Imperial College London

ADAPTIVE SIGNAL PROCESSING
AND
MACHINE INTELLIGENCE

Assignment Report

Matilde Piccoli
01764158

April 5, 2023

IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

Contents

1 Classical and Modern Spectrum Estimation

1.1 Properties of Power Spectral Density

Assuming the covariance sequence $r(k)$ decays rapidly (1), we will demonstrate that the two definitions of PSD (2) are equivalent, both analytically and in simulation.

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

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

The second definition of PSD 2 can be expanded and rearranged in the following way:

$$P(\omega) = \lim_{N\to\infty} E\left\{ \frac{1}{N} \sum_{n=0}^{N-1} x(n)e^{-jn\omega} \sum_{n=0}^{N-1} x^*(m)e^{jn\omega} \right\} \tag{3}$$

$$= \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} \sum_{n=0}^{N-1} E\left\{ x(n)x^*(m)e^{-j(n-m)\omega} \right\} \tag{4}$$

Substituting $n - m$ with k, we obtain:

$$P(\omega) = \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} \sum_{k=n}^{n-N+1} r(k)e^{-jk\omega} \Bigg\} \tag{5}$$

$$= \lim_{N\to\infty} \frac{1}{N} \sum_{k=-N+1}^{N-1} (N - |k|)r(k)e^{-jk\omega} \tag{6}$$

$$= \lim_{N\to\infty} \sum_{k=-N+1}^{N-1} r(k)e^{-jk\omega} - \lim_{N\to\infty} \frac{1}{N} \sum_{k=-N+1}^{N-1} |k|r(k)e^{-jk\omega} \tag{7}$$

Given the assumption in 1, the limit quickly converges to zero, therefore:

$$P(\omega) = \sum_{k=-\infty}^{\infty} r(k)e^{-jk\omega} \tag{8}$$

The output of the simulation is shown in figure 1. The plot of the two Power Spectral Density definitions overlaps as long as the assumption in 1 holds; therefore, for a signal with a fast-decaying ACF (e.g. impulse), the two definitions lead to equivalent results, whilst for the slow-decaying ACF (e.g. sinusoidal function) one the will differ. In the latter, in fact, the periodicity of the sinusoidal signal is characterised by an ACF that decays to zero at a slower rate than the one at which the limit approaches infinity.

1.2 Periodogram-based Methods Applied to Real–World Data

Part A Before computing the periodogram, the data were preprocessed using the Matlab functions `detrend` and `mean` to respectively eliminate the straight-line trend, reducing the effects of the low-frequency components and average value from the signal, removing the DC offset. Additionally, the logarithm of the data has been computed, and the mean value is subtracted to evaluate any alterations in the data's perception of periodicity. This greatly reduces the peak amplitude in the time domain and enables us to identify the peak periodicity in the frequency domain, with a dominant frequency at $f = 0.56Hz$ and its harmonics identifiable in Figure 2. The `periodogram` function has been used to estimate the spectral content of the signal at different stages of the preprocessing.

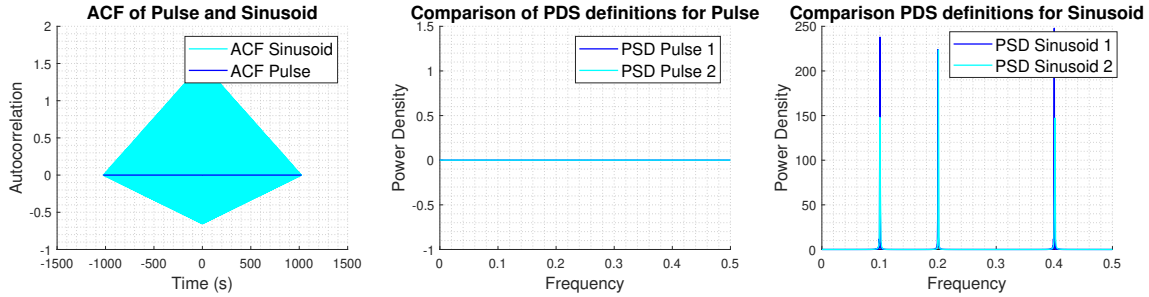

Figure 1: Autocorrelation and PSD for impulse and sinusoidal function

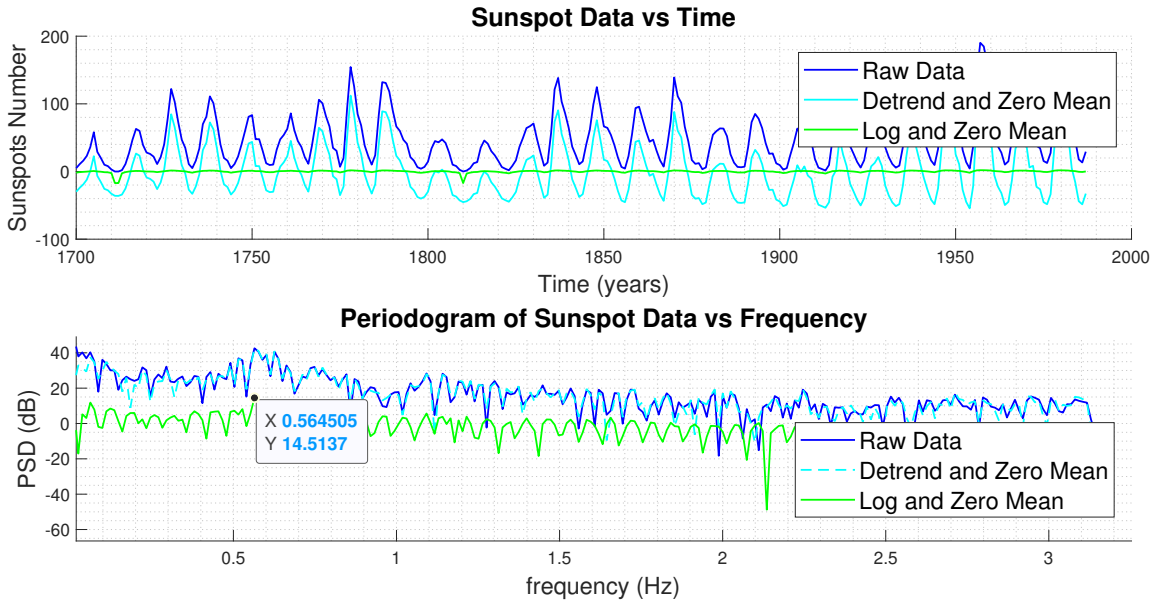

Figure 2: Sunspot time series and Periodogram

Part B The EEG data of a participant under visual stimulation is analysed to identify the SSEP (steady-state visual evoked potential). The PSD of the EEG data was computed, and the raw and windowed data were compared as shown in figure 3. Five main peaks can be identified in the frequency spectrum of the EEG data: one peak at 50Hz due to electrical interference, a group of peaks around 8.6Hz due to the alpha-rhythm caused by the mental state of the participant (tiredness), and the peaks at around 13.01Hz, 25.98Hz, and 39Hz, corresponding respectively to the SSVEP and its two main harmonics.
Different window lengths (1s,5s, and 10s) are plotted together with the raw data in figure 3; the comparison highlights how the window length of 10s is characterised by a decrease in variance compared to the regular non-averaged periodogram. This is because averaging helps to decrease the noise present in the signal due to the finite length of the data. On the other hand, a much smaller window length (1s) leads to an even more reduced variance, wider main lobes, making it harder to identify the peak frequencies, and an overall lower resolution as the PSD is computed on shorter segments of the signal.

1.3 Correlation Estimation

Part A The correlogram of different functions (White Gaussian Noise (WGN), noisy sinusoid and filtered WGN) was computed using both biased and unbiased spectral estimators as defined in 9, 10.

$$\textbf{Biased:} \hat{r}(k) = \frac{1}{N} \sum_{n=k+1}^{N} x(n)x*(n-k) \tag{9}$$

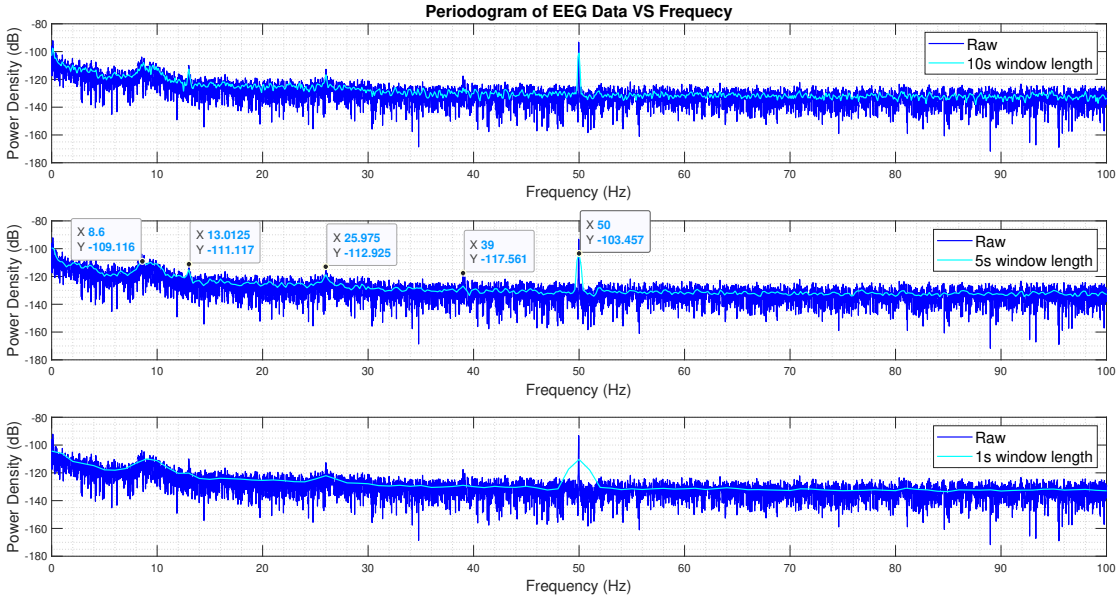

Figure 3: Periodogram of EEG Data with windows length of 1s,5s,10s

$$\textbf{Unbiased:} \hat{r}(k) = \frac{1}{N} \sum_{n=k+1}^{N} x(n)x*(n-k) \text{with } (0 \leq k \leq N-1) \tag{10}$$

As it can be observed from figure 4, the ACFs of all signals estimated using the unbiased method are characterised by negative values at high large values of k, leading to negative values for the PSD as a lower number of samples is available. Instead, the biased method assigns progressively lower weighting at higher values of k, leading to ACF values tending to 0 when k tends to infinity (asymptotically unbiased). These observations can be traced back to the physical mean of the two estimators: using the unbiased estimator corresponds to windowing the function with a rectangular window, characterised by both positive and negative values ($sin(x)/x = sinc$ function in the frequency domain); as described in section 1.1, this does not rapidly decrease, therefore the assumption 1 does not hold for high k close to N.
Conversely, the biased method corresponds to a triangular window, characterised by only positive-valued lobes in the frequency domain and leading to a real-valued ACF. Since both the triangular window and the ACF are odd-length symmetrical (symmetrical with respect to their midpoint), the PSD will result in a real-valued spectrum as well. Therefore, the biased method, even if less close to the true mean of PSD compared to the unbiased one, conserves the function's physical meaning with a low number of samples and maintains lower variance.

Part B and C PSD estimate of 200 realisations of a random process were generated, and their spectrum, mean and standard deviation were plotted on a linear (Figure 5 (top graphs)) and logarithmic scale (Figure 5 (bottom graphs)). The signal was computed as:

$$\texttt{noisy sinusoid} = sin(2*\pi*0.1*t) + 3*sin(2*\pi*0.3*t) + 5*sin(2*\pi*0.5*t) + randn(1,N); \tag{11}$$

`randn(1,N)` being the additive WGN with normal distribution.
The fundamental frequencies of the sinusoid are plotted on the linear scale. Having used the biased estimator, the spectrum is real-valued and asymptotically stable. However, the standard deviation is proportionally greater around the frequency of interest, as it is proportional to the square root of the amplitude. Independently of the number of samples, even if averaging across different periodograms can help, this does not necessarily cancel out the noise and inconsistency of each signal or the amplitude dependency.
When displaying the PSD in decibels, this compresses the scale and allows for a more uniform display of the PSD over the entire range, making it easier to see the relative contributions of different frequency components while smoothening the overall spectrum. In this case, in fact, the fundamental frequencies are identified in the lowest values of the standard deviation spectrum.

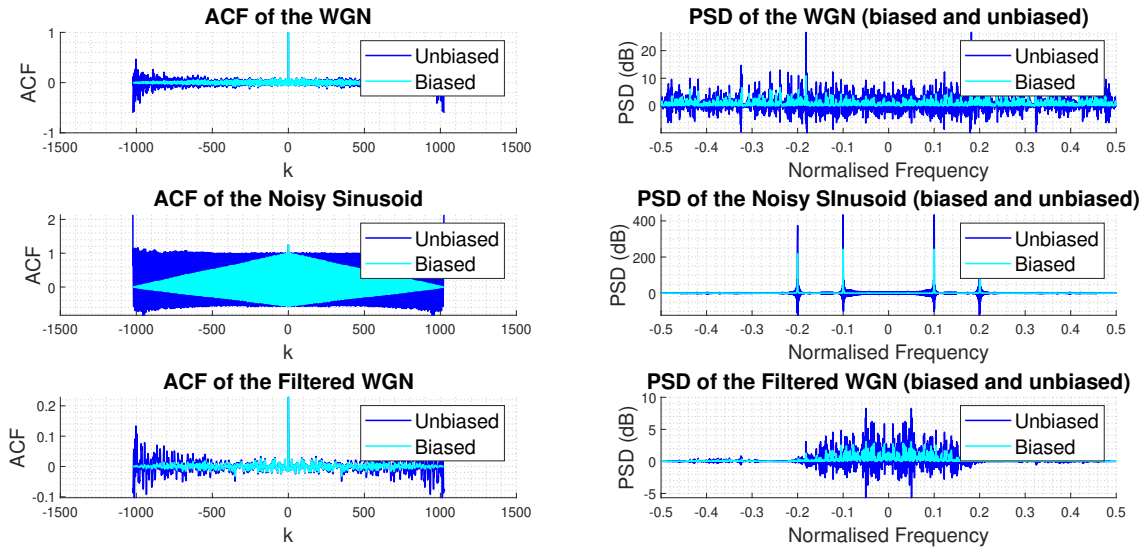

Figure 4: Unbiased and Biased PSD Estimation for White Gaussian Noise (WGN), noisy sinusoid and filtered WGN.

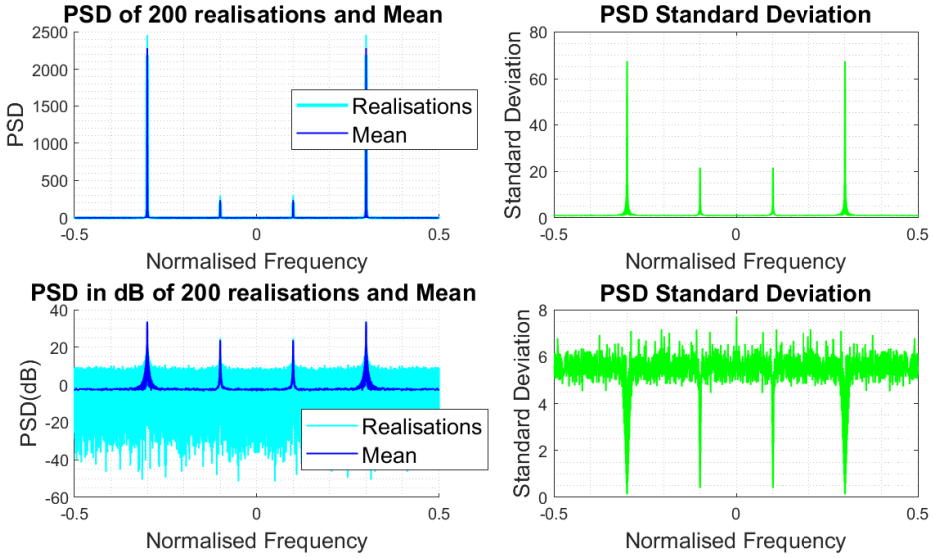

Figure 5: PSD estimate of 200 realisations of a random process, mean and standard deviation; linear scale (top) and logarithmic (bottom)

Part D Figure 6 shows the periodogram computed for a noisy sinusoidal signal sampled at different rates (N = 30,90,210). The signal has been defined in Matlab as:

```
noise = 0.2/sqrt(2)*(randn(size(n))+1j*randn(size(n)));
  x = exp(1j*2*pi*0.3*n)+exp(1j*2*pi*0.32*n)+ noise;
```

From the plot in figure 6, it is clearly noticeable how an increased number of samples eases the identification of the fundamental frequency of the signal against the additive noise. However, with a lower number of samples ($N < 90$), we can only identify a single peak as the frequency resolution is not accurate enough compared to the half of the distance between the peaks ($f_{resolution} = f_s/N = 1/90 = 0.011 > 0.01$)

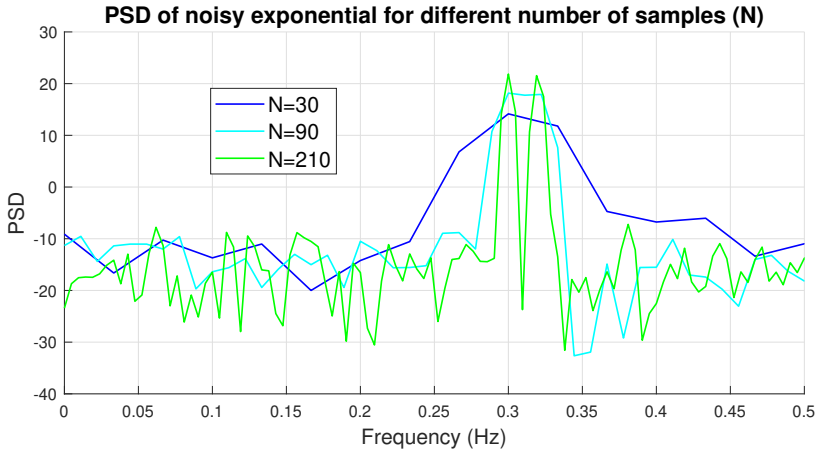

Figure 6: Periodogram of a noisy sinusoidal signal with different number of samples (N)

Part E The MUSIC method was used to generate a pseudo-spectrum of the same signal described in Part D, to ease the comparison of different PSD methodologies. The following code was used to generate the spectrum in Matlab:

```
[X,R] = corrmtx(x,14,'mod'); [S,F] = pmusic(R,2,[ ],1,'corr');
```

The spectrogram in figure 7 show the estimated PSD using the MUSIC method. Similarly to the effect of changing the number of samples described in the previous sections, it has been noticed that with increased order complexity (p), the estimate is able to distinguish the two distinct peaks. In particular, with an order of 2, the two peaks are already clearly distinguishable at 0.3 and 0.32Hz, meaning that we need to be aware of the number of peaks to choose the order that will better represent the signal. Although this method has a greater standard deviation compared to the standard periodogram, by estimating the mean of the realisations, we obtain a better estimate of the true signal and its magnitude.

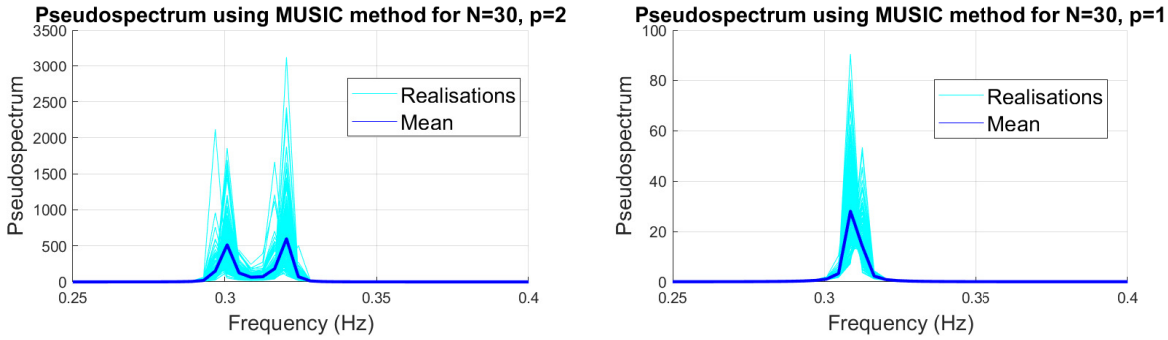

Figure 7: PSD estimate in dB of 200 realisations of a random process and its mean for different orders p.

1.4 Spectrum of Autoregressive Processes

Part A When using the unbiased ACF, the estimate can become more inconsistent with increased lag size due to the reduced number of samples available. This issue, together with the use of non-stationary data, can lead to inaccurate estimates of the AR coefficients and lead to unstable processes. To avoid the dependency on the lag size, a different strategy can be implemented, such as the use of overlapping windows or modified ACF estimates like the Yule-Walker or Burg methods.

Part B Figure 8 (left) shows the PSD estimation using different model orders with 1000 samples. As expected, due to the low number of samples (under-sampling), we need a relatively complex model in order to be able to detect the two peaks in the spectrum. In fact, with $p < 10$ the AR estimate only represents the strongest peak in the original signal.

Part C As shown in figure 8 (right), increasing the number of samples available offers the possibility for a lower-order system to represent the multiple peaks in the data. In particular, with 10000 samples, an order of 4 is now sufficient to represent the two peaks in the signal. It is redundant to increase the order further as this does not benefit the estimation whilst increasing the computation complexity; however, below $p = 4$ the estimate is once again under-sampled with respect to the complexity of the model, and therefore, we lose estimation accuracy.

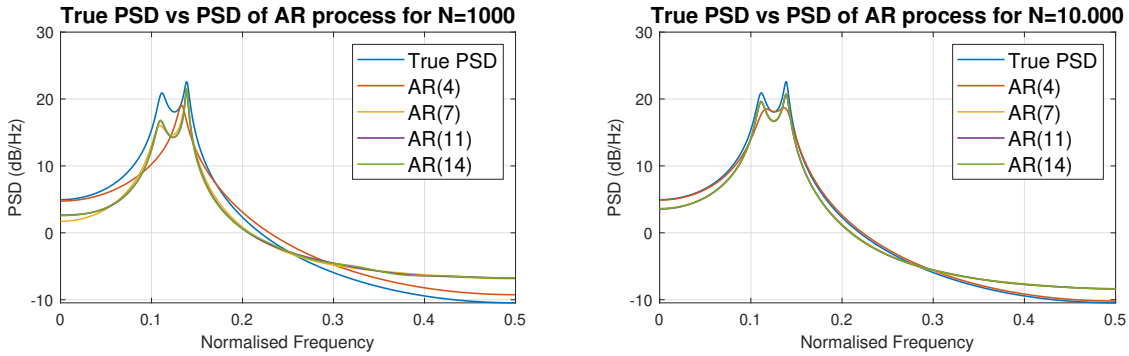

Figure 8: Plot of the true PSD and PSD from autoregressive process (AR) of difference orders (4,7,11,14) for sample number of 1000 and 10.000.

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

Part A The standard and averaged periodogram with different window lengths were applied to the ECG data after separating the three R-R Intervals (RRI) and selecting the part of the signal less affected by noise and artefacts. The results are shown in figure 9, using a window length of 50s and 300s; as expected, the wider window length allows for capturing greater complexity of the frequency spectrum, and some peaks become identifiable, although the variance of the data is larger.

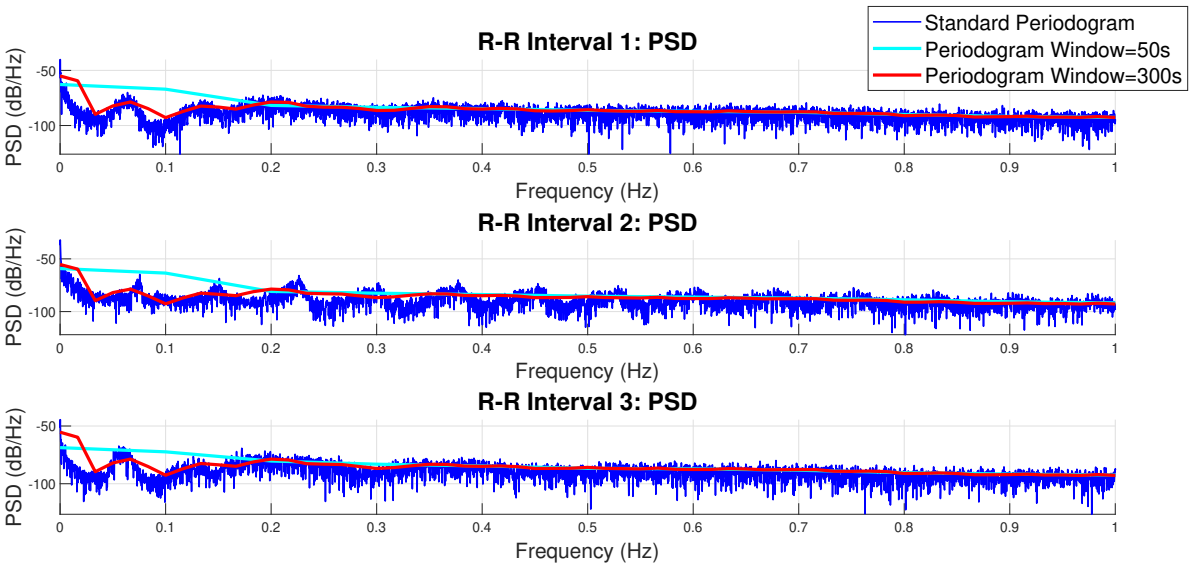

Figure 9: Spectrum of ECG data with standard and averaged periodogram of different window lengths

Part B For the different R-R intervals, we are expecting peaks in different parts of the spectrum, corresponding to the breathing frequency at which the participant was asked to breathe. The breathing paces are evaluated to be approximately 12-16 breaths per minute when breathing normally, 25 breaths per minute when breathing faster and 7.5 breaths per minute when breathing more slowly; these correspond to $f = 14/60s = 0.23Hz$ for RRI1, $f = 0.42Hz$ for RRI2 and $f = 0.125Hz$ for RRI3. Although, as expected, we can identify a peak around 0.2Hz in the spectrogram

of RRI1, the clear breathing frequency is hardly distinguishable due to the high level of noise and movement artefacts in the recordings.

Part C The AR spectrum estimation is displayed in figure 10; due to noisy data, it is hard to distinguish the peaks corresponding to the breathing frequency in the signal; this could be also caused by the multiplicity of the peaks present across the frequency spectrum, making it harder to isolate the informative variation of amplitude against the effects of noise. However, different model orders were tested, and the AR(10) was found to be the most suitable, considering the trade-off between complexity and degree of freedom of representation.

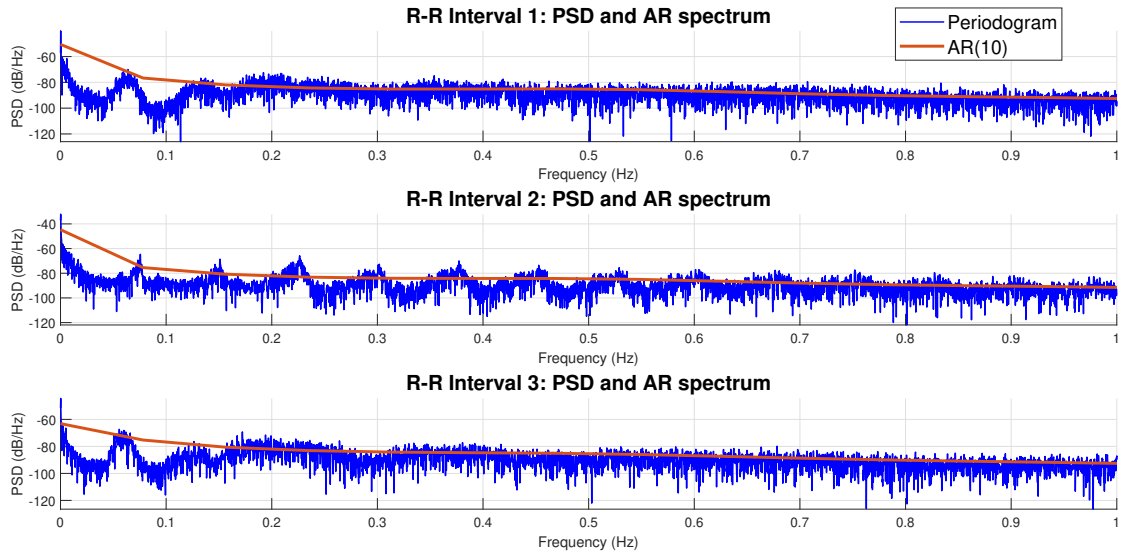

Figure 10: Spectrum of ECG data with standard periodogram and processed using AR(10)

1.6 Robust Regression

Part A We use Singular Value Decomposition (SVD) to obtain the singular values of the input data and its noisy version (zero mean Gaussian noise). Figure 11 shows the singular values of the $\mathbf{X}$ and $\mathbf{X}_{noise}$ and the square error between the two. It is clearly observable that the input data has a rank of 3 and that beyond that singular value index, the Square Error increases evidently. The performance of retrieval of the input data is dependent on the Signal-to-Noise Ratio (SNR) and, therefore, on the noise variance size with respect to the magnitude of the signal. When the magnitude of the singular values drops abruptly, it becomes, in fact, hard to distinguish the real value of the signal against the noise. If the noise variance was even greater, leading to an increase in the magnitude of the components outside the rank, this would have made it impossible to distinguish the rank of the matrix.

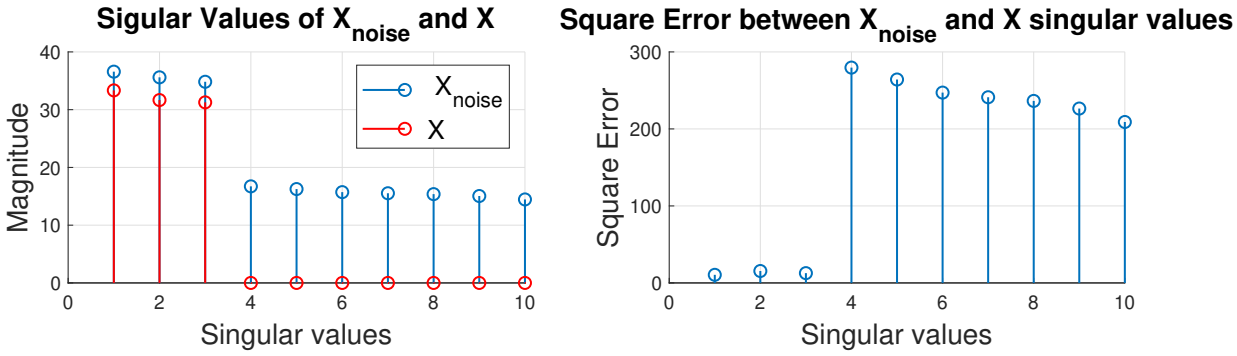

Figure 11: Singular values and Square Error between the $\mathbf{X}$ and $\mathbf{X}_{noise}$ matrices.

Part B We create a low-rank approximation of the matrix $\mathbf{X}_{noise}$ using only the 3 most significant principal components. As shown in figure 12, the error of the noisy matrices with respect to the true input data is consistently reduced by the denoising process. This is expected as the 3 most significant singular values, as shown in the previous section, were characterised by lower squared error whilst also containing enough information about the input data to retrieve the original signal. However, disregarding the other singular components and due to the non-negligible SNR of the other r most significant ones causes the denoise signal to retain a non-zero error with respect to the original data.

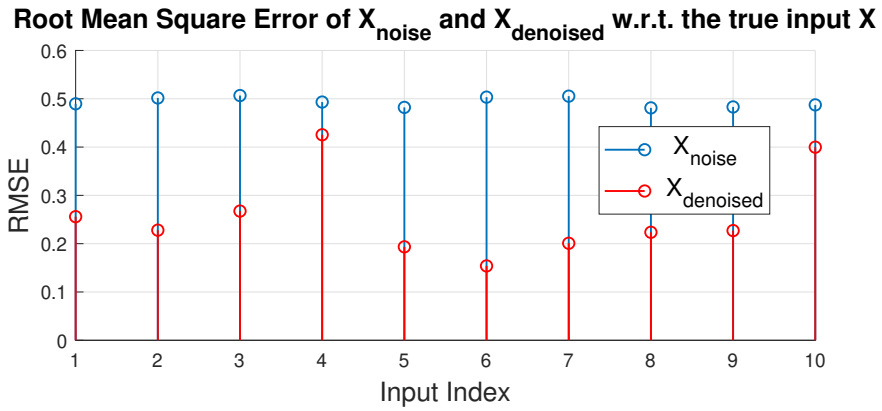

Figure 12: Difference between the noisy ($\mathbf{X}_{noise}$) and denoised ($\mathbf{X}_{denoised}$) matrices with respect to the true signal $\mathbf{X}$

Part C The parameter matrix $\mathbf{B}$ has been computed using Ordinary Least Square (OLS) and Principal Component Regression (PCR), and the MSE between the estimated ($\hat{\mathbf{Y}} = \mathbf{X}\hat{\mathbf{B}}$) and real value of the output $\mathbf{Y}$ has been computed for test and training data (see table 1 and figure 13). We found that the OLS performs better in the training data (lower MSE); however, it does not generalise as well as the PCR solution in the test set. This could be an indication of the OLS overfitting to the noise components of the training data.

Estimator	Total Training RMSE	Total Test RMSE
OLS	$4.1425e^3$	$3.2956e^3$
PCR	$4.1580e^3$	$3.2748e^3$

Table 1: Comparison between the Estimation MSE for the Training and Test sets using OSL and PCR

Part D Similarly to the previous section, we estimate the MSE of the estimation using OLS and PCR (table 2 and figure 14). In this case, we test the two methods over 100 iterations using the function `regval` to obtain the true and estimated output. As expected, the OLS estimation leads once again to a bigger MSE in the test set.

Estimator	Total MSE
OLS	$2.3414e^3$
PCR	$2.3326e^3$

Table 2: Comparison between the Estimation MSE for the Test sets using OSL and PCR over 100 iterations

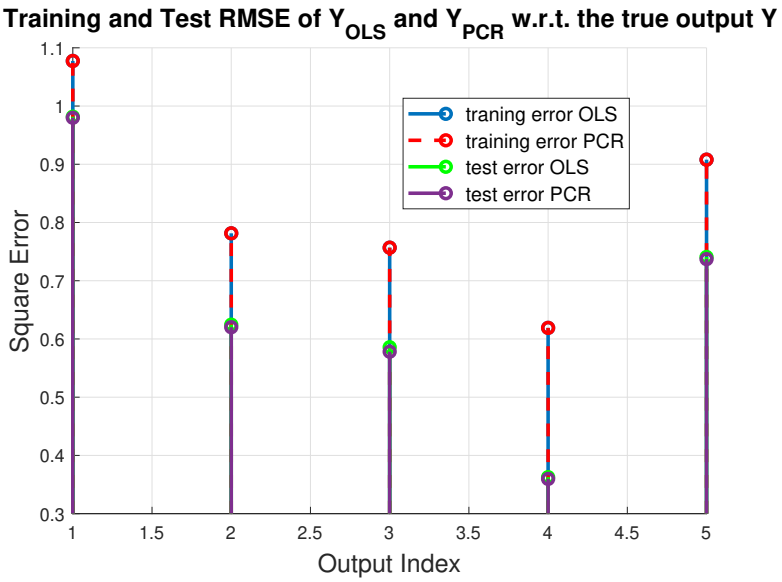

Figure 13: Comparison between test and training estimates $\hat{\mathbf{Y}}$, and the true output $\mathbf{Y}$

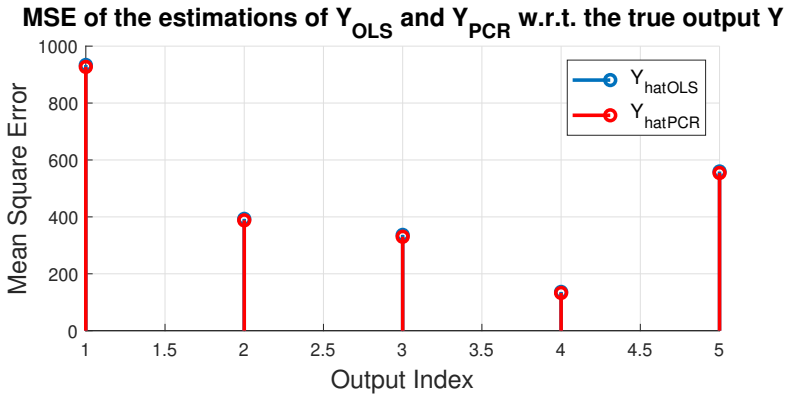

Figure 14: MSE of the estimates $\hat{\mathbf{Y}}$ with respect to the true output $\mathbf{Y}$ over 100 iterations

2 Adaptive signal processing

2.1 Least Mean Square (LMS) Algorithm

Part A For a second-order auto-regressive process satisfying the differential equation $x(n) = a_1x(n-1)+a_2x(n-2)+\eta(n)$ with $\eta(n) \sim N(0,\sigma_\eta^2)$, $\sigma_\eta^2 = 0.25$, and $a = [0.1, 0.8]$ we can find the autocorrelation matrix of the input vector $x(n) = [x(n-1), x(n-2)]^T$ as:

$$R_{xx} = E\{[x(n-1), x(n-2)]^T[x(n-1), x(n-2)]\} = E\begin{bmatrix} x(n-1)x(n-1) & x(n-1)x(n-2) \\ x(n-2)x(n-1) & x(n-2)x(n-2) \end{bmatrix} = \begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix} \tag{12}$$

With $r(k) = Ex(n)x(n-k)$ being the autocorrelation function and computed on x(n) as:

$$r(k) = Ea_1x(n-1)x(n-k) + Ea_2x(n_2)x(n-k) + E\eta(n)x(n-k) \tag{13}$$

Noting that $E\eta(n)x(n-k) = 0$ from k¿0, we can obtain r(0) and r(1) from the simultaneous equations as:

$$\begin{aligned} r(0) &= a_1r(1) + a_2r(2) + \sigma_\eta^2 \\ r(1) &= a_1r(0) + a_2r(1) + \sigma_\eta^2 \\ r(2) &= a_1r(1) + a_2r(0) + \sigma_\eta^2 \end{aligned} \tag{14}$$

As a consequence, the autocorrelation matrix R_{xx} will be:

$$R_{xx} = \begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix} = \begin{bmatrix} 0.926 & 0.463 \\ 0.926 & 0.463 \end{bmatrix} \tag{15}$$

If we estimate $\hat{x(n)}, \hat{a_1}$ and $\hat{a_2}$ using a Least Mean Square (LMS) algorithm with step size μ, its convergence will depend on the biggest eigenvalue $\lambda_{max} = 1.389$ of the matrix R_{xx} and bounded by the sufficient condition $0 < \mu < \frac{2}{\lambda_{max}}$, giving an upper bound for the step size of 1.440.

Part B We implement an LMS adaptive predictor using $N = 1000$ samples of x(n) and step size of μ or 0.05 and 0.01. The squared prediction error in decibels for 1 realisation and the learning curve obtained from the average of 100 iterations in figure15. As expected, the error oscillated around a steady state for both step sizes as the LMS weight

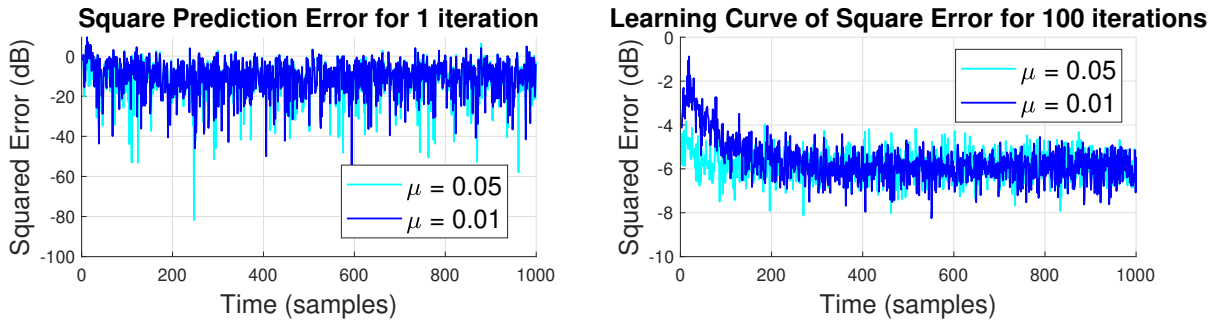

Figure 15: Squared prediction error in decibels for 1 realisation and the learning curve obtained from the average of 100 iterations

estimates fluctuate around the optimal solution; however, when averaging across numerous iterations, a noticeable difference appears: a higher step size μ leads to faster convergence, as the update is greater, but higher steady-state error over time, as the resolution of the update is lower.

Part C The misadjustments in the steady-state value of the learning curve are obtained by time-averaging over 100 independent trials the steady-state of learning curves. These values have been compared to the theoretical ones obtained using the approximated formula for small step size:

$$M_{LM} = \frac{EMSE}{\sigma_\eta^2} \approx \frac{\mu}{2} Tr\{R_{xx}\} \tag{16}$$

EMSE being the excess mean square error, i.e. the difference between the mean-square error (MSE) introduced by adaptive filters and the minimum attainable mean square error of a Wiener filter σ_η^2. As can be observed from table 3, the theoretical values of the M are close to the experimental observations $\hat{M}$, with larger misadjustments for higher a step size, similarly to what has been observed in part B.

Step Size μ	**Theoretical** M	**Experimental** $\hat{M}$
0.05	0.0463	0.0508
0.01	0.0093	0.0125

Table 3: Comparison between the experimental and theoretical values of the LMS misadjustment for different step sizes.

Part D An estimation of the steady state values of the adaptive filter coefficients w was obtained by averaging the final results obtained by over 100 independent trials and plotting with true coefficients' values in figure 16. The results were obtained using a step size of 0.05 and 0.01, confirming how a lower μ corresponds to a slower yet steadier convergence, with reduced variance and lower bias error at steady-state.

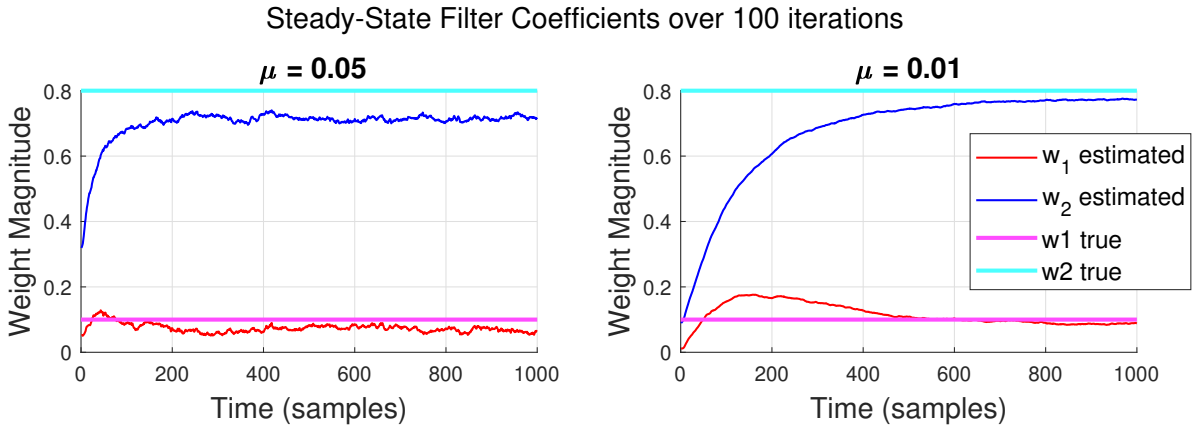

Figure 16: True and estimated coefficient w of the adaptive filter obtained averaging over 100 independent iterations.

Part E The leaky LMS equation 17 with leakage coefficient $0 < \gamma < 1$ can be obtained from minimising the cost function 18.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \nabla_w J_2(n) = (1 - \mu\gamma)\mathbf{w}(n) + \mu e(n)x(n) \tag{17}$$

$$J_2(n) = \frac{1}{2}(e^2(n) + \gamma \|\mathbf{w}(n)\|_2^2) \tag{18}$$

$$\nabla_w J_2(n) = \frac{1}{2}\partial e^2(n)\partial \mathbf{w}(n) + \frac{\partial(\mathbf{w}^T(n)\mathbf{w})}{\partial \mathbf{w}(n)} \tag{19}$$

$$= \frac{1}{2}\partial e^2(n)\partial e(n)\frac{(n)}{\partial y(n)}\frac{\partial y(n)}{\partial \mathbf{w}(n)} + \frac{\partial(\mathbf{w}^T(n)\mathbf{w})}{\partial \mathbf{w}(n)} \tag{20}$$

$$= \frac{2}{2}e(n)(-1)x(n) + \gamma \mathbf{w}(n) = -e(n)x(n) + \gamma \mathbf{w}(n) \tag{21}$$

We can then obtain the updated equation by using the formula in 17:

$$w(n+1) = \mathbf{w}(n) - \mu\gamma\mathbf{w}(n) - e(n)x(n) = (1-\mu\gamma)w(n) + \mu e(n)x(n) \tag{22}$$

Part F Figure 17 shows the effect of different μ and γ values of the estimation of the leaky LMS filter coefficients averaged across 100 realisations. Even if the aim of the leaky LMS is to avoid instability when the autocorrelation matrix has zero eigenvalues, this leads to worse convergence to the true values of the coefficients (higher error at steady state). The largest values of γ lead to the worst divergence (particularly evident in the w_2 due to its larger magnitude), despite the lower oscillation for w_1.

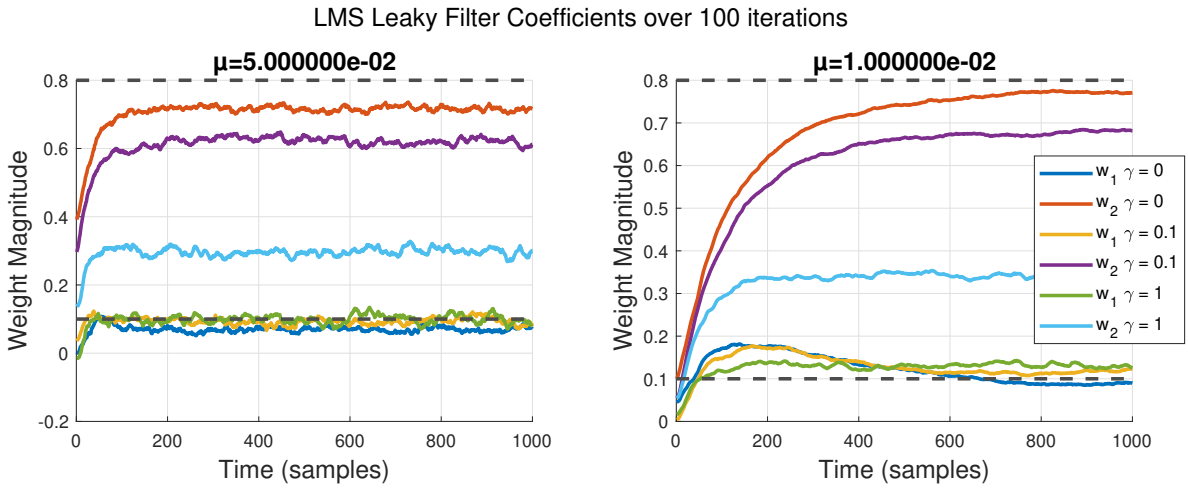

Figure 17: Leaky LMS Filter coefficient estimation from averaging 100 realisations, comparing different leakage coefficient γ values for high (left) and low (right) step sizes μ

2.2 Adaptive Step Sizes

Part A The performance of three versions of the Gradient Adaptive Step Size (GASS) algorithm learning rate update $\psi(n)$ (equations 23, 24, 25) has been compared when identifying a real-valued MA(1) system (26).

$$\text{Benveniste: } \psi(n) = [I - \mu(n-1)x(n-1)x^T(n-1)]\psi(n-1) + e(n-1)x(n-1) \tag{23}$$
$$\text{Ang \& Farhang: } \psi(n) = \alpha\psi(n-1) + e(n-1)x(n-1), 0 < \alpha < 1 \tag{24}$$
$$\text{Matthews \& Xie: } \psi(n) = e(n-1)x(n-1) \tag{25}$$
$$\text{MA(1) system: } x(n) = 0.9\eta(n-1) + \eta(n)\eta \sim N(0, 0.5) \tag{26}$$

Introducing a variable learning rate helps overcome the trade-off between speed of convergence and steady-state error, as discussed before; the optimal behaviour would be achieved with a large learning rate at the start and an increasingly smaller one approaching steady-state. As expected, figures 18 shows how GASS overall outperforms the LMS using fixed step size for 1 and 100 realisations; in particular, a very small fixed step size ($\mu = 0.01$) has the slowest convergence overall, whilst all three GASS have convergence speed in a similar order of magnitude to the one using larger fixed step size ($\mu = 0.1$), expect for Matthews&Xie.
Beneviste and Ang&Farhang both converge quicker and achieve a lower steady-state error, although the first one overshoots before settling on the correct steady-state value for weight w_0. This is given by the fact that both Beneviste and Ang&Farhang have knowledge of the previous value of the learning rate update, making them more perceptive to almost-instantaneous changes, whilst Matthews&Xie, despite being slower, has also a lower complexity of update

overall as it only takes into considerations the previous input and error. The lower complexity of this model leads to worse performance than even one of the models solely using a large fixed-step update.
In general, Beneviste and Ang&Farhang achieve the best performance, although being limited by the trade-off between variance, bias and speed of convergence on one side (better for the first one), and computational complexity on the other side (lower for the second).

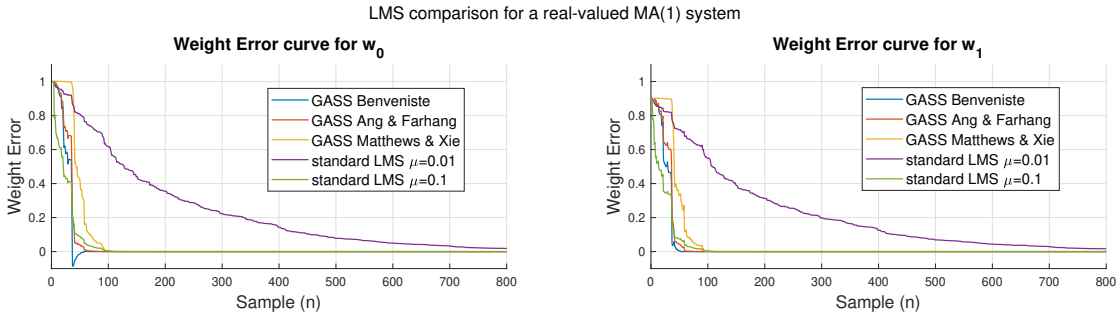

Figure 18: Performance comparison of the weight error curve of LMS for different GASS algorithms and fixed step size.

Part B We can obtain the normalised LMS (NLMS) algorithm by the update equation $\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e_p(n)x(n)$, with $e_p(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n+1)$ being the a posteriori error.

First, we express the a posteriori error using the weight update equation:

$$e_p(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n+1) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n) + \mu e_p(n)x(n) \tag{27}$$
$$= d(n) - \mathbf{x}^T(n)\mathbf{w}(n) - \mu e_p(n)|x(n)|^2 \tag{28}$$

Rearranging, we obtain the a posterior error in terms of the a priori error e(n):

$$e_p(n) + \mu e_p(n)|x(n)|^2 = d(n) - \mathbf{x}^T(n)\mathbf{w}(n) \tag{29}$$

$$e_p(n) = \frac{d(n) - \mathbf{x}^T(n)\mathbf{w}(n)}{1 + +\mu|x(n)|^2} = \frac{e(n)}{1 + +\mu|x(n)|^2} \tag{30}$$

Now, we can substitute into the weight update equation and simplify:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e_p(n)x(n) = \mathbf{w}(n) + \mu \frac{e(n)}{1 + +\mu|x(n)|^2} x(n) \tag{31}$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{e(n)}{1/\mu + +|x(n)|^2} x(n) \tag{32}$$

We can then obtain the equation for the NLMS if we set $\epsilon = 1/\mu$ and $\beta = 1$:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\beta e(n)}{\epsilon + +|x(n)|^2} x(n) \tag{33}$$

This highlights the inverse proportionality between the regularisation factor and the learning rate, leading to a trade-off between stability and speed of convergence.

Part C The generalized normalized gradient descent (GNGD) algorithm has been used to estimate the weight coefficients of a MA(1) system. As it is shown in figure 19, using the same learning rate ($\mu = 0.01$), GNGD is consistently

faster in convergence to the correct weight value comparatively to GASS by Benveniste, leading to a steeper error curve across iterations. It has to be noticed that despite it being faster, this does not lead to a worse steady-state error, meaning that GNGD can be said to overall outperform. Furthermore, the performance can also be compared by analysing the computational complexity of the two algorithms by counting the number of operations for each update of ψ, w, μ and e; whilst the GASS by Benveniste has a complexity of $O(M^2)$, making exponentially more expensive as the order increases, GNGD is characterised by lower complexity ($O(M)$), leading to a better performance also from this perspective.

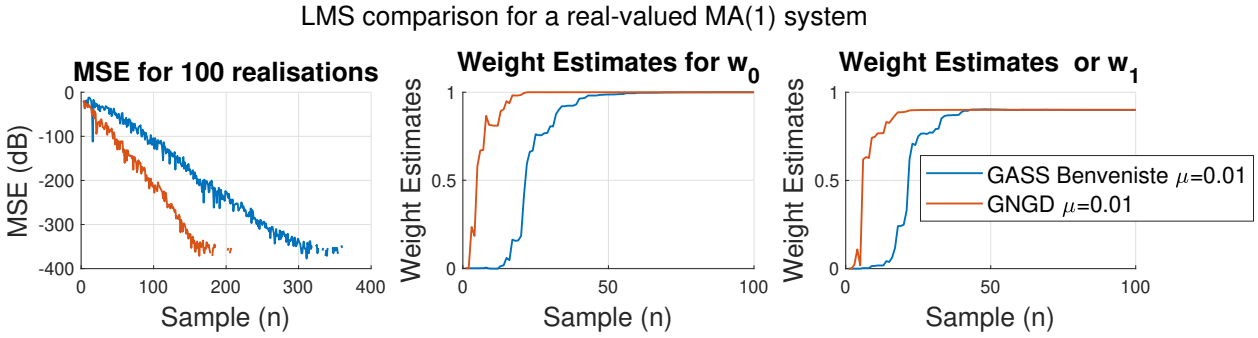

Figure 19: Comparing between GASS Benveniste and GNGD algorithms for weight estimation

2.3 Adaptive Noise Cancellation

Part A We use the Mean Square Prediction Error 34 to evaluate the performance of Adaptive Line Enhancer (ALE) and Adaptive Noise Cancellation (ANC) denoising techniques.

$$MSPE = \frac{1}{N}\sum_{n=0}^{N-1}(x(n)-\hat{x}(n))^2 \tag{34}$$

With x(n) being the clean signal and $\hat{x}(n)$ its estimation.
The signal s(n) to be denoised by a $M > 1$ ALE is composed of x(n), a unit amplitude sinusoid of angular frequency $\omega_o = 0.01\pi$, and $\eta(n)$, coloured noise generated using the moving average filter ($\eta(n) = v(n) + 0.5v(n-2)$). To find the minimum delay Δ of the ALE, we start by analysing the effect of the noise correlation on the linear predictor:

$$MSPE = E\{(s(n)-\hat{x}(n))^2\} = E\{(\eta(n)+x(n)-\hat{x}(n))^2\} = E\{x(n)-\hat{x}(n)\} + 2E\{(x(n)-\hat{x}(n))\eta(n)\} + E\{\eta(n)^2\} \tag{35}$$

Given that there are only two terms depending on the noise level η and that $E\{\eta(n)^2\}$ constitutes a lower bound for MSPE entirely dependent on the noise power, the only term we can minimise is $E\{(x(n)-\hat{x}(n))\eta(n)\}$. Assuming x(n and the noise are uncorrelated, we can then disregard the first component and then substitute the equation of the η and x to get:

$$E\{(x(n)-\hat{x}(n))\eta(n)\} = E\{\hat{x}(n))\eta(n)\} = E\{(v(n)+0.5v(n-2)) * w^T x\} \tag{36}$$

$$= E\{(v(n)+0.5v(n-2)) * \sum_{i=0}^{M} w_i(x(n-\Delta-i)+\eta(n-\Delta-i))\} \tag{37}$$

Assuming that v and x are uncorrelated, the first term of the sum is zero, therefore:

$$E\{\hat{x}(n))\eta(n)\} = E\{(v(n)+0.5v(n-2)) * \sum_{i=0}^{M} w_i\eta(n-\Delta-i))\} \tag{38}$$

$$= E\{(v(n)+0.5v(n-2)) * \sum_{i=0}^{M} w_i(v(n-\Delta-i)+0.5v(n-\Delta-i-2))\} \tag{39}$$

Given the above, if $\Delta > 2$, the equation is equal to the expectation of the uncorrelated noise samples, meaning the term goes to zero.
This result is confirmed by the results in figure 20, in which it can be observed that across 100 realisations of the denoising process, the retrieval of the original signal from the noisy one is far better with a $\Delta > 2$, as the noise effects are attenuated. However, it can also be observed that $\hat{x}(n)$ suffers from a slight phase shit compared to the original signal when the delay increases As shown in figure 20

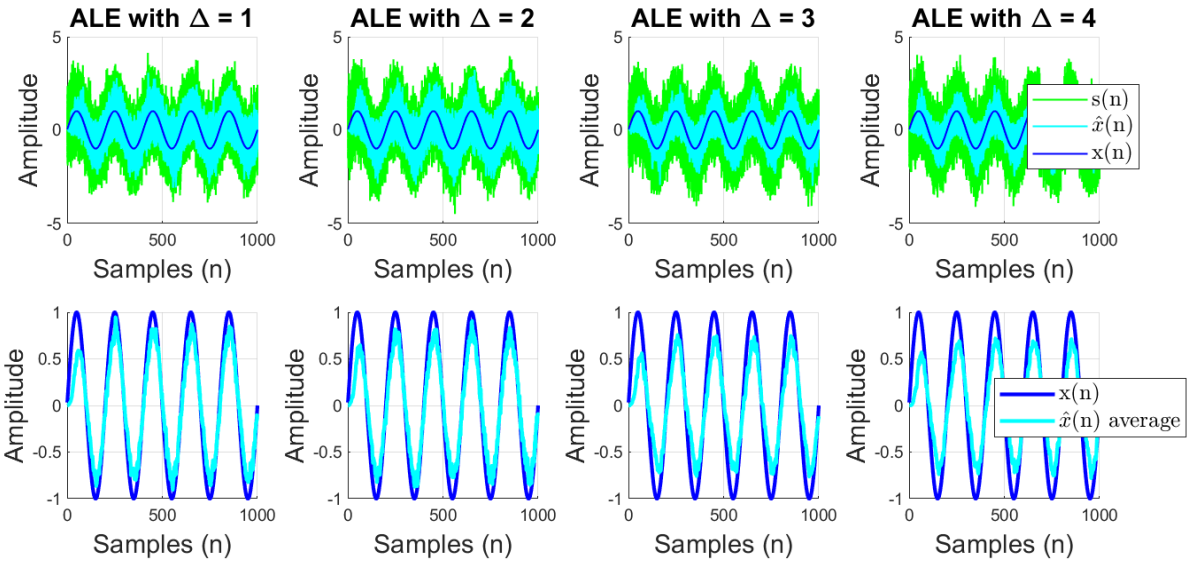

Figure 20: Denoising results using ALE with variable delay Δ

Part B The results of applying ALE across 100 realisations with variable delay and filter order are shown in figure 21. As expected, the Mean Square Prediction Error is significantly lower when using a $\Delta = 3$ across all orders as this delay allows a better noise performance whilst avoiding an increased phase shift; similarly, the order M also affect the MSPE as a greater M will allow fitting to the complexity of the model (avoiding under-fitting), but if the M increases further, this leads to a higher MSPE due to overfitting. The sweet spot for this system is found at $M = 7$ and $\Delta = 3$.

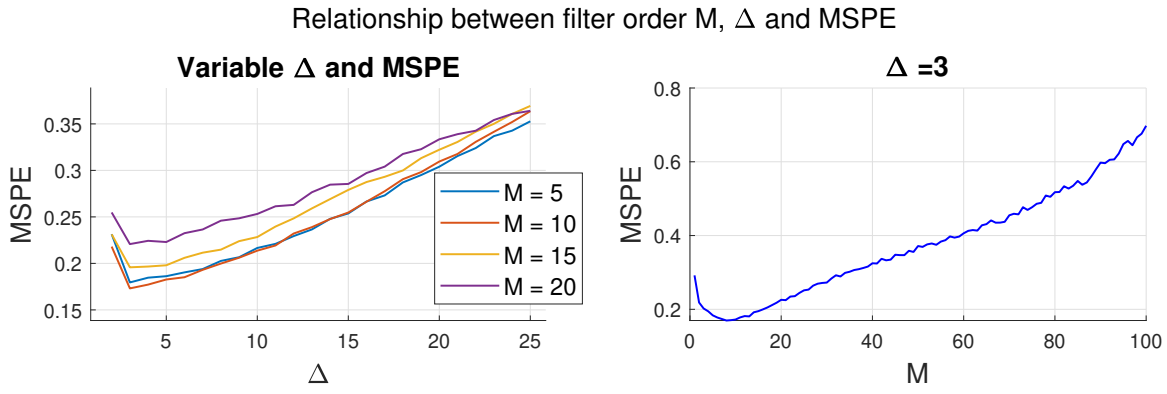

Figure 21: Dependence of ALE performance (in terms of MSPE) upon delay Δ and filter order M

Part C We implemented the Adaptive Noise Cancellation technique and tested on the signal defined in the previous sections. As can be seen from picture 22, using ANC instead of ALE leads to a low MSPE ($MSPE_{ALE} = 0.173$ and $MSPE_{ANC} = 0.068$) and better overall performance, even though it takes some time for the ANC denoising process to converge. Therefore, ANC would be more suitable only if we have a large number of samples; otherwise, the ALE is to be preferred despite its greater steady-state error thanks to the speed of its convergence.

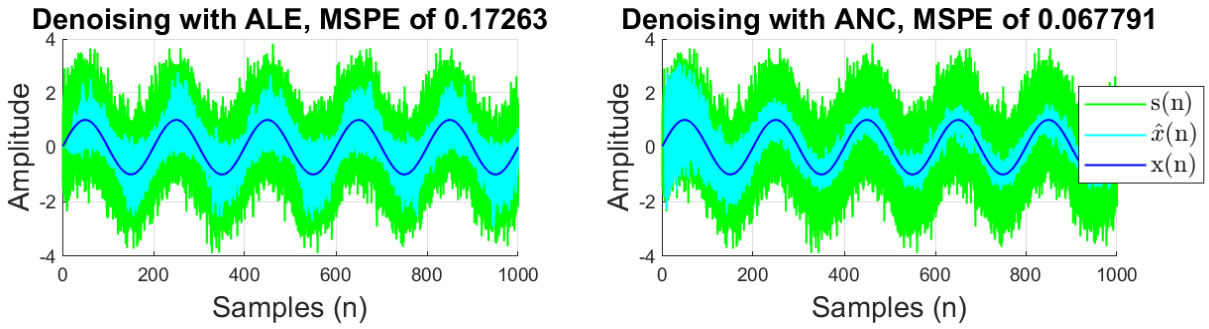

Figure 22: Comparison between ALE and ANC denoising performance.

Part D We use the ANC strategy to denoise EEG data from the POz electrode (original signal in figure 23). The denoising results for different learning rates and model orders are compared in figure 24. In this section, we focus on removing the 50Hz interference present in the data. As it can be observed, a greater model order enables better noise removal up to around $M = 20$, after which overfitting occurs, and we observe power attenuation around the frequency of the interference. Moreover, the size of the learning rate μ used also influences the performance, characterised by a trade-off between greater attenuation of the noise and the attenuation of the original signal as well. To avoid overfitting and obtain effective noise removal, a sweet spot for this system is found at around $\mu = 0.05$ and $M = 25$. These results are also confirmed by the spectrograms in figure 25.

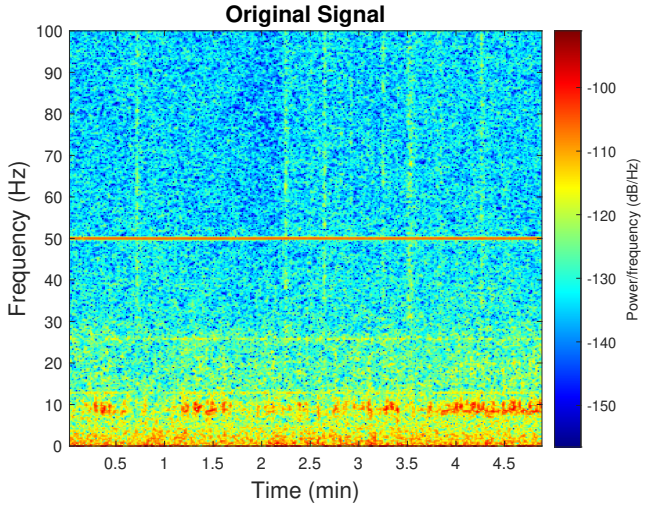

Figure 23: Raw EEG data from POz electrode with interference at 50Hz.

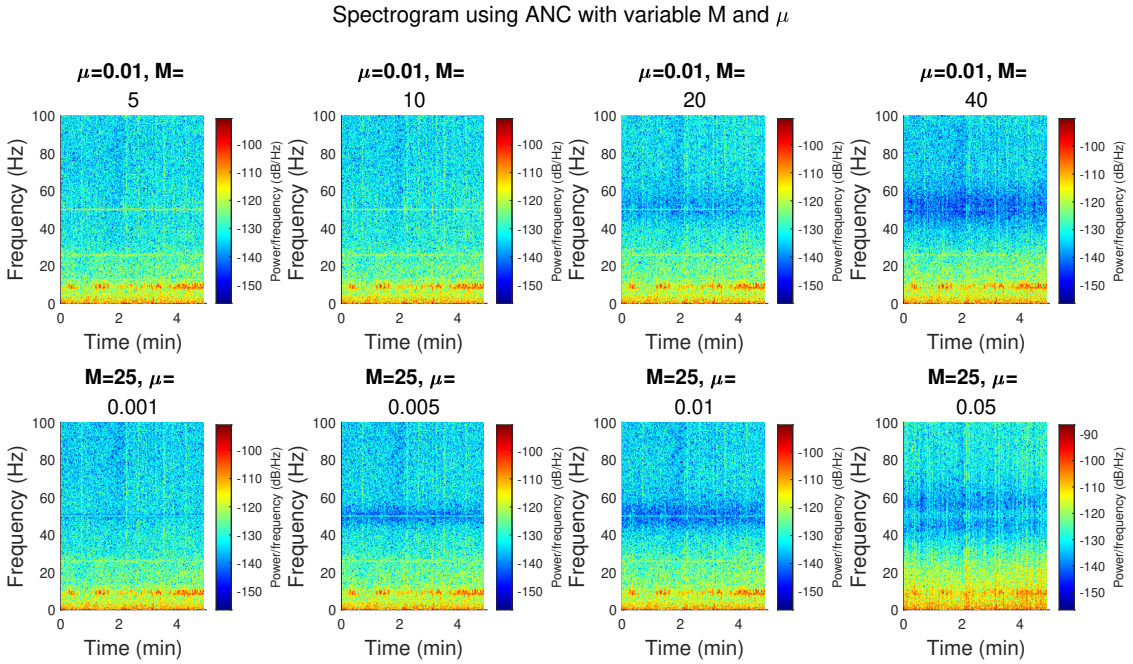

Figure 24: Effect of different learning rates μ and model orders M on the denoising performance of ANC on EEG data (Time-Frequency Diagram)

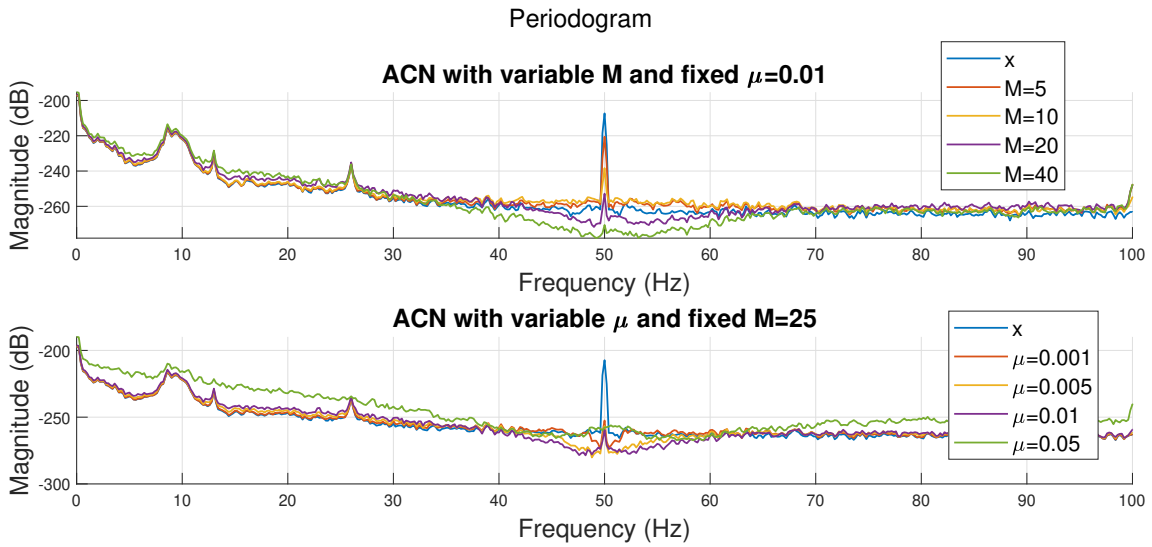

Figure 25: Effect of different learning rates μ and model orders M on the denoising performance of ANC on EEG data (Spectrogram)

3 Widely Linear Filtering and Adaptive Spectrum Estimation

3.1 CLMS and Widely Linear Modelling

Part A Given the first order Widely Linear Moving Average process WLMA(1) (plotted in figure 26):

$$y(n) = x(n) + b_1 x(n-1) + b_2 x*(n-1) \tag{40}$$

$$\text{with } x(n) \sim N(0,1),\ b_1 = 1.5 + 1j,\ b_2 = 2.5 - 0.5j$$

We obtain the circularity coefficients of the signal as $\rho_{WGN} = 0.053$ and $\rho_{WLMA} = 0.864$ by using the formula $\rho = \frac{E\{zz*\}}{E\{zz^T\}}$. This confirms the circularity of WGN, as opposed to the low values for WLMA.
Then, we apply the Complex LMS (CLMS) and Augmented Complex LMS (ACLMS) to identify the processes. The results shown in figure 26 confirm that the CLMS performs poorly with non-circular data, whilst ALMS adapts better to any type of complex data due to its augmented input vector that enables greater degrees of representation.
This is also confirmed by the plot of the learning curve in figure 26, in which the squared error of CLMS across alliterations is consistently larger, although the convergence of ALMS is not as immediate.

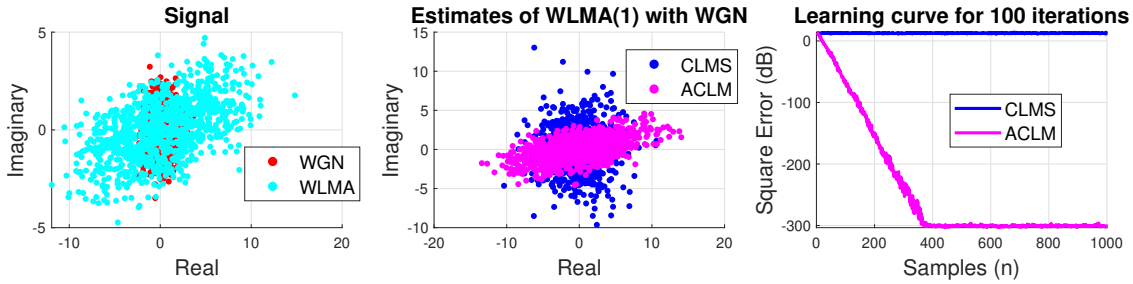

Figure 26: Comparison of CLMS and ACLMS performance in identifying (non-circular) WLMA(1).

Part B The bivariate wind data of the wind speed is combined to obtain complex-valued data ($v(n) = v_{EST} + j * v_{NORTH}$), and CLMS and ALMS are applied to identify the underlying process.
We first compute the circularity of the different wind intensity (low, medium, and high wind plots in figure 27), respectively $\rho_{low} = 0.160$, $\rho_{medium} = 0.454$, $\rho_{high} = 0.624$. As the low wind data has an overall lower circularity, we would expect the ACLMS to perform comparatively better with this data. This is confirmed by figure 28, in which the behaviour of the CLMS and ALMS MSPE is plotted across different model orders. It can be noticed that with relatively low model orders ($M > 10$ for low wind data), the ACLMS outperforms CLMS and reaches the minimum error in all three datasets. However, if the model order is increased further, the ACLMS starts to overfit the noise to the point in which the error is greater than the one using the simpler CLMS algorithm. Therefore, we can say that the model order of the algorithm has to be chosen very carefully in order to minimise MSPE; otherwise, the extra degree of freedom of the ACLMS over CLMS becomes detrimental to the overall performance due to noise.

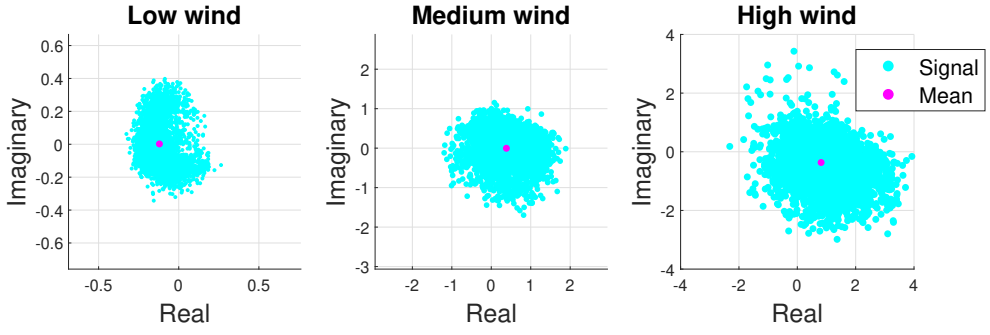

Figure 27: Plots of the bivariate wind dataset divided by wind intensity; highest circularity in the high wind group

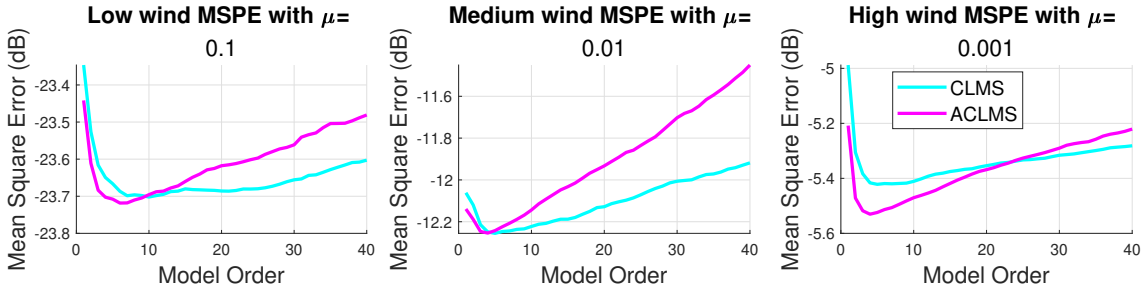

Figure 28: Effect of the order M on the performance (in terms of MSPE) of ACLMS vs CLMS

Part C Three sets of complex voltages are generated as shown in figure 29, respectively a balanced system, a magnitude unbalanced and a phase unbalanced one. It is clearly noticeable how the circularity of the system is evidently impacted by their balance. In fact, to a misalignment in magnitude or phase of the three phases follows a clear decrease in the circularity coefficient ($\rho_{bal} = [3.995e-16] \approx 0$, $\rho_{Vunbal} = [0.228, 0.533, 0.866]$, $\rho_{\phi unbal} = [0.330, 0.134, 0.348]$). Therefore, by looking at the circularity coefficients, it is very intuitive to identify and quantify a fault in the power system.

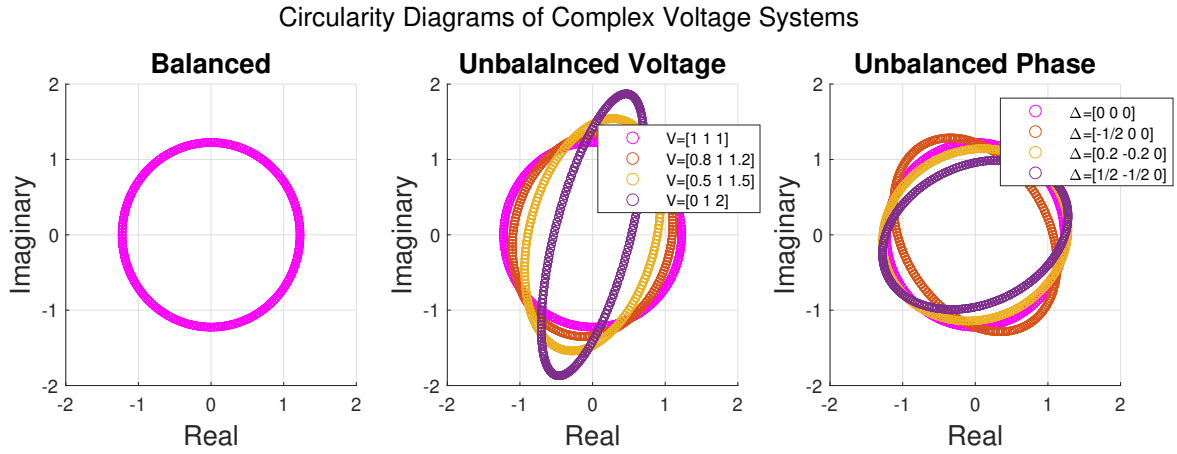

Figure 29

Part D Given strictly linear and widely linear autoregressive models of order 1:

$$v(n+1) = h^*(n)v(n) \tag{41}$$

$$v(n+1) = h^*(n)v(n) + g^*(n)v^*(n) \tag{42}$$

We derive the frequency of the balanced complex voltage ($v(n) = \sqrt{2/3}Ve^{j(2\pi n\frac{f_o}{f_s}+\phi)}$)from the h(n) coefficients by substituting the balance voltage equation into 41:

$$v(n) = \sqrt{2/3}Ve^{j(2\pi(n+1)\frac{f_o}{f_s}+\phi)} = h^*(n)v(n) = \sqrt{2/3}Ve^{j(2\pi n\frac{f_o}{f_s}+\phi)} \tag{43}$$

Leading to the complex conjugate of h(n) to be:

$$h^*(n) = e^j(arctan(\frac{Im\{h\}}{Re\{h\}})) = e^j(2\pi\frac{f_o}{f_s}) \tag{44}$$

Therefore, we can obtain the oscillation frequency as:

$$2\pi\frac{f_o}{f_s} = -arctan(\frac{Im\{h(n)\}}{Re\{h(n)\}}) \tag{45}$$

With the minus indicating the rotation of the system, we obtain:

$$f_o = \frac{f_s}{2\pi}arctan(\frac{Im\{h(n)\}}{Re\{h(n)\}}) \tag{46}$$

Now, we derive the frequency of the unbalanced complex voltage $(v(n) = A(n)e^{j(2\pi n\frac{f_o}{f_s}+\phi)} + B(n)e^{-j(2\pi n\frac{f_o}{f_s}+\phi)}$.
Given that:

$$A(n) = \frac{\sqrt{6}}{6}(V_a(n) + V_b(n)e^{j\Delta_b} + V_c(n)e^{j\Delta_c}) \tag{47}$$

$$B(n) = \frac{\sqrt{6}}{6}(V_a(n) + V_b(n)e^{-j\Delta_b+2/3\pi} + V_c(n)e^{-j\Delta_c-2/3\pi}) \tag{48}$$

Similarly to what was done for the balanced system, we substitute the voltage equation into 42, although in this case, we can separate between the terms with positive and negative exponentials:

$$A(n+1)e^{j(2\pi n\frac{f_o}{f_s}+\phi)} = h^*(n)A(n) + g^*(n)B^*(n) \tag{49}$$

$$B(n+1)e^{-j(2\pi n\frac{f_o}{f_s}+\phi)} = h^*(n)B(n) + g^*(n)A^*(n) \tag{50}$$

Assuming the rate of change of the voltage magnitudes over time is negligible, we can consider $A(n+1) \approx A(n)$ and $B(n+1) \approx B(n)$, therefore:

$$e^{j(2\pi n\frac{f_o}{f_s}+\phi)} = h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} = conj(e^{-j(2\pi n\frac{f_o}{f_s}+\phi)}) = h^*(n) + g^*(n)\frac{A^*(n)}{B(n)} \tag{51}$$

$$e^{j(2\pi n\frac{f_o}{f_s}+\phi)} = conj(e^{-j(2\pi n\frac{f_o}{f_s}+\phi)}) = h^*(n) + g^*(n)\frac{A^*(n)}{B(n)} \tag{52}$$

$$h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} = h(n) + g(n)\frac{A(n)}{B^*(n)} \tag{53}$$

$$h^*(n)(\frac{B^*(n)}{A(n)}) + g^*(n)(\frac{B^*(n)}{A(n)})^2 = h(n)(\frac{B^*(n)}{A(n)}) + g(n) \tag{54}$$

$$g^*(n)x^2 + (h^*(n) - h(n))x - g(n) = 0 \tag{55}$$

Obtaining a second order equation in $x = \frac{B^*(n)}{A(n)}$. We can now find the solution for the equation:

$$x = \frac{(h^*(n) - h(n)) \pm \sqrt{(h^*(n) - h(n))^2 + 4g^*(n)g(n)}}{2g^*(n)} \tag{56}$$

Since $(h^*(n) - h(n)) = 2jIm\{h(n)\}$ we substitute and simplify to obtain:

$$x = \frac{jIm\{h(n)\} \pm \sqrt{Im^2\{h(n)\} - \|g(n)\|^2}}{g^*(n)} \tag{57}$$

By substituting x into 51 and simplifying, we obtain:

$$e^{j(2\pi n\frac{f_o}{f_s}+\phi)} = h^*(n) + jIm\{h(n)\} \pm \sqrt{Im^2\{h(n)\} - \|g(n)\|^2} \tag{58}$$

$$e^{j(2\pi n\frac{f_o}{f_s}+\phi)} = Re\{h(n)\} \pm \sqrt{Im^2\{h(n)\} - \|g(n)\|^2} = e^{j*arctan(\frac{\pm\sqrt{Im^2\{h(n)\} - \|g(n)\|}}{Re\{h(n)\}})} \tag{59}$$

Therefore, we can also obtain the oscillation frequency as:

$$2\pi\frac{f_o}{f_s} = arctan(\frac{\pm\sqrt{Im^2\{h(n)\} - \|g(n)\|^2}}{Re\{h(n)\}}) \tag{60}$$

$$f_o = \frac{f_s}{2\pi}arctan(\frac{\sqrt{Im^2\{h(n)\} - \|g(n)\|^2}}{Re\{h(n)\}}) \tag{61}$$

Part E We can use CLMS and ACLMS algorithms to estimate the operating frequency of the power systems. Figure 30 shows the performance comparison using the two strategies on the systems presented in the previous sections. As expected, the CLMS is ideal for estimating the balanced system due to its near-perfect circularity, leading to no estimation bias and faster convergence compared to ACLMS. However, for unbalanced systems, the error of both estimators is far greater, with wide oscillations for CLMS and more stable convergence with ALCMS. The high steady-state error could be due to the inappropriate choice of the model order.

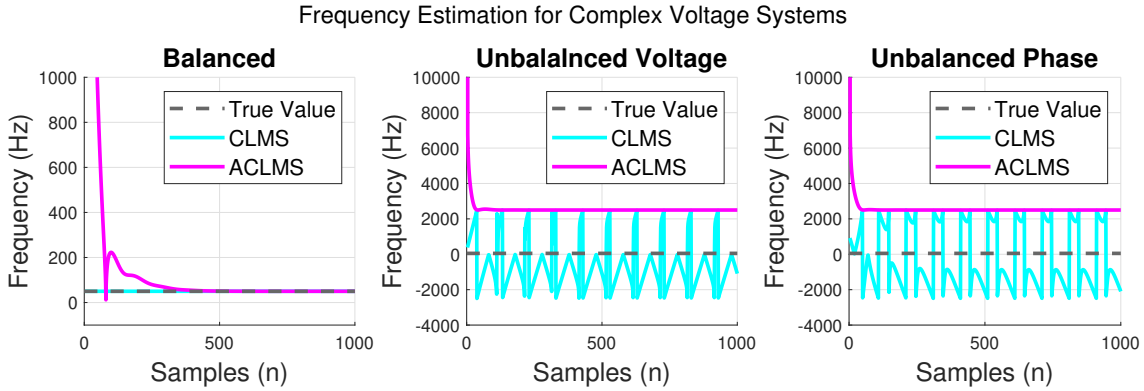

Figure 30: Frequency estimation of three-phase voltage systems using CLMS and ALCMS.

3.2 Adaptive AR for Time-Frequency Estimation

Part A We generate a frequency modulated (FM) signal $y(n) = e^{j(2\pi f_s \phi(n))} + \eta(n)$ with $\eta(n)$ being circular complex-valued white noise with zero mean and variance of 0.05 and find the AR(1) coefficients; however, as it can be seen in figure 31, an AR(1) system cannot completely capture the varying frequency nature of the signal, as it represents only one frequency component at around 200Hz. Therefore, we plot the power spectrum of also higher-order systems to identify multiple frequency components; we obtain that with AR(M) with M¿10, we start to see two main peaks, clearly distinguishable with AR(20).
Although this method might be able to identify the segments with constant frequency, it does not capture the FM nature of the signal. Therefore, we divide the signal into the three components and analyse their power spectrum separately (figure 32). In this way, the part of the signal characterised by constant frequency ($0 < n < 500$) is still well identified by all model orders, with higher precision with increasing complexity; however, we also obtain that mean frequency and even the frequency range of the other two segments is more accurately captured, particularly by the higher order systems (100-350Hz for$501 < n < 1000$ and 100-500Hz for $1001 < n < 1500$). Although this constitutes an improvement, this method still does not capture the non-stationary nature of the signal.

Part B Conversely to a simple AR system, using a CLMS method to identify the frequency of a signal can help us explain the temporal characteristics of the frequency behaviour, thanks to its iterative approach to the coefficients updates and the circularity of the data. As can be seen in figure 33, the time-frequency diagram obtained using CLMS clearly expresses the frequency modulation of the signal, with increased accuracy when the learning rate of the algorithm is greater ($\mu = 0.1$). For lower values of μ, the CLMS convergence is not fast enough to properly capture frequency behaviour in the beginning samples, and the overall accuracy is lower, although this estimate is characterised by lower variance.

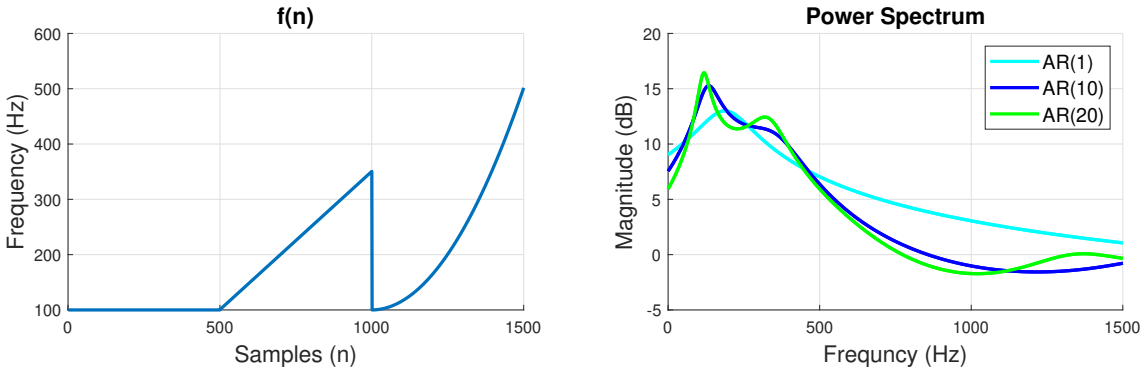

Figure 31: FM signal and its power spectrum estimation using different AR systems

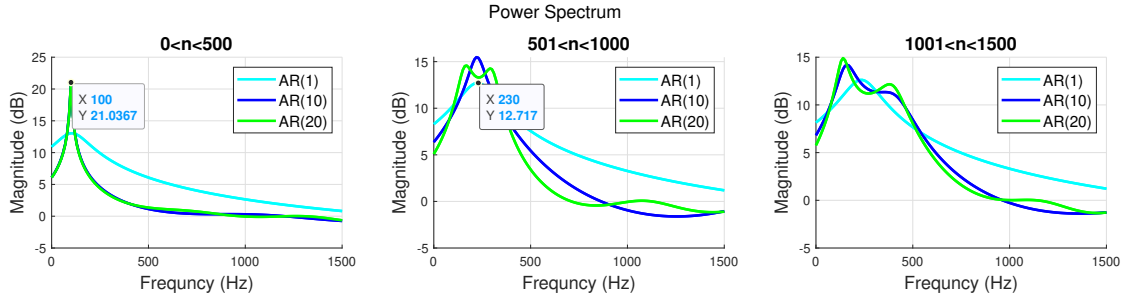

Figure 32: Power spectrum of different segments of the FM signal comparing the results with different AR complexity

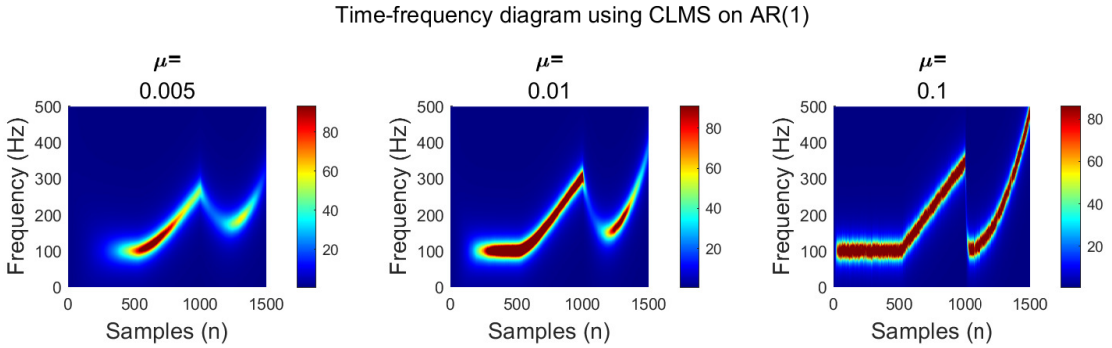

Figure 33: Time-Frequency diagram of the FM signal using CLMS with different learning rates

3.3 A LMS Real-Time Spectrum Analyser

Part A & B We can represent a signal y(n) using its Fourier transform, i.e. a linear combination of N harmonically related sinusoids and

$$\hat{y}(n) = \sum_{k=0}^{N-1} w(k) e^{j2\pi kn/N} \tag{62}$$

If we represent this in vector form $\hat{\mathbf{y}} = \mathbf{F}\mathbf{w}$, we can find the optimal weights by minimising the estimation error, i.e. the sum of square errors across all samples:

$$min_w \|\mathbf{y} - \hat{\mathbf{y}}\| = min_w \sum_{n=0}^{N-1} \|y(n) - \hat{y}(n)\|^2 \tag{63}$$

This can be also expanded as:

$$\|\mathbf{y}-\hat{\mathbf{y}}\| = (\mathbf{y}-\hat{\mathbf{y}})^H(\mathbf{y}-\hat{\mathbf{y}}) = (\mathbf{y}\text{-}\mathbf{F}\mathbf{w})^H(\mathbf{y}\text{-}\mathbf{F}\mathbf{w}) \tag{64}$$
$$= \mathbf{y}^H\mathbf{y} - \mathbf{y}^H\mathbf{F}\mathbf{w} - \mathbf{w}^H\mathbf{F}^H\mathbf{y} + \mathbf{w}^H\mathbf{F}^H\mathbf{F}\mathbf{w} \tag{65}$$

To minimise this, we differentiate and obtain the optimal weights:

$$\frac{\partial\|\mathbf{y}-\hat{\mathbf{y}}\|}{\partial\mathbf{w}} = 0 - \mathbf{y}^H\mathbf{F} - \mathbf{F}^H\mathbf{y} + 2\mathbf{F}^H\mathbf{F}\mathbf{w} \tag{66}$$
$$\mathbf{w} = (\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H\mathbf{y} \tag{67}$$

Finally, the estimation of y can be computed as:

$$\hat{\mathbf{y}} = \mathbf{F}\mathbf{w} = \mathbf{F}(\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H\mathbf{y} \tag{68}$$

Therefore, the estimated signal $\hat{\mathbf{y}}$ can be defined as one of the projections of $\mathbf{y}$ onto the columns space of the matrix $\frac{1}{N}F$ (N orthonormal basis) that minimises the error (as defined in 63). The projection matrix is then $\mathbf{P} = \mathbf{F}(\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H$.

Part C We implement the DFT-CLMS and test it on the FM signal from the previous section (3.2). Figure 34 shows the Time-Frequency diagram of the signal; although this estimate is more accurate compared to the one based on the AR system, as it captures more individual frequency components, this is still not suitable for representing the non-stationary behaviour. As we can observe from the spectrum, once a component appears in the diagram, it will remain, propagating across all following samples. A solution to this would be to introduce the leaky version of the DFT-CLMS. This, in fact, takes into account the temporal features by using a leakage factor, as previously discussed. Figure 34 shows how increasing γ indeed makes the system more sensitive to non-stationary behaviour, although it ultimately leads to a far greater variance if the leakage coefficient is too large ($\gamma = 0.1$).

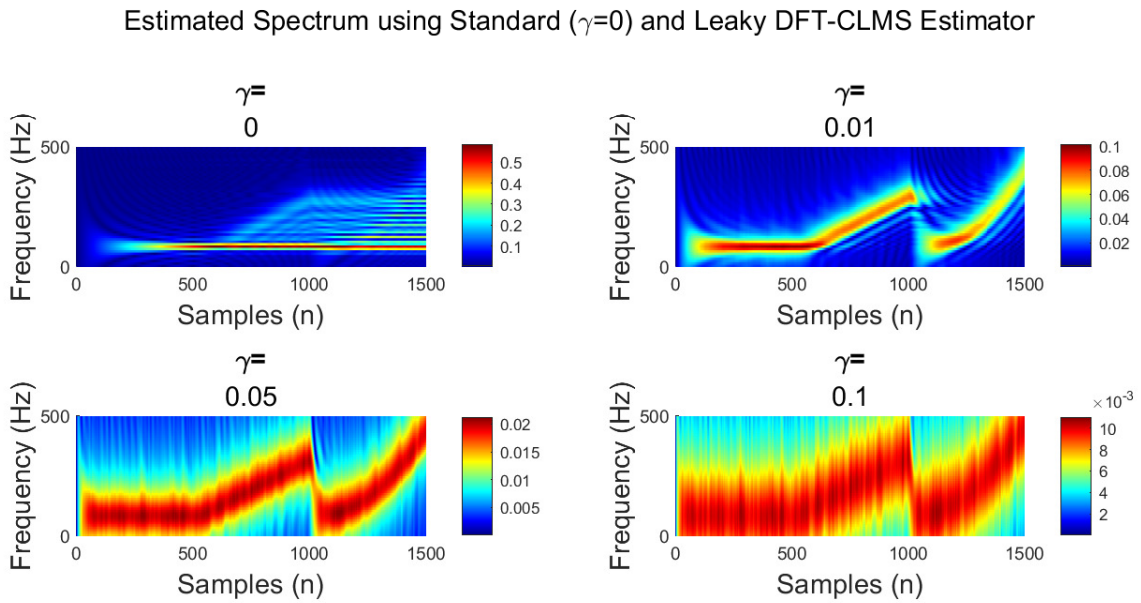

Figure 34: DFT-CLMS ($\gamma = 0$) and Leaky DFT-CLMS estimation for he time-frequency behaviour of a FM signal

Part D We use the DFT-CLMS to analyse EEG data from the POz electrode. From the Time-Frequency diagram in figure 35, we can clearly identify the noise interference at 50Hz and other frequency components, such as the 8.6Hz components due to the alpha-rhythm and the steady-state visual evoked potential (SSVEP) at 13Hz. Since the EEG data is fairly stationary, it is sufficient to use the standard CLMS with $\gamma = 0$. Although this is an effective representation of the EEG data, this method is computationally expensive and ultimately less clear than the initial power spectrum.

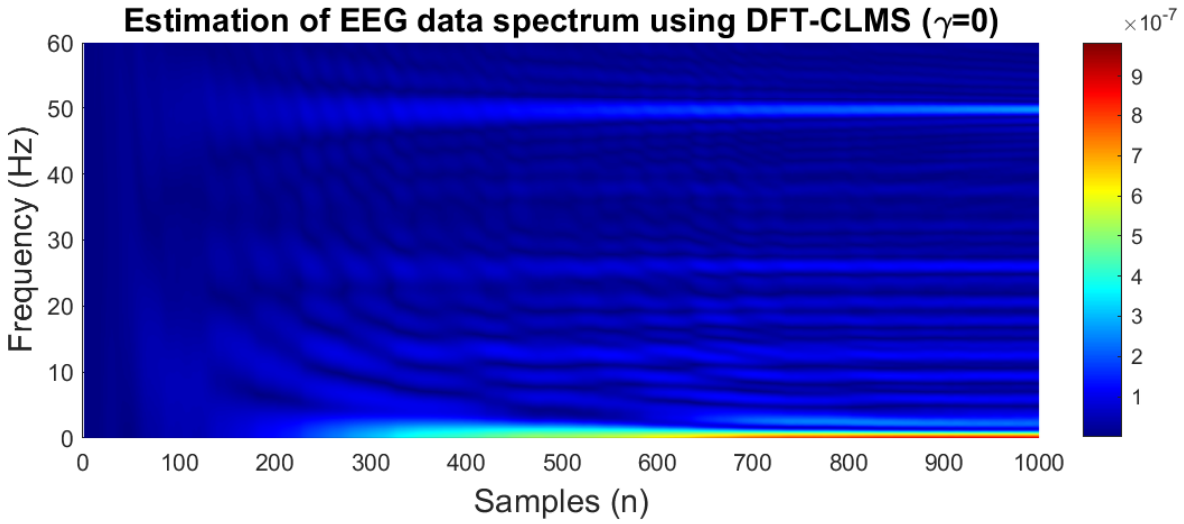

Figure 35: Time-Frequency diagram of EEG data obtained using DFT-CLMS

4 From LMS to Deep Learning

4.1

Figure 36 shows the plots of the zero mean signal and its one-step ahead prediction using AR(4) and $\mu = 0.00001$. Due to the low learning rate, the estimation is not characterised by fast convergence, with a large error in the initial samples. However, the estimation ultimately converges with low steady-state error. Overall, this estimator leads to a high MSPE ($MSPE = 40.094$) and low prediction gain ($R_p = 10log_{10}\frac{\sigma_y^2}{\sigma_e^2} = 5.197$), indicative of a high error variance. This highlights how this method might not be the most suitable for this type of estimation.

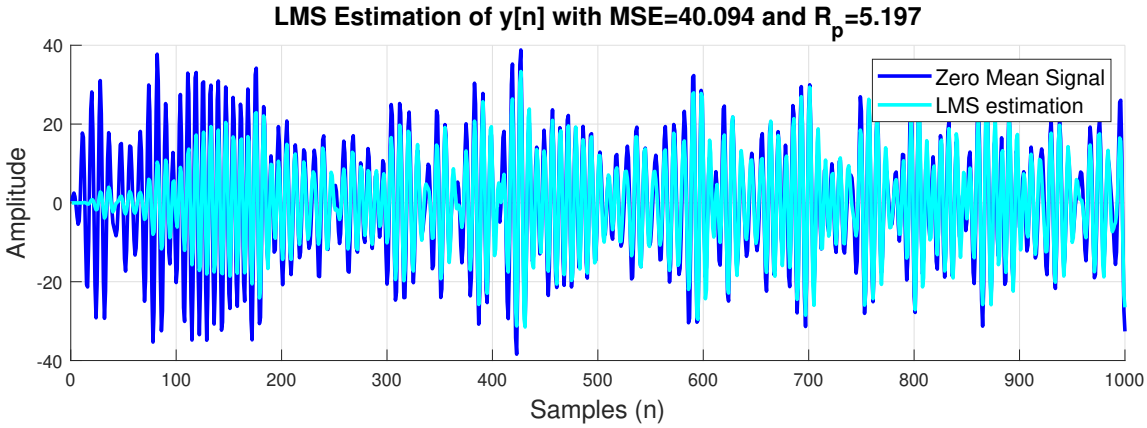

Figure 36: AR(4) LMS estimation

4.2

We add non-linearity to the output of the LMS by using tanh as an activation function. The resulting dynamical perceptron is now suitable to capture the non-linear behaviour of the data and its frequency fluctuation; however, the limited range ($[-1,1]$) of the tanh (coefficient $a = 1$) forces the estimation to a narrow range of values (figure 37), not representative of the magnitude of the data, and this leads to bad overall performance ($MSE = 196.730$ and $R_p = -23.190$).

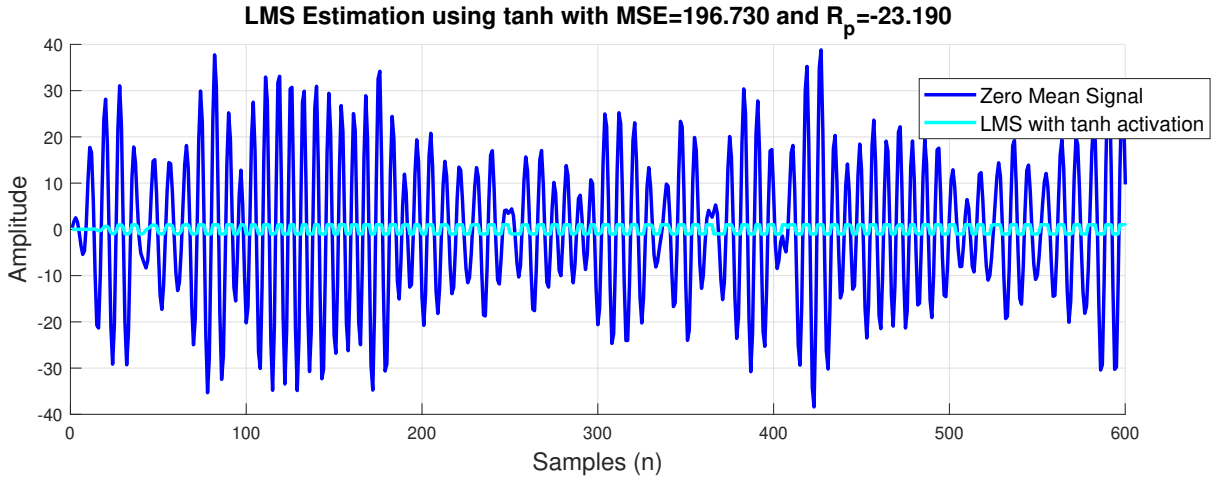

Figure 37: LMS estimation using tanh activation

4.3

The natural solution to the limitation described in the previous section is to use a coefficient a to expand the range of the activation function. The best performance is obtained when using a coefficient of 80, achieving maximum prediction gain and minimum MSE ($MSE = 4.784$ and $R_p = 16.494$). Increasing or decreasing the value of a leads to

worse performance, as the sweet spot is found when the range of tanh targets the peak-to-peak range of the original signal, given that this is symmetrical around 0. Although this method gives an enhanced performance, its success is based on the assumption that we know the amplitude of the signal, which may not always be the case.

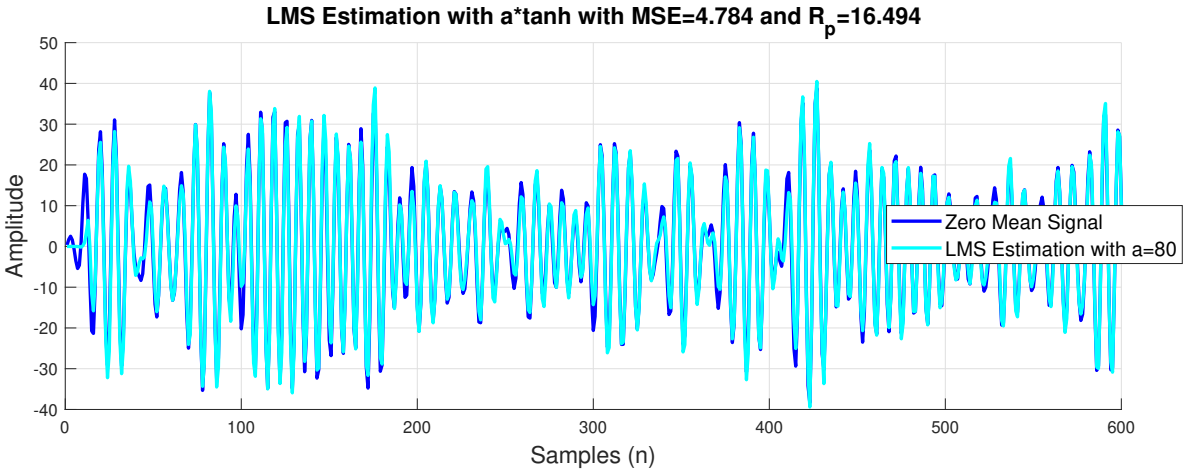

Figure 38: LMS estimation using $80 * tanh$ activation

4.4

Assuming the signal is a non-zero mean, the activation function needs to take into account the bias of the system. The bias in the activation function ($a * tanh(\mathbf{w}^T\mathbf{x} + b)$) has been implemented considering the 'augmented' input to the algorithm as $[1, x]^T$. We retrieve the original non-zero mean signal x and plot with the estimation resulting from the new activation function in figure 39. Since we have a bias, the best a coefficient is adjusted to 50, corresponding to the max amplitude of the original signal, resulting in $MSE = 3.716$ and $R_p = 12.241$. This performance is lower than without the bias due to the convergence time of the estimator in the beginning samples.

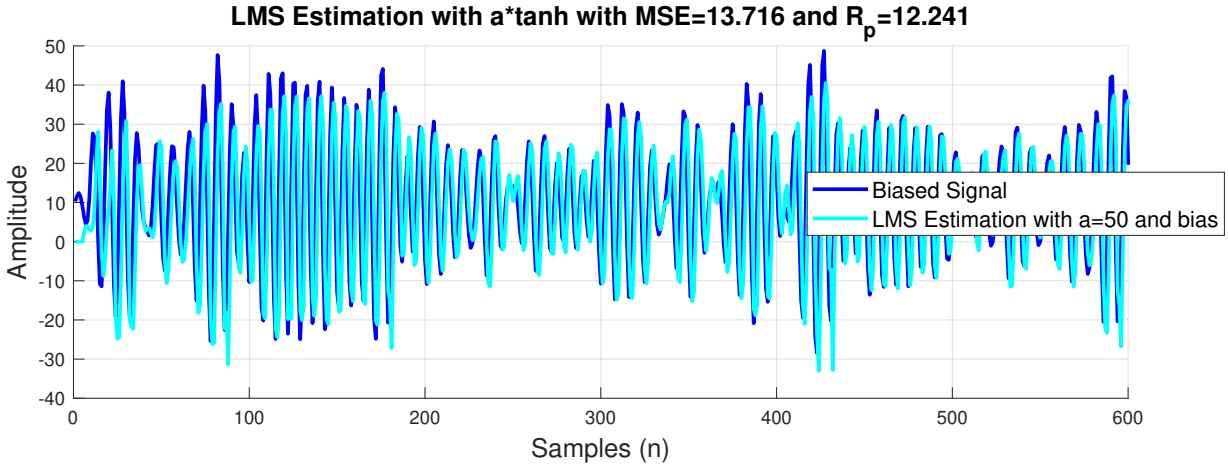

Figure 39: LMS estimation using tanh activation and bias.

4.5

To take into account the limitation in the speed of convergence discussed in the previous section, we can pre-train the weight to overfit to some initial sample. In this case, we compute $w_{initial}$ for 100 iterations and 20 samples. The resulting estimation is plotted in figure 41, giving an improvement in performance ($MSE = 13.630$ and $R_p = 12.270$) compared to the non-pre-trained LMS, as expected.

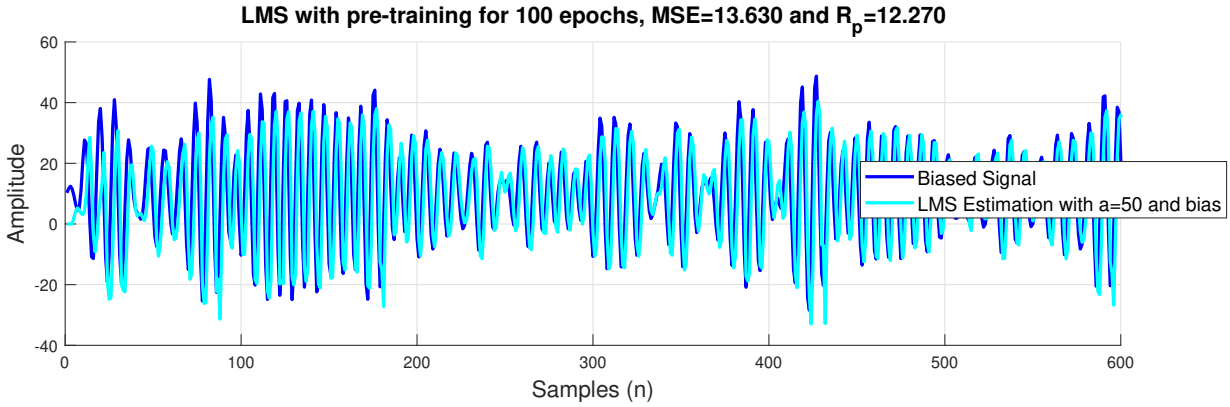

Figure 40: Pre-trained LMS estimation using tanh activation and bias.

4.6

Backpropagation is the process of propagating the errors back through the network, fine-tuning the weights of the individual neurons. The training process consists in iterating between two steps until the training error reaches a minimum:

- **Forward Propagation** the input x is fed into the network, and the output is calculated by applying the weights w to each neuron and passing the result through the activation function (e.g. $tanh(w^T x)$). Then, the training error between the predicted and the actual y is calculated using a given cost function (e.g. $\frac{1}{2}|y - \hat{y}|$).
- **Backward Propagation** The error is propagated back through the network by computing the partial derivatives of the cost function with respect to each weight. The weights are then updated using optimization techniques such as gradient descent; then, the network computes the forward pass for the next epoch, and the iterative process continues until convergence to a low error.

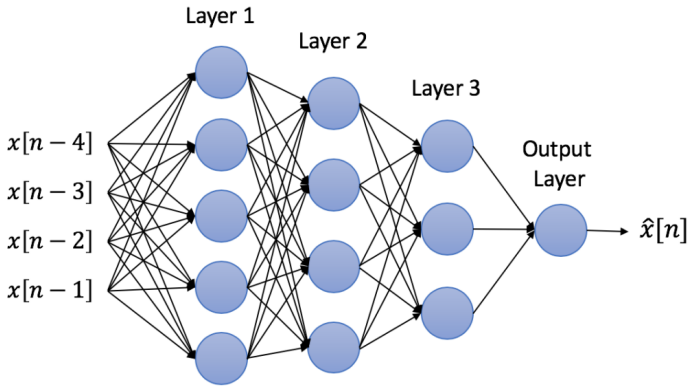

Figure 41: Backpropagation diagram from the assignment notes.

4.7

A 4-layer deep network with a learning rate of 0.01 and noise power of 0.05 has been trained and tested, and compared against the performance of simple dynamical perceptron with linear and tanh activation (input and out signals in figure 42). It can be observed from figure 43 that the Deep Network follows the behaviour of the true output more closely during training and ultimately results in a more accurate test output. This is due to the higher complexity of the network that, conversely to the linear dynamical perceptions, enables to model the non-linear behaviour of the data. Furthermore, the tanh activation function of the second perception, despite being non-linear, has a very limited degree of representation compared to the deep network, resulting in a very similar performance to one of the other perceptron. Furthermore, the learning curves in figure 44 highlight how the deep network, despite having a

lower convergence speed due to its complexity, achieves the lowest training error and consistently lower test error. Therefore, the deep network performance is characterised by a trade-off between speed of convergence and stability on one side and low test error on the other.

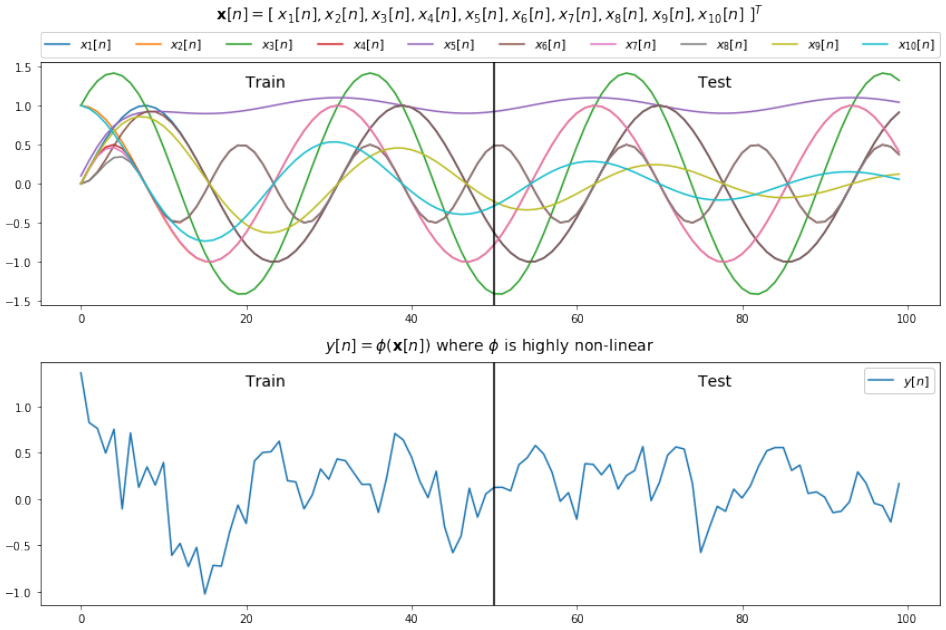

Figure 42: Input data x and output prediction y for test and training epochs

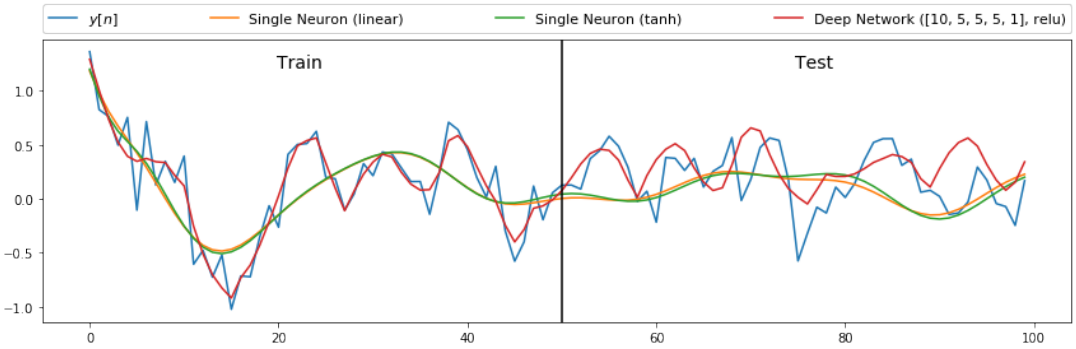

Figure 43: Comparison of the y prediction using different models across training and test sets, $\sigma^2 = 0.05$

4.8

When testing the three networks with different noise power values, we start to observe that the benefits of using a deep network over the perceptron highlighted in the previous section are not always true, depending on the amount of noise. If the noise power is reduced ($\sigma^2 = 0.005$) (figure 45 and 46), the deep network achieves even lower test error levels compared to the ones from the perceptions (despite some instability due to weights initialisation at the beginning). However, if the noise power is large ($\sigma^2 = 0.5$) (figure 45 and 46), the stability of the deep network performance across all epochs is compromised by oscillations due to instability. In this case, despite the deep network achieving lower training error, it does not generalise well in the test set as it overfitted to noise during training; therefore, with large noise power, a dynamical perception architecture is preferable thanks to its high noise sensitivity. Therefore, the choice of predictor should depend upon the amount of noise in the signal, considering the unavoidable trade-off between computation complexity, speed of convergence and instability on one hand and higher degrees of representation and lower test error on the other.

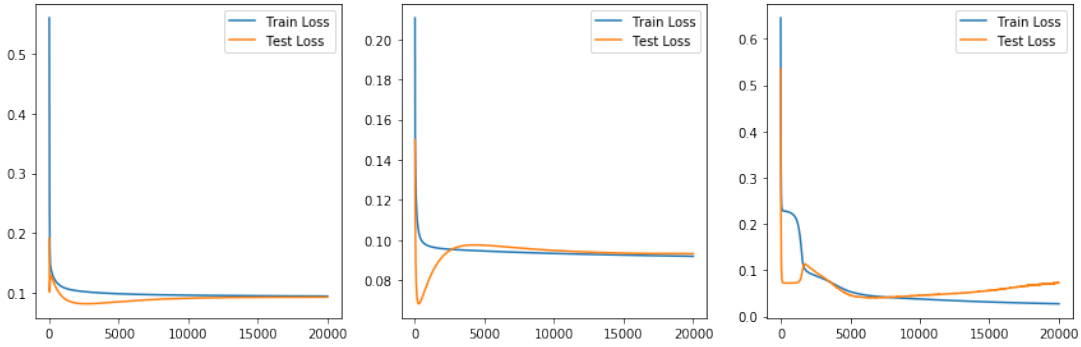

Figure 44: Comparison of the learning curve for Single Neuron with linear and tanh activation, and Deep Network (respectively), $\sigma^2 = 0.05$

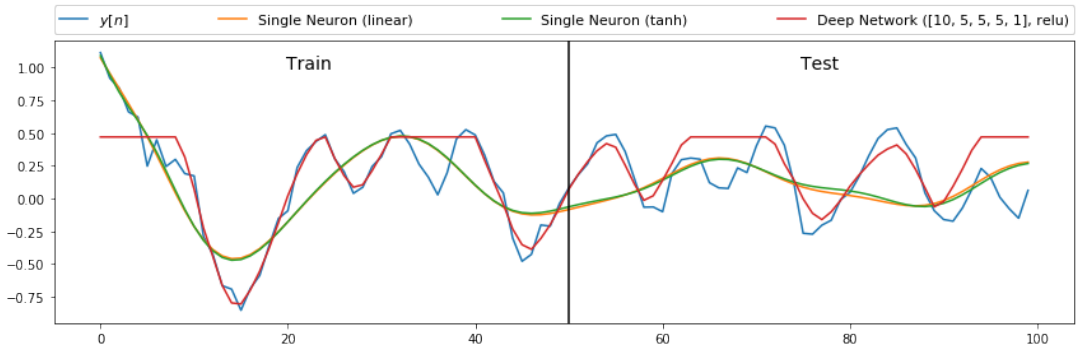

Figure 45: Comparison of the y prediction using different models across training and test sets, $\sigma^2 = 0.005$

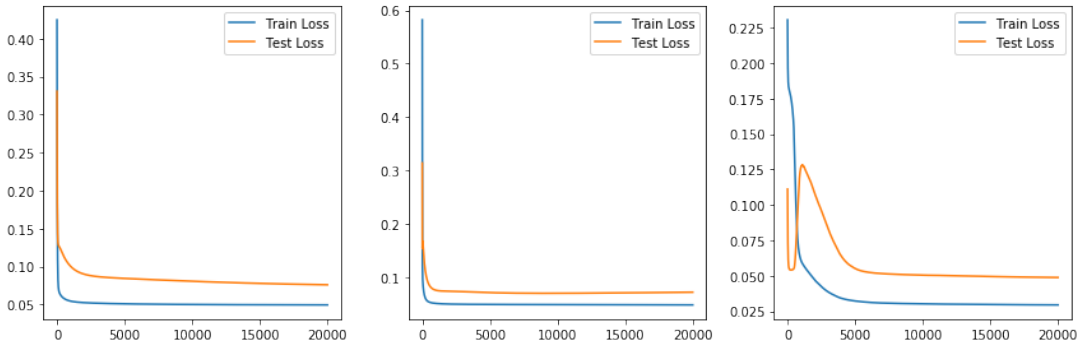

Figure 46: Comparison of the learning curve for Single Neuron with linear and tanh activation, and Deep Network (respectively), $\sigma^2 = 0.005$

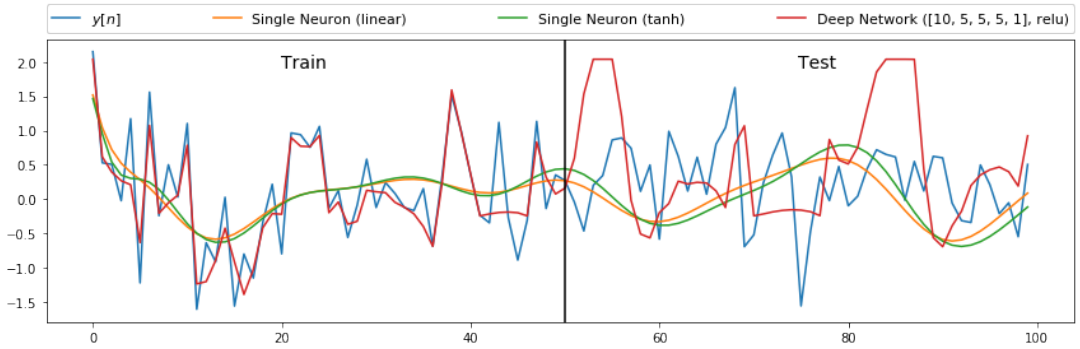

Figure 47: Comparison of the y prediction using different models across training and test sets, $\sigma^2 = 0.5$

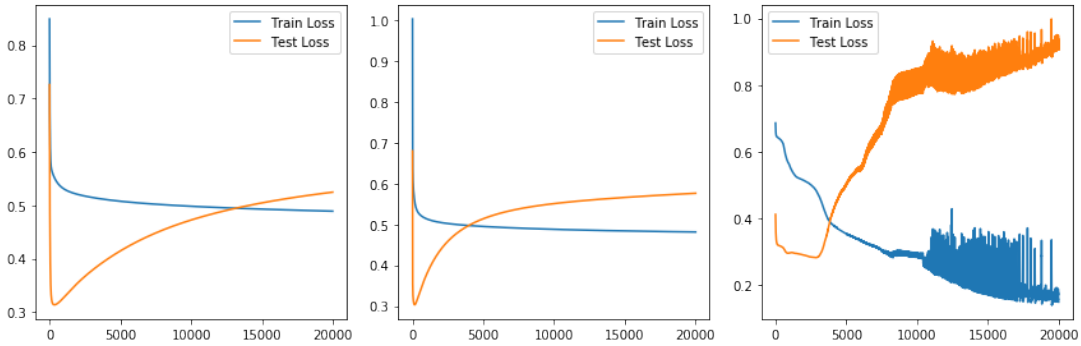

Figure 48: Comparison of the learning curve for Single Neuron with linear and tanh activation, and Deep Network (respectively), $\sigma^2 = 0.5$