Imperial College London

Department of Bioengineering

# Adaptive Signal Processing & Machine Intelligence Coursework

*Author:* João Pereira

*CID:* 01381740

*Professor:* Professor Mandic

April 10, 2021

# Contents

# 1 Classical and Modern Spectrum Estimation

## 1.1 Properties of Power Spectral Density (PSD)

For this section, we were tasked with showing that the definition of PSD is equivalent in (1.1) and (1.2) under the mild assumption that the covariance sequence decays rapidly, equivalent to that shown in (1.3):

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

$$P(\omega) = \lim_{x\to\infty} E\{\frac{1}{N} \mid \sum_{n=0}^{N-1} x(n)e^{-j\omega n} \mid^2\} \tag{1.2}$$

$$\lim_{x\to\infty} \frac{1}{N} \sum_{k=-(N-1)}^{N-1} |k||r(k)| = 0 \tag{1.3}$$

We can show this analytically by expanding the definition in (1.2):

$$P(\omega) = \lim_{N\to\infty} E\{\frac{1}{N}(\sum_{n=0}^{N-1} x(n)e^{-j\omega n})(\sum_{n=0}^{N-1} x(n)e^{-j\omega n})^*\} \tag{1.4}$$

$$= \lim_{N\to\infty} E\{\frac{1}{N}(\sum_{n=0}^{N-1} x(n)e^{-j\omega n})(\sum_{n=0}^{N-1} x^*(n)e^{j\omega n})\} \tag{1.5}$$

$$= \lim_{N\to\infty} E\{\frac{1}{N}(\sum_{n=0}^{N-1}\sum_{n=0}^{N-1} x(n)x(m)e^{-j\omega(n-m)})\} \tag{1.6}$$

$$= \lim_{N\to\infty} \frac{1}{N}(\sum_{n=0}^{N-1}\sum_{n=0}^{N-1} E\{x(n)x(m)\}e^{-j\omega(n-m)}) \tag{1.7}$$

$$= \lim_{N\to\infty} \frac{1}{N}(\sum_{n=0}^{N-1}\sum_{n=0}^{N-1} r(n-m)e^{-j\omega(n-m)}) \tag{1.8}$$

Given that our expression is a function of $n - m$, we can substitute $\tau = n-m$ as a dummy variable, allowing us to represent both summations as a single sum. However, we must account for the fact that multiple combinations of $n$ and $m$ can give rise to the same value of $\tau$. We know the values $\tau$ can take are in the range between $0-(N-1) = -(N-1)$ and $(N-1)-0 = N-1$. Additionally, by looking at table 1 containing the number of $(n, m)$ pairs that result in the same $\tau$, we see that the number of $(n, m)$ pairs is equal to $N - |\tau|$. This results in our expression being re-written as follows:

$$\lim_{N\to\infty} \frac{1}{N} \sum_{\tau=-(N-1)}^{N-1} (N - |\tau|)r(\tau)e^{-j\omega\tau} \tag{1.9}$$

$$= \lim_{N\to\infty} \sum_{\tau=-(N-1)}^{N-1} r(\tau)e^{-j\omega\tau} + \lim_{N\to\infty} \frac{1}{N} \sum_{\tau=-(N-1)}^{N-1} |\tau|r(\tau)e^{-j\omega\tau} \tag{1.10}$$

| $\tau$ | (n, m) pairs | (n, m) pairs |
|---|---|---|
| N-1 | 1 | (N-1, 0) |
| N-2 | 2 | (N-1, 1), (N-2, 0) |
| ⋮ | ⋮ | ⋮ |
| 1 | N - 1 | (N-1, N-2), (N-2, N-3), ... |
| 0 | N | ⋮ |
| -1 | N-1 | ⋮ |
| ⋮ | ⋮ | ⋮ |
| -(N-1) | 1 | (0, N-1) |

**Table 1:** Number of $(n, m)$ pairs for a given $\tau$

Let us now only consider the second term of the expression. Given that the result will be in $\mathcal{C}$, we can consider its magnitude and phase components. Given the nonnegativity of the absolute value and the triangle inequality, we can define the range of magnitude as follows:

$$0 \leq| \lim_{N\to\infty} \frac{1}{N} \sum_{\tau=-(N-1)}^{N-1} |\tau| r(\tau) e^{-j\omega\tau} | \leq \lim_{N\to\infty} \frac{1}{N} \sum_{\tau=-(N-1)}^{N-1} |\tau||r(\tau)| \tag{1.11}$$

Where, given the assumption in (1.3):

$$0 \leq| \lim_{N\to\infty} \frac{1}{N} \sum_{\tau=-(N-1)}^{N-1} |\tau| r(\tau) e^{-j\omega\tau} | \leq 0 \tag{1.12}$$

Therefore the magnitude of second term of our expression is 0, meaning the term itself is 0:

$$\lim_{N\to\infty} \sum_{\tau=-(N-1)}^{N-1} r(\tau) e^{-j\omega\tau} = \sum_{\tau=-\infty}^{\infty} r(\tau) e^{-j\omega\tau} \tag{1.13}$$

This equality can also be shown with simulations, as shown in Figure 1.1, which uses an example of a fast-decaying covariance sequence and identical PSD estimates (top) and an example with a slow-decaying covariance sequence and different PSD estimates (bottom).

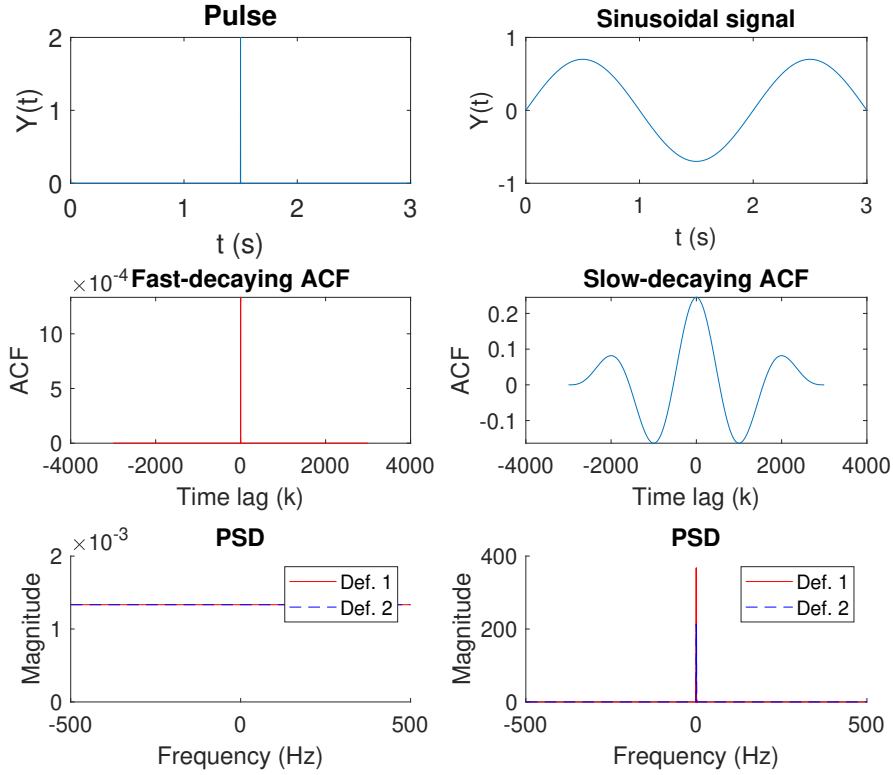


**Figure 1.1:** Signals, their ACFs and PSD estimates ($f_s = 1000Hz$), where Def. 1 and Def. 2 correspond to estimates using (1.1) and (1.2) respectively

## 1.2 Periodogram-based Methods Applied to Real–World Data

**(a)** A periodogram-based spectral estimation technique was applied to the sunspot time series (after preprocessing) . The results are shown in Figure 1.2. We can see by either centering or detrending the data, the PSD remains approximately the same, without a peak at $w = 0$, representing the DC offset. For stationary data, the trend (line of best fit) is the mean, reflecting why both methods of preprocessing lead to the same result. Furthermore, by taking the $log_{10}(.)$ of the data before preprocessing and computing the PSD estimates, given that it is monotonic, we observe the same periodicities at different magnitudes, with a distinct peak at the same normalized frequency as for the original data.

**(b)** We were also tasked with carrying out spectral analysis on EEG samples (Volts) to determine the steady state visual evoked potential (SSVEP), obtained by flashing a visual stimulus at the subject at a fixed rate. The PSD estimates were computed with a standard periodogram, and averaged periodograms of window lengths (10s, 5s, 1s), and are shown in Figure 1.3. The periodograms displayed were computed using Welch's method with 0 overlap and a Hamming window.

From the standard periodogram, 5 small, yet discernible peaks peaks can be observed at 8.8Hz, 13Hz, 26Hz, 39Hz, and a large peak at 50Hz. The peak at 8.8Hz corresponds to the alpha-rhythm of the subject, as they were

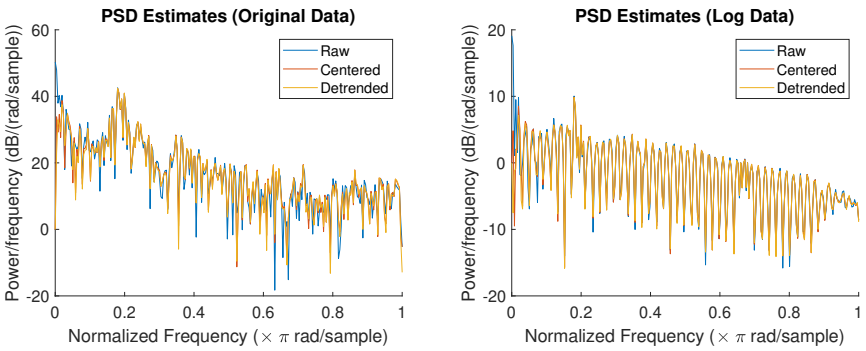


**Figure 1.2:** PSD estimates for different methods of preprocessing for the original data (left) and the log-transformed data (right)

tired, and the peak at 50Hz corresponds to power-line interference induced in the recording apparatus. Furthermore, the peak at 13Hz corresponds to the SSVEP, and the ones at 26Hz and 39Hz are the harmonics of the response.

While the averaged periodogram of window length 10s results in a large reduction in variance relative to the standard periodogram, as seen as in Figure 1.3, the averaging process also introduces more bias to the estimate. This can be observed as the harmonic at 39Hz is no longer discernible in the averaged periodogram. This trade-off between variance and bias is emphasised by the plot with window length 1s. While the variance is largely reduced by averaging, the bias introduced makes it so that SSVEP (13Hz) can no longer be differentiated from the alpha rhythm of the subject (8.8Hz).

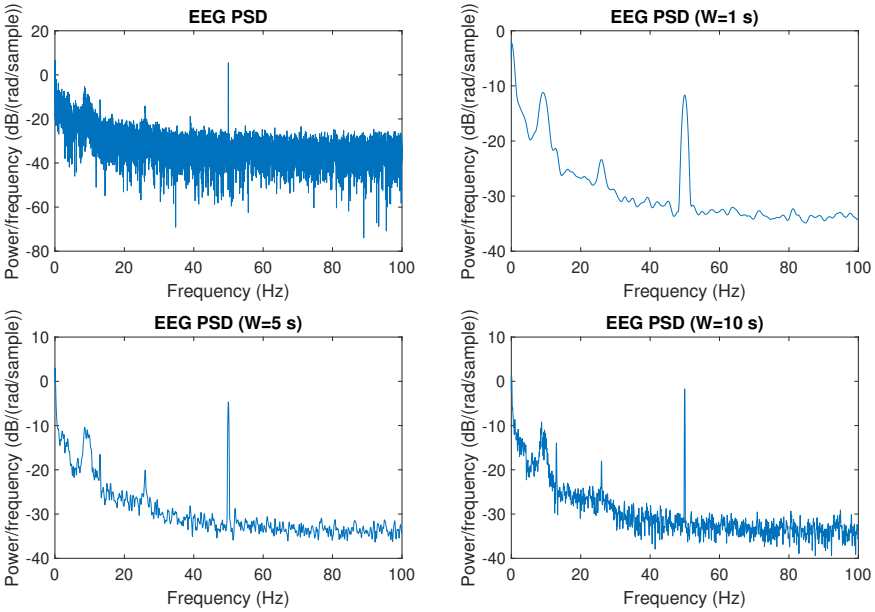


**Figure 1.3:** PSD estimates for a standard periodogram, and average periodograms for different window lengths.

## 1.3 Correlation Estimation

(a) A MATLAB script were created that calculate both biased and unbiased ACF estimates of a signal, where the estimates are then used to compute the correlogram. This is a simpler version of the Blackman-Tukey correlogram without windowing or removable of outer estimates. This method was tested on and computed for a WGN process, a noisy sinusoidal process and an AR(4) process, and the results are displayed in Figure 1.4.

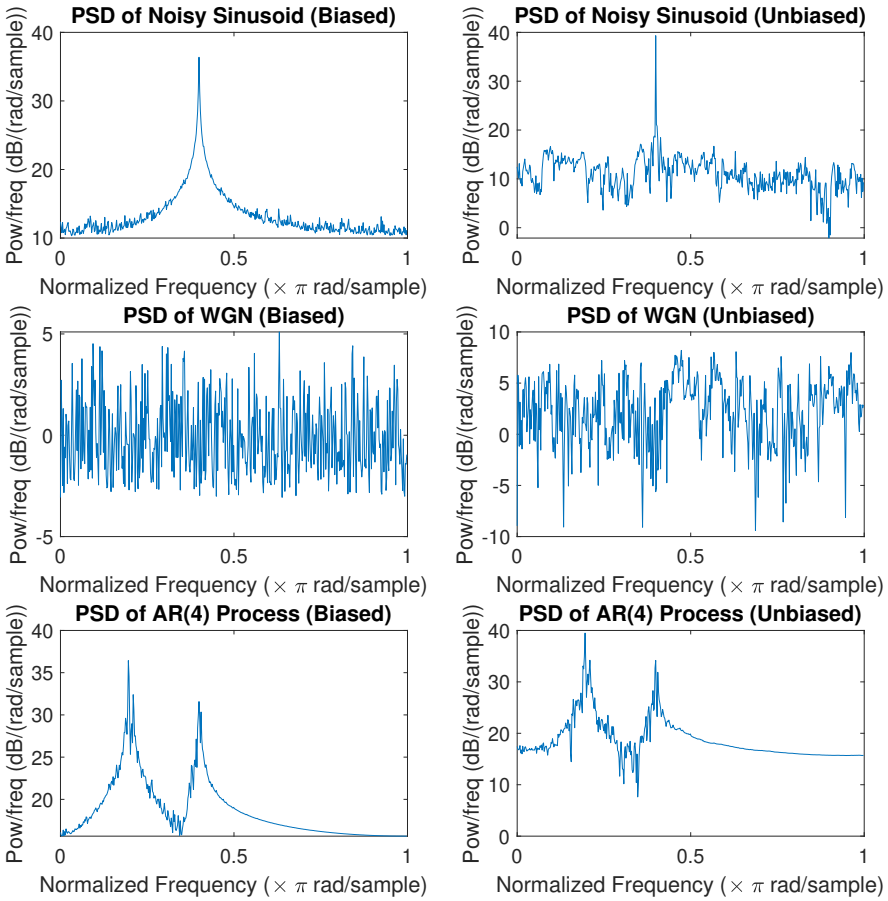


**Figure 1.4:** Correlogram spectral estimates, with biased (left) and unbiased (right) estimation of the ACF

As shown in Figure 4, the biased correlogram seems to exhibit much less variance than the unbiased, and from the MATLAB program, the unbiased has, on average, a larger imaginary part in its PSD estimate than its biased counterpart. This is likely due to the ACF not being positive definite because of erratic estimates for larger lags in the unbiased estimate.

**(b,c)** We were tasked with generating several realisations of a random process and different signals composed of sinusoids corrupted by noise, both normally and in dB. The results are displayed in Figure 1.5 for a noisy sinusoid with normalized frequencies 0.2 and 0.4.

In Figure 1.5, the mean of the peaks is more discernible in dB than when displayed normally. Plotting the PSD in dB also facilitates the visualization of different realizations when they exhibit large ensemble variance.

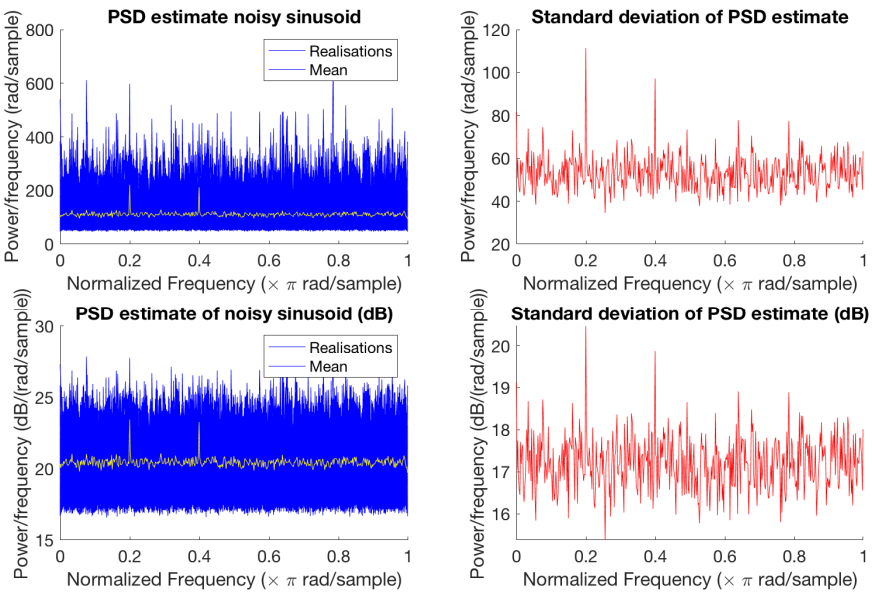


**Figure 1.5:** Correlogram spectral estimates of 100 realizations and their mean and standard deviations, plotted normally (top) and in dB (bottom)

**(d)** We were then tasked with generating complex exponential signals of different frequencies and length. In Figure 1.6, different periodogram estimates were computed for a different number of samples (normalized frequencies of 0.76 and 0.8). From Figure 1.6, it can be seen that as more data samples are considered, the periodogram tends to the correct line spectra.

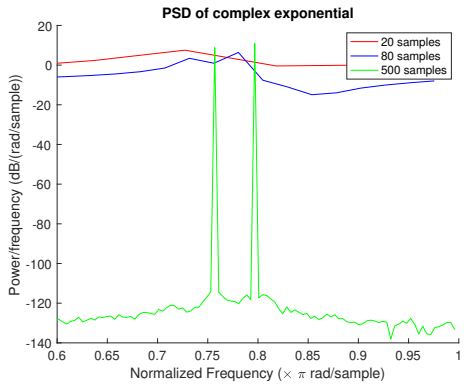


**Figure 1.6:** PSD periodogram estimate of complex exponentials (2 frequencies) for different numbers of samples

**(e)** Lastly, we were tasked with implementing the Multiple Signal Classification (MUSIC) algorithm using the `corrmtx` and `pmusic` functions in MATLAB. The former function is used to compute the correlation matrix, R. The 'mod' argument implies that function uses the forward-backward method, where AR parameters are computed and used for the ACF estimate. In the case of the code given, the argument '14' represents the model order. The latter function takes in the autocorrelation matrix estimate, R (allowed due to the 'corr' argument), and assumes 2 components in its analysis. It is then plotted within the range [0.25 0.40] Hz ($f_s = 1Hz$), which corresponds to the range [0.5, 0.8] in normalized frequency.

As opposed to estimating the PSD, the MUSIC algorithm computes a frequency-estimation function, also known as a pseudospectrum, with sharp peaks at the detected frequencies. The periodogram estimate and music pseudospectrum for a noisy complex exponential with two normalized frequencies (0.6, 0.64) and are displayed in Figure 1.7. From this figure, the MUSIC algorithm exhibits superior resolution to the periodogram, detecting all signal components, highlighting the algorithm's ability to estimate frequencies with accuracy higher than one sample. However, the MUSIC algorithm does have its limitations, suc as computational complexity, shown by the pseudospectrum's computation being almost 3 times slower than that of the periodogram for a signal of 30 samples. Additionally, the periodogram is also more flexible, as it does not require an input number of signal components like the MUSIC algorithm.

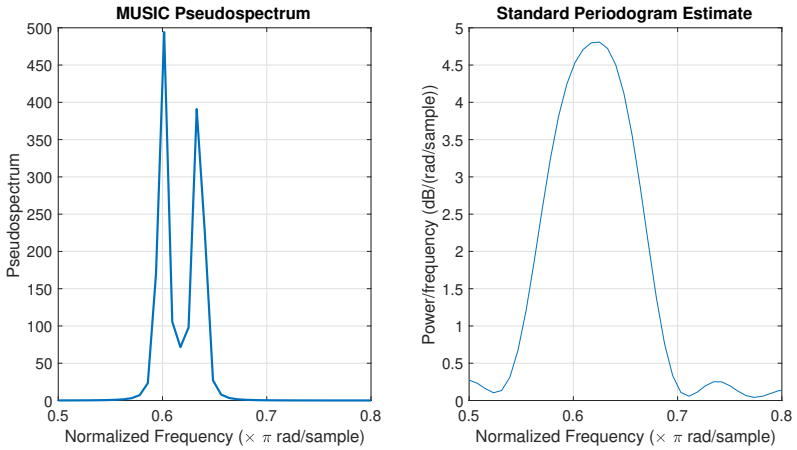


**Figure 1.7:** MUSIC pseudospectrum (left) and standard periodogram (right) of noisy complex exponential signal with two frequency components

## 1.4 Spectrum of Autoregressive Processes

**(a)** As we have so far seen, using the unbiased ACF estimate, while less biased, results in highly erratic estimates for the ACF at larger lags, where fewer samples are used to compute the estimate. If variant enough, the estimate might lead to AR parameters of an unstable process.

**(b, c)** 1000 samples were generated from the process defined by equation 1.14, describing an AR(4) process with driving input $w \sim \mathcal{N}(0, 1)$.

$$x(n) = 2.76x(n-1) - 3.81x(n-2) + 2.65x(n-3) - 0.92x(n-4) + w(n) \quad (1.14)$$

This was done with the use of the `filter` MATLAB command and discarding the first 500 samples. The true PSD of this process was obtained by taking the square of the magnitude of the frequency response of the AR coefficients, as the theoretical PSD of the driving input is 1. The frequency response and PSD estimates were obtained with the `freqz` and `pyulear` MATLAB functions respectively. These estimates were computed for both 1000 and 10000 samples, and the results are displayed below in Figure 1.8.

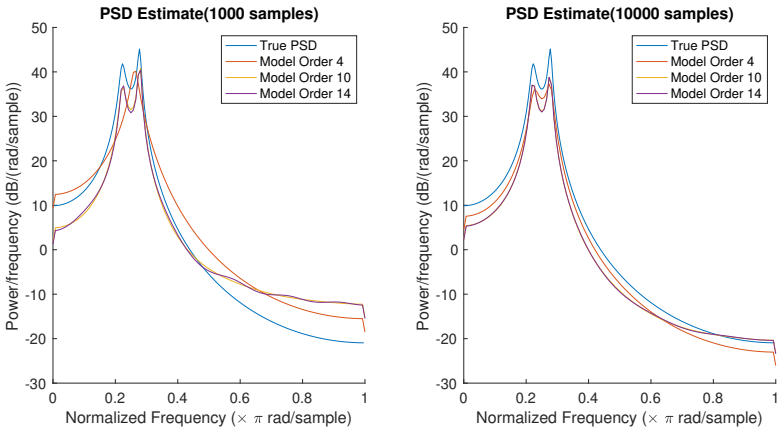


**Figure 1.8:** AR Spectral Estimates for model orders 4, 10, 14 with 1000 samples (left) and 10000 samples (right) for an AR(4) process

From Figure 1.8, it can be seen that the 500 samples used to estimate the AR spectral estimate were not sufficient for the 4th order estimate (true model order) to distinguish between the two peaks, whereas more complex estimates at model orders of 10 and 14 are able to distinguish between the two peaks, at the price of higher computational complexity and overfitting. None of the model orders used accurately estimate the PSD at higher frequencies. For 9500 samples (discarding the first 500 from a 10000-sample series), however, the 4th order estimate is able to discern the two peaks almost as well as the higher model orders, with all model orders providing better estimates for higher frequencies of the PSD.

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

**(a)** There were 3 different trials in the ECG experiment: trial 1 of unconstrained breathing, trial 2 with constrained breathing at 50 breaths per minute, and trial 3 with constrained breathing at 15 breaths per minute. Theoretically, the primary frequency components in RRI data correspond to the breathing rate of the subject. Before any estimation, the data was normalized and detrended. As displayed in Figure 1.9, three periodogram estimates were computed per trial: one standard periodogram, and two averaged periodograms (window lengths 50s and 150s). The periodogram methods were computed with the `pwelch` MATLAB function, with 0 overlap and a Hamming window.

**(b)** From the spectral analysis, the relationship $breaths/min(BPM) = 2 \times f$, can be used to determine estimates for BPM. From Figure 1.9, we can see that for the estimates from trial 1 that instead of a distinct peaks, there is a initial high amplitude band at low frequencies in the range 0-0.25Hz (0-12 breaths/min), representative of unconstrained breathing at natural breathing rates. From the estimates of trial 2, we observe one peak at 0.42Hz (50.4 breaths/min) and what is likely to be a harmonic at 0.83. Lastly, from trial 3, we observe a single peak at 0.12 Hz (14.4 breaths/min). Our estimates are consistent with the breathing restrictions of each trial (unconstrained, 50 BPM, 15 BPM).

**(c)** The optimal model orders were empirically determined to be 3, 7, and 4 for trials 1, 2 and 3 respectively. The AR spectral estimates are displayed in Figure 1.10, and were computed using the `pyulear` function. The most discernible peak is for trial 3 at 0.12Hz, matching the frequency obtained from the periodogram estimates. What the low frequency band in trial 1 is represented as a wide peak. The AR spectral estimate for trial 2 shows weakly defined peaks at the frequencies observed in the periodograms. While the periodograms exhibit much larger variance than AR spectral estimates, they also provide more well-defined peaks.

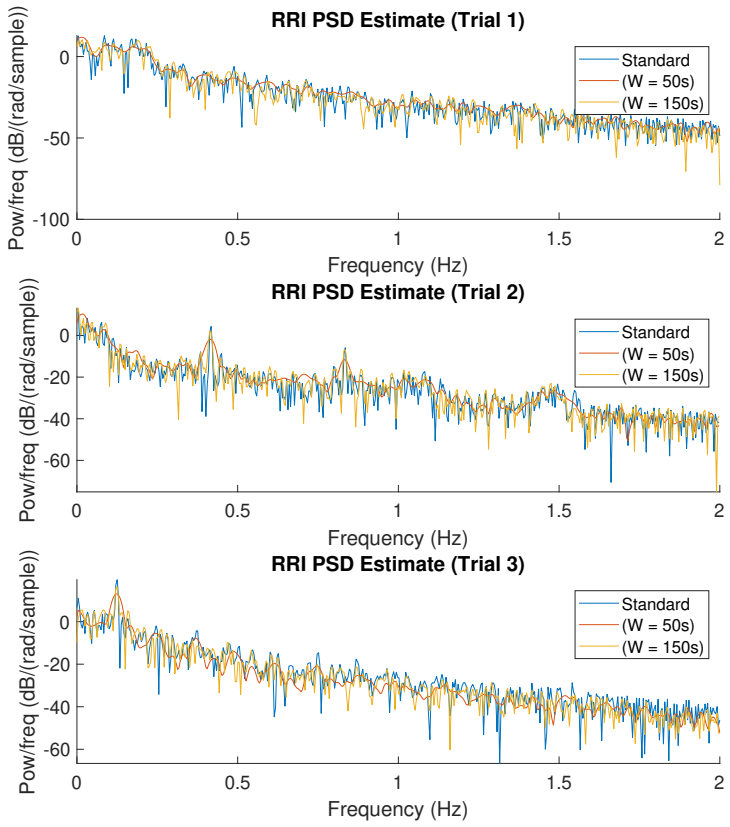


**Figure 1.9:** Welch periodogram estimates for the three trials of RRI data

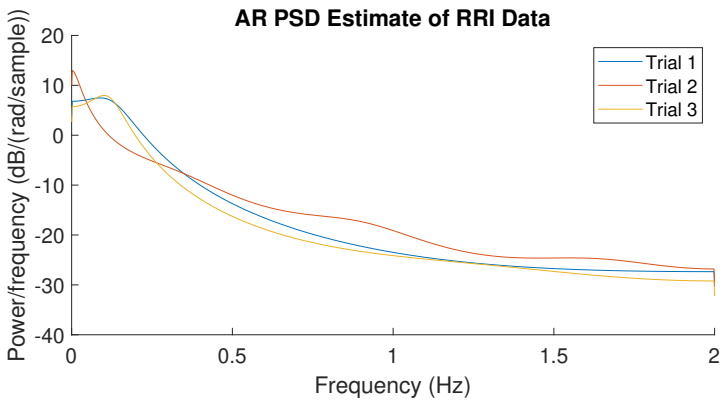


**Figure 1.10:** AR spectral estimates for the three trials of RRI data

## 1.6 Robust Regression

**(a)** We were given a file containing 5 matrices: an input matrix X, a noise-corrupted input matrix $\mathrm{X}_{noise}$, an training output matrix Y, and test input and output matrices $\mathrm{X}_{test}$ and $\mathrm{Y}_{test}$. In Figure 1.11, we have plotted the singular values (SVs) of X and $\mathrm{X}_{noise}$. While the underlying principal components are of rank 3 (as shown for $X$), the embedded noise in $X_{noise}$ makes it rank 10. In Figure 1.11 it can also be seen that the square error between the singular values becomes significantly larger for components in the noise subspace. As noise gets sufficiently large, the SVs of the noise subspace will be as large as the SVs of the signal subspace.

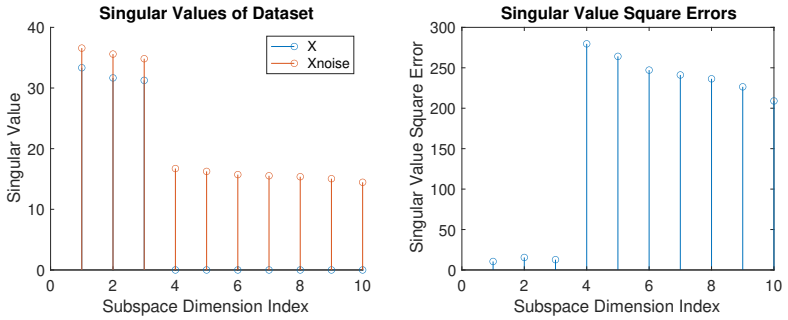


**Figure 1.11:** SVs of X and $\mathrm{X}_{noise}$ (left), and the square error between their SVs (right)

**(b)** A low-rank approximation of $\mathrm{X}_{noise}$ was obtained with the use of the `svds` function to only keep the 3 principal components, resulting the approximation $\tilde{X}_{noise}$. The MSE was computed for each input feature between X and $\mathrm{X}_{noise}$ and X and $\tilde{X}_{noise}$, and is shown in Figure 1.12. We can observe from Figure 1.12 that the MSE is significantly larger for (X, $\mathrm{X}_{noise}$) for all input features (columns). This difference in MSE is further reflected by the total MSEs (MSE(X, $\mathrm{X}_{noise}$)=0.2435, MSE(X, $\tilde{X}_{noise}$)=0.0733).

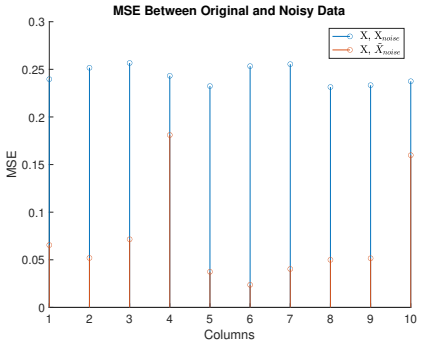


**Figure 1.12:** MSE Between the original data, $X$, and the data embedded in noise before and after its low-rank approximation ($X_{noise}$ and $\tilde{X}_{noise}$)

**(c)** We were then tasked with computing the ordinary least squares (OLS) and principal component regression (PCR) solutions for the parameter matrix B, which relates $\mathrm{X}_{noise}$ and Y, and comparing the solutions obtained through both methods. To obtain the PCR and OLS solutions, (1.15) and (1.16) were used.

$$\hat{\mathbf{B}}_{\mathbf{OLS}} = (\mathbf{X^T X})^{-1}\mathbf{X^T Y} \tag{1.15}$$

$$\hat{\mathbf{B}}_{\mathbf{pcr}} = \mathbf{V_{1:r}}(\mathbf{\Sigma_{1:r}})^{-1}\mathbf{U_{1:r}^T Y} \tag{1.16}$$

From part (a), it was determined that the signal space is composed of three principal components, meaning $\mathbf{r} = 3$. From the metrics shown below and Figure 1.13 for estimation and test error, we observe negligible differences in performance between the OLS and PCR methods for the given data.

**OLS** : $RMSE_{est} = 0.847, MAE_{est} = 0.658, RMSE_{test} = 0.689, MAE_{test} = 0.528$

**PCR** : $RMSE_{est} = 0.846, MAE_{est} = 0.658, RMSE_{test} = 0.686, MAE_{test} = 0.525$

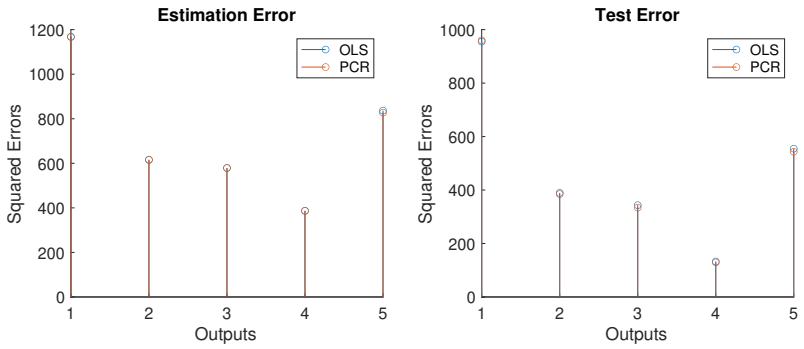


**Figure 1.13:** Squared Error between training and test data ($Y$ and $Y_{test}$)and the estimates generated from the PCR and OLS solutions

**(d)** Lastly, we were tasked with using the `regval` script provided to generate new realisations of data and the corresponding estimate for a given set of parameters, $B$. This was used for both the OLS and PCR solutions obtained to generate 100 realizations for each method, obtaining the averaged square error for each output as shown in Figure 1.14. This reflects how both methods exhibit strong performance in the estimation task, where the PCR method only outperforms the OLS method by 1.6% However, there are limitations, as the number of principal components either needs to be determined before-hand, which may not be straighforward.

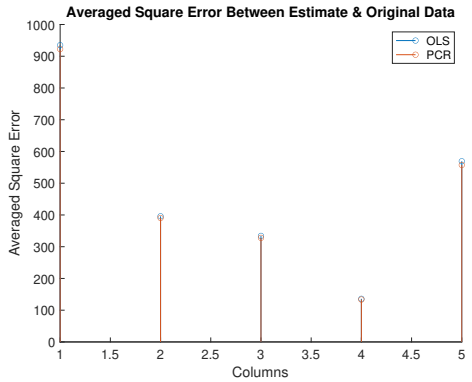


**Figure 1.14:** Averaged squared error between PCR & OLS estimates for different outputs (100 realizations)

# 2 Adaptive signal processing

## 2.1 The Least Mean Square (LMS) Algorithm

(a) This section will deal with adaptive filters. In particular, by treating the LMS filter as a dynamical system with inputs $x(n-1)$ and $x(n-2)$, outputs being the estimates $\hat{a}_1, \hat{a}_2, \hat{x}(n)$, we are asked to determine the entries of the correlation matrix of the input vector $\mathbf{x}(n) = [x(n-1), x(n-2)]^T$. The original parameters in this case are given by $a_1 = 0.1, a_2 = 0.8, \sigma_\eta^2 = 0.25$. The correlation matrix, $\mathbf{R}$, of the given input is shown in (2.1). Additionally, we have the relationships shown in (2.2) and (2.3) for any given AR(2) process.

$$\mathbf{R} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(1) & r_{xx}(0) \end{bmatrix} = \begin{bmatrix} \sigma_x^2 & r_{xx}(1) \\ r_{xx}(1) & \sigma_x^2 \end{bmatrix} \tag{2.1}$$

$$\sigma_x^2 = \left(\frac{1-a_2}{1+a_2}\right) \frac{\sigma_\eta^2}{(1-a_2)^2 - a_1^2} \tag{2.2}$$

$$\rho_1 = \frac{r_{xx}(1)}{r_{xx}(0)} = \frac{r_{xx}(1)}{\sigma_x^2} = \frac{a_1}{1-a_2} \tag{2.3}$$

Therefore, we can compute the following values to obtain the corresponding correlation matrix in (2.5).

$$\sigma_x^2 = \left(\frac{0.2}{1.8}\right) \frac{0.25}{0.2^2 - 0.1^2} = 0.926, \; r_{xx}(1) = 0.926\frac{0.1}{0.2} = 0.463 \tag{2.4}$$

$$\mathbf{R} = \begin{bmatrix} 0.926 & 0.463 \\ 0.463 & 0.926 \end{bmatrix} \tag{2.5}$$

We were also tasked with determining the range of step size values, $\mu$, will ensure that the LMS will converge in the mean. This range is given in (2.6).

$$0 < \mu < \frac{2}{\lambda_{max}} \tag{2.6}$$

Where $\lambda_{max}$ is the largest eigenvalue of the correlation matrix. Given the eigenvalues of the previously computed correlation matrix being $\lambda_1 = 0.463$ and $\lambda_2 = 1.389$, the step size range is $0 < \mu < 1.44$.

(b) An LMS adaptive predictor was implemented using N = 1000 samples for the same parameters as in the previous task, and plotting the squared error prediction over time for 1 trial and the average of the squared prediction error in dB over 100 realizations. For this task, a `lms_estimator` function was created, and the `filter` function was used to generate the samples (1500 samples and removing initial 500). The LMS algorithm was implemented by using instantaneous estimates of the ACF and CCF, resulting in the update rule shown in (2.7).

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{x}(n) \tag{2.7}$$

The results from the algorithm for the given parameters are displayed in Figure 2.1.

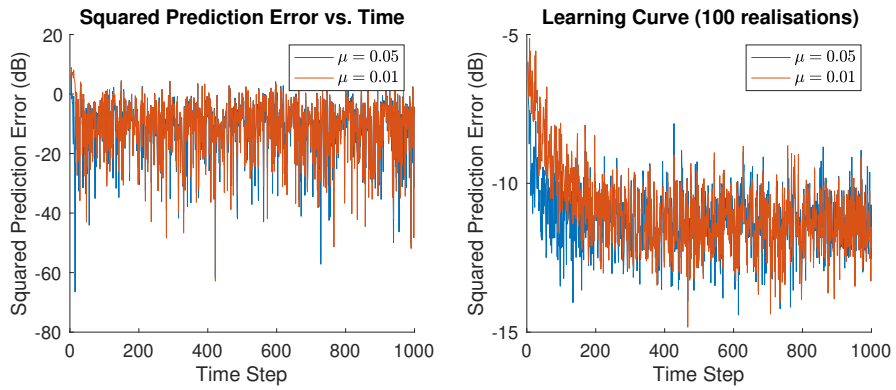


**Figure 2.1:** Squared Error Prediction Error (1 realization) and Learning Curve (100 realizations) for different step sizes

(c) We then had to estimate the misadjustment of the LMS by time-averaging over the steady state of the ensemble-averaged learning curves using 100 independent trials (realizations), comparing our results with the theoretical LMS misadjustment, given by (2.8). The misadjustment is the ratio of additional error power introduced by the adaptive filter, quantified by the ratio of the excess mean square error (EMSE) and the minimum attainable MSE of a Wiener filer, as shown in (2.9).

$$M_{LMS} \approx \frac{\mu}{2} Tr\{\mathbf{R}\} \tag{2.8}$$

$$M = \frac{EMSE}{\sigma_\eta^2} \tag{2.9}$$

For this section, the learning rates $\mu_1 = 0.05$ and $\mu_2 = 0.01$ were used. Given $\mathbf{R}$ from (a), we can compute the corresponding theoretical misadjustments to be $M_1 = 0.0463$ and $M_2 = 0.0093$. To estimate $M$ from the data, 100 realizations of 200,000 samples were generated and MSE curves were obtained for both learning rates. For each MSE curve, the first 1000 samples were discarded to not include non-steady state dynamics. Each MSE curve was then averaged over time to give the MSE estimate of the steady state, which was used to obtain the estimates $M_1 = 0.523$ and $M_2 = 0.0085$, giving an estimation error of approximately 10.8%. The estimate of $M_2$ exhibits less error, as the step size is smaller.

(d)We were then tasked with estimating the steady-state value of the adaptive filter coefficients with step sizes $\mu = 0.05$ and $\mu = 0.01$. This was done by averaging the parameter learning dynamics of 1,500 samples over 100 realizations, resulting in Figure 2.2. The parameters were estimated by the average of the last 100 samples in steady-state resulting in Table 2. As shown, while a smaller learning rate results in a closer approximation to the true model parameters, it also required more samples before convergence (120 for $\mu = 0.05$ and 900 for $\mu = 0.01$), highlighting a learning rate trade-off.

| | $\mu$ | |
|---|---|---|
| Parameter Estimate | 0.05 | 0.01 |
| $a_1$ | 0.069 | 0.086 |
| $a_2$ | 0.714 | 0.777 |
| Estimation Error | 20.9% | 8.4% |

**Table 2:** AR(2) LMS parameter estimates and estimation error

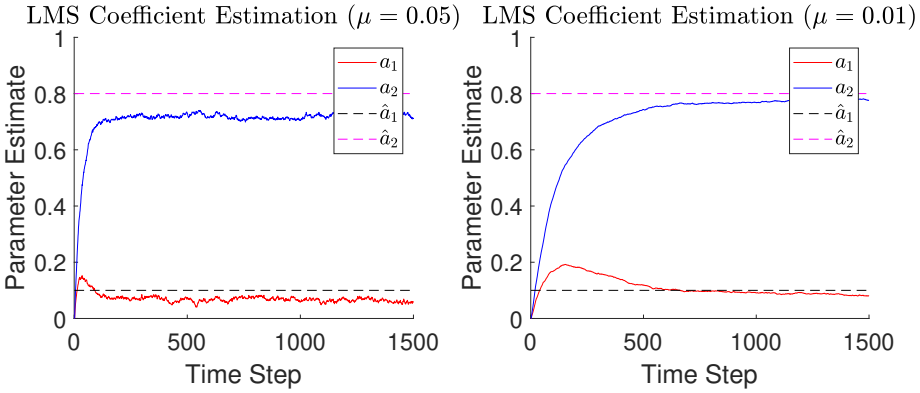


**Figure 2.2:** LMS parameter learning dynamics for an AR(2) process and different $\mu$

(e) Our next task was to derive the LMS coefficient update for $\mathbf{w}(n)$ that minimizes the cost function in (2.10).

$$J_2(n) = \frac{1}{2}(e^2(n) + \gamma||\mathbf{w}(n)||_2^2) \tag{2.10}$$

We first start by re-writing the quadratic term and 2-norm as matrix products (where $y(n)$ is the $n^{th}$ sample of the desired output):

$$J_2(n) = \frac{1}{2}\left[(y(n) - \mathbf{w^T}(n)\mathbf{x}(n))(y(n) - \mathbf{w^T}(n)\mathbf{x}(n))^T + \gamma\mathbf{w^T}(n)\mathbf{w}(n)\right] \tag{2.11}$$

$$= \frac{1}{2}\left[(y(n)^2 - d\mathbf{x^T}(n)\mathbf{w}(n) - d\mathbf{w^T}(n)\mathbf{x}(n) + \mathbf{w^T}(n)\mathbf{x}(n)\mathbf{x^T}(n)\mathbf{w}(n) + \gamma\mathbf{w^T}(n)\mathbf{w}(n)\right] \tag{2.12}$$

$$= \frac{1}{2}\left[(y(n)^2 - 2d\mathbf{w^T}(n)\mathbf{x}(n) + \mathbf{w^T}(n)\mathbf{x}(n)\mathbf{x^T}(n)\mathbf{w}(n) + \gamma\mathbf{w^T}(n)\mathbf{w}(n)\right] \tag{2.13}$$

We can then compute the gradient function w.r.t. $\mathbf{w}(n)$:

$$\nabla_w J_2(n) = -d\mathbf{x}(n) + \mathbf{x}(n)\mathbf{x^T}(n)\mathbf{w}(n) + \gamma\mathbf{w}(n) \tag{2.14}$$

$$= -e(n)\mathbf{x}(n) + \gamma\mathbf{w}(n) \tag{2.15}$$

By substituting (2.15) into the general steepest descent rule, the leaky LMS update rule is obtained as in (2.18)

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\nabla_w J_2(n) \tag{2.16}$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{x}(n) - \mu\gamma\mathbf{w}(n) \tag{2.17}$$

$$\mathbf{w}(n+1) = (1-\mu\gamma)\mathbf{w}(n) + \mu e(n)\mathbf{x}(n) \tag{2.18}$$

(f) LMS estimates were computed for the parameters defined in part (a) using a leaky LMS algorithm for different values of $\mu$ and $\gamma$. The results are shown in Figure 2.3. Unlike the standard LMS algorithm previously implemented, the parameter estimates do not converge to their true values, where the difference from the ground truth increases with an increasing leakage coefficient. This algorithm is typically implemented when one or more weights do not converge and cause instability, and has an effect on the estimates as shown in (2.19):

$$w_o = R^{-1}p \longrightarrow w_{leaky} = (R+\gamma I)^{-1}p \tag{2.19}$$

Thus causing parameters to shift away from their ground-truth estimates, but solving the problem posed by a singular $\mathbf{R}$.

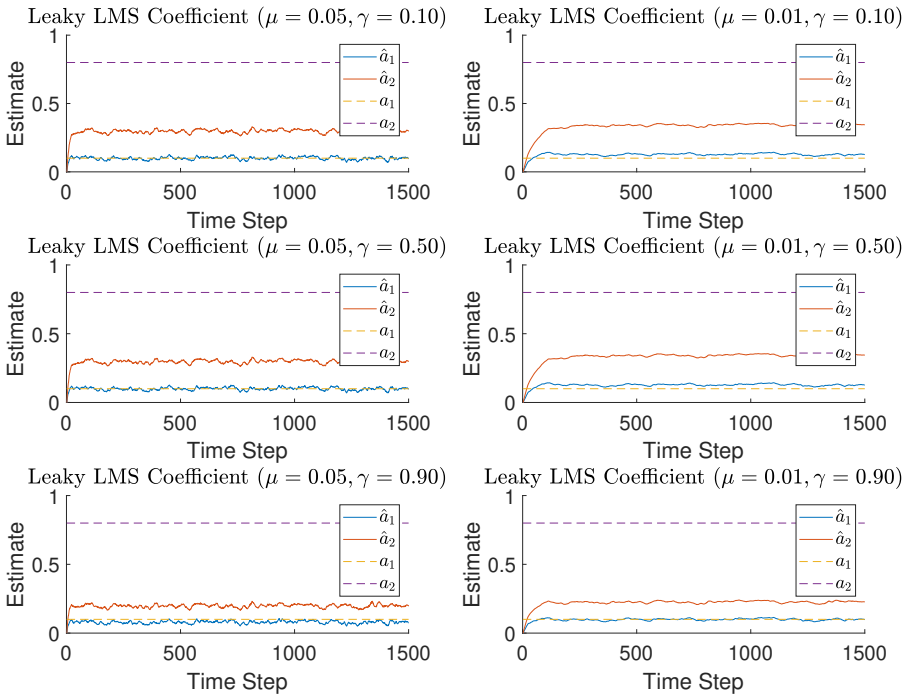


**Figure 2.3:** LMS parameter learning dynamics for an AR(2) process and different learning rates ($\mu$) and leakage coefficient ($\gamma$)

## 2.2 Adaptive Step Sizes

(a) For this section, we are working with adaptive step sizes, and were tasked with implementing the Benveniste, Ang & Farhang, and Mathews & Xie gradient adaptive step-size (GASS) algorithms. Within these algorithms, the weight update is modified to the form in (2.20), with step-size update rule given by (2.21). The update rules for the respective -(n) shown in (2.22), (2.23) and (2.24).

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(n)e(n)\mathbf{x}(n) \tag{2.20}$$

$$\mu(n+1) = \mu(n) + \rho e(n)\mathbf{x}^{\mathrm{T}}(n)\psi(n) \tag{2.21}$$

$$\text{Benveniste: } \psi(n) = \left[\mathbf{I} - \mu(n-1)\mathbf{x}(n-1)\mathbf{x}^{\mathrm{T}}(n-1)\right]\psi(n-1) + e(n-1)\mathbf{x}(n-1) \tag{2.22}$$

$$\text{Ang \& Farhang: } \psi(n) = \alpha\psi(n-1) + e(n-1)\mathbf{x}(n-1) \tag{2.23}$$

$$\text{Matthews \& Xie: } \psi(n) = \alpha\psi(n-1) + e(n-1)\mathbf{x}(n-1) \tag{2.24}$$

These algorithms were implemented and had their performance evaluated on the MA(1) model given by equation 2.25.

$$x(n) = 0.9\eta(n-1) + \eta(n) \qquad \eta \sim N(0, 0.5) \tag{2.25}$$

The performance of the algorithms w.r.t standard LMS is displayed in Figure 2.4. The weight error curves were obtained by averaging over 100 realizations of 100 samples for each algorithm. The hyper parameters were $\rho = 0.002$ (step-size adaptation parameter) and $\alpha = 0.8$. All three GASS algorithms exhibit faster convergence than standard LMS for $\mu = 0.01$, as well as for $\mu = 0.1$ given the initial learning rate ($\mu_0$) is large enough. More specifically, the Benveniste seems to converge the fastest, followed by Ang & Farhang, followed by Matthews & Xie. The GASS algorithms also exhibit less steady-state error than the standard LMS algorithm for larger step sizes, as their step sizes are relatively smaller after convergence. While GASS algorithms exhibit superior performance, they pose the disadvantage of the greater computational complexity involved in continuously adapting the step size as opposed to the static step size in the standard LMS algorithm.

(b) We were introduced to the normalized LMS (NLMS) algorithm, which can be used as shown in (2.26), to normalize step size, which can be useful since the weight update is proportional to $\mathbf{x}(n)$. The parameter $\epsilon$ is the regularization factor that prevents the algorithm from becoming unstable.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\beta}{\epsilon + ||\mathbf{x}(n)||_2^2}e(n)x(n) = \mathbf{w}(n) + \frac{\beta}{\epsilon + \mathbf{x}^{\mathbf{T}}(n)\mathbf{x}(n)}e(n)x(n) \tag{2.26}$$

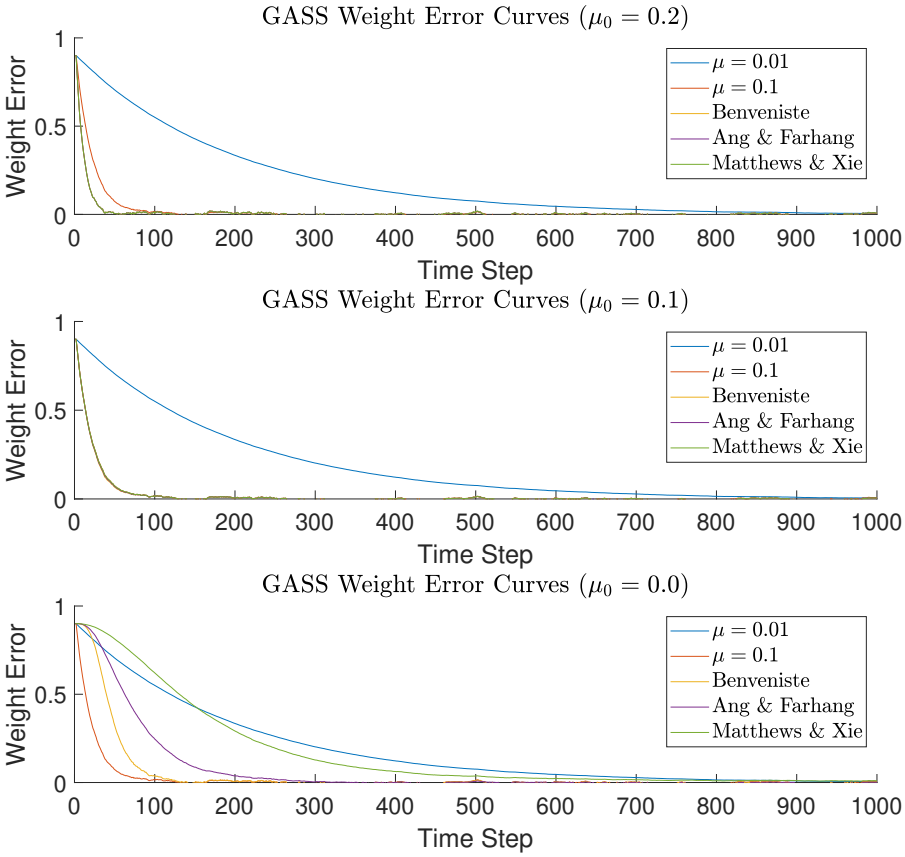


**Figure 2.4:** Weight error curves for standard LMS algorithms with $\mu = 0.01$ and $\mu = 0.1$, and GASS algorithms for different initial learning rates ($\mu_0$)

We were asked to show that the update rule in equation 2.27 in terms of the *a posteriori* error $e_p$ given by (2.28) is equivalent to the NLMS algorithm.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e_p(n)\mathbf{x}(n) \tag{2.27}$$

$$e_p(n) = d(n) - \mathbf{x}^{\mathbf{T}}(n)\mathbf{w}(n+1) \tag{2.28}$$

By expanding (2.28), and substituting in (2.27), we can obtain the *a posteriori* error in terms of the *a priori* error as follows:

$$e_p(n) = d(n) - \mathbf{x}^{\mathbf{T}}(n)\mathbf{w}(n+1) \tag{2.29}$$

$$e_p(n) = d(n) - \mathbf{x}^{\mathbf{T}}(n)(\mathbf{w}(n) + \mu e_p(n)\mathbf{x}(n)) \tag{2.30}$$

$$e_p(n)(1 - \mu\mathbf{x}^{\mathbf{T}}(n)\mathbf{x}(n)) = d(n) - \mathbf{x}^{\mathbf{T}}(n)\mathbf{w}(n) \tag{2.31}$$

$$e_p(n)(1 - \mu||\mathbf{x}(n)||_2^2) = e(n) \tag{2.32}$$

$$e_p = \frac{e(n)}{(1 - \mu||\mathbf{x}(n)||_2^2)} \tag{2.33}$$

By substituting the equation obtained for the *a posteriori* error into (2.27), we obtain the following:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \frac{e(n)}{(1 - \mu||\mathbf{x}(n)||_2^2)}\mathbf{x}(n) \tag{2.34}$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \frac{1}{\frac{1}{\mu} - ||\mathbf{x}(n)||_2^2} e(n)\mathbf{x}(n) \tag{2.35}$$

Which is equivalent to the NLMS algorithm for $\beta = 1$ and $\epsilon = \frac{1}{\mu}$.

(c) Lastly, we were tasked with implementing the GNGD algorithm and compare its estimates for parameters in task (a) with Benveniste's algorithm. The algorithm was implemented and the results are displayed in Figure 2.5 for the optimal hyper parameter values determined ($GNGD: \ \rho = 0.005, \ \mu_0 = 1$, $GNGD: \ \rho = 0.002, \ \mu_0 = 0.1$). From Figure 2.5, it can be seen that the GNGD algorithm converges with fewer samples (18 as opposed to 70 for Benveniste). Additionally, for a given number of parameters to be estimated, $M$, for the updates carried for a given sample, GNGD has a computational complexity of $\mathcal{O}(M)$, whereas Benveniste has a computational complexity of $\mathcal{O}(M^2)$. This is due to the first term of the $\psi$ update shown in (2.22), which is a filtering term used to provide low pass filtering of the instantaneous gradient.

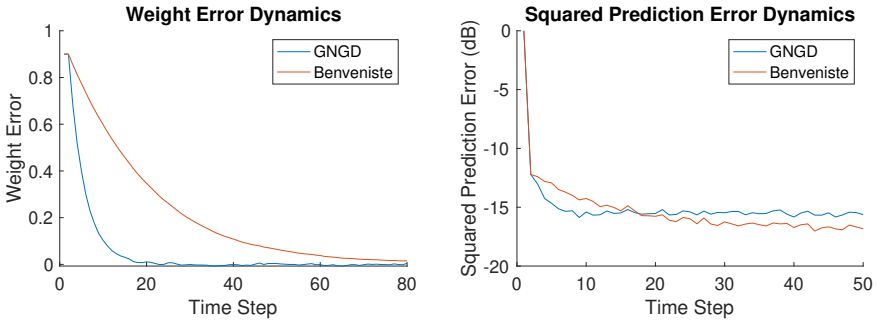


**Figure 2.5:** Weight error and squared prediction curves for GNGD and Benveniste algorithms

## 2.3 Adaptive Noise Cancellation

(a) This exercise deals with noise suppression for two de-noising adaptive filtering configurations. Given the signal in (2.36), whose coloured noise is given by the MA model in (2.37) ($v(n) \sim \mathcal{N}(0,1)$), we were tasked with determining the minimum delay ($\Delta$) that may be used in the adaptive linear enhancer (ALE) with filter length($M > 1$).

$$s(n) = x(n) + \eta(n) = \sin(0.01\pi n) + \eta(n) \tag{2.36}$$

$$\eta(n) = v(n) + 0.5v(n-2) \tag{2.37}$$

We can start by expanding the MSE as in (2.38):

$$E\left\{[s(n) - \hat{x}(n)]^2\right\} = E\left\{[(x(n) - \hat{x}(n)) + \eta(n)]^2\right\} \tag{2.38}$$

$$= E\left\{(x(n) - \hat{x}(n))^2\right\} + 2E\left\{\eta(n)x(n)\right\} - 2E\left\{\eta(n)\hat{x}(n)\right\} + E\left\{\eta^2(n)\right\} \quad (2.39)$$

The first term corresponds to the true mean square prediction error (MSPE), hence is irrelevant to the noise for a fixed model. Given that $x(n)$ is deterministic, the second term becomes 0. From equation 2.37, we can compute $E\left\{\eta(n)\right\} = 0$ and $E\left\{\eta^2(n)\right\} = \text{Var}(\eta(n)) = 1.25$. Additionally, the last term of the equation corresponds to the variance of the noise, which is constant. Therefore, to minimize the MSE with respect to the correlations in the noise, we must focus on the third term. For a given ALE system, $\hat{x}(n)$ is given by equation 2.40, whereas $\mathbf{u}(n)$ is given by equation 2.41.

$$\hat{x}(n) = \mathbf{w}^T(n)\mathbf{u}(n) \quad (2.40)$$

$$\mathbf{u}(n) = [s(n-\Delta), s(n-\Delta-1), ..., s(n-\Delta-M+1)]^T \quad (2.41)$$

We can thus take a closer look at the term $-2E\left\{\eta(n)\hat{x}(n)\right\}$.

$$-2E\left\{\eta(n)\hat{x}(n)\right\} = -2\mathbf{w}^T \begin{bmatrix} E\left\{\eta(n)(x(n-\Delta) - \eta(n-\Delta)\right\} \\ \vdots \\ E\left\{\eta(n)(x(n-\Delta-M+1) - \eta(n-\Delta-M+1)\right\} \end{bmatrix} \quad (2.42)$$

In the expansion of each term, as before, given the deterministic signal, and the zero-mean noise term, the first term goes to zero. Furthermore, the negatives cancel and we get a vector composed of values from the ACF of $\eta(n)$. Given that for this model, $E\left\{\eta(n)\eta(n-k)\right\} = 2\delta(k) + 0.5\delta(k-2)$ and that $M > 1$, we get that $\Delta_{min} = 3$, being the minimum $\Delta$ to set the above term to 0.

(b) The ALE system was implemented via an LMS algorithm, where a step-size of $\mu = 0.01$ was used and with optimal parameters determined to be $\Delta = 3$ and $M = 5$, as shown in Figure 2.6. It is shown to improve SNR in Figure 2.6. We were also tasked determining the dependence of MSPE on filter order and delay. Results are shown in Figure 2.7, where MSPE decreases drastically until reaching the optimal delay, then increases further as delay increases, likely reflecting the model overfitting. Likewise, we also see that beyond an optimal filter order, the MSPE increases. These results reflect the need for determining the appropriate hyperparameters for a given task.

(c) We were also tasked with implementing the adaptive noise cancellation (ANC) system with MATLAB. Given that for this configuration, the secondary noise input must be correlated with the primary noise input, the secondary noise input is given by (2.43).

$$\epsilon(n) = 0.5\eta(n) - 0.1\eta(n-2) \quad (2.43)$$

The performance of the algorithms was compared and the results are displayed in Figure 2.8, where it can be seen that while the ALE algorithm converges rapidly to the clean signal reconstruction, the takes longer to converge, reflecting an advantage of the ALE algorithm. However, given enough samples, the ANC clean signal reconstruction obtains a significantly

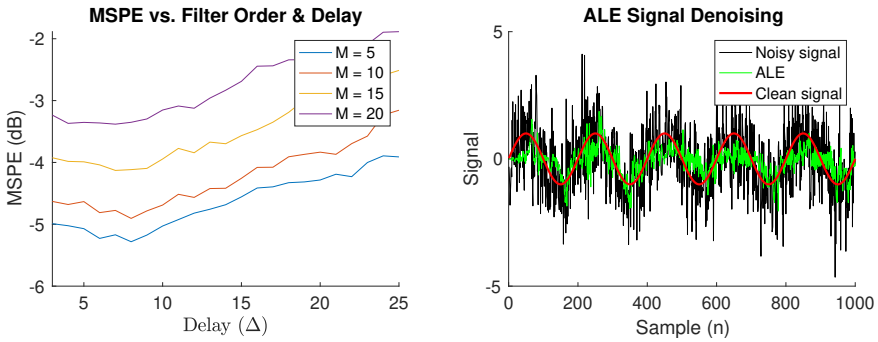


**Figure 2.6:** MSPE of the ALE plotted against multiple delays and filter orders (left) and its application to the noisy sinusoid generated (right)

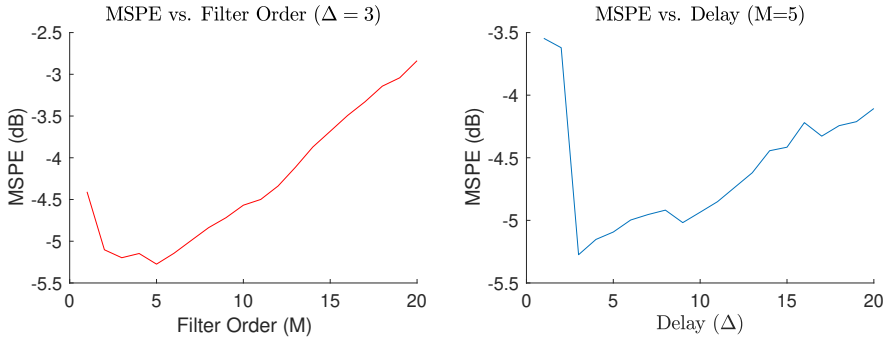


**Figure 2.7:** MSPE of the ALE plotted against filter order with $\Delta = 3$ (left) and against delay with $M = 5$ (right)

smaller steady-state MSPE, reflecting an advantage of the the ANC system that is supported by the MPSE plot in Figure 2.8. Therefore, if enough samples are provided, the ANC system exhibits superior performance. However, it is important to note that ANC requires a secondary noise input, which may be difficult/impossible to obtain in particular use cases.

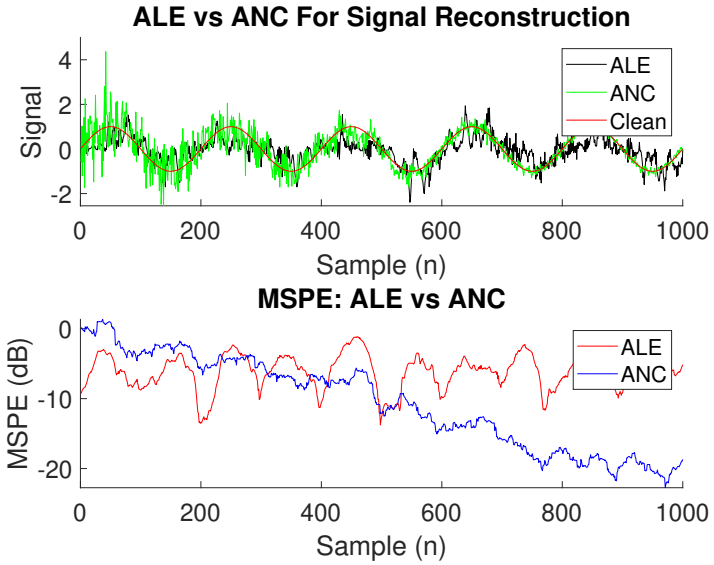


**Figure 2.8:** Clean signal reconstruction comparison between ALE and ANC systems (left) and the MSPE dynamics of ALE and ANC systems (right))

(d) Lastly, we were tasked with loading in the EEG data and using the ANC configuration to remove the 50Hz component introduced by mains by using a synthetic reference input composed of a noisy sinusoid of 50Hz corrupted by white noise. The white noise was chosen to be $w(n) \sim \mathcal{N}(0, 0.01^2)$. Spectrograms were computed with rectangular windows of length of 3540, with an overlap ratio of 0.3. The data was detrended before the procedure.

This procedure was carried out for both the POz and Cz data as shown in Figures 2.9 and 2.10, where the optimal filter order and learning rate combinations were determined empirically. While determining the optimal parameters, lower filter orders were not able to remove the frequency component, whereas higher filter orders were able to remove the component, but the relevant frequency band was replaced with a low intensity band that did not match the original baseline of the spectrogram. This observation further highlights the need for determining optimal parameter values.

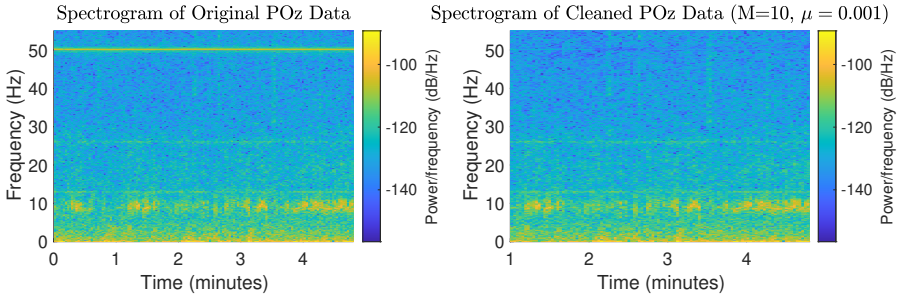


**Figure 2.9:** Spectrograms of raw POz (left) and of ANC cleaned POz (right)

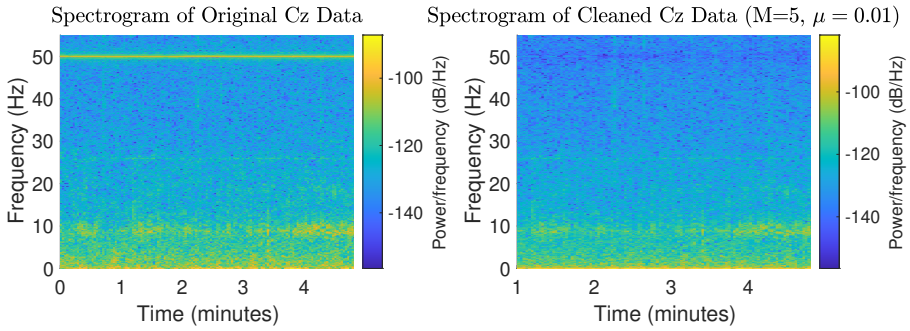


**Figure 2.10:** Spectrograms of raw Cz (left) and of ANC cleaned Cz (right)

# 3 Widely Linear Filtering and Adaptive Spectrum Estimation

## 3.1 Complex LMS and Widely Linear Modelling

(a) This section deals with complex-valued signals and their pivotal role in communications. We were first tasked with generating a first-order widely-linear-moving-average process, WLMA(1), in (3.1)

$$y(n) = x(n) + b_1 x(n-1) + b_2 x^*(n-1) \tag{3.1}$$

where $x \sim \mathcal{N}(0,1)$, $b_1 = 1.5 + 1j$ and $b_2 = 2.5 - 0.5j$. Once generated, the circularity coefficient, $\eta$, was computed for $x(n)$ and $y(n)$ as in (3.2)

$$\eta = |E[zz^*]^{-1}E[zz]| \tag{3.2}$$

The coefficients computed were $\eta_x = 0.036$ and $\eta_y = 0.856$, reflecting how $x(n)$ is second-order circular, whereas $y(n)$ is second-order noncircular, or improper, which is further reflected by the circularity plots in Figure 3.1. We are also tasked with implementing the complex LMS (CLMS) and the augmented CLMS (ACLMS) to estimate $b_1$ and $b_2$. Since the CLMS acts as a generic extension of LMS, it is only valid for circular random variables and is not guaranteed to capture the second-order statistical relationship between input and the output. The ACLMS uses an augmented input and is able to fully exploit second order statistics. This can be seen in Figure 3.1, where the ACLMS is able to learn the parameters, with an average steady-state MSPE of -305.78 dB, whereas the CLMS algorithm does not have enough degrees of freedom and remains at approximately -8.75 dB.

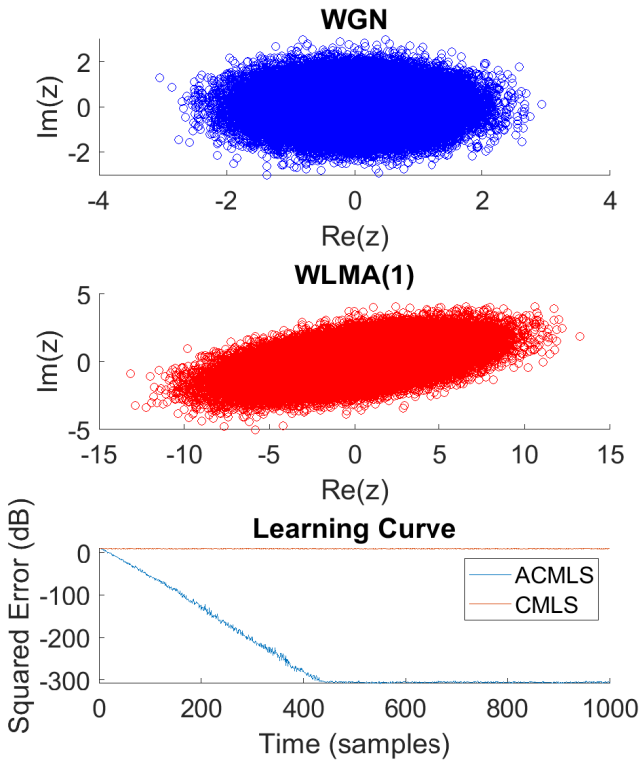


**Figure 3.1:** Circularity plots of $x(n)$, $y(n)$, and learning curve for ACMLS and CMLS algorithms ($\mu = 0.1$)

(b) We were then tasked with applying the CLMS and ACLMS algorithms to three regimes of bivariate wind data ($v_{east}[n]$,$v_{north}[n]$): low, medium and high wind speeds. The bivariate data was modelled as a complex-valued wind signal, as in (3.3)

$$v[n] = v_{east} + jv_{north} \tag{3.3}$$

Once in complex form, the circularity coefficients were determined to be $\eta_{low} = 0.159$, $\eta_{medium} = 0.454$ and $\eta_{high} = 0.623$. This implies that the low regime can be treated as approximately proper, whereas medium and high speed data are improper. This is further reflected by the circularity plots shown in Figure 3.2.

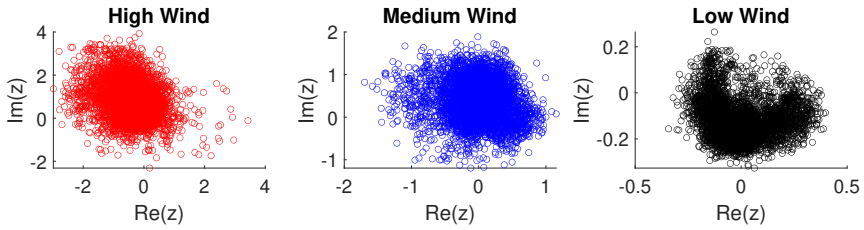


**Figure 3.2:** Circularity plots of wind data for three regimes

The CLMS and ACLMS filters were used in a prediction setting to perform a one-step ahead prediction of the complex wind data for each regime. The optimal learning rates were determined empirically to be $\mu_{high} = 0.001$, $\mu_{medium} = 0.01$ and $\mu_{low} = 0.1$, as to minimize prediction error and maintaining the filter stable. A grid-search was carried out for model orders $M \in [1:1:20]$ and the MSPE for different filters/regimes is shown in Figure 3.3, which shows that for all wind regimes, the ACLMS outperforms the CLMS filter if at the optimal model order. Given larger number of degrees of freedom of ACMLS relative to CLMS, overfitting increases more rapidly as a function of model order.

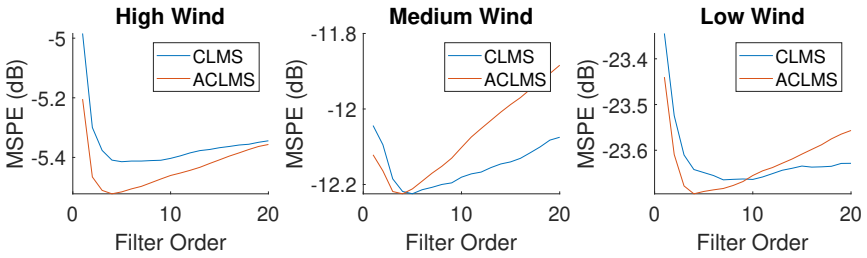


**Figure 3.3:** MSPE vs. filter order for different wind regimes and filters

(c) The next task involves frequency estimation in three phase power systems, composed of three-phase voltages $v_a$, $v_b$ and $v_c$ with magnitudes $V_a(n)$, $V_b(n)$ and $V_c(n)$ as well as phase distortions relative to the balanced system $\Delta_b$ and$\Delta_c$. Using the Clarke transform, these can be projected onto $v_0$, $v_\alpha$ and $v_\beta$. Under balanced conditions, all voltages have the same, constant amplitude, $V$, and have a phase shift of exactly $\frac{2\pi}{3}$. If balanced, $v_0$ becomes 0 and the $\alpha$ and $\beta$ components can be represented as a complex Clarke voltage, $v(n) = v_\alpha(n) + jv_\beta(n)$. The balanced Clarke voltage is given by (3.4), whereas if unbalanced, it is given by (3.5)

$$v(n) = \sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_0}{f_s} n + \phi)} \tag{3.4}$$

$$v(n) = A(n)e^{j(2\pi \frac{f_0}{f_s} n + \phi)} + B(n)e^{-j(2\pi \frac{f_0}{f_s} n + \phi)} \tag{3.5}$$

where $A(n) = \frac{\sqrt{6}}{6}(V_a(n) + V_b e^{j\Delta_b} + V_c e^{j\Delta_c})$ and $B(n) = \frac{\sqrt{6}}{6}(V_a(n) + V_b(n)e^{-j(\Delta_b+\frac{2\pi}{3})} + V_c(n)e^{-j(\Delta_c-\frac{2\pi}{3})})$. We were asked to generate balanced and unbalanced voltage systems (Clarke voltages). Three voltages were generated ($f_s = 480Hz$): one balanced, one unbalanced due to differences in magnitude ($\Delta V$), and one unbalanced due to phase distortions ($\Delta\phi$). The circularity coefficients were determined to be $\eta_{balanced} = 0$, $\eta_{\Delta V} = 0.528$ and $\eta_{\Delta\phi} = 0.661$ respectively. The circularity plot of these three systems are shown in Figure 3.4, where only the balanced system is circular. Figure 3.4 also shows a monotonic increase circularity as the system becomes more unbalanced, which means that circularity diagrams can be used to visualize how faulty (unbalanced) the system is.

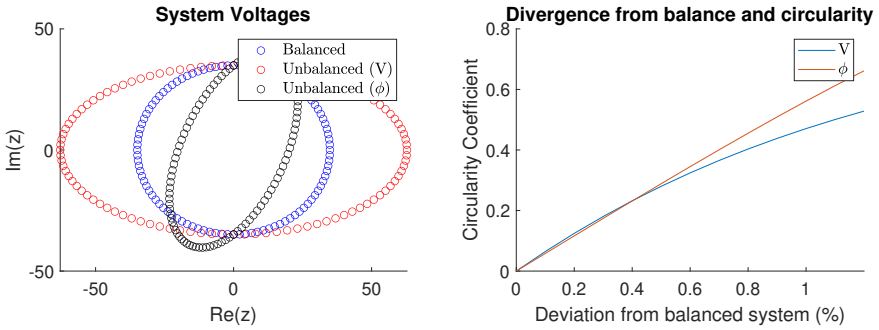


**Figure 3.4:** Circularity plots of different systems (left) and circularity coefficient as a function of deviations from the system (modulating $\Delta_b$ and $V_a$ with other parameters constant)

(d) We were then asked to show how the frequency, $f_0$, a balanced and unbalanced system can be derived from the coefficients learned in the Strictly Linear AR(1) (SLAR(1)) and Widely Linear AR(1) (WLAR(1)) models respectively. The SLAR(1) is computed as in (3.6) and the WLAR(1) is determined as in (3.7)

$$v(n+1) = h^*(n)v(n) \tag{3.6}$$

$$v(n+1) = h^*(n)v(n) + g^*(n)v^*(n) \tag{3.7}$$

In the balanced case, this can be done by using (3.4) for $v(n+1)$ and (3.6) to get (3.8)

$$v(n+1) = \sqrt{\frac{3}{2}}Ve^{j(2\pi\frac{f_0}{f_s}(n+1)+\phi)} = h^*(n)v(n) \tag{3.8}$$

$$\sqrt{\frac{3}{2}}Ve^{j(2\pi\frac{f_0}{f_s}n+\phi)}e^{j(2\pi\frac{f_0}{f_s})} = h^*(n)\sqrt{\frac{3}{2}}Ve^{j(2\pi\frac{f_0}{f_s}n+\phi)} \tag{3.9}$$

$$e^{j(2\pi\frac{f_0}{f_s})} = h^*(n) \tag{3.10}$$

where, for equality, the phase must also be equal, resulting in (3.11)

$$2\pi\frac{f_0}{f_s} = \arctan\left\{\frac{-\mathcal{I}\{h(n)\}}{\mathcal{R}\{h(n)\}}\right\} \tag{3.11}$$

The result can be further simplified by neglecting the sign, which only determines the direction of rotation, resulting in (3.12)

$$f_0 = \frac{f_s}{2\pi}\arctan\left\{\frac{\mathcal{I}\{h(n)\}}{\mathcal{R}\{h(n)\}}\right\} \tag{3.12}$$

The same reasoning can be applied for the unbalanced system, whereby combining (3.7) and (3.5) for v(n+1) results in (3.13)

$$A(n+1)e^{j(2\pi\frac{f_0}{f_s}(n+1)+\phi)} + B(n+1)e^{-j(2\pi\frac{f_0}{f_s}(n+1)+\phi)} = h^*(n)v(n) + g^*(n)v^*(n) \tag{3.13}$$

$$\begin{aligned} &A(n+1)e^{j(2\pi\frac{f_0}{f_s}n+\phi)}e^{j(2\pi\frac{f_0}{f_s})} + B(n+1)e^{-j(2\pi\frac{f_0}{f_s}n+\phi)}e^{-j(2\pi\frac{f0}{f_s})} \\ &= h^*(n)(A(n)e^{j(2\pi\frac{f_0}{f_s}n+\phi)} + B(n)e^{-j(2\pi\frac{f_0}{f_s}n+\phi)}) \\ &\quad + g^*(A^*(n)e^{-j(2\pi\frac{f_0}{f_s}n+\phi)} + B^*(n)e^{j(2\pi\frac{f_0}{f_s}n+\phi)}) \end{aligned} \tag{3.14}$$

$$\begin{aligned} &A(n+1)e^{j(2\pi\frac{f_0}{f_s}n+\phi)}e^{j(2\pi\frac{f_0}{f_s})} + B(n+1)e^{-j(2\pi\frac{f_0}{f_s}n+\phi)}e^{-j(2\pi\frac{f0}{f_s})} \\ &= (h^*(n)A(n) + g^*(n)B^*(n))e^{j(2\pi\frac{f_0}{f_s}n+\phi)} \\ &\quad + (h^*(n)B(n) + g^*(n)A^*(n))e^{-j(2\pi\frac{f_0}{f_s}n+\phi)} \end{aligned} \tag{3.15}$$

Comparing the coefficients of $e^{j(2\pi\frac{f_0}{f_s}n+\phi)}$ and $e^{-j(2\pi\frac{f_0}{f_s}n+\phi)}$ on both sides, results in (3.16)

$$\begin{aligned} A(n+1)e^{j(2\pi\frac{f_0}{f_s})} &= (h^*(n)A(n) + g^*(n)B^*(n)) \\ B(n+1)e^{-j(2\pi\frac{f_0}{f_s})} &= (h^*(n)B(n) + g^*(n)A^*(n)) \end{aligned} \tag{3.16}$$

where by using the simplifying assumption held implicitly in frequency estimation by adaptive filtering that $A(k+1) \approx A(k)$ and $B(k+1) \approx B(k)$

$$\begin{aligned} A(n)e^{j(2\pi\frac{f_0}{f_s})} &= h^*(n)A(n) + g^*(n)B^*(n) \\ B(n)e^{-j(2\pi\frac{f_0}{f_s})} &= h^*(n)B(n) + g^*(n)A^*(n) \end{aligned} \tag{3.17}$$

$$\begin{aligned} e^{j(2\pi\frac{f_0}{f_s})} &= h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} \\ e^{-j(2\pi\frac{f_0}{f_s})} &= h^*(n) + g^*(n)\frac{A^*(n)}{B(n)} \end{aligned} \tag{3.18}$$

By considering the LHS of both equations in (3.18), it can be seen that the expression on the RHS of both equations are conjugates of each other, resulting in (3.20)

$$h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} = h(n) + g(n)\frac{A(n)}{B^*(n)} \tag{3.19}$$

$$g^*(n)\left(\frac{B^*(n)}{A(n)}\right)^2 - 2j\mathcal{I}\{h(n)\})\frac{B^*(n)}{A(n)} + g(n) = 0 \tag{3.20}$$

which is quadratic in $\frac{B^*(n)}{A(n)}$ and can be solved with the quadratic formula

$$\frac{B^*(n)}{A(n)} = \frac{2j\mathcal{I}\{h(n)\})}{2} \pm \frac{1}{2}\sqrt{(-2j\mathcal{I}\{h(n)\})^2 + 4g^*(n)g(n)} \tag{3.21}$$

$$\frac{B^*(n)}{A(n)} = \frac{j\mathcal{I}\{h(n)\}}{g^*} \pm \frac{1}{2g^*(n)}\sqrt{-\mathcal{I}\{h(n)\}^2 + |g(n)|^2} \tag{3.22}$$

$$\frac{B^*(n)}{A(n)} = \frac{j\mathcal{I}\{h(n)\}) \pm \sqrt{-\mathcal{I}\{h(n)\}^2 + |g(n)|^2}}{g^*(n)} \tag{3.23}$$

$$\frac{B^*(n)}{A(n)} = j\frac{\mathcal{I}\{h(n)\}) \pm \sqrt{\mathcal{I}\{h(n)\}^2 - |g(n)|^2}}{g^*(n)} \tag{3.24}$$

Since $\frac{B^*(n)}{A(n)}$ is expressed in terms of the coefficients, (3.24) can be substituted into the expressions in (3.18)

$$e^{j(2\pi\frac{f_0}{f_s})} = h^*(n) + g^*(n)j\frac{\mathcal{I}\{h(n)\}) \pm \sqrt{\mathcal{I}\{h(n)\}^2 - |g(n)|^2}}{g^*(n)} \tag{3.25}$$

$$e^{j(2\pi\frac{f_0}{f_s})} = \mathcal{R}\{h(n)\} - j\mathcal{I}\{h(n)\}) + j\mathcal{I}\{h(n)\}) \pm j\sqrt{\mathcal{I}\{h(n)\}^2 - |g(n)|^2} \tag{3.26}$$

$$e^{j(2\pi\frac{f_0}{f_s})} = \mathcal{R}\{h(n)\} \pm j\sqrt{\mathcal{I}\{h(n)\}^2 - |g(n)|^2} \tag{3.27}$$

where under the assumption that $\mathcal{I}\{h(n)\}^2 - |g(n)|^2 > 0$ and since the phase must be equal for the LHS and RHS

$$2\pi\frac{f_0}{f_s} = \arctan\left(\pm\frac{\sqrt{\mathcal{I}\{h(n)\}^2 - |g(n)|^2}}{\mathcal{R}\{h(n)\}}\right) \tag{3.28}$$

$$f_0 = \frac{f_s}{2\pi}\arctan\left(\frac{\sqrt{\mathcal{I}\{h(n)\}^2 - |g(n)|^2}}{\mathcal{R}\{h(n)\}}\right) \tag{3.29}$$

where the positive definition is chosen given that $f_s >> f_0$, implying $e^{j(2\pi\frac{f_0}{f_s})}$ has a positive imaginary part.

(e) Lastly, the CLMS and ACLMS filters were used to estimate the frequency of the generated balanced and unbalanced voltages ($f_0 = 50Hz$) using (3.12) and (3.29). This procedure was carried out for the same three systems as in part (c), where $f_s = 480Hz$. The results are shown in Figure 3.5. A grid-search was carried out to determine the optimal $\mu$ to prevent unstable learning, which was found to be $\mu = 1 \times 10^{-5}$. As expected, while CLMS and ACLMS are both able to learn the appropriate coefficients for the estimating the frequency accurately of a balanced system (second-order circular), only the ACLMS converges to the correct frequency for unbalanced systems. Additionally, only ACLMS is able to overcome oscillation errors after converging for unbalanced systems.

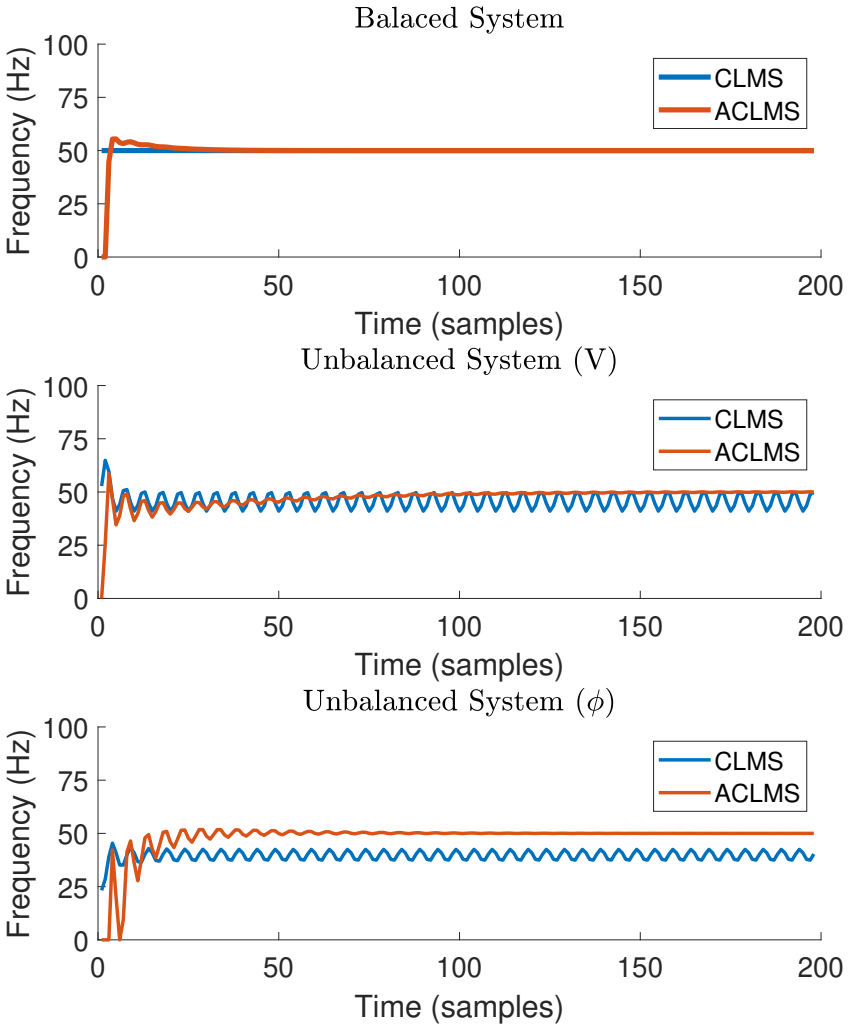


**Figure 3.5:** Frequency estimation using CLMS and ACLMS algorithms for a balanced system, and two unbalanced systems (magnitude and phase distortions)

## 3.2 Adaptive AR Model Based Time-Frequency Estimation

(a) A frequency modulated (FM) signal $y(n) = e^{j()\frac{2\pi}{fs}\phi(n)} + \eta(n)$, where $\eta(n) \sim \mathcal{N}(0, 0.05)$ and $\phi(n)$ was generated from (3.30)

$$\frac{d\phi(n)}{dn} = \begin{cases} 100 & 1 \leq n \leq 500 \\ 100 + \frac{n-500}{2} & 501 \leq n \leq 1000 \\ 100 + \left(\frac{n-1000}{25}\right)^2 & 1001 \leq n \leq 1500 \end{cases} \tag{3.30}$$

The signal has a frequency response generated as in Figure 3.6. The `aryule` function was used to find the AR(1) coefficient for the signal, as well as to compare the AR(1) estimate with other model orders. The power spectral estimates obtained are displayed in Figure 3.6. As autoregressive spectral estimation assumes stationarity, it is not able to capture the changes

in frequency for any model order and instead finds the optimal stationary representation, capturing the average frequency content. This is reflected by the AR(1) peak being at the average frequency of the FM signal ($\text{FM}_{\text{avg}}$).

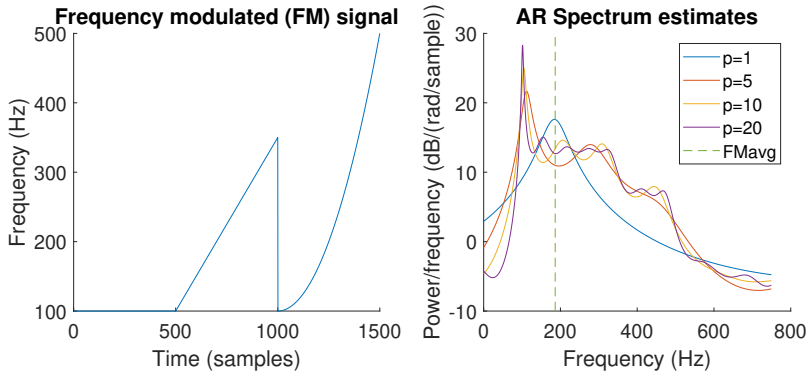


**Figure 3.6:** FM signal frequency content (left) and AR spectral estimates (right)

(b) The next task was to implement the CLMS algorithm to estimate the AR coefficients of $y(n)$ in (a) at each time instant, and compute the frequency spectrum of the signal. The time-frequency spectrum was obtained by using the `mesh` function, and is displayed in Figure 3.7. The CLMS is algorithm is able to successfully capture the changes of frequency over time. However, it is important to choose an appropriate learning rate to ensure learning the correct coefficients within the given interval, which is not achieved by using $\mu = 0.001$ in this example.

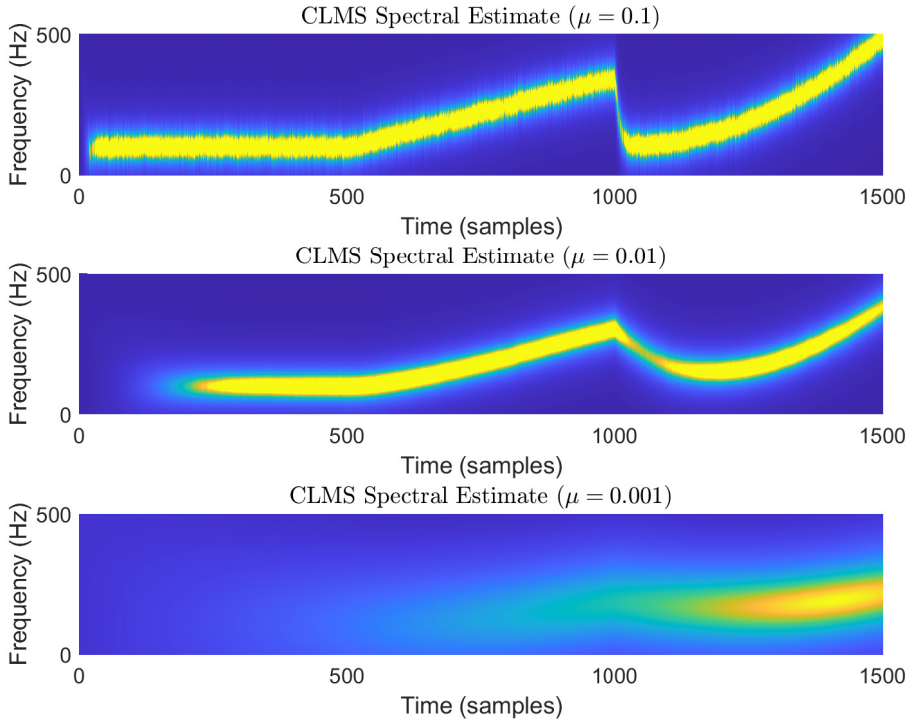


**Figure 3.7:** Time-frequency spectra of FM signal for different learning rates

## 3.3 A Real-Time Spectrum Analyser Using Least Mean Square

(a) A signal, $y(n)$, can be estimated as a linear combination of N harmonically related sinusoids given by (3.31)

$$\hat{y}(n) = \sum_{k=0}^{N-1} w(k) e^{j\frac{2\pi kn}{N}} \tag{3.31}$$

which can be expressed in vector form as in (3.32)

$$\hat{\mathbf{y}} = \mathbf{F}\mathbf{w} \tag{3.32}$$

The optimal weights, $\mathbf{w}$, can be obtained by minimizing the sum of squared errors between the estimate and true signal (3.33)

$$\min_{\mathbf{w}} ||\mathbf{y} - \hat{\mathbf{y}}||^2 = \min_{\mathbf{w}} \sum_{n=1}^{N-1} ||y(n) - \hat{y}(n)||^2 \tag{3.33}$$

This equation can be simplified to

$$\min_{\mathbf{w}} ||\mathbf{y} - \hat{\mathbf{y}}||^2 = \min_{\mathbf{w}} (\mathbf{y} - \mathbf{F}\mathbf{w})^H (\mathbf{y} - \mathbf{F}\mathbf{w}) \tag{3.34}$$

$$= \min_{\mathbf{w}} (\mathbf{y}^H\mathbf{y} - \mathbf{w}^H\mathbf{F}^H\mathbf{y} - \mathbf{y}^H\mathbf{F}\mathbf{w} + \mathbf{w}^H\mathbf{F}^H\mathbf{F}\mathbf{w}) \tag{3.35}$$

where since the two centre terms are scalars and each other's transposes

$$= \min_{\mathbf{w}} (\mathbf{y}^H\mathbf{y} - 2\mathbf{w}^H\mathbf{F}^H\mathbf{y} + \mathbf{w}^H\mathbf{F}^H\mathbf{F}\mathbf{w}) \tag{3.36}$$

The gradient w.r.t. $\mathbf{w}$ can be taken to obtain (3.37)

$$\nabla_{\mathbf{w}} ||\mathbf{y} - \hat{\mathbf{y}}||^2 = 0 = -2\mathbf{F}^H\mathbf{y} + 2\mathbf{F}^H\mathbf{F}\mathbf{w} \tag{3.37}$$

which can be simplified to the equation of the optimal weights in the least-squares sense in (3.38)

$$\mathbf{w} = (\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H\mathbf{y}. \tag{3.38}$$

Since the relationship in (3.31) analogous to the inverse DFT, $\mathbf{w}$ can be interpreted as DFT coefficients.

(b) Given that $\mathbf{F}$ is multiplied by $\mathbf{w}$ to get $\hat{\mathbf{y}}$, this estimate must be in the subspace spanned by the columns of $\mathbf{F}$, (basis vectors), denoted as its column space C($\mathbf{F}$). Therefore, this estimate can be seen as the projection of $\mathbf{y}$ onto C($\mathbf{F}$) that minimizes the sum of squared errors. This is further highlighted by the projection matrix, $\mathbf{P}$, obtained by substituting (3.38) into (3.32), resulting in (3.39)

$$\begin{aligned} \mathbf{P} &= \mathbf{F}(\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H \\ \hat{\mathbf{y}} &= \mathbf{P}\mathbf{y} \end{aligned} \tag{3.39}$$

(c) Just as in 3.2, by treating DFT as a least squares solution allows us to implement an adaptive version with CLMS, where the input to the filter at time $n$ is given by (3.40)

$$\mathbf{x}(n) = \frac{1}{N}\left[1, e^{j\frac{2n\pi}{N}}, e^{j\frac{4n\pi}{N}}, ..., e^{j\frac{2n\pi(N-1)}{N}}\right] \tag{3.40}$$

This allows time-frequency estimation as with the previous AR estimates. The CLMS algorith created was adapted into a DFT-CLMS algorithm and was used for the FM signal in 3.2 b), where results are shown in Figure 3.8.

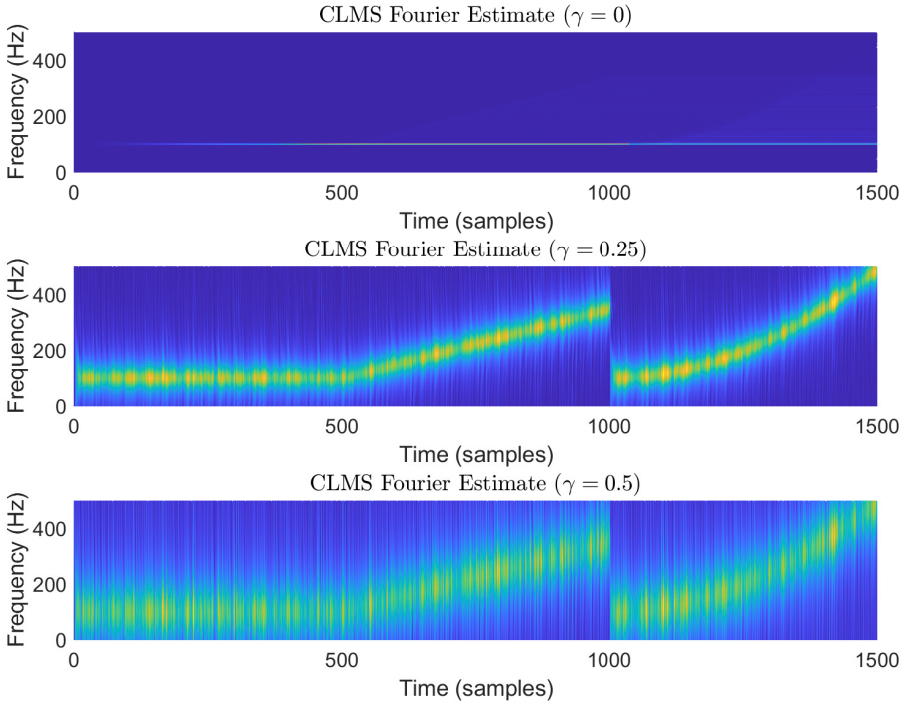


**Figure 3.8:** DFT-CLMS time-frequency diagram estimates for different $\gamma$

The standard DFT-CLMS is unable to capture the changes in frequency over time as it does not learn rapidly enough, resulting in only capturing the information from the first 500 samples. By introducing a leak parameter, $\gamma$, the algorithm is able to forget what it has learned and can robustly capture the frequency information of the signal over time. The $\gamma$ parameter must be chosen to obtain fast updates for frequency estimation, without distorting the estimate, as shown for $\gamma = 0.5$.

(d) The DFT-CLMS algorithm from (c) was then implemented for the EEG signal `POz` used in Part 1.2. To reduce computational complexity, the range $n \in [3000 : 1 : 4199]$. Given that the signal is stationary, introducing a $\gamma$ parameter is not necessary. The sampling rate used was $f_s = 1200Hz$ and the signal was detrended before the estimates. The time-frequency diagram obtained is shown in Figure 3.9. After approximately 400 samples of learning, the 50Hz component introduced by mains and the low frequency component at 3Hz become visible and exhibit a strong response by the end of the segment. Another frequency component exhibits a weak response near 9Hz, which may become more apparent given more training data. While this estimate results in less noisy spectral estimates, it exhibits greater computational complexity and requires a substantial number of observations to learn the optimal parameters.

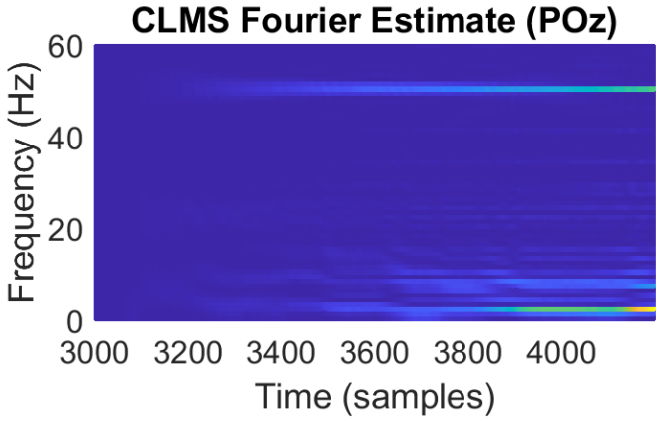


**Figure 3.9:** DFT-CLMS time-frequency diagram estimate for `POz` segment

# 4 From LMS to Deep Learning

## 4.1 LMS Time-Series Prediction

The first task was to apply the previously constructed LMS algorithm to a time-dependent, nonstationary time-series from 'time-series.mat'. The mean of the series was removed and the LMS algorithm was applied with AR(4) and a step size $\mu = 1 \times 10^{-5}$. The prediction of the mean-centered signal is displayed in Figure 4.1. This figure reflects how after the first 200-300 samples, the algorithm has converged and is able to predict the next step with $MSE = 40.109$ and $R_p = 10log_{10}\left(\frac{\hat{\sigma_y^2}}{\hat{\sigma_e^2}}\right) = 5.195$, where $\hat{\sigma_y^2}$ and $\hat{\sigma_e^2}$ are the variances of the output and prediction error respectively. While this algorithm performs relatively well for a linear model, it is unable to capture non-linearities in the signal.

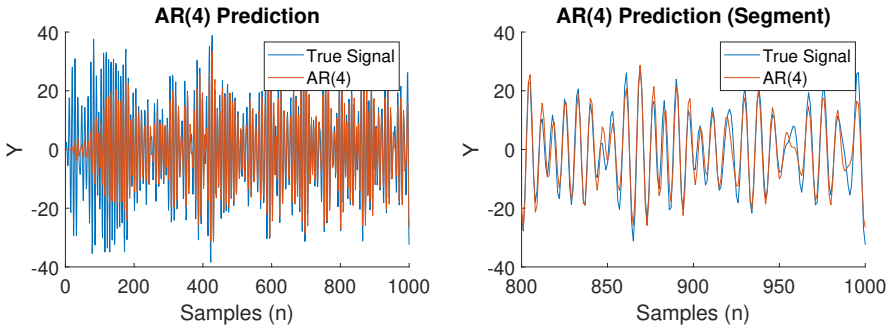


**Figure 4.1:** LMS next-step prediction of mean-centered nonstationary signal

## 4.2 Dynamical Perceptron

To enable the standard LMS algorithm to capture non-linear patterns in the signal, a non-linearity (activation function) can be added to the output. For this task, the `tanh` function was used for modelling the time-series from (4.1). Given the introduced non-linearity, the cost function has changed and, consequently, the weight updates. Given that $\tanh'(x) = 1 - \tanh^2(x)$, the weight update follows (4.1)

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)(1 - \tanh^2(\mathbf{w}(n)^T\mathbf{x}(n)))\mathbf{x}(n) \tag{4.1}$$

The zero-mean signal was predicted with the LMS-Tanh algorithm, a dynamical perceptron, with results displayed in Figure 4.2. While able to capture non-linearities, since $\tanh(x) \in [-1, 1]$, the model is not able to accurately predict the signal to its magnitude, with $MSE = 208.083$ and $R_p = -30.643$.

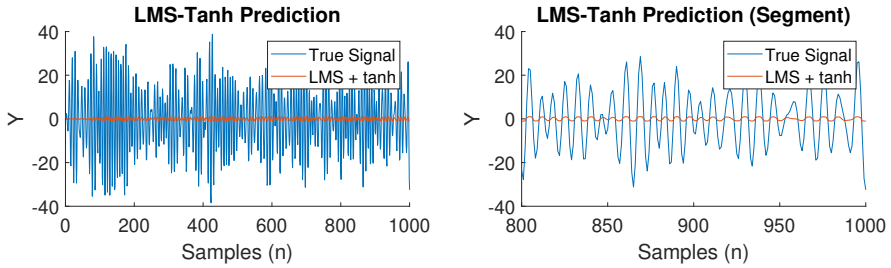


**Figure 4.2:** LMS-Tanh next-step prediction of mean-centered, nonstationary signal

## 4.3 Scaled Activation Function

In order to improve the dynamical perceptron, the activation function can be scaled (i.e. $a.\texttt{tanh}$). Given that $\tanh(x) \in [-1, 1]$, the scaling factor should be large enough to capture the highest amplitude peaks, yet low enough to prevent instability. Therefore, the optimal scaling factor, $a$, is in the range $25 \leq a \leq 40$. The optimal scaling factor, $a^*$, can be obtained by using gradient descent just as for the weights, with update given by (4.2), under the assumption that it is constant over time. Given the change in activation function, the weight update also changes to (4.3).

$$a = a + \mu e(n)\tanh(\mathbf{w}(n)^T\mathbf{x}(n)) \tag{4.2}$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu a e(n)(1 - \tanh^2(\mathbf{w}(n)^T\mathbf{x}(n)))\mathbf{x}(n) \tag{4.3}$$

The learning rates for the weights and scaling factor were chosen to be $\mu_{\mathbf{w}} = 2 \times 10^{-7}$ and $\mu_a = 0.1$ respectively. The scaling factor was initialized at $a_{init} = 30$, within the optimal range discussed. The results are displayed in Figure 4.3. The scaled LMS-Tanh algorithm significantly outperforms its non-scaled and linear counterparts, with $MSE = 10.981$ and $R_p = 12.612$.

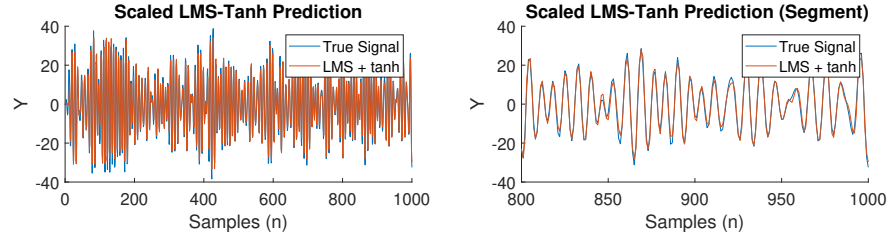


**Figure 4.3:** Scaled LMS-Tanh next-step prediction of mean-centered non-stationary signal

## 4.4 Dynamical Perceptron & Bias

The scaled LMS-Tanh algorithm can be improved with the addition of a bias term such that the output of the model is described as $\tanh(\mathbf{w}^T\mathbf{x} + b)$. This can be implemented with minimal modifications by considering the augmented input $\tilde{\mathbf{x}} = [1, \mathbf{x}^T]^T$, such that the weight always multiplied by 1 represents the bias. This updated algorithm was applied to the original signal and the results are displayed in Figure 4.4. The algorithm performs worse than the prediction of the mean-centered series, given that error is introduced during the period before the bias is fully learned, with $MSE = 17.976$ and $R_p = 10.940$. However, post-convergence, the algorithm exhibits similar performance to the unbiased prediction.

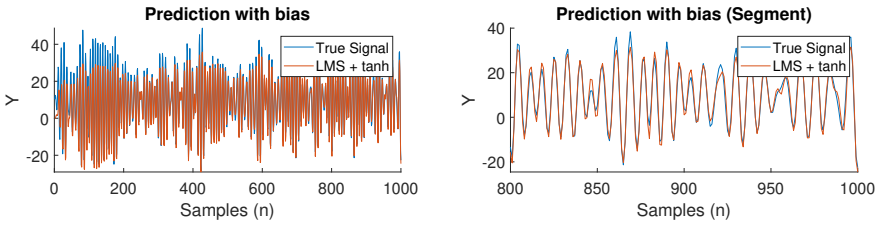


**Figure 4.4:** Scaled LMS-Tanh next-step prediction of original nonstationary signal

## 4.5 Pre-training Weights

Given that single weight updates are carried out per time-step, it may take a lot of samples to converge to a reasonable prediction, and with the addition of the scaling factor and bias terms, may contribute significantly to the error. To prevent this, these parameters can pre-trained by over-fitting to a small number of samples for a more favorable initial condition. Pre-training was carried out for the scaled LMS-Tanh algorithm with a bias for the first 20 samples and 100 epochs, where $\mathbf{w}(0) = \mathbf{0}$. The prediction results are shown in Figure 4.5. As expected, pre-training the parameters prior to prediction resulted in optimal performance, with $MSE = 5.337$ and $R_p = 16.1145$, exhibiting near-ideal predictions from the start of the signal.

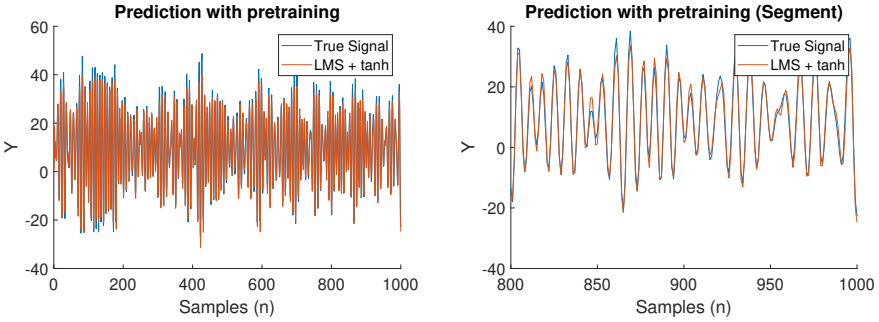


**Figure 4.5:** Scaled LMS-Tanh next-step prediction of original nonstationary signal (with pre-training)

## 4.6 The Backpropagation Algorithm

The backpropagation algorithm, short for backward propagation of errors, is a widely used algorithm for training deep neural networks. Backpropagation refers to computing the gradient in weight space with respect to some loss function, such as sum of squared errors (regression) or cross-entropy (classification). Practically, the algorithm accounts for activation functions when computing the gradients and determines the proportion of the error each neuron in the previous layer is responsible for. This is used to update weights (larger error accountability results in larger updates), and to propagate the error from each neuron to its connections in the layer prior to it. This process is carried out recursively until the input layer is reached, updating all the weights in the deep network along the way.

## 4.7 Deep Neural Network

A non-linear time-series was generated based on 10 sinusoids of different frequencies, phases and amplitudes and noise, as shown in Figure 4.6. Noise was added to the time-series with power equal to 0.05. The time-series was divided into a 50% train-test split and the linear LMS algorithm, the dynamical perceptron using `tanh`, and a deep neural network were trained on the first half of the data and were tested on the remaining samples. The deep network was trained on 20,000 epochs, had 4 hidden layers (10,5,5,5,1), and used a learning rate of 0.01. The predictions are displayed in Figure 4.7, reflecting the deep networks ability to capture more temporal information and provide more accurate predictions, as opposed to the single neurons, whose performance is dictated by the largest amplitude frequency components of the signal. The training and test losses were obtained and computed as a function of epoch number in Figure 4.8, where both the single perceptron and LMS exhibit converge more rapidly than the deep network, whereas all three algorithms exhibit similar test loss to the deep network (approximately 0.08). This is likely to the noise introduced to the signal, where the deep network's complexity enables the model to fit the signal as well as noise.

## 4.8 DNNs & Noise

Finally the above procedure was repeated for different values of noise power. While for lower noise powers, the deep network outperforms the other two algorithms due to its greater complexity, it over-fits the training data for greater noise powers, fitting the noise as well as the signal to minimize loss. This highlights a major limitation of deep networks, as it may only exhibit optimal performance on relatively clean signals. This is supported by the loss curves in Figures 4.9 and 4.10. While for low noise (power of 0), the deep network significantly outperforms the other two algorithms given sufficient training, for high noise (power of 1), the LMS and dynamical perceptron algorithms outperform the the deep network. While all three models over-fit given the noise in the signal, due to the deep network's complexity, it exhibits the worst generalization. Additionally, both the training and testing curves for the deep network exhibit instability, reflecting another issue of such a complex system being applied to noisy signals.

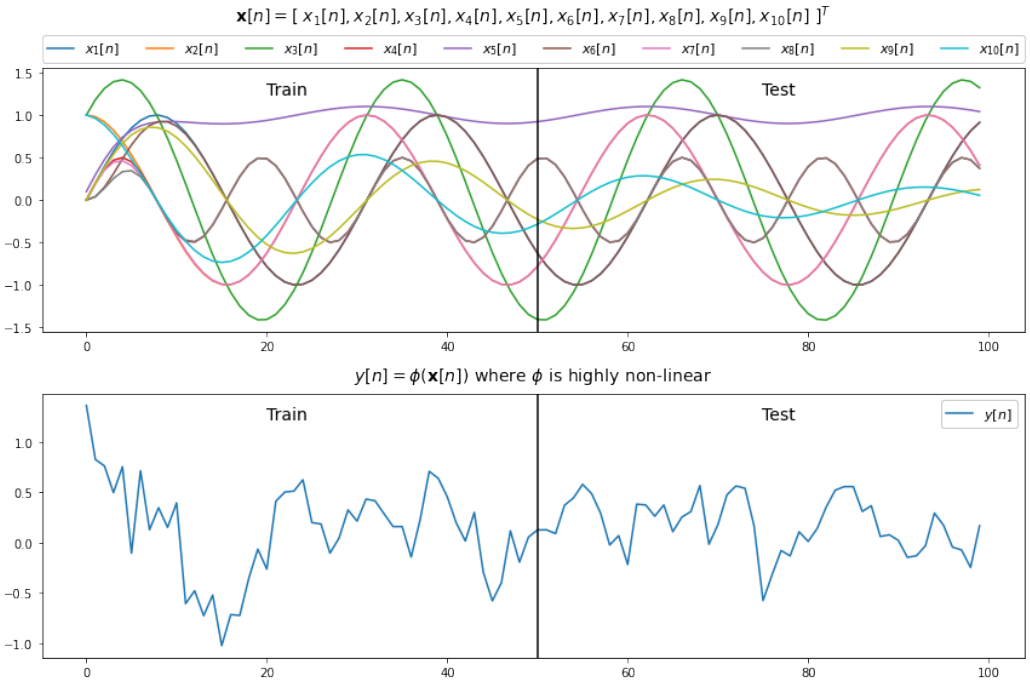


**Figure 4.6:** Sinusoids superposed to obtain non-linear time-series

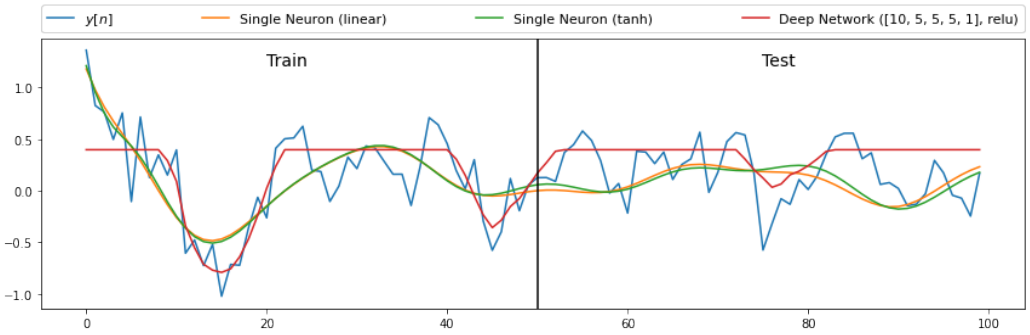


**Figure 4.7:** LMS, dynamical perceptron and deep network predictions of low noise, non-linear time-series

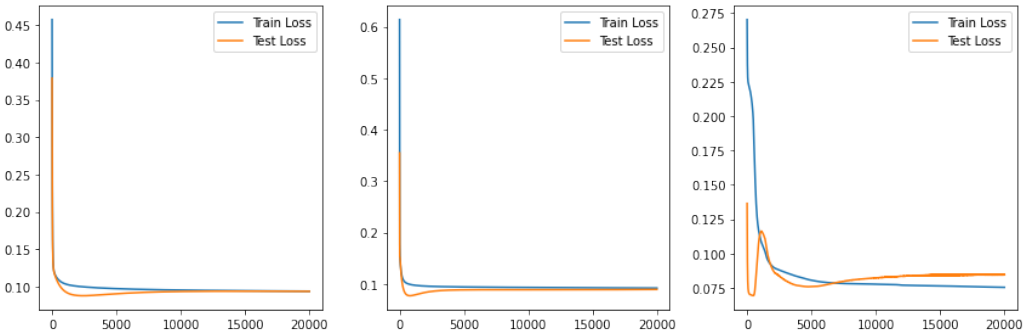


**Figure 4.8:** Training and testing loss curves for the standard LMS (left), non-linear perceptron (middle) and deep network (right) as a function of epoch count for 0.05 noise power

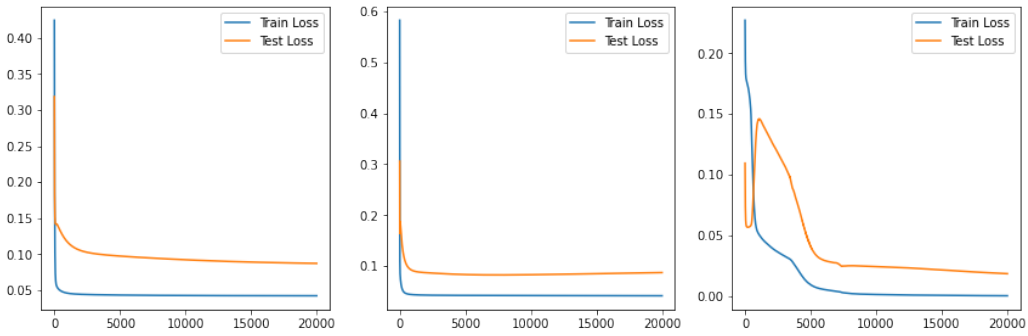


**Figure 4.9:** Training and testing loss curves for the standard LMS (left), non-linear perceptron (middle) and deep network (right) as a function of epoch count for 0 noise power

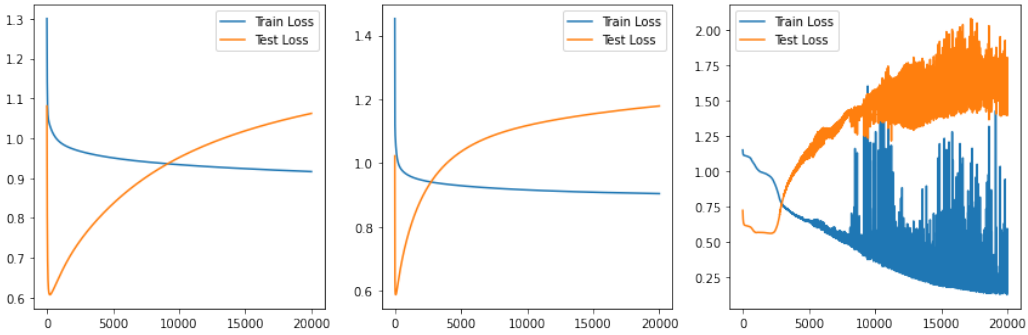


**Figure 4.10:** Training and testing loss curves for the standard LMS (left), non-linear perceptron (middle) and deep network (right) as a function of epoch count for a noise power of 1

# 5 Appendix: Code

For the MATLAB scripts/functions used and Jupyter notebooks for this coursework, visit
`https://github.com/joao-binenbojm/Adaptive-Signal-Processing-Machine-Intelligence.git`