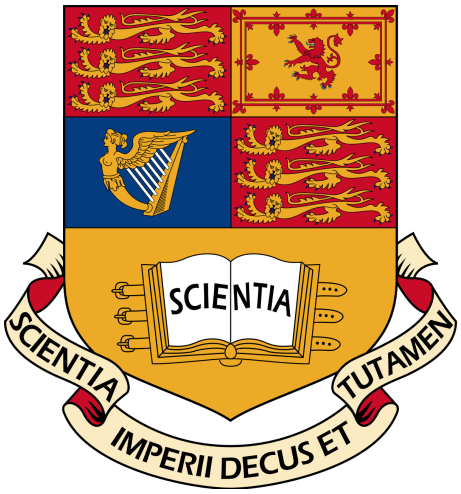
Imperial College London

Department of Electrical and Electronic Engineering



Adaptive Signal Processing and Machine Intelligence Coursework

Author:
Baptiste PROVENDIER

CID:
01553706

Lecturer:
Prof. Danilo MANDIC

April 2022

Contents

1 Classical and Modern Spectrum Estimation

1.1 Properties of Power Spectral Density (PSD)

In this section, we will investigate how the definitions of a PSD in equations (1) and (2) are equivalent under the assumption that the covariance sequence $r(k)$ decays rapidly, shown in equation (3). We will prove it analytically and see simulations examples to support that claim.

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

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

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

Starting from equation (2), we derive the PSD as follow:

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

We know from equation (3), that for a rapidly decaying covariance sequence, the right part of the last equation goes to 0, and we are left with the DTFT of the ACF:

$$P(\omega) = \sum_{k=-\infty}^{\infty} r(k)e^{-j\omega k} \tag{5}$$

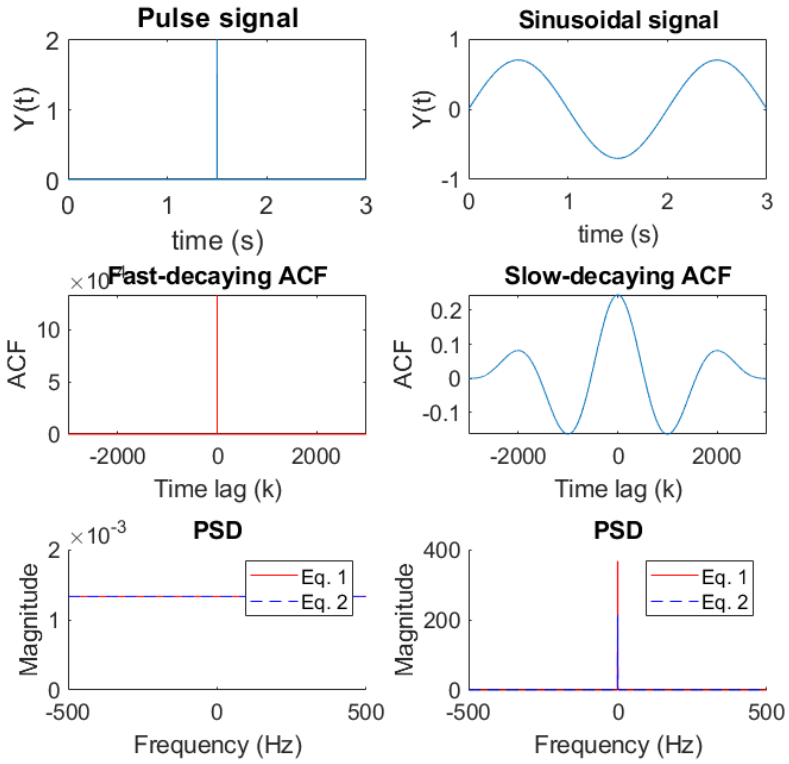


Figure 1: Comparison of an impulse signal and a sinusoidal wave, their ACFs and PSD estimates.

From Figure 1, we see that in the case of an impulse signal, which has rapid decay, the PSD estimates of the signal are the same when we use equation (1) and equation (2). This is not the case when we take the sinusoidal signal and plot the PSD estimates using the same equations. These simulations prove that using equation (3), that is a rapidly decaying covariance sequence, the definition of the PSD hold for both equations (1) and (2).

1.2 Periodogram-based Methods Applied to Real–World Data

Part a) - Sunspot Time Series

In this section, we use a modified periodogram technique with a Hanning window using the MATLAB function periodogram() , to analyse the spectral estimate of sunspot time series. To remove any zero-valued points, and later be able to apply the logarithmic function to the data, we add a small DC offset value to the data using the eps function in MATLAB and obtain the periodogram. We then proceed to remove the mean from the data (i.e. the DC component at $\omega = 0$) and linearly detrend the data which allows to remove lower frequency components of the spectrum in order to more easily identify higher frequency components. We also applied the logarithm to the raw pre-processed data (with the additional small DC offset) and subtracted the mean of the log data to observe the changes in the periodicity of the data through this process.

As shown in Figure 2, we see that by centering and detrending the raw data we identify more easily the peaks in the spectrum. From the dominant spectral peak, we observe a maximal sunspot activity at 0.09 cycles/year (which corresponds to 11.1 years). However, using by doing that, the first harmonic at around 0.17 cycles/year is hardly visible on the periodogram. When applying centered logarithmic data, the spectral peak at 0.09 cycles/year is still dominant but this time we can identify more easily the first and second harmonics respectively at 0.17 and 0.26 cycles/year. We also note the presence of noise in the form of a peak at 0.01 cycles/year in all three plots.

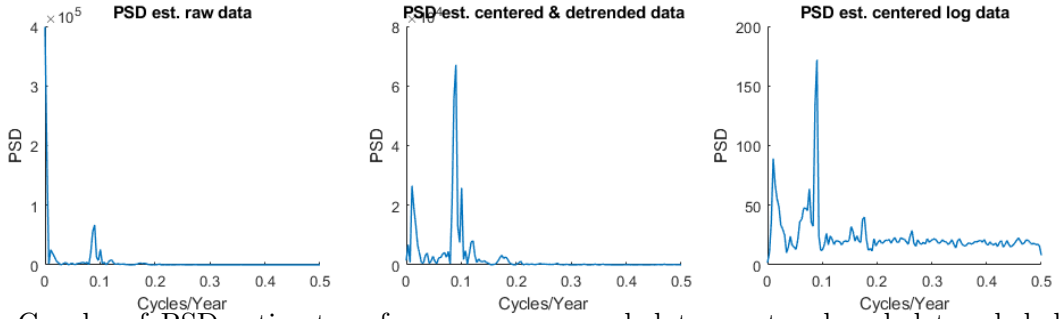


Figure 2: Graphs of PSD estimates of raw pre-processed data, centered and detrended data and centered logarithmic data respectively.

Part b) - The basis for brain computer interface (BCI)

The following task aims to carry out a spectral analysis on the electroencephalogram (EEG) to determine the steady state visual evoked potential (SSVEP) obtained by flashing a visual stimulus at a subject and detect the frequency of the signal.

Initially the PSD estimates were obtained by applying the standard periodogram technique to the entire recording. This is shown in the top graph of Figure 3. Then, we used the Bartlett's method of averaged periodograms with different window lengths ($\Delta t = 10s, 5s, 1s$ respectively). This is shown in the bottom graph of Figure 3, where all the PSD amplitudes were converted to decibel value for easier peak detection. This is due to the low amplitude of higher harmonics of the SSVEP fundamental frequency and confirms the discussion in part 1.2.1. To be able to fairly compare the different spectral analysis approaches, we used the same number of DFT samples per Hz. From Figure 3, see that in both cases we are able to detect the fundamental frequency at 13 Hz along with the first harmonic (26 Hz) and the third harmonic (39 Hz; all of them highlighted in the top graph but excluded from, the bottom one for clarity). The Bartlett method reveals another strong peak around 8-10 Hz which represents the alpha-rhythm and is attributed due to the tiredness of the subject rather than the SSVEP. Both methods are able to detect a peak at 50 Hz, which is the power-line interference and is not the SSVEP either. Finally, using window lengths of 5s and 10s, we observe the third harmonic of the SSVEP with a weak peak at 52 Hz, which is not detectable using the standard method.

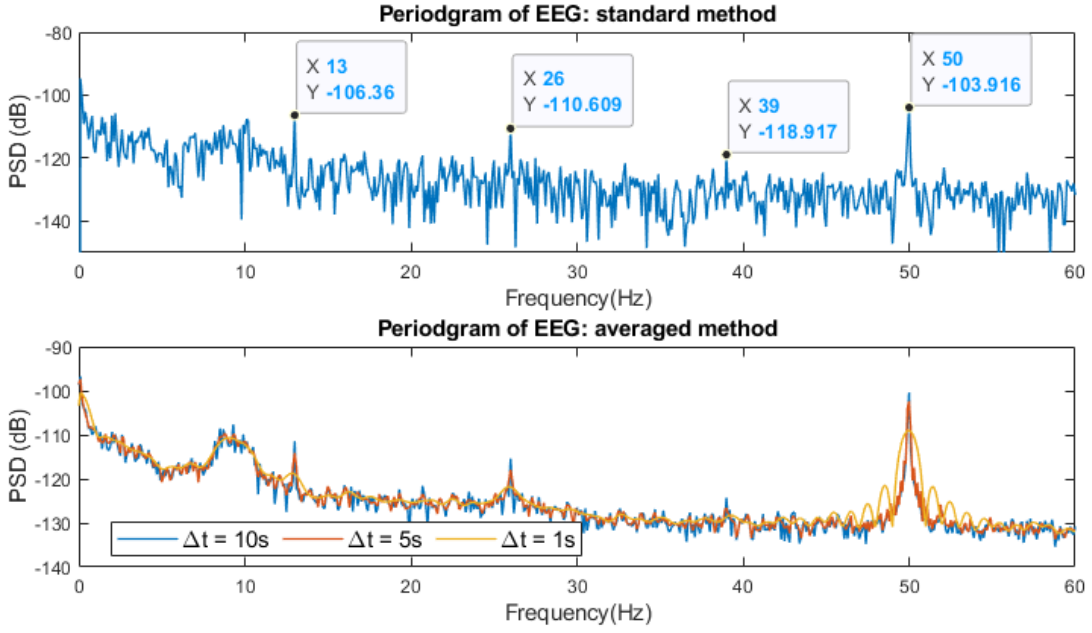


Figure 3: PSD estimates periodogram using standard and averaged approaches for EEG data.

From Figure 3, we observe that the Bartlett method (averaged periodogram) produces clearer peaks in the spectrum than the standard periodogram method which makes it easier for peak identification. Comparing the standard method with different window sizes in more detail in Figure 4, we observe that the averaged periodogram of window length of 10 seconds result in a large reduction in variance compared to the standard method. However, the harmonic at 39 Hz becomes no longer discernible in the averaged periodogram suggesting the introduction of a bias concerning the PSD estimate. This trade off between the bias and the variance is further observable in the second plot of Figure 4 which compares the standard method with the averaged method with window size 1s. Whilst, we vastly reduce the variance, the introduced bias makes the alpha rhythm (8-10 Hz) and the SSVEP (13 Hz) indistinguishable from each other. This trade-off means that averaging periodogram M times can reduce variance by a factor of 1/M.

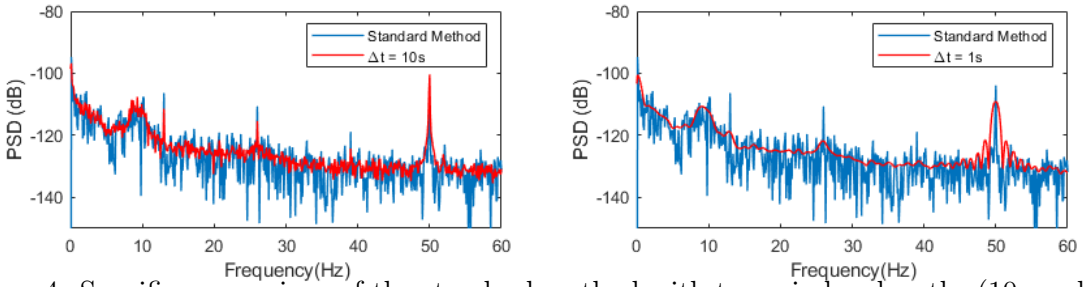


Figure 4: Specific comparison of the standard method with two window lengths (10s and 1s).

1.3 Correlation Estimation

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

Figure 5 shows the PSD estimates for three types of signal: WGN, noisy sinusoidal signals and low-pass filtered WGN. Those estimates are obtained are obtained from the biased and unbiased estimators derived in equations (6) and (7):

$$\textbf{Biased: } \hat{r}(k) = \frac{1}{N} \sum_{n=k+1}^{N} x(n)x^*(n-k) \tag{6}$$

$$\textbf{Unbiased: } \hat{r}(k) = \frac{1}{N-k} \sum_{n=k+1}^{N} x(n)x^*(n-k) \qquad (0 \leq k \leq N-1) \tag{7}$$

We observe from Figure 5 that for small values ($|k| < 500$) the ACF estimates with biased and unbiased estimators are very similar. However, as the absolute value of k increases, the ACF estimate with biased estimator converges to 0 while the one with unbiased estimator increases in value. Looking at the plots on the right, we observe that the PSD estimates using the biased ACF estimators tend to have a similar spectrum shapes: flat for WGN, impulse train for the noisy sinusoidal signal and attenuated high-frequency and flat low-frequency components for the filtered WGN. The PSD estimates with unbiased estimators however include negative values which are not present in true PSD.

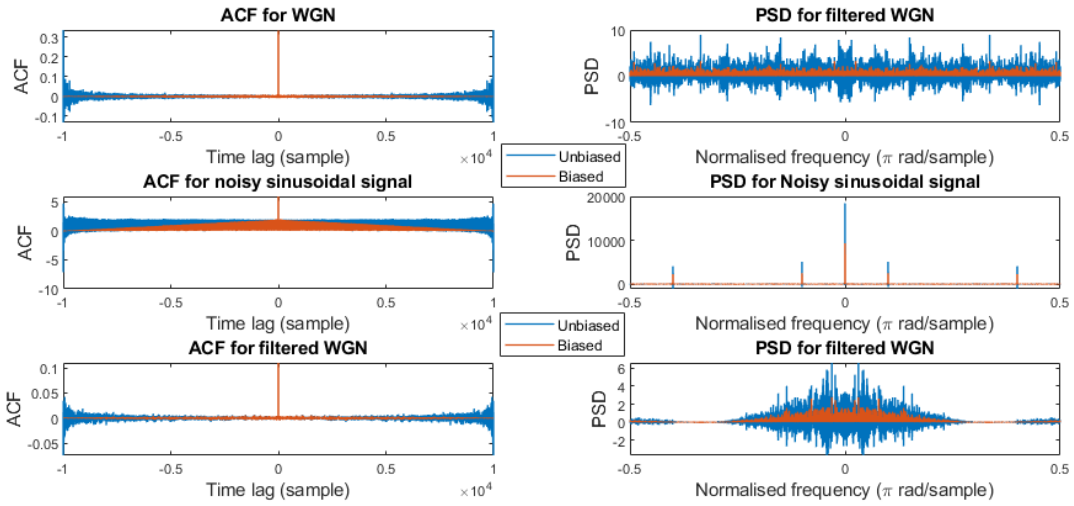


Figure 5: Plot of ACF and PSD estimates of biased and unbiased WGN, noisy sinusoidal signals and low-pass filtered WGN (n=10'000)

Judging from equation (7), it may look like that the unbiased estimate is more appropriate as its mean matches the true mean of PSD, it can be highly erratic for large lags (k-values) close to N. Because of the denominator $N - |k|$ from equation (7) involved in the estimator and the small number of samples available for large k values, the estimate of the ACF may not be positive definite which can result in negative PSD values (as seen in Figure 5). Therefore, we usually prefer to use the biased estimator because it has lower variance and is asymptotically unbiased for large k values (close N).

Part b & c) - Multiple realisations of PSD estimate (absolute and dB)

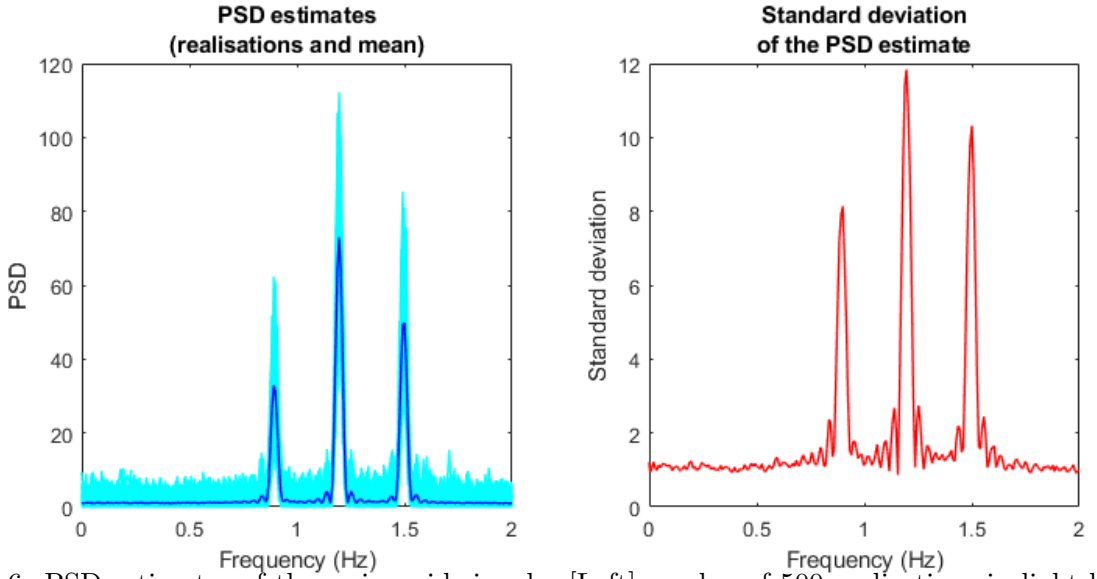


Figure 6: PSD estimates of three sinusoid signals. [Left] overlay of 500 realisations in light blue and their mean in dark blue. [Right] standard deviations of the 500 estimates.

Figure 6 shows the PSD estimates of 500 realisations of a random process when using the ACF biased estimator. The random process consists of three sinusoids (0.9, 1.2, 1.5Hz) corrupted by noise. The left plot of Figure 6 shows that the PSD estimate can detect the location of the three spectral peaks. In other locations, the mean value converges to 1. As N increases to infinity, we expect

the mean value of the PSD estimate to converge to the true PSD. The right plot of Figure 6 shows the standard deviation of the PSD estimate, which is proportional to the PSD value and hence we recognise the spectral peaks in the maximums of the standard variation. However, because variance is proportional to the square magnitude of the true PSD value, which is independent of N, as the length of data increases the variance will not go to zero making the periodogram an inconsistent estimator.

In comparison, Figure 7 shows the same estimates and standard deviation but plotted in decibels (dB). Because we use a logarithmic operation to convert the data in decibels, we perform a smoothing operation on the parts of the spectrum with the largest gradients. As a result, the resulting spectral peaks of the gradient get attenuated and the noise fluctuations are amplified. This method allows for an easier detection of the peaks in the spectrum since the variations are reduced by the logarithmic smoothing operation. The right graph of Figure 7 also shows that the peaks are identifiable at minimums of the standard variation compared to maximums before. Lastly, this method also allows clearer detection of low-amplitude signals in spectrum especially when a signal component has significantly larger amplitude, which may be harder to identify in a normal spectrum.

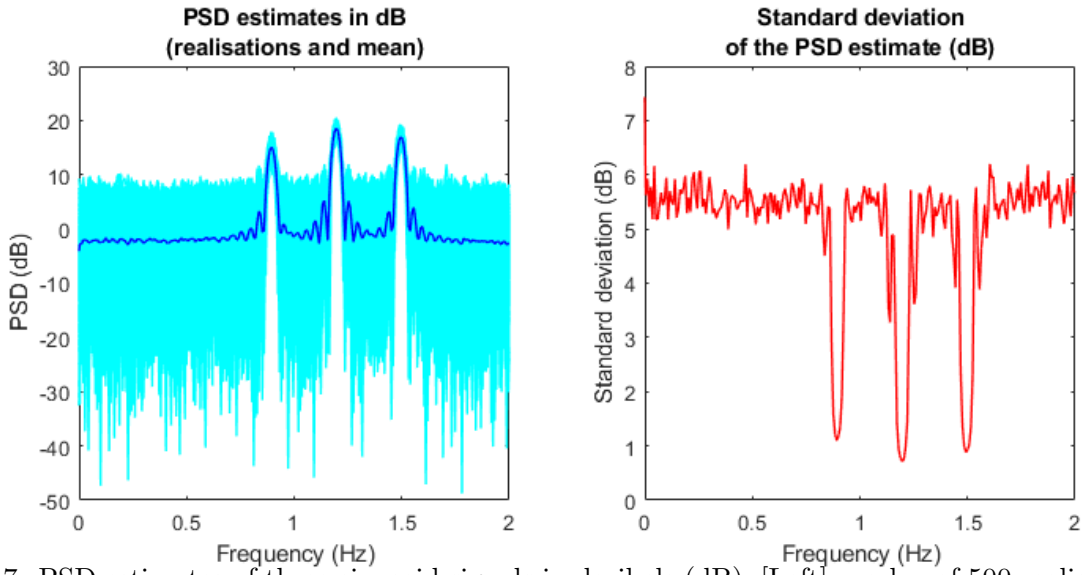


Figure 7: PSD estimates of three sinusoid signals in decibels (dB). [Left] overlay of 500 realisations in light blue and their mean in dark blue. [Right] standard deviations of the 500 estimates.

Part d) - Complex exponential signals generation

Figure 8 shows the periodogram of complex exponential signals with frequencies of 0.3Hz and 0.32Hz with the spectral estimate of different signal lengths (N). We notice that for N=20, the periodogram is not able to identify the two lines in the spectrum that corresponds to the two signals frequencies. When we increase the number of data samples, the periodogram begins to show the correct line spectra (N=40). However, because the resolution of the periodgram Δf is proportional to the reciprocal of the number of sample (1/N), we need to have a minimum of samples to start clearly separating the complex exponentials. In the case of normalised frequencies 0.3 and 0.32Hz, we require at least $N = 0.89/\Delta f = 44.5$. As the number of samples increases, because of the one-to-one mapping between time and frequency domains, there are more frequency kernels available and the periodogram becomes sharper and closer to the ideal pulses. For lengths lower than N=45, other alternatives methods should be considered such as subspace method.

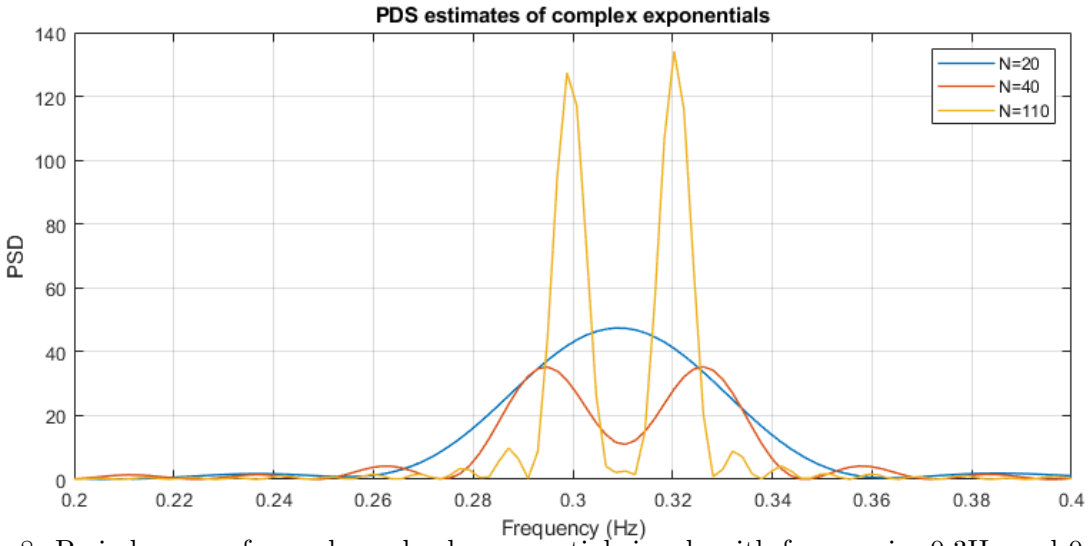


Figure 8: Periodogram of complex valued exponential signals with frequencies 0.3Hz and 0.32Hz..

Part e) - Frequency estimation by MUSIC

Figure 9 shows the code used to implement the Multiple Signal Classification algorithm (MUSIC) to estimate signal spectrum as part of subspace method.

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

Figure 9: Code implementation of the Multiple Signal Classification algorithm (MUSIC).

The first line uses the `corrmtx` function to obtain the auto correlation matrix estimate. The parameters used are `x` which is the noisy exponential input, `14` specifies the dimension of the correlation matrix and `'mod'` which corresponds to the modified covariance method which may produce more accurate results. The output of this function are `X` which is the rectangular Toeplitz matrix and `R` which is the autocorrelation matrix. The second line perform the MUSIC algorithm with inputs `R` being the autocorrelation matrix, `2` being the dimensionality of the signal subspace, `[]` uses the default FFT points, `1` denotes the normalised frequency and `'corr'` indicates that `R` is the correlation matrix estimate. Out of this function we get two outputs: `S` the pseudospectrum and `F` the normalised frequency. Finally, the third line simply plots the pseudospectrum against the normalised frequency while limiting the displayed frequency range between 0.25 and 0.40.

Figure 10 shows the PSD estimate of the signal used in part d) using the MUSIC method for N=20 to be able to compare the periodogram with the one obtained in Figure 8. Whilst the standard periodogram was unable to retrieve the spectral peaks because of the high bias, the MUSIC algorithm successfully identifies the peaks at 0.3 and 0.32 Hz even for low data lengths. MUSIC method has a lower bias (hence a higher resolution) than the periodogram estimate and is able to exploit the data structure better.

However, one of the main disadvantages of the MUSIC method is the need to know, or estimate, parameter `p` for eigenvalues distributions. We also see from the right graph in Figure 10 that the variance is significantly larger than the periodogram. Finally, the computational complexity of the MUSIC method is significantly larger than the periodogram as we need algorithms to determine the signal subspace dimension. A large `p` increases the complexity, and a small `p` can reduce the accuracy due to a loss of information.

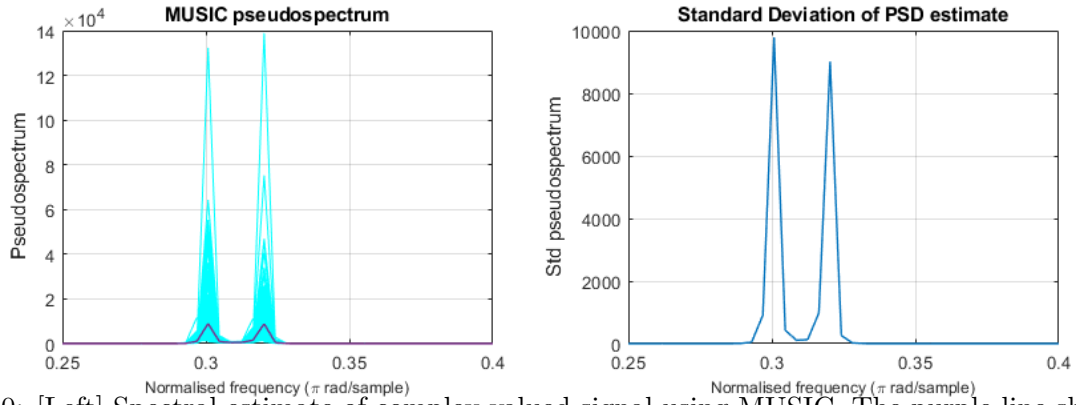


Figure 10: [Left] Spectral estimate of complex valued signal using MUSIC. The purple line shows the mean. [Right] Standard deviation of the MUSIC method.

1.4 Spectrum of Autoregressive Processes

Part a)

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

$$\mathbf{R}_{xx}\mathbf{a} = \mathbf{r}_x \Rightarrow \mathbf{a} = \mathbf{R}_{xx}^{-1}\mathbf{r}_x \tag{8}$$

where $\mathbf{R}_{xx}$ is the autocorrelation matrix of the signal of interest. The biased estimator provides a positive definite Toeplitz $\mathbf{R}_{xx}$ which is invertible since all the eigenvalues of the ACF will be positive. On the other hand, when the unbiased estimator is used then the ACF will not decay to zero for lags of large values which may not satify the positive definite condition. In this case, we may not be able to get the AR parameter $\mathbf{a}$ from the equation above.

Part b)

Figure 11 illustrates that low order autoregressive models with large prediction errors are not able to describe the variations of the signal properly. As we increase the order of the model, there is more freedom to fit the sample and the prediction error consistently decreases. This phenomena is true until model order reaches 9 where the minimum mean squared prediction error happens. There is then a trade off between computational complexity and the model accuracy as well as the risk of data overfitting when the model order increases. The poor performance of the Ar model for small orders is explained by the ACF bias being high for small sample size. Hence we cannot consider the unbiased estimator as an alternative and we ought to increase the sample size to improve the model.

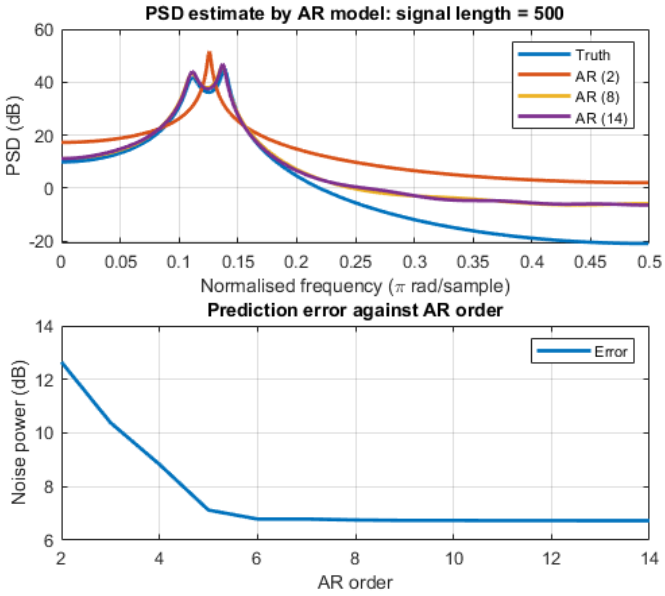


Figure 11: Spectral estimates by AR models of different orders and prediction error (N=500).

Part c)

Figure 12 illustrates that increasing the sample size helps to reduce the influence of randomness on the model selection. Having a larger sample size gives more information about the signal to the AR model which allows a better detection of the underlying data pattern. Model orders as low as 4 is enough to successfully estimate the signal with enough data samples. With 10'000 samples, the minimum MSE value is found at order 6 while it was at 9 for 1'000 samples. Beyond that point the complexity increases and the MSE stays around the same level, while below that level AR models cannot capture the variations as well.

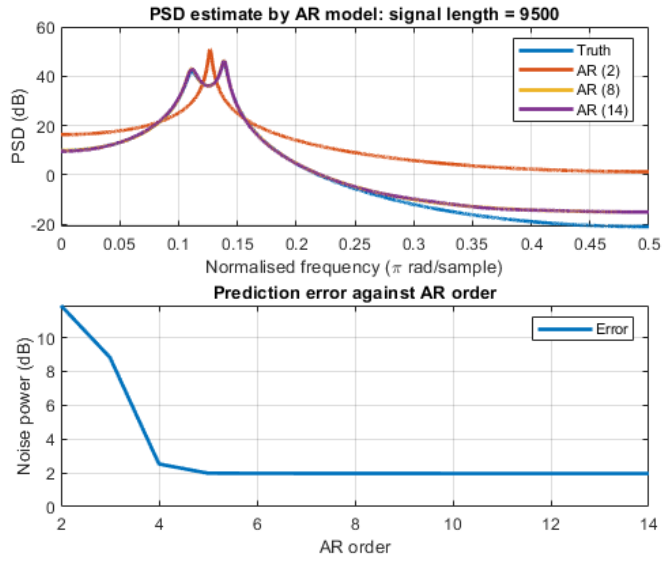


Figure 12: Spectral estimates by AR models of different orders and prediction error (N=9500).

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

Part a)

The ECG experiment was composed of the RR interval (RRI) data from three trials: unconstrained breathing, constrained breathing at 50 beats per minute (fast breathing) and constrained breathing at 15 beats per minute (slow breathing). The frequency component of the RRI data thus corresponds to the breathing rate of the studied subject. We observe from Figure 13 three periodograms estimates for each trial: one standard periodogram, one averaged periodogram with window length 50 seconds and one averaged periodogram with window length 150 seconds. These periodograms estimates were computed using the MATLAB function `pwelch` with a Hamming window and previously normalised and detrended data.

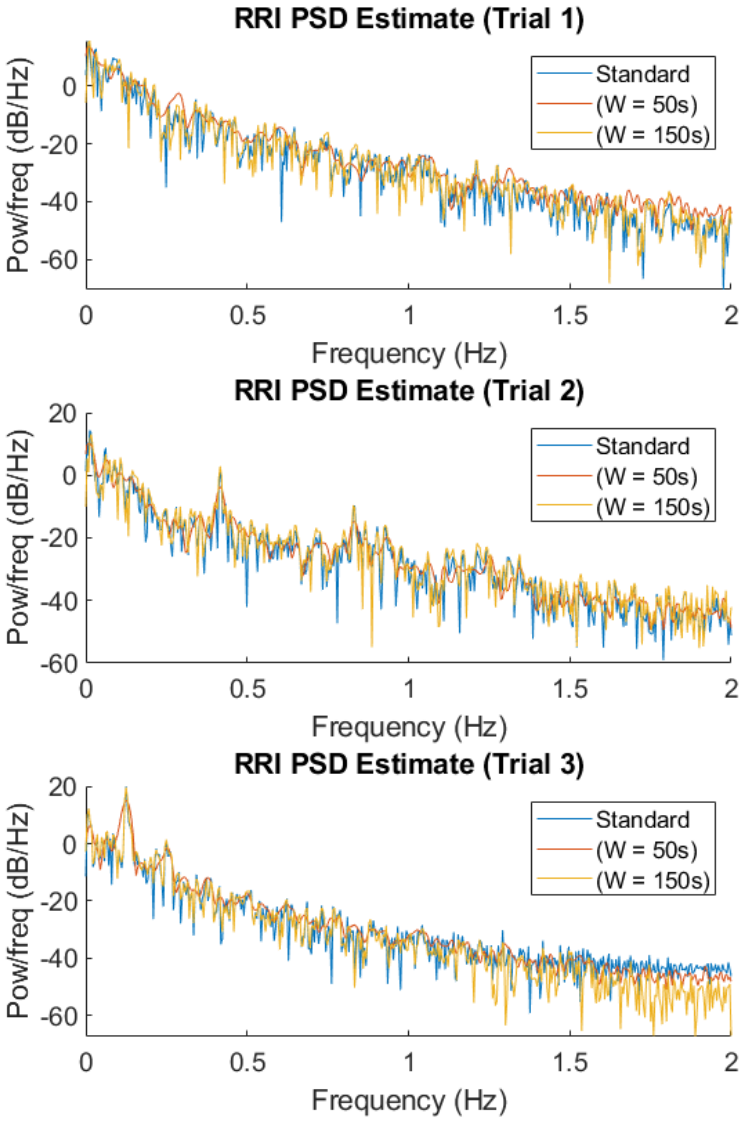


Figure 13: Spectral estimates by AR models of different orders and prediction error (N=9500).

Part b)

We can use the relationship $breaths/min(BPM) = 2 \times f$ in order to determine estimates of the BPM analysing the graphs. Using Figure 13, we see that in the first trial we fail to observe any distinct peak but rather initial high amplitude at low frequencies between 0 and 0.25Hz (which corresponds to 12BPM) which corresponds to an unconstrained breathing rate. The graph of the second trial shows a spectral peak just before 0.42Hz (50 BPM) which is in line with the fast constrained breathing trial. Finally, the third trial shows a spectral peak at 0.12 Hz (14.5 BPM) which is consistent with the last constrained trial at 15 BPM.

Part c)

Empirically, the optimal model orders for trials 1,2 and 3 were found to be 3,7 and 4 respectively. We display the AR estimates for each trial in Figure 14 using the `pyulear` function. Trial 3 has the most discernible peak at 0.12Hz, which corresponds to our finding using the periodogram earlier. The low frequency band of trial 1 can be thought to be represented via a wide peak in Figure 14 while the AR estimate exhibits weak peaks at the frequencies of interest in trial 2. Overall, we observe that even though AR spectral estimates show much lower variance that periodograms, they provide much weaker peaks that are hardly identifiable.

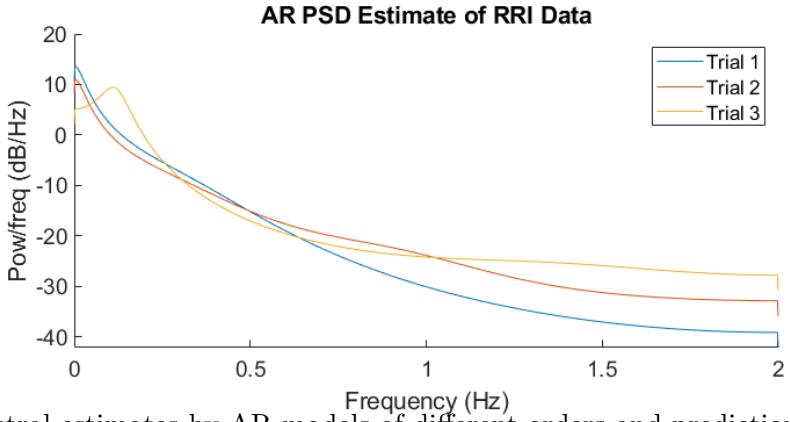


Figure 14: Spectral estimates by AR models of different orders and prediction error (N=9500).

1.6 Robust Regression

Part a)

In this section, we will use five matrices: an input matrix X with its noise corrupted version X_{noise}, a training output matrix Y as well as their their test equivalent X_{test} and Y_{test}. In Figure 15, we plotted the singular values of X and X_{noise}. As we can see from the graph on the left, the principal components of X are of rank 3, while the noise corrupted version of the matrix X_{noise} is of rank 10. The right graph shows the square error between the singular values and we observe that for larger components, this error increases. In fact, when the noise gets sufficiently large, the singular values of the noise subspace become as large as the singular values in the signal subspace.

Part b)

Using the matlab command `svd`, we obtain the low-rank approximation of X_{noise} keeping only the three principal components and obtaining $\tilde{X}_{noise}$. We then computed the MSE for each input feature respectively between X and X_{noise} and between X and $\tilde{X}_noise$ as shown in Figure 16. We observe that the MSE is significantly larger between X and X_noise for all the input features. This

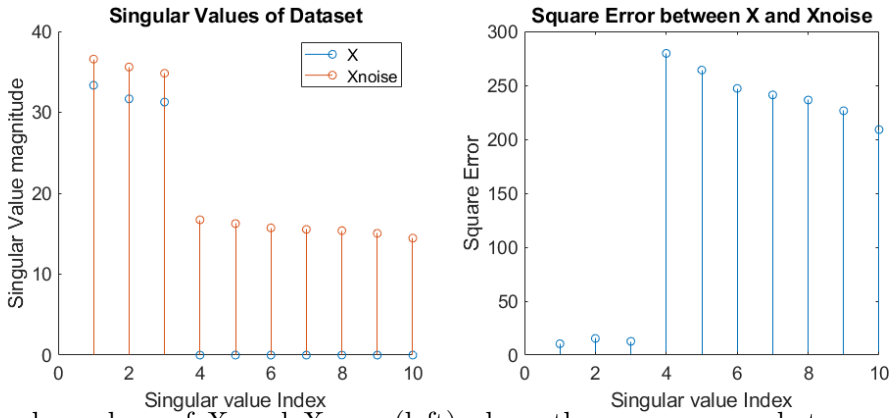


Figure 15: Singular values of X and X_{noise} (left) along the square error between their respective singular values (right).

is reflected in the total MSE figures which is 0.2435 between X and X_{noise}, while it is only 0.0733 between X and $\tilde{X}_{noise}$.

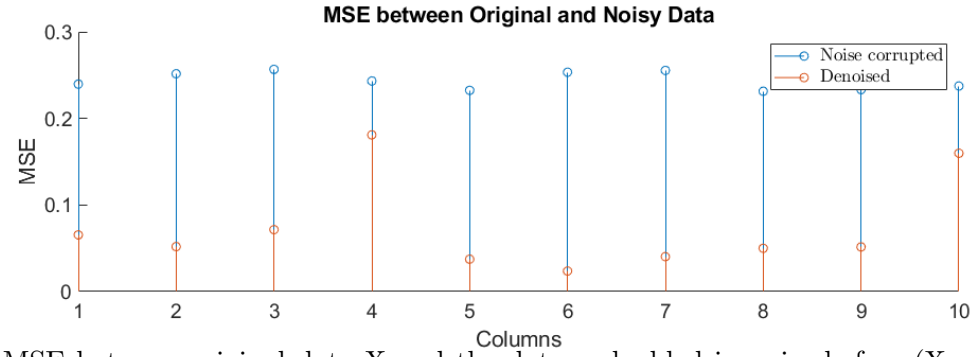


Figure 16: MSE between original data X and the data embedded in noise before (X_{noise}) and after ($\tilde{X}_{noise}$) the low-rank approximation.

Part c)

In this section, we will be computing the ordinary least squares (OLS) and the principal component regression (PCR) solutions for the parameter matrix B which relates X_{noise} and Y and comparing the solutions obtained using each method. We used the following equations to obtain the OLS (9) and PCR (10) solutions:

$$\hat{\mathbf{B}}_{OLS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{9}$$

$$\hat{\mathbf{B}}_{PCR} = \mathbf{V}_{1:r}(\mathbf{\Sigma}_{1:r})^{-1}\mathbf{U}_{-1}^T\mathbf{Y} \tag{10}$$

We have determined in part a) that the signal space is composed of the three principal components. Hence, in equation (10), $r = 3$. From Figure 17, we observe negligible discrepancies in terms of performance between the OLS and PCR methods for the given data.

	RMSE_{est}	RMSE_{test}	MAE_{est}	MAE_{test}
OLS	0.847	0.689	0.658	0.528
PCR	0.846	0.686	0.658	0.525

Table 1: Comparison of RMSE and MAE estimation and test values using the OLS and PCR methods.

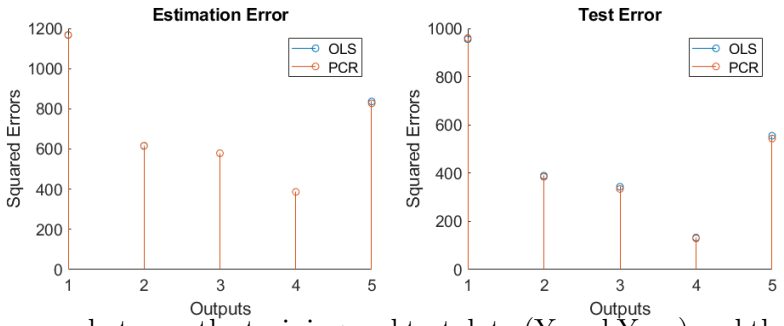


Figure 17: Squared errors between the training and test data (Y and Y_{test}) and the estimation obtained using the PCR and OLS methods.

Part d)

In this section, we use the `regval` script provided to generate new data realisations and their estimates for the given set of parameters B. In Figure 18 is shown the averaged squared error after 100 realisations with each of the two methods (OLS and PCR). We can observe that both methods exhibit very strong performances in the estimation task where the PCR method only outperforms the OLS method by 1.6%. However, these have some limitations as we need to determine the number of principal components prior to the estimation which can be challenging.

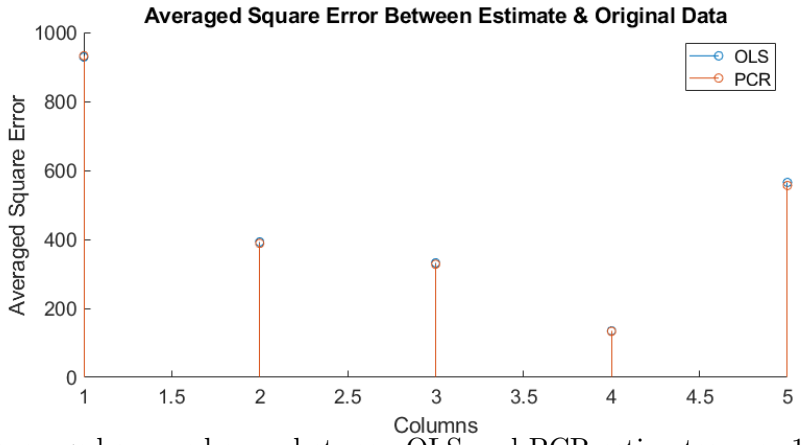


Figure 18: Averaged squared error between OLS and PCR estimates over 100 realisations.

2 Adaptive signal processing

2.1 The Least Mean Square Algorithm (LMS)

Part a)

In this section, we will investigate adaptive filters and in particular using LMS filter as a dynamical system with lagged inputs $x(n-1)$ and $x(n-2)$, outputs estimates $\hat{a}_1, \hat{a}_2, \hat{X}(n)$. The task is to determine the entries of the correlation matrix $\mathbf{x}(n) = [x(n), x(n-1)]^T$, using LMS as a dynamical system. The original parameters are given to be $a_1 = 0.1, a_2 = 0.8, \sigma_\eta^2 = 0.25$. The correlation matrix $\mathbf{R}$ is shown in equation (11):

$$\mathbf{R} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(1) & r_{xx}(0) \end{bmatrix} = \begin{bmatrix} \sigma_n^2 & r_{xx}(1) \\ r_{xx}(1) & \sigma_n^2 \end{bmatrix} \tag{11}$$

For any AR(2) process we also obtain the following relationships:

$$\sigma_x^2 = \left(\frac{1-a_2}{1+a_2}\right)\frac{\sigma_\eta^2}{(1-a_2)^2 - a_1^2} \tag{12}$$

$$\rho_1 = \frac{r_{xx}(1)}{r_{xx}(0)} = \frac{r_{xx}(1)}{\sigma_x^2} = \frac{a_1}{1-a_2} \tag{13}$$

Using the given parameters we can calculate the parameters of equations (12) and (13) and plug them in the correlation matrix of equation (11):

$$\sigma_x^2 = \left(\frac{0.2}{1.8}\right)\frac{0.25}{0.2^2 - 0.1^2} = 0.926 \tag{14}$$

$$r_{xx}(1) = 0.926 \times \frac{0.1}{1-0.8} = 0.463 \tag{15}$$

$$\mathbf{R} = \begin{bmatrix} 0.926 & 0.463 \\ 0.463 & 0.926 \end{bmatrix} \tag{16}$$

In order for the LMS to converge to the mean of the Wiener optimal solution, the step size μ must lie in the range:

$$0 < \mu < \frac{2}{\lambda_{max}} \tag{17}$$

In this case, λ_{max} is the largest eigenvalue of the correlation matrix. The eigenvalues of the correlation matrix previously calculated are $\lambda_1 = 0.463$ and $\lambda_2 = 0.1389$, the step size range is $0 < \mu < 1.44$.

Part b)

In this section, we used an LMS adaptive predictor using 1000 realisations with the same parameters as in the previous section, and plotted the squared prediction error over time for a single trial and the average over 100 realisations. For this task, we used a `lms_estimator` function that we created along with the `filter` function to generate the samples. The LMS algorithm was implemented by using estimates of the ACF and CCF and updating it using equation (18):

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{x}(n) \tag{18}$$

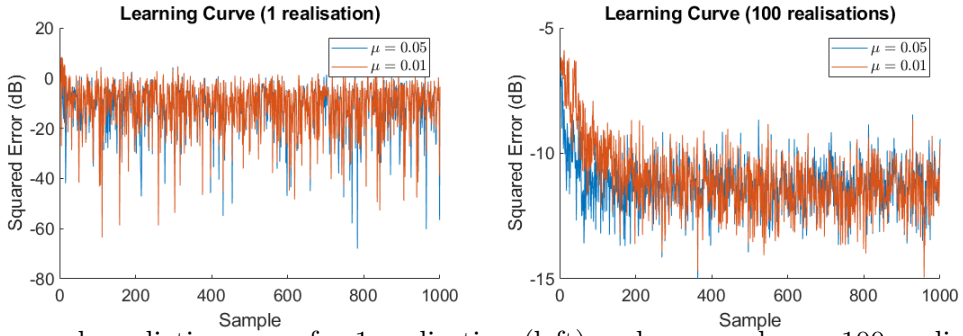


Figure 19: Squared prediction error for 1 realisation (left) and averaged over 100 realisation (right) for different step sizes.

We can see the results from Figure 19. Due to high variance of the squared error, we cannot draw any conclusive result over the convergence of the algorithm or the effect of the step sizes over a single realisation. However, when we average the result over 100 realisation, we observe faster convergence

for the larger step size of $\mu = 0.05$, which converges to a steady state after around 100 samples, whereas for $\mu = 0.01$ this occurs later around 200 samples. However, the error converges higher with the larger step size of $\mu = 0.05$ than it does for step size $\mu = 0.01$.

Part c)

In this section, we estimate the misadjustment of the LMS by time-averaging over the steady state of the ensemble-averaged learning curves using 100 independents trials of the experiment, comparing the theoretical LMS misadjustment given in equation (20). This misadjustment is the ratio of additional error power due to the adaptive filter, quantified by the excess mean squared error (EMSE), by the minimum attainable MSE of a Wiener filter, as shown in equation (19):

$$M = \frac{EMSE}{\sigma_\eta^2} \tag{19}$$

For small step-sizes, we can approximate the misadjustments of the LMS with:

$$M_{\text{LMS}} \approx \frac{\mu}{2}\text{Tr}\{\mathbf{R}\} \tag{20}$$

In this section, we used the two values of $\mu_1 = 0.05$ and $\mu_2 = 0.01$. Given matrix $\mathbf{R}$ from equation 16, we can compute the theoretical misadjustments as $M_1 = 0.0463$ and $M_2 = 0.0093$. To estimate M from our data, we generated 200 thousand samples over 100 realisations and obtained the MSE curve for each learning rate. To remove non steady state dynamics, we discarded the first 1000 samples and averaged the MSE curve over time. The results can be found in the table below:

	$M_{esimated}$	$M_{theoretical}$
$\mu = 0.05$	0.0523	0.0463
$\mu = 0.01$	0.0085	0.0093

Table 2: Comparison of theoretical and estimate LMS misadjustments.

We observe from the table that the stimated misadjustment values are very close to the theoretical values with M being higher for larger μ since $M \approx 0.5\mu N\sigma_x^2$. Specifically we find the absolute rror to be 13% for $\mu = 0.05$ and 8.6% for $\mu = 0.01$. We can explain these from the fact that the theoretical value of M is only an approximation and only a finite number of realisation was done. In though, both step sizes ensure convergence to the mean, using a larger step size increases the MSE of LMS algorithm which results in a higher value of M. Therefore, using a larger step size will bring faster convergence but will increase the bias and variance of LMS coefficients (for finite number of realisations) and result in higher mean square weight error.

Part d)

In this section, we will estimate the steady state value of adaptive filters with step size $\mu = 0.05$ and $\mu = 0.01$. As shown in Figure 20, this was done by averaging the parameters learning dynamics of 1,500 samples over 100 realisations. The estimates of the parameters are down in the table below:

	a_1	a_2
$\mu = 0.05$	0.069	0.714
$\mu = 0.01$	0.086	0.777
Real values	0.1	0.8

Table 3: AR(2) LMS parameters estimates compared to real values.

As explained in the previous section, we see that for larger μ values, we get faster converges to the steady state values but we get larger variance oscillations due a higher step size. Doing so, can cause optimal solutions to be missed resulting in higher error in estimations compared to using lower and safer μ values. We often observe a trade-off between convergence speed and convergence in MSE.

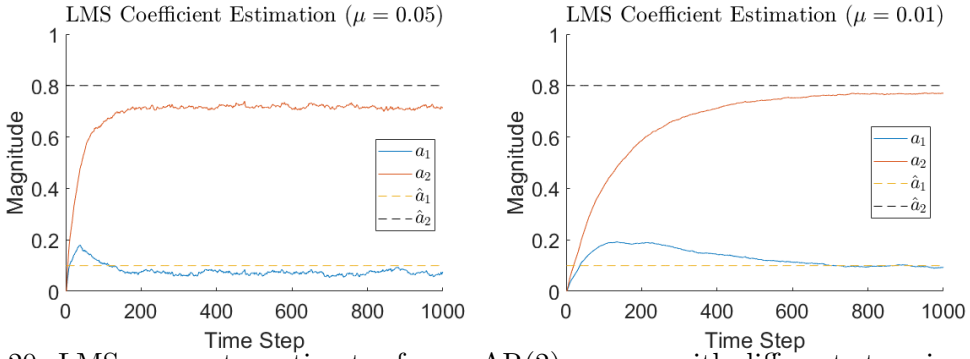


Figure 20: LMS parameter estimates for an AR(2) process with different step size values.

Part e)

In this section, we want to derive the LMS coefficient update for $\mathbf{w}(n)$ that minimises the cost function in equation (21):

$$J_2(n) = \frac{1}{2}(e^2(n) + \gamma\|\mathbf{w}(n)\|_2^2) \tag{21}$$

$$= \frac{1}{2}e^2(n) + \frac{\gamma}{2}\mathbf{w}^T(n)\mathbf{w}(n) \tag{22}$$

From there, we can derive the gradient of J_2:

$$\nabla_{\mathbf{w}} J_2(n) = \frac{1}{2}\frac{\partial e^2(n)}{\partial e(n)}\frac{\partial e(n)}{\partial y(n)}\frac{\partial y(n)}{\partial \mathbf{w}(n)} + \frac{\gamma}{2}\frac{\partial}{\partial \mathbf{w}}(\mathbf{w}^T(n)\mathbf{w}(n)) \tag{23}$$

$$= \frac{-1}{2}(2e(n))(\mathbf{x}(n)) + \gamma\mathbf{w}(n) \tag{24}$$

$$= -e(n)\mathbf{x}(n) + \gamma\mathbf{w}(n) \tag{25}$$

Now, we know that the update equation is the sharpest descent to find the minimum of the cost function J_2 and can be defined by (26):

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\nabla_{\mathbf{w}} J_2(n) \tag{26}$$

If we replace the value in equation (25) into equation (26), we get:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu[\gamma\mathbf{w}(n) - e(n)\mathbf{x}(n)] \tag{27}$$

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

Hence, we see that the update rule of the Leaky LMS algorithm, described by equation (28) is in the direction of the steepest descent of the cost function J_2.

Part f)

In this section, we computed the parameters defined in Part a) using a Leaky-LMS algorithm for different values of μ and γ as reported in Figure 21. Unlike when we used the standard LMS algorithm, in this case the parameters estimates do not converge to their true values and the absolute error increases with larger leakage coefficients. The Leaky-LMS algorithm allows stability of the LMS algorithm when the autocorrelation matrix of the process has zero eigenvalues by forcing filter weights to converge. However, as we can see from Figure 21, in our case as γ gets higher, the deviation from the true AR coefficients increases. This can be explained by the fact that the optimal Wiener-Hopf can be expressed with equation 29, while the Leaky-LMS weights are expressed with equation 30:

$$w_{opt} = \mathbf{R}^{-1}\mathbf{p} \tag{29}$$

$$w_{optLeaky} = (\mathbf{R} + \gamma\mathbf{I})^{-1}\mathbf{p} \tag{30}$$

Thgis means that the optimal Leaky-LMS weights converge to Wiener-Hopf solution as γ tendsa to zero. Similarly, the MSE of the Leaky-LMS algorithm does not converge to the minimal MSE of a Wiener filter but the variance and bias estimates increase. Additionally, we see from equation 28 that the leakage coefficient γ reduces the sensitivity of the algorithm coefficient updates to current coefficients values.

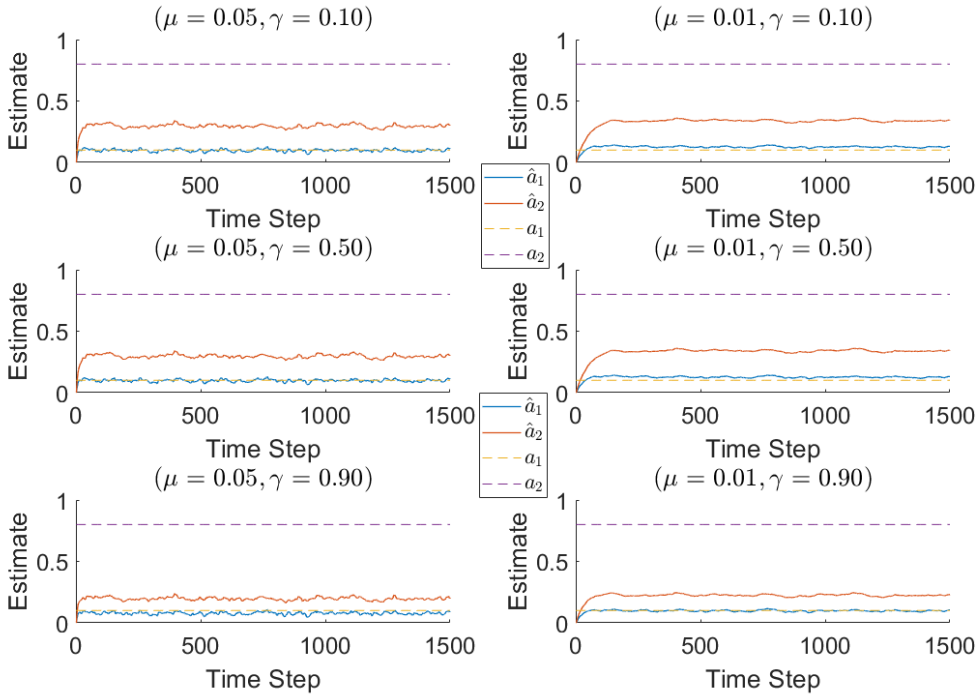


Figure 21: LMS parameter estimates for an AR(2) process with different learning rates μ and leakage coefficients γ.

2.2 Adaptive Step Sizes

Part a)

In this section, we will use adaptive step sizes and implement different gradient adaptive step size (GASS) algorithms. Namely, the Benveniste, Ang & Farhang and the Mathews & Xie algorithm. Using these algorithms, the weight update is modified in the form shown in equation (31) and the step size rule is given in equation (32).

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(n)e(n)\mathbf{x}(n) \tag{31}$$

$$\mu(n+1) = \mu(n) + \rho e(n)\mathbf{x}^T(n)\psi(n) \tag{32}$$

The three GASS algorithms were implemented and had their performance compared to the MA(1) model given by equation (33):

$$x(n) = 0.9\eta(n-1) + \eta(n), \qquad \eta \sim N(0, 0.5) \tag{33}$$

The performances of the different algorithms are shown in Figure 22 with respect to the standard LMS. The weight error curves are obtained from averaging 100 iterations of the algorithms, each having 100 samples. The hyper parameters were $\rho = 0.002$ and $\alpha = 0.8$. In general, standard LMS algorithm suffer from a trade-off between convergence speed and steady-state error variance. Due to its low complexity, the standard LMS algorithm is very fast to compute but is slower to converge than all of the GASS algorithm in most cases. We see from Figure 22, that given a large enough initial learning rate, all the GASS algorithms are able to minimise the cost function with respect to the step-size μ, thus reducing the steady-state error and converging faster. The GASS algorithms are generally more robust than standard LMS algorithm in terms of error and provide faster convergence, in the expense of higher computational complexity. As these algorithms constantly adapt the step size to minimise the cost function with respect to the weight vector, they are able to obtain better results but will naturally have greater computational complexity compared to the standard LMS algorithm which uses static step size.

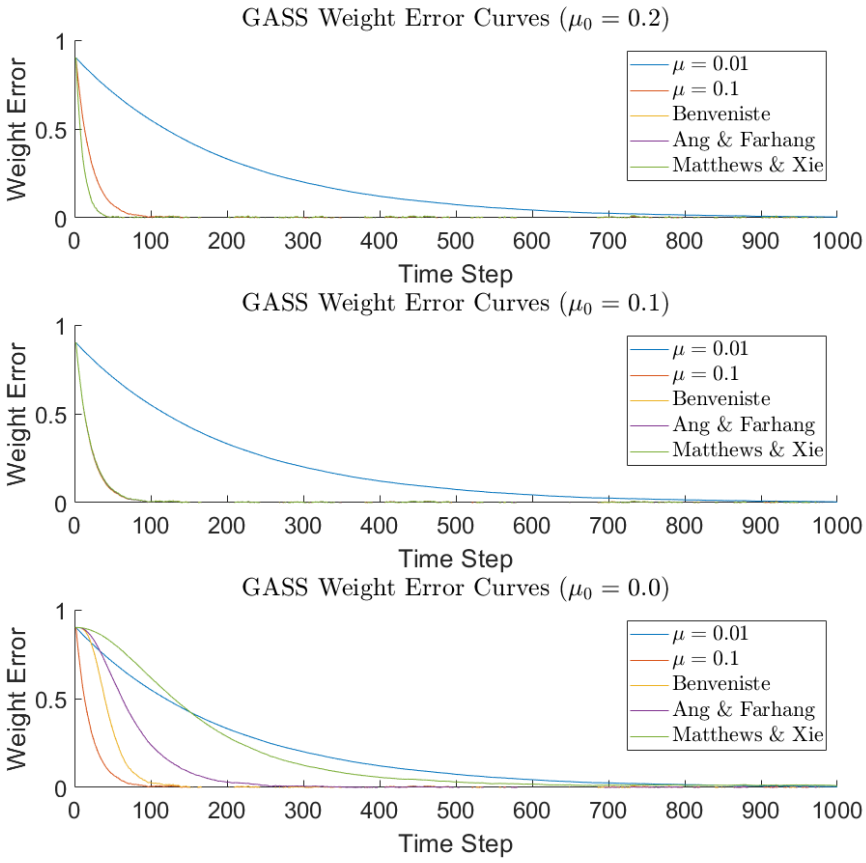


Figure 22: Weight error curves for standard LMS algorithms and GASS algorithms compared for different values of initial learning rate (μ_0

Part b)

In this section, we will use the normalised LMS algorithm, which normalises the step size which

is described in equation (35). The parameter ϵ is the regularisation factor preventing the algorithm from becoming unstable.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\beta}{\epsilon + \|\mathbf{x}(n)\|_2^2} e(n)x(n) \tag{34}$$

$$= \mathbf{w}(n) + \frac{\beta}{\epsilon + \mathbf{x}^T(n)\mathbf{x}(n)} e(n)x(n) \tag{35}$$

The task will be to show that the NLMS algorithm described by equation (35) is equivalent to the update rule from equation (36) based on the *a posteriori* error e_p from equation (37):

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(n)e(n)\mathbf{x}(n) \tag{36}$$

$$e_p(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n+1) \tag{37}$$

We can start by substituting the value of $\mathbf{w}(n+1)$ in equation (36) into equation (37) we can obtain the *a posteriori* error in terms of the *a priori* error e_n as follows:

$$e_p(n) = d(n) - \mathbf{x}^T(n)(\mathbf{w}(n) + \mu e_p(n)\mathbf{x}(n)) \tag{38}$$

$$e_p(n)(1 - \mu\mathbf{x}^T(n)\mathbf{x}(n)) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n) \tag{39}$$

$$e_p(n)(1 - \mu\|\mathbf{x}(n)\|_2^2) = e(n) \tag{40}$$

$$e_p = \frac{e(n)}{(1 - \mu\|\mathbf{x}(n)\|_2^2)} \tag{41}$$

We can use the value of the *a posteriori* error from equation (41) and plug it in equation (36) to obtain:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\frac{e(n)}{1 - \mu\|\mathbf{w}(n)\|_2^2}\mathbf{x}(n) \tag{42}$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\frac{e(n)}{\frac{1}{\mu} - \|\mathbf{w}(n)\|_2^2} e(n)\mathbf{x}(n) \tag{43}$$

Equation (43) is equivalent to the NLMS algorithm from equation (35) for $\beta = 1$ and $\epsilon = 1/\mu$.

Part c)

In this section, we will implement the GNGD algorithm and compare its parameter estimates from task a) to the ones of the Benveniste's algorithm we used in Part 2.2.b. The results for the estimates of both algorithms are shown is Figure 23 for the determined optimal hyperparameters (Benveniste: $\rho = 0.005$, $\mu_0 = 1$ and GNGD: $\rho = 0.002$, $\mu_0 = 0.1$. From Figure 23 we observe that the GNGD algorithm converges faster than the Benveniste while displaying lower variance. Additionality, the computational complexity of the GNGD algorithm grows linearly with the model order ($\mathcal{O}(n)$), while in the case of the Benveniste, the computational complexity grows quadratically with the model order ($\mathcal{O}(n^2)$). This is mainly due to the outer product of $\mathbf{x}(n-1)\mathbf{x}^T(n-1)$ involved in the update of the sensitivity vector $\psi(n)$ which can be seen in equation (44):

$$\psi(n) = [\mathbf{I} - \mu(n-1)\mathbf{x}(n-1)\mathbf{x}^T(n-1)]\psi(n-1) + e(n-1)\mathbf{x}(n-1) \tag{44}$$

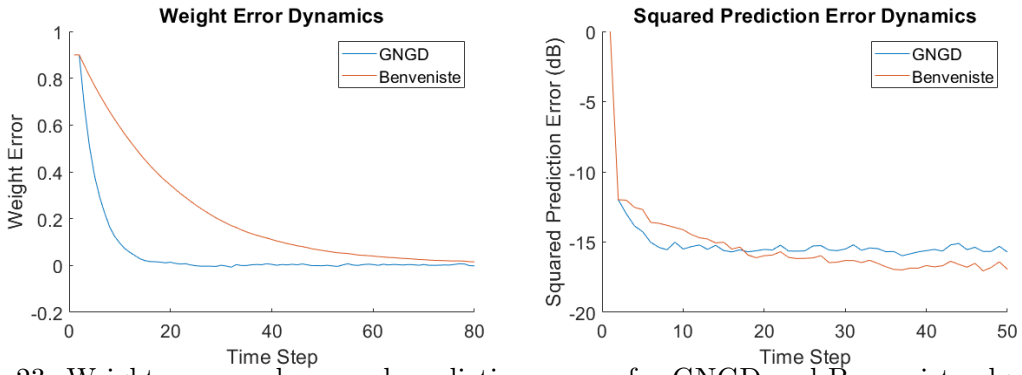


Figure 23: Weight error and squared prediction curves for GNGD and Benveniste algorithms

2.3 Adaptive Noise Cancellation

Part a)

In this section, different adaptive noise cancellation techniques will be investigated. We used the noise corrupted signal described by equation (45). The corresponding underlying signal has frequency 0.005 Hz and the noise is given by $\eta(n) = \nu + 0.5\nu(n-2)$ where $\nu(n)$ is a white noise with unit variance.

$$s(n) = x(n) + \eta(n) \tag{45}$$

The adaptive Line enhancer (ALE) is a noise reduction algorithm consisting of a delay operator followed by a linear operator. The delay is chosen such that the noise in the signal and the linear predictor input are uncorrelated. We can express the MSE as:

$$\begin{aligned}
\mathbb{E}[(s(n)-\hat{x})^2] &= \mathbb{E}[(x(n)+\eta(n)-\hat{x}(n))^2] && (46)\\
&= \mathbb{E}[(x(n)-\hat{x}(n))^2] + \mathbb{E}[\eta^2(n)] + 2\,\mathbb{E}[(x(n)-\hat{x}(n))\eta(n)] && (47)\\
&= \mathbb{E}[(x(n)-\hat{x}(n))^2] + \mathbb{E}[\eta^2(n)] + 2\,\mathbb{E}[x(n)\eta(n)] - 2\,\mathbb{E}[\hat{x}(n)\eta(n)] && (48)
\end{aligned}$$

Looking at equation (47), the first term $\mathbb{E}[(x(n)-\hat{x}(n))^2]$ is not function of noise. The second term $\mathbb{E}[\eta^2(n)]$ is the power of the noise which is a constant term. The third term $2\,\mathbb{E}[x(n)\eta(n)]$ is equal to zero as the signal and the noise are uncorrelated and the noise has zero mean. Hence, we only consider the forth and last term in order to minimise the MSE:

$$\begin{aligned}
\mathbb{E}[\hat{x}_\Delta(n)\eta(n)] &= \mathbb{E}[w^T(n)\mathbf{u}_\Delta(n)\eta(n)] && (49)\\
&= \mathbb{E}\left[\sum_{k=0}^{M-1} w_k s(n-\Delta-k)\eta(n)\right] && (50)\\
&= \mathbb{E}\left[\sum_{k=0}^{M-1} w_k (x(n-\Delta-k)+\eta(n-\Delta-k)\eta(n))\right] && (51)\\
&= \mathbb{E}\left[\sum_{k=0}^{M-1} w_k \eta(n-\Delta-k)\eta(n)\right] && (52)\\
&= \mathbb{E}\left[\sum_{k=0}^{M-1} w_k (v(n-\Delta-k)+0.5v(n-\Delta-k-2))(v(n)+0.5v(n-2))\right] && (53)
\end{aligned}$$

Since $v(n)$ is an i.i.d. white noise, we can write:

$$\mathbb{E}[v(n-i)v(n-j)] = 0, \qquad \text{for i} \neq \text{ j} \tag{54}$$

Because of that, we need $\Delta > 2$ and hence $\Delta_{min} = 3$ to ensure no overlapping between the terms $v(n-\Delta-k) + 0.5v(n-\Delta-k-2)$ and $v(n) + 0.5v(n-2)$ from equation (53). In this case, we get $\mathbb{E}[\hat{x}_\Delta(n)\eta(n)]$ which minimises the MSE. We can confirm that by looking at the graph on the left of Figure 24 which shows minimal MSPE for all the different filter orders for a delay $\Delta = 3$.

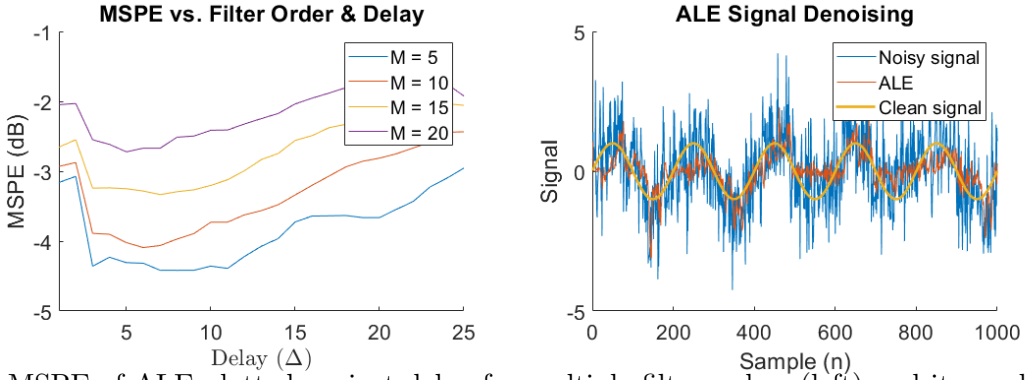


Figure 24: MSPE of ALE plotted against delay for multiple filter orders (left) and its application to noisy sinusoid generated (right).

Part b)

The same procedure for the ALE system was repeated for 100 realisations generating 1000-sample noised corruped signal $s(n)$ while varying both the delay Δ as well as the filter order M keeping $\mu_{MLS} = 0.01$. We can the results both from Figures 24 and 25. From Figure 24, we see the relationship between the delay parameter and the MSPE for different model orders and observe that for $\Delta < 3$ the MSPE has a very high value which decreases as we increase the delay and plateau from $\Delta = 3$ to $\Delta = 6$, after that the value of the error increases. This is due to the fact that as we increase the delay beyond a value of 6, the increased phase shift (delay) introduced in the ALE filter output creates larger deviations of the denoised signal $\hat{x}(n)$ from the clean signal $x(n)$. As the value of the MSPE increases, the performance of the ALE algorithm gets worse and hence we settle for an optimal delay of $\Delta = 3$. We further confirm that in the right plot of Figure 25 which reaches the lowest MSPE for a delay of 3.

From Figure 25, we can observe the left plot which shows the variations of MSPE for the optimal delay of $\Delta = 3$ and different filter order from 1 to 25. We observe that the MSPE decreases as we increase the filter order up to $M = 5$ and then increases back further. This is due to the algorithm overfitting the noise in the signal as increasing the filter order increases the complexity which results in higher error and reduces performance. From the figures, we can take our optimal parameters to be $\Delta = 3$ and $M = 5$ which gives us the best performances while remaining at a low complexity.

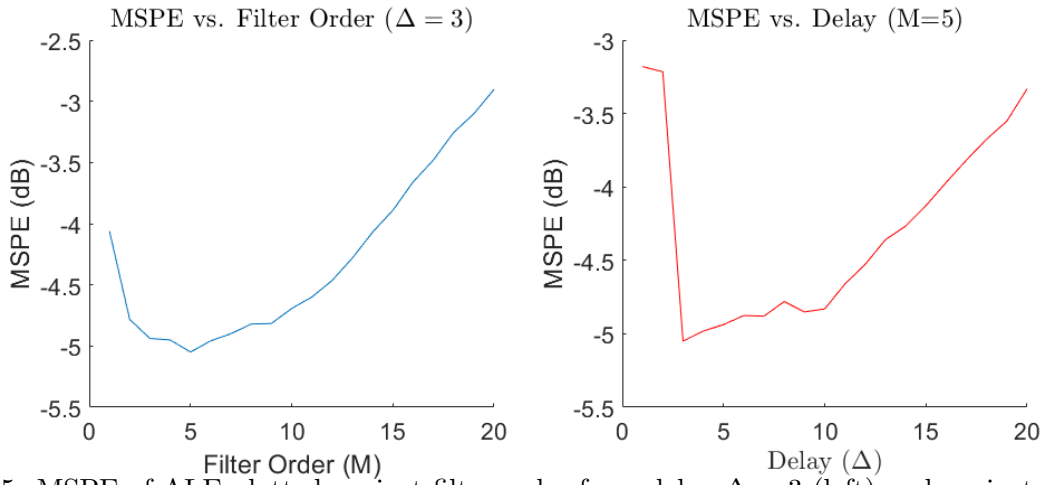


Figure 25: MSPE of ALE plotted against filter order for a delay $\Delta = 3$ (left) and against delay for a filter order $M = 5$ (right).

Part c)

In this section, we used MATLAB to implement adaptive noise cancellation as seen in Figure 26. The given configuration, a sinusoidal input was corrupted by a noise given by the equation below:

$$\eta(n) = v(n) + 0.5v(n-2) \tag{55}$$

From Figure 26we observe that while the ALE algorithm converges more rapidly to the clean signal reconstruction, after enough samples, the ANC algorithm obtains significantly smaller steady-state MSPE. This suggests that the ALE performance is constant over time and might be preferable for small sample size data, while ANC which has increasing performance over time, is better when given larger samples sizes. We can also note that the performance of ANC is dependent of the given secondary noise signal, which must be sufficiently linearly correlated to the noise of the original data which might be difficult to obtain in certain cases.

Part d)

In this section, we apply the ANC method to EEG recordings from the Cz electrode on the scalp which are corrupted by a 50Hz noise components. In order to empirically determine the parameter values of the 50Hz component in EEG data, we vary the filter order M and the step-size μ. This procedure will not affect the other frequency components. The white noise was chosen to be $w(n) \sim \mathcal{N}(0, 0.1^2)$ and the spectograms have window length 3550 and overlap ratio of 0.3. We see the spectogram plots for varying M and μ values in Figure 27:

As we can observe from the Figure 27, the order filter is used to determine the amount of denoising done to the EEG data for a fixed step-size μ. Lower filter orders are not able to successfully remove the noise at 50Hz due to underfitting the signal. Increasing the filter order to 5, we can suppress the noise component in the spectrum with no effect on other components for a large enough step size. However, increasing that filter order too much, not only remove the noise but also the neighbor components nearby. Changing the step-size μ also affects the ANC performance as seen in Figure 27. For step-size that are too small, the performance is poor with slow convergence speed and the noise component cannot be removed. On the other hand, a step size that is too large prevent to obtain the optimal solution and result in an distorted signal. Looking at the plots, we choose a middle ground with a filter order of $M = 5$ and a step size of $\mu = 0.01$ which removes the noise component without affecting the other spectral components.

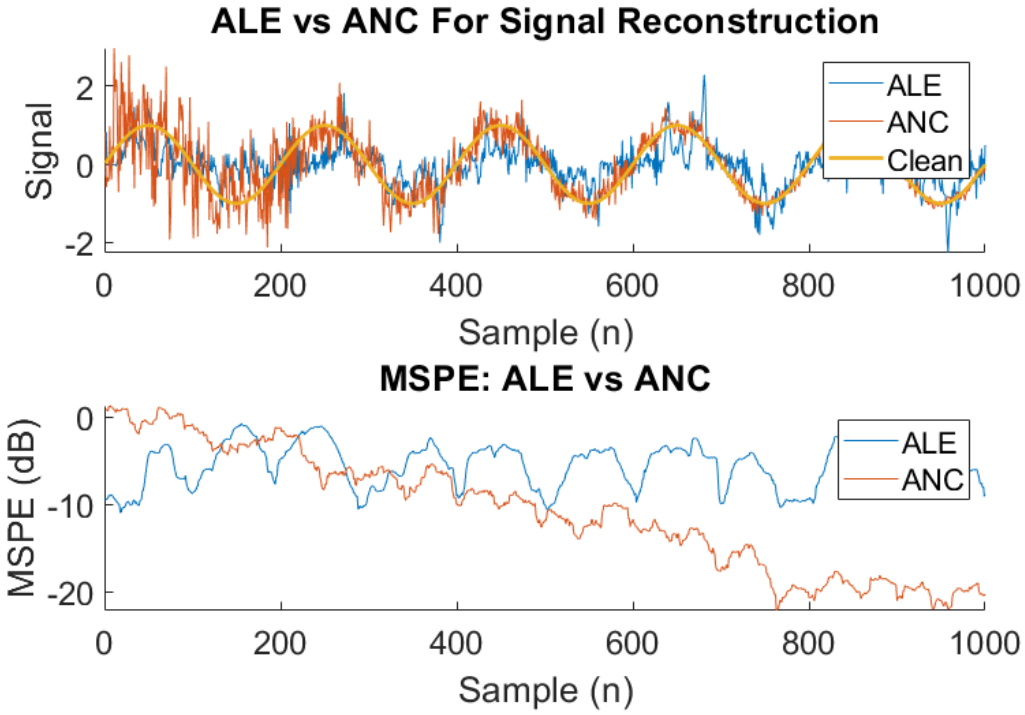


Figure 26: Clean signal reconstruction from the ALE and ANC systems (top) and comparison of the MSPE between the two systems (bottom).

Comparing our denoised signal to the original EEG signal in Figure 28, we can observe that the 50Hz components are succesffully suppressed and our choice of parameters works great.

3 Widely Linear Filtering and Adaptive Spectrum Estimation

3.1 Complex LMS and Widely Linear Modeling

Part a)

In this section, we will study the pivotal role that complex-valued signals play in communications. We first, generate a first-order widely linear moving-average process, WLMA(1), in equation 56:

$$y(n) = x(n) + b_1 x(n-1) + b_2 x^*(n-1) \tag{56}$$

with $x \sim \mathcal{N}(0,1), b_1 = 1.5 + 1j$ and $b_2 = 2.5 - 0.5j$. The real and imaginary parts of the WGN and WLMA(1), respectively, are plotted in Figure 29. We can calculate the circularity coefficients ρ, as described in equation 57:

$$\rho = \frac{E[zz^*]}{E[zz^T]} \tag{57}$$

We find the coefficient $\rho_x = 0.036$ for the WGN, verifying the circularity of the process, and $\rho_y = 0.856$ which shows that the WLMA(1) process is non-circular as expected. We were then tasked to implement the complex LMS (CLMS) and the augmented complex LMS (ACLMS) to find estimates of the parameters b_1 and b_2. The learning curves, $10log|e(n)|^2$, for CLMS and ACLMS algorithm are shown in the top plot of Figure 29, which are obtained by plotting an ensemble average of the learning curve of 100 independent realisations of process with 1000 data samples.

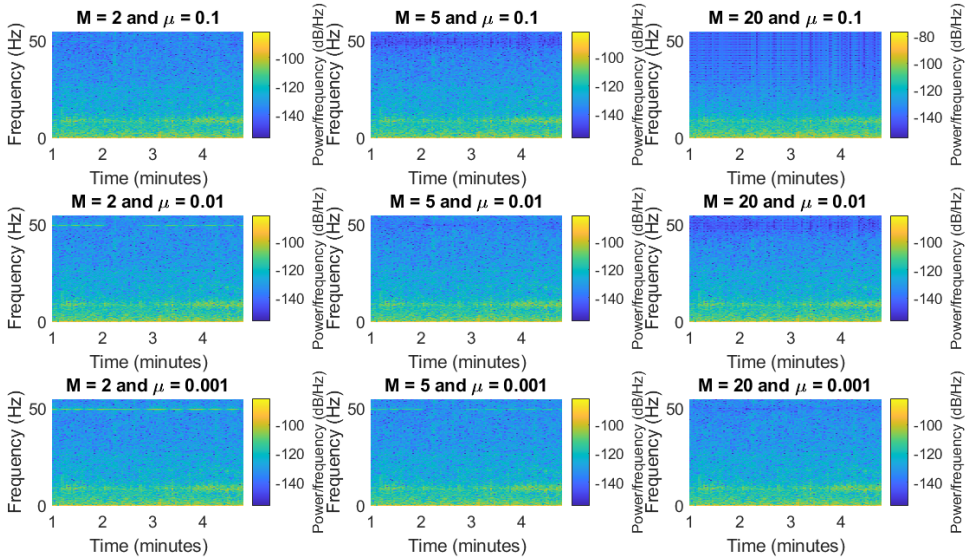


Figure 27: Plots of spectogram of ANC denoised signal for different filter orders M and step-sizes μ.

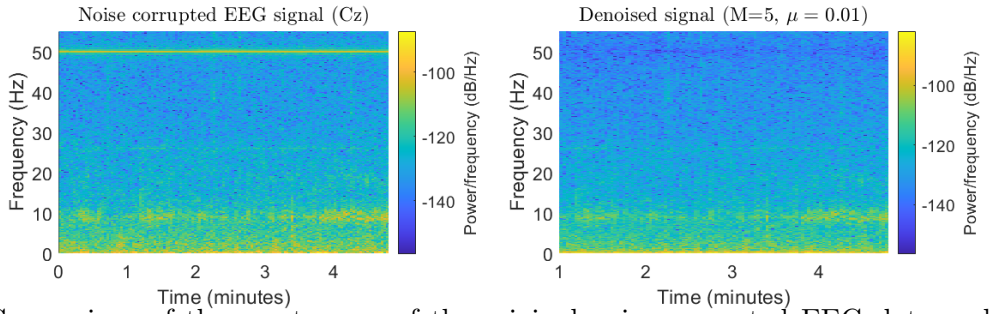


Figure 28: Comparison of the spectogram of the original noise corrupted EEG data and the ANC denoised using filter order $M = 5$ and step-size $\mu = 0.01$.

In general, the CLMS approach is only valid for circular random variables, therefore it is expected that the CLMS algorithm will be unable to model successfully the WLMA(1) process. On the other hand, the ACLMS algorithm is valid for general complex data, both circular and non-circular, which means that it is expected to successfully learn the process parameters. This is verified by Figure 3.2, where ACLMS algorithm performs much better than CLMS approach, achieving an steady-state MSPE of -305.78 dB, compared to -8.75 dB error of the CLMS algorithm.

The difference is due to the fact that the ACLMS algorithm exploits the full statistical information from the complex representation of data, which guarantees to capture the complete second-order statistics of in the data. Therefore, Figure 29 verifies that ACLMS is more suitable for identifying non-circular processes. On the other hand, CLMS can successfully identity circular data only, since it considers only the covariance matrix of data and not the pseudo-covariance matrix for noncircular data (pseudo-covariance is zero for proper (circular) complex random variables, but has lower computational complexity than ACLMS algorithm).

Part b)

In this section, We were tasked with applying the CLMS and ACLMS algorithms to the bivariate wind data in two directions, East-West v_{east} and North-South v_{north}, at three different wind speeds: low, medium and high. The bivariate data was modelled as a complex-valued wind signal, as in equation 58:

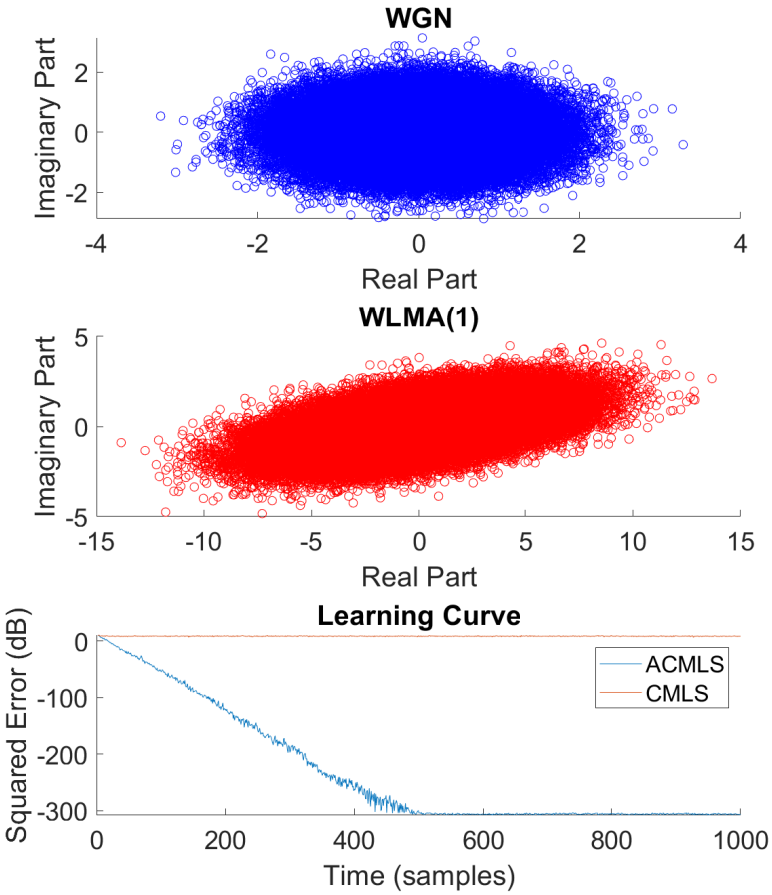


Figure 29: Circularity plots of $x(n)$, $y(n)$, and learning curve for ACMLS and CMLS algorithms ($\mu = 0.1$)

$$v[n] = v_{east}[n] + jv_{north}[n] \tag{58}$$

Once in the complex form, we determined the circularity coefficients at each wind speed to be $\rho_{low} = 0.159$, $\rho_{medium} = 0.454$ and $\rho_{high} = 0.623$. This implies that the low regime can be treated as approximately proper, whereas the medium and high wind speed data are improper. This is further reflected by the circularity plots shown in Figure 30 below.

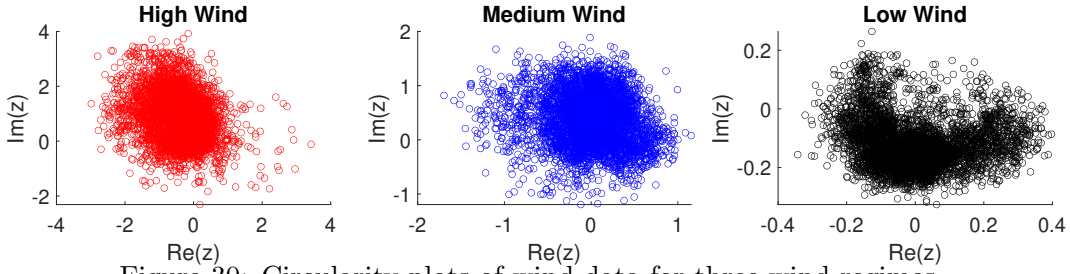


Figure 30: Circularity plots of wind data for three wind regimes.

The CLMS and ACLMS filters were used in a prediction setting to perform a one-step ahead prediction of the complex wind data for each regime. The optimal learning rates were determined empirically to be $\mu_{high} = 0.001$, $\mu_{medium} = 0.01$ and $\mu_{low} = 0.1$, as to minimize prediction error and

maintaining the filter stable. A grid-search was carried out for model orders $M \in [1:1:20]$ and the MSPE for different filters/regimes is shown in Figure 31. By considering the minimum MSPE error achieved by the two algorithms, for CLMS and ACLMS, we can observe that the the widely linear ACLMS outperformed standard CLMS, for all three regimes due to the non-circular nature of the wind data. Specifically, for more non-circular data such us the high-speed wind data with large variation in their dynamics, the ACLMS algorithm significantly outperformed the CLMS algorithm. The increase in performance of prediction by ACLMS, however, decreases as the degree of circularity within the signal increases.

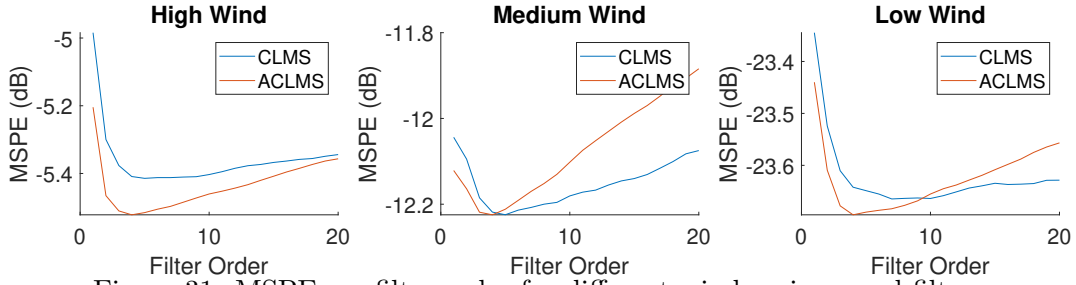


Figure 31: MSPE vs. filter order for different wind regimes and filters

Part c)

In this section, we were tasked to estimate the frequency in three phase power systems, composed of three-phase voltages v_a, v_b and v_c with magnitudes $V_a(n)$, $V_b(n)$ and $V_c(n)$ as well as phase distortions relative to the balanced system Δ_b and Δ_c. Using the Clarke transform, these can be projected onto v_0, v_α and v_β. Under balanced conditions, all voltages have the same, constant amplitude, V, and have a phase shift of exactly $\frac{2\pi}{3}$ If balanced, v_0 becomes 0 and the α and β components can be represented as a complex Clarke voltage, $v(n) = v\alpha(n) + jv\beta(n)$. The balanced Clarke voltage is given by equation 59, whereas if unbalanced, it is given by equation 60:

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

$$v(n) = A(n) e^{j(2\pi \frac{f_0}{f_s} n + \phi)} + B(n) e^{-j(2\pi \frac{f_0}{f_s} n + \phi)} \tag{60}$$

where $A(n) = \frac{\sqrt{6}}{6}(V_a(n) + V_b(n)e^{j\Delta_b} + V_c(n)e^{j\Delta_c})$ and $B(n) = \frac{\sqrt{6}}{6}(V_a(n) + V_b(n)e^{-j(\Delta_b + \frac{2\pi}{3})} + V_c(n)e^{-j(\Delta_c - \frac{2\pi}{3})})$. We were asked to generate balanced and unbalanced voltage systems (Clarke voltages). Three voltages were generated ($f_s = 480Hz$): one balanced, one unbalanced due to differences in magnitude (ΔV), and one unbalanced due to phase distortions ($\Delta\phi$). The circularity coefficients were determined to be $\rho_{balanced} = 0$, $\rho\Delta V = 0.528$ and $\rho\Delta\phi = 0.661$ respectively. The circularity plot of these three systems are shown in Figure 32, where only the balanced system is circular. Figure 32 also shows a monotonic increase circularity as the system becomes more unbalanced, which means that circularity diagrams can be used to visualise how faulty (unbalanced) a system is.

Part d)

In this section, we are going to use the strictly linear and widely linear autoregressive models of order 1 described in equations 61 and 62 to compute the frequency f_0 of the balanced and unbalanced complex $(\alpha - \beta)$ voltages $v(n)$:

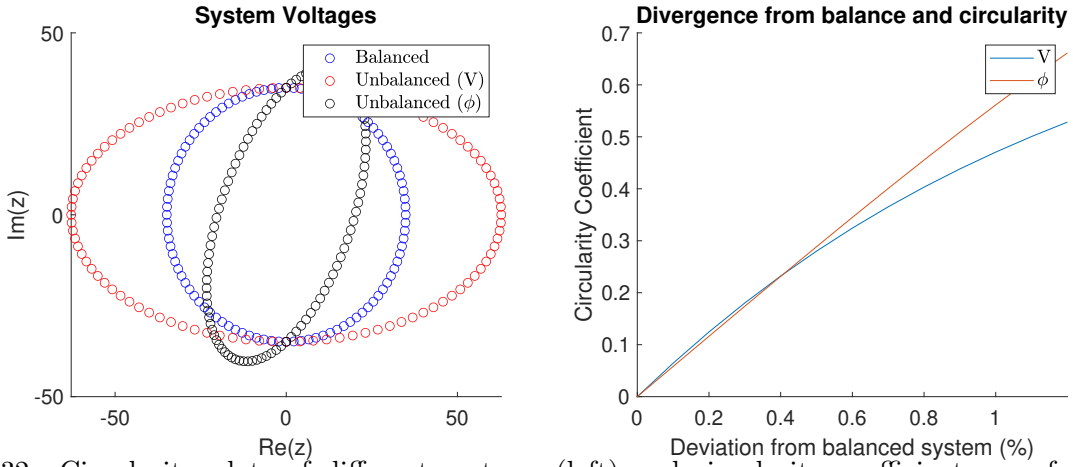


Figure 32: Circularity plots of different systems (left) and circularity coefficient as a function of deviations from the system (modulating Δ_b and V_a with other parameters constant)

$$\text{Strictly linear: } v(n+1) = h^*(n)v(n) \tag{61}$$

$$\text{Widely linear: } v(n+1) = h^*v(n) + g^*(n)v^*(n) \tag{62}$$

In the case of a balanced system, we get that the Clarke voltage $v(n)$ is given by equation 63:

$$v(n+1) = \sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_0}{f_s}(n+1)+\phi)} \tag{63}$$

where f_s is the sampling frequency, ϕ is the phase shift and n is the time index. By substituting Equation 63 into the strictly linear AR(1) model in Equation 61 we obtain:

$$\sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_0}{f_s}(n+1)+\phi)} = h^*(n)\sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_0}{f_s}n+\phi)} \tag{64}$$

$$e^{j(2\pi \frac{f_0}{f_s}(n+1)+\phi)} = h^*(n)e^{j(2\pi \frac{f_0}{f_s}n+\phi)} \tag{65}$$

$$e^{j(2\pi \frac{f_0}{f_s}n+\phi)}e^{j(2\pi \frac{f_0}{f_s})} = h^*(n)e^{j(2\pi \frac{f_0}{f_s}n+\phi)} \tag{66}$$

$$e^{j(2\pi \frac{f_0}{f_s})} = h^*(n) \tag{67}$$

Using that result, we can define $h(n) = |h(n)|e^{j\phi h}$ with its conjugate $h^*(n) = |h(n)|e^{-j\phi h}$, or equivalently $h(n) = |h(n)|e^{-j\phi h}$ with $h^*(n) = |h(n)|e^{j\phi h}$. Since the negative sign in the phase of h(n) only indicates the direction of rotation of the signal, we can choose any of the two definitions for $h(n)$. For this derivation the second definition is used so that we arrive at a positive expression for frequency f_0. We can also express $h^*(n)$, in terms of of real and imaginary components of $h(n)$ such that:

$$h^*(n) = e^{j\times \arctan\left(\frac{\Re\{h(n)\}}{\Im\{h(n)\}}\right)} \tag{68}$$

If we replace the value found in equation 68 into equation 67, we obtain:

$$e^{j(2\pi \frac{f_0}{f_s})} = |h(n)|e^{j\times \arctan\left(\frac{\Re\{h(n)\}}{\Im\{h(n)\}}\right)} \tag{69}$$

Since two complex values are equal only if their magnitudes are equal, then we know that $|h(n)| = 1$ since magnitude of exponential is 1, and hence:

$$\frac{2\pi f_0(n)}{f_s} = \arctan\left(\frac{\Re\{h(n)\}}{\Im\{h(n)\}}\right) \tag{70}$$

$$f_0(n) = \frac{f_s}{2\pi}\arctan\left(\frac{\Re\{h(n)\}}{\Im\{h(n)\}}\right) \tag{71}$$

which gives the desired result. It should be noted that f_0 is a function of n since right-hand side of equation 71 is also a function of n and all other terms are constants.

In a similar way, in the case of an unbalanced system, we use the general form of the complex-valued voltage that arises from the Clarke Transform given in equation 60. In the equation, we get that $V_a(n)$, $V_b(n)$ and $V_c(n)$ are the peak values of the three-phase voltages of the power system. Substituting the right side of that equation into the widely linear AR(1) model definition from equation 62, we obtain:

$$v(n+1) = A(n+1)e^{j(2\pi\frac{f_0}{f_s}n+\phi)} + B(n+1)e^{-j(2\pi\frac{f_0}{f_s}n+\phi)} \tag{72}$$

$$v(n+1) = h^*(n)A(n)e^{j(2\pi\frac{f_0}{f_s}n+\phi)} + h^*B(n)e^{-j(2\pi\frac{f_0}{f_s}n+\phi)} + g^*(n)A^*(n)e^{-j(2\pi\frac{f_0}{f_s}n+\phi)} + g^*B^*(n)e^{j(2\pi\frac{f_0}{f_s}n+\phi)} \tag{73}$$

Equating equations 72 and 73, we obtain:

$$\begin{aligned} &A(n+1)e^{j(2\pi\frac{f_0}{f_s}(n+1)+\phi)} + B(n+1)e^{-j(2\pi\frac{f_0}{f_s}(n+1)+\phi)} = \\ &h^*(n)A(n)e^{j(2\pi\frac{f_0}{f_s}n+\phi)} + h^*B(n)e^{-j(2\pi\frac{f_0}{f_s}n+\phi)} + g^*(n)A^*(n)e^{-j(2\pi\frac{f_0}{f_s}n+\phi)} + g^*B^*(n)e^{j(2\pi\frac{f_0}{f_s}n+\phi)} \end{aligned} \tag{74}$$

$$\begin{aligned} &A(n+1)e^{j(2\pi\frac{f_0}{f_s}n+\phi)}e^{j(2\pi\frac{f_0}{f_s})} + B(n+1)e^{-j(2\pi\frac{f_0}{f_s}n+\phi)}e^{-j(2\pi\frac{f_0}{f_s}} = \\ &h^*(n)A(n)e^{j(2\pi\frac{f_0}{f_s}n+\phi)} + h^*B(n)e^{-j(2\pi\frac{f_0}{f_s}n+\phi)} + g^*(n)A^*(n)e^{-j(2\pi\frac{f_0}{f_s}n+\phi)} + g^*B^*(n)e^{j(2\pi\frac{f_0}{f_s}n+\phi)} \end{aligned} \tag{75}$$

By factorising equation 75, and collecting common exponential terms, i.e. separating terms associated with $e^{j(2\pi\frac{f_0}{f_s}n+\phi)}$ and the terms associated with $e^{-j(2\pi\frac{f_0}{f_s}n+\phi)}$, we can obtain the following relations:

$$A(n+1)e^{j2\pi\frac{f_0}{f_s}} = h^*(n)A(n) + g^*(n)B^*(n) \tag{76}$$

$$B(n+1)e^{-j2\pi\frac{f_0}{f_s}} = h^*(n)B(n) + g^*(n)A^*(n) \tag{77}$$

If we assume that the three-phase voltage magnitudes $V_a(n)$, $V_b(n)$ and $V_c(n)$ in equation 60 are constant with time or their rate of change with time is negligible, then we can set $A(n+1) \approx A(n)$ and $B(n+1) \approx B(n)$, such that:

$$A(n)e^{j2\pi\frac{f_0}{f_s}} = h^*(n)A(n) + g^*(n)B^*(n) \Rightarrow e^{j2\pi\frac{f_0}{f_s}} = h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} \tag{78}$$

$$B(n)e^{-j2\pi\frac{f_0}{f_s}} = h^*(n)B(n) + g^*(n)A^*(n) \Rightarrow e^{-j2\pi\frac{f_0}{f_s}} = h^*(n) + g^*(n)\frac{A^*(n)}{B(n)} \tag{79}$$

Now taking the conjugate of equation 79 and equating it to the value in equation 78, we obtain:

$$e^{j2\pi\frac{f_0}{f_s}} = h^*(n) + g^*(n)\left(\frac{B^*(n)}{A(n)}\right) = h(n) + g(n)\left(\frac{A^*(n)}{B(n)}\right) \tag{80}$$

Let's define $K(n) = \frac{B^*(n)}{A(n)}$ and substitute it into equation 80, we can obtain the following relation between K(n), g(n) and h(n):

$$h^*(n) + g^*(n)K(n) = h(n) + g(n)\left(\frac{1}{K(n)}\right) \tag{81}$$

$$g^*(n)K^2(n) + [h^*(n) - h(n)]K(n) - g(n) = 0 \tag{82}$$

We can solve for K in equation 82 in terms of $g(n)$ and $h(n)$ such that:

$$K(n) = \frac{[h(n) - h^*(n)] \pm \sqrt{[h^*(n) - h(n)]^2 + 4g^*(n)g(n)}}{2g^*(n)} \tag{83}$$

Since we previously defined $h(n) = |h(n)|e^{-j\phi h}$ and $h^*(n) = |h(n)|e^{j\phi h}$ then we can define $h(n) - h^*(n) = -2j\Im\{h(n)\}$ and $(h^*(n) - h(n))^2 = -4\Im^2\{h(n)\}$. Finally, we define $g^*(n)g(n) = |g(n)|^2$ and we use these to simplify equation 83 to:

$$K(n) = \frac{-2j\Im\{h(n)\} \pm \sqrt{-4\Im^2\{h(n)\} + 4|g(n)|^2}}{2g^*(n)} \tag{84}$$

$$K(n) = j\frac{-\Im\{h(n)\} \pm \sqrt{\Im^2\{h(n)\} - |g(n)|^2}}{g^*(n)} \tag{85}$$

Substituting the value of K found in equation 85 into equation 80, we obtain:

$$e^{j2\pi\frac{f_0}{f_s}} = h^*(n) + jg^*(n)\frac{-\Im\{h(n)\} \pm \sqrt{\Im^2\{h(n)\} - |g(n)|^2}}{g^*(n)} \tag{86}$$

$$= h^*(n) - j\Im\{h(n)\} \pm j\sqrt{\Im^2\{h(n)\} - |g(n)|^2} \tag{87}$$

$$= \Re\{h(n)\} \pm j\sqrt{\Im^2\{h(n)\} - |g(n)|^2} \tag{88}$$

where $\Re\{h(n)\} = h^*(n) - j\Im\{h(n)\}$. We can express the right side of equation 88 into exponential form given by:

$$\Re\{h(n)\} \pm j\sqrt{\Im^2\{h(n)\} - |g(n)|^2} = Ce^{j\arctan\left(\frac{\pm j\sqrt{\Im^2\{h(n)\} - |g(n)|^2}}{\Re\{h(n)\}}\right)} \tag{89}$$

where C is a real constant, with value of 1 in order for equality to hold. Since the sign of phase only indicated direction of rotation, as previously mentioned, we can choose positive sign in the argument of arctan. By equating phases of both terms in equation 89, we get:

$$\frac{2\pi f_0(n)}{f_s} = \arctan\left(\frac{\Im^2\{h(n)\} - |g(n)|^2}{\Re\{h(n)\}}\right) \tag{90}$$

$$f_0(n) = \frac{f_s}{2\pi}\arctan\left(\frac{\Im^2\{h(n)\} - |g(n)|^2}{\Re\{h(n)\}}\right) \tag{91}$$

Equation 91 gives us the desired result. Since the right side of the equation is a function of n, then the function f_0 is also a function of n.

Part e)

In this section, the CLMS and ACLMS filters were used to estimate the frequency of the generated balanced and unbalanced voltages ($f_0 = 50Hz$) using equations 71 and 91. We used the same procedure for the three systems presented in part 3.1.c with $f_s = 480Hz$. The results are shown in Figure 33. A grid-search was carried out to determine the optimal μ to prevent unstable learning, which was found to be $\mu = 10^{-5}$. As expected, while CLMS and ACLMS are both able to learn the appropriate coefficients in order to estimate the frequency accurately of a balanced system (second-order circular), only the ACLMS converges to the correct frequency for unbalanced systems. Additionally, only ACLMS is able to overcome oscillation errors after converging for unbalanced systems.

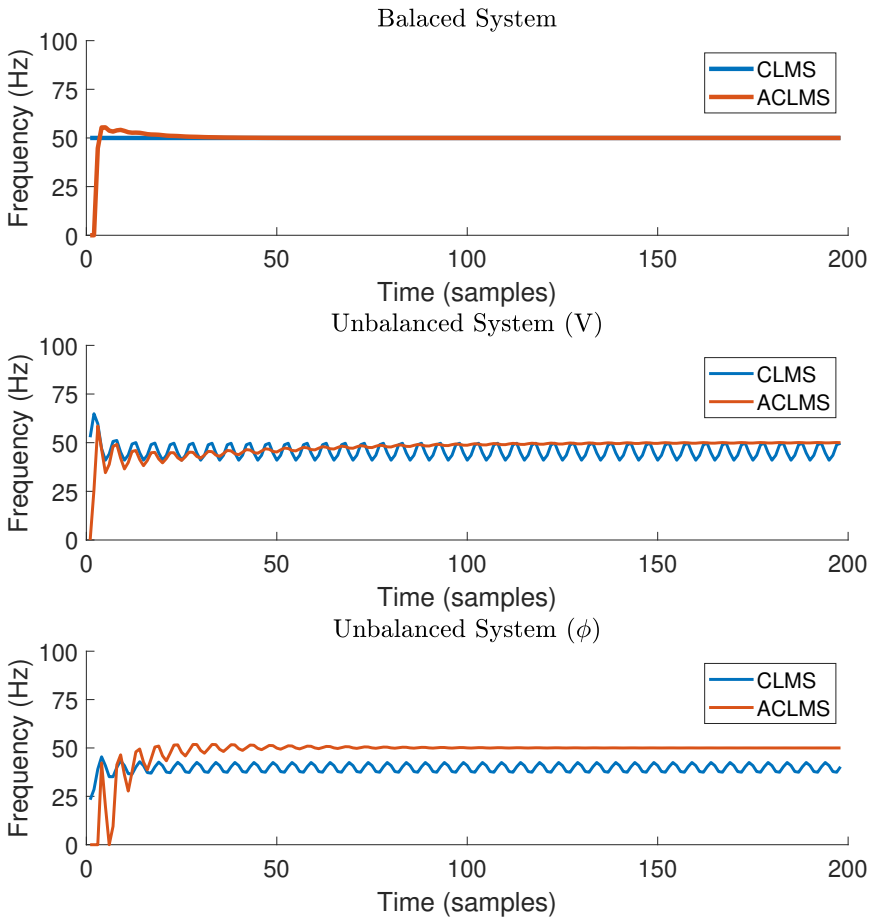


Figure 33: Frequency estimation using CLMS and ACLMS algorithms for a balanced system, and two unbalanced systems (magnitude and phase distortions)

3.2 Adaptive AR Model Based Time-Frequency Estimation

Part a)

In this section, we generated a frequency modulated (FM) signal $y(n) = e^{j\left(\frac{2\pi}{f_s}\phi(n)\right)+\eta(n)}$, where $\eta(n)$ is circular complex-valued white noise with zero mean and variance $\sigma_\eta^2 = 0.05$ and the phase $\phi(n) = \int f(n)dn$ is generated as:

$$f(n) = \frac{d\phi(n)}{dn} = \begin{cases} 100, & 1 \leq n \leq 500 \\ 100 + \frac{n-500}{2}, & 501 \leq n \leq 100 \\ 100 + \left(\frac{n-1000}{25}\right)^2, & 1001 \leq n \leq 1500 \end{cases} \tag{92}$$

The signal has a frequency response generated as in Figure 34. The `aryule` MATLAB function was used to find the AR(1) coefficient for the signal, as well as to compare the AR(1) estimate with other model orders. The power spectral estimates obtained are displayed in Figure 34. As the autoregressive spectral estimation assumes stationarity, it is not able to capture the change in frequency for any model order and instead finds the optimal stationary representation, capturing the average frequency content. This is reflected by the AR(1) peak being at the average frequency of the FM signal (FM_{avg}).

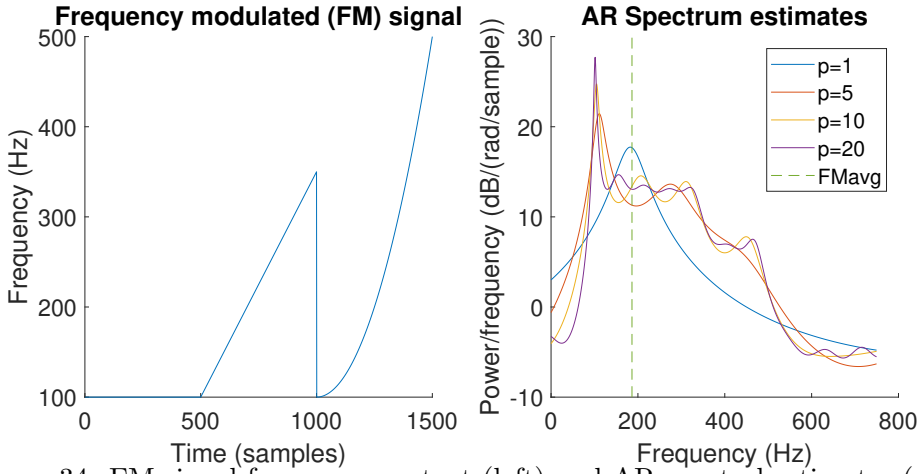


Figure 34: FM signal frequency content (left) and AR spectral estimates (right)

Part b)

In this section,we implemented the CLMS algorithm to estimate the AR coefficients of $y(n)$ in Part 3.2.a at each time instant, and computed the frequency spectrum of the signal. The time-frequency spectrum was obtained by using the `mesh` MATLAB function, and is displayed in Figure 35. The CLMS is algorithm is able to successfully capture the changes of frequency over time. However, it is important to choose an appropriate learning rate to ensure learning the correct coefficients within the given interval. We can see that this is not achieved when using $\mu = 0.001$ as shown in Figure 35.

3.3 A Real-Time Spectrum Analyser Using Least Mean Square

Part a)

We can estimate a signal $y(n)$ as a linear combination of N harmonically related sinusoids as described in equation 93:

$$\hat{y}(n) = \sum_{k=0}^{N-1} w(k) e^{j\frac{2\pi kn}{N}} \tag{93}$$

We can express this expression in vector form as described in equation 94:

$$\hat{\mathbf{y}} = \mathbf{F}\mathbf{w} \tag{94}$$

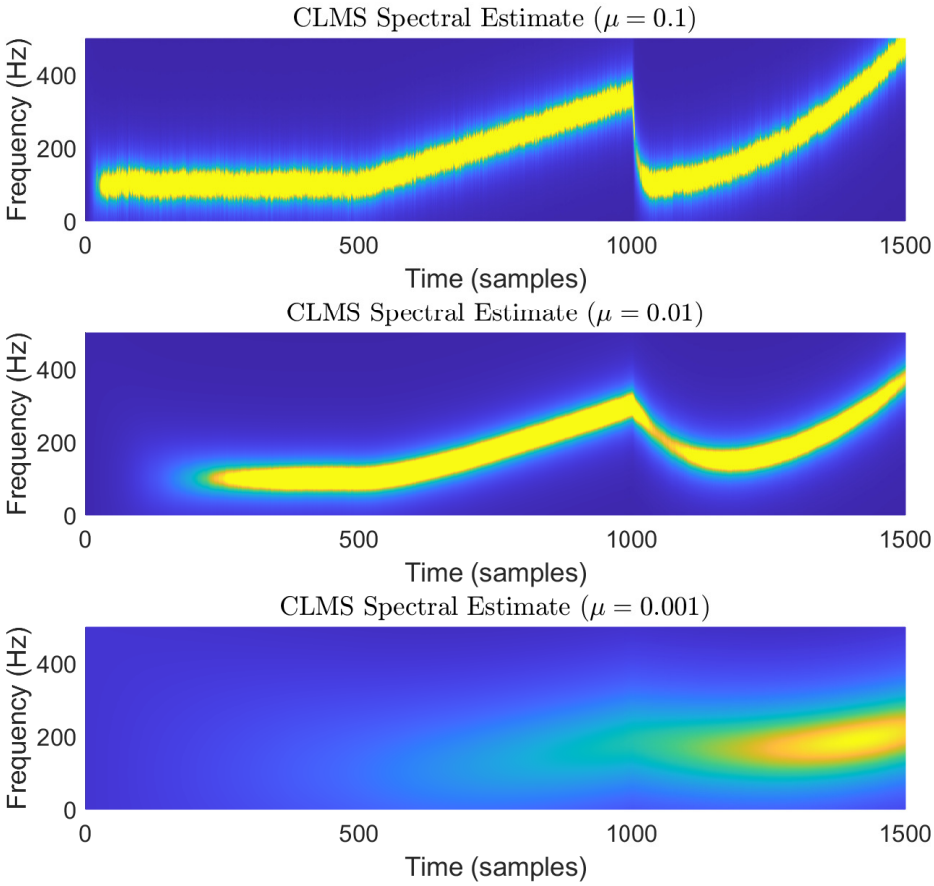


Figure 35: Time-frequency spectra of FM signal for different learning rates

We can obtain the optimal weights $\mathbf{w}$ by minimising the sum of squared errors between the estimate and true signal.

$$\min_{\mathbf{w}} \|\mathbf{y} - \hat{\mathbf{y}}\|^2 = \min_{\mathbf{W}} \sum_{n=1}^{N-1} \|y(n) - \hat{y}(n)\|^2 \tag{95}$$

$$= \min_{\mathbf{w}} (\mathbf{y} - \mathbf{F}\mathbf{w})^H (\mathbf{y} - \mathbf{F}\mathbf{w}) \tag{96}$$

$$= \min_{\mathbf{w}} (\mathbf{y}^H\mathbf{y} - \mathbf{w}^H\mathbf{F}^H\mathbf{y} - \mathbf{y}^H\mathbf{F}\mathbf{w} + \mathbf{w}^H\mathbf{F}^H\mathbf{F}\mathbf{w}) \tag{97}$$

$$= \min_{\mathbf{w}} (\mathbf{y}^H\mathbf{y} - 2\mathbf{w}^H\mathbf{F}^H\mathbf{y} + \mathbf{w}^H\mathbf{F}^H\mathbf{F}\mathbf{w}) \tag{98}$$

We can simplify in equation 98 as the two centers terms were scalars and each other's transposes. We can now obtain the gradient such as:

$$\nabla_{\mathbf{w}} \|\mathbf{y} - \hat{\mathbf{y}}\|^2 = -2\mathbf{F}^H\mathbf{y} + 2\mathbf{F}^H\mathbf{F}\mathbf{w} = 0 \tag{99}$$

We can simplify equation 99 to get the optimal weights in the least square sense as:

$$\mathbf{w}_{opt} = (\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H\mathbf{y} \tag{100}$$

We find that matrix $\mathbf{F}$ is full-rank, since all its column vectors are orthogonal. As a result, matrix $\mathbf{F^H F}$ is also full-rank and is symmetric diagonal with positive diagonal elements, ensuring that its

determinant is non-zero and thus it is invertible. We also know that the definition of Inverse Discrete Fourier Transform (IDFT) is a scaled version of the definition 93 and hence we can interpret the set of optimal weights $\mathbf{w}_{opt}$ to be the DFT coefficients.

Part b)

As described in equation 94, we must multiply $\mathbf{F}$ by $\mathbf{w}$ in order to obtain the estimate of our signal $\hat{\mathbf{y}}$. This means that the obtained estimate must be in the subspace spanned by the columns of $\mathbf{F}$, as these are the basis vectors, and hence belong the the column space denoted $C(\mathbf{F})$. We conclude that this estimate can be treated as the projection of vector $\mathbf{y}$ onto the column space $C(\mathbf{F})$ which minimises the sum of squared errors. We can verify that by using the value of the sets of optimal weights obtained in equation 100 and use it in equation 94 to obtain equation 101:

$$\hat{\mathbf{y}} = \mathbf{F}(\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H\mathbf{y} \tag{101}$$

Finally, we can also define the projection matrix $\mathbf{P}$ and rewrite equation 101 including it:

$$\mathbf{P} = (\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H \tag{102}$$

$$\hat{\mathbf{y}} = \mathbf{P}\mathbf{y} \tag{103}$$

Part c)

Similarly to what we have previously done in section 3.2.a, we treat the DFT as a least square solution. Doing so, we are able to implement an adaptive version of the CLMS algorithm in which the input of the filter at time n is given by:

$$\mathbf{x}(n) = \frac{1}{N}\left[1, \quad e^{j\frac{2n\pi}{N}}, \quad e^{j\frac{4n\pi}{N}} \quad \cdots, \quad e^{j\frac{2n(N-1)\pi}{N}}\right]^T \tag{104}$$

Doing so, we are able to get a time-frequency estimation in the same way as wtih previous AR estimates. We then adapted the CLMS algorithm into a DFT-CLMS algorithm and used it on the FM signal mentioned in section 3.2.b as shown in Figure 36:

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

Part d)

In this section, we implemented the DFT-CLMS algorithm defined in Part 3.3.c onto the EEG Signal PO_z defined in Part 1.2. In order to reduce the computational complexity, the range $n \in [3000 : 4199]$ was used (for a signal length of 1200). Given that the signal is stationary, introducing a γ parameter is not necessary. The sampling rate used was $f_s = 1200Hz$ and the signal was detrended before the estimates. The time-frequency diagram obtained is shown in Figure 37. After approximately 400 samples of learning, the $50Hz$ component introduced by mains and the low frequency component at $3Hz$ become visible and exhibit a strong response by the end of the segment. Another frequency component exhibits a weak response near $9Hz$. Whist this estimate results in less noisy spectral estimates, it exhibits greater computational complexity and requires a substantial number of observations to learn the optimal parameters.

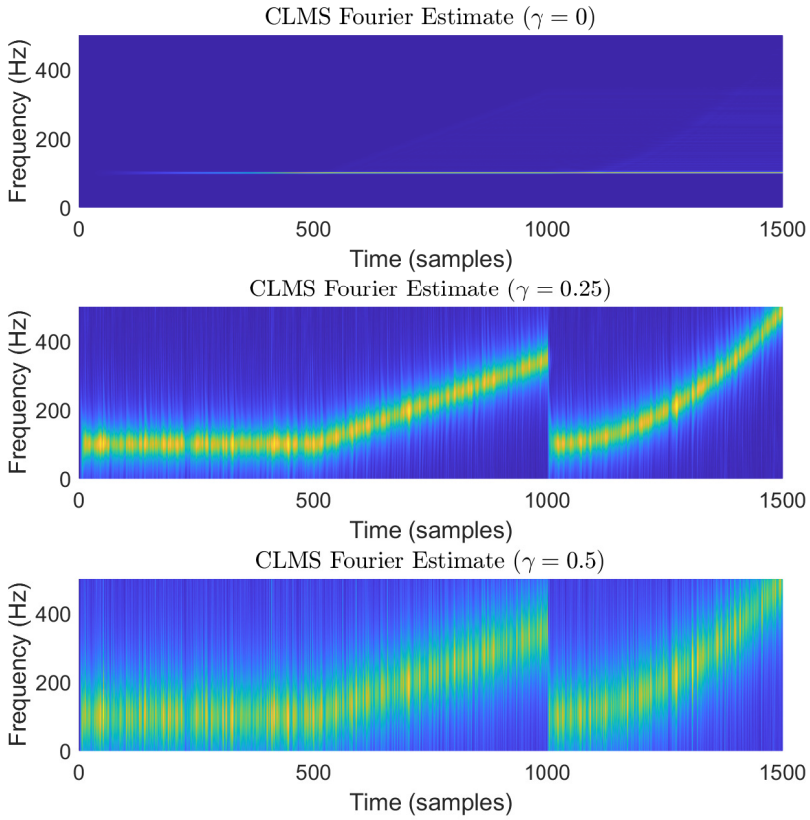


Figure 36: DFT-CLMS time-frequency diagram estimates for different γ

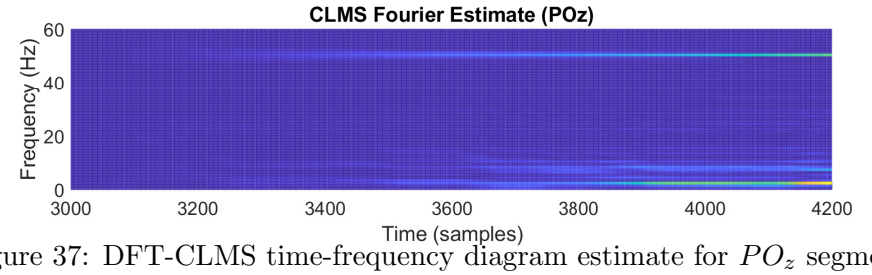


Figure 37: DFT-CLMS time-frequency diagram estimate for PO_z segment

4 From LMS to Deep Learning

4.1 LMS Time-Series Prediction

The first task was to apply the previously constructed LMS algorithm to a time-dependent, nonstationary time-series from `time-series.mat`. The mean of the series was removed and the LMS algorithm was applied with AR(4) and a step size $\mu = 10^{-5}$. The prediction of the mean-centered signal is shown in Figure 38. This figure reflects how after the first 200-300 samples, the algorithm has converged and is able to predict the next step with $MSE = 40.1$ and $R_p = 10log_{10}\left(\frac{\hat{\sigma_y^2}}{\sigma_e^2}\right) = 5.2$, where $\hat{\sigma_y^2}$ and σ_e^2 are the variances of the output and prediction error respectively. We can observe from Figure 38 that whilst the algorithm performs well for a linear model, it is unable to capture non-linearities in the signal.

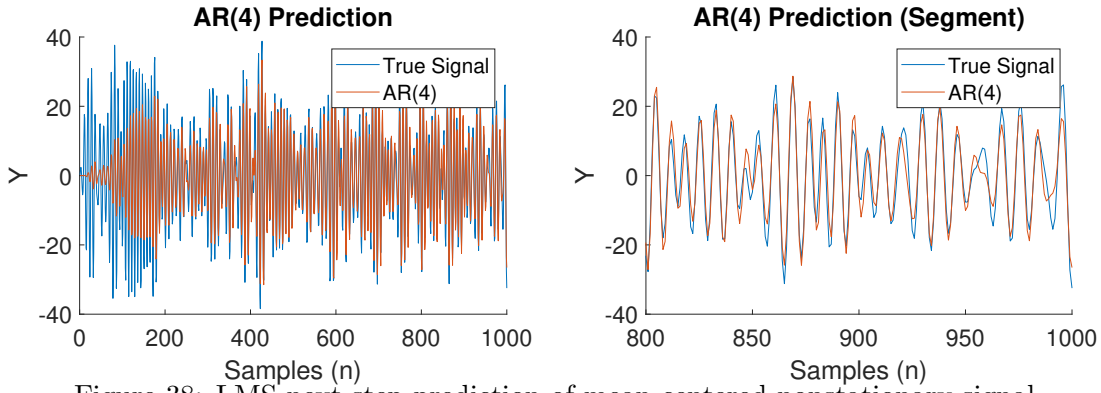


Figure 38: LMS next-step prediction of mean-centered nonstationary signal

4.2 Dynamical Perceptron

In this section, we will study a method to enable the standard LMS algorithm to capture non-linear patterns in the signal. To do so, we add a non-linearity (activation function) to the output. For this task, the `tanh` function was used in order to model the time-series from Section 4.1. As we introduce some non-linearity, the cost function has changed and hence the weight update has to. As $\tanh' = 1 - \tanh^2$, we can write the weight update formula as follows:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)[1 - \tanh^2(\mathbf{w}(n)^T \mathbf{x}(n))]\mathbf{x}(n) \tag{105}$$

We predicted the zero-mean signal using the LMS-tanh algorithm, which is a dynamical perceptron with the results being displayed in Figure 39. Whilst we are now able to capture non-linearities, because $\tanh(x) \in [-1; 1]$, the model is not able to accurately predict the signal to its magnitude, with $MSE = 208$ and $R_p = 30.64$.

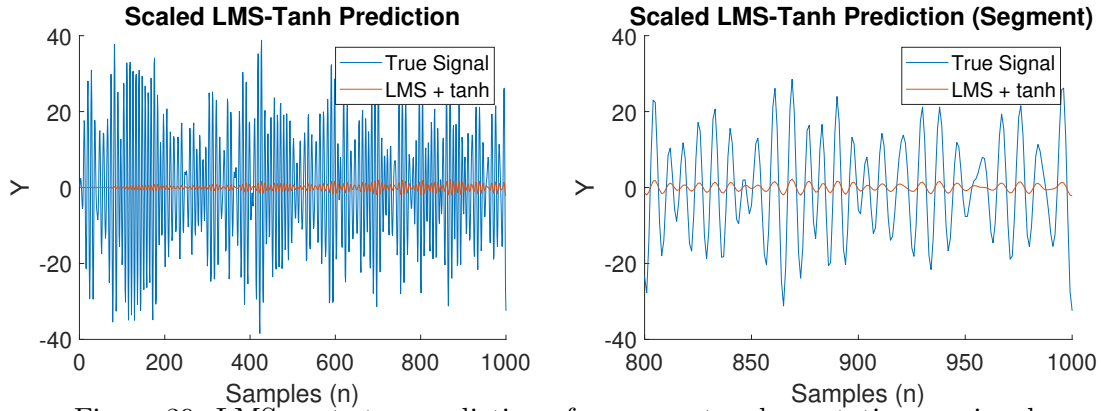


Figure 39: LMS next-step prediction of mean-centered nonstationary signal

4.3 Scaled Activation Function

In this section, we will try to improve the dynamical perceptron by scaling the `tanh` activation function. As $\tanh(x) \in [-1; 1]$, we need a scaling factor that is large enough to capture the highest amplitude peaks, yet low enough to prevent instability. Empirically, this scaling factor a, is determined to lie within the $25 \leq a \leq 40$ range. In order to determine the optimal value of this scaling factor, that we denote a^*, we use gradient descent as we previously did for the weights, with the update algorithm at time n given by equation 106. Given the change in the activation function, we also redefine the weight update formula in equation 107:

$$a(n+1) = a(n) + \mu e(n) \tanh(\mathbf{w}(n)^T \mathbf{x}(n)) \tag{106}$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + a\mu e(n)[1 - \tanh^2(\mathbf{w}(n)^T \mathbf{x}(n))]\mathbf{x}(n) \tag{107}$$

The learning rates for the weights and scaling factor were empirically chosen to be $\mu_w = 210^{-1}$ and $\mu_a = 0.1$ respectively. The scaling factor was initialised at $a_{start} = 30$, within the optimal range discussed. The results are displayed in Figure 40. The scaled LMS-Tanh algorithm significantly outperforms its non-scaled and linear counterparts, with $MSE = 11$ and $R_p = 12.6$.

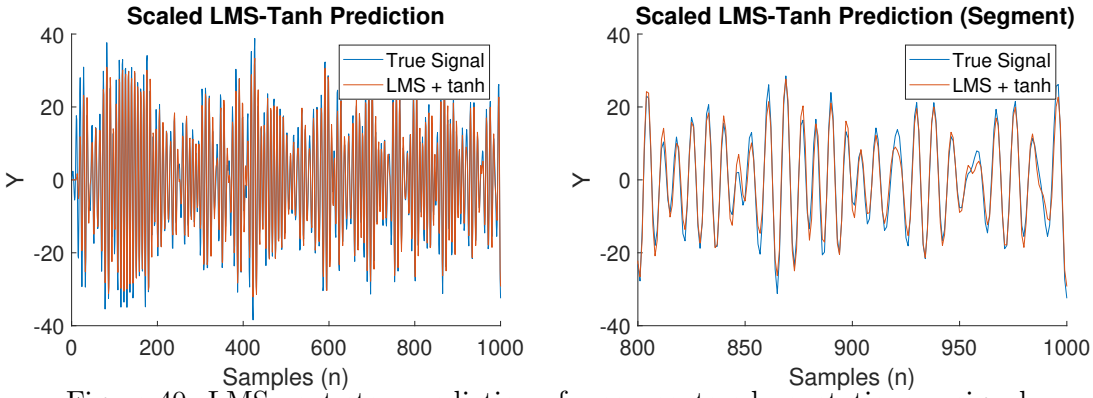


Figure 40: LMS next-step prediction of mean-centered nonstationary signal

4.4 Dynamical Perceptron Bias

In this section, we will try to further improve the LMS-tanh algorithm by adding a bias term such that the output of the model is described by the $\tanh(\mathbf{w}^T\mathbf{x}+b)$ activation function. We can implement this feature with minimal change by simply considering the augmented input $\hat{\mathbf{x}} = [1, \mathbf{x}^T]$, such that the weight always represents the bias. We applied this updated version of the algorithm with bias to the original signal with the results being displayed in Figure 41. This time however, the algorithm performs worse that the prediction of the mean-centered series, as the error is introduced during the period before the bias is fully learnt hence we obtain $MSE = 18$ and $R_p = 10.9$. However, we observe that after convergence, the algorithm exhibits similar performance to the unbiased prediction but still does not show significant improvement.

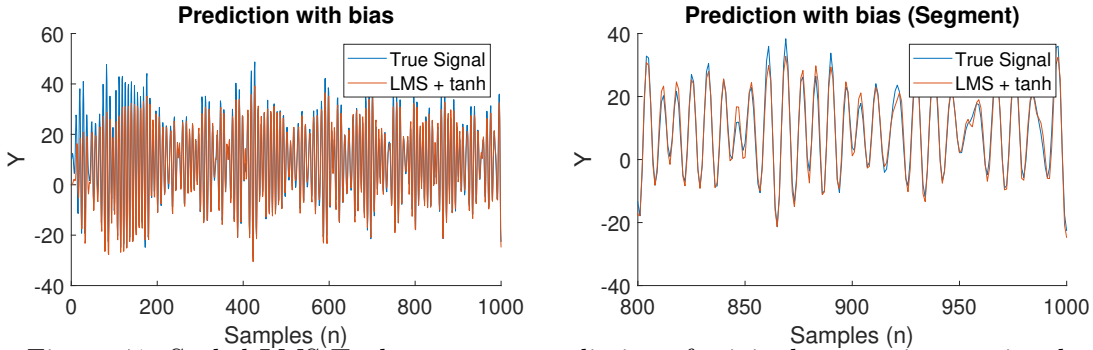


Figure 41: Scaled LMS-Tanh next-step prediction of original nonstationary signal

4.5 Pre-training Weights

Given that single weight updates are carried out per time-step, it may take a lot of samples to converge to a reasonable prediction, and with the addition of the scaling factor and bias terms, may

contribute significantly to the error. To prevent this, these parameters can pre-trained by over-fitting to a small number of samples for a more favorable initial condition. Pre-training was carried out for the scaled LMS-Tanh algorithm with a bias for the first 20 samples and 100 epochs, where $w(0) = 0$. The prediction results are shown in Figure 42. As expected, pre-training the parameters prior to prediction resulted in optimal performance, with $MSE = 5.3$ and $R_p = 16.1$, exhibiting near-ideal predictions from the start of the signal.

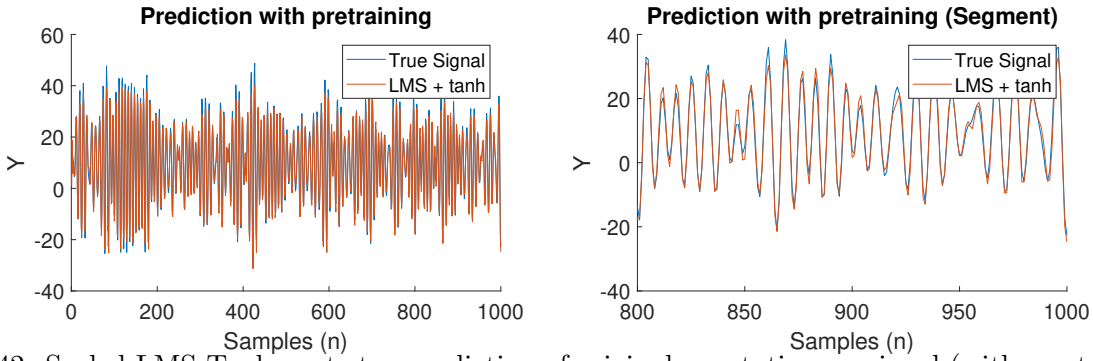


Figure 42: Scaled LMS-Tanh next-step prediction of original nonstationary signal (with pre-training)

4.6 The Backpropagation Algorithm

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

4.7 Deep Neural Network

In this section, we generated a non-linear time-series based on 10 sinusoids of deifferent frequencies, amplitudes and noises as shown in Figure 43. We then added noise to the time-series with its power equal to 0.05. The time-series was then divided into a 50% train and test split and the linear LMS algorithm, the dynamical perceptron using tanh, and a deep neural network were trained on the first half of the data and were tested on the remaining samples. The deep network was trained on 20,000 epochs, had 4 hidden layers (10,5,5,5,1), and used a learning rate of $\mu = 0.01$. The predictions are displayed in Figure 44 and reflect the deep networks ability to capture more temporal information and provide more accurate predictions, as opposed to the single neurons, whose performance is dictated by the largest amplitude frequency components of the signal. The training and test losses were obtained and computed as a function of epoch number in Figure 45, where both the single perceptron and LMS converge faster than the deep network, whereas all three algorithms exhibit similar test loss to the deep network (approximately 0.08). This is likely due to the noise introduced to the signal, where the deep network's complexity enables the model to fit both the signal and the noise.

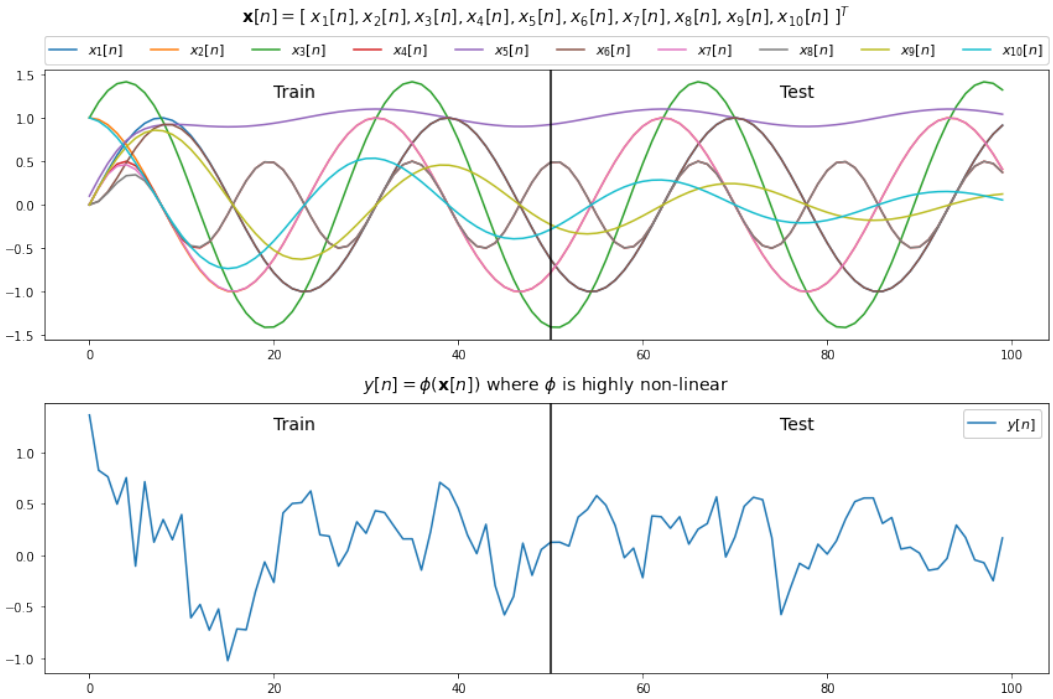


Figure 43: Sinusoids superposed to obtain non-linear time-series

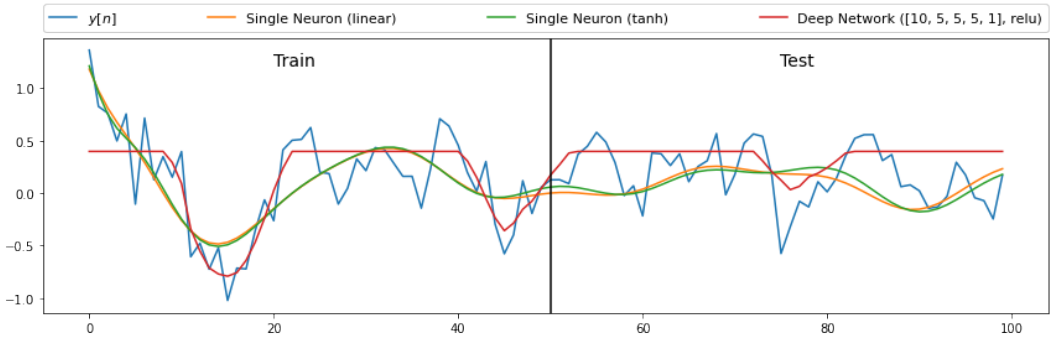


Figure 44: LMS, dynamical perceptron and deep network predictions of low noise, non-linear time-series

4.8 DNNs & Noise

Finally, we repeated the procedure in Part 4.7 for different values of noise power. While for lower noise powers, the deep network outperforms the other two algorithms due to its greater complexity, it over-fits the training data for greater noise powers, fitting the noise as well as the signal to minimise loss. This highlights a major limitation of deep networks, as it may only exhibit optimal performance on relatively clean signals. This is supported by the loss curves in Figures 46 and 47. While for low noise (power of 0), the deep network significantly outperforms the other two algorithms given sufficient training, for higher noise (power of 1), the LMS and dynamical perceptron algorithms outperform the the deep network. We also observe that whilst all three models over-fit due to the noise in the signal, because of its complexity, the deep neural network exhibits the worst generalisation out of the three models. Additionally, both the training and testing curves for the deep network exhibit instability, reflecting another issue of such a complex system being applied to noisy signals.

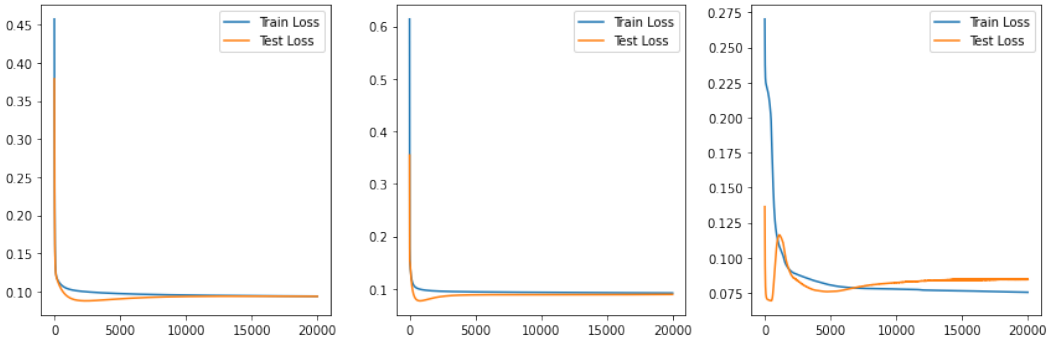


Figure 45: Training and testing loss curves for the standard LMS (left), non-linear perceptron (middle) and deep network (right) as a function of epoch count for 0.05 noise power

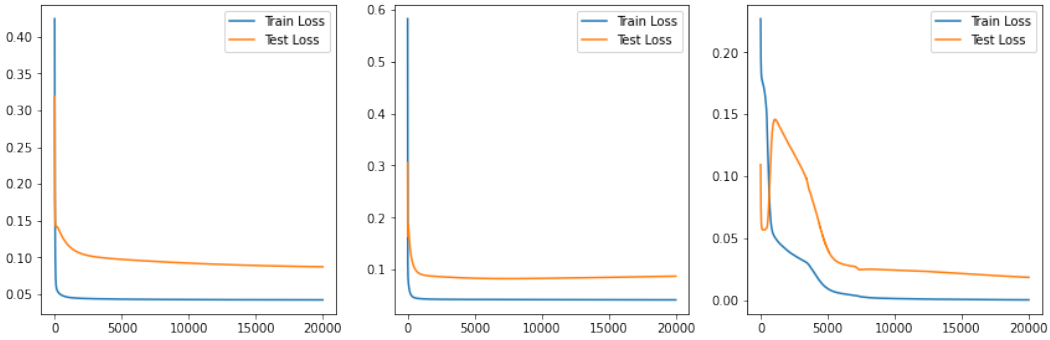


Figure 46: Training and testing loss curves for the standard LMS (left), non-linear perceptron (middle) and deep network (right) as a function of epoch count for 0 noise power

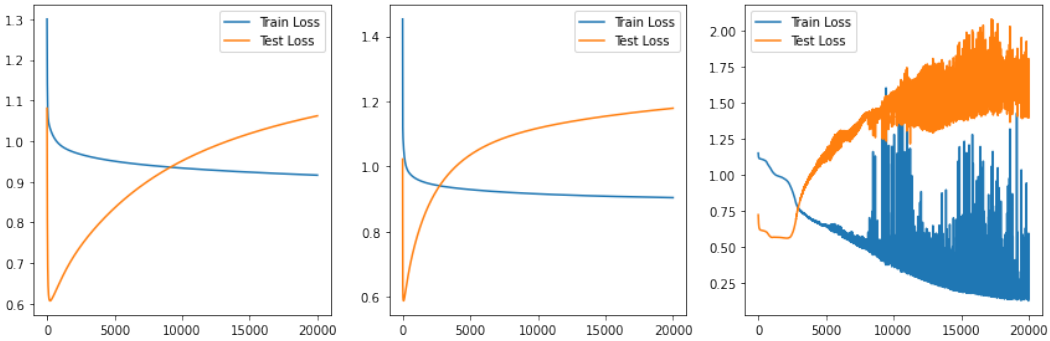


Figure 47: Training and testing loss curves for the standard LMS (left), non-linear perceptron (middle) and deep network (right) as a function of epoch count for a noise power of 1