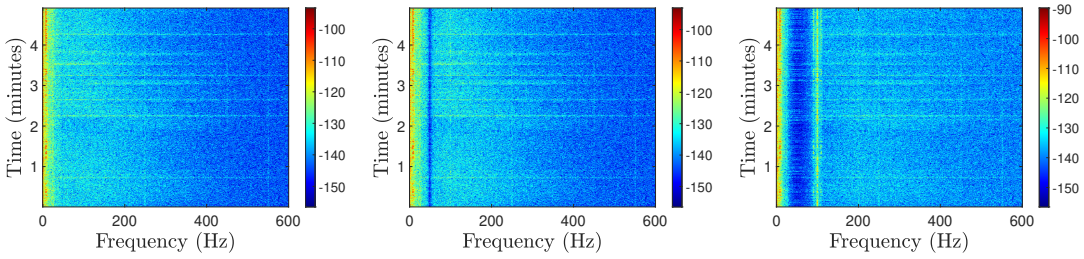
where $w(n)$ is the white Gaussian noise.

Figure 2.8 demonstrates how increasing the model order M affects the suppression of the 50Hz noise component. A model order of $M = 15$, as depicted in Figure 2.8(b), is appropriate choice for satisfactory suppression (dark vertical line at 50Hz), while lower orders proving insufficient as shown in Figure 2.8(a). However, excessively high orders lead to distortion of the spectrogram, as shown in Figure 2.8(c). Besides, increasing the step size μ yields similar outcomes regarding the effect of model order M increase, as illustrated in Figure 2.9.



(a) $\mu = 0.01, M = 5$ (b) $\mu = 0.01, M = 15$ (c) $\mu = 0.01, M = 50$

Figure 2.8: Spectrum of denoised EEG data using different M



(a) $\mu = 0.001, M = 15$ (b) $\mu = 0.01, M = 15$ (c) $\mu = 0.1, M = 15$

Figure 2.9: Spectrum of denoised EEG data using different μ

3

Widely Linear Filter and Adaptive Spectrum Estimation

3.1 Complex LMS and Widely Linear Modelling

3.1.1 Task a: Comparison of CLMS and ACLMS

In this task, we aim to compare the CLMS and ACLMS algorithms. Initially, we consider a first-order widely-linear-moving-average process, WLMA(1), driven by circular white Gaussian noise as follows:

$$y(n) = x(n) + b_1 x(n-1) + b_2 x^*(n-1), \quad x \sim \mathcal{CN}(0,1) \tag{3.1}$$

where $b_1 = 1.5 + 1j$ and $b_2 = 2.5 - 0.5j$. Both CLMS and ACLMS algorithms are implemented using step size $\mu = 0.01$ and filter order $M = 1$ for the identification of the WLAM(1) model.

The learning curves, expressed as $10\log|e(n)|^2$, for the CLMS and ACLMS algorithms are depicted as Figure 3.1. These curves are obtained by plotting the ensemble average across 100 independent realizations of the process, each involving 1,000 data samples. It is shown that the squared error for the CLMS algorithm consistently falls between 6dB and 8dB. In comparison, the squared error of the ACLMS algorithm converges to approximately -3dB at around step 200. This demonstrates that the superior performance of ACLMS over CLMS in steady-state conditions.

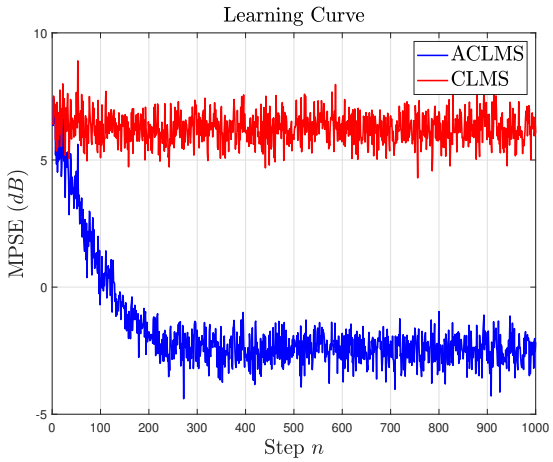


Figure 3.1: Learning curves of CLMS and ACLMS

3.1.2 Task b: CLMS and ACLMS for Bivariate Wind Data Processing

In this task, we aim to apply the CLMS and ACLMS algorithms to bivariate wind data, considering two directional components: East-West (v_{east}) and North-South (v_{north}), at three distinct wind speeds: low, medium, and high. The bivariate dataset was represented as a complex-valued wind signal, as delineated as follows:

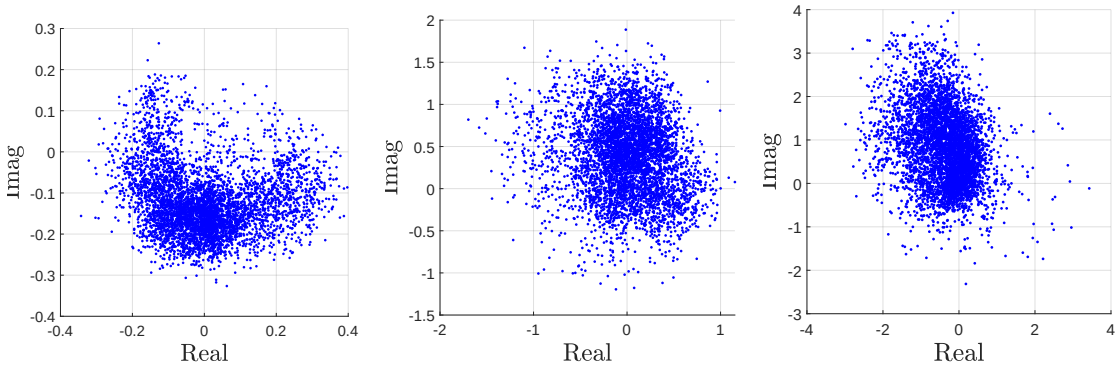
$$v[n] = v_{\text{east}}[n] + jv_{\text{north}}[n] \tag{3.2}$$

The scatter diagrams for the three wind speeds signals are presented in Figure 3.2. It is evident that these datasets is not circular due to the property of the data changing with rotation. The circularity of these signals can also be quantitatively assessed through the circularity coefficient, defined as follows:

$$|\rho| = \frac{|\mathbb{E}\{zz^*\}|}{\mathbb{E}\{zz^T\}} \tag{3.3}$$

For high-speed wind, $|\rho| = 0.6237$. For medium-speed wind, $|\rho| = 0.4543$. For low-speed wind, $|\rho| = 0.1592$. These results indicate that the circularity is highest for low-speed wind and lowest for high-speed wind, suggesting that as wind speed increases, the circularity of the signal diminishes.

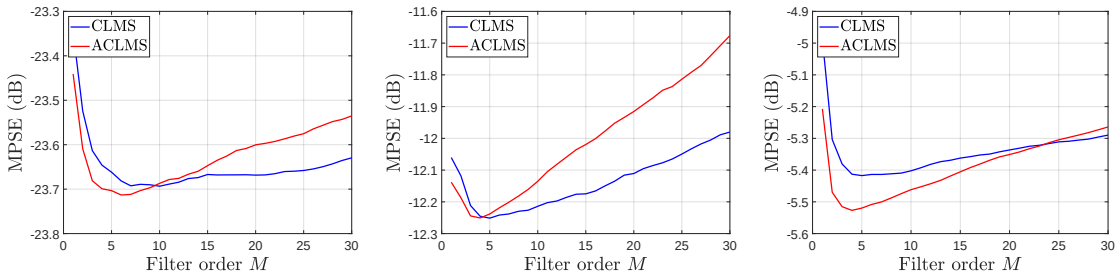
Next, the CLMS and ACLMS were employed to conduct a one-step ahead forecast of complex wind data across different regimes. The optimal learning rates, $\mu_{\text{high}} = 0.001$, $\mu_{\text{medium}} = 0.01$, and $\mu_{\text{low}} = 0.1$, were empirically chose to minimize prediction errors while ensuring filter stability. Figure 3.3 illustrates the trend change of MPSE with order M. It is evident that for high-speed wind data, the ACLMS algorithm significantly outperforms CLMS, indicating superior performance of



(a) Low Wind (b) Medium wind (c) High wind

Figure 3.2: Scatter diagrams of bivariate wind data

ACLMS for data with low circularity. In contrast, for medium-speed and low-speed wind data characterized by relatively high circularity, the performance gap of both CLMS and ACLMS algorithms becomes less marked. Meanwhile, it is also notable that across all wind conditions, as the filter order rises, the issue of overfitting becomes significantly more pronounced in ACLMS.

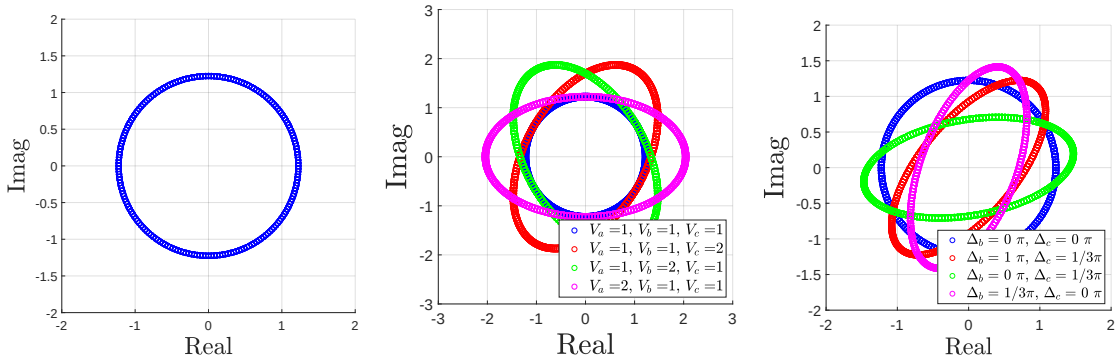


(a) Low Wind (b) Medium wind (c) High wind

Figure 3.3: MSPE against filter order for different wind regimes

3.1.3 Task c: Circularity Diagrams of Complex Voltages

In this task, balanced and unbalanced complex voltages systems are implemented with system frequency $f_o = 50$Hz, sampling frequency $f_s = 1000$Hz and 1,000 data samples. The system is balanced complex voltages system when $V_a(n) = V_b(n) = V_c(n)$ and $\Delta_b = \Delta_c = 0$. The circularity diagrams of balanced systems is fully circular, characterized by rotational invariance as shown in Figure 3.4(a). When the voltage and phase Δ vary, the system shifts towards unbalanced system like an ellipse rather than a perfect circle depicted in Figure 3.4(b) and 3.4(c). Consequently, the circularity diagram can serve as a tool for fault detection within the system by quantifying its circularity and assessing the extent of deviation.



(a) Balanced System (b) Unbalance System: Voltages (c) Unbalanced system: Phase

Figure 3.4: Circularity Diagrams of Complex Balanced and Unbalanced Systems

3.1.4 Task d: Derivations of System Frequency Estimation

1) Firstly, consider the strictly linear autoregressive model, defined as:

$$v(n+1) = h^*(n)v(n) \tag{3.4}$$

For balanced system, the complex voltage is given as

$$v(n) = \sqrt{\frac{3}{2}} V e^{j\left(2\pi \frac{f_o}{f_s} n + \phi\right)} \tag{3.5}$$

By substituting equation (3.4) into equation (3.5), we obtain:

$$h^*(n) = |h(n)| e^{j \arctan\left(\frac{\Im\{h(n)\}}{\Re\{h(n)\}}\right)} = \frac{\sqrt{\frac{3}{2}} V e^{j\left(2\pi \frac{f_o}{f_s}(n+1) + \phi\right)}}{\sqrt{\frac{3}{2}} V e^{j\left(2\pi \frac{f_o}{f_s} n + \phi\right)}} = e^{j 2\pi \frac{f_o}{f_s}} \tag{3.6}$$

By taking the argument of both sides, we get:

$$\arctan\left(\frac{\Im\{h(n)\}}{\Re\{h(n)\}}\right) = 2\pi \frac{f_o}{f_s} \tag{3.7}$$

From which we can finally derive the frequency of the balanced complex α-β voltage as:

$$f_o(n) = \frac{f_s}{2\pi} \arctan\left(\frac{\Im\{h(n)\}}{\Re\{h(n)\}}\right) \tag{3.8}$$

2) Similarly, in unbalanced system, the complex voltage is given as

$$v(n) = A(n) e^{j\left(2\pi \frac{f_o}{f_s} n + \phi\right)} + B(n) e^{-j\left(2\pi \frac{f_o}{f_s} n + \phi\right)} \tag{3.9}$$

Here, consider widely linear autoregressive model as follows:

$$v(n+1) = h^*(n)v(n) + g^*(n)v^*(n) \tag{3.10}$$

By substituting equation (3.8) into equation (3.10), we obtain:

$$\begin{aligned} A(n+1)e^{j\left(2\pi\frac{f_0}{f_s}n+\phi\right)}e^{j\left(2\pi\frac{f_0}{f_s}\right)} + B(n+1)e^{-j\left(2\pi\frac{f_0}{f_s}n+\phi\right)}e^{-j\left(2\pi\frac{f_0}{f_s}\right)} = \\ h^*(n)A(n)e^{j\left(2\pi\frac{f_0}{f_s}n+\phi\right)} + h^*(n)B(n)e^{-j\left(2\pi\frac{f_0}{f_s}n+\phi\right)} + \\ g^*(n)A^*(n)e^{-j\left(2\pi\frac{f_0}{f_s}n+\phi\right)} + g^*(n)B^*(n)e^{j\left(2\pi\frac{f_0}{f_s}n+\phi\right)} \end{aligned} \tag{3.11}$$

Comparing the coefficients of $e^{j\left(2\pi\frac{f_0}{f_s}n+\phi\right)}$ and $e^{-j\left(2\pi\frac{f_0}{f_s}n+\phi\right)}$ on both sides, then we have a particular solution as follows:

$$\begin{cases} A(n+1)e^{j\left(2\pi\frac{f_0}{f_s}\right)} = h^*(n)A(n) + g^*(n)B^*(n) \\ B(n+1)e^{-j\left(2\pi\frac{f_0}{f_s}\right)} = h^*(n)B(n) + g^*(n)A^*(n) \end{cases} \tag{3.12}$$

Assuming the three-phase voltage magnitudes are time-invariant, we can approximate $A(n+1) \approx A(n)$ and $B(n+1) \approx B(n)$. Consequently, Equation (3.12) can be rewrote as:

$$\begin{cases} e^{j\left(2\pi\frac{f_0}{f_s}\right)} = \dfrac{h^*(n)A(n) + g^*(n)B^*(n)}{A(n+1)} \approx \dfrac{h^*(n) + g^*(n)\frac{B^*(n)}{A(n)}}{\frac{A(n+1)}{A(n)}} \\ e^{-j\left(2\pi\frac{f_0}{f_s}\right)} = \dfrac{h^*(n)B(n) + g^*(n)A^*(n)}{B(n+1)} \approx \dfrac{h^*(n) + g^*(n)\frac{A^*(n)}{B(n)}}{\frac{B(n+1)}{B(n)}} \end{cases} \tag{3.13}$$

Let $S(n) = \frac{B^*(n)}{A(n)}$, Equation (3.13) can be rearranged as:

$$g^*(n)S(n)^2 + (h^*(n) - h(n))S(n) - g(n) = 0 \tag{3.14}$$

By solving this quadratic equation for $S(n)$, we obtain

$$\begin{aligned} S(n) &= \frac{(h(n) - h^*(n)) \pm \sqrt{(h^*(n) - h(n))^2 + 4g^*(n)g(n)}}{2g^*(n)} \\ &= \frac{2j\Im\{h(n)\} \pm \sqrt{-4(\Im\{h(n)\})^2 + 4|g(n)|^2}}{2g^*(n)} \\ &= \frac{j\Im\{h(n)\} \pm \sqrt{(\Im\{h(n)\})^2 - |g(n)|^2}}{g^*(n)} \end{aligned} \tag{3.15}$$

Substituting Equation(3.15) back into Equation (3.13), we obtain:

$$e^{j\left(2\pi\frac{f_0}{f_s}\right)} = \Re\{h(n)\} \pm j\sqrt{\Im^2\{h(n)\} - |g(n)|^2} \tag{3.16}$$

$$= |C|e^{j\arctan\left(\frac{\pm\sqrt{\Im^2\{h(n)\}-|g(n)|^2}}{\Re\{h(n)\}}\right)} \tag{3.17}$$

Since the phase sign only changes the rotation direction and $|e^{j\left(2\pi\frac{f_0}{f_s}\right)}| = 1$, we can finally derive the frequency of the unbalanced complex α-β voltage in Equation (3.18) by setting the positive frequency and $|C| = 1$.

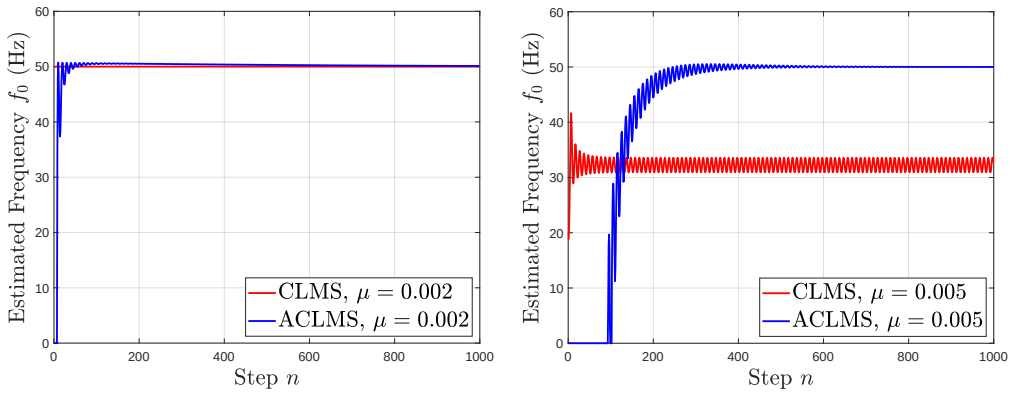
$$f_0 = \frac{f_s}{2\pi}\arctan\left(\frac{\sqrt{\Im^2\{h(n)\} - |g(n)|^2}}{\Re\{h(n)\}}\right) \tag{3.18}$$

3.1.5 Task e: Frequency Estimation Using CLMS and ACLMS

In this task, the CLMS and ACLMS algorithms are utilized for estimating voltages in both balanced and unbalanced systems. The parameters for these systems are presented in Table 3.1.

Table 3.1: Parameters of balanced and unbalanced systems

	f_0	f_s	V_a	V_b	V_c	ϕ	Δ_b	Δ_c
Balanced	50Hz	1000Hz	1	1	1	0	0	0
Unbalanced	50Hz	1000Hz	1	2	3	0	$\frac{1}{3}\pi$	$\frac{1}{3}\pi$



(a) Balanced System (b) Unbalanced System

Figure 3.5: Comparison of CLMS and ACLMS for frequency estimation

In the balanced systems, the CLMS algorithm initially provides an accurate estimate of f_0. However, it's the ACLMS algorithm that yields correct f_0 estimates after several iterations. Conversely, within unbalanced systems, CLMS falls in accurately estimating f_0, whereas ACLMS achieves accurate f_0 estimates over numerous steps. Moreover, ACLMS only addresses oscilla-

tion errors upon convergence in unbalanced systems. This arises because CLMS perform well for circular signals, whereas unbalanced system signals are non-circular.

3.2 Adaptive AR Model Based Time-Frequency Estimation

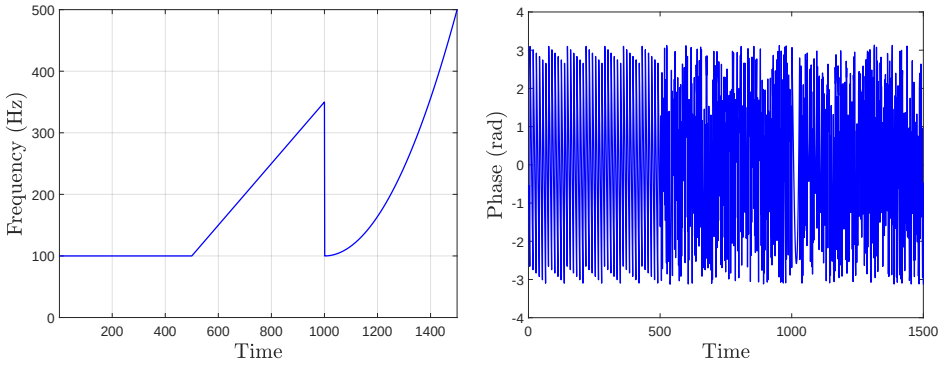
In this section, we generated a frequency modulated (FM) signal given as:

$$y(n) = e^{j\left(\frac{2\pi}{f_s}\phi(n)\right)} + \eta(n) \tag{3.19}$$

where $\eta(n) \sim \mathcal{CN}(0, 0.05)$ and $\phi(n) = \int f(n)dn$. $f(n)$ is a frequency signal which is defined as:

$$f(n) = \begin{cases} 100, & 1 \leq n \leq 500 \\ 100 + \frac{n-500}{2}, & 501 \leq n \leq 1000 \\ 100 + \left(\frac{n-1000}{25}\right)^2, & 1001 \leq n \leq 1500 \end{cases} \tag{3.20}$$

The sample frequency is set to $f_s = 2000$Hz. The Frequency and Phase of FM Signal is presented in the Figure 3.6.



(a) Frequency (b) Phase

Figure 3.6: Frequency and Phase of FM Signal

The spectrum of $y(n)$ is firstly estimated using the conventional AR(1) model. To calculate the AR(1) coefficient, the MATLAB function `aryule` is utilized, while spectrum density is obtained by `freqz` as shown in Figure 3.7.This approach fails to capture frequency changes due to its assumption of signal stationarity, whereas FM signals are inherently non-stationary.

In order to capture the frequency changes, The CLMS algorithm was utilized to estimate the AR coefficients of the FM signal, with its frequency spectrum obtained using `freqz` at each time

instance. Displayed in Figure 30, with a step size of $\mu = 0.1$, the yellow region of the spectrum highlights the frequency component of the FM signal, closely matching the original spectrum shown in Figure 3.6(a). Unlike the stationary AR spectrum, the CLMS-based Time-Frequency Spectrum effectively estimates the frequency of non-stationary signals, demonstrating its superiority over traditional AR methods in handling non-stationarity.

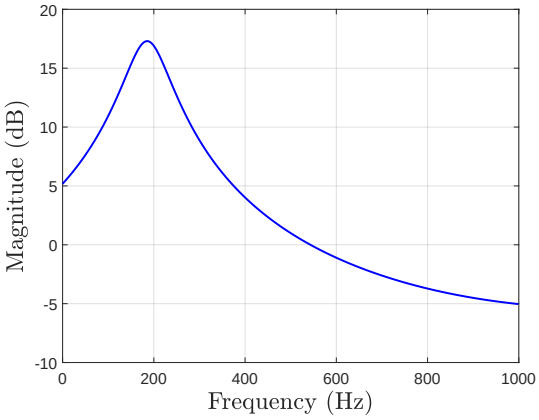


Figure 3.7: Power Spectrum estimated by AR(1) model

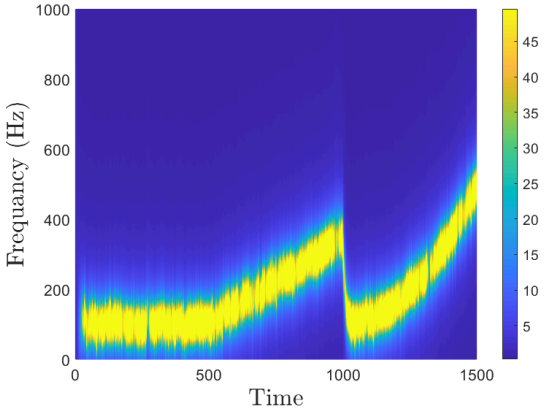


Figure 3.8: Time-frequency spectrum estimated by adaptive AR(1) model

3.3 A Real Time Spectrum Analyser Using Least Mean Square

3.3.1 Task a & b: LS Solution and DFT

The signal $y(n)$ can be estimated by a linear combination of N harmonically related sinusoids as $\hat{\mathbf{y}} = \mathbf{F}\mathbf{w}$, where the matrix $\mathbf{F}$ is defined as follows:

$$\mathbf{F} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{j\frac{2\pi}{N}(1)(1)} & \cdots & e^{j\frac{2\pi}{N}(1)(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e^{j\frac{2\pi}{N}(N-1)(1)} & \cdots & e^{j\frac{2\pi}{N}(N-1)(N-1)} \end{bmatrix} \tag{3.21}$$

The least squares problem is solved by minimising the cost function as follows:

$$\begin{aligned} \min_{\mathbf{w}} \|\mathbf{y} - \hat{\mathbf{y}}\|^2 &= \min_{\mathbf{w}} (\mathbf{y} - \hat{\mathbf{y}})^H (\mathbf{y} - \hat{\mathbf{y}}) \\ &= \min_{\mathbf{w}} (\mathbf{y} - \mathbf{F}\mathbf{w})^H (\mathbf{y} - \mathbf{F}\mathbf{w}) \\ &= \min_{\mathbf{w}} (\mathbf{y}^H\mathbf{y} - \mathbf{w}^H\mathbf{F}^H\mathbf{y} - \mathbf{y}^H\mathbf{F}\mathbf{w} + \mathbf{w}^H\mathbf{F}^H\mathbf{F}\mathbf{w}) \end{aligned} \tag{3.22}$$

Taking the gradient with respect to $\mathbf{w}$ yields:

$$\begin{aligned}\frac{\partial\|\mathbf{y}-\hat{\mathbf{y}}\|^2}{\partial\mathbf{w}} &= (\mathbf{F}^H\mathbf{F}+\mathbf{F}\mathbf{F}^H)\mathbf{w}-2\mathbf{F}^H\mathbf{y}\\ &= 2\mathbf{F}^H\mathbf{F}\mathbf{w}-2\mathbf{y}\mathbf{F}\end{aligned} \tag{3.23}$$

By setting gradient to zero, we obtain optimal weights in the least-squares sense as follows:

$$\mathbf{w}_{\text{opt}} = (\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H\mathbf{y} \tag{3.24}$$

The optimal weights derived in Equation ((3.25)) are closely related to the discrete Fourier transform (DFT). The DFT of $\mathbf{y}$ by $\mathbf{x} = [x(0), ..., x(N-1)]^T$ is defined as:

$$y(n) = \frac{1}{N}\sum_{k=0}^{N-1} x(k)e^{j\frac{2\pi}{N}nk}, \quad n = 0, ..., N-1 \tag{3.25}$$

To represent it in matrix form, Equation ((3.25)) can be rewritten as:

$$\mathbf{y} = \frac{1}{N}\mathbf{F}\mathbf{x} \tag{3.26}$$

By substituting Equation ((3.25)) into Equation ((3.23)), we get:

$$\mathbf{w} = \frac{1}{N}(\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H\mathbf{F}\mathbf{x} = \frac{1}{N}\mathbf{x} \tag{3.27}$$

Consequently, the least squares solution $\mathbf{w}$ equates to the DFT of $\mathbf{y}$ scaled by $\frac{1}{N}$.

In the least squares interpretation of the DFT, the matrix $\mathbf{F}$ lies in its composition from orthogonal columns that not only span an orthogonal subspace but also form a basis for it. The orthogonality is derived from the inner product relation as follows:

$$\left\langle e^{j\frac{2\pi k}{N}n}, e^{j\frac{2\pi m}{N}n}\right\rangle = \sum_{n=0}^{N-1} e^{j\frac{2\pi(k-m)}{N}n} = \begin{cases}1 & \text{if } n=m,\\ 0 & \text{otherwise.}\end{cases} \tag{3.28}$$

Given $\hat{\mathbf{y}} = \mathbf{F}\mathbf{w}$, the estimated $\hat{\mathbf{y}}$ exists within the subspace spanned by $\mathbf{F}$'s columns, whereas $\mathbf{y}$ exists in another completely different subspace. The least squares solution projects $\mathbf{y}$ onto $\mathbf{F}$'s subspace, seeking the nearest approximation of $\mathbf{y}$ that minimizes the sum of squared errors. This is achieved through the projection matrix $\mathbf{P} = \mathbf{F}(\mathbf{F^H}\mathbf{F})^{-1}\mathbf{F^H}$, applied to $\mathbf{y}$ to yield $\hat{\mathbf{y}}$. The estimate's accuracy can be improved with the column dimension N of $\mathbf{F}$, approaching ideal accuracy as $N \to \infty$.

3.3.2 Task c & d : DFT-CLMS Algorithm

In the task c, the DFT-CLMS algorithm is implemented to compute the weight $w(n)$ at each time step to get the time-frequency spectrum of FM-signal defined in Equation (3.19). Parameters include the signal length of 1500, the sampling frequency $f_s = 2000$Hz, the step size $\mu = 1$, and the initial weight $w(0) = 0$.

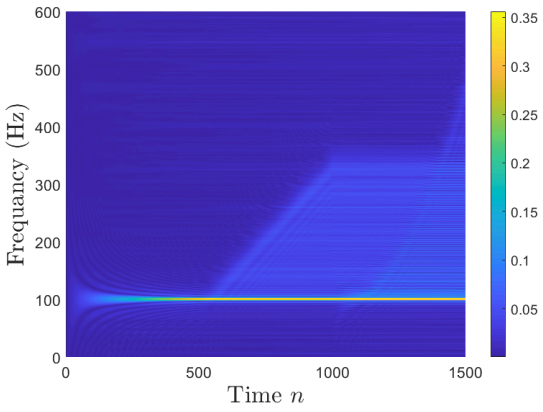


Figure 3.9: Time-Freq Spectrum of FM Signal

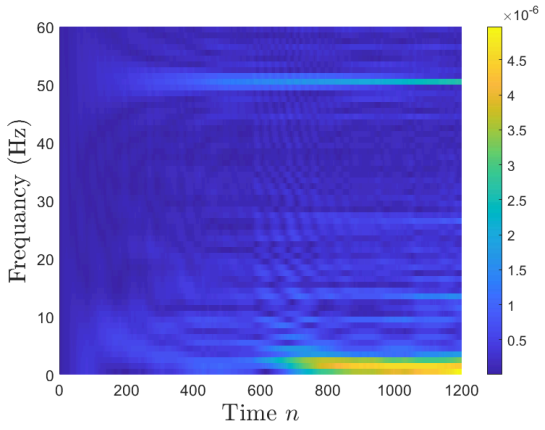


Figure 3.10: Time-Freq Spectrum of EEG data

Figure3.9 presents the DFT-CLMS based time-frequency spectrum of the FM signal. In contrast to the adaptive AR spectrum depicted in Figure 3.8, the DFT-CLMS spectrum fails to accurately capture the changes in frequency over time. The primary issue lies in the frequency estimation process: once a frequency is identified, it remains across the spectrum. This is due to the fact that the weight $\mathbf{w}(k)$ in DFT-CLMS algorithm is the DFT of the sequence $\mathbf{y}_k = \mathbf{y}(0 : k) = [y(0), ..., y(k)]^T$. Consequently, instead of lines delineating specific frequency components as seen in the accurate spectrum, the DFT-CLMS spectrum exhibits as a highlight region.

In the task d, The DFT-CLMS algorithm is also used to get time-frequency spectrum of EEG data (POz) shown in Figure 3.10. To reduce computational complexity, the EEG data is truncated to a length of 1200. Parameters is set as the sampling frequency $f_s = 1200$Hz and the step size $\mu = 1$. In Figure 3.10, the SSVEP signals at 13Hz and 26Hz have visible response after around 600 steps, with mains interference at 50Hz also clearly evident. However, it is noteworthy that the third harmonic is not visible as previous analysis. Although, this estimation approach yields spectral estimates with less noise, it introduces higher computational complexity and requires a considerable number of observations to accurately learn the optimal parameters.

4

From LMS to Deep Learning

4.1 Task 1: One-step Ahead Prediction for Time-series

In this task, one-step ahead prediction on the time series data is implemented using the LMS algorithm with step size $\mu = 1 \times 10^{-5}$ and order $M = 4$. The iteration of estimated signal compared to grouth-truth signal is presented in Figure 4.1. It is evident that the difference between the estimated signal and the ground-truth signal decreases over iteration. The Mean Square Error (MSE) of the estimated signal = 40.094 and prediction gain $R_p = 5.198\ dB$.

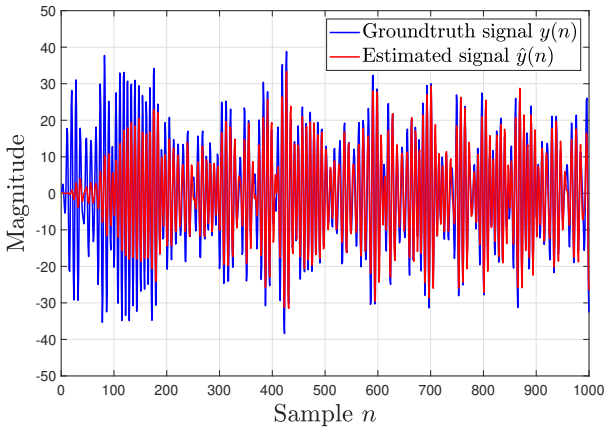


Figure 4.1: One-step ahead prediction using LMS

4.2 Task 2: Dynamical Perceptron

In this task, a non-linearity activation function **tanh** is used in the LMS algorithm ($\mu = 1 \times 10^{-5}$ and $M = 4$) to achieve a dynamic perception of the estimated signal, as illustrated in Figure 4.3. However, this results in a completely distorted estimated signal, indicating that the chosen activation function is unsuitable. This also can be quantified by the MSE of the estimated signal = 196.73 and prediction gain R_p = -23.19 dB.

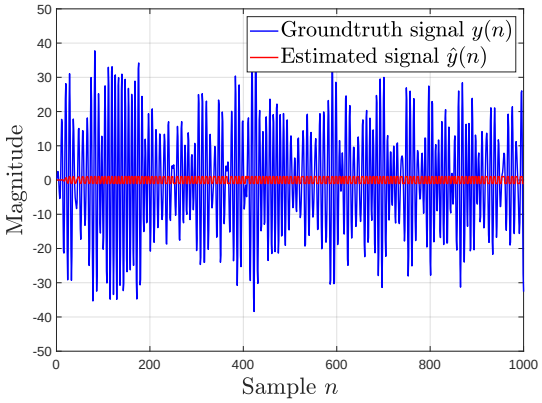


Figure 4.2: Dynamical Perception using tanh

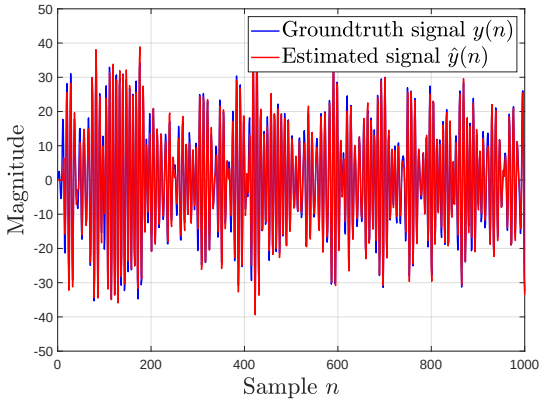


Figure 4.3: Dynamical Perception using scaled tanh

4.3 Task 3: Dynamical Perceptron With Scaled Activation Function

In this task, the activation layer is modified by employing a scaled activation function which alter the function's amplitude. Specifically, the activation function is transformed to **a** $\times$ **tanh**. Analysis of the original signal's amplitude reveals maximum value approximately at 38, indicating that α should exceed 38. Therefore **a** = 80 is selected and $\mu = 1 \times 10^{-5}$ and $M = 4$ are kept as before. The One-step ahead prediction of zero-mean signal with scaled activation function is depicted in Figure 35.This figure shows an rapid convergence to the original signal rather than a gradual approximation over several steps in Figure 4.1. The MSE is recorded at 4.784, with a prediction gain of $R_p = 16.494$ dB.

4.4 Task 4: Dynamical Perceptron with Bias

In this task, the model adds a bias into the input, transforming it into $[1, \mathbf{x}]^T$. Consequently, the model's output is modified to $\phi(\mathbf{w}^T\mathbf{x} + b)$, where $\phi(\cdot)$ is the activation function. $\mu = 1 \times$

10^{-5} and $M = 4$are kept as before. Figure 4.4 also illustrates the non-linear, one-step-ahead prediction, where the estimated signal begins to approximate the original signal more closely from the beginning, rather than after several iterations in Figure 4.1. The MSE is 4.792, and the prediction gain is $R_p = 16.486\ dB$.

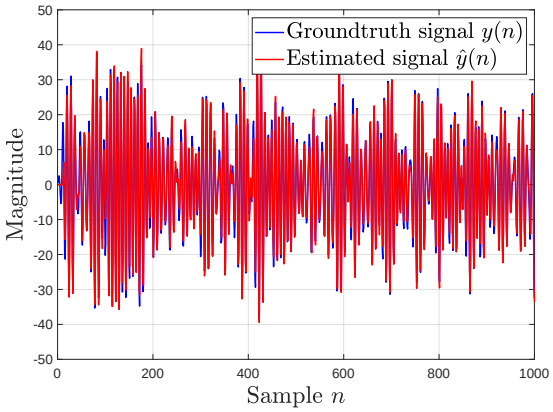


Figure 4.4: Dynamical Perception with Bias

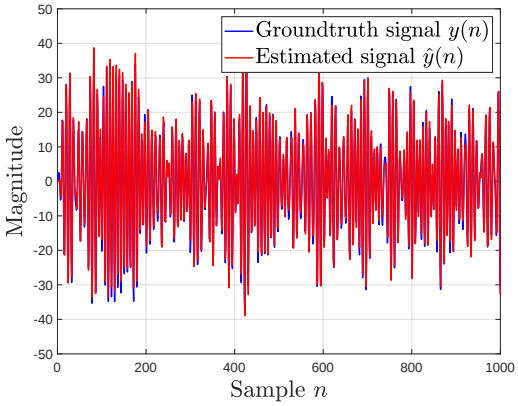


Figure 4.5: Dynamical Perception with Bias and Pretrained Weights

4.5 Task 5: Dynamical Perception with Bias and Pretrained Weights

Considering single weight updates at each time-step, convergence to a precise prediction requires numerous samples. Adding scaling factors and bias may increase error. To avoid this, parameters can be initially adjusted by over-fitting a limited dataset, improving starting conditions. Specifically, for the dynamical perception with bias defined in Task 4, pre-training was applied to the first 20 samples across 100 epochs, starting with $w(0) = 0$. $\mu = 1 \times 10^{-5}$ and $M = 4$ are kept as before. Figure 4.5 shows the one-step ahead prediction outcomes. This pre-training enhanced prediction accuracy significantly, achieving an MSE of 2.402 and an R_p of 19.988 dB. Thus it enables almost perfect predictions from the signal's beginning and achieve the best performance compared to all previous model in Task 1-4.

4.6 Task 6: Backpropagation in Deep Network

Deep networks utilize the backpropagation algorithm to iteratively adjust the weights of the network based on the error between the predicted output and the actual output. This process relies

on the principle of gradient descent to minimize the error by propagating it backward through the network layers. Crucial to backpropagation's success is the use of differentiable activation functions, which ensure the gradient of the error can be computed at each layer. The backpropagation process comprises the following steps:

1. **Feedforward**: For each layer, calculate the output of each neuron using the formula $y = \phi(\mathbf{w}^T\mathbf{x} + b)$, where ϕ is the activation function, $\mathbf{w}$ is the weight matrix, $\mathbf{x}$ is the input vector, and b is the bias.

2. **Compute Prediction Error**: Evaluate the error (E) between the predicted output (y) and the actual output ($\hat{y}$), typically using a loss function like Mean Squared Error (MSE) or Cross-Entropy Loss.

3. **Backpropagate Error**: For each neuron, calculate the gradient of the loss with respect to each weight ($\frac{\partial E}{\partial w_{ij}}$) by applying the chain rule. This involves computing the derivative of the error with respect to the neuron's output ($\frac{\partial E}{\partial y_i}$) and the derivative of the neuron's output with respect to its weights ($\frac{\partial y_i}{\partial w_{ij}}$).

4. **Update Weights**: Adjust the weights in the direction that minimally reduces the error, using the formula $w_{ij}(n+1) = w_{ij}(n) - \eta\frac{\partial E}{\partial w_{ij}}$. Here, η is the learning rate, a small positive scalar determining the size of the step taken in the gradient descent.

4.7 Task 7: Comparison of DNNs and Simple Dynamical Perceptron

The deep network is set with four hidden layers, trained over 20,000 epochs, using a learning rate of 0.01, and a noise power set to 0.05. The input and ideal signals are illustrated in Figure 4.6. The comparison between the predicted and ideal signals is depicted in Figure 4.7. Performance across single neurons is poor. Although the deep network can capture the general trend of the ideal signals, its accuracy is still poor.

Figure 4.9 illustrates the training and testing loss for various models: a single neuron with linear activation, a single neuron with tanh activation, and a deep network. The initial testing loss for the single neuron models is high, but they converge rapidly to approximately 0.1. In contrast, the deep network starts with a lower initial testing loss but converges more slowly due to its complex

structure. However, the deep network achieves a final testing loss around 0.08, which is lower than that of the single neuron models, indicating superior performance.

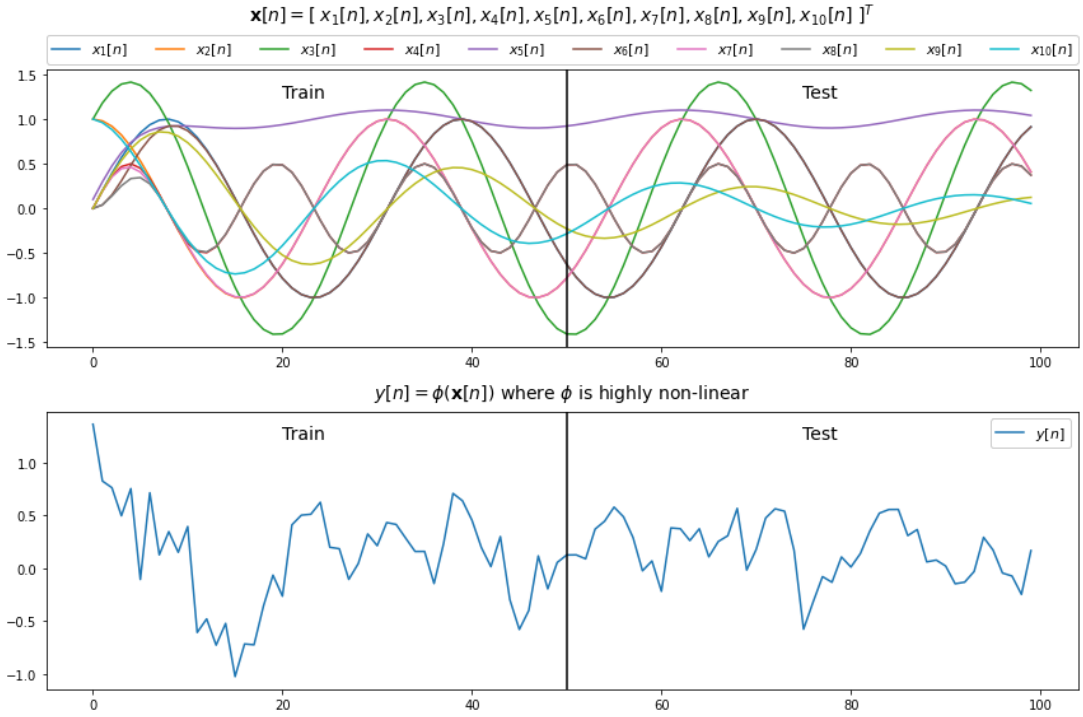


Figure 4.6: Input signal (top) and target signal (bottom)

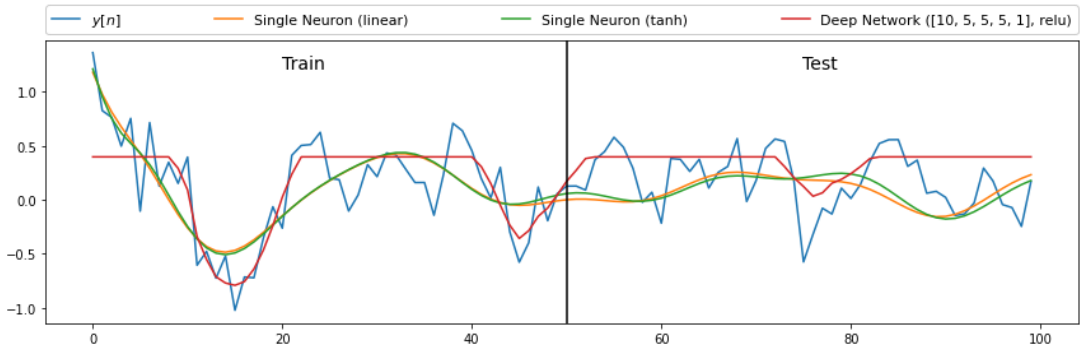


Figure 4.7: Outputs of deep network and single neuron

4.8 Task 8: Effect of Noise Power on DNNs

To evaluate the deep network's robustness, simulations with varying noise levels were conducted. Figures 4.9 ,4.8 and 4.10 display the network's loss at different noise powers ($\sigma^2 = 0.05$, 0.0 and 1.0, respectively). At $\sigma^2 = 0.0$, the network's test loss converges to a relatively low value. However, as noise power increases, the single neuron model maintains its convergence capability. In contrast,

the deep network fails to converge, reflected in both training and testing loss. Notably, the deep network's test loss worsens as epochs increase, surpassing the poor performance of the single neuron model. This trend suggests significant overfitting within the deep network.

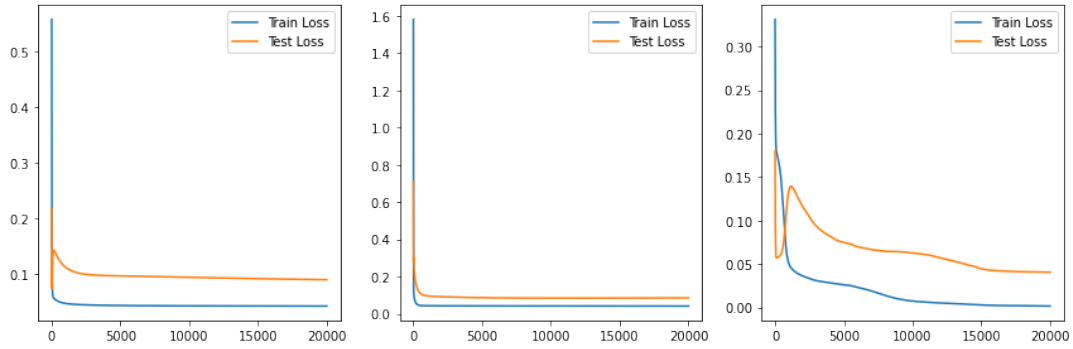


Figure 4.8: Loss of Single Neuron (Linear), Single Neuron (tanh) and Deep Network ($\sigma^2 = 0$)

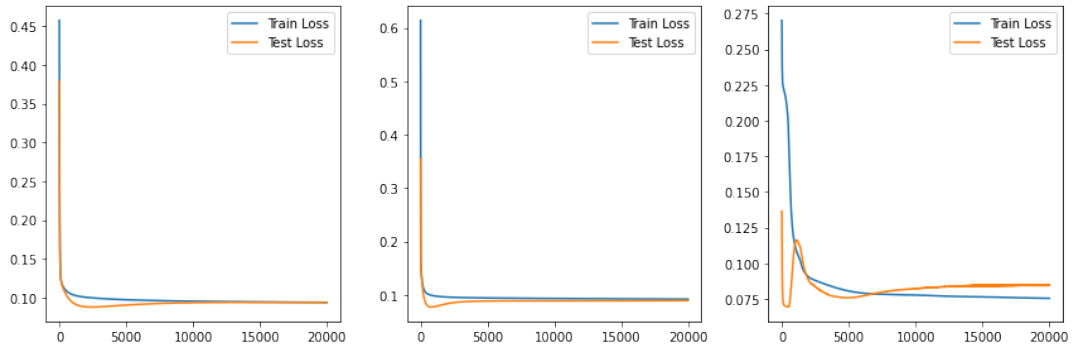


Figure 4.9: Loss of Single Neuron (Linear), Single Neuron (tanh) and Deep Network ($\sigma^2 = 0.05$)

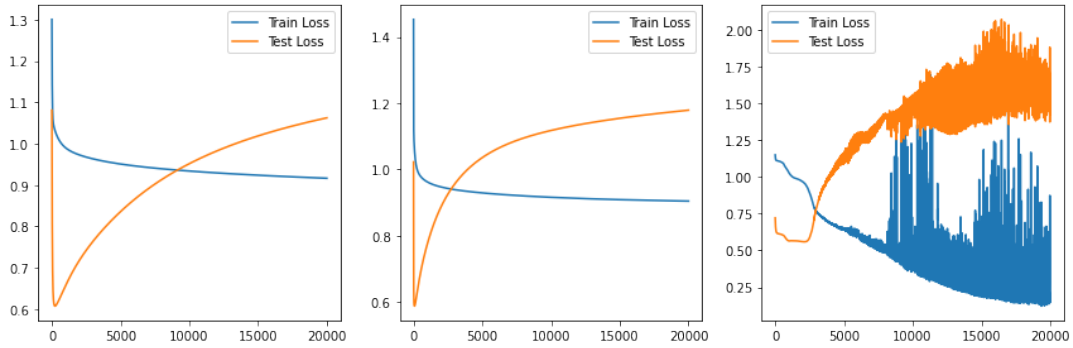


Figure 4.10: Loss of Single Neuron (Linear), Single Neuron (tanh) and Deep Network ($\sigma^2 = 1$)