Imperial College London

Department of Electrical and Electronic Engineering

Adaptive Signal Processing and Machine Intelligence Coursework

Author:
Nima Karsheneas

CID:
01500753

Lecturer:
Prof. Danilo MANDIC

April 2023

Contents

1 Classical and Modern Spectrum Estimation

1.1 Properties of Power Spectral Density (PSD)

As given, there are two definitions for the PSD of a discrete-time deterministic sequence of finite energy, $\{x(n)\}$ ($\sum_{n=-\infty}^{\infty} |x(n)|^2 < \infty$) - shown in equations 1 and 2.

$$P(\omega) = \sum_{k=-\infty}^{\infty} r(k)e^{-j\omega k} \tag{1}$$

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

The aim is to to prove their equivalence, under the mild assumption that the covariance sequence, $r(k) = \mathbb{E}\{x(k)x^*(k-m)\}$ is rapidly decaying, that is equation 3 holds true.

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=1-N}^{N-1} |k||r(k)| = 0 \tag{3}$$

Using the property of the multiplication of complex conjugates, and the linearity of the expectation operator, equation 2 can be rewritten as:

$$\begin{aligned}
\lim_{N\to\infty} P(\omega) &= \lim_{N\to\infty} E\left(\frac{1}{N}\sum_{n=0}^{N-1} x(m)e^{-jm\omega}\sum_{n=0}^{N-1} x^*(n)e^{-jn\omega}\right) \\
&= \lim_{N\to\infty} \frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} E\left(x(m)x^*(n)\right)e^{-j\omega(m-n)} \\
&= \lim_{N\to\infty} \frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} r(m-n)e^{-j\omega(m-n)}
\end{aligned} \tag{4}$$

Now introducing the dummy variable τ, where $\tau = k - m$, the double summation can be reduced into a single summation. τ takes values between $-(N-1)$ and $(N-1)$, therefore the single summation must now be taken between these two values. Furthermore, the number of combinations of $k - m$ for each unique value of τ must be considered. By observing the combinations of $k - m$, this value is determined to be $N - \tau$. Given this, equation 4 can be rewritten as:

$$\begin{aligned}
\lim_{N\to\infty} P(\omega) &= \lim_{N\to\infty} \frac{1}{N}\sum_{k=1-N}^{N-1} (N-|k|)r(k)e^{-j\omega k} \\
&= \lim_{N\to\infty}\left(\sum_{k=1-N}^{N-1} r(k)e^{-j\omega k} - \frac{1}{N}\sum_{k=1-N}^{N-1} |k|r(k)e^{-j\omega k}\right) \\
&= \sum_{k=-\infty}^{\infty} r(k)e^{-j\omega k} - \lim_{N\to\infty}\frac{1}{N}\sum_{k=1-N}^{N-1} |k|r(k)e^{-j\omega k}
\end{aligned} \tag{5}$$

Substituting equation 3 into the equation above, it follows that for a rapidly decaying covariance sequence:

$$P(\omega) = \sum_{k=-\infty}^{\infty} r(k)e^{-j\omega k} \tag{6}$$

as required.

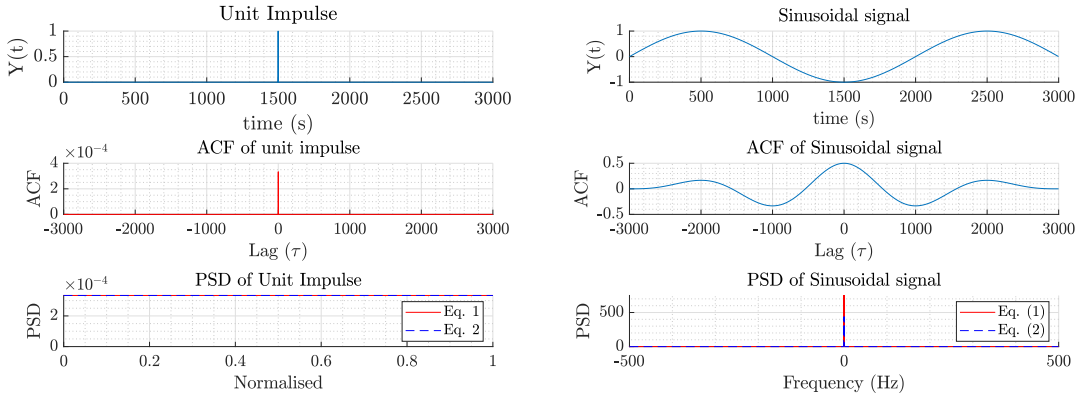


Figure 1: Comparison of an impulse signal and a sinusoidal wave, their ACFs and PSD estimates.

The unit impulse, with a rapidly decaying ACF, has no mismatch between PSD estimates when using equation 1 and 2. Whereas, the sinusoidal signal, with a slowly decaying ACF shows a mismatch between the PSD estimates of the two equations, proving the need for equation 3 to hold true in order for the results of equation 1 and 2to be equal.

1.2 Periodogram-based Methods Applied to Real–World Data

1.2.1 Part a) Sunspot Time Series

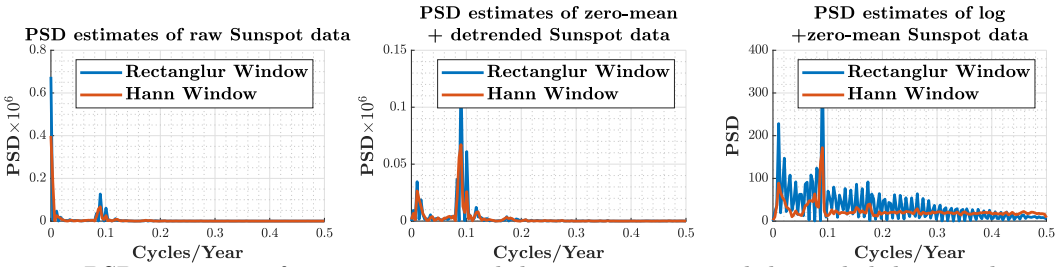


Figure 2: PSD estimates of raw pre-processed data, zero-mean and detrended data, and centered logarithmic data, respectively.

The spectral estimations, using Hann and rectangular windowing of the Sunspot series for various transformations to the data are shown in Figure 2. As can be seen in the first graph, the periodogram is dominated by the DC component at $\omega = 0$. To rectify this, the data is zero-centred by subtracting the mean, and de-trended by subtracting the linear line of best fit to ensure the first-order moment is stationary. The non-DC spectral peaks are now more easily identifiable, namingly the maximal sunspot activity is observed to be at around 0.09 cycles/year, corresponding to a time-period of 11.1 years. Despite this improvement, there is some activity in the lower PSD values that are hardly visible, to magnify the smaller values, the raw data is transformed onto the logarithmic scale and zero-centred(same reason as before), and indeed now some spectral peaks are observed at 1st and 2nd harmonics of the dominant peak 0.09 cycles/year.

1.2.2 Part b) The basis for Brain Computer Interface (BCI)

The PSD of the electroencephalogram (EEG) is estimated in order to determine the steady state visual evoked potential (SSVEP) resulting from a flashing visual stimulus, of fixed frequency, being observed by a subject. PSD estimates were first derived from the standard periodogram (with Hann windowing) used in the previous sections, and then from Bartlett's method of averaged periodograms of different window lengths (Δ_t = 10s, 5s, 1s respectively), the results of which are shown in Figure 3. To be able to fairly compare the different methods, frequency resolutions were fixed at 5 DFT samples/Hz.

From the results it can be seen that there are four (somewhat) distinct narrow peaks in the frequency spectrum of all four methods, namely at 13Hz, 26Hz, 39Hz and 50Hz. The peak at 50Hz corresponds

to the power-line interference induced in the sensors; the remainder of the peaks correspond to the the fundamental frequency of the SSVEP and it's corresponding harmonics. The third harmonic at 52Hz is not distinguishable because of it's proximity to the high-power peak as a result of mains interference. There is an additional wider peak observed between 8-10Hz and corresponds to subject fatigue, and not the SSVEP. It can also be seen that as the window is getting smaller, the variance of the PSD estimate is also getting smaller, making the spectral peaks, in general, more easily identifiable. The reduction of variance originates from the fact that as window length gets smaller, the number of segments in which the PSD is averaged over is increasing, thereby reducing the variance. Since each segment used to estimate the PSD is now composed of less samples, there is a reduction in its true frequency resolution (before zero-padding), which for example, when using a window length of 1s, makes the peak at 13Hz no longer distinguishable, because of its proximity to the peak between 8-10Hz.

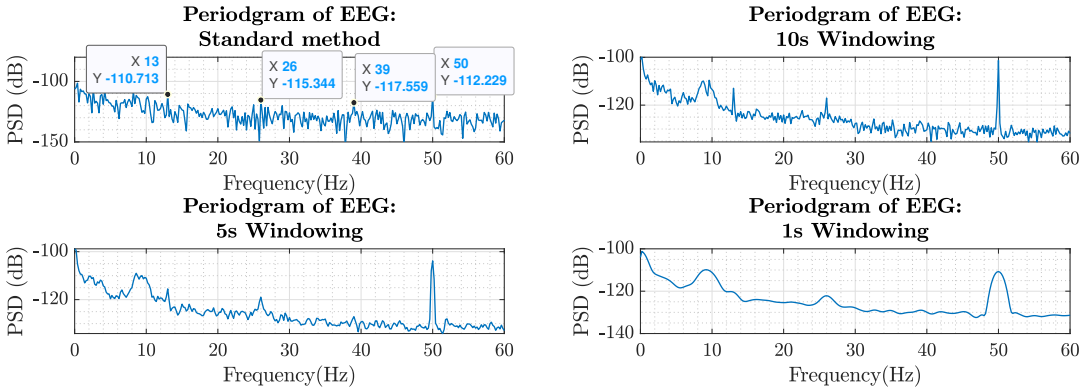


Figure 3: PSD estimates using standard periodogram and averaged periodograms of different window lengths.

Figure 4, shows a direct comparison between the Bartlett's windowing method and the standard periodogram. It can be seen that along with reducing the variance, windowing introduces some bias to the PSD estimate, alluding to the bias-variance trade-off.

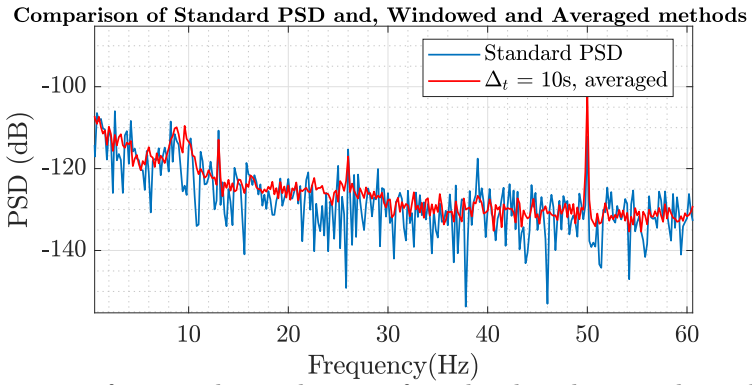


Figure 4: Comparison of averaged periodogram of window length 10s and standard periodogram

1.3 Correlation Estimation

1.3.1 Part a) - Unbiased correlation estimation and preservation of non-negative spectra

The results of the ACF estimation of 10,000 sample realisations of WGN, a noisy sinusoidal, and low-pass filtered WGN, using their respective biased and unbiased estimates (see equations 7 and 8), are shown in Figure 5. The correlogram/PSD associated with these biased and unbiased ACF estimates are also shown in Figure 5.

$$\textbf{Biased: } \hat{r}(k) = \frac{1}{N} \sum_{n=k+1}^{N} x(n)x^*(n-k) \tag{7}$$

$$\textbf{Unbiased: } \hat{r}(k) = \frac{1}{N-k} \sum_{n=k+1}^{N} x(n)x^*(n-k) \qquad 0 \leq k \leq N-1 \tag{8}$$

From Figure 5, it can be seen that, for $|k| < 5000$ the results of unbiased and biased estimates are approximately equal, after this point, the difference between the two estimates becomes more and more significant. As $|k|$ increases, the number of samples contributing to the summation $\sum_{n=k+1}^{N} x(n)x^*(n-k)$, decreases, consequently reducing it's value to where it comprises of only one value at a lag equal to $N-1$. For the unbiased estimate, the value in which this sum is scaled by $(\frac{1}{N-|k|})$ decreases with lag, reaching a value of 1 at $|k| = N-1$, meaning that it is an (unbiased) estimate of the ACF using only one sample, which by Central Limit Theorem (CLT) would have a estimation variance of N times the variance of the estimation at $k = 0$ where all N samples are used to calculate it's estimate. Indeed, this is why we see exploding values of the ACF as k approaches N for the unbiased estimate. As for the biased estimate, the number the summation is scaled by remains the same for all k, thus as k increases, and the number of samples contributing to the summation decreases, so does the resulting estimate for the ACF, which will decay to zero for $N \to \infty$, it must be noted that this particular unbiased estimate only matches the theoretical ACF because the theoretical ACF is also asymptotically decaying to 0, it would not perform well if this were not to be the case.

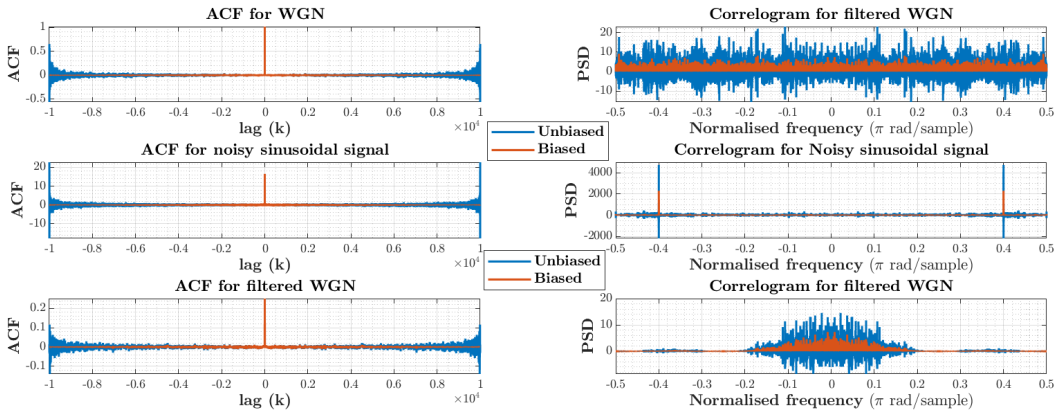


Figure 5: Plot of biased and unbiased estimates of the ACF and PSD of WGN, noisy sinusoidal ($f = 0.4\ rads/sample$) and low-pass filtered WGN.

As for the PSD, these are the expected theoretical responses for each of the signals: **WGN**:flat spectrum. **Noisy sinusoid**: impulses at the positive negative frequency of the sinusoid. **Filtered WGN**: a similar response to WGN at low frequencies and tending to zero after the cut-off frequency of the LPF. Both PSD estimates seem to largely capture the features that have just been described, however, the unbiased estimate exhibits some negative values, which is not the case for the true PSD.

1.3.2 Parts b & c) - Multiple realisations of the PSD estimate (absolute and dB)

Using the same unbiased ACF estimate as before, the PSD of 100 realisations of the process described in equation 9 is now estimated.

$$y(n) = 0.8sin\left(\frac{2\pi n \cdot 1}{f_s}\right) + sin\left(\frac{2\pi n \cdot 1.5}{f_s}\right) + 1.2sin\left(\frac{2\pi n \cdot 2}{f_s}\right) + \zeta(n) \qquad \zeta(\cdot) \sim WGN(0,1) \tag{9}$$

The PSD, the standard deviation of the PSD estimate, the PSD expressed in decibels, and it's corresponding standard deviation are shown in Figure 6. In the plot of the absolute value of the PSD, the three peaks attributed to the three frequencies of the sinusoidals of $y(n)$ are easily identifiable. Although on the plot it seems as if the PSD values converge to zero away from these peaks, this is actually a value of 1, which corresponds to the noise power, signifying the asymptotic unbiasedness of the estimate. The mean helps smooth calculations, making the nature of these peaks more easily

identifiable, this in fact according to CLT is reducing the variance of estimates 100 fold.

The PSD estimates of the same realisations of the process are now plotted onto the decibel scale ($10\log_1 0$ of PSD values). The logarithm function compresses values as they get larger, essentially amplifying the activity of low-power regions, from this, the variabilty of the estimates away from the peaks are now more discernible. The biased estimate of the ACF corresponds to an unbiased estimate of triangular Bartlett windowing of the original signal, in the frequency domain, this corresponds to a convolution with a sinc function, the sidelobes of which are now visible in the plot of the PSD in dB. Although not the case in the for this experiment, if the power of the peaks were closer to the noise power, plotting in decibels would become more easily identifiable, due to the compression property of the logarithm.

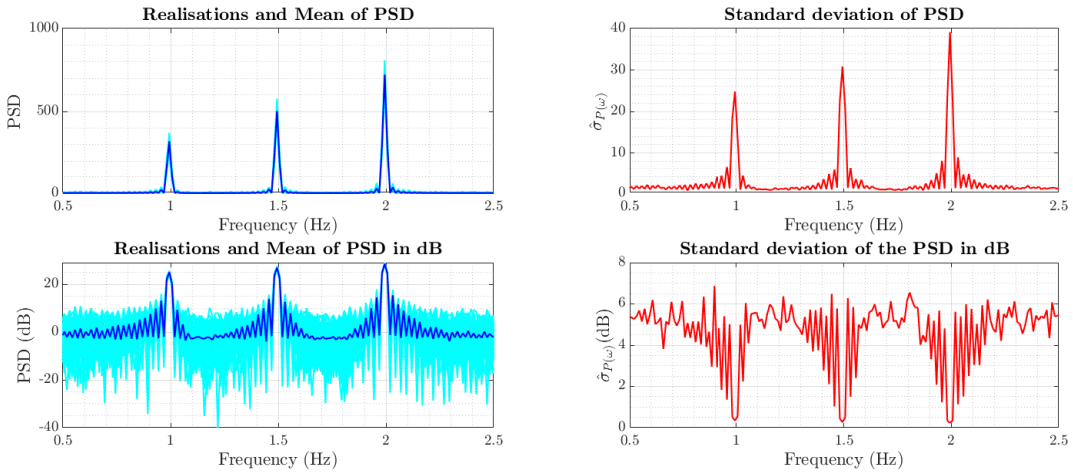


Figure 6: Graphs showing both the PSD estimates and the Standard Deviations of PSD estimates of 100 realisations of $y(n)$ and their corresponding empirical mean.

1.3.3 Part d) - Complex exponential signals

The PSD estimate of the complex valued signal (described in the coursework specification) composing of the summation of 2 exponential signals of frequency 0.3 and 0.32 Hz, is shown in Figure 7 for varying signal lengths. For a Barlett window, the resolution of the periodogram is approximately $\frac{0.89}{N}$. To be able to capture the two peaks the resolution would have to be at least equal to, but preferably finer than the difference between the two peaks. From this, the minimum number of samples required is equal to $\frac{0.89}{0.2} = 44.5$. This is indeed observed in Figure 7, for N=20, which is significantly less than the minimum required samples, only one peak is exhibited. As N=45 is approached, some splitting of the spectrum is observed, however these are still wide and not centred around the correct frequencies. As N increases past N=45, the main lobes at the correct frequencies become more and more visible, with an increasingly narrow main lobe bandwidth.

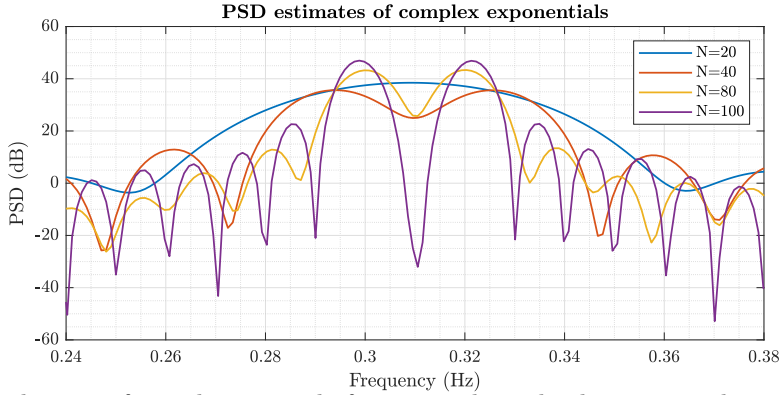


Figure 7: Periodogram of signal composed of two complex valued exponential signals of frequencies 0.3Hz and 0.32Hz.

1.3.4 Part e) - PSD estimation with MUSIC

The Multiple Signal Classification algorithm (MUSIC) is now considered and is used to estimate the PSD of the same expnential signal as in the previous part, for a process length of $N = 20$, a situation where the standard biased ACF method could not detect the two spectral peaks. Estimations are produced using the code below, the process of which is repeated 1000 times, the results of which, including their empirical mean and standard deviation is shown in Figure 8.

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

The `corrmtx` function is first used to obtain an estimation of the auto correlation matrix, `R`. The first argument of `corrmtx`, `x`, corresponds to the 20 samples of the complex exponential signal. The second argument specifies the model order, M. Finally, `'mod'` refers to the nature of the Toepelitz Matrix `X` (equation 10), where `R` is equal to the multiplication of the pseudo-inverse of `X` and `X`, such that: $\mathbf{R} = \mathbf{X}^{\dagger}\mathbf{X}$.

$$\mathbf{X}_{mod} = \frac{1}{\sqrt{2(n-m)}} \begin{bmatrix} x(m+1) & \cdots & x(1) \\ \vdots & \ddots & \vdots \\ x(n) & \cdots & x(n-m) \\ x^*(1) & \cdots & x^*(m+1) \\ \vdots & \ddots & \vdots \\ x^*(n-m) & \cdots & x^*(n) \end{bmatrix} \tag{10}$$

As for the second line of the code, the MUSIC algorithm is executed. The arguments are described in order as follows: `R` is the autocorrelation matrix that has just been derived, `2` is an indication to the MUSIC algorithm of the number of spectral peaks, `[]` sets the fft length to its default length of 256, `1` corresponds to the sampling frequency, `'corr'` indicates that `R` is a correlation matrix. The MUSIC algorithm uses the orthogonality of the noise and deterministic(signal) spaces to split the correlation matrix into the corresponding orthogonal subspaces. p, the model order, in the case of the code above is equal to two. The algorithm calculates the eigenvectors that span the noise subspace, and take the $M - p$ eigenvectors $\{\mathbf{u}_{p+1}, ...\mathbf{u}_{M-p}\}$ corresponding to the smallest eigenvalues. The estimate of the PSD is then constructed according to equation :

$$\hat{P}_{music}(\omega) = \frac{1}{\sum_{i=p+1}^{M} |\mathbf{u}_i^H \mathbf{e}|^2} \tag{11}$$

where $\mathbf{e}$ is the set of complex exponential that span the signal subspace. The outputs `S` and `F` correspond to the calculated PSD estimates, and it's corresponding normalised frequencies. The third line of the code plots the estimate, and limits the normalised frequency range from 0.25-0.45 πradians/sample. From Figure 8 it can be seen that the MUSIC algorithm is able to identify the two peaks, even with a process length of less than $N = 45$. The downsides of this algorithm are that it requires knowledge of the order of process model (number of peaks). It also exhibits a far larger variance than the periodogram. Finally, compared to the periodogram, it has a far greater computational complexity.

1.4 Spectrum of Autoregressive Processes

1.4.1 Part a)

The AR parameters, $\mathbf{a}$, of an autoregressive (AR) process is derived from the Yule-Walker equation:

$$\mathbf{R}_{xx}\mathbf{a} = \mathbf{r}_x \Rightarrow \mathbf{a} = \mathbf{R}_{xx}^{-1}\mathbf{r}_x \tag{12}$$

Where $\mathbf{R}_{xx}$ is the autocorrelation matrix of the signal, $\mathbf{x}$. To guarantee inversion, $\mathbf{R}_{xx}$ must be positive definite. The biased estimator guarantees this property since it can be shown that the eigenvalues

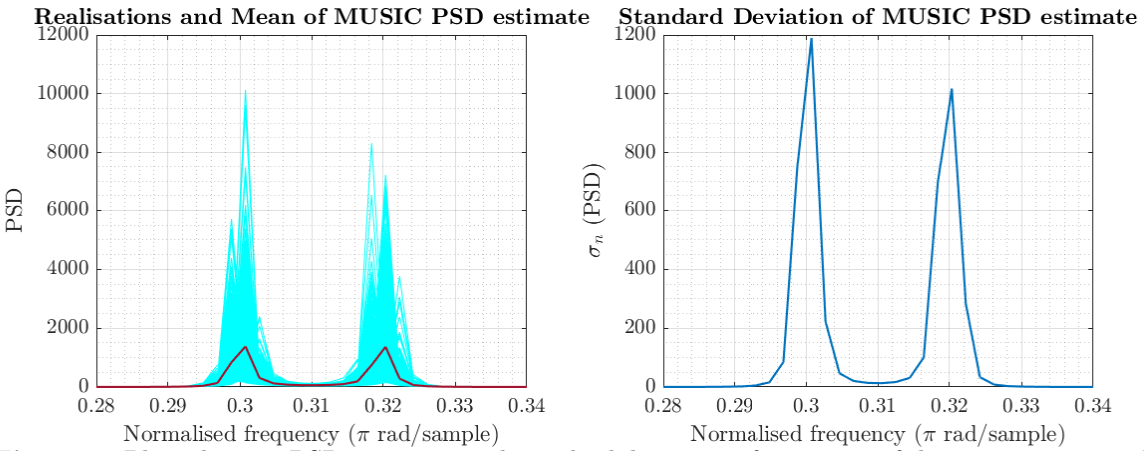


Figure 8: Plots showing PSD estimates and standard deviation of estimates of the same exponential signal of the previous section, for a process length of 14, using the MUSIC algorithm

associated with it's ACF, are always positive. On the other hand, the fact that the unbiased estimator does not decay to zero at large lags means that the positive definite condition may not be guaranteed, upon which, the matrix will be uninvertible and the parameters **a**, will not be able to be derived.

1.4.2 Part b and c)

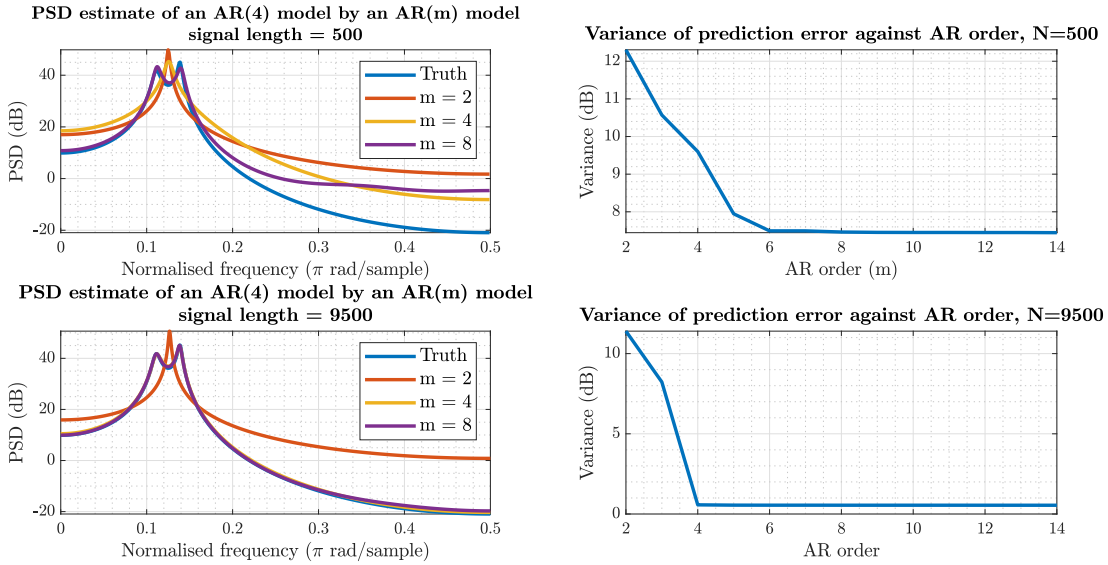


Figure 9: Spectral estimates by AR models of different orders along with their corresponding prediction errors, for different process lengths.

The AR(4) process, where $x(n) = 2.76x(n-1) - 3.81x(n-2) + 2.65x(n-3) - 0.92x(n-4) + w(n)$, and $w(n)$ is white Gaussian noise of mean 0, and variance 1, is realised for 1000 and 10000 samples respectively, with the final 500 samples of each realisation being removed to negate the transient effects of the AR filter that generates the sequence. Using the Yule-Walker equations, the parameters of the model are estimated for different estimation model orders, with the resulting PSDs of the estimated models being shown in Figure 9. For both process lengths, an AR(2) prediction model, due to being under parametrised, is unable to model the two peaks of the true PSD. For the 1000 sample realisation, even though the estimation model is of the same order of the true model, it is unable to capture the two peaks of the true spectrum, whereas, for a signal length of 10000, it's able to capture the two peaks well, suggesting that the resolution of the power spectrum is also important. For the shorter process length, when overfitting to the data for a model order of 8 (variance of prediction error flattens after this point), the estimate is able to capture the two peaks, however there's a significant discrepancy between the two 'tails' of the true and estimated PSDs. For the longer process, increasing model order past 4 gives no improvement, at which point the estimate is managing to match the true PSD very

well.

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

1.5.1 Part a)

The ECG of a subject was recorded for 3 separate breathing trials, the nature/properties of which are to be determined by examining the PSD of their corresponding Respiratory Rate intervals (RRI), calculated by taking the time-difference between successive peaks in the ECG. PSD estimates of the normalised and detrended RRI data were obtained via the standard periodogram method and Bartlett's method of averaged periodograms (using MATLAB's `pwelch` function), using Hamming windows of length 50 and 150 seconds, the results of which are shown in Figure 10.

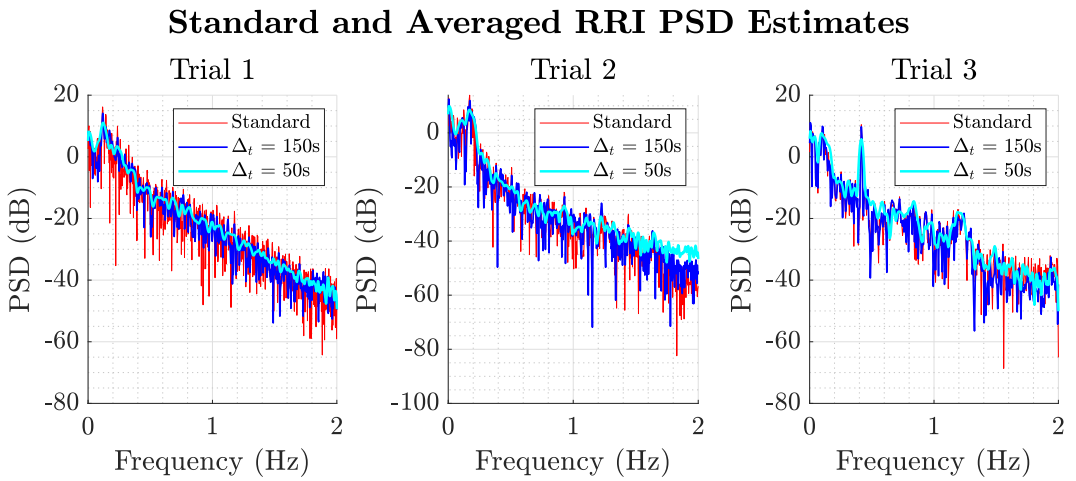


Figure 10: PSD estimations of RRI data using standard periodogram and windowed and averaged periodograms for three different trials.

1.5.2 Part b)

Each trial has a distinguishable peak in their PSD, which correspond to the breathing rate of the subject. For the first trial, the peak is located at 0.113 Hz which corresponds to around 13.5 breaths per minute (bpm). The fact that this peak is almost indistinguishable, and corresponds to a normal respiration rate for an adult, suggests that during this trial the subject's breathing was unconstrained. For the second trial, the peak is more discernible, and is found at a frequency of 0.175 Hz, which corresponds to 21bpm, above the normal breathing rate, suggesting that the subject's breathing during this trial was constrained to around 21bpm. In the final trial, the peak is found at 0.421 Hz, corresponding to 50.5bpm. Just as discussed in Section 1.2.2, the averaged periodogram reduces the variance of the PSD at the cost of an additional bias, which is observed in Figure 10.

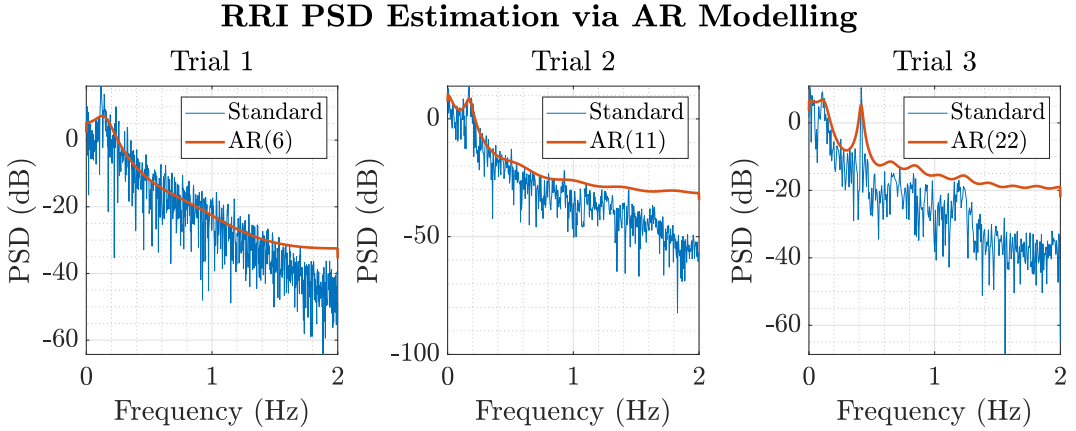


Figure 11: PSD estimates of RRI data using standard periodogram and AR estimation model.

1.5.3 Part c)

The minimum AR model orders for which the nature of the dominant peak is appropriately captured for trials 1, 2, and 3 were found to be 6, 11, and 22 respectively. The PSD's of the estimated models are shown in Figure 11 along with the PSD estimate using the standard periodogram. It can be seen that the AR estimation models provide a deterministic (noise-removed) representation of the RRI PSDs, making the identification of peaks much easier, however this comes at a cost of an inability to capture the high frequency roll-off of their respective PSDs.

1.6 Robust Regression

1.6.1 Part a)

The singular values of the input dataset, $\mathbf{X}$, and noise corrupted input dataset, $\mathbf{X}_{noise}$ are shown in Figure 12, along with their corresponding squared error. The rank of the matrix $\mathbf{X}$ is equal to its number of non-zero eigenvalues/singular values, which from the graph, is equal to 3. As for $\mathbf{X}_{noise}$, singular values (SVs) beyond the third dimension are no longer zero, however, there is still a significant discrepancy between the singular values of the first 3 dimensions and the remaining dimensions. Looking at the squared error, it can be seen that the introduction of noise introduces a far greater error to the SVs corresponding to a true value of 0, this means that as noise power approaches the non-zero SVs of $\mathbf{X}$, the rank of $\mathbf{X}_{noise}$ will no longer be distinguishable.

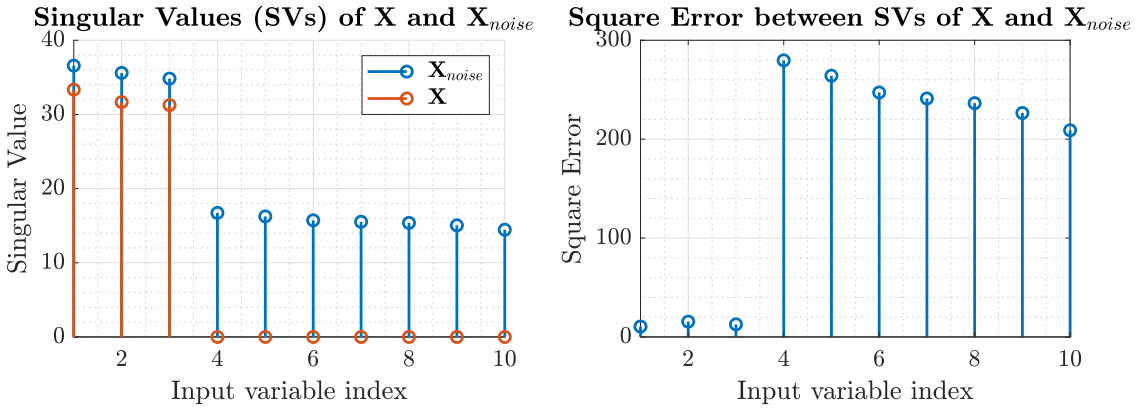


Figure 12: Graphs showing the Singular Values of $\mathbf{X}$ and $\mathbf{X}_{noise}$ along with their corresponding squared difference/error.

1.6.2 Part b)

In the previous part, the rank of the matrix $\mathbf{X}$ and thus $\mathbf{X}_{noise}$ was found to be 3. The data is now projected onto the 3 most significant principal components to obtain a low-rank approximation of $\mathbf{X}_{noise}$, denoted by $\tilde{\mathbf{X}}_{noise}$. The corresponding MSE between $\mathbf{X}$ and $\mathbf{X}_{noise}$, and $\mathbf{X}$ and $\tilde{\mathbf{X}}_{noise}$ are shown in Figure 13. The redundant data that is corrupted by noise is discarded in $\tilde{\mathbf{X}}_{noise}$, and as a result, the MSE is significantly lower for almost every variable.

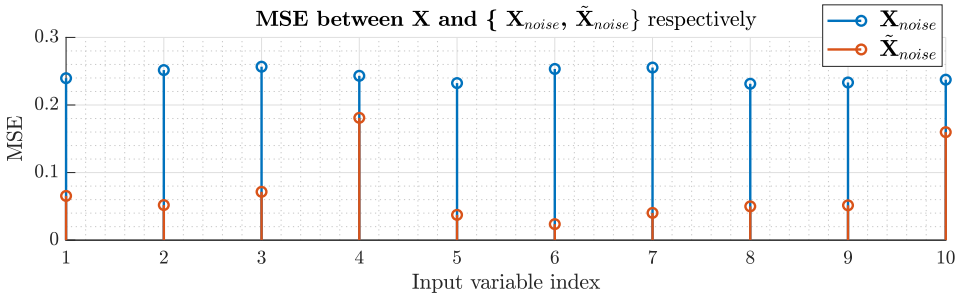


Figure 13: MSE between the original input data $\mathbf{X}$, the noise corrupted $\mathbf{X}_{noise}$ and the low-rank approximation of the noise corrupted data, $\tilde{\mathbf{X}}_{noise}$.

1.6.3 Part c)

The output data is obtained via equation 13, where $\mathbf{B}$ is an unknown regression matrix to be determined and $\mathbf{N_Y}$ is the noise matirx, whose samples are drawn from a zero-mean Gaussian distribution.

$$\mathbf{Y} = \mathbf{XB} + \mathbf{N_Y} \tag{13}$$

The regression matrix is estimated via two different methods, Ordinary Least Squares and Principle Component Regression, given by equations 14, and 15 respectively.

$$\hat{\mathbf{B}}_{OLS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{14}$$

$$\hat{\mathbf{B}}_{PCR} = \mathbf{V}_{1:r}(\mathbf{\Sigma}_{1:r})^{-1}\mathbf{U}_{1:r}^T\mathbf{Y} \tag{15}$$

The estimated outputs $\hat{\mathbf{Y}}_{OLS}$, $\hat{\mathbf{Y}}_{PCR}$, are now obtained where: $\hat{\mathbf{Y}}_{OLS} = \mathbf{X}_{noise}\hat{\mathbf{B}}_{OLS}$ and $\hat{\mathbf{Y}}_{PCR} = \mathbf{X}_{noise}\hat{\mathbf{B}}_{PCR}$. The corresponding squared error between the estimated outputs and the true observed output 'training' data, $\mathbf{Y}$ is shown on the left graph in Figure 14. Test estimates are obtained by multiplying test inputs $\mathbf{X}_{test}$ with the estimated regression matrices. They are then compared with the ground truth test data, $\mathbf{Y}_{test}$, their squared error being shown in the middle graph of Figure 14. Both methods show a very similar performance, however it seems that by a very small margin, the OLS performs better than PCR for the training data, and worse for the test data, suggesting that the OLS is more prone to overfitting to the noise in the training data.

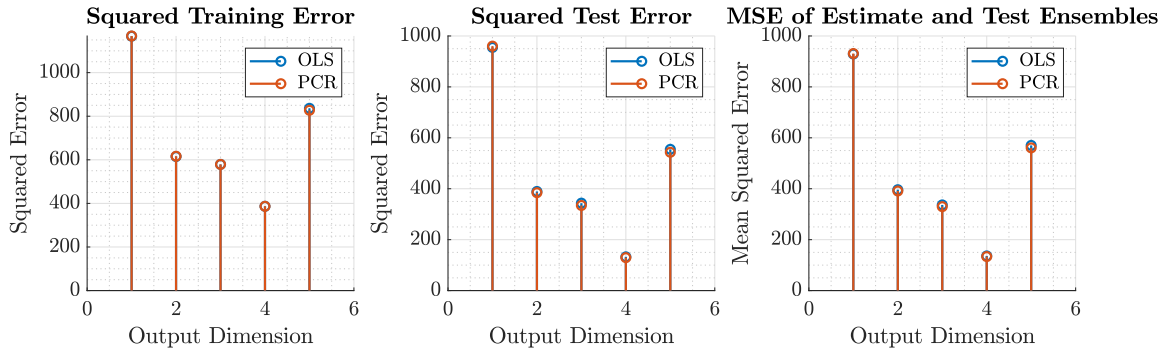


Figure 14: Error analysis of various experiments on estimated outputs and training and test data.

1.6.4 Part d)

Finally, 100 new test data realisations and their corresponding estimates (using the estimated regression matrices found in the previous section), are generated with the provided `regval` script. The MSE for both methods was calculated over the ensemble of generated test data, the results of which are shown in the far right graph of Figure 14. By taking the ensemble MSE, there is less variance in the resulting value of error, meaning that a more reliable conclusion can be made regarding the performance of the algorithms, especially since their values are so close to each other. Looking at the results, it can be seen that indeed PCR is performing better with out-of-sample data, however, the difference between the performance of the methods is very small, and the selection of method should be considered in tandem with other design considerations, such as computational complexity—the SVD required for PCR may become arduous for much higher-rank input data.

2 Adaptive signal processing

2.1 The Least Mean Square Algorithm (LMS)

2.1.1 Part a)

In this section, the following true AR(2) process is considered:

$$x(n) = a_1x(n-1) + a_2x(n-2) + w(n) \qquad w(n) \sim N(0, \sigma_w^2) \qquad a_1 = 0.1,\ a_2 = 0.8,\ \sigma_w^2 = 0.25 \tag{16}$$

The process is realised, and the parameters will be attempted to be estimated using adaptive linear filtering via the LMS algorithm. To begin with, the autocorrelation matrix is of this process is considered and is denoted by:

$$\mathbf{R}_{xx} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(1) & r_{xx}(0) \end{bmatrix} \tag{17}$$

The autocorrelation function, $r_{xx}(k)$ is defined as:

$$r_{xx}(k) = \mathbb{E}[x(n)x(n-k)] = a_1\,\mathbb{E}[x(n-1)x(n-k)] + a_2\,\mathbb{E}[x(n-2)x(n-k)] + \mathbb{E}[w(n)x(n-k)] \tag{18}$$

Substituting the required values of k for the autocorrelation matrix, and using the fact that the autocorrelation function is even, it follows that:

$$r_{xx}(0) = a_1 r_{xx}(1) + a_2 r_{xx}(2) + \sigma_w^2 \tag{19}$$

$$r_{xx}(1) = a_1 r_{xx}(0) + a_2 r_{xx}(1) \tag{20}$$

From equation 19, it can be seen that to obtain $r_{xx}(0)$, the value of $r_{xx}(2)$ is required, which is given by:

$$r_{xx}(2) = a_1 r_{xx}(1) + a_2 r_{xx}(0) \tag{21}$$

This leaves a set of simultaneous equations that are solved providing the final expressions for $r_{xx}(0)$ and $r_{xx}(1)$:

$$\sigma_x^2 = \left(\frac{1-a_2}{1+a_2}\right)\frac{\sigma_\eta^2}{(1-a_2)^2 - a_1^2} \tag{22}$$

$$r_{xx}(1) = r_{xx}(0)\frac{a_1}{1-a_2} \tag{23}$$

The given true parameter values are then substituted to obtain the final value of the autocorrelation function, such that:

$$r_{xx}(0) = \left(\frac{0.2}{1.8}\right)\frac{0.25}{0.2^2 - 0.1^2} = 0.926 \tag{24}$$

$$r_{xx}(1) = 0.926 \times \frac{0.1}{1-0.8} = 0.463 \tag{25}$$

$$\mathbf{R}_{xx} = \begin{bmatrix} 0.926 & 0.463 \\ 0.463 & 0.926 \end{bmatrix} \tag{26}$$

For convergence of the LMS algorithm to the optimal Wiener solution, the step size, μ, must satisfy the condition:

$$0 < \mu < \frac{2}{\lambda_{max}} \tag{27}$$

In this case, λ_{max} is the largest eigenvalue of the correlation matrix. The eigenvalues of the correlation matrix are given by, $\lambda_1 = 0.463$ and $\lambda_2 = 0.1389$, hence, for convergence the step size must satisfy: $0 < \mu < 1.44$ (2 d.p.).

2.1.2 Part b)

The LMS algorithm (equation 28) is now used to obtain predictions for the AR parameters $[a_1 a_2]^T = \mathbf{w}$. This experiment was conducted by first realising the process using MATLAB's inbuilt `filter` function (1000 sample realisation), and then obtaining LMS estimates by implementing equation 28 as a function. This is first done for a single trial, and then by taking the average of 100 realisations, both of which are plotted, and shown in Figure 15.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{x}(n) \tag{28}$$

For a single realisation, it is difficult to draw any conclusions regarding the convergence of the estimator or the effect of the step size, since the variance of the squared error is so high. By repeating the process 100 times and averaging the resulting squared errors, the variance of the squared error plot is

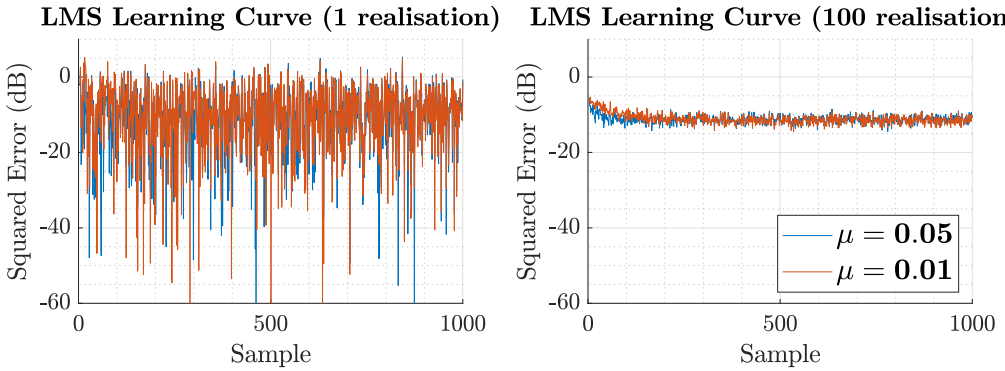


Figure 15: Squared prediction error for 1 realisation (left) and averaged over 100 realisation (right) for different step sizes.

significantly reduced, and hence the behaviour of the estimator is much more discernible. As expected, since both step sizes satisfy the convergence condition (equation 27), they both converge. Additionally, for a larger step size ($\mu = 0.05$), the estimator converges much faster (after around 100 samples) than for a smaller step size (converges after around 200 samples), but exhibits a higher variance in its prediction squared error.

2.1.3 Part c)

The Excess Mean Squared Error (EMSE) is now introduced in order to capture and evaluate the excess error of the varying LMS weights about the optimal Wiener solution, as seen in Figure 15. The EMSE captures this by negating the optimal error power from the MSE, such that:

$$MSE = \sigma_w^2 + EMSE \tag{29}$$

Additionally, the theoretical misadjustment $\mathcal{M}$ is defined as the ratio between the EMSE and the theoretical minimum error of the optimal Wiener filter.

$$\mathcal{M} = \frac{EMSE}{\sigma_\eta^2} = \frac{MSE - \sigma_w^2}{\sigma_w^2} \tag{30}$$

For small step-sizes, the misadjustment of the LMS algorithm can be approximated as:

$$\mathcal{M}_{LMS} \approx \frac{\mu}{2}\text{Tr}\{\mathbf{R}\} \tag{31}$$

As before, the same two step sizes are considered, $\mu_1 = 0.05$ and $\mu_2 = 0.01$. Considering equation 31 and using the previously calculated autocorrelation matrix $\mathbf{R}$, the theoretical misadjustments for the two step sizes are calculated to be $\mathcal{M}_1 = 0.0463$ and $\mathcal{M}_2 = 0.00926$. Using equation 30, the empirical misadjustments are now estimated by taking the average MSE of the LMS over 100, 100,000 sample realisations of the original AR process, the results of which are shown in Table 3.

Step size (μ)	$\mathcal{M}_{empirical}$	$\mathcal{M}_{theoretical}$
0.01	0.0512	0.0463
0.05	0.00807	0.00926

Table 1: Comparison of theoretical and empirical LMS misadjustments.

As expected, the empirical misadjustments are similar, but not equal to (a finite realisation of the process is used) their corresponding theoretical values, which show that the EMSE and misadjustment are larger for a larger step size. This property can be understood from a high-level point-of-view; for a larger step size, the 'jump' the algorithm makes in its estimated value is greater, which corresponds to a higher overall variance. A larger step size allows for faster convergence at the cost of greater uncertainty.

2.1.4 Part d)

The steady state adaptive filter weights (for each step size) are examined by taking the average filter value at each time step over 100, 2000 sample realisations of the AR(2) process. The final steady-state values are shown in the table below, and the filter weights at each time step are shown in Figure 16.

Parameter	True Value	$\mu = 0.05$	$\mu = 0.01$
a_1	0.1	0.0689	0.0907
a_2	0.8	0.717	0.775

Table 2: Final AR(2) parameter LMS estimates compared to real values.

It can be clearly seen that for both parameter estimates, the LMS converges to an estimate far closer to the true value when the step size is smaller ($\mu = 0.01$), it however, as observed and discussed before, comes at a cost of slower convergence. The reason for the larger error offset for a larger step value is due to the fact that corrections may not be fine enough to converge to the optimal solution, and thus estimates bypass and oscillate about the true value.

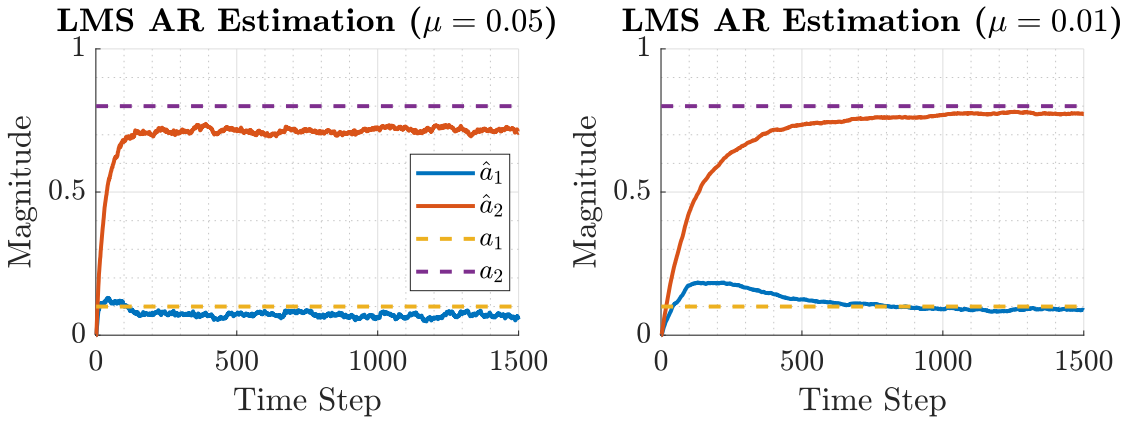


Figure 16: AR(2) parameter estimates using the LMS algorithm with different step size values.

2.1.5 Part e)

To allow for stability in the case of zero-valued eigenvalues of the autocorrelation matrix $\mathbf{R}_{xx}$, the leakage coefficient, γ is introduced as a regularisation parameter, meaning that the cost function $J_2(n)$ now becomes

$$J_2(n) = \frac{1}{2}(e^2(n) + \gamma\|\mathbf{w}(n)\|_2^2) \tag{32}$$

$$= \frac{1}{2}e^2(n) + \frac{\gamma}{2}\mathbf{w}^T(n)\mathbf{w}(n) \tag{33}$$

For gradient descent, the derivative of $J_2(n)$ with respect to the LMS coefficients $\mathbf{w}(n)$ is derived:

$$\nabla_{\mathbf{w}} J_2(n) = \frac{1}{2}\frac{\partial e^2(n)}{\partial e(n)}\frac{\partial e(n)}{\partial y(n)}\frac{\partial y(n)}{\partial \mathbf{w}(n)} + \frac{\gamma}{2}\frac{\partial}{\partial \mathbf{w}}(\mathbf{w}^T(n)\mathbf{w}(n)) \tag{34}$$

$$= -\frac{1}{2}(2e(n))(\mathbf{x}(n)) + \gamma\mathbf{w}(n) = -e(n)\mathbf{x}(n) + \gamma\mathbf{w}(n) \tag{35}$$

Considering the fact that the global minimum of a convex function always lies in its negative gradient direction, the update (or gradient descent) equation is written as:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\nabla_{\mathbf{w}} J_2(n) \tag{36}$$

The derived expression for the gradient is substituted into equation 36 to obtain the final leaky LMS update equation, shown in equation 38, as required.

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu[\gamma\mathbf{w}(n) - e(n)\mathbf{x}(n)] \tag{37}$$

$$= \mathbf{w}(n)[1 - \gamma\mu] + \mu e(n)\mathbf{x}(n) \tag{38}$$

2.1.6 Part f)

Using the Leaky-LMS algorithm defined in the previous part, the parameters of the same AR(2) process as defined in equation 16, are estimated for different values of μ and γ over 100 realisations, the results of which are shown in Figure 17. The only estimate that appears to converge to its correct true value is the estimate for a_1 with a step size of 0.05, the other estimates fail to converge to their respective true values. Furthermore, it can be seen that for a larger value of γ, the deviation from the true value increases, this can be explained by observing the optimal solution of the Leaky LMS compared to the optimal Wiener solution, shown in equations 39 and 40.

$$\mathbf{w}_{opt} = \mathbf{R}_{xx}^{-1}\mathbf{p} \tag{39}$$

$$\mathbf{w}_{leaky} = (\mathbf{R}_{xx} + \gamma\mathbf{I})^{-1}\mathbf{p} \tag{40}$$

Clearly, as γ approaches 0, the Leaky-LMS optimal solution approached the Wiener solution, and as it increases it deviates from the optimal solution. A larger value of both μ and the unknown parameter $\mathbf{w}(n)$ (in this case a_2) also are observed to increase the steady-state error, which can be explained by referring back to the update equation (38). A larger value of μ ,γ and $\mathbf{w}(n)$ introduce a larger nonoptimal (does not satisfy the first order necessary condition of optimality) bias to weight updates.

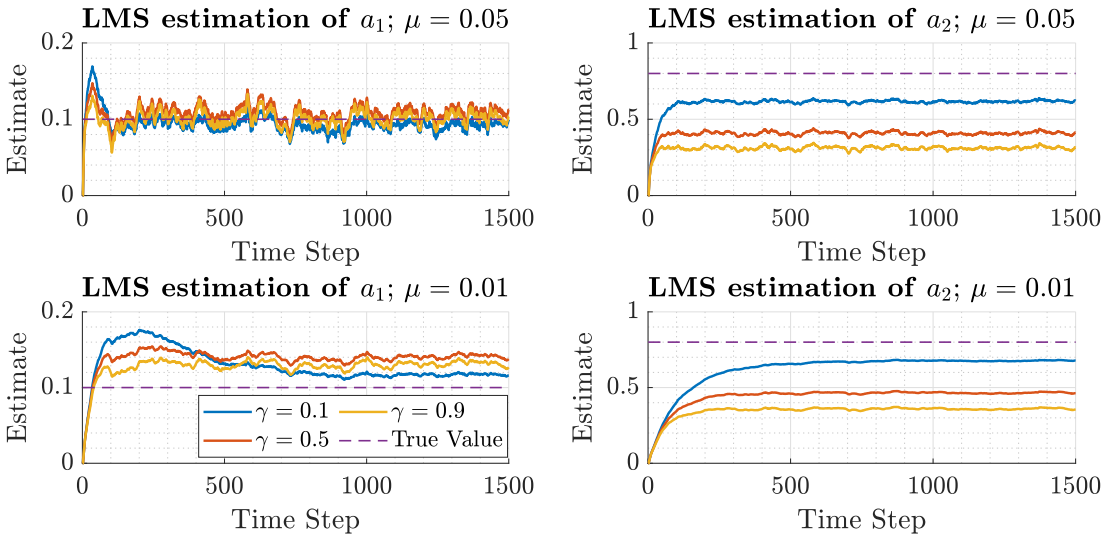


Figure 17: AR(2) parameter estimates using LMS with different learning rates μ and leakage coefficients γ.

2.2 Adaptive Step Sizes

2.2.1 Part a)

Following on from the work in the previous section, the goal is to now develop and investigate gradient adaptive step size (GASS) algorithms to find a satisfactory balance between convergence and accuracy, namely the Benveniste, Ang & Farhang and the Mathews & Xie algorithms. Each follow the general update equation, defined in equation 41 and 42, where ρ is a hyperparameter, $e(n)$ in the prediction error and $\psi(n)$ is determined by the algorithm design.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(n)e(n)\mathbf{x}(n) \tag{41}$$

$$\mu(n+1) = \mu(n) + \rho e(n)\mathbf{x}^T(n)\psi(n) \tag{42}$$

The three GASS algorithms were implemented and used in order to predict the MA(1) process defined in equation 43, their results are compared to the standard LMS (fixed μ)for different step sizes in Figure 18

$$x(n) = 0.9\eta(n-1) + \eta(n), \qquad \eta \sim N(0, 0.5) \tag{43}$$

The weight error curves are obtained by averaging the results from 100 repetitions of each algorithm, based on a 1000 sample realisation of the MA process. The hyperparameters were arbitrarily set to be: $\rho = 0.002$ and $\alpha = 0.8$. A fixed step size of $\mu = 0.01$ seems to be too small for this experiment, with the algorithm only converging towards the end of the 1000 sample sequence. For an initial learning rate of 0.2 each of the GASS algorithms performs similarly, converging faster than both the LMS algorithms, showing that the $\psi(n)$ is very similar for each of the GASS algorithms in this particular setup. For an initial learning rate of 0.1, the standard LMS algorithm with $\mu = 0.01$, performs almost identically to all 3 GASS algorithms i.e. $\psi(n)$ is close to 0 for each iteration and for each of the GASS algorithms, suggesting that little adjustment needs to be made to the initial value of μ throughout the learning process. Finally, for an initial learning rate of 0, the difference between the GASS algorithms becomes more discernible, the Mathews & Xie algorithm is slow to converge, with the Benveniste Algorithm converging the quickest, but still performing worse than a fixed learning rated of 0.1. This suggests that the Beneviste algorithm has the greatest 'freedom' for step size adjustment, allowing for the algorithm to quickly adjust towards a more optimal step size. In general, the GASS algorithms, if initialised with a reasonable value of μ, allow for quicker, and more robust convergence to the true parameter value. This improvement in performance of course comes at a cost of higher computational complexity.

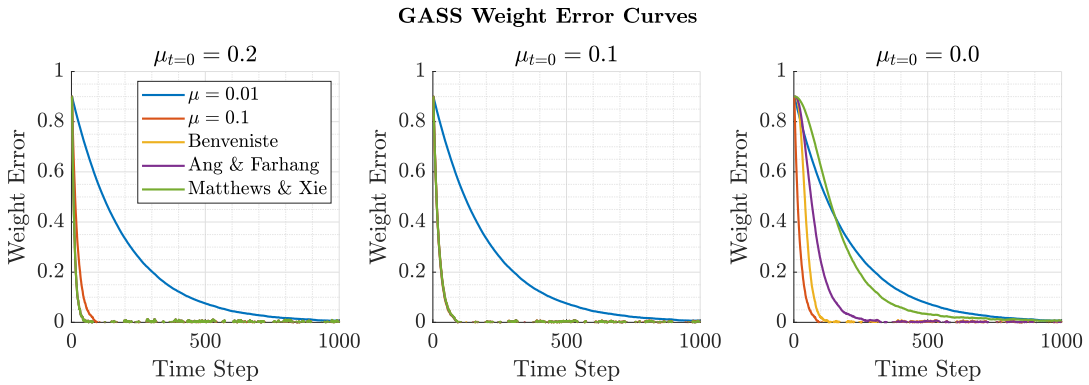


Figure 18: Weight error curves of GASS algorithms compared with different initial learning rates, $\mu_{t=0}$, compared to the standard LMS algorithm.

2.2.2 Part b)

The normalised LMS (NLMS) algorithm is now investigated, which is defined in equation 44.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\beta}{\epsilon + \|\mathbf{x}(n)\|_2^2} e(n)x(n) = \mathbf{w}(n) + \frac{\beta}{\epsilon + \mathbf{x}^T(n)\mathbf{x}(n)} e(n)x(n) \tag{44}$$

It's designed to update weights based on the *a posteriori* error $e_p(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n+1)$, such that the expression in equation 44 is equivalent to the update equation shown in equation 45.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e_p(n)\mathbf{x}(n) \tag{45}$$

Now to verify this, equation 45 is first substituted into the expression for the *a posteriori* error such that

$$\begin{aligned}
e_p(n) &= d(n) - \mathbf{x}^T(n)(\mathbf{w}(n) + \mu e_p(n)\mathbf{x}(n)) \\
e_p(n)(1 - \mu\mathbf{x}^T(n)\mathbf{x}(n)) &= d(n) - \mathbf{x}^T(n)\mathbf{w}(n) \\
e_p(n)(1 - \mu\|\mathbf{x}(n)\|_2^2) &= e(n) \\
e_p(n) &= \frac{e(n)}{(1 - \mu\|\mathbf{x}(n)\|_2^2)}
\end{aligned} \tag{46}$$

Now this expression for the *a posteriori* error is substituted back into equation 45 to obtain the final equivalent expression for the update equation:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \frac{e(n)}{1 - \mu \|\mathbf{w}(n)\|_2^2} \mathbf{x}(n)$$
$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \frac{e(n)}{\frac{1}{\mu} - \|\mathbf{w}(n)\|_2^2} e(n)\mathbf{x}(n) \tag{47}$$

Equation 47 is equivalent to the original expression for the NLMS algorithm from equation 44 for $\beta = 1$ and $\epsilon = 1/\mu$, as required.

Part c)

The Generalised Normalised Gradient Descent (GNGD) algorithm makes a slight modification to the NLMS defined in equation 44, by parametrising the regularisation term ϵ and adaptively updating it at each time step, according to equation 48.

$$\epsilon(n+1) = \epsilon(n) - \rho\mu \frac{e(n)e(n-1)\mathbf{x}^T(n)\mathbf{x}(n-1)}{(\epsilon(n-1) + ||\mathbf{x}(n-1)||^2)^2} \tag{48}$$

The performance of the GNGD algorithm ($\rho = 0.002$, $\mu_0 = 0.1$) is compared to the previously investigated Beneveniste algorithm ($\rho = 0.005$, $\mu_0 = 1$) in the same MA(1) prediction problem. The weight error averaged over 10000 repetitions of each experiment is shown in Figure 19. The GNGD algorithm converges faster than Beneveniste, but with a slightly higher variance in its estimation. As for the complexity of the two algorithms, the update equation of the Benveniste algorithm is first inspected, defined in equations 49 50. The calculation of $\psi(n)$, namely the computation of the cross-correlation matrix $\mathbf{x}(n-1)\mathbf{x}^T(n-1)$ is of order $\mathcal{O}(n^2)$, this is the most complex computation in the update equation and thus Benveniste has a complexity of $\mathcal{O}(n^2)$. On the other hand, the most complex calculation in the update equation GNGD is the computation of the dot product $\mathbf{x}^T(n)\mathbf{x}(n-1)$ and the squared norm, $||\mathbf{x}(n-1)||^2$ (equation 48), which are of order $\mathcal{O}(n)$. In short, the GNGD algorithm converges quicker than Benveniste, and has a lower computational complexity.

$$\mu(n+1) = \mu(n) + \rho e(n)\mathbf{x}^T(n)\psi(n) \tag{49}$$
$$\psi(n) = [\mathbf{I} - \mu(n-1)\mathbf{x}(n-1)\mathbf{x}^T(n-1)]\psi(n-1) + e(n-1)\mathbf{x}(n-1) \tag{50}$$

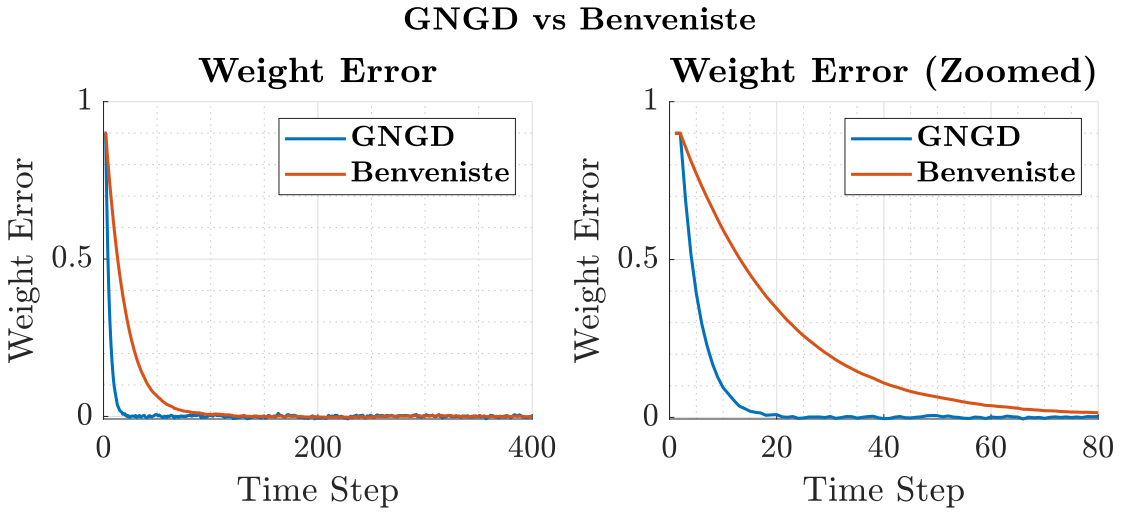


Figure 19: Weight error and squared prediction curves for GNGD and Benveniste algorithms

2.3 Adaptive Noise Cancellation

2.3.1 Part a & b) Adaptive Line Enhancer

The goal of the following analysis will be to investigate the Adaptive Line Enhancer (ALE), in predicting a signal $s(n) = x(n) + \eta(n)$ where $x(n)$ is a sinusoid of frequency 0.005Hz and $\eta(n)$ is coloured noise such that $\eta(n) = v(n) + 0.5v(n-2)$, and $v(n)$ is white noise with unit variance. The adaptive Line enhancer (ALE) is a noise reduction algorithm consisting of a delay operator, Δ followed by a linear operator which are chosen such that the linear predictor and the noise present in the signal are uncorrelated. The output of the ALE is the predicted underlying signal, $\hat{x}_\Delta(n)$-its MSE can be written as:

$$\begin{aligned}\mathbb{E}[(s(n) - \hat{x}_\Delta(n))^2] &= \mathbb{E}[(x(n) + \eta(n) - \hat{x}_\Delta(n))^2] \\ &= \mathbb{E}[(x(n) - \hat{x}_\Delta(n))^2] + \mathbb{E}[\eta^2(n)] + 2\,\mathbb{E}[x(n)\eta(n)] - 2\,\mathbb{E}[\hat{x}_\Delta(n)\eta(n)] \end{aligned} \tag{51}$$

Now, $\mathbb{E}[\eta^2(n)]$ is equal to the noise power and thus is constant, $\mathbb{E}[x(n)\eta(n)]$ is equal to zero based on the assumption that $x(n)$, a deterministic signal, is uncorrelated with noise. The term $\mathbb{E}[(x(n) - \hat{x}_\Delta(n))^2]$ will ideally be equal to 0 when an appropriate Δ is chosen, leaving the term $\mathbb{E}[\hat{x}_\Delta(n)\eta(n)]$ to be minimised, which is dependent on Δ. Considering this, the term to be minimised is determined using the fact that $\hat{x}_\Delta(n) = \mathbf{w}^T\mathbf{u}_\Delta(n)$ where $\mathbf{u}_\Delta(n)$ is the predictor input such that $\mathbf{u}_\Delta(n) = [s(n-\Delta), \dots, s(n-\Delta-M+1)]^T$:

$$\begin{aligned}\min_\Delta\ \mathbb{E}[\hat{x}_\Delta(n)\eta(n)] &= \min_\Delta\ \mathbb{E}[w^T(n)\mathbf{u}_\Delta(n)\eta(n)] \\ &= \min_\Delta\ \mathbb{E}\left[\sum_{k=0}^{M-1} w_k s(n-\Delta-k)\eta(n)\right] \\ &= \min_\Delta\ \mathbb{E}\left[\sum_{k=0}^{M-1} w_k (x(n-\Delta-k) + \eta(n-\Delta-k)\eta(n))\right] \\ &= \min_\Delta\ \mathbb{E}\left[\sum_{k=0}^{M-1} w_k \eta(n-\Delta-k)\eta(n)\right] \\ &= \min_\Delta\ \mathbb{E}\left[\sum_{k=0}^{M-1} w_k (v(n-\Delta-k) + 0.5v(n-\Delta-k-2))(v(n) + 0.5v(n-2))\right] \end{aligned} \tag{52}$$

For white noise, $\mathbb{E}[v(n-i)v(n-j)] = 0$, where for i $\neq$ j, thus for the term to be minimised $\Delta > 2$ such that there is no overlap and thus correlation between $v(n-\Delta-k) + 0.5v(n-\Delta-k-2)$ and $v(n) + 0.5v(n-2)$, confirmed in Figure 20, which exhibits significantly higher MPSE for the first two samples of delay

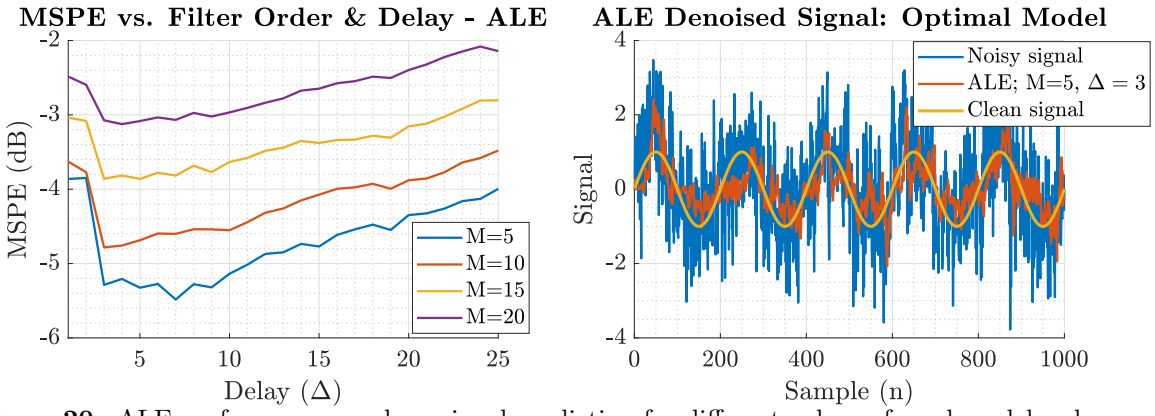


Figure 20: ALE performance on clean signal prediction for different values of and model orders, M, along with the denoised signal with optimal model settings.

Observing the relationship between MSPE and delay for M=[5, 10, 15, 20] in Figure 20, it can be seen that increasing model order leads to an approximately constant increase in MSPE for all delay

values. The ALE was then investigated over two experiments which were repeated and averaged for 100 realisations, one was fixing the model order and varying delay, and the second was fixing the delay and varying model order. Firstly, looking at the effect of delay for a fixed model of 5 (Figure 21), the MSPE stays roughly constant from $\Delta = 3$ to $\Delta = 8$, after which the MSPE begins significantly increasing. As the delay increases the decorrelation also increases, however, as a result of the delay, there is an increased phase difference between the predicted and the clean signal which outweighs the increase in decorrelation, leading to a higher overall MSPE. Between $\Delta = 3$ to $\Delta = 8$, these two effects seem to balance each other out. Prioritising the 'responsiveness' of the ALE predictor model, a delay of $\Delta = 3$ is concluded to be optimal.

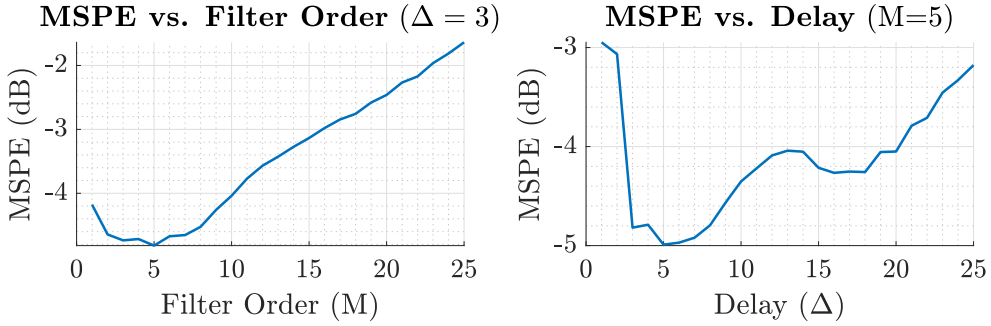


Figure 21: Left: MSPE of ALE plotted against filter order for a fixed delay $\Delta = 3$. **Right:** MSPE of ALE against delay for a fixed filter order $M = 5$.

Now looking at the relationship between model order and MSPE for a fixed delay, it can be seen that there is an 'elbow' in performance for a model order M=3, with the minimum MSPE exhibited at M=5, after which the performance begins to drop with model order at a relatively constant rate (in the log sense). The improvement in performance from M=3 to M=5 is however minimal, and given that the computational complexity significantly increases for increasing model order, the optimal model order is chosen to be M=3.

2.3.2 Part c)

The Adaptive Noise Cancellation (ANC) is now implemented and compared to the optimal ALE determined in the previous section on the same signal reconstruction problem as before (sinusoid corrupted by coloured noise). ANC requires a secondary noise input $\epsilon(n)$ which is in some way correlated to the primary noise $\eta(n)$, thus for the experiment, $\epsilon(n) = 0.7\eta(n) + 0.1\eta(n-1) + 0.1$. The performance of the two models in the signal reconstruction of $x(n)$ are compared in Figure 19, along with their MSPE measured in sliding windows of 40 samples in order to capture their time-varying properties. The performance of the ALE stays relatively constant, only improving on average by a few dB over the 1000 sample realisation. On the other hand, the ANC performs worse than ALE for the first ≈150 samples, but converges well, and exhibits a significantly lower MSPE than the ALE after this point. The performance of the ANC is heavily dependent on the correlation between the estimated/measured secondary noise input and the primary noise input, thus for cases where this may not necessarily be guaranteed/possible, or the sample length is low, the ALE will be preferred.

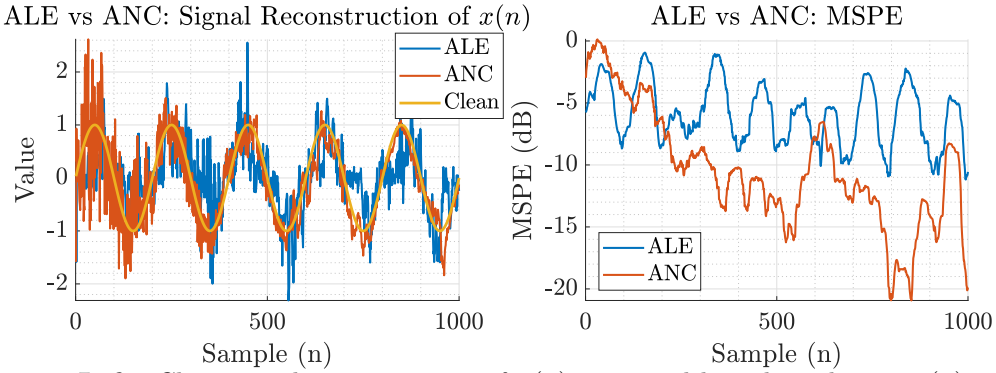


Figure 22: Left: Clean signal reconstruction of $x(n)$ corrupted by coloured noise $\eta(n)$ using the ALE and ANC algorithms. **Right:** The corresponding MSPE's for a sliding window of 40 samples.

2.3.3 Part d)

The following section will aim to apply the ANC algorithm in order to remove the 50 Hz mains interference from the Cz electrode EEG recordings without loss of information in the surrounding frequency bands. The spectrogram of the original, detrended, Cz recordings is shown in Figure 23- there's clearly a strong 50Hz component at all-time points. To ensure that only the 50Hz region is targetted, and is seen as 'noise', the secondary noise input of the ANC is chosen to be a 50Hz sinusoid, noise corrupted by WGN of variance 0.1 (this noise power achieved the best results).

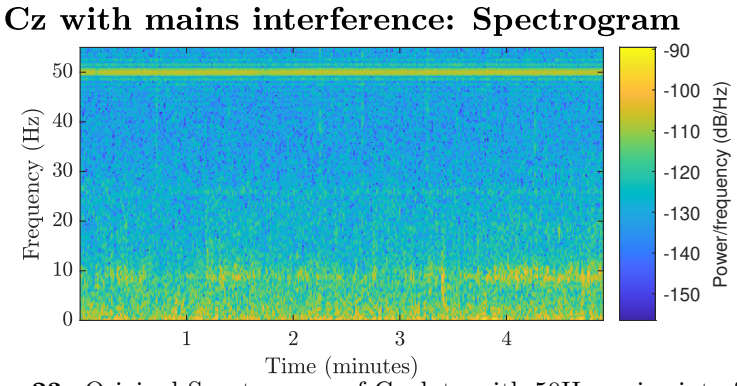


Figure 23: Original Spectrogram of Cz data with 50Hz mains interference.

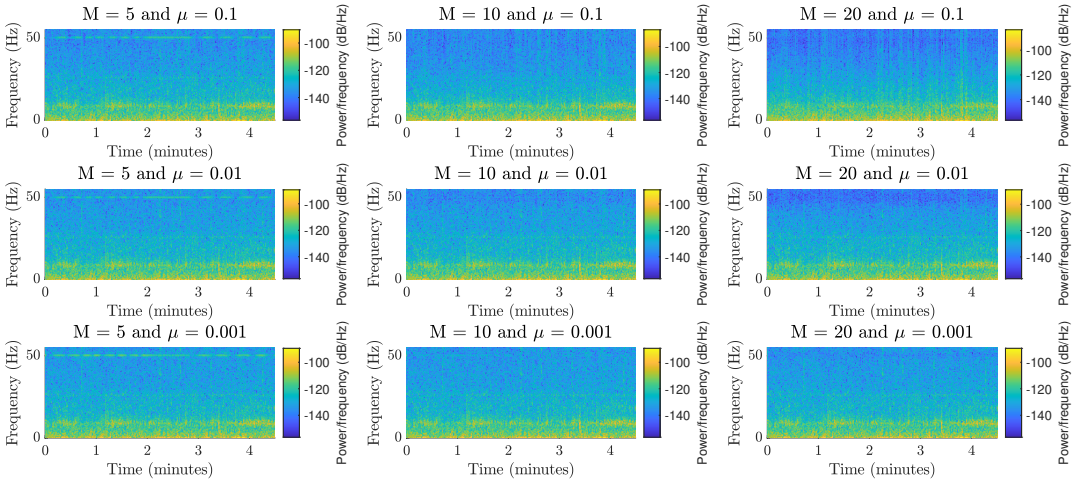


Figure 24: Spectrograms of ANC denoised Cz signals for different filter orders M and step-sizes μ.

The resulting spectrogram of the ANC denoised Cz signals are shown for different model orders and step-sizes in Figure 24. It can be seen that an increasing value of μ, attenuates a wider range of frequencies surrounding the 50Hz peak, and has little effect on the removal of the 50Hz component apart from for μ=0.001, where convergence is slow, and thus some artefacts are yet be fully removed for the same model order; this is most discernible for a model order of 10. For a model order of 5, the 50Hz component is still observable, meaning that the model is not complex enough to fit the underlying clean signal. On the other hand, for a model order of 20, the ANC model attenuates the signal too much around the 50Hz component, introducing a notch in the expense of a peak, this is best represented in Figure 25, where the PSD of the resulting denoised signal (M=5, $\mu = 0.01$) is plotted alongside the original signal's PSD. Considering these observations, a model order of M=10, and a step size of $\mu = 0.01$ is determined to be the optimal choice of hyperparameter values. The resulting PSD of the denoised signal, using these settings, is also plotted alongside the PSD of the original signal in Figure 25. The model does not completely remove the peak, it is still visible, however it is heavily attenuated, and is no longer the dominant peak in the spectrum. Furthermore, it has a minimal effect on the surrounding frequencies.

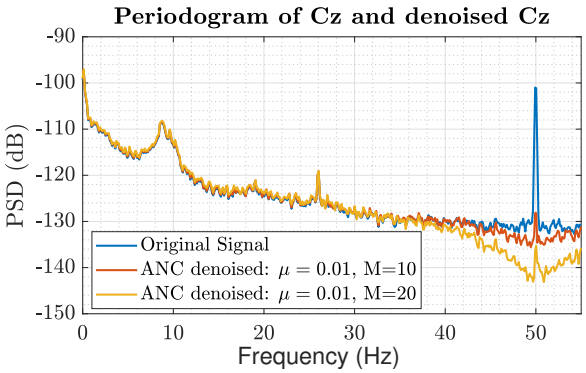


Figure 25: PSD of the denoised Cz signal with optimal, and suboptimal model order and step size, along with the PSD of the original Cz signal.

3 Widely Linear Filtering and Adaptive Spectrum Estimation

3.1 Complex LMS and Widely Linear Modeling

3.1.1 Part a)

Compared to real-valued signals, complex valued signals provide another dimension to their representation, allowing for a representation of direction as well as magnitude, which can be extremely useful in a multitude of problems. To begin with, the widely-linear-moving-average first-order process, WLMA(1), driven by circular white Gaussian noise is considered:

$$y(n) = x(n) + b_1 x(n-1) + b_2 x^*(n-1), \qquad x \sim \mathcal{N}(0,1) \tag{53}$$

Where $b_1 = 1.5 + 1j$ and $b_2 = 2.5 - 0.5j$. A 1000 sample realisation of the WLMA(1) process is generated in MATLAB. The WGN and the resulting output of the process are plotted onto the complex plane in Figure 26 (Left). It can be seen that WGN is indeed a circular process, and that the output of the process, $y(n)$, is non-circular. The circularity coefficient, ρ, is a ratio between the pseudocovariance/properness of a signal and it's covariance, and gives an indication of the degree of circularity of data, and is defined in equation 54. For circular data, the pseudocovariance is zero, and thus ρ is expected to be small. The pseudocovariance for the input $x(n)$ is calculated to be 0.0646, and $\rho = 0.8316$ for $y(n)$ verifying the previous observation regarding their respective circularities.

$$\rho = \frac{E[zz^*]}{E[zz^T]} \tag{54}$$

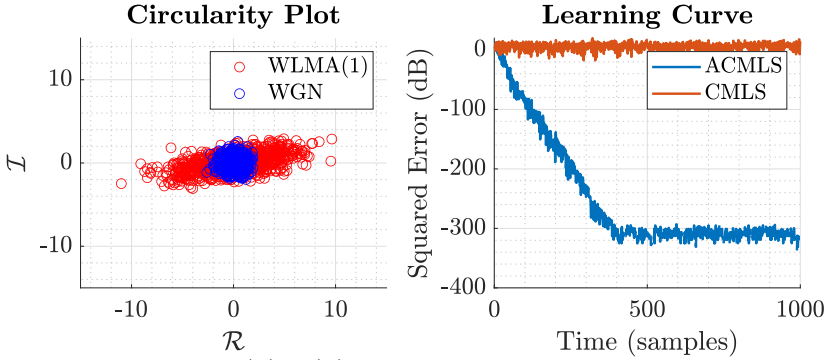


Figure 26: Circularity plots of $x(n)$, $y(n)$, and learning curve for ACMLS and CMLS algorithms ($\mu = 0.1$)

The Complex LMS (CLMS) and the Augmented CLMS (ACLMS) are investigated to find out their ability to model the non-circular process. The CLMS, due to its inability to sufficiently capture the second-order statistics, and therefore the pseudocovariance matrix, can only effectively model circular data, and therefore is expected to perform badly in this task. The ACLMS has the sufficient degrees of freedom to capture the pseudocovariance matrix of the process, and thus is expected to perform much

better than the CLMS. The two algorithms were employed in a prediction problem for a 1000 sample realisation of $y(n)$, with their respective squared prediction errors being shown in Figure 26 (right). The results clearly confirm our predictions regarding their performances, with CLMS converging to an error of $\approx -8dB$ and the ACLMS converging to an error of $\approx -305dB$.

3.1.2 Part b)

Following from the discussion in the previous section, real-world bivariate data in the form of wind speeds given from East-West, v_{east}, and North-South, v_{north}, is now considered, and is modelled as a complex valued signal, shown in equation 55:

$$v[n] = v_{east}[n] + jv_{north}[n] \tag{55}$$

Three different wind regimes are included in the considered dataset, low wind-speeds, medium wind-speeds and high-wind speeds, the circularity plots of which are shown in Figure 27. Visually, it can be seen as the wind speed is reducing, the data is increasing in circularity (data is more centred around 0), and this is indeed confirmed when the circularity coefficients for each set of data is calculated ($\rho_{low} = 0.159$, $\rho_{med} = 0.454$ and $\rho_{high} = 0.623$). As a result, only low-wind speeds can really be approximated as being proper. One step ahead prediction was then performed on each of the

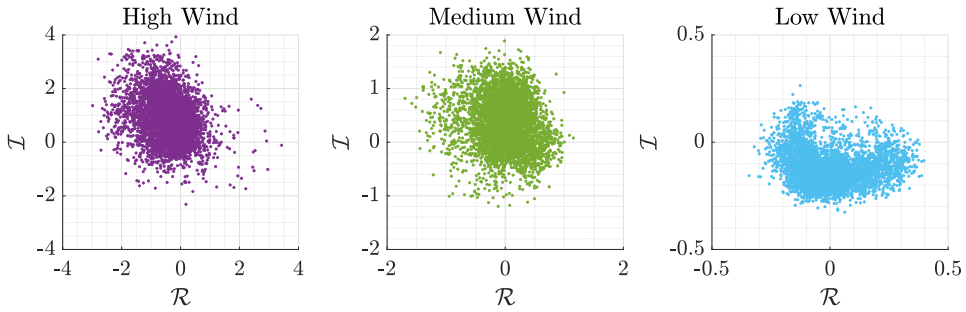


Figure 27: Circularity plots of wind data for the three given wind regimes.

wind regimes using the CLMS and ACLMS adaptive filters. The optimal learning rates that allowed for sufficiently fast convergence and minimum prediction error for each of the wind regimes were empirically found to be, $\mu_{high} = 0.001$, $\mu_{medium} = 0.007$ and $\mu_{low} = 0.1$. Using these optimal step sizes, the prediction problem was repeated for each filter for model orders $M = \{1, 2, .. 20\}$, with the resulting MPSE being plotted and shown in Figure 28. As expected, since the data is not circular, and the ACLMS filter is able to sufficiently capture the statistics of the pseudocovariance matrix, for low and optimal model orders, the ACLMS outperforms CLMS for each of the wind regimes. Furthermore, both models perform better on more circular data, namely for low wind-speeds. The ACLMS achieves optimal performance at slightly lower model orders, which means that it will be slightly preferable in terms of computational efficiency. As the model order increases however, the relative decrease in performance of the ACLMS is greater than the CLMS, this likely as a result of higher degrees of freedom in ACLMS, meaning that it is more susceptible to overfitting to noise.

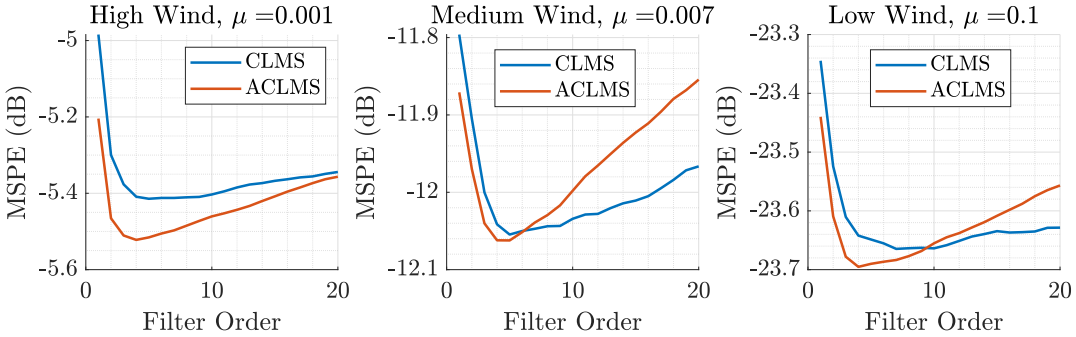


Figure 28: MSPE vs. filter order for the three different wind regimes and filters

3.1.3 Part c)

A three-phase power system, composed of different amplitudes ($V_a(n)$, $V_b(n)$,$V_c(n)$) and phase distortions (Δ_b, Δ_c) is now considered, and defined in equation 56.

$$\begin{bmatrix} v_a(n) \\ v_b(n) \\ v_c(n) \end{bmatrix} = \begin{bmatrix} V_a(n)cos(2\pi\frac{f_o}{f_s}n + \phi) \\ V_b(n)cos(2\pi\frac{f_o}{f_s}n + \phi + \Delta_b - \frac{2\pi}{3}) \\ V_c(n)cos(2\pi\frac{f_o}{f_s}n + \phi + \Delta_b + \frac{2\pi}{3}) \end{bmatrix} \tag{56}$$

Where f_o is the system frequency, f_s is the sampling frequency and ϕ is a constant phase shift. The Clarke Transform is then used to project the three-dimensional phase voltages onto two orthogonal axes, forming I and Q components ($v_a(n)$ and $v_b(n)$) that can be represented on the complex plane such that the three-phase voltage system can be represented as a Clarke voltage $v(n)$, where $v(n) = v_\alpha(n) + jv_\beta(n)$. If the system is balanced, i.e. $V_a(n) = V_b(n) = V_c(n)$, and $\Delta_b = \Delta_c = 0$, then $|v(n)|$ is constant such that the complex Clarke voltage is circular. To investigate the effect on the complex Clarke voltage due to different system imbalances, a three-phase system was generated with $f_0 = 50$Hz, $f_s = 5000$ samples/sec and $\phi = \pi/3$ for changing peak voltages and phase distortions, with the corresponding circularity plots and coefficients being shown in Figure 29.

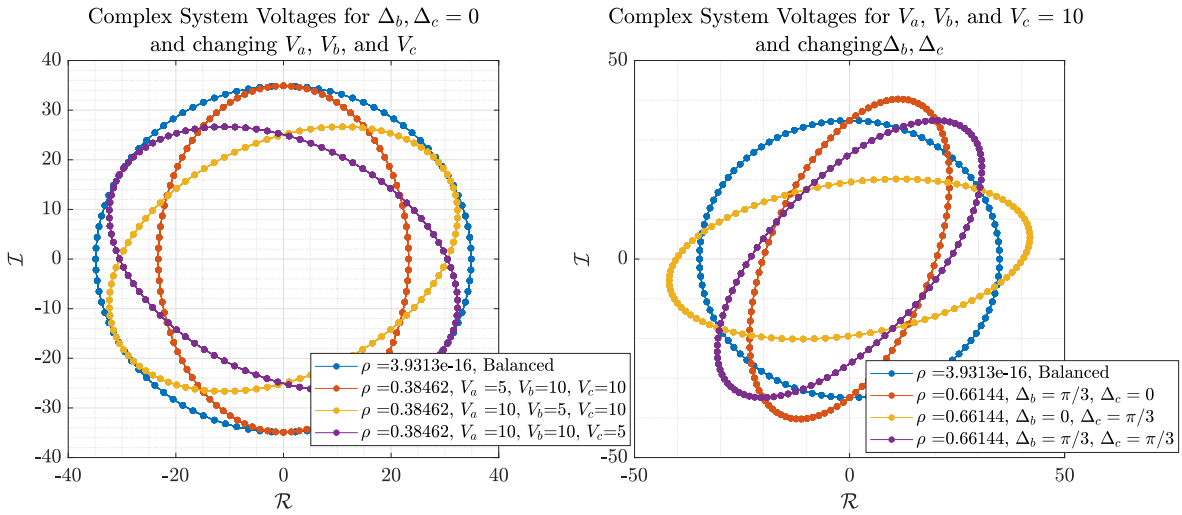


Figure 29: Balanced vs. Unbalanced complex Clarke voltage circularity plots.

A system imbalance, regardless of which parameters it has arisen from (amplitude or phase distortion), is reflected in an increase of the circularity coefficient, and monitoring its value can be used to immediately detect any system imbalances due to faults as well as its severity (higher coefficient means corresponds to a more severe fault). For a change in Peak Voltage the Clarke voltage is unchanged at phases perpendicular to the phase of the voltage whose amplitude has been changed (expected since its voltage value will be equal to 0 at this point anyway), and it's value at the phase of the corresponding voltage component is 'squashed'. Using these features, the fault can be determined if it has arisen from a change in one of the peak voltages, as well as which specific one (only if exactly one has changed).

3.1.4 Part d)

The strictly linear and widely linear autoregressive models of order 1, as described in equations 57 and 58 are used to compute the frequency f_0 of a balanced and unbalanced three-phase voltage system (as described in previous section) with complex Clarke voltage $v(n)$.

$$\text{Strictly linear: } v(n+1) = h^*(n)v(n) \tag{57}$$

$$\text{Widely linear: } v(n+1) = h^*(n)v(n) + g^*(n)v^*(n) \tag{58}$$

Balanced: To begin with, a balanced system is considered, in such a case, the complex Clarke voltage at time instant n can be written as:

$$v(n) = \sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_0}{f_s}(n)+\phi)} \tag{59}$$

Where f_s is the sampling frequency and ϕ is a constant phase shift, as before. Substituting this into the expression for a strictly linear model (equation 57), it follows that:

$$\begin{aligned}
\sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_0}{f_s}(n+1)+\phi)} &= h^*(n)\sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_0}{f_s} n+\phi)} \\
e^{j(2\pi \frac{f_0}{f_s}(n+1)+\phi)} &= h^*(n) e^{j(2\pi \frac{f_0}{f_s} n+\phi)} \\
e^{j(2\pi \frac{f_0}{f_s} n+\phi)} e^{j(2\pi \frac{f_0}{f_s})} &= h^*(n) e^{j(2\pi \frac{f_0}{f_s} n+\phi)} \\
e^{j(2\pi \frac{f_0}{f_s})} &= h^*(n)
\end{aligned} \tag{60}$$

Expressing $h(n)$ in terms of amplitude and phase, such that $h(n) = |h(n)|e^{j\phi_h}$ and $h^*(n) = |h(n)|e^{-j\phi_h}$, where the phase $\phi_h = \arctan\left(\frac{\Re\{h(n)\}}{\Im\{h(n)\}}\right)$, equation 60 can be rewritten as:

$$e^{j(2\pi \frac{f_0}{f_s})} = |h(n)| e^{j \cdot \arctan\left(\frac{\Re\{h(n)\}}{\Im\{h(n)\}}\right)} \tag{61}$$

The two complex values are equal only if their magnitudes are equal, therefore $|h(n)| = 1$, leaving:

$$f_0(n) = \frac{f_s}{2\pi} \arctan\left(\frac{\Re\{h(n)\}}{\Im\{h(n)\}}\right) \tag{62}$$

Noting that the sign of $\left(\frac{\Re\{h(n)\}}{\Im\{h(n)\}}\right)$ is discarded since it only determines the direction of rotation.

Unbalanced: Now considering an unbalanced system, the complex Clarke voltage can be written as:

$$v(n) = A(n)e^{j(2\pi \frac{f_0}{f_s} n+\phi)} + B(n)e^{-j(2\pi \frac{f_0}{f_s} n+\phi)} \tag{63}$$

$$A(n) = \tfrac{\sqrt{6}}{6}\left(V_a(n) + V_b(n)e^{j\Delta_b} + V_c(n)e^{j\Delta_c}\right),\ B(n) = \tfrac{\sqrt{6}}{6}\left(V_a(n) + V_b(n)e^{-j(\Delta_b+2\pi/3)} + V_c(n)e^{j(\Delta_c-2\pi/3)}\right)$$

Substituting this into the expression for a widely linear AR model (equation 58), it follows that:

$$\begin{aligned}
&A(n+1)e^{j(2\pi \frac{f_0}{f_s}(n+1)+\phi)} + B(n+1)e^{-j(2\pi \frac{f_0}{f_s}(n+1)+\phi)} = \\
&\quad h^*(n)A(n)e^{j(2\pi \frac{f_0}{f_s} n+\phi)} + h^*(n)B(n)e^{-j(2\pi \frac{f_0}{f_s} n+\phi)} + g^*(n)A^*(n)e^{-j(2\pi \frac{f_0}{f_s} n+\phi)} + g^*(n)B^*(n)e^{j(2\pi \frac{f_0}{f_s} n+\phi)}
\end{aligned} \tag{64}$$

$$\begin{aligned}
&\Rightarrow A(n+1)e^{j(2\pi \frac{f_0}{f_s} n+\phi)}e^{j(2\pi \frac{f_0}{f_s})} + B(n+1)e^{-j(2\pi \frac{f_0}{f_s} n+\phi)}e^{-j(2\pi \frac{f_0}{f_s})} = \\
&\quad h^*(n)A(n)e^{j(2\pi \frac{f_0}{f_s} n+\phi)} + h^*(n)B(n)e^{-j(2\pi \frac{f_0}{f_s} n+\phi)} + g^*(n)A^*(n)e^{-j(2\pi \frac{f_0}{f_s} n+\phi)} + g^*(n)B^*(n)e^{j(2\pi \frac{f_0}{f_s} n+\phi)}
\end{aligned} \tag{65}$$

By separating and then equating terms with common exponentials ($e^{j(2\pi \frac{f_0}{f_s})}$ and $e^{-j(2\pi \frac{f_0}{f_s})}$), and then simplifying, it follows that:

$$A(n+1)e^{j2\pi \frac{f_0}{f_s}} = h^*(n)A(n) + g^*(n)B^*(n) \tag{66}$$

$$B(n+1)e^{-j2\pi \frac{f_0}{f_s}} = h^*(n)B(n) + g^*(n)A^*(n) \tag{67}$$

Assuming that the rate of change in peak voltage and phase distortion is negligible in an unbalanced system, it follows that $A(n+1) \approx A(n)$ and $B(n+1) \approx B(n)$ such that:

$$e^{j2\pi \frac{f_0}{f_s}} = h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} \tag{68}$$

$$e^{-j2\pi \frac{f_0}{f_s}} = h^*(n) + g^*(n)\frac{A^*(n)}{B(n)} \tag{69}$$

Taking the conjugate of equation 69 and equating it to equation 68:

$$e^{j2\pi\frac{f_0}{f_s}} = h^*(n) + g^*(n)\left(\frac{B^*(n)}{A(n)}\right) = h(n) + g(n)\left(\frac{A^*(n)}{B(n)}\right) \tag{70}$$

Letting $X(n) = \frac{B^*(n)}{A(n)}$:

$$g^*(n)X(n) + h^*(n) = h(n) + g(n)\left(\frac{1}{X(n)}\right) \tag{71}$$

$$g^*(n)X^2(n) + [h^*(n) - h(n)]X(n) - g(n) = 0 \tag{72}$$

Solving the quadratic equation:

$$X(n) = \frac{[h(n) - h^*(n)] \pm \sqrt{[h^*(n) - h(n)]^2 + 4g^*(n)g(n)}}{2g^*(n)} \tag{73}$$

For any complex number $h(n)$ it holds that $h(n) + h^*(n) = 2\Re\{h(n)\}$, $h(n) - h^*(n) = 2j\Im\{h(n)\}$, and $h^*(n)h(n) = |h(n)|^2$, using these results it follows that:

$$X(n) = \frac{-2j\Im\{h(n)\} \pm \sqrt{-4\Im\{h(n)\}^2 + 4|g(n)|^2}}{2g^*(n)} = j\frac{-\Im\{h(n)\} \pm \sqrt{\Im\{h(n)\}^2 - |g(n)|^2}}{g^*(n)} \tag{74}$$

Now substituting this expression for $X(n)$ back into equation 68, using the fact that for any complex number it also holds that $\Re\{h(n)\} = h^*(n) - j\Im\{h(n)\}$, and then finally expressing the resulting complex number in Euler form:

$$e^{j2\pi\frac{f_0}{f_s}} = h^*(n) + jg^*(n)\frac{-\Im\{h(n)\} \pm \sqrt{\Im^2\{h(n)\} - |g(n)|^2}}{g^*(n)} \tag{75}$$

$$= h^*(n) - j\Im\{h(n)\} \pm j\sqrt{\Im^2\{h(n)\} - |g(n)|^2} \tag{76}$$

$$= \Re\{h(n)\} \pm j\sqrt{\Im^2\{h(n)\} - |g(n)|^2} = Ke^{j\arctan\left(\frac{\pm j\sqrt{\Im^2\{h(n)\} - |g(n)|^2}}{\Re\{h(n)\}}\right)} \tag{77}$$

Where K is a real constant, and must be equal to 1 for the equality to hold. Since the phase sign is only indicative of rotation direction, and thus is unimportant to the resulting value of f_0, the positive value is taken for ease of notation. Finally, for the equality to hold, the phases of both expression must be equal, such that:

$$\frac{2\pi f_0(n)}{f_s} = \arctan\left(\frac{\sqrt{\Im^2\{h(n)\} - |g(n)|^2}}{\Re\{h(n)\}}\right) \Rightarrow f_0(n) = \frac{f_s}{2\pi}\arctan\left(\frac{\sqrt{\Im^2\{h(n)\} - |g(n)|^2}}{\Re\{h(n)\}}\right) \tag{78}$$

Giving us the final solution for the frequency of an unbalanced complex Clarke voltage, as required.

3.1.5 Part e)

Following from the analysis in the previous section, acknowledging the fact that the CLMS (order 1) is the adaptive filter associated with the Strictly linear model defined in equation 57 and the ACLMS is the adaptive filter associated with the widely linear model defined in 58, the two filters are used to first find estimations of $h^*(n)$ and $g^*(n)$ and then substitute the corresponding values into equations 62 and 78 to find their corresponding estimates for the fundamental frequency in the three-phase system $f_0 = 50$Hz. The two adaptive filters were first used to estimate the fundamental frequency in a balanced system, and then in an unbalanced system where V_a is half of V_b and V_c (no phase distortion), and then in a different unbalanced system where peak voltages are equal, but a phase distortion $\Delta_b = \pi/3$ is present. Since $\Re\{h(n)\}$, appears in the denominator in both equation 62 and 78, $h(0)$ was thus set to 1 for each filter to prevent instability. The step size that allowed for sufficiently fast convergence as well as stability was empirically found to be $\mu = 0.0001$. The results from these experiments are plotted and shown in Figure 30.

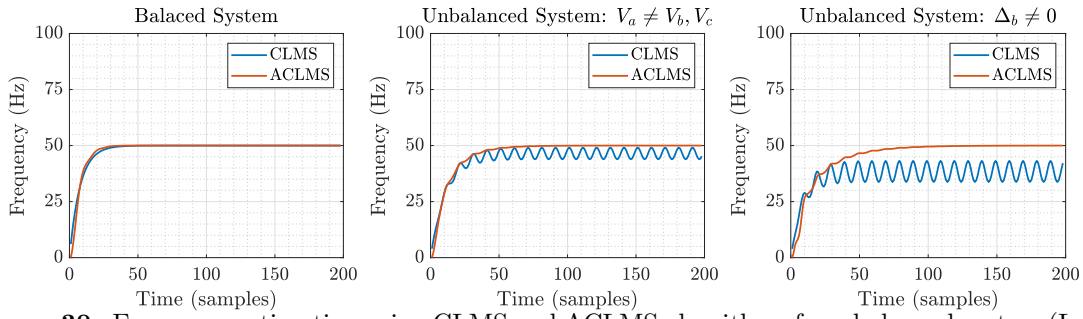


Figure 30: Frequency estimation using CLMS and ACLMS algorithms for a balanced system (Left), an unbalanced system in terms of peak voltages (**Middle**) and an unbalanced system in terms of phase distortion (**Right**)

Given the fact that only one parameter (degree of freedom) is required to model the nominal frequency f_0 for a balanced system, shown in the analysis of the previous section, both filters converge to the true value since they possess the sufficient degrees of freedom. For the unstable system, only the ACLMS has the sufficient degrees of freedom (2) to model the nominal frequency (see equation 78). For this reason, the ACLMS successfully converges to the true nominal frequency for both imbalanced systems, whereas the CLMS converges to an incorrect frequency value and also exhibits oscillation about this steady-state value, where the oscillation present in the ACLMS's estimate is quickly dampened.

3.2 Adaptive AR Model Based Time-Frequency Estimation

3.2.1 Part a)

A frequency modulated (FM) signal $y(n) = e^{j\left(\frac{2\pi}{fs}\phi(n)\right)+\eta(n)}$ is now considered, 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 defined in equation 79.

$$f(n) = \frac{d\phi(n)}{dn} = \begin{cases} 100, & 1 \leq n \leq 500 \\ 100 + \frac{n-500}{2}, & 501 \leq n \leq 100 \\ 100 + \left(\frac{n-1000}{25}\right)^2, & 1001 \leq n \leq 1500 \end{cases} \tag{79}$$

The frequency of the resulting signal at each time instance is plotted on the left in Figure 31. Its resulting frequency response is thus expected to have a spike at 100 Hz, to be (relatively) flat between 100-350Hz, (relatively) flat between 350-500Hz, and 0 otherwise. The `aryule` MATLAB function was used to find the AR(1) coefficient for the signal, as well as its corresponding power spectrum, which is plotted on the right in Figure 31 along with the spectra corresponding to higher model orders. The AR(1) is not sufficiently parametrised in order to capture the frequency response of the FM signal, instead, its peak is centred around the average frequency content of the signal, whereas higher model orders are able to approximately capture the shape of the true signal's frequency response. Furthermore, since `aryule` assumes stationarity, it is only able to capture the overall frequency response of the signal and not its time-varying nature.

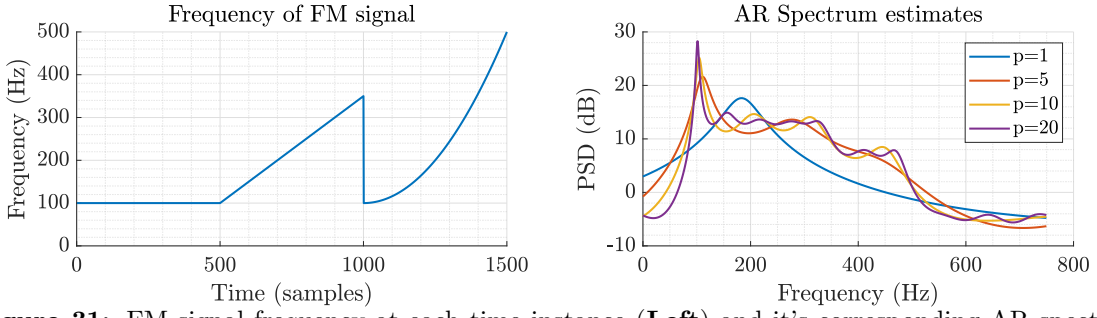


Figure 31: FM signal frequency at each time instance (**Left**) and it's corresponding AR spectral estimates (**Right**)

3.2.2 Part b)

The CLMS algorithm is now used to adaptively estimate the AR coeffient of $y(n)$ at each time instance, in this way, as the frequency content of the signal is changing, so will the corresponding AR coefficient, and so CLMS can change with it. Furthermore, the signal is approximately circular ($|\rho| = 0.018$) and thus the CLMS has the sufficient degrees of freedom to model the signal. This experiment is carried out for different step sizes μ, with the resulting spectrograms of the predicted signals being shown in Figure 32. When the step size is too small, the CLMS is unable to adapt its parameter values quick enough to match the varying frequency content. For $\mu = 0.1$ the filter is able to capture the change in frequency well, however there is increased uncertainty in its estimate at each time point since large steps in learning increase the variance of the estimate.

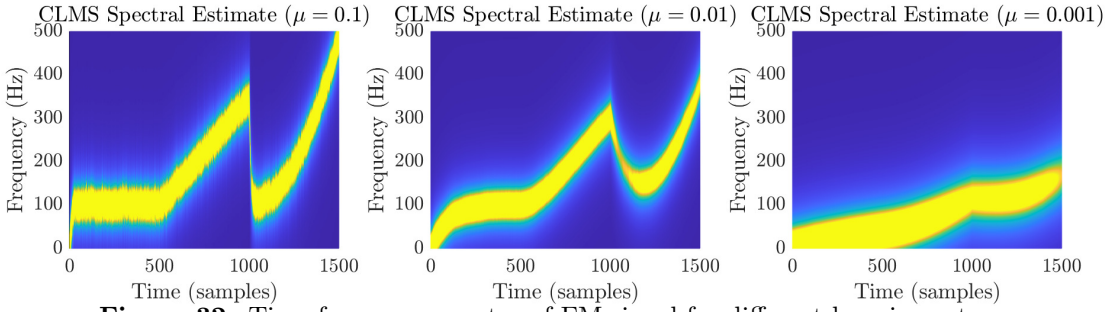


Figure 32: Time-frequency spectra of FM signal for different learning rates

3.3 A Real-Time Spectrum Analyser Using Least Mean Square

Part a)

A signal $y(n)$ can be estimated as a linear combination of N harmonically related sinusoids, as described in equation 3.3:

$$\hat{y}(n) = \sum_{k=0}^{N-1} w(k) e^{j\frac{2\pi kn}{N}} \tag{80}$$

Or equivalently, it can be expressed in vector form as seen in equation 81:

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}(0) \\ \hat{y}(1) \\ \vdots \\ \hat{y}(N-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & e^{\frac{j2\pi(1)(1)}{N}} & \dots & e^{\frac{j2\pi(1)(N-1)}{N}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e^{\frac{j2\pi(N-1)(1)}{N}} & \dots & e^{\frac{j2\pi(N-1)(N-1)}{N}} \end{bmatrix} \begin{bmatrix} w(0) \\ w(1) \\ \vdots \\ w(N-1) \end{bmatrix} = \mathbf{Fw} \tag{81}$$

The optimal weights $\mathbf{w_{opt}}$, in the least-squares sense, can be found by minimising the L_2-norm of the prediction error such that

$$\min_{\mathbf{w}} \|\mathbf{y} - \hat{\mathbf{y}}\|^2 = \min_{\mathbf{W}} \sum_{n=1}^{N-1} \|y(n) - \hat{y}(n)\|^2 \tag{82}$$

$$= \min_{\mathbf{w}} (\mathbf{y} - \mathbf{Fw})^H (\mathbf{y} - \mathbf{Fw}) \tag{83}$$

$$= \min_{\mathbf{w}} (\mathbf{y}^H\mathbf{y} - \mathbf{w}^H\mathbf{F}^H\mathbf{y} - \mathbf{y}^H\mathbf{Fw} + \mathbf{w}^H\mathbf{F}^H\mathbf{Fw}) \tag{84}$$

$$= \min_{\mathbf{w}} (\mathbf{y}^H\mathbf{y} - 2\mathbf{w}^H\mathbf{F}^H\mathbf{y} + \mathbf{w}^H\mathbf{F}^H\mathbf{Fw}) \tag{85}$$

Given that $\|\mathbf{y} - \hat{\mathbf{y}}\|^2$ is a convex function, the minimum (with respect to the unknown weights) can be found by differentiating the function with respect to $\mathbf{w}$ and setting it equal to 0, such that:

$$\nabla_{\mathbf{w}} \|\mathbf{y} - \hat{\mathbf{y}}\|^2 = -2\mathbf{F}^H\mathbf{y} + 2\mathbf{F}^H\mathbf{Fw} = 0 \tag{86}$$

Solving for $\mathbf{w}$, the final result for the set of optimal weights is found

$$\mathbf{w}_{opt} = (\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H\mathbf{y} \tag{87}$$

The matrix $\mathbf{F}$ is full-rank, since all its column vectors are orthogonal, and as a result, $\mathbf{F}$ is invertible. Furthermore, $\mathbf{F}^H\mathbf{F} = N\mathbf{I}$, therefore the solution for $\mathbf{w}_{opt}$ can be rewritten as:

$$\mathbf{w}_{opt} = \frac{1}{N}\mathbf{F}^H\mathbf{y} \tag{88}$$

Looking at the definition for the Inverse Discrete Fourier Transform (IDFT) in equation 89, and comparing it to equation 3.3, it can clearly be seen that the coefficients $\mathbf{w}$ simply correspond to the scaled estimated frequency coefficients/components of the signal $\mathbf{y}$, and therefore the optimal weights are found by taking the scaled DFT of the signal $\mathbf{y}$

$$\textbf{IDFT} \ : \qquad \hat{x}(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j\frac{2\pi}{N}nk} \tag{89}$$

3.3.1 Part b)

The range space of the matrix $\mathbf{F}$ is defined by the span of the basis vectors that form its columns, as seen in equation 81. This means that the estimated signal $\hat{\mathbf{y}}$, must lie in the range space of $\mathbf{F}$. Using the least squares solution to the weights from equation 87, and substituting it into equation 80, the estimated signal and the true signal can be related as follows:

$$\hat{\mathbf{y}} = \mathbf{F}(\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H\mathbf{y} \tag{90}$$

Meaning that the estimated signal is the projection of the true signal on to the space spanned by the columns of $\mathbf{F}$ that minimises the euclidean distance between the estimated and true signal. Finally, knowing that equation 88 is a scaled DFT transform, it can be said that the DFT is simply the linear transformation of the true signal $\mathbf{y}$ onto the row space of matrix $\mathbf{F}$ whose basis vectors compose of a series of rotations in the complex space.

3.3.2 Part c)

The CLMS algorithm is adapted to employ the analysis of the previous two sections to predict the time-frequency response of the same FM signal as discussed in Section 3.2.1. Considering equation 80, the input signal can be considered as the set of complex exponentials that transform frequency components $\{w(1), w(2), ..., w(N-1)\}$ into the time-domain, such that the input $x(n)$ can be written as

$$\mathbf{x}(n) = \frac{1}{N}\left[1, \quad e^{j\frac{2n\pi}{N}}, \quad e^{j\frac{4n\pi}{N}} \quad \cdots, \quad e^{j\frac{2n(N-1)\pi}{N}}\right]^T \tag{91}$$

and the frequency components are learnt ($\{w(1), w(2), ..., w(N-1)\}$) as the model weights of the adapted CLMS algorithm, which will be denoted as the DFT-CLMS algorithm. As aforementioned, the DFT-CLMS was employed in the time-frequency prediction of the FM signal of Section 3.2.1, and the parameters used in this experiment were $f_s = 1000$Hz, and learning rate $\mu = 1$.

As seen on the far left plot in Figure 33, the standard DFT-CLMS is unable to adapt its parameters fast enough in order to match the time-varying frequency response of the FM signal. To combat this, a leak parameter γ was introduced such that the model can 'forget' frequency information, such that it no longer dominates the next time step's output, and plays a synonymous role to the leakage coefficient in the leaky LMS algorithm. Considering this, the update equation for the DFT-CLMS now becomes:

$$\mathbf{w}(n+1) = (1-\mu\gamma)\mathbf{w}(n) + \mu e^*(n)\mathbf{x}(n) \tag{92}$$

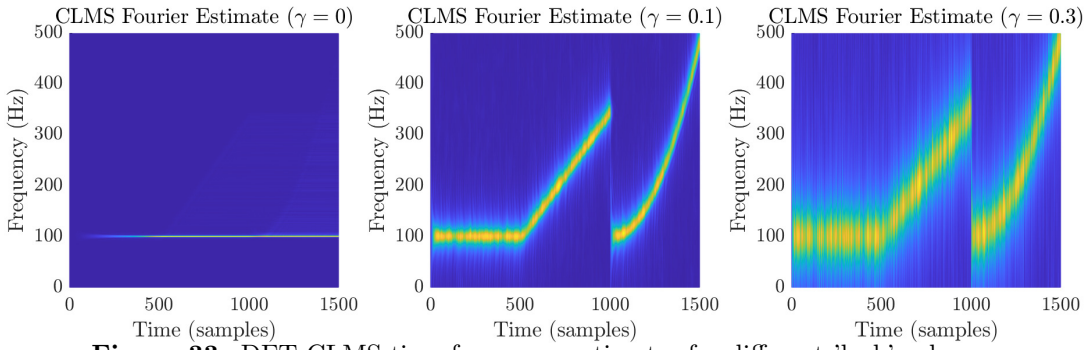


Figure 33: DFT-CLMS time-frequency estimates for different 'leak' values γ

The experiment was then repeated for different 'leak' values, with the results also being shown in Figure 33. It can be seen that out of the three different γ values, $\gamma = 0.1$ provides the best balance between the variance of the estimate at each time point and the responsiveness to the changing frequency values. A higher leak values allows for previous time-step frequency information to be forgotten more quickly, however, as seen with $\gamma = 0.3$, this comes at the cost of a significantly increased variance in the frequency estimate at each time point.

3.3.3 Part d)

The DFT-CLMS algorithm is now employed to estimate the frequency content of the EEG signal, POz, as seen in section 1.2.2. To control the computation complexity of the process, the data was trimmed to only include sample 3000 to 4200. Given the fact that the POz signal's frequency content is stationary, the standard DFT-CLMS could be used. The resulting time-frequency estimate was calculated and shown in Figure 34. It takes roughly 800 samples for the dominant frequency components to become discernible. After this point, the low frequency 3Hz component, and 50Hz mains component become clearly identifiable with very little noise outside these components, excluding the harmonics of the 3Hz component which are faintly visible. This improvement in dominant frequency identification comes at the cost of a greater computational complexity (compared to periodogram techniques) and requirement of a substantial number of observations to learn the optimal parameters.

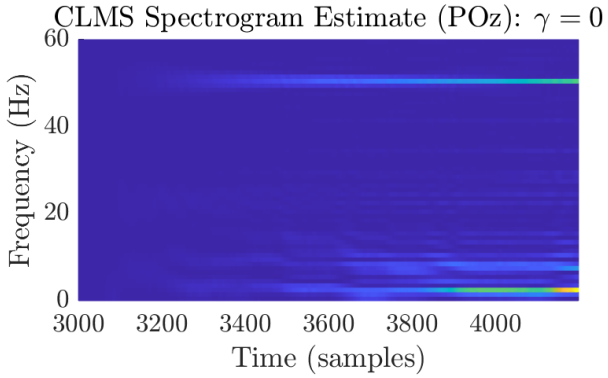


Figure 34: DFT-CLMS time-frequency estimate for samples 3000-4200 of the POz signal.

4 From LMS to Deep Learning

4.1 LMS Time-Series Prediction

To begin with, the same LMS algorithm from Section 2.1 was used to perform one-step ahead prediction on zero-meaned non-stationary time-series data provided to us (`time-series.mat`). A model order of 4 and a step size of $\mu = 10^{-5}$ was used, the result of which are shown in Figure 35. It can be seen that the algorithm converges after around 300 samples. Over the final 200 samples, the algorithm

exhibits an MSE of 17.0, and a prediction gain $R_p = 10\log_{10}\left(\hat{\sigma_y^2}/\sigma_e^2\right) = 9.44$dB. The performance of the model over the final 200 samples will be used throughout the following experiments such that transient effects of a particular model are accounted for and negated. Over the 1000 samples as a whole, MSE=40.1, and $R_p = 5.2$dB.

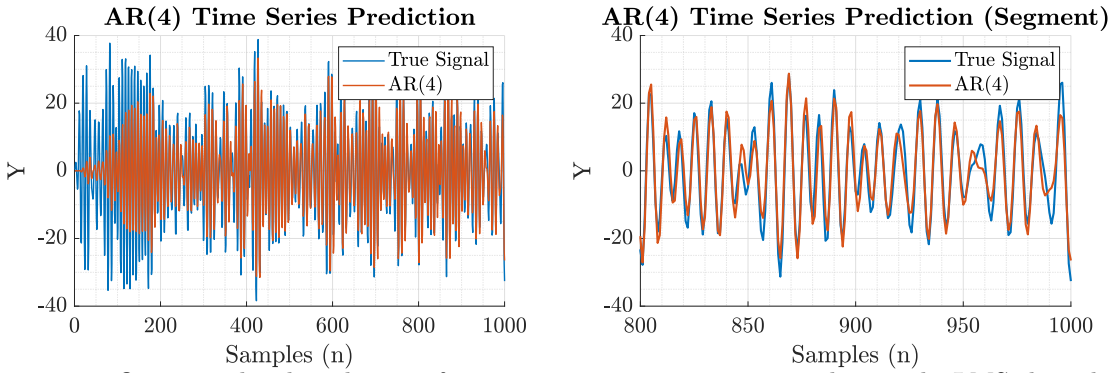


Figure 35: One-step ahead prediction of a zero-mean non-stationary signal using the LMS algorithm of order 4 and learning rate $\mu = 10^{-5}$

4.2 Dynamical Perceptron

In order to capture the non-linearity of the time-series signal, a non-linear tanh activation function is introduced at the output of the perceptron ($\mathbf{w}(n)^T\mathbf{x}(n)$), such that $y(n) = \tanh(\mathbf{w}(n)^T\mathbf{x}(n))$. Since the derivative of $\tanh(x)$ is $1 - \tanh^2(x)$, the update equation for the LMS weights now becomes:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)[1 - \tanh^2(\mathbf{w}(n)^T\mathbf{x}(n))]\mathbf{x}(n) \tag{93}$$

The LMS with the tanh activation was now performed on the same one-step ahead prediction task on the time-series data, the results being shown in Figure 36. It is clear that this model is performing very poorly in this task, it displays some ability in modelling non-linearities, however, since $\tanh(x) \in [-1; 1]$, the resulting prediction is also bounded between those values, and as a result, it is unable to match the real-signal values that are far greater in magnitude. Because of this limitation to the algorithm, an MSE of 205.98 and an R_p of -29.8dB is exhibited over the 1000 samples.

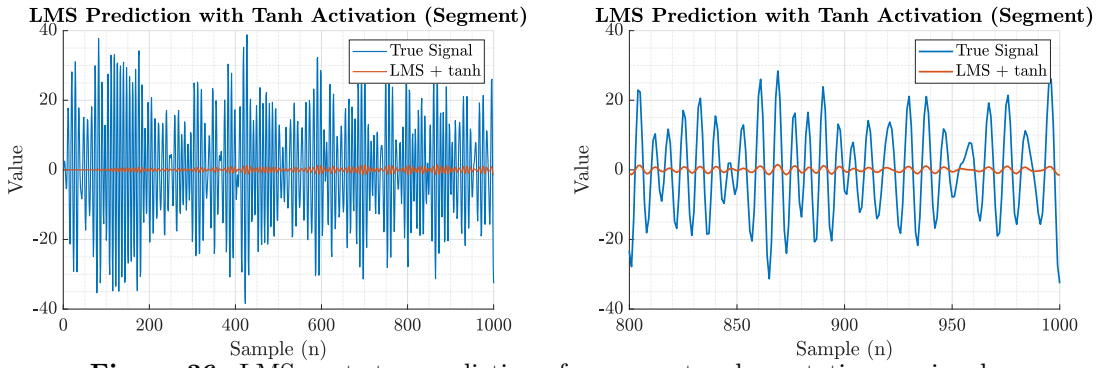


Figure 36: LMS next-step prediction of mean-centered nonstationary signal

4.3 Scaled Activation Function

Considering the limitations of introducing the tanh activation function in the previous section, a scaling factor a is introduced after tanh activation such that the fact that $\tanh(x) \in [-1; 1]$ no longer clips the model output. The update equation now becomes:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu a e(n)[1 - \tanh^2(a\mathbf{w}(n)^T\mathbf{x}(n))]\mathbf{x}(n) \tag{94}$$

The scaling factor, if it is not to clip the model output at any point, must be greater than the maximum amplitude (both negative and positive) present in the signal, which for the case of the given time series

signal, $a \geq 38.8$. The modified LMS with update equation as defined above was implemented for a fixed step size $\mu = 1 \times 10^{-7}$ and changing scaling factor values. The MSE of the final 200 samples (to account for transient effects of filter), was determined for $38 \leq a \leq 100$, the result of which, alongside the predicted output from the resulting optimal a value, are shown in Figure 38. This is

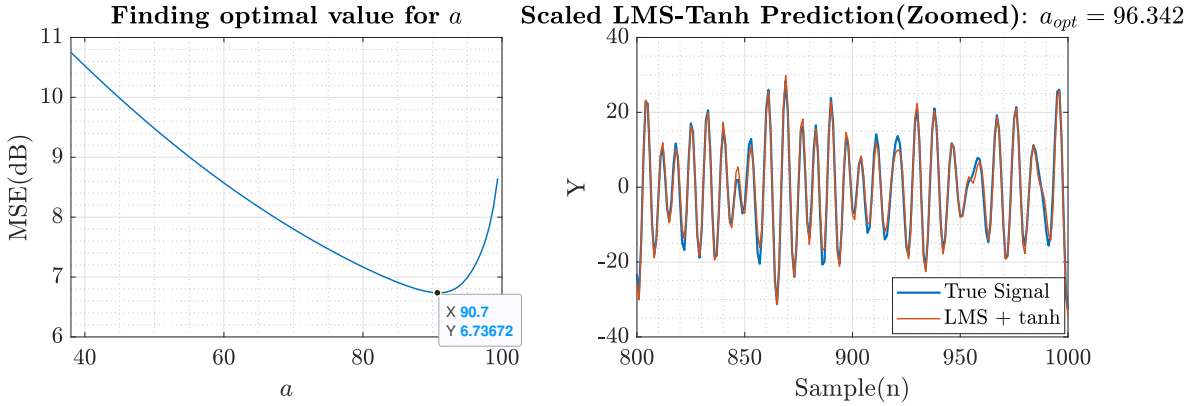


Figure 37: MSE of final 200 samples for a fixed μ and changing

clearly performing much better than the results from the previous section, in fact, for the optimal value of a, the modified LMS algorithm, over the final 200 sample has MSE $= 2.89$, and $R_p = 17.70$dB, and is now performing much better than the standard LMS. To find the optimal value for a however, 1000 repititions of the algorithm had to be performed, this is a very computationally complex task, therefore instead, the scaling factor is adaptively updated using gradient descent, meaning that the update equation now becomes:

$$a(n+1) = a(n) + \mu_a e(n) \tanh(\mathbf{w}(n)^T \mathbf{x}(n)) \tag{95}$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + a(n)\mu_w e(n)[1 - \tanh^2(\mathbf{w}(n)^T \mathbf{x}(n))]\mathbf{x}(n) \tag{96}$$

The parameters μ_a and μ_w were empirically chosen to be 3 and 1×10^{-7}, and the starting value for a, a_0 was chosen to be 80. The resulting prediction from this adaptive scaling is shown alongside the true signal in Figure 38. Unsurprisingly, the value of a seems to converging around its corresponding optimal value. Convergence is just as fast as with fixed scaling, both of which are converging much faster than the standard LMS algorithm (~ 50 samples). Although the computational complexity in a single run-through of adaptive scaling is now higher compared to fixed scaling, the fact that a must be selected from N repetitions of the algorithm means that the overall complexity of the adaptive scaling process is much smaller than if it were to be fixed. In terms of performance metrics, MSE $= 2.80$ and $R_p = 17.84$dB for the final 200 samples, which is a slight increase in performance compared to fixed scaling. Adaptive scaling is thus preferred and will be used henceforth.

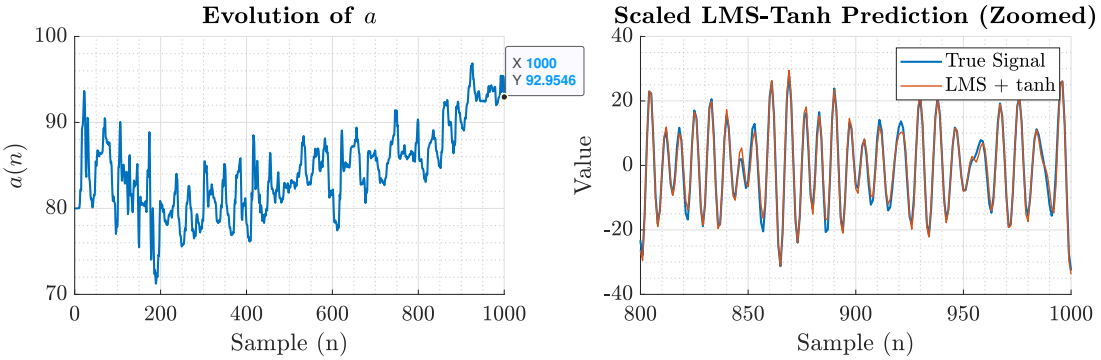


Figure 38: LMS next-step prediction of mean-centered non-stationary signal with an adaptive, scaled, tanh activation function.

4.4 Dynamical Perceptron & Bias

Following on from the analysis in the previous section, the LMS-tanh algorithm is aimed to be improved by introducing a bias parameter before tanh activation such that the output of the dynamic perceptron

now becomes $\phi(\mathbf{w}^T\mathbf{x} + b)$, where $\phi(\cdot) = \tanh(\cdot)$. The additional parameter provides an additional degree of freedom that allows for the model to learn any DC bias that is present in the signal. To implement the bias, the augmented input $\hat{\mathbf{x}} = [1, \mathbf{x}^T]^T$ is considered, the model thus learns an additional parameter w_0, where $\mathbf{w} = [w_0, w_1, w_2, w_3, w_4]^T$, which is equal to the bias term, $w_o = b$. The adaptive scaling algorithm from the previous section was adapted to include the bias, and the same parameters were used as before, $\mu_a = 3$, $\mu_w = 1 \times 10^{-7}$, $a_{start} = 80$. The adapted algorithm was implemented in the same one step-ahead prediction problem setup, but now the true signal is no longer zero-mean, the results being shown in Figure 39.

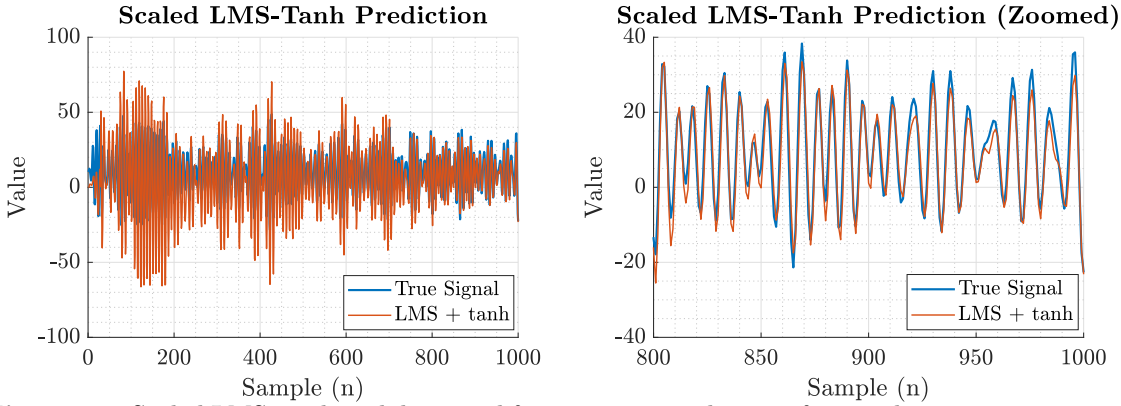


Figure 39: Scaled LMS-Tanh with bias used for next-step prediction of original nonstationary signal

For the final 200 samples, MSE= 3.87 and $R_p = 16.54$ dB, which is a slight decrease in performance to the results in the previous section (no bias), however, it must be noted that the model now has to learn the additional bias present in the signal, which is an additional source of error. Looking at the evolution of the parameter weights in Figure 40, it can be seen that the algorithm is converging after around 500 samples, which is much longer than for the zero-mean signal, this is because the error of the prediction is dependent on the bias, which is essentially learning the mean of the signal, and this takes 500 samples to converge. As a result, over the full 1000 samples, the model performs poorly, MSE= 26.24 and $R_p = 9.43$ dB.

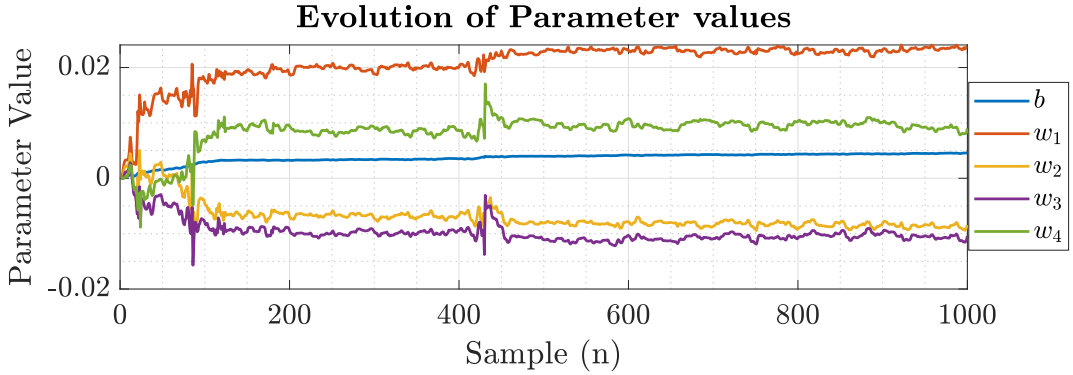


Figure 40: Evolution of parameter weigths of Scaled LMS-Tanh with bias.

4.5 Pre-training Weights

In the previous section, to improve the robustness of the model that is being developed, to account for what may not be a constant DC value, the full original signal was considered, and the bias term was introduced such that the DC value of the signal could be adaptively learnt. The introduction of this parameter, and the consideration of the non-zero mean signal, meant that the initial conditions of the filter weights, which are all set to 0, were no longer suitable, and for this reason the dynamical perceptron with bias took around 500 samples to converge. Now, to account for this, and to produce more suitable initial conditions for the filter weights, the dynamical perceptron is trained on (overfitted to) the first 20 samples of the data, on the same prediction task, the process of which is repeated for 100 epochs. Once the initial conditions are determined from pretraining, the same model as the

previous section is run on the data, to produce a one step-ahead prediction, the result of which along with the evolution of filter weights (post-pretraining) is shown in Figure 41.

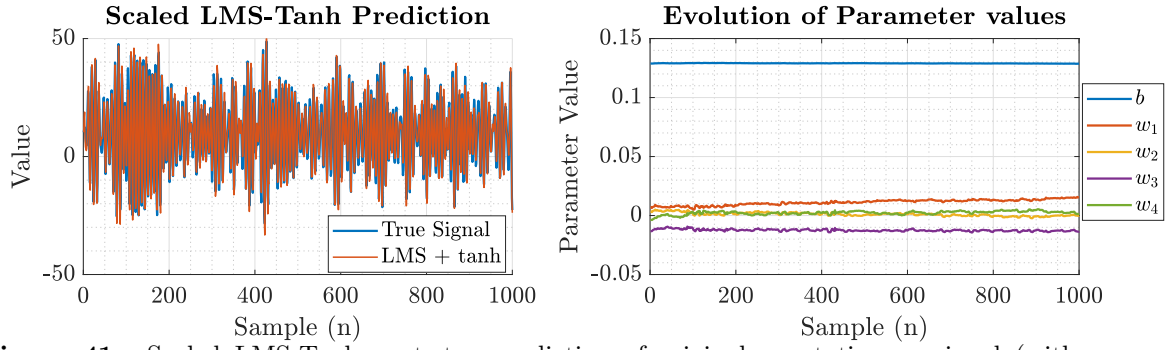
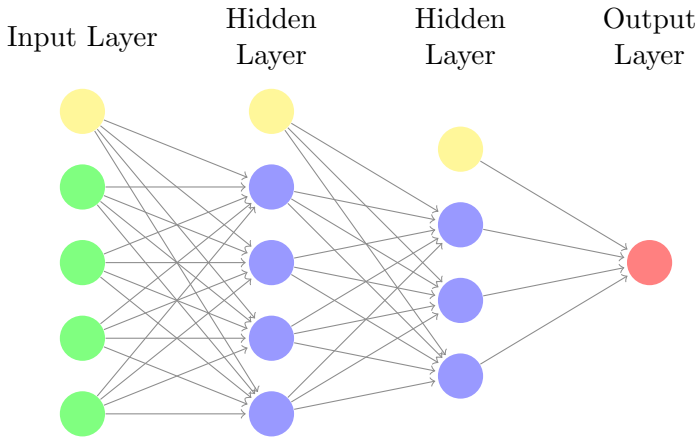


Figure 41: Scaled LMS-Tanh next-step prediction of original nonstationary signal (with pre-training).

The model now converges almost instantaneously, there is very little change in filter weights throughout the 1000 samples, and it's worth noting that these values are very different to when the filter weights are initialised to 0. Over the 1000 sample period, MSE= 6.82 and $R_p = 14.96$ dB, which is a big improvement over the results from the previous section, however, over the final 200 samples, MSE= 5.93 and $R_p = 14.52$ dB, which is in fact worse than the results of the previous section over those final 200 samples. Since the weights are being pretrained on the first 20 samples over 100 times, the model weights are going to be heavily overfit to that data, consequently when the model begins to adapt to the rest of the data, it may be possible that the model is 'stuck' in local minima, and hence generalises worse than the model which has no skew toward a subset of the data (once that model has converged).

4.6 The Backpropagation Algorithm

Thus far, the dynamical perceptron has been considered is essentially a single layer of a Neural Network with 4 neurons. A Deep Neural Network (DNN) essentially densely connects multiple layers of these together, as demonstrated in the diagram below. The input data is passed through the first layer, where they are multiplied by neuron weights (denoted as filter weight in LMS), a bias is added and then passed through a non-linear activation function (tanh has been used thus far), the outputs, known as **activations**, serve as the inputs to the next layer, and this is progressively repeated until the output layer where the prediction is made, which in whole, forms a **forward pass**.



The prediction at the output of the neural network is then evaluated via a cost function J, which thus far has been the MSE, but can be defined differently according to the problem. In order to update the weights $\mathbf{W}$ and biases $\mathbf{b}$ of the network, the **backpropagation** algorithm is used. The output of each node of the network is dependent on the previous layers, and this chain of dependency can be used to compute the gradients of the error with respect to the weights and biases via the chain rule, this is then used to perform gradient descent on each of the parameters(as done with the dynamic

perceptron). Since the cost function is only a function of the activation function at the output node, first the gradient of that node is calculated, and as mentioned before, the chain rule is used to calculate the gradients of the nodes of the layer prior, the process of which is carried out recursively until the input layer is reached. This process of a forward and backward pass is repeated for all the data, which is referred to as one **epoch**, the model can then be trained for N epochs until a presumed global optimum for the cost function is reached.

4.7 Deep Neural Network

A non-linear time-series is now generated, composing of 10 sinusoids of different frequencies, amplitudes, and phases as shown in the top graph of Figure 42. The resulting signal is then passed through an unknown activation function, and then noise of power 0.05 is added such that $y(n) = \phi\{x(n)\} + \sqrt{0.05}\sigma_n$, where σ_n is unit variance white noise. The resulting time-series signal is shown on the bottom graph in Figure 42.

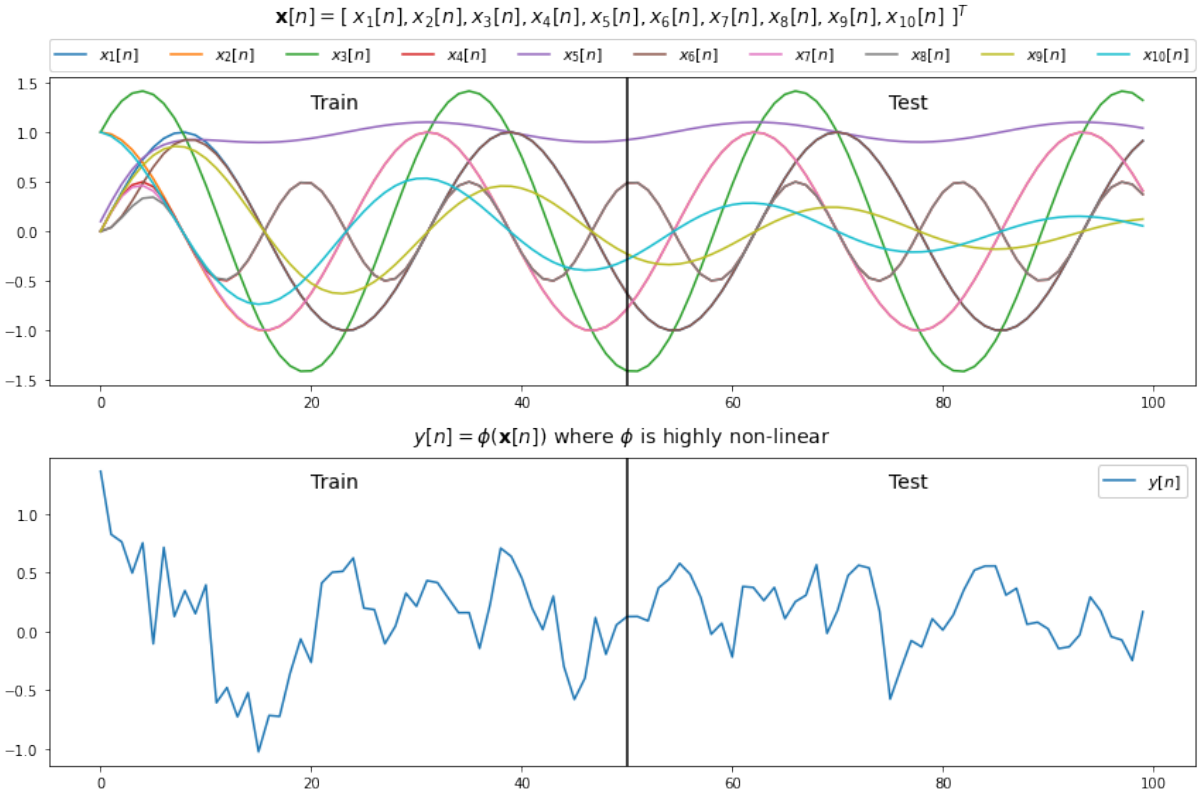


Figure 42: Sinusoids summed and passed through non-linear activation function to obtain non-linear time-series

The time-series signal was then divided into a 50% train/test split, and a linear LMS (Model 1), dynamical perceptron with tanh activation (Model 2), and a DNN of 4 hidden layers and ReLU (equation 97) activation (Model 3), were used to train on the first half of the data for 20,000 epochs and with a step size of $\mu = 0.1$. The trained model was then evaluated on the test set, with the error being measured with the MSE. The resulting predicted signals from each of the models is shown alongside the true signal for both the training, and test sets in Figure 43. The MSE loss curves for each of the models is also shown in Figure 44.

$$ReLU(x) = max\{0, x\} \tag{97}$$

All three models exhibit a lower training loss as the number of epochs increases, this is expected since each of the models are directly updating their parameters such that the error is minimised with respect to the training data. As expected, the DNN exhibits the lowest training loss over the 20,000 epochs since it has many more parameters and thus degrees of freedom, and thus is able to overfit to the noise better. For the same reason, after 20,000 epochs, the DNN exhibits the highest test loss out of the three different models, this is because it has the sufficient degrees of freedom to overfit to the noise in the training data, and thus its learned parameters are not reflective of the underlying function. On

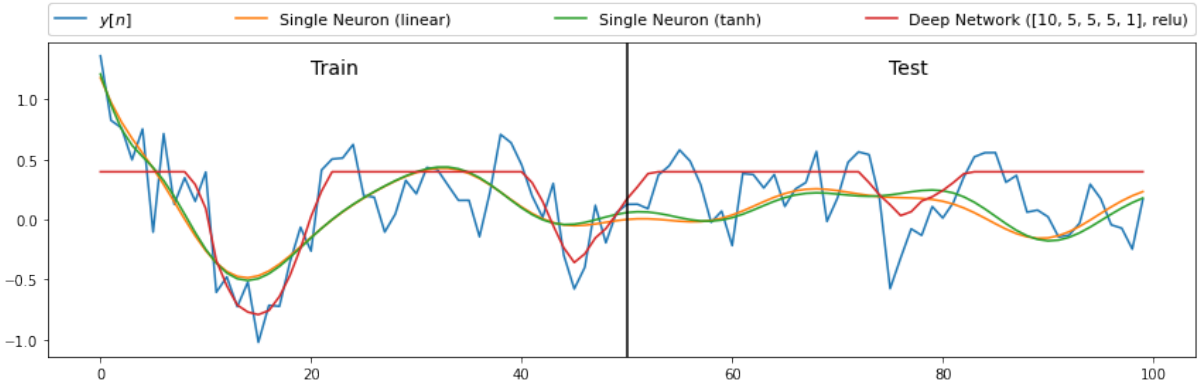


Figure 43: LMS, dynamical perceptron and deep network predictions of low noise, non-linear time-series

the other hand, when selecting the optimal model out of all the epochs, the DNN generalises to the lowest testing MSE out of the 3 models and after far fewer epochs, this is again reflective of the extra degrees of freedom in the model which allows it to learn the underlying function far quicker than the other two models. The model should stop training after this point, as after this, is where it begins overfitting to the training data.

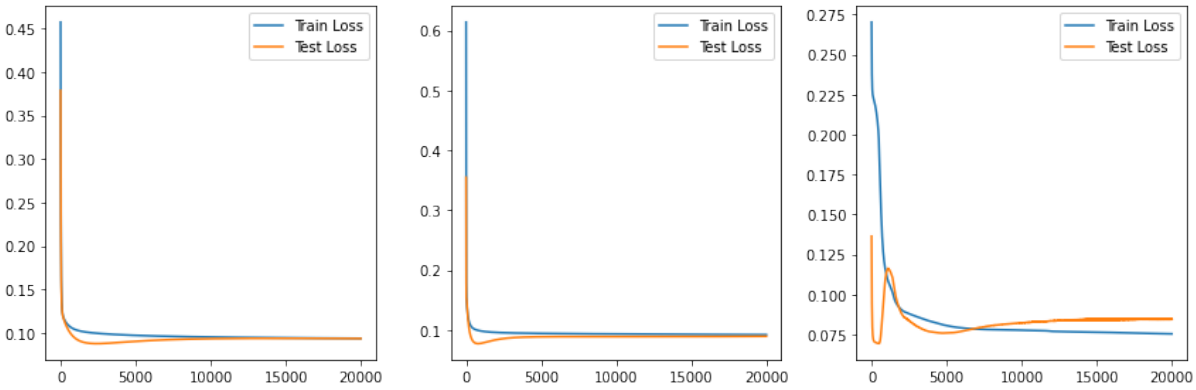


Figure 44: Training and testing loss curves for the standard LMS (**Left**), dynamical perceptron with tanh activation (**Middle**) and DNN with 4 hidden layers and ReLU activation (**Right**)

Model	Min Training MSE	Min Testing MSE	Min Testing Index
1	0.096	0.091	2412
2	0.091	0.079	817
3	0.071	0.067	487

Table 3: Comparison of theoretical and empirical LMS misadjustments.

4.8 DNNs & Noise

The same three models are trained and tested in the same problem setup as the previous section but for two new values of noise power, 0.0001 and 1, such that the output is given by $y(n) = \phi\{x(n)\} + \sqrt{0.0001}\sigma_n$ and $y(n) = \phi\{x(n)\} + \sigma_n$ respectively, where σ_n is unit variance white noise. Their corresponding loss curves are shown in Figures 45 and 46. For extremely low noise power, the signal-to-noise ratio is so high that the true signal is largely unaffected by the noise, and for this reason, the DNNs higher degrees of freedom allows for it to more intricately capture the non-linearities of the signal, and for this reason, it exhibits the lowest training and test MSE. The test MSE for all the models keeps decreasing as epochs are increasing, showing no signs of overfitting, this is of course because there is almost no noise to overfit to, and training will only ever allow for the model to better understand the underlying function.

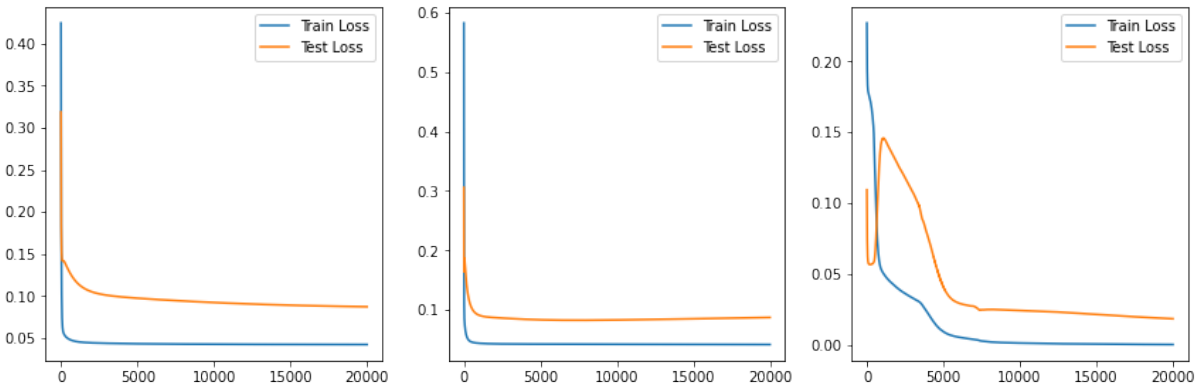


Figure 45: Training and testing loss curves for the standard LMS (**Left**), dynamical perceptron with tanh activation (**Middle**) and DNN with 4 hidden layers and ReLU activation (**Right**) for 0 noise power.

As for a much higher noise power, the models all unsurprisingly see a significant decrease in performance. The Linear and Tanh LMS models reach their optimal test error and begin overfitting to the training data very quickly, highlighting their sensitivity to signals with low signal-to-noise ratios. The DNN takes longer to overfit, perhaps because it has many more parameters, and thus it takes longer to learn as an ensemble with such volatile data. However, overall in terms of lowest MSE, it does not provide a significant increase in performance in comparison with the other two models, which in comparison with its performance with low noise power, suggests the model's dependency on training data quality. To combat its large degrees of freedom from overfitting to the data, some techniques can be used, e.g. the introduction of regularisation could help limit the model complexity, dropout introduces more randomness into the model learning process, and cross-validation helps keep track of the model's ability to generalise (model can stop training if it begins overfitting).

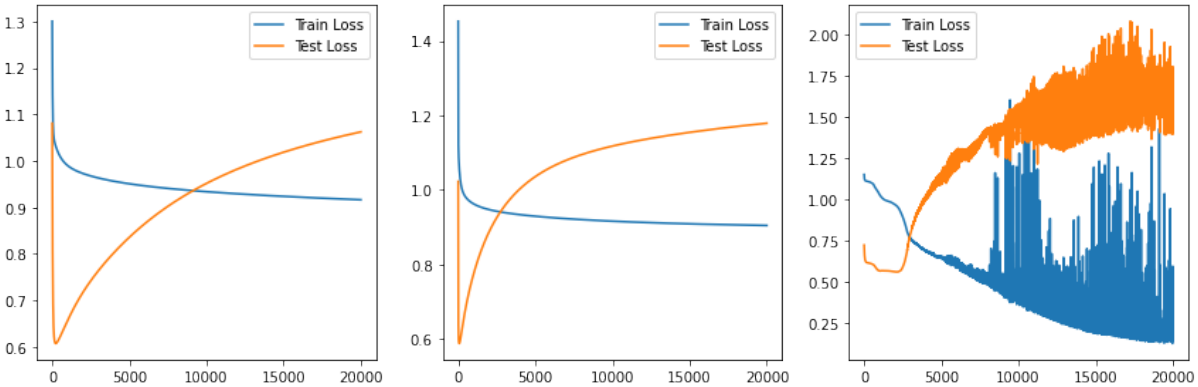


Figure 46: Training and testing loss curves for the standard LMS (**Left**), dynamical perceptron with tanh activation (**Middle**) and DNN with 4 hidden layers and ReLU activation (**Right**) for a noise power of 1.