Imperial College London

IMPERIAL COLLEGE LONDON

Adaptive Signal Processing and Machine Intelligence

Coursework 2022

Author: Livia Soro
CID: 01549029
Professor: Prof. Danilo P. Mandic

Due Date: April 13, 2022

Contents

1 Classical and Modern Spectrum Estimation

1.1 Properties of Power Spectral Density (PSD)

We aim to analytically show that the definitions of PSD in (1.1) and (1.2) are equivalent, under a mild assumption that the covariance sequence *r(k)* decays rapidly, as formalised in (1.3).

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

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

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

We begin by expanding the definition in (1.2), then rearrange using the linearity of the expectation operator. We assume the process is stationary, meaning $E\{x(n)x^*(m)\} = r(n-m)$:

$$\begin{aligned}
P(\omega) &= \lim_{N\to\infty} E\Big\{\frac{1}{N}\Big(\sum_{n=0}^{N-1} x(n)e^{-j\omega n}\Big)\Big(\sum_{n=0}^{N-1} x(n)e^{-j\omega n}\Big)^*\Big\} \\
&= \lim_{N\to\infty} E\Big\{\frac{1}{N}\Big(\sum_{n=0}^{N-1} x(n)e^{-j\omega n}\Big)\Big(\sum_{n=0}^{N-1} x^*(n)e^{j\omega n}\Big)\Big\} \\
&= \lim_{N\to\infty} \frac{1}{N}\Big(\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} E\{x(n)x^*(m)\}e^{-j\omega(n-m)}\Big) \\
&= \lim_{N\to\infty} \frac{1}{N}\Big(\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} r(n-m)e^{-j\omega(n-m)}\Big)
\end{aligned} \tag{1.4}$$

The expression inside the limit has the form of $\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} f(n-m)$, which can be split into three sums, considering the cases $m < n$, $m = n$ and $m > n$, respectively:

$$\sum_{n=0}^{N-1}\sum_{m=0}^{n-1} f(n-m) + \sum_{n=0}^{N-1} f(0) + \sum_{n=0}^{N-1}\sum_{m=n+1}^{N-1} f(n-m)$$

First term: We use a dummy variable and substitute $\tau = n-m$, which in this case is always positive (as m goes from 0 to $n-1$):

$$\sum_{n=0}^{N-1}\sum_{m=0}^{n-1} f(n-m) = \sum_{n=0}^{N-1}\sum_{\tau=n}^{1} f(\tau) = \sum_{n=0}^{N-1}\sum_{\tau=1}^{n} f(\tau)$$

Since we have a function of τ, we exchange the order of the sums. Originally, τ ends with n; when we switch the order of summation, n starts with τ:

$$\sum_{\tau=1}^{N-1}\sum_{n=\tau}^{N-1} f(\tau) = \sum_{\tau=1}^{N-1} f(\tau)\sum_{n=\tau}^{N-1} 1 = \sum_{\tau=1}^{N-1}(N-1-\tau+1)f(\tau) = \sum_{\tau=1}^{N-1}(N-\tau)f(\tau) = \boxed{\sum_{\tau=1}^{N-1}(N-|\tau|)f(\tau)}$$

Since in this case $\tau > 0$, $\tau = |\tau|$.

Second term: In this case $n = m$. Using the same dummy variable, we rearrange as above:

$$\sum_{n=0}^{N-1} f(0) = Nf(0) = \boxed{\sum_{\tau=0}^{0} (N - |\tau|) f(\tau)}$$

Third term: In this case $m > n$. Using the same dummy variable, we rearrange:

$$\sum_{n=0}^{N-1} \sum_{m=n+1}^{N-1} f(n-m) = \sum_{n=0}^{N-1} \sum_{\tau=-1}^{n-(N-1)} f(\tau) = \sum_{n=0}^{N-1} \sum_{\tau=n-(N-1)}^{-1} f(\tau)$$

Again, since we simply have a function of τ we exchange the order of the sums.

$$\sum_{\tau=-(N-1)}^{-1} \sum_{n=0}^{\tau+(N-1)} f(\tau) = \sum_{\tau=-(N-1)}^{-1} (\tau + N - 1 + 1) f(\tau) = \sum_{\tau=-(N-1)}^{-1} (\tau + N) f(\tau) = \boxed{\sum_{\tau=-(N-1)}^{-1} (N - |\tau|) f(\tau)}$$

Since in this case $\tau < 0$, $\tau = -|\tau|$.
Combining all the results found above, we obtain:

$$\sum_{n=0}^{N-1} \sum_{m=0}^{N-1} f(n-m) = \sum_{\tau=-(N-1)}^{N-1} (N - |\tau|) f(\tau)$$

Substituting back into the definition of PSD (1.4), we obtain:

$$P(\omega) = \lim_{N\to\infty} \frac{1}{N} \Big(\sum_{n=0}^{N-1} \sum_{m=0}^{N-1} r(n-m) e^{-j\omega(n-m)} \Big) = \lim_{N\to\infty} \frac{1}{N} \Big(\sum_{\tau=-(N-1)}^{N-1} (N - |\tau|) r(\tau) e^{-j\omega\tau} \Big) \tag{1.5}$$

Which can be split into two parts:

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

Considering the second term in (1.6) and using the triangle inequality, we define its magnitude range as follows:

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

The magnitude of the exponential is 1. Given the assumption in (1.3), we obtain:

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

Hence, the second term in (1.6) is 0; which demonstrates that definitions (1.1) and (1.2) are equivalent under the assumption in (1.3):

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

The derived equality is shown to hold through simulation provided that the covariance sequence decays rapidly in Figure 1 (top). Figure 1 (bottom) exemplifies a case where the covariance sequence decays slowly, hence the equality does not hold - the two PSD estimates are not equivalent.

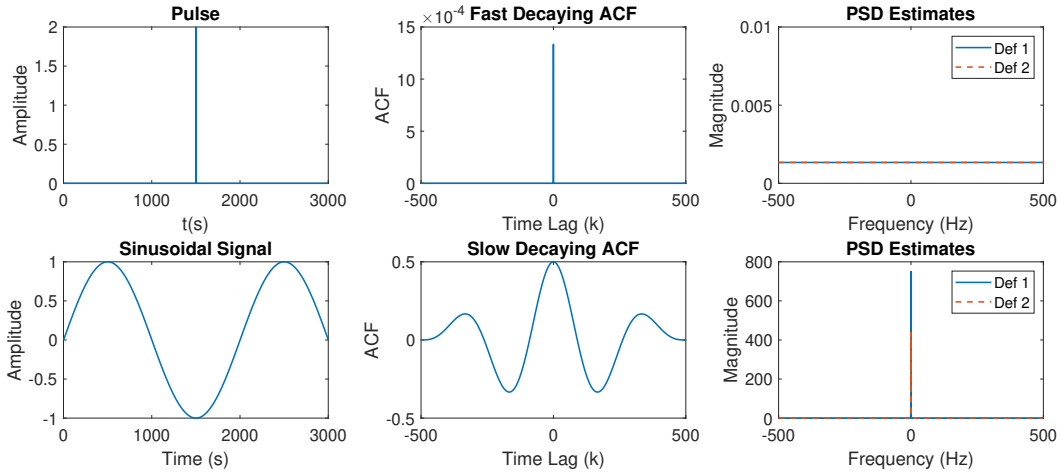

Figure 1: Top: Pulse (left), rapidly decaying autocorrelation of pulse (middle) and estimate of the PSD of the pulse for both definitions (right). **Bottom**: Sine wave (left), slowly decaying autocorrelation of the sine wave (middle) and estimate of the PSD of the sine wave for both definitions.

1.2 Periodogram-based Methods Applied to Real-World Data

a) A periodogram-based spectral estimation with a Hamming window of the same length as the data was applied to the the sunspot data preprocessed in different ways. Figure 2 shows the results. Removing the mean and the trend centers the data in the time domain. This is equivalent to removing the DC offset and some low frequency component driving the data - which can be observed at low frequencies in Figure 2. The influence on the high frequency components is negligible. In comparison, applying the logarithm to each data sample compresses the data. This makes the spectrum much smoother and key peaks can be distinguished easily - facilitating the perception of periodicities in the data. Subtracting the sample mean from the logarithmic data once again removes the DC offset from the PSD estimate.

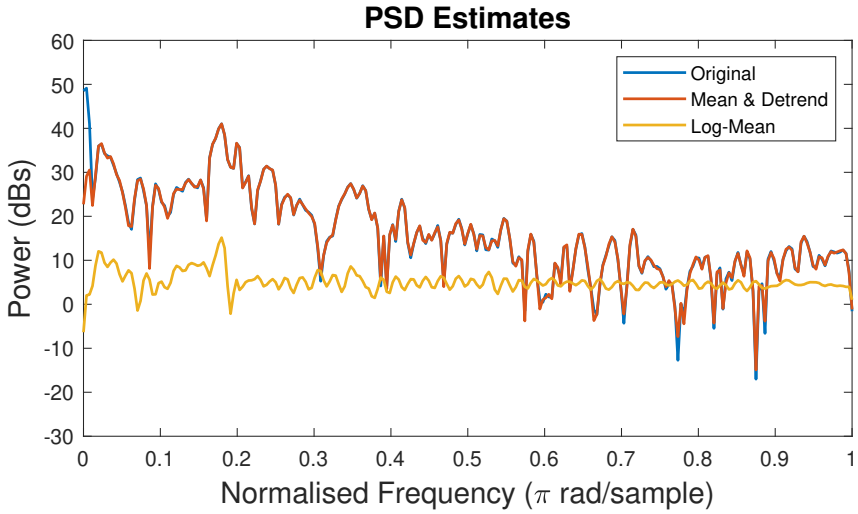

Figure 2: PSD estimates of the original and preprocessed sunspot time series

The basis for brain computer interface (BCI)

b) By obtaining the power spectrum of the EEG data, we aim to identify the the flashing rate of the visual stimulus, which produces a response in the EEG at the same frequency, known as the steady state visual evoked potential (SSVEP). This peak response frequency can be identified from Figure 3 as being 13Hz - where the peaks at 26Hz and 39Hz are harmonics of the response.

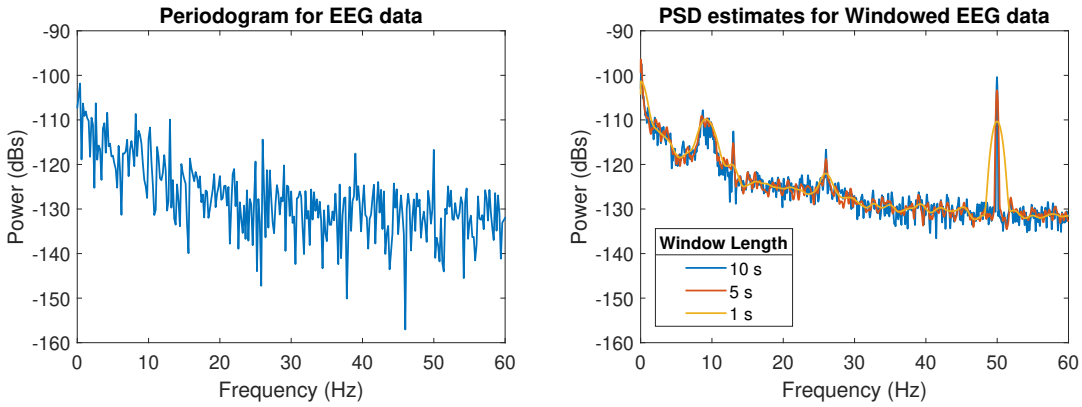

Figure 3: PSD estimates of the EEG data using the standard periodogram approach (left) and Welch's method with different window sizes (right)

Taking the averaged periodogram with window length of 10s greatly reduces the variance of the estimate compared with the standard periodogram, emphasizing the main peaks. However, the averaging process also introduces a reduction in spectral resolution, as it is observed by the 39Hz harmonic becoming indiscernible. This trade-off between variance and resolution is accentuated as the window length is decreased. Indeed, with a window length of 1s, the main 13Hz peak is not distinguishable from the alpha-rhythm in the 8-10Hz band. This highlights the importance of choosing an appropriate window length.

1.3 Correlation Estimation

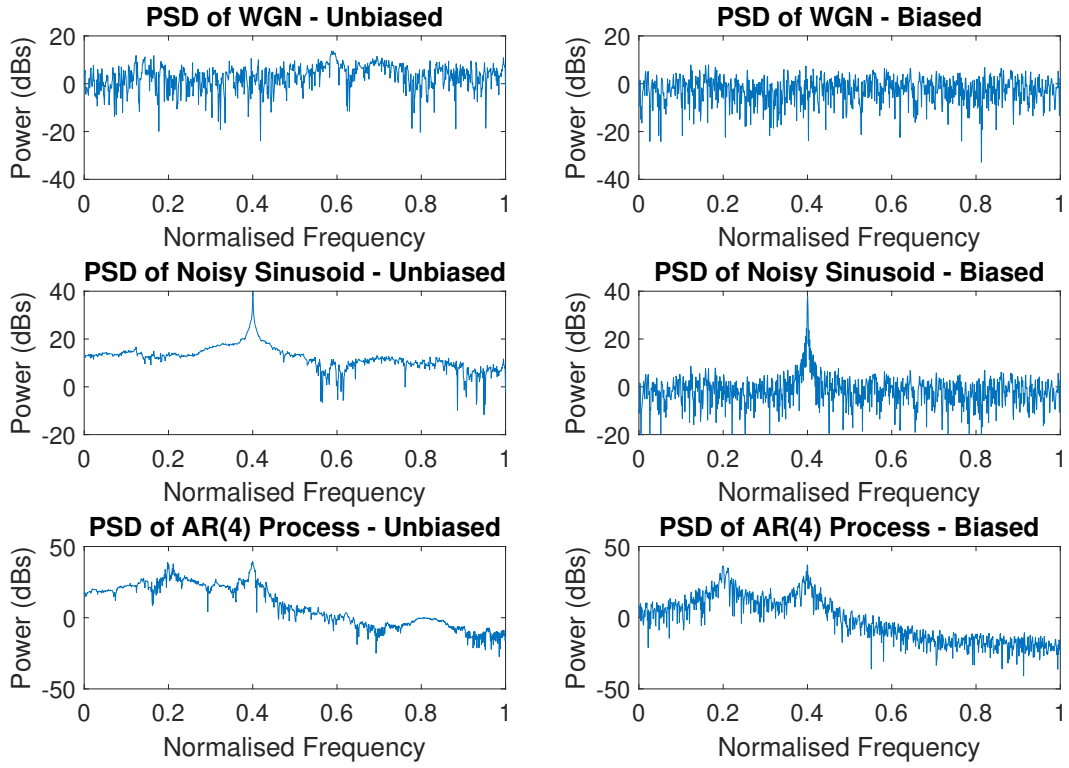

Figure 4: PSD estimates of test signals using unbiased (left) and biased (right) estimates of the auto-correlation function

a) Figure 4 shows the PSD estimates for three different signals - white Gaussian noise (WGN), a noisy sinusoid and filtered WGN - computed with the correlogram spectral estimator using unbiased (left) and biased (right) estimates of the autocorrelation.
The unbiased estimate shows erratic behaviour for larger lags (Figure 5) which leads to high variance in the PSD (Figure 4). The biased estimate is therefore used to reduce variance at the price of introducing a bias - however this estimate is asymptotically unbiased for large N. This is a simplified version of the Blackman-Turkey method.

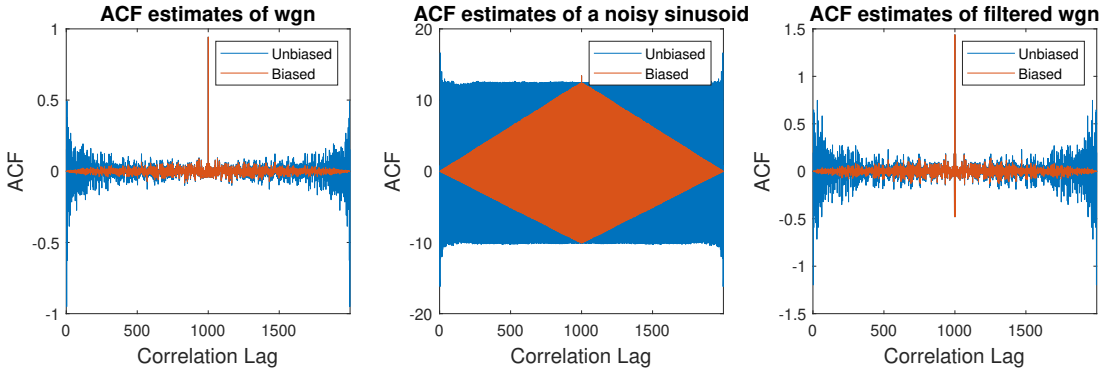

Figure 5: Autocorrelation function of test signals using unbiased and biased estimates of the autocorrelation function

b), c) Figure 6 shows the plot, mean and standard deviation of several realisations of signals containing two sine waves and corrupted by noise - plotted over a linear scale (top) and in dBs (bottom). We observe that the maximum deviation is seen at the main frequency peaks, at normalised frequencies of 0.1 and 0.4. Again, plotting in dBs compresses the signal (a property of the logarithm). This facilitates the visualisation of the different realisation and helps distinguish patterns in the noise (i.e. whether there is one or several prevalent frequency bands in the noise).

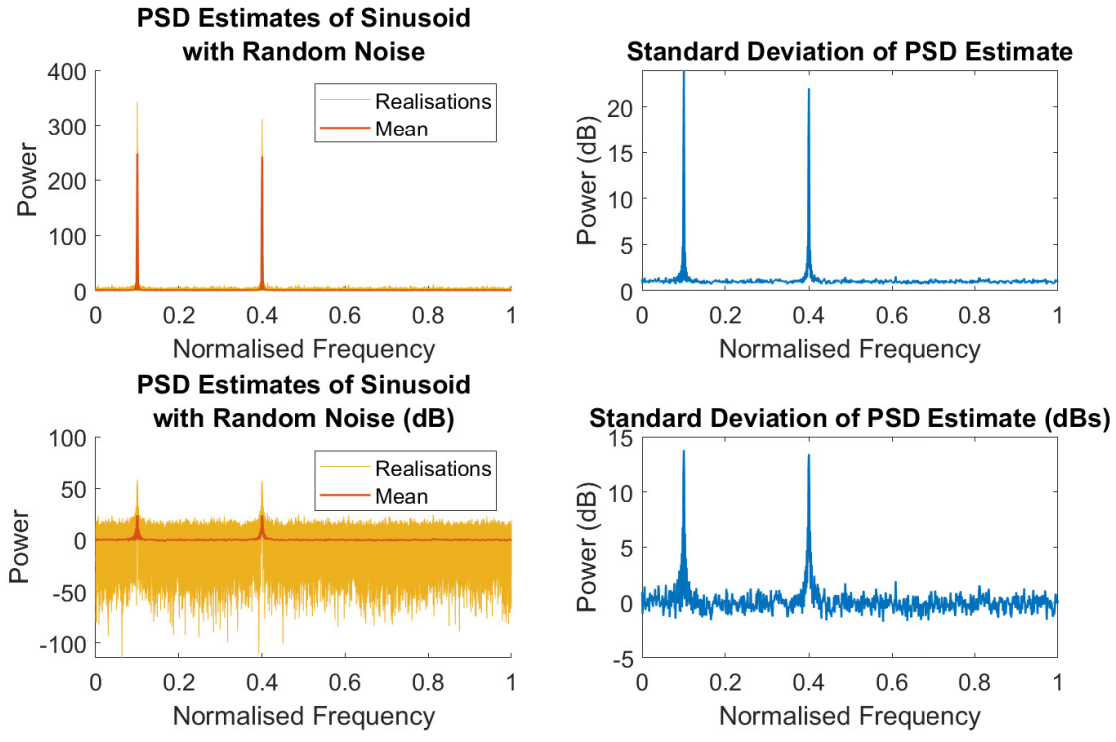

Figure 6: PSD estimates of 100 realisations noisy sinusoidal signals and their mean (left) and their standard deviation (right)

d) Figure 7 shows the periodogram of a noisy complex-valued signal containing two complex exponentials computed for different numbers of data samples. The main signal frequencies are 0.3Hz and 0.32Hz and the resolution of the estimates is given by the inverse of the number of data samples used. Indeed, the two peaks cannot be distinguished when the periodogram is computed with 30 samples in which case the resolution is only of $\frac{1}{N} = \frac{1}{30} = 0.033$Hz, which is bigger than the frequency difference between the peaks.

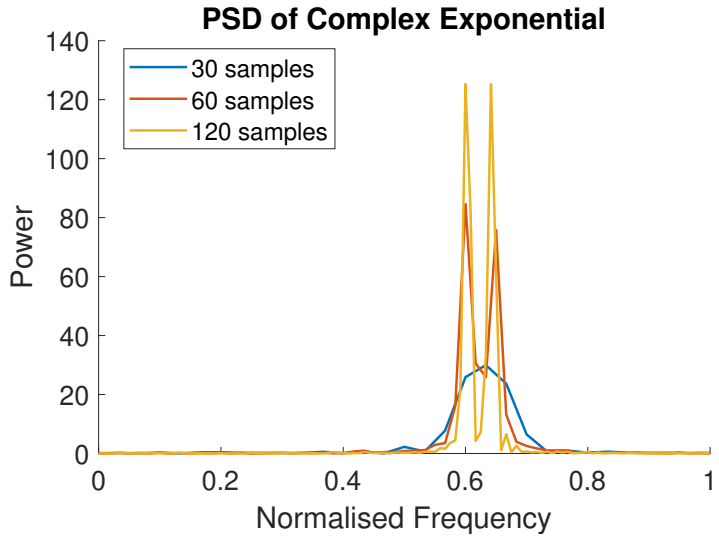

Figure 7: Periodogram of complex exponential computed considering different data lengths

e) In order to perform frequency estimations of a signal we can also use subspace methods such as the MUltiple Signal Classification (MUSIC) algorithm. This was performed in MATLAB using the functions `corrmtx` and `pmusic`. The function `corrmtx` is used to compute `R`, the biased autocorrelation matrix estimate found as the conjugate transpose of `X`. `R` is an *(m+1)by(m+1)* matrix, where m is the predicted model order, here 14. The `'mod'` argument means that the autocorrelation is derived using forward and backward prediction error estimates. The function `pmusic` performs the MUSIC algorithm, which assumes knowledge of the signal subspace dimension (p), here $p = 2$ as we have two complex exponentials in our input signal `x`. The argument `'corr'` forces the first argument to be interpreted as a correlation matrix rather than a matrix of signal data. The pseudospectrum is then plotted against the vector of frequencies `F` at which it is computed, within the range [0.25, 0.40]Hz ($f_s = 1Hz$).

Figure 8 displays MUSIC's pseudospectrum (left) and the periodogram estimate (right) for a noisy complex exponential with two frequencies (0.3, 0.32), the signal being of 30 data points in length. Since it exploits the knowledge of the signal subspace dimension, we observe that MUSIC outperforms the periodogram, with sharp peaks at the main frequencies. This is done by ranking the eigenvalues of the autocorrelation matrix and considering only the p biggest values for the estimation of the spectrum. As such, the biggest advantage of MUSIC is being able to correctly identify the main frequencies even with few samples, unlike the periodogram. However, MUSIC does not provide an accurate estimate of the magnitude of the power at the main frequencies, and has high magnitude variance at the frequency peaks.

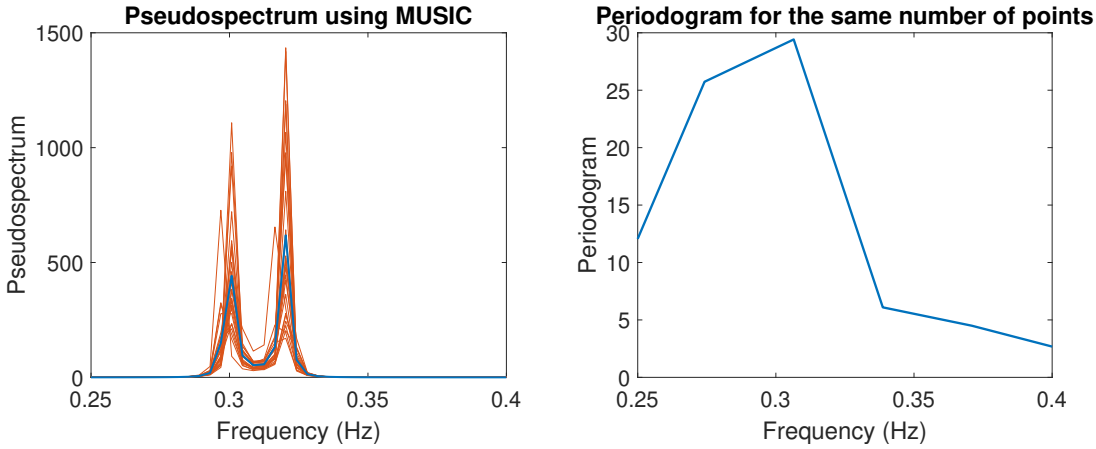

Figure 8: Pseudospectrum computed with the MUSIC algorithm of a corrupted exponential of 30 data samples in length with two main frequencies (left) and its periodogram (right)

1.4 Spectrum Estimation Process

The assumption of an underlying data model is the key difference between classical and modern spectrum estimation methods.

a) The set of autoregressive parameters can be found by means of the inverse Yule-Walker equation: $a = R_{xx}^{-1} r_{xx}$. This implies that the ACF matrix R_{xx} must be positive definite to guarantee matrix inversion. However, the unbiased estimate for the ACF does not guarantee the ACF being positive definite meaning the set of AR parameters might not be computable. This is linked to the fact that it can be highly erratic when computed for large lags (close to N) where fewer samples are available to estimate the PSD .

b), c) The autoregressive process is defined in equation 1.7. It was generated using 1000 and 10000 data samples, by filtering a random signal with the transfer function corresponding to the coefficients in the equation.

$$x(n) = 2.76x(n-1) - 3.81x(n-2) + 2.65x(n-3) - 0.92x(n-4) + w(n) \tag{1.7}$$

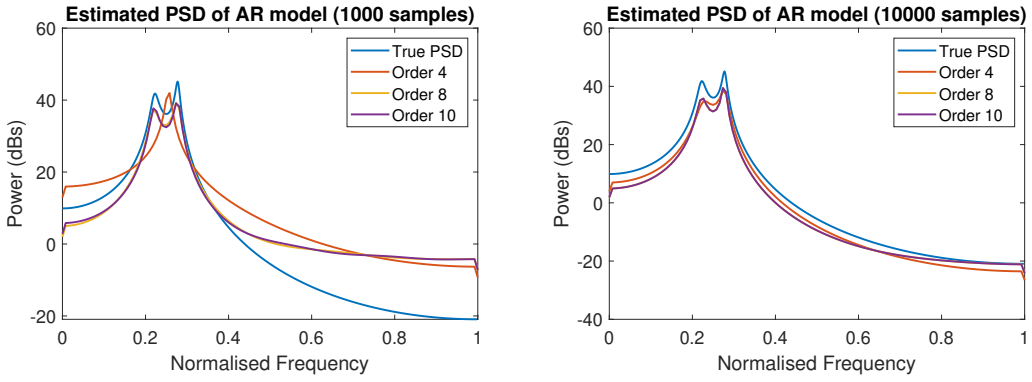

Figure 9: Periodogram of autoregressive model of order 4 computed assuming different model orders, considering a signal of 1000 samples (left) and of 10000 samples (right)

Figure 9 shows a comparison between the true PSD of the signal and different estimated PSDs using different model orders of the (assumed) model. We observe that the PSD estimates of 500 samples (1000 samples without the transient response) are not sufficient to correctly capture the true model order of 4 with the 4th order model. The two peaks are captured by models of higher orders (e.g. 8 and 10) at the price of overfitting - which can be observed by the small oscillations in power at higher frequencies - and higher computational complexity. None of the models were

accurate estimates of the PSD at higher frequencies. On the other hand, using 9500 sample for the estimates means that the 4th order model is able to capture two peaks in the PSD. This shows, as previously discussed, the importance of signal length for frequency resolution.

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

a) Figure 10 shows the standard and averaged periodogram PSD estimates of the three trials of RRI data, corresponding to ECG data where the patient is breathing unconstrained, at a fast pace (50 breaths per minute) and at a slow pace (15 BPM). The data was first preprocessed; the mean was subtracted and the result detrended. The PSD estimates were then obtained using the MATLAB function `pwelch`, with no overlap and Hamming windows of different lengths: the length of the data, 50s and 150s.

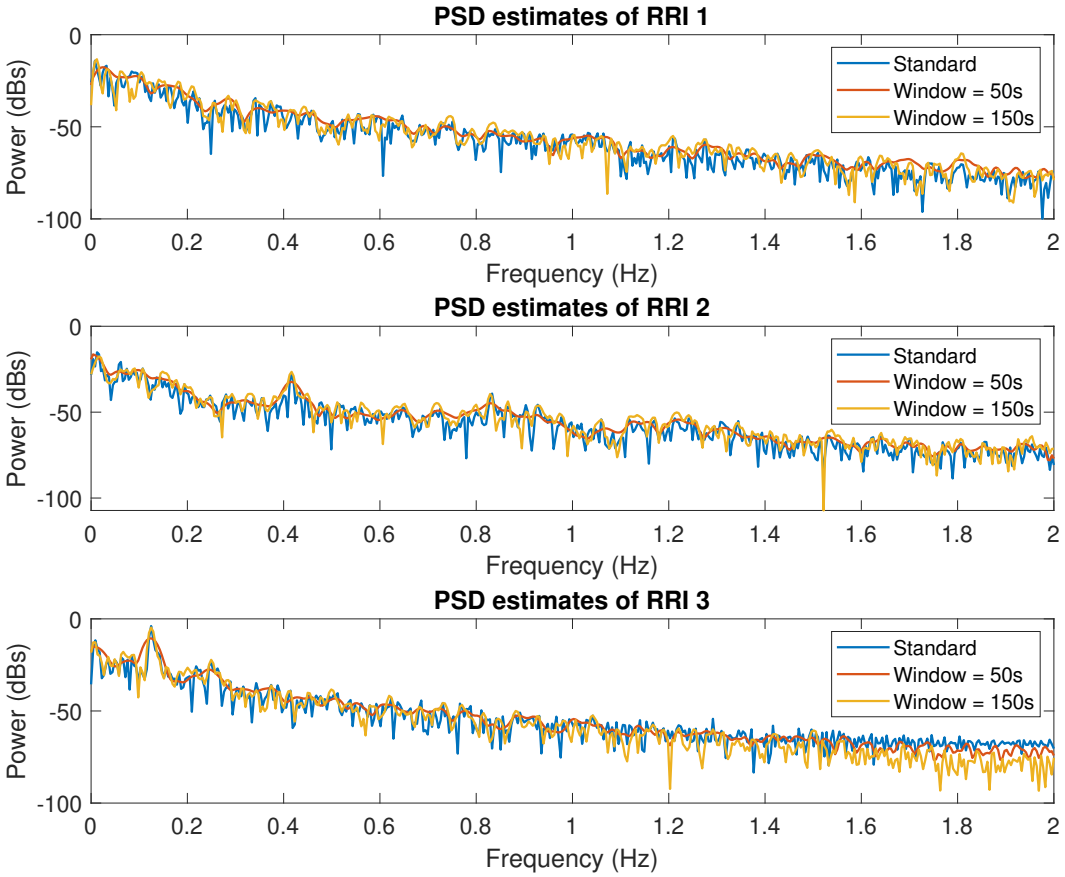

Figure 10: Estimates of the power spectral density using Bartlett's method of three trials of RRI data, each corresponding to different breathing types

b) Because respiration modulates the cardiac function, the primary frequency components of the RRI data should theoretically correspond to the patient's breathing rate, following the relationship: $\textit{breathing rate (BPM)} = 2 \cdot 60 \cdot f$. This is most evident in trials 2 and 3, with peaks at 0.42 Hz and 0.12 Hz, corresponding to fast and slow breathing paces of 50.4 BPM and 14.4 BPM, respectively. Harmonics can also be observed at 0.83 Hz and 1.2 Hz in trial 2 and at 0.25 Hz and 0.37 Hz in trial 3. In the PSD estimate of trial 1, it is harder to distinguish a main frequency because the breathing is unconstrained and might be irregular, but the frequency band with the most power was between 0.09 Hz and 0.21 Hz, corresponding to breathing rates between 11 BPM and 25 BPM, which are

considered to be normal[1]. Once again, we observe a trade off between frequency resolution and noise variance as the window size is decreased.

c) The optimal model orders were determined empirically by testing different orders with the MATLAB function `aryule`, which finds the AR coefficients of a data series given a model order. The optimal order for each trial was chosen to be the smallest one showing a peak at the main frequency identified in **b)**. This is shown in Figure 11. For trial 1, the broad peak observed corresponds to the low frequency band. For trial 2, the optimal model order was found to be as high as 9 because lower orders did not allow to discern any specific peak. The AR spectral estimate for trial 3 has a well discernible peak at the main frequency. Overall, AR models exhibit much less variance than the periodogram, showing smooth peaks at the main frequencies. However, they heavily rely on the correct selection of model order and may be inaccurate in terms of power magnitude.

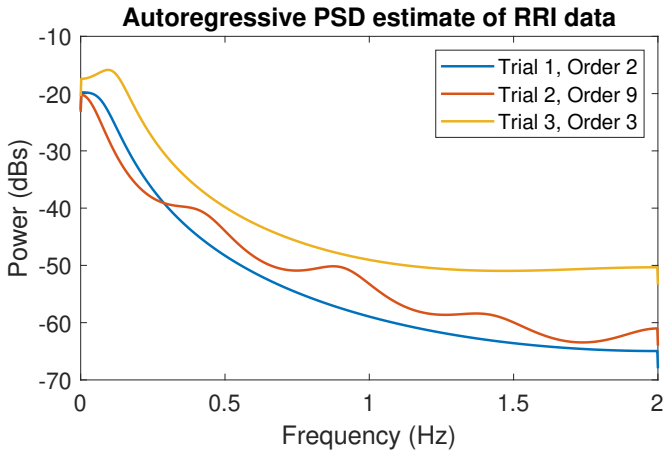

Figure 11: Autoregressive estimates of the power spectral density using of the three trials of RRI data.

1.6 Robust Regression

a) Singular value decomposition (SVD) allows to find the rank of a matrix by identifying its singular values (SVs). Figure 12 shows that the input, $\mathbf{X}$, has three non-zero singular values, making it a matrix of rank 3. Adding noise increases the rank of $\mathbf{X}_noise$ to 10, however the rank of $\mathbf{X}$ is still identifiable. If the noise were to get sufficiently larger, the SVs of the signal subspace would not be distinguishable from those of the noise subspace. Moreover, the square error between SVs is significantly larger for components in the noise subspace.

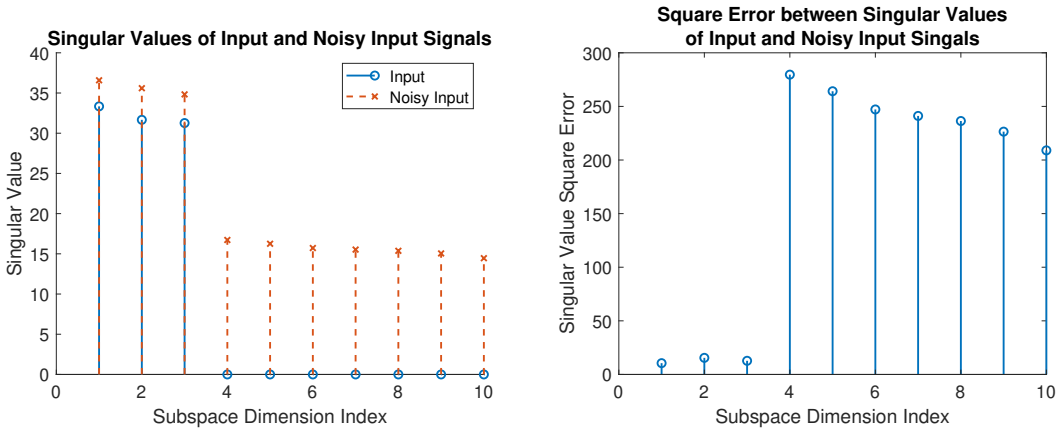

Figure 12: The singular values of the input matrix and of a noisy version of it (left) and the square error between these two matrices (right).

[1]Cleveland Clinic. 2022, *Vital Signs*

b) A low rank approximation matrix of $\mathbf{X}_{noise}$ was created using only the most significant principal components, which are 3, as determined by the rank of $\mathbf{X}$. Figure 13 shows that the error between $\mathbf{X}$ and the denoised version $\tilde{X}_{noise}$, on average 0.073, is much lower that the error between $\mathbf{X}$ and $\mathbf{X}_{noise}$, which is on average 0.24.

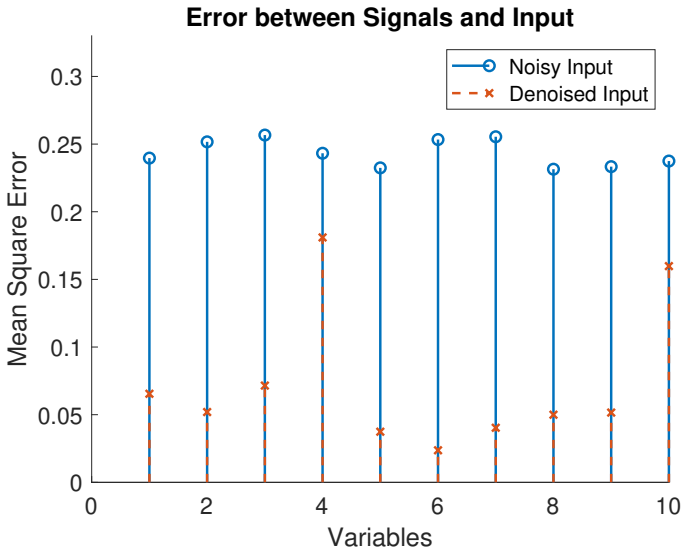

Figure 13: Error between variables of the noiseless input matrix and those in the noise corrupted matrix and denoised matrix.

c) The regression matrix $\mathbf{B}$ used to obtain the output data as $Y = XB + N_Y$ was computed using two different methods: the ordinary least square (OLS) estimate and the principal component regression (PCR) method - shown in Equations (1.8) and (1.9) respectively.

$$\hat{B}_{OLS} = (X^T X)^{-1} X^T Y \tag{1.8}$$

$$\hat{B}_{PCR} = V_{1:3}(\Sigma_{1:3})^{-1} U_{1:3}^T Y \tag{1.9}$$

As seen in Figure 14, results between the OLS and PCR were comparable for both the training and test datasets. The MSE of both methods is compared numerically in Table 1.

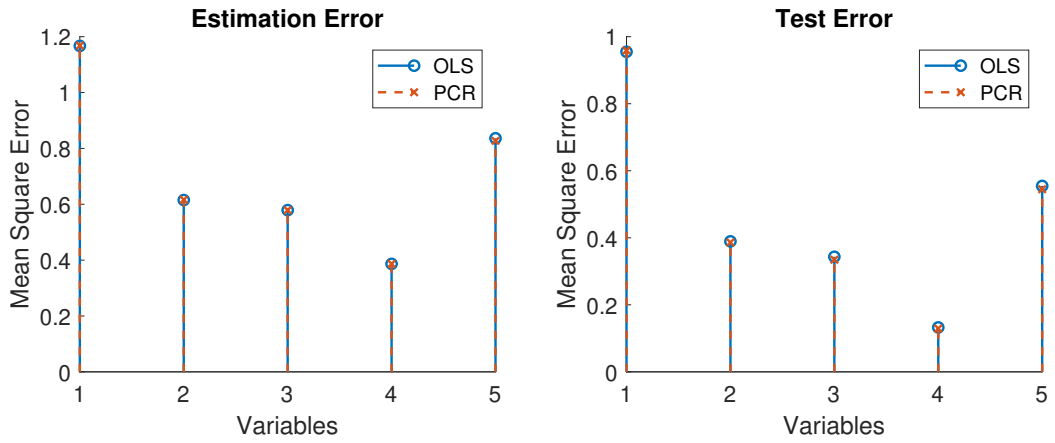

Figure 14: Mean square error of training (left) and test (right) data between estimates obtained from the OLS and PCR solutions.

	OLS	PCR
Training Mean Error	0.7168	0.7154
Testing Mean Error	0.4750	0.4708

Table 1: MSE for the OLS and PCR solutions in both training and testing.

d) The estimated regression coefficients computed with each method were further compared over an ensemble of test data using the provided `regval` script, which generates a new realisation of the data and its estimate, given the regression coefficients matrix. The comparison was run 100 times and Figure 15 shows the mean square error for each output. It can be seen that the methods are comparable in the estimation of the regression parameters, with the PCR method outperforming OLS by only 1.6%.

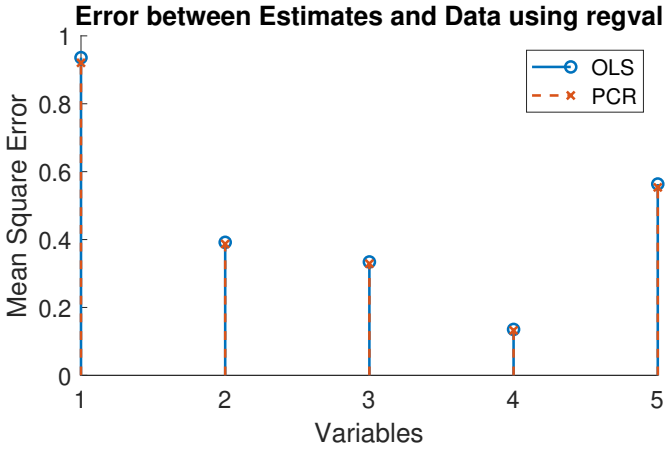

Figure 15: Mean square error of realisations of data and their estimates computed with `regval`, from the OLS and PCR methods.

2 Adaptive Signal Processing

2.1 The Least Mean Square (LMS) Algorithm

a) Given the second order autoregressive process satisfying the difference equation

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

where $a_1 = 0.1$, $a_2 = 0.8$ and $\eta(n) \sim \mathcal{N}(0,\ \sigma_\eta^2 = 0.25)$, we can estimate coefficients a_1 and a_2 with the Least Mean Square (LMS) algorithm, using the signal x(n). The weights, or coefficients, are updated iteratively as in (2.2), where $J(n)$ is a cost function.

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu \nabla_w J(n) \tag{2.2}$$

An important parameter of the algorithm is the step size μ. To ensure the convergence of the algorithm to the filter coefficients, the step size has to be within the interval $0 < \mu < \frac{2}{\lambda_{max}}$, where λ_{max} is the biggest eigenvalue of the correlation matrix $\mathbf{R_{xx}}$ of the input vector $\mathbf{x}_{IN}(n) = [x(n-1),\ x(n-2)]^T$.

$$\mathbf{R_{xx}} = \mathbb{E}\{[x(n-1),\ x(n-2)]^T [x(n-1),\ x(n-2)]\} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(1) & r_{xx}(0) \end{bmatrix} = \begin{bmatrix} \sigma_x^2 & r_{xx}(1) \\ r_{xx}(1) & \sigma_x^2 \end{bmatrix} \tag{2.3}$$

We find expressions for $r_{xx}(0)$ and $r_{xx}(1)$ by multiplying (2.1) by $x(n-k)$ and taking the expectation, noting that $\mathbb{E}\{x(n-k)\eta(n)\}$ vanishes when $k > 0$.

$$\begin{aligned} r_{xx}(0) &= a_1 r_{xx}(1) + a2 r_{xx}(2) + \sigma_\eta^2 \\ r_{xx}(1) &= a_1 r_{xx}(0) + a_2 r_{xx}(1) \\ r_{xx}(2) &= a_1 r_{xx}(1) + a_2 r_{xx}(0) \end{aligned} \tag{2.4}$$

We solve to find $\mathbf{R_{xx}} = \begin{bmatrix} 0.926 & 0.463 \\ 0.463 & 0.926 \end{bmatrix}$.
The eigenvalues of $\mathbf{R_{xx}}$ are $\lambda_1 = 0.463$ and $\lambda_2 = 1.389$, hence the LMS converges in the mean for the range of values $0 < \mu < 1.440$.

b) We implemented an LMS adaptive predictor using 1000 samples of the process in (2.1), with the custom function `LMS`. The weights are updated as in (2.2), with $J(n) = -e(n)\mathbf{x}_{IN}(n)$. Figure 16 shows the squared prediction error, $e^2(n) = x(n) - \mathbf{w}^T(n)\mathbf{x}_{IN}(n)$ in dB, for a single realisation (left) and the average over 100 realisations (right).

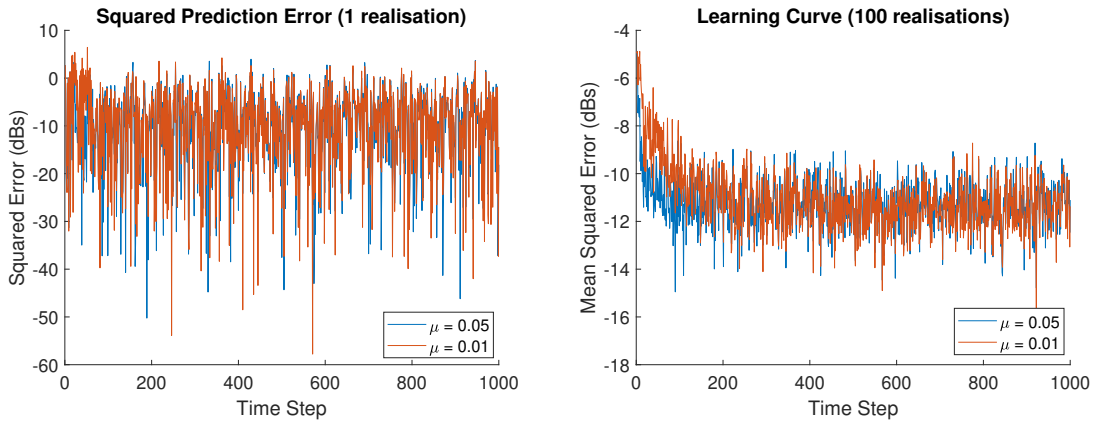

Figure 16: The squared prediction error of the implemented LMS algorithm for 1 realisation (left) and the average over 100 independent realisations (right), for different step sizes.

We observe from the learning curves that the solution converges faster with a larger step size but that there is greater steady state variability.

c) The misadjustment $\mathcal{M}$, is the ratio between the excess error introduced by the fluctuations around the optimal solution, EMSE, and the minimum achievable error signal, σ_η^2, as shown in (2.5). For small step sizes, the misadjustment of the LMS is approximated as in (2.6).

$$\mathcal{M} = \frac{\text{EMSE}}{\sigma_\eta^2} = \frac{\text{MSE} - \sigma_\eta^2}{\sigma_\eta^2} = \frac{\lim_{x\to\infty} \mathbb{E}\{e^2(n)\} - \sigma_\eta^2}{\sigma_\eta^2} \tag{2.5}$$

$$\mathcal{M}_{LMS} \approx \frac{\mu}{2}\mathbf{Tr}\{\mathbf{R}\} \tag{2.6}$$

We obtained estimated values for the misadjustment using the empirical mean of the steady state error squared as the MSE and substituting in (2.5). The results are presented in Table 2. As expected, a smaller step size leads to smaller values for the misadjustment as it introduces less fluctuations around the optimal solution. In addition, we note a bigger change between the two estimates for the bigger step size, possibly because the theoretical value is less well approximated by (2.6).

	$\mu = 0.05$	$\mu = 0.01$
$\mathcal{M}$	0.0127	0.0029
$\mathcal{M}_{LMS}$	0.0463	0.0093
Percentage Change	68.3%	72.7 %

Table 2: Estimated and theoretical misadjustment values of the LMS with step sizes $\mu = 0.05$ and $\mu = 0.01$.

d) We obtained the steady state values of the adaptive filter coefficients, as shown in Table 3. From the table, we observe that the LMS with smaller step size converges to a more accurate estimate of the filter coefficients. However, as previously mentioned and shown in Figure 17, the LMS with bigger step size converges faster. This highlights a trad-off between speed of convergence and variance.

	$\mu = 0.05$	$\mu = 0.01$
a_1	0.0819	0.0948
a_2	0.7685	0.7919
Euclidean Error	0.0363	0.0096

Table 3: Estimated steady state values of a_1 and a_2 for step sizes $\mu = 0.05$ and $\mu = 0.01$, and their associated error.

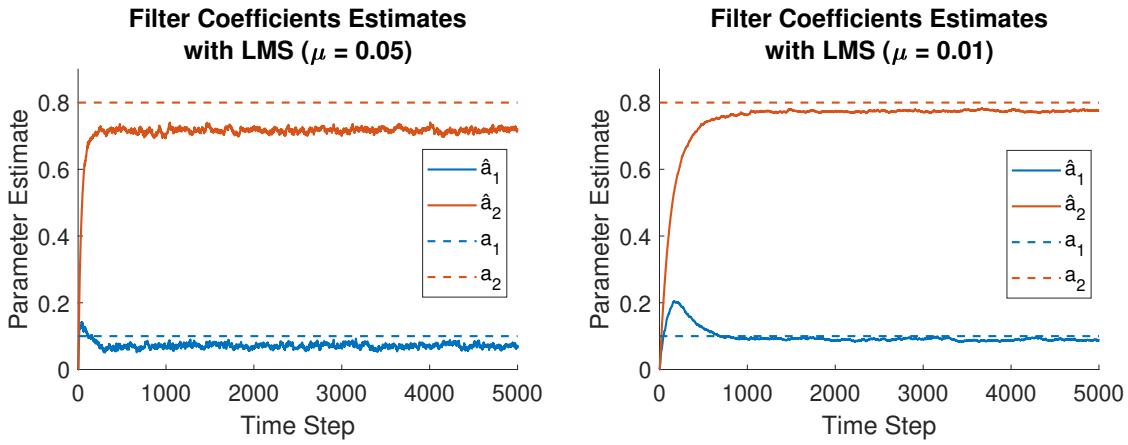

Figure 17: Average over 100 realisations of the evolution of the estimated filter coefficients over 5000 time steps for step sizes of $\mu = 0.05$ (left) and $\mu = 0.01$ (right).

The Leaky LMS Algorithm

e) The leaky LMS algorithm introduces a leakage coefficient $0 < \gamma < 1$ as a way to stabilise the algorithm when the autocorrelation matrix of the input signal $\mathbf{x}(n)$ has eigenvalue(s) of zero. We aim to derive the update rule for filter weights $\mathbf{w}(n)$, whose general form is shown in (2.7), that minimises the cost function in (2.8).

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu \nabla_{\mathbf{w}} J(n) \tag{2.7}$$

$$J_2(n) = \frac{1}{2}(e^2(n) + \gamma \|\mathbf{w}(n)\|_2^2) \tag{2.8}$$

We define the terms in (2.8), where $d(n)$ is the desired signal, and $x(n)$ the input signal. We then find the gradient of the cost function and substitute back into (2.7).

$$e(n) = d(n) - \mathbf{w}^T(n)\mathbf{x}(n)$$

$$\|w(n)\|_2 = \sqrt{\mathbf{w}^T(n)\mathbf{w}(n)}$$

$$\nabla_{\mathbf{w}} J_2(n) = \frac{1}{2}\left\{\frac{\partial (d(n) - \mathbf{w}^T\mathbf{x})^2}{\partial \mathbf{w}} + \gamma \frac{\partial \mathbf{w}^T\mathbf{w}}{\partial \mathbf{w}}\right\} = \frac{1}{2}(-2\mathbf{x}(d(n) - \mathbf{w}^T\mathbf{x}) + 2\gamma\mathbf{w}) = -\mathbf{x}e(n) + \gamma\mathbf{w}$$

$$\implies \mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{x}(n)e(n) - \mu\gamma\mathbf{w}(n) = (1 - \mu\gamma)\mathbf{w}(n) + \mu e(n)\mathbf{x}(n) \tag{2.9}$$

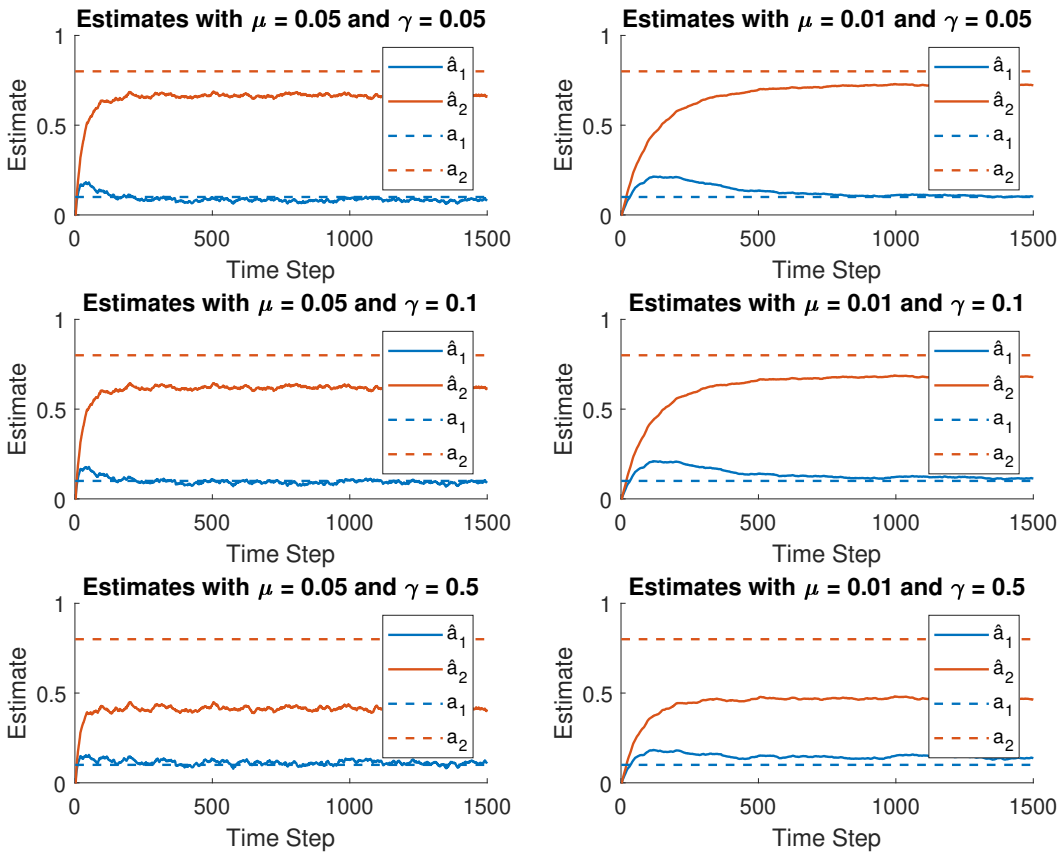

Figure 18: Estimates of AR(2) coefficients using different parameter combinations with the Leaky LMS algorithm.

f) Figure 18 shows the coefficients' estimates evolution using the leaky LMS algorithm with different learning rates and leakage coefficients. We know that the optimal coefficients are found

using the weight update rule $\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{x}(n)$, meaning that the weights of the leaky algorithm (2.9) do not correspond to the optimal ones. Hence, to achieve stability, the advantage of introducing a leakage, while reaching the best possible coefficient estimation, γ should be kept the small. This is confirmed by observation of Figure 18: increasing γ means estimates differ increasingly from the true value.

2.2 Adaptive Step Sizes

a) In Gradient Adaptive Step-Size (GASS) algorithms, the learning rate is updated according to

$$\mu(n+1) = \mu(n) + \rho e(n)\mathbf{x}^T(n)\psi(n) \tag{2.10}$$

where the term $\psi(n)$ can take one of the following forms

$$\text{Benveniste}: \ \psi(\mathbf{n}) = \Big[\mathbf{I} - \mu(n-1)\mathbf{x}(n-1)\mathbf{x}^T(n-1)\Big]\psi(n-1) + e(n-1)\mathbf{x}(n-1) \tag{2.11}$$

$$\text{Ang \& Farhang}: \ \psi(\mathbf{n}) = \alpha\psi(n-1) + e(n-1)\mathbf{x}(n-1) \tag{2.12}$$

$$\text{Matthews \& Xie}: \ e(n-1)\mathbf{x}(n-1) \tag{2.13}$$

The weight error curves $\tilde{w}(n) = w_o - w(n)$ (where $w_o = 0.9$) in Figure 19 are used to compare the performance of these three algorithms and the LMS when identifying the real-valued MA(1) process in (2.14). These were obtained by averaging the weight curves over 100 realisations, with the hyper-parameters being the step-size adaptation parameter $\rho = 0.005$ and $\alpha = 0.8$.

$$x(n) = 0.9\eta(n-1) + \eta(n) \quad \eta \sim \mathbb{N}(0, 0.5) \tag{2.14}$$

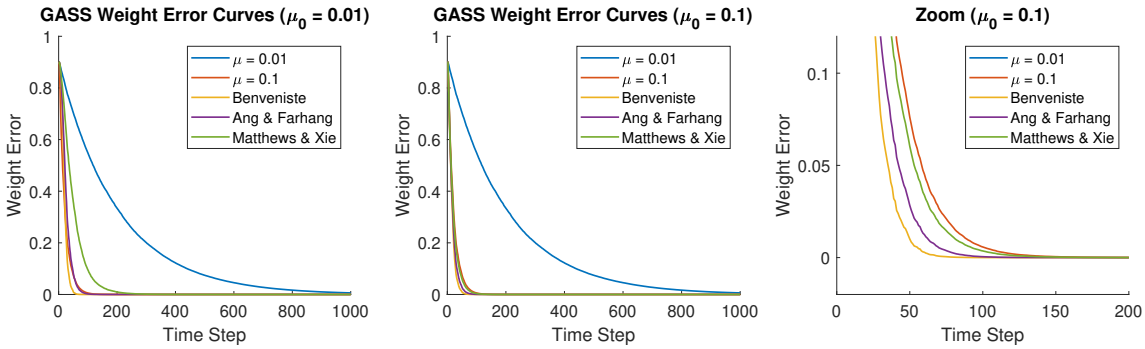

Figure 19: Comparison of the weight error curves of the standard LMS against GASS-LMS algorithms for different initial learning rates μ_0.

The three GASS algorithms converge faster than the LMS with step sizes $\mu = 0.01$ and $\mu = 0.1$ provided that the initial step size is large enough (e.g. $\mu_0 = 0.1$). In particular, the Benveniste algorithm converges the fastest, followed by Ang & Farhang and then Matthews & Xie. In addition, the GASS algorithms present a smaller steady state error (Table 4), as they adaptively reduce the learning rate when reaching the solution.

Algorithm	LMS ($\mu = 0.01$)	LMS ($\mu = 0.1$)	B	A & F	M & X
$\mathcal{O}$(Steady State Error)	10^{-5}	10^{-14}	10^{-30}	10^{-20}	10^{-14}

Table 4: Comparison of the order of the steady state error of the standard LMS against GASS-LMS algorithms with initial learning rates $\mu_0 = 0.1$.

Because of their faster and lower steady state error, the GASS algorithms can be said to have a superior performance with respect to the standard LMS. However, their principal disadvantage compared to the LMS is their increased computational complexity due to adapting the step size after each step.

b) The normalised LMS algorithm (NLMS), shown in (2.15), is used to normalise the step size, to avoid the weight update from being proportional to the input vector.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\beta}{\epsilon + ||\mathbf{x}(n)||^2} e(n)\mathbf{x}(n) \tag{2.15}$$

To verify that the update equation $\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e_p(n)\mathbf{x}(n)$ based upon the *a posteriori* error $e_p(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n+1)$ is equivalent to the NLMS algorithm, we start by rewriting the expression for e_p recursively, using the update rule. Note that $e(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n)$. We then substitute the result back into the update rule to obtain a form of (2.15).

$$\begin{aligned}
e_p(n) &= d(n) - \mathbf{x}^T(n)\mathbf{w}(n+1) \\
e_p(n) &= d(n) - \mathbf{x}^T(n)[w(n) + \mu e_p(n)\mathbf{x}(n)] \\
e_p(n) &= d(n) - \mathbf{x}^T(n)w(n) - \mathbf{x}^T(n)\mu e_p(n)\mathbf{x}(n) \\
e_p(n) &= e(n) - \mathbf{x}^T(n)\mu e_p(n)\mathbf{x}(n) \\
e_p(n) &= \frac{e(n)}{1 + \mu\mathbf{x}^T(n)\mathbf{x}(n)} = \frac{e(n)}{1 + \mu||\mathbf{x}(n)||^2}
\end{aligned}$$

Substituting back into the update rule:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\frac{e(n)}{1 + \mu||\mathbf{x}(n)||^2}\mathbf{x}(n) = \mathbf{w}(n) + \frac{e(n)}{\frac{1}{\mu} + ||\mathbf{x}(n)||^2}\mathbf{x}(n) \tag{2.16}$$

Setting the regularisation factor $\epsilon = \frac{1}{\mu}$, and $\beta = 1$, makes equations (2.15) and (2.16) equivalent.

c) The Generalised Normalised Gradient Descent (GNGD) algorithm makes the regularisation factor ϵ adaptive following (2.17). Its performance was compared against Benveniste algorithm and the results are seen in 20. The complexity of the Benveniste GASS algorithm is $\mathcal{O}(M^2)$ due to the outer product arising in the computation of ψ. On the other hand, the complexity of GNGD is $\mathcal{O}(M)$, as there is no outer product in (2.17). Therefore, the GNGD converges faster, reaching a comparable steady state error.

$$\epsilon(n+1) = \epsilon(n) - \rho\mu\frac{e(n)e(n-1)\mathbf{x}^T(n)\mathbf{x}^T(n-1)}{(\epsilon(n-1) + ||\mathbf{x}(n-1)||^2)^2} \tag{2.17}$$

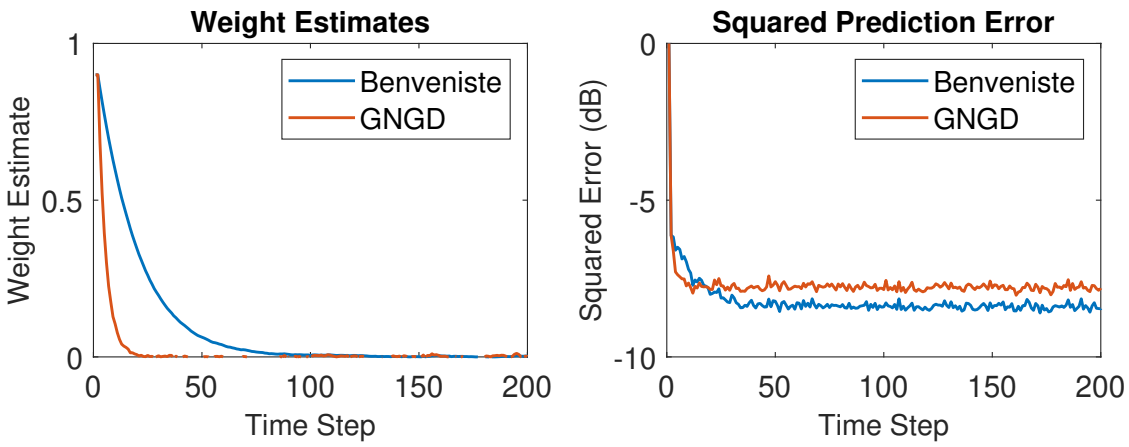

Figure 20: Weight error and squared prediction error curves for GNGD and Benveniste (GASS) algorithms.

2.3 Adaptive Noise Cancellation

In this section, we will discuss noise suppression using two de-noising adaptive filtering configurations: the adaptive line enhancer (ALE) and adaptive noise cancellation (ANC). For both noise suppression configurations, the prediction error $\mathcal{E}(n)$ is defined as the difference between the clean signal, $x(n)$ and the estimated de-noised signal $\hat{x}(n)$ - obtained through the algorithm from the noise corrupted signal $s(n) = x(n) + \eta(n)$. Here, $x(n)$ is a unit amplitude sinusoid of angular frequency $\omega_0 = 0.01\pi$ and η is coloured noise generated using a moving average filter on white Gaussian noise $v(n)$ as in (2.18).

$$\eta(n) = v(n) + 0.5v(n-2) \tag{2.18}$$

a) The delay, Δ must be chosen such that the noise in the signal $s(n)$ and the predictor input $\mathbf{u}(n) = [s(n-\Delta),...,s(n-\Delta-M+1]^T$ are uncorrelated, so that only the noise is suppressed by the linear predictor. To analytically find the minimum value of the delay Δ that may be used in the ALE given the model for noise in (2.18) and for a filter of length $M > 1$, we start by expanding the Mean Square Error $\mathbb{E}\{(s(n) - \hat{x}(n))^2\}$.

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

Using the linearity of the expectation operator to rearrange, we find

$$\mathbb{E}\Big\{(s(n) - \hat{x}(n))^2\Big\} = \mathbb{E}\Big\{(x(n) - \hat{x}(n))^2\Big\} + \mathbb{E}\Big\{\eta^2(n)\Big\} + 2\mathbb{E}\Big\{x(n)\eta(n)\Big\} - 2\mathbb{E}\Big\{\hat{x}(n)\eta(n)\Big\} \tag{2.19}$$

The first three terms do not depend on Δ; the first term corresponds to the true mean square prediction error and does not impact the noise signal $\eta(n)$, the second term is a constant equivalent to the variance of the noise (it is the minimum achievable cost) and the third term is 0, since $x(n)$ is deterministic with mean zero. Hence, to find the minimum value for the delay Δ, we focus on the last term, which should be zero for uncorrelated $\eta(n)$ and $\hat{x}(n)$.

$$-2\mathbb{E}\Big\{\hat{x}(n)\eta(n)\Big\} = -2\mathbb{E}\Big\{(\mathbf{w}^T(n)\mathbf{u}(n))\eta(n)\Big\} = -2\mathbf{w}^T\begin{bmatrix}\mathbb{E}\{(x(n-\Delta)+\eta(n-\Delta))\eta(n)\} \\ \vdots \\ \mathbb{E}\Big\{(x(n-\Delta-M+1)+\eta(n-\Delta-M+1))\eta(n)\Big\}\end{bmatrix}$$

We solve the system setting $k \in [1, M]$, knowing that the clean signal $x(n)$ and the noise signal $\eta(n)$ are uncorrelated.

$$\begin{aligned}&-2\mathbf{w}^T\mathbb{E}\Big\{x(n-\Delta)\eta(n)\Big\} - 2\mathbf{w}^T\mathbb{E}\Big\{\eta(n-\Delta)\eta(n)\Big\} \\ =&0 - 2\mathbf{w}^T\mathbb{E}\{[v(n-\Delta-k+1) + 0.5v(n-2-\Delta-k+1)][v(n) + 0.5v(n-2)]\}\end{aligned} \tag{2.20}$$

The minimum delay that can ensure no overlap between the two factors in (2.20), making it go to 0, is $\Delta_{min} = 3$. Figure 21 justifies this choice. On the left, we see that the autocorrelation function of $s(n)$ is dominated by the signal's sinusoid for delays bigger or equal than 3, and on the right we see a drop in prediction error from $\Delta = 3$.

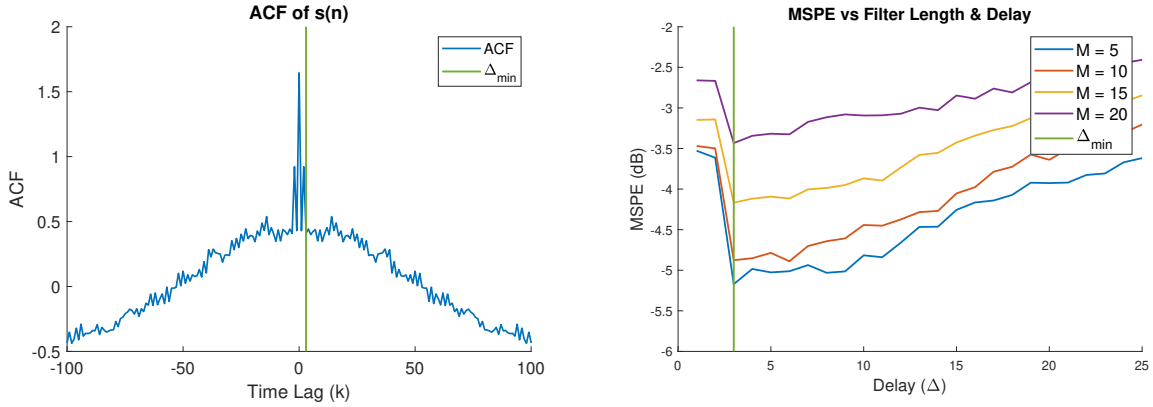

Figure 21: Autocorrelation function of $s(n)$ (left) and Mean Square Prediction Error as a function of delay Δ and filter length M.

b) The mean square prediction error (MSPE) is found as $\frac{1}{N}\sum_{n=0}^{N-1}\mathcal{E}^2(n) = frac1N\sum_{n=0}^{N-1}(x(n) - \hat{x}(n))^2$. Figure 21 (right) that an increase in delay from Δ_{min} leads to an increase in the MSPE, likely a result of overfitting. Similarly, an increase in model order M makes the MSPE increase. Hence, a good choice for filter order would be $M = 5$.

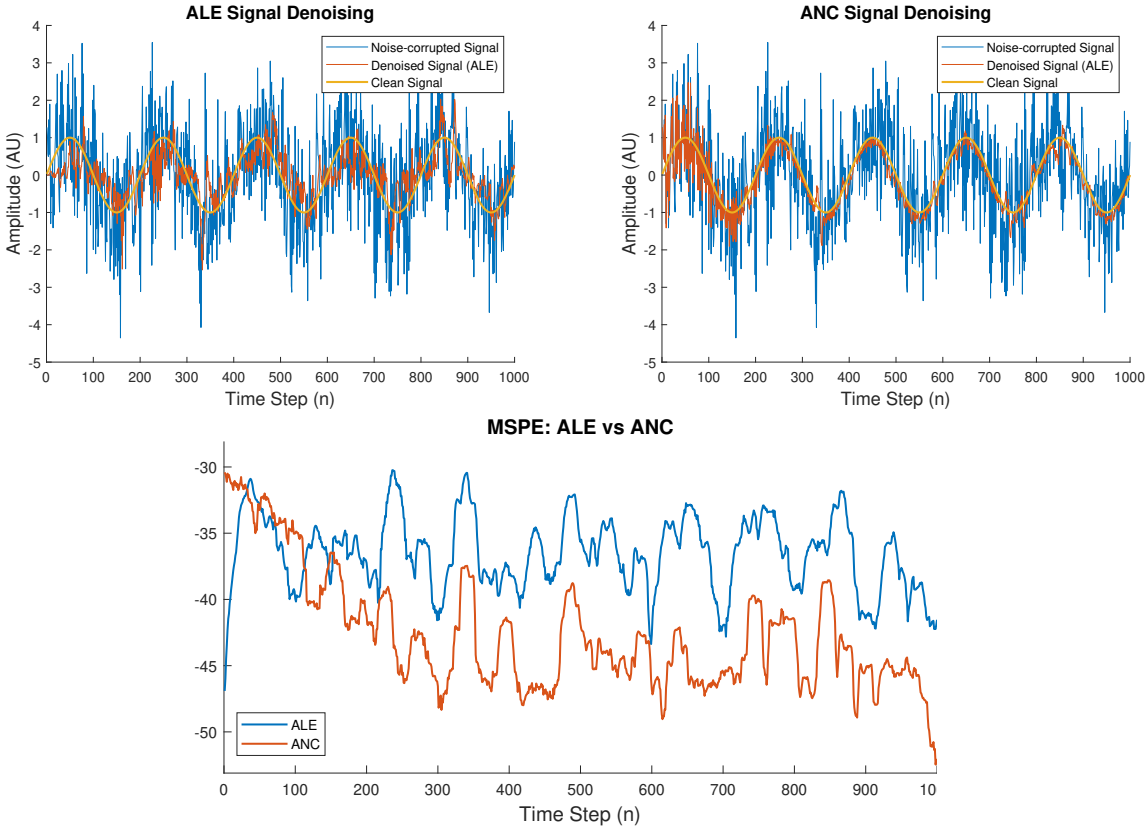

Figure 22: De-noised signal by ALE and ANC algorithms against the noisy signal and the original signal (top). Performance comparison of the two algorithms using the MSPE.

c) For adaptive noise cancellation (ANC), the input of the linear predictor (in our case, LMS) is a secondary noise input $\epsilon(n)$, that must be correlated with the primary noise input $\eta(n)$ (2.21).

$$\epsilon(n) = \eta(n) + 0.5\eta(n-1) + 0.2 \tag{2.21}$$

We compare the performances of ALE and ANC algorithms in Figure 22. ANC performs better if the signal has a long enough duration. However, we note that ANC requires a secondary noise input

correlated to the primary noise input, which may be difficult or impossible to obtain depending on the application.

d) We pass EEG data containing 50Hz mains noise through the ANC configuration, using a synthetic secondary noise input composed of a sinusoid of 50Hz corrupted by white Gaussian noise. The performance of the algorithm is made visual in Figure 23 which shows the spectrum of the corrupted (left) and denoised (right) EEG data. This was obtained using MATLAB's `spectrogram` function with Hamming windows of $L = 3540$ and 33% overlap. We carried out a grid search to find the optimal filter length and step-size, which we found to be $M = 40$ and $\mu = 0.01$, respectively. Figure 23 (right) shows that the 50Hz noise signal was indeed removed, and the SSVEP at 13Hz (see Section 1.2) and its harmonics are visible.

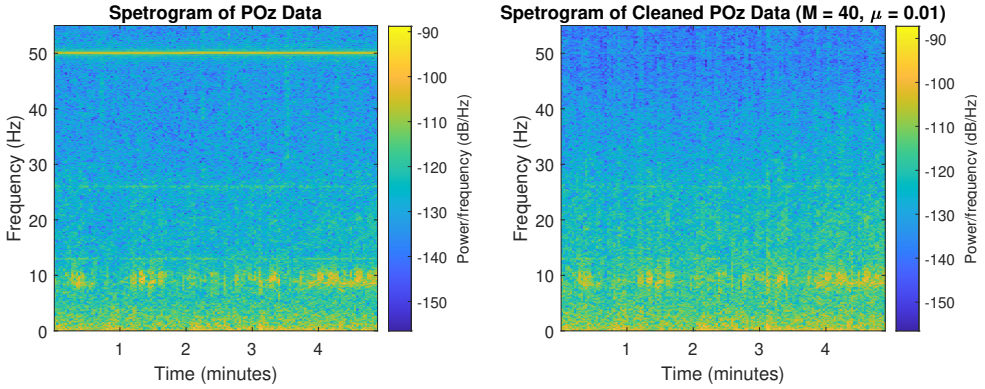

Figure 23: Spectrum of the original corrupted EEG signal (left) and of the denoised signal using ANC (right) with the optimal parameters.

Figure 24 shows the effect of the filter length on the denoising algorithm. With a low filter length, ANC does not successfully remove the mains noise, while a higher filter length leads to distortions in the spectrogram which affects the SSVEP (and its harmonics).

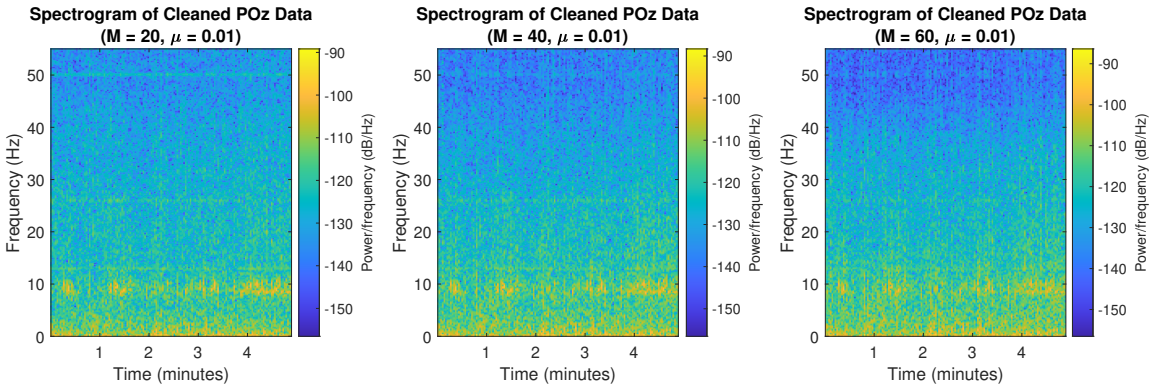

Figure 24: Spectrum of the signal denoised using ANC with different filter lengths.

Figure 25 shows the effect of the learning rate or step size on the denoising algorithm. Lower step sizes successfully denoise the signal, while higher step sizes do not and cause significant distortions in the spectrum.

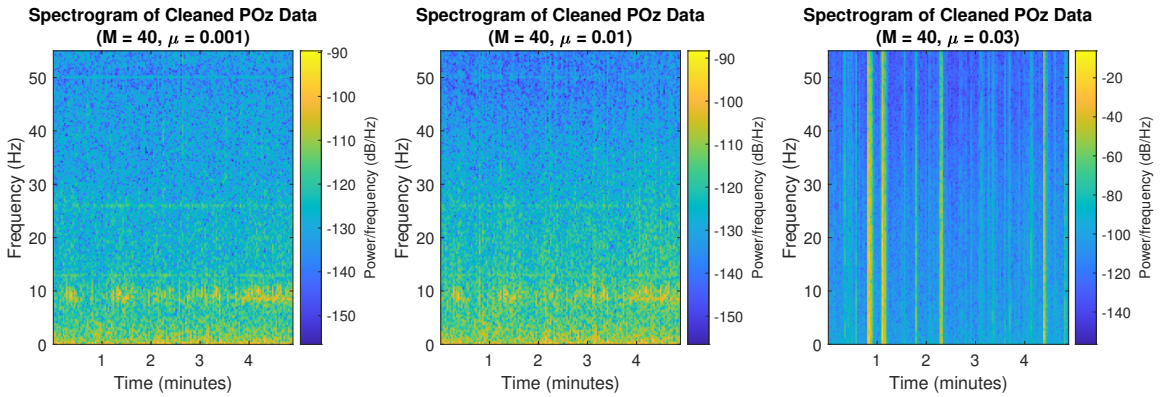

Figure 25: Spectrum of the signal denoised using ANC with different step sizes.

3 Widely Linear Filtering and Adaptive Spectrum Estimation

3.1 Complex LMS and Widely Linear Modelling

a) We generated the first-order Widely-Linear Moving Average process, WLMA(1) driven by circular white Gaussian noise $x(n)$ as in (3.1)

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

Figure 26 shows the learning curves, $10\log|e(n)|^2$, for the complex LMS (CLMS) and the Augmented CLMS (ACLMS) for the WLMA(1) process in (3.1). As expected, the CLMS does not learn the weights b_1 and b_2 successfully, hence it has a steady state error that remains at approximately 8.8dB. This is due to the CLMS acting as a generic extension of its real counterpart, the LMS, making it only a valid approach for circular random variables, i.e. with rotation invariant probability distributions. We know this is not the case for the WLMA(1) process, which has a circularity coefficient of $|\rho| = 0.86$ and whose circularity plot is shown in Figure 26 (bottom right). For comparison, the circularity plot of the circular white Gaussian noise has been plotted next to it. The ACLMS on the other hand, uses employs the widely linear framework. It uses an augmented input vector $\mathbf{x}^a = [\mathbf{x}^T \ \mathbf{x}^H]^T$, making it equipped with sufficient degrees of freedom to fully exploit the second order statistics of the data. Therefore, the steady state error of the ACLMS is much lower, approximately -307dB.

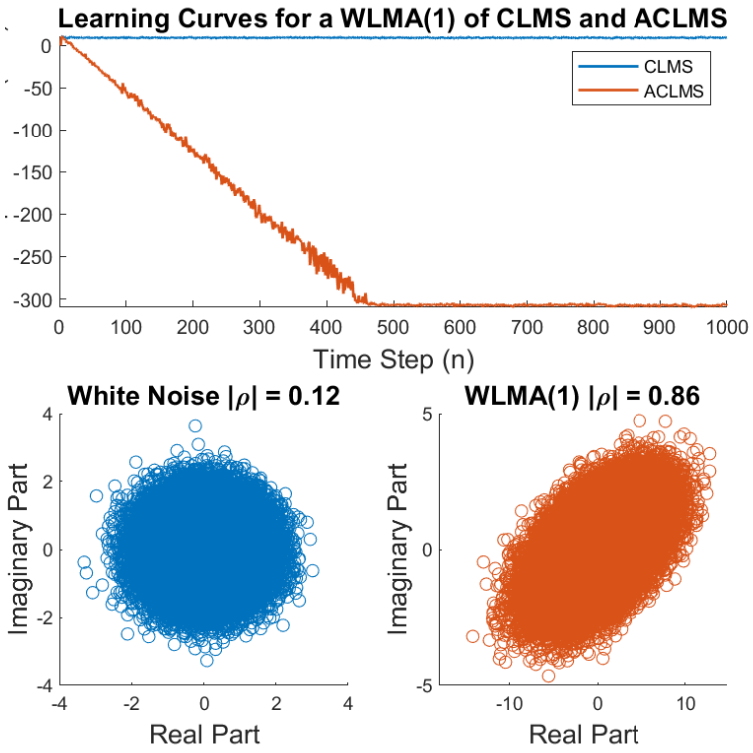

Figure 26: Learning curves averaged over 100 realisations for the CLMS and ACLMS for a WLMA(1) process (top). Circularity plots of white Gaussian noise and a WLMA(1) driven by the white Gaussian noise (bottom).

b) The circularity plots of wind data for the three wind regimes represented as complex valued signals, $v_k[n] = v_{k,east}[n] + jv_{k,north}[n]$, are shown in Figure 27. The degree of non-circularity is represented by the circularity coefficients, which were respectively found to be $|\rho|_{High} = 0.62$, $|\rho|_{Medium} = 0.45$ and $|\rho|_{Low} = 0.16$. Therefore, the low regime can be treated as circular or proper, while the medium and high regimes are treated as improper.

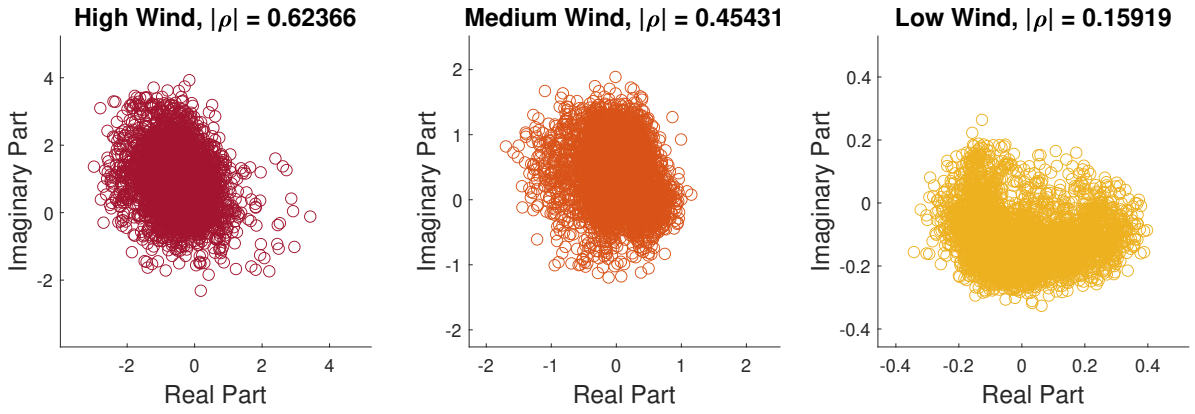

Figure 27: Circularity plots of wind data for the three wind regimes.

The CLMS and ACLMS filters were configured in a prediction setting to perform a one step ahead prediction of the complex wind data. The performance of the two algorithms on the wind regimes was evaluated as a function of filter length $Min[1,20]$ for different step sizes. In Figure 28 (top), we used the same step size $\mu = 0.001$ for the three regimes. As expected, the ACLMS performs better for the high wind regime, and the CLMS performs increasingly similarly to the ACLMS as the data becomes more proper. To contrast this, in Figure 28 (bottom) we show the performance of the two algorithms with the optimal step size for each wind regime, i.e. the one leading to the lowest MSPE for the optimal filter length. The ACLMS performs better than the CLMS in the three cases. Due to its higher number of degrees of freedom, the ACLMS is more prone to overfitting with larger filter lengths M, resulting in a higher MSPE.

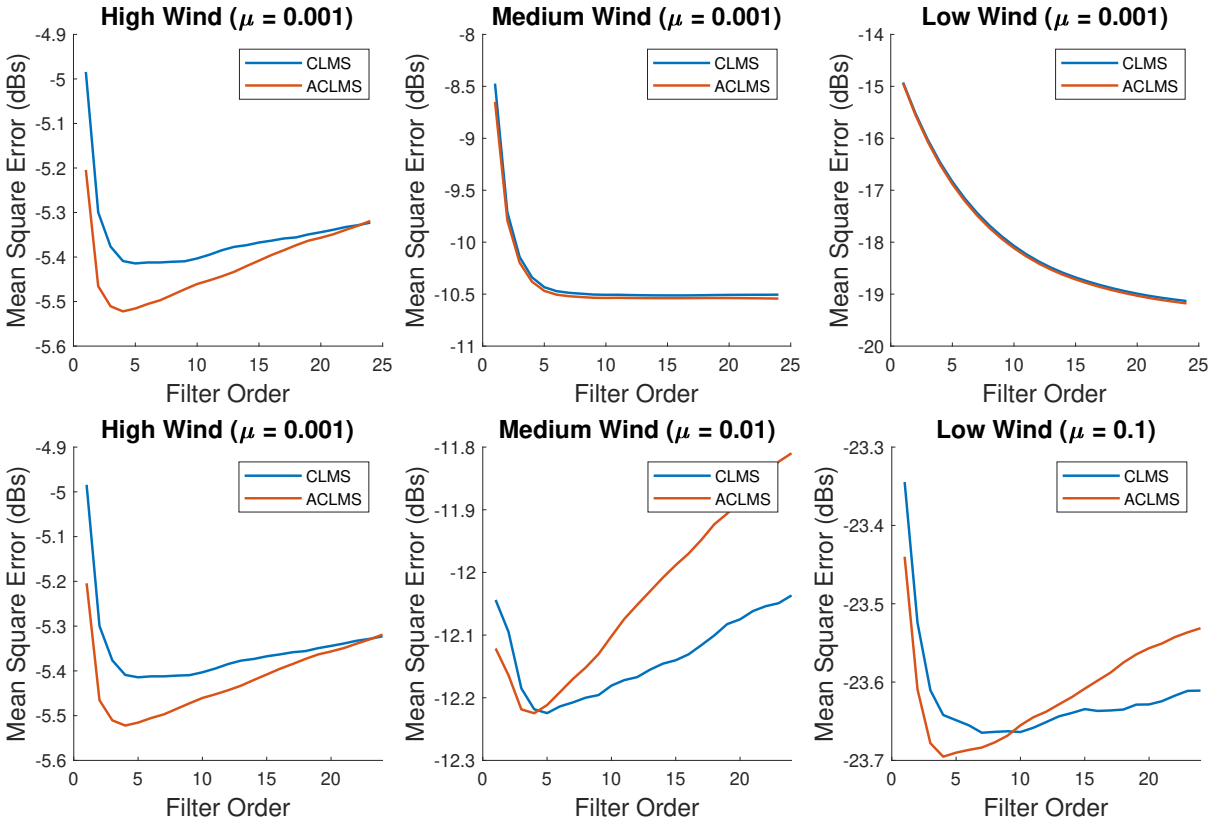

Figure 28: MSPE as a function of filter order for different wind regimes and filters with $\mu = 0.001$ (top) and optimal μ (bottom).

Frequency Estimation in Three Phase Power Systems

c) A three phase power system can be represented by a vector $[v_a(n), v_b(n), v_c(n)]^T$, which can be projected using the Clarke transform onto two orthogonal axes, forming the components v_α and v_β. The α and β components can be represented as a complex voltage, as shown shown in (3.2) for a balanced system and in (3.3) for an unbalanced system

$$v(n) = \sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_o}{f_s} n + \phi)} \tag{3.2}$$

$$v(n) = A(n)e^{j(2\pi \frac{f_o}{f_s} n + \phi)} + B(n)e^{-j(2\pi \frac{f_o}{f_s} n + \phi)} \tag{3.3}$$

where $A(n) = \frac{\sqrt{6}}{6}(V_a(n) + V_b(n)e^{j\Delta_b} + V_c(n)e^{j\Delta_c})$ and $B(n) = \frac{\sqrt{6}}{6}(V_a(n) + V_b(n)e^{-j(\Delta_b + 2\pi/3)} + V_c(n)e^{-j(\Delta_c - 2\pi/3)})$. When the system is balanced, implying $V_a(n) = V_b(n) = V_c(n) = V$ and $\Delta_b = \Delta_c = 0$, the complex-valued representation of the system is circular, as shown in Figure 29 (left). When the voltage amplitudes are made to differ, as in Figure 3.2 (middle) or one/both of the phase distortion parameters are non-zero, as in Figure 29 (right), the complex voltage representations are not circular. Therefore, the circularity diagrams can be used to identify a fault in the system by measuring the deviation from circularity.

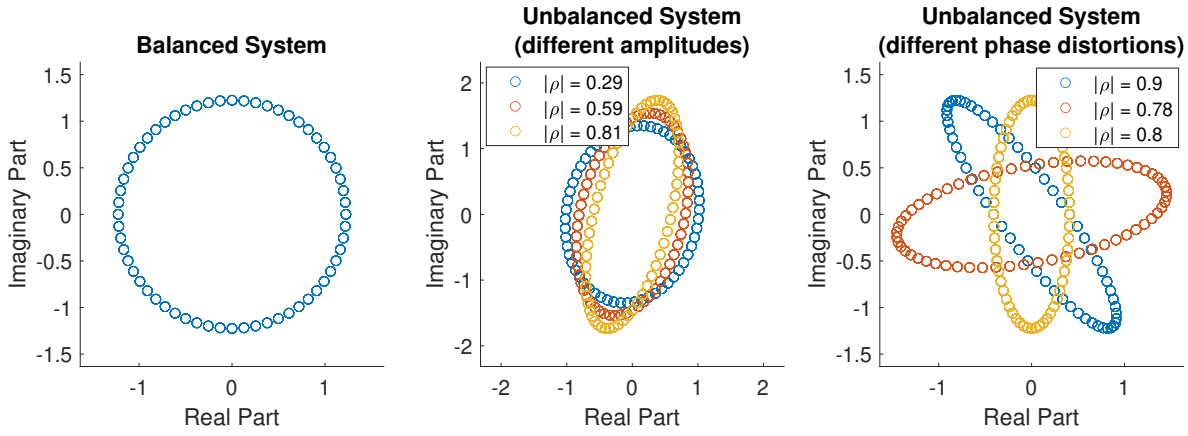

Figure 29: Circularity plots of a balanced (left), unbalanced due to different amplitudes (middle) and unbalanced due to non-zero phase distortions (right) three-phase voltage system.

d) Balanced System: We aim to find an expression for the frequency of the balanced complex $\alpha - \beta$ voltage in (3.2), considering the strictly linear autoregressive model of order 1 in (3.4).

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

We begin by substituting the expression for $v(n)$ in (3.2) into the strictly linear model:

$$\begin{aligned} v(n+1) &= h^*(n)v(n) \\ \sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_o}{f_s}(n+1)+\phi)} &= h^*(n)\sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_o}{f_s} n+\phi)} \\ \implies h^*(n) &= e^{-j(2\pi \frac{f_o}{f_s} n+\phi)+j(2\pi \frac{f_o}{f_s}(n+1)+\phi)} = e^{j(2\pi \frac{f_o}{f_s})} \\ \implies h(n) &= e^{-j(2\pi \frac{f_o}{f_s})} \end{aligned}$$

In the last expression, $h(n)$ can be seen in its polar form $h(n) = |h(n|e^{j \arctan(\frac{\mathrm{Im}\{h(n)\}}{\mathrm{Re}\{h(n)\}})}$. Therefore, we find the intended expression for f_o:

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

$$f_o(n) = \frac{f_s}{2\pi} \arctan\left(\frac{\mathrm{Im}\{h(n)\}}{\mathrm{Re}\{h(n)\}}\right) \tag{3.5}$$

We neglected the negative sign (negative frequency).
Unbalanced System: We aim to find an analogous expression for the unbalanced complex $\alpha - \beta$ voltage in (3.3), considering the widely linear autoregressive model of order 1 in (3.6).

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

We begin by substituting the expression for $v(n)$ in (3.3) into the widely linear model:

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

Comparing the coefficients of $e^{j(2\pi\frac{f_o}{f_s}n+\phi)}$ and $e^{-j(2\pi\frac{f_o}{f_s}n+\phi)}$, we find:

$$\begin{aligned}
A(n+1)e^{j(2\pi\frac{f_o}{f_s})} &= h^*(n)A(n) + g^*(n)B^*(n) \\
B(n+1)e^{-j(2\pi\frac{f_o}{f_s})} &= h^*(n)B(n) + g^*(n)A^*(n)
\end{aligned} \tag{3.7}$$

Knowing that $e^{j(2\pi\frac{f_o}{f_s})}$ and $e^{-j(2\pi\frac{f_o}{f_s})}$ are complex conjugates and assuming that the change of magnitude between subsequent samples is negligible ($A(n) \approx A(n+1), B(n) \approx B(n+1)$):

$$\begin{aligned}
&h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} = h(n) + g(n)\frac{A(n)}{B^*(n)} \\
&h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} - h(n) - g(n)\frac{A(n)}{B^*(n)} = 0 \\
&\text{Multiply by } \frac{B^*(n)}{A(n)} \\
&g^*(n)\Big(\frac{B^*(n)}{A(n)}\Big)^2 + (h^*(n) - h(n))\frac{B^*(n)}{A(n)} - g(n) = 0 \\
&g^*(n)\Big(\frac{B^*(n)}{A(n)}\Big)^2 - 2j\,\mathrm{Im}\{h(n)\}\frac{B^*(n)}{A(n)} - g(n) = 0
\end{aligned}$$

This is in the form of a quadratic equation, hence we solve for $\frac{B^*(n)}{A(n)}$:

$$\begin{aligned}
\frac{B^*(n)}{A(n)} &= \frac{2j\,\mathrm{Im}\{h(n)\} \pm \sqrt{(-2j\,\mathrm{Im}\{h(n)\})^2 + 4g^*(n)g(n)}}{2g^*(n)} \\
&= j\frac{\mathrm{Im}\{h(n)\} \pm \sqrt{\mathrm{Im}^2\{h(n)\} - |g(n)|^2}}{g^*(n)}
\end{aligned}$$

We substitute back into (3.7):

$$\begin{aligned}
e^{j(2\pi\frac{f_o}{f_s})} &= h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} = h^*(n) + g^*(n)j\frac{\mathrm{Im}\{h(n)\} \pm \sqrt{\mathrm{Im}^2\{h(n)\} - |g(n)|^2}}{g^*(n)} \\
&= \mathrm{Re}\{h(n)\} - j\,\mathrm{Im}\{h(n)\} + j\,\mathrm{Im}\{h(n)\} \pm j\sqrt{\mathrm{Im}^2\{h(n)\} - |g(n)|^2}
\end{aligned}$$

Using the equation for the polar angle once again, we find the intended expression for f_o:

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

$$f_o(n) = \frac{f_s}{2\pi} \arctan\left(\frac{\sqrt{\mathrm{Im}^2\{h(n)\} - |g(n)|^2}}{\mathrm{Re}\{h(n)\}} \right) \tag{3.8}$$

e) Finally, Figure 30 compares the performances of the CLMS and ACLMS algorithms in finding the nominal frequency (f_o = 50Hz) of both balanced and unbalanced three-phase power system, using respectively (3.5) and (3.8).

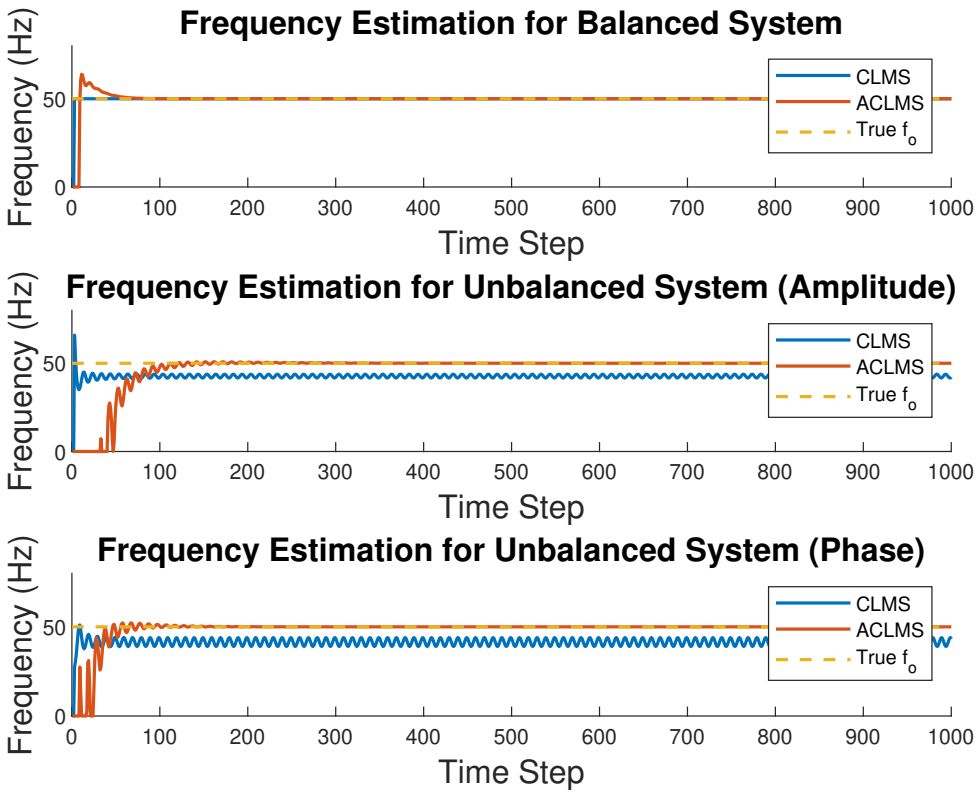

Figure 30: Frequency estimation with the CLMS and ACLMS algorithms for balanced (top), and unbalanced (middle and bottom) systems.

As expected, for the balanced system both algorithms converge to the true value of f_o, although the ACLMS has an initial transient phase in which it overshoots before converging. The CLMS performs better, possibly because it starts with the assumption that the data is circular. For unbalanced system voltages, the ACLMS converges to f_o but the CLMS does not. Indeed, the CLMS is not equipped with the sufficient degrees of freedom to fully exploit the second order statistics in the data. In addition, the ACLMS does not present oscillation errors in the steady state.

3.2 Adaptive AR Model Based Time-Frequency Estimation

a) We generated the frequency modulated (FM) signal $y(n) = e^{j(\frac{2\pi}{f_s}\Phi(n)} + \eta(n)$, where $\eta(n)$ is circular complex-valued white noise with zero mean and variance $\sigma_\eta^2 = 0.05$ and the phase $\Phi(n) = \int f(n)dn$ is generated from (3.9).

$$f(n) = \frac{d\Phi(n)}{dn} = \begin{cases} 100, & 1 \leq 0 \leq 500 \\ 100 + \frac{n-500}{2} & 501 \leq n \leq 1000 \\ 100 + \left(\frac{n-1000}{25}\right)^2 & 1001 \leq n \leq 1500 \end{cases} \tag{3.9}$$

The `aryule` function was used to find the AR(1) coefficients for the complete of the signal, as well as for each individual segment as dictated by $f(n)$ in (3.9). Figure 31 displays the power

spectrum of the signal and of its individual segments, showing that this method does not capture the changes in frequency given by (3.9). Indeed, AR models assume the signal to be stationary. Therefore, while models of higher order would perform marginally better in identifying multiple frequency components, they would not reflect the true dynamics, governed by a single time-varying coefficient.

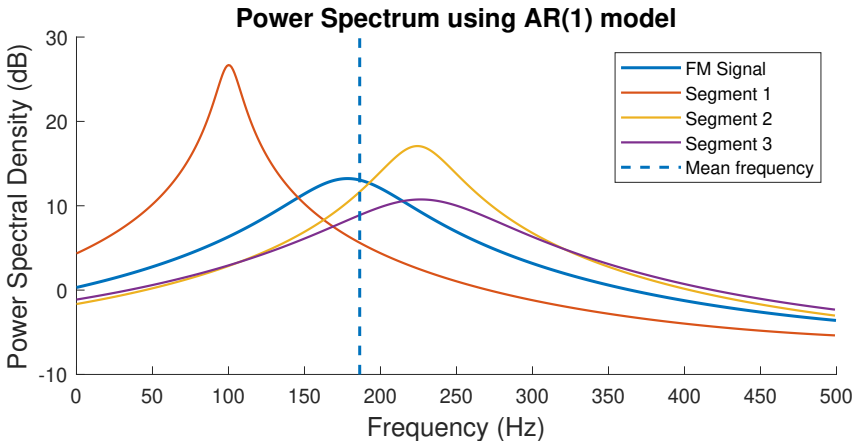

Figure 31: Power spectrum of the FM signal and of its three individual segments using an AR(1) model to estimate the coefficient.

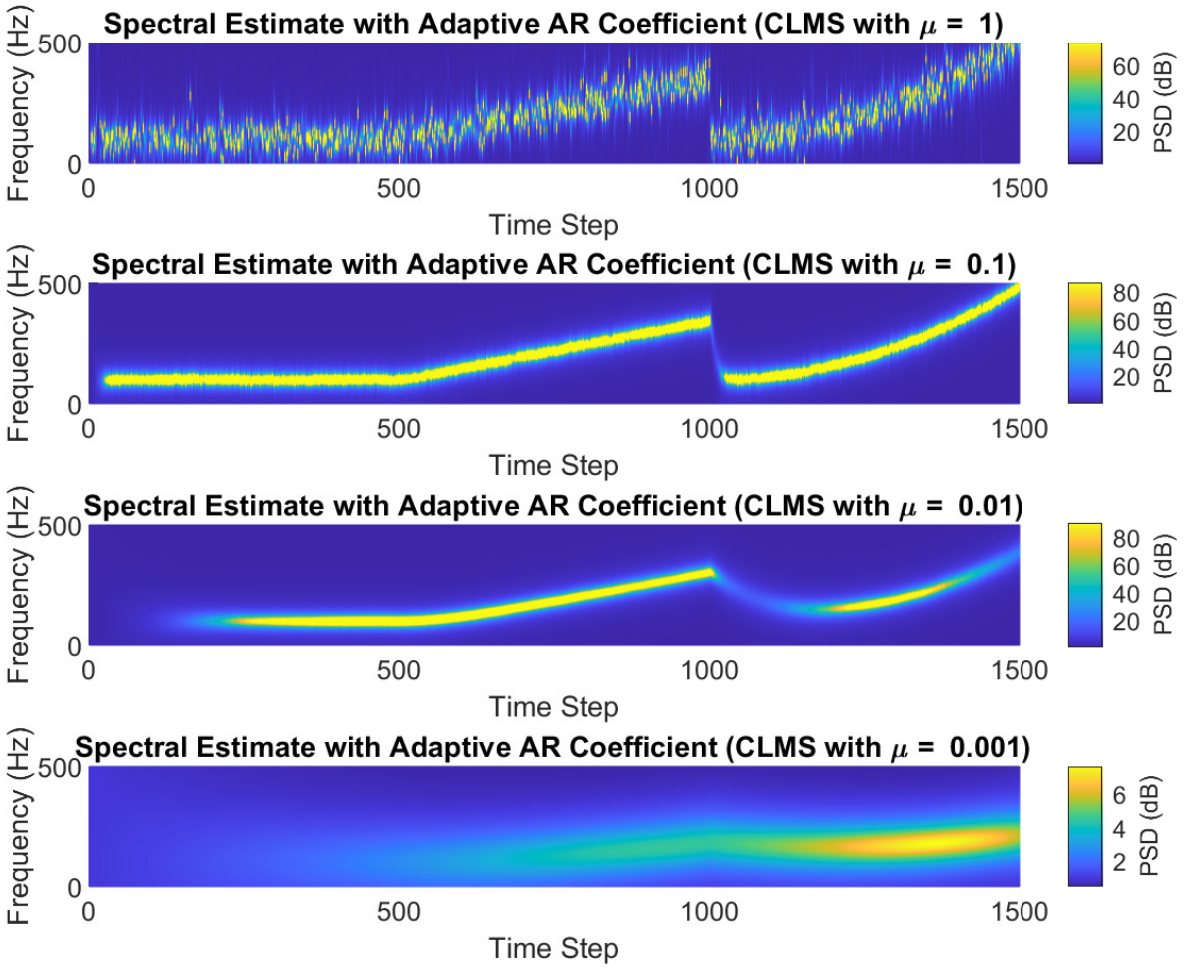

Figure 32: Power spectrum of the FM signal and of its three individual segments using an AR(1) model to estimate the coefficient.

b) We implemented the CLMS algorithm to estimate the AR coefficient of $y(n)$ and computed the frequency spectrum at each time instant. As shown in Figure 32, the CLMS successfully captures the changes in frequency over time through a time varying coefficient, provided the step size parameter is chosen appropriately. The step size must be small enough as to avoid non stable estimation and oscillations around the optimum, and it must be large enough to ensure learning within the correct interval. In this case $\mu = 0.1$ was a reasonable choice.

3.3 A Real Time Spectrum Analyser Using Least Mean Square

a) A signal $y(n)$ can be estimated using a linear combination of N harmonically related sinusoids as in (3.10)

$$\hat{y}(n) = \sum_{k=0}^{N-1} w(k) e^{j2\pi kn/N} \tag{3.10}$$

which can also be written in vectorial form $\hat{\mathbf{y}} = \mathbf{F}\mathbf{w}$.
Within the least squares framework, the vector $\mathbf{w}$ is found by minimising the sum of the squared errors between the estimated signal $\hat{y}(n)$ and the true signal $y(n)$ (3.11).

$$\min_{\mathbf{w}} \|\mathbf{y} - \hat{\mathbf{y}}\|^2 = \min_{\mathbf{w}} \sum_{n=0}^{N-1} |y(n) - \hat{y}(n)|^2 \tag{3.11}$$

We expand and find the cost function:

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

The second and third term of the equations are scalars and equal, hence can be combined. To find the optimal weights in the least squares error sense (3.12), we set the derivative w.r.t $\mathbf{w}$ to zero.

$$\begin{aligned} \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} &= -2\mathbf{F}^H\mathbf{y} + 2\mathbf{F}^H\mathbf{F}\mathbf{w} = 0 \\ &\implies \mathbf{w} = (\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H\mathbf{y} \end{aligned} \tag{3.12}$$

Equation (3.10) is analogous to the Inverse Discrete Fourier Transform (IDFT), hence $\mathbf{w}$ is the vector of Fourier coefficients.

b) We know that $\hat{\mathbf{y}} = \mathbf{F}\mathbf{w}$ meaning that $\hat{\mathbf{y}} \in col(\mathbf{F})$, where $col(\mathbf{F})$ is the column space of $\mathbf{F}$ (basis vectors). Least squares attempts to find the vector that minimises the Euclidean distance from $\hat{\mathbf{y}}$ to $\mathbf{y}$ with the constraint that $\hat{\mathbf{y}}$ lies in the $col(\mathbf{F})$. This is the vector in the column space of F that is closest to $\mathbf{y}$ (in Euclidean norm), which is the orthogonal projection of $\mathbf{y}$ onto the column space of F. The projection mapping is $\mathbf{P} = \mathbf{F}(\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H$, found by substituting $\hat{\mathbf{y}} = \mathbf{F}\mathbf{w}$ into (3.12). This explains the DFT as a projection and change of basis.

c) We implemented the DFT-CLMS for the FM signal in Section 3.2 by treating the DFT as a least squares solution, where the desired signal to the CLMS can be treated as the time-domain signal to be transformed and the input to the filter at time n is the complex phasor $x(n) = \frac{1}{N}\left[1, e^{j\frac{2n\pi}{N}}, e^{j\frac{4n\pi}{N}}, \ldots, e^{j\frac{2n(N-1)\pi}{N}}\right]^T$. The result is shown in Figure 33 (top).

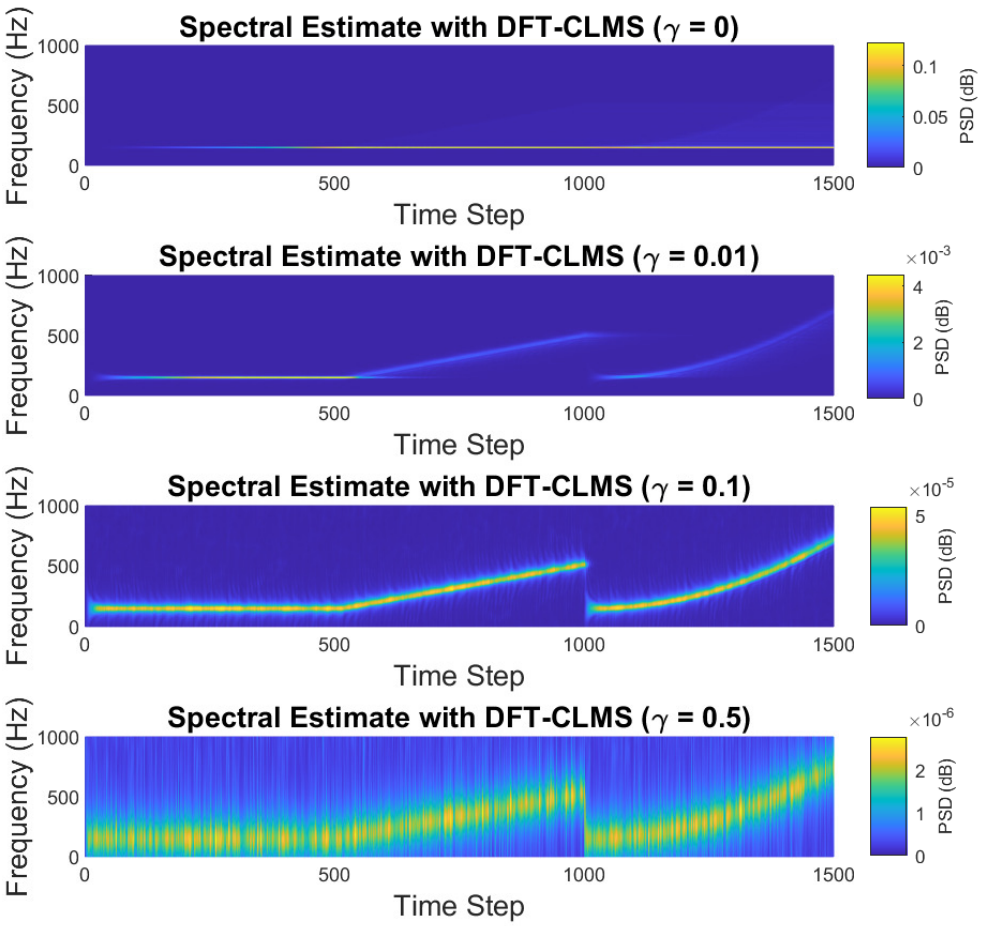

Figure 33: Time-frequency spectrum of the FM signal using the DFT-CLMS with varying leakage coefficients.

We note that the standard CLMS performs successfully for the constant segment, however, in the linear and exponential segments the new found coefficients are superimposed instead of being updated (faint on the image). This is a result of the Fourier weights being updated too slowly as well as the memory of the algorithm. To deal with this, we introduce a leakage parameter γ, which, as seen in Section 2.1, weights the previous coefficients so that they have less importance in the update (3.13).

$$w(n+1) = (1-\gamma\mu)\, w(n) + \mu e^*(n)x(n) \tag{3.13}$$

The results are shown in Figure 33. When γ is small, $\gamma = 0.01$, memory of older weights still influences subsequent segments. This improves as γ increases. When γ is too large however, the output becomes noisy, as weights further into the past are less important and as the update equation diverges more from the equation for the optimal weights. A reasonable value for the leak parameter was found to be $\gamma = 0.1$.

d) We implemented the same DFT-CLMS algorithm as above on the POz channel of the EEG signal used in Sections 1.2 and 2.3. To reduce computational complexity, this was done on a segment of the signal, POz$(4000:5200)$. In this case we do not introduce a leakage parameter as the signal is stationary. The result is shown in Figure 34. The algorithm is successful in learning all the weights: the mains frequency (50Hz), the SSVEP (13Hz) and its harmonics. However this takes around 400 to 600 time samples, which could be too slow for some applications.

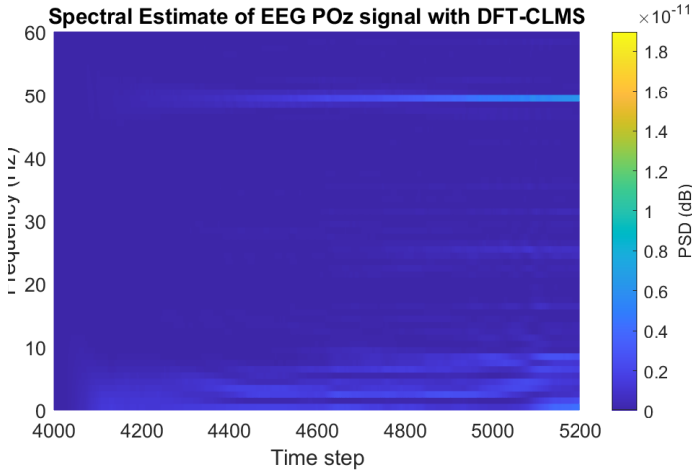

Figure 34: Time-frequency spectrum of the EEG signal using the DFT-CLMS.

4 From LMS to Deep Learning

The table below shows performance metrics for the five models presented in this section. These are the mean-square error (MSE), and the prediction gain, $R_p = 10\log_{10}(\frac{\hat{\sigma}_y^2}{\sigma_e^2})$ - where $\hat{\sigma}_y^2$ is the variance of the output of the LMS and σ_e^2 is the variance of the prediction error.

Model	4.1	4.2	4.3	4.4	4.5
MSE (dB)	16.03	22.92	6.80	11.31	3.19
Prediction Gain (dB)	5.20	-23.17	16.47	12.28	20.35

Table 5: Mean-square error and prediction gain of the five models presented in Section 4.

4.1 Time Series Prediction with Standard LMS

Assuming the data from a time series is generated from an AR(4) process, we plot the zero-mean time series $y[n]$ and its one step ahead prediction obtained with the LMS algorithm in Figure 35. We note that the algorithm converges after circa 200 samples. While the performance was satisfactory for a linear model, the algorithm is unable to capture non-linearities in the signal, as seen in Figure 35 (left). The resulting MSE is relatively high (Table 5).

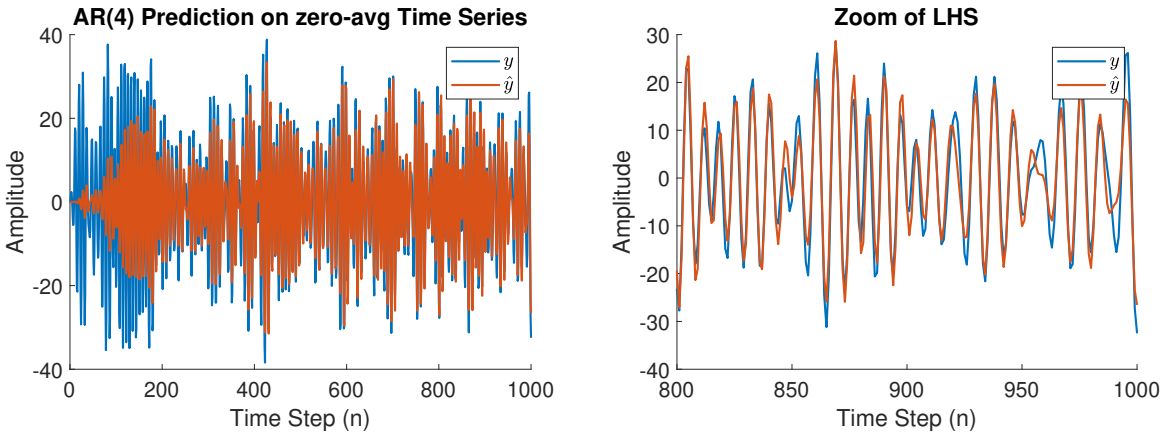

Figure 35: Comparison between a zero-mean non-stationary time-series and its one-step ahead prediction generated using the LMS algorithm assuming an AR(4) model.

4.2 Dynamical Perceptron

In order for the model to become more expressive, we add a non-linearity to the output of the LMS, the activation function Φ. The error and and the weight updates are then computed as in (4.1) and (4.2).

$$e(n) = d(n) - \Phi\left(\mathbf{x}^T(n)\mathbf{w}(n)\right) \tag{4.1}$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\Phi'(n)e(n)\mathbf{x}(n) \tag{4.2}$$

Figure 36 shows the zero-mean signal against the output of the LMS where the activation function is the `tanh` function. We observe that, while it might be able to capture the non-linearities, the model is unable to predict the signal to its magnitude, as $\tanh(x) \in [-1, 1]$. The MSE and prediction gain are shown in Table 5. As expected, the MSE is higher than with other methods shown in 4.1-4.5 as the prediction does not approximate the signal.

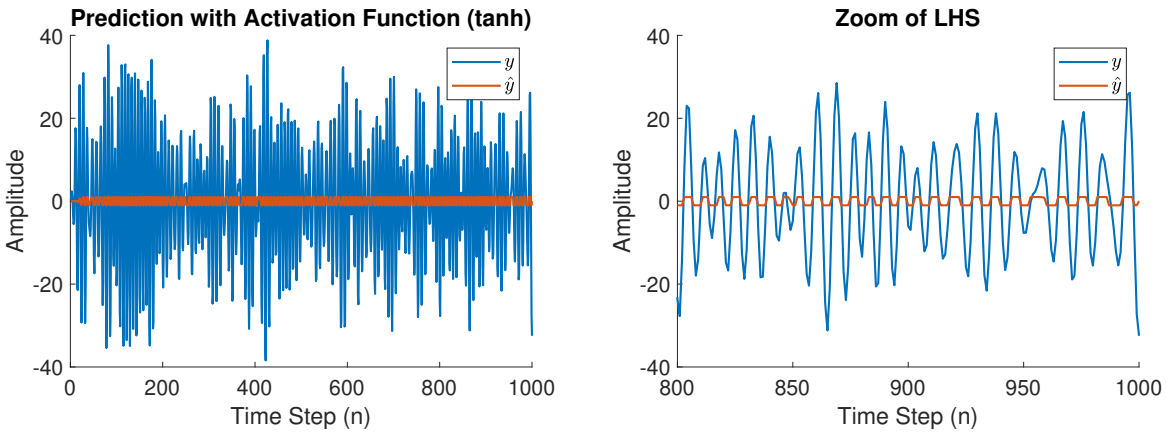

Figure 36: Comparison between a zero-mean non-stationary time-series and its one-step ahead prediction generated using the dynamical perceptron with `tanh` as the activation function.

4.3 Dynamical Perceptron with Scaled Activation Function

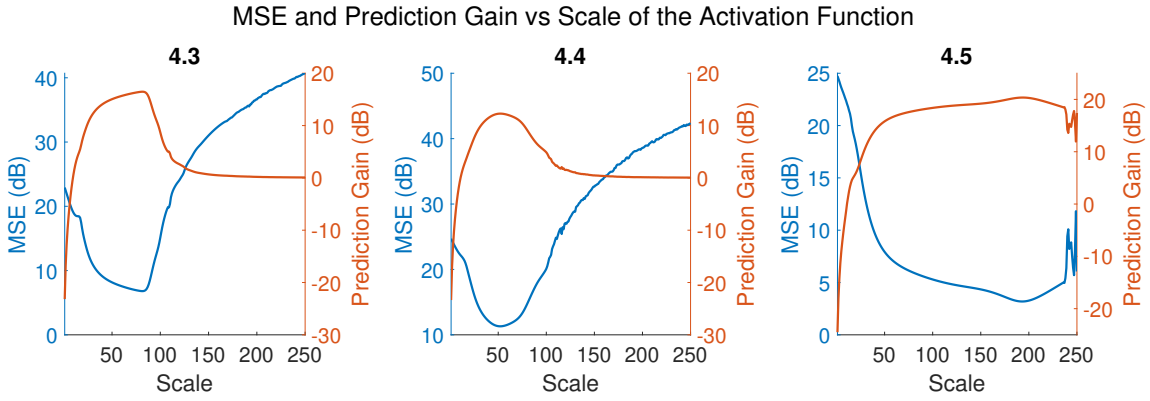

Figure 37: Plots of the MSE and prediction gain as a function of the scale of the activation function for models 4.3, 4.4 and 4.5.

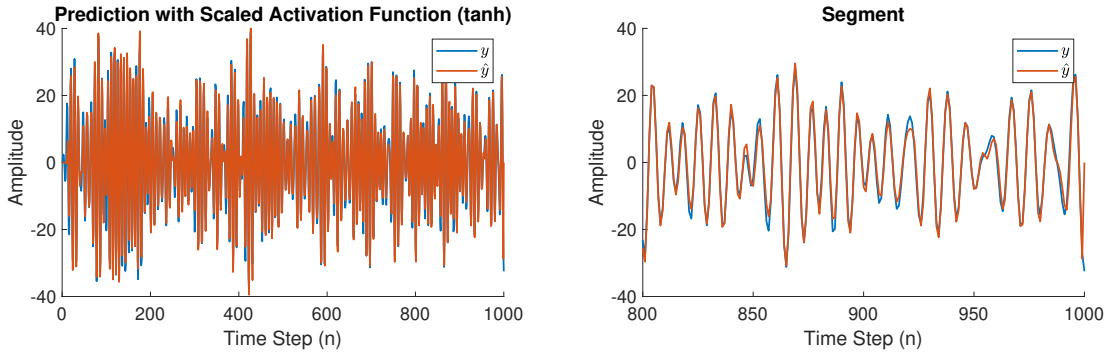

Figure 38: Comparison between a zero-mean non-stationary time-series and its one-step ahead prediction generated using the dynamical perceptron with `tanh` scaled by a factor $a = 82$ as the activation function.

To tackle the problem encountered in 4.2, we generalise the activation by scaling the activation function. The scaling coefficient a should be large enough to predict the signal to its magnitude but small enough to prevent instabilities. We ran the model with scaling $a \in [1 : 250]$ and find the best scaling value to be $a = 82$, as seen in Figure 37 (left). This same process was repeated for models 4.4 and 4.5. The MSE and prediction gain shown in Table 5 demonstrate that this model

outperforms 4.1 and 4.2, showing the benefits of accounting for the non-linear nature of the signal and of appropriately selecting the scaling of the activation function. This can also be observed in Figure 38.

4.4 Dynamical Perceptron with Bias

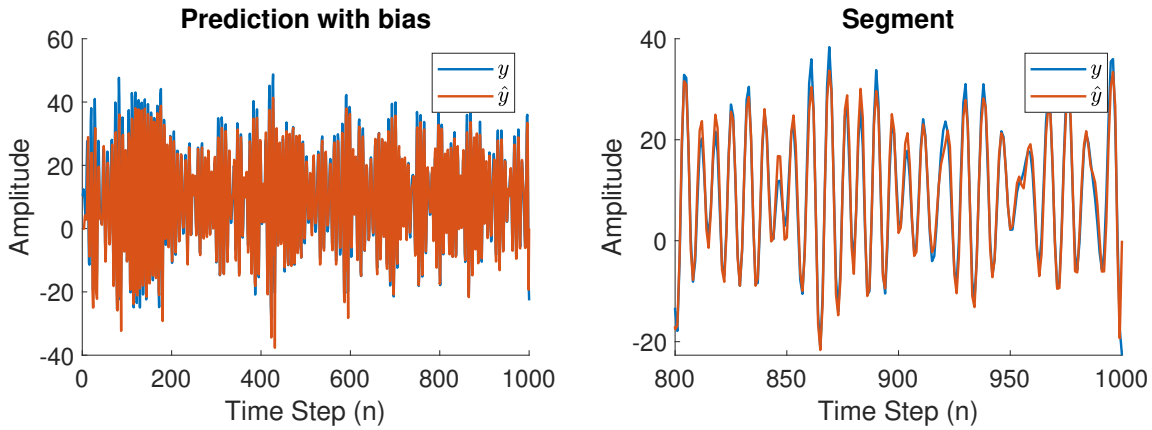

Figure 39: Comparison between a non-zero mean non-stationary time-series and its one step ahead prediction generated using the scaled dynamical perceptron and taking an augmented input to account for the bias.

We now consider the original time series $y[n]$, which exhibits a non-zero mean. To account for the mean automatically, we can add a bias to our model described by $\tanh(\mathbf{w}^T\mathbf{x} + b)$. This can be implemented by considering an 'augmented' input to the algorithm $[1, \mathbf{x}]^T$, where the weight vector iteratively multiplied by 1 represents the bias. The bias is adaptively learned, as observed in Figure 39. Therefore, this model exhibits a higher error than model 4.3, as shown in Table 5. In this case the optimal scaling was found to be $a = 53$.

4.5 Pre-training Weights

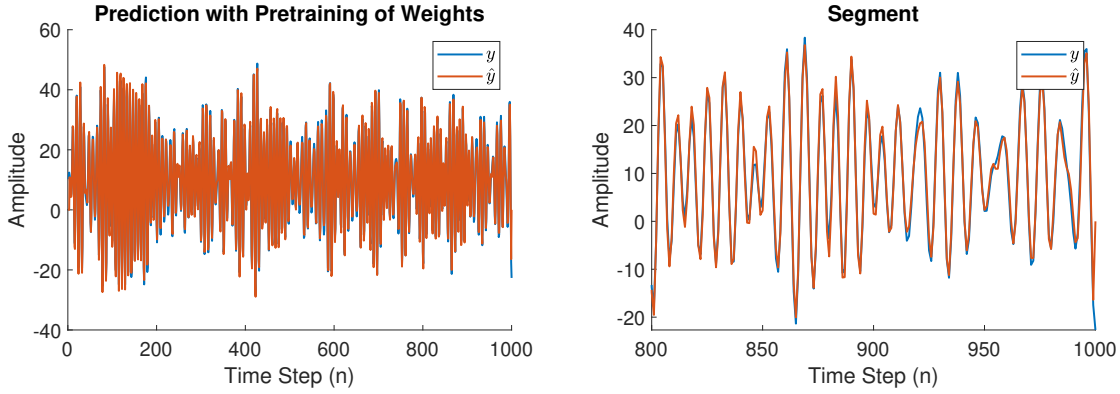

Figure 40: Comparison between a non-zero mean non-stationary time-series and its one step ahead prediction generated using the scaled dynamical perceptron accounting for bias with pre-trained weights.

Since a single weight update is performed per time-step, convergence to a reasonable prediction may take a lot of samples. To tackle this issue, one can pre-train the weights by over-fitting to a small number of samples, which allows to find a better initial condition for the weights. Pre-training was implemented starting with $\mathbf{w}(0) = 0$ to fit the model to the first 20 samples for 100 epochs or iterations. The prediction performance is shown in Figure 40. This model outperformed

all the others discussed, with the lowest MSE and highest prediction gain as seen in Table 5, exhibiting accurate predictions starting from the initial samples.

4.6 Backpropagation

Backpropagation, short for "backward propagation of errors" is an algorithm used for training neural networks in the context of supervised learning using gradient descent in the weight space with respect to some loss function. The algorithm is such that each neuron in a given layer computes weights based on information received from the previous layer (feed forward), until the output neuron, where the output is compared with the teaching signal, producing the prediction error. The weights in its receptive field are then updated in the LMS manner hence considering the error gradient on the weights $\frac{\partial E}{\partial w_{i,j}}$. This in turn affects the neurons in the receptive fields of the newly updated neurons, thus effectively iteratively propagating information about the error back through the layers of the deep network from the output to the first layer.

4.7 Deep Neural Network

In this section we will compare the performance of a deep network against a single neuron (linear activation function and tanh activation function) in predicting a signal corrupted by noise $y[n]$, by looking at the learning curves of the algorithms over 20,000 epochs. Figure 41 shows the comparison of the different configurations in the prediction task. We observe that the single neurons fail to capture the signal's behaviour while the deep network seems to at least capture its patterns, however exhibiting flat regions that do not follow the original signal. By comparing the loss functions in Figure 42, we see that both single neuron configurations reach convergence faster than the deep network. However, the deep network leads to better steady-state results with a lower test loss. We note that in all cases the best prediction is lower that the steady-state result and reached at an early stage. This is evidence of overfitting which might indicate the need to reduce the number of epochs or change the number and size of layers in the case of the deep network.

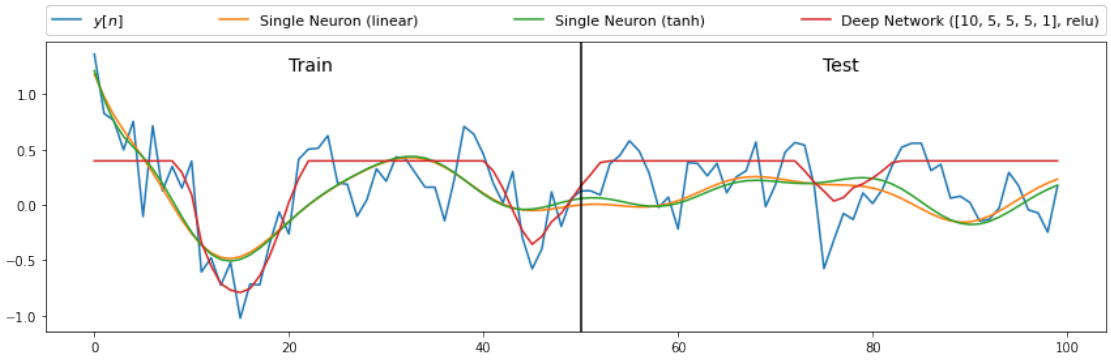

Figure 41: Prediction of a noisy signal for different methods after 20,000 epochs, with noise power 0.05.

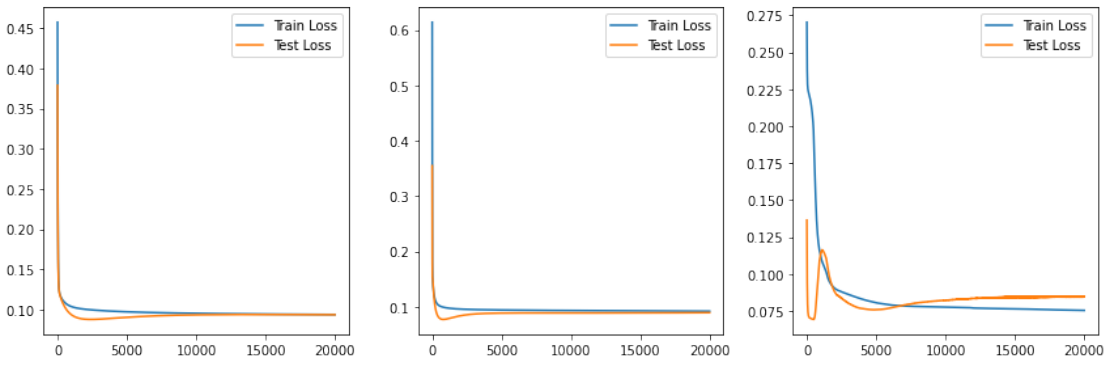

Figure 42: Training and testing learning curves for the prediction of a noisy signal for the standard LMS, non-linear dynamical perceptron (middle) and deep network (right), with noise power 0.05.

4.8 DNN and Noise Power

First, we set the noise power to be 0. Figure 43 shows that in this case the deep network is able to do a near-ideal prediction of the noise corrupted signal, which also translates into low a steady state test loss as seen in Figure 44 (right). Once again the learning curve oscillates before reaching a steady state in the case of the deep network. We observe that the performance of the single neurons (both configurations) is not significantly different than in section 4.7 and that they are not able to predict the signal accurately.

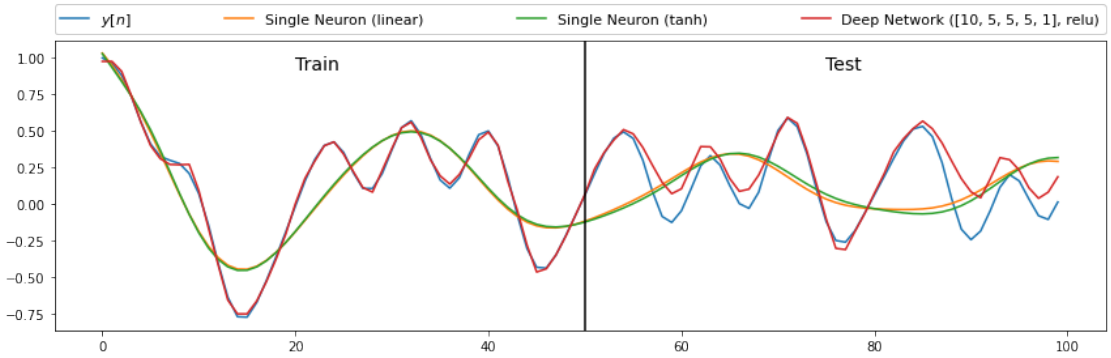

Figure 43: Prediction of a noisy signal for different methods after 20,000 epochs, with noise power 0.0.

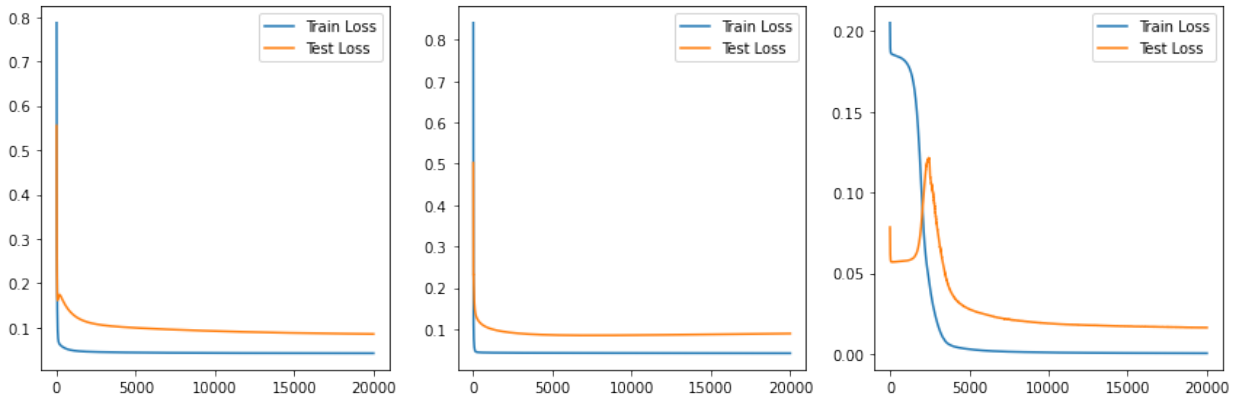

Figure 44: Training and testing learning curves for the prediction of a noisy signal for the standard LMS, non-linear dynamical perceptron (middle) and deep network (right), with noise power 0.0.

Then, we increased the noise power to be 0.1. In this case, the performance of the three configurations is worse than when the noise power was 0 or 0.05. In particular, the behaviour of the deep

network is erratic with the increase of epochs, not reaching a steady state. This shows that the deep network is more susceptible to noise and does not perform reliably for noisy input signals.

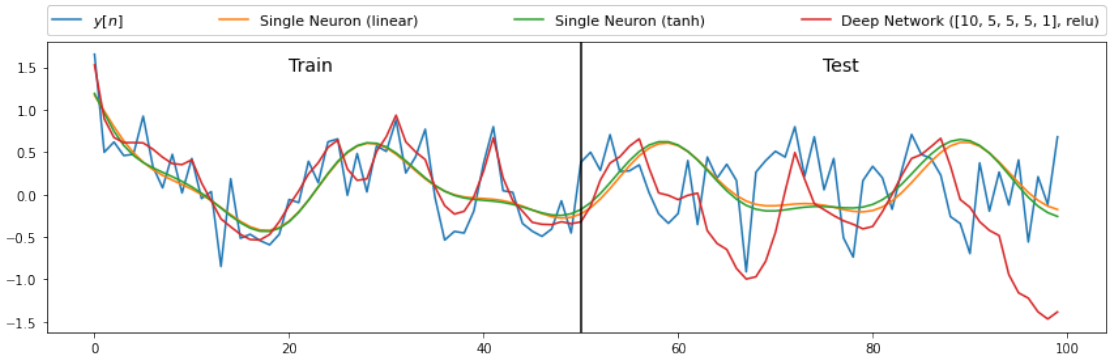

Figure 45: Prediction of a noisy signal for different methods after 20,000 epochs, with noise power 0.1.

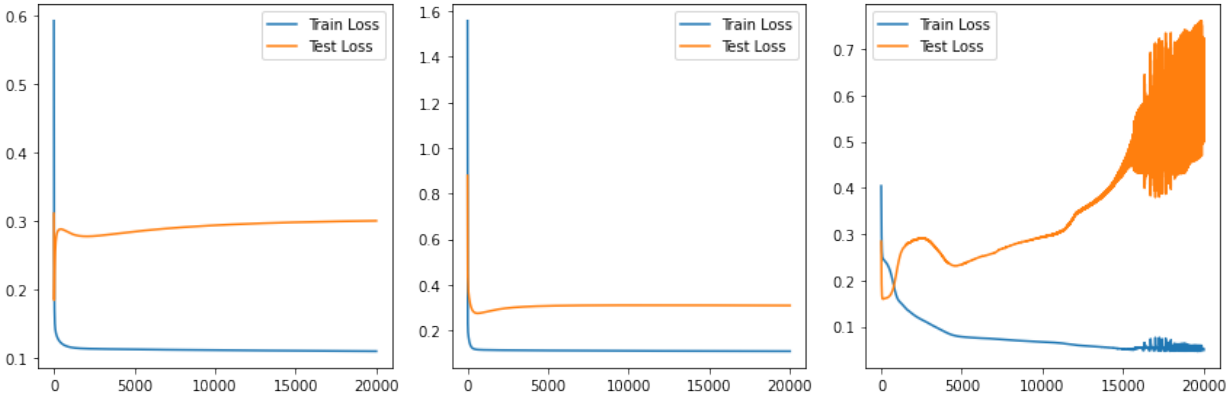

Figure 46: Training and testing learning curves for the prediction of a noisy signal for the standard LMS, non-linear dynamical perceptron (middle) and deep network (right), with noise power 0.1.

5 Tensor Decomposition for Big Data Applications

The code for this section was submitted on Blackboard, alongside the code for each section of this coursework.