Imperial College London

Imperial College London,
Department of Electrical and Electronic Engineering

Adaptive Signal Processing and Machine Intelligence Coursework

Submitted in partial fulfilment of the Honour code for coursework submission requirements from Imperial College London.

Author:
Bastien Caba

Course Leader:
Prof. Danilo P. Mandic

10th April 2019

Number of pages: 37/42

Contents

1 Classical and Modern Spectrum Estimation

1.1 Properties of Power Spectral Density (PSD)

We wish to demonstrate the equivalence in power spectrum definitions expressed through Equation 1 under the assumption that the covariance sequence $r(k)$ decays rapidly, as formulated in Equation 2. Let us note that Equation 2 is always satisfied for the realistic case of a finite-time signal, given that there will always exist a value k' such that $r(k)=0 \; \forall \; k>k'$.

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

$$\lim_{N\to\infty}\frac{1}{N}\left[\sum_{k=1-N}^{N-1}|k||r(k)|e^{-j\omega k}\right]=0 \tag{2}$$

Since the squared magnitude of a complex number z is given by the multiplication of z by its complex conjugate z^*, $|z|^2=zz^*$:

$$\lim_{N\to\infty} E\left\{\frac{1}{N}\left|\sum_{n=0}^{N-1} x(n)e^{-j\omega n}\right|^2\right\}=\lim_{N\to\infty}\frac{1}{N}\left[E\left\{\sum_{n=0}^{N-1} x(n)e^{-j\omega n}\sum_{m=0}^{N-1} x^*(m)e^{j\omega m}\right\}\right]=\lim_{N\to\infty}\frac{1}{N}\left[\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} E\Big\{x(n)x^*(m)\Big\}e^{-j\omega(n-m)}\right] \tag{3}$$

We now assume wide-sense stationary (WSS) zero-mean data such that the autocovariance function (ACF) is given by the equation $r(m)=E\{x(k)x^*(k-m)\}$. We also use the double summation formula $\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} f(n-m)=\sum_{k=-(N-1)}^{N-1}(N-|k|)f(n-m)$.

Let $k=n-m$:

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

From the linearity property of the sum operator, and using the assumption of Equation 2, we can conclude:

$$P(\omega)=\lim_{N\to\infty}\frac{1}{N}\sum_{k=1-N}^{N-1} Nr(k)e^{-j\omega k}-\lim_{N\to\infty}\frac{1}{N}\sum_{k=1-N}^{N-1}|k|r(k)e^{-j\omega k}=\lim_{N\to\infty}\sum_{k=1-N}^{N-1} r(k)e^{-j\omega k}-0=\sum_{k=-\infty}^{\infty} r(k)e^{-j\omega k} \quad \texttt{QED} \tag{5}$$

Equation 1 offers two equivalent ways to compute the PSD of a discrete-time signal under the assumption of zero-mean, WSS data where an infinite number of infinite-time realisations of the process are available. In practice, only one finite-time realisation of the process is available, therefore we neglect the limit and the ensemble average to define PSD estimators known as the periodogram and correlogram, based on the right-hand and left-hand expressions from Equation 1, respectively.

It is not possible to practically implement either the limit or the expectation operator present in either PSD definitions from Equation 1. In addition, while defining $x(n)$ as a deterministic signal would allow us to ignore the expectation operator, the PSD definitions proposed are designed for random signals. Therefore, as a substitute, we offer to compare the periodogram and the correlogram PSD estimates via the norm of their difference denoted by E, where the same white Gaussian noise signal is fed to both algorithms. Significantly, the unbiased correlogram does not satisfy Equation 2, causing the equivalence not to hold. In contrast, the equivalence holds when comparing the periodogram estimate to the biased correlogram estimate, as shown in Figure 1 for white Gaussian noise.

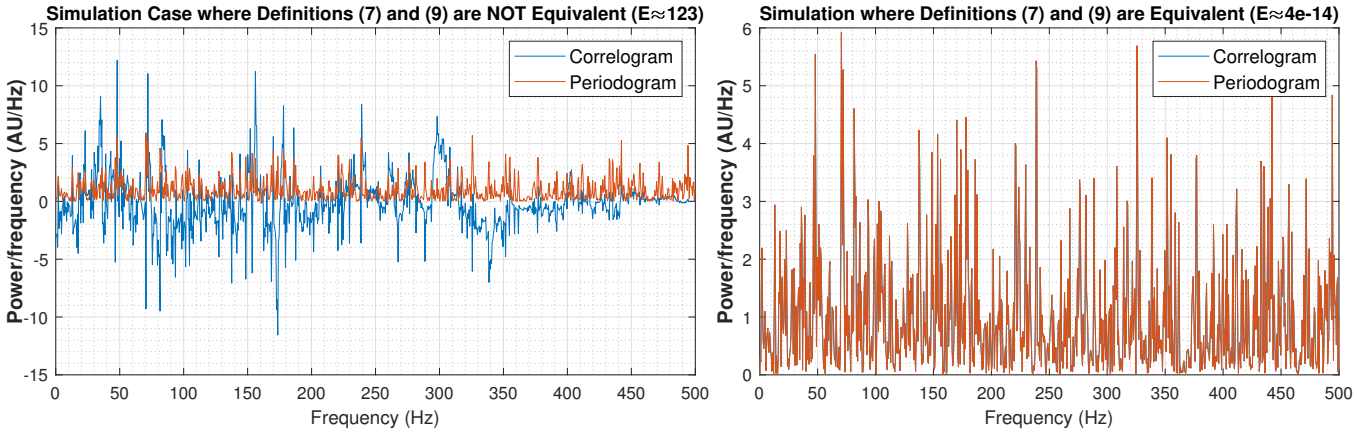

Figure 1: Left: Simulation of the periodogram and unbiased correlogram applied to white Gaussian noise demonstrating the non-equivalence of Equation 1, where the error term $E\approx 123$ is not negligible. **Right:** Simulation of the periodogram and biased correlogram applied to WGN demonstrating the equivalence of Equation 1, where the error term $E\approx 4\times 10^{-14}$ becomes negligible.

1.2 Periodogram-based Methods Applied to Real–World Data

a. We first consider the real-world real-valued sunspot signal $x(t)$: previously to any manipulation, the latter was scaled by its standard deviation σ_x. In order to explore different types of windowing, the periodogram was applied to the data for different pre-processing conditions after multiplication of the pre-processed data with windows of length equal to that of x but differing in shape. The effect of rectangular, Hamming and Hanning windowing on the performance of periodogram-based spectral estimation was evaluated. It was observed from Figure 2 that while Hamming and Hanning windowing offer smooth periodogram traces at the cost of wide lobes, rectangular windowing results narrow lobes with however high trace variability as the high slope of the truncation edges of a rectangular window is translated into high-frequency components appearing in the power spectrum. With regards to this trade-off, in order to satisfy our design criterion of clear identification of the main spectral component, the Hamming window was preferred for this application.

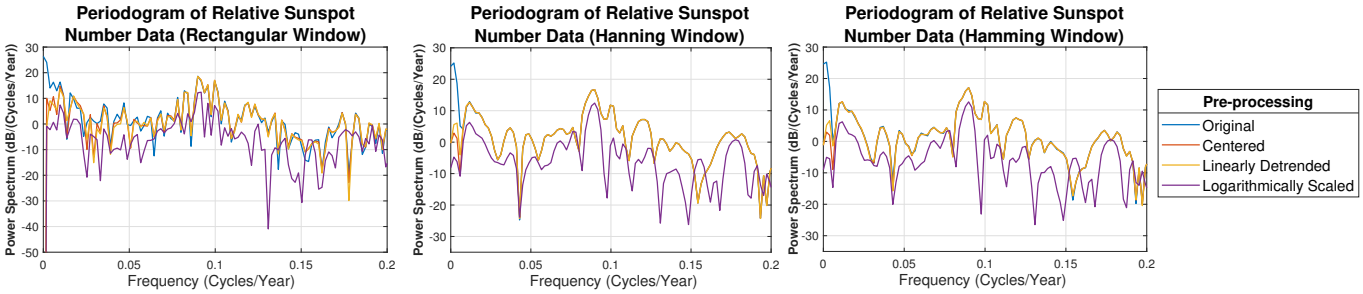

Figure 2: Detail of the periodogram-based PSD estimates obtained for different pre-processing conditions applied to the relative sunspot data $x(n)$, where only the low-frequency range $[0 \rightarrow 0.2]$ cycles/year is shown. The variability/lobe width trade-off can be observed.

We observe from Figures 2 and 3 that removing the signal mean eliminates the zero-frequency component of PSD estimate, which is coherent with the definition of the signal mean value as a constant (i.e. zero-frequency) term. Similarly, removing the data trend is analogous to low-pass filtering. The `MATLAB` function `detrend` removes a best straight-line fit to the data: this linear trend could be approximated by any sinusoid with period some orders of magnitude above the complete signal length, hence removing it suppresses oscillating components of frequency some order of magnitude below the inverse of the signal duration. It follows that the higher the slope of the linear trend, the higher the cut-off frequency of the conceptual low-pass filter associated with the detrending process. Indeed, a higher-slope linear component can be approximated by a higher-frequency sinusoid with period some orders of magnitude above the complete signal length, which justifies the claim. The plot of the original signal from Figure 3 demonstrates that the linear data trend exhibits a slow slope, hence the effect of detrending on the spectral estimate is similar to that of centring. Overall, while centring ensures the data is zero-mean as expected from the ACF form used in Equation 1, detrending can enforce strict stationarity of the data.

Lastly, logarithm scaling and subsequent centring of the data results in fewer and more distinct lobes in the PSD estimate, notably at the 0.09 cycles/year frequency and its harmonics. This indicates that logarithm scaling reinforces the existing data periodicity structure. In fact, since the derivative of the logarithm function is the inverse function, variations around small-amplitude points are amplified while variations around high-amplitude points are relatively dampened. In the case of sunspot data x, which behaves as a series of positive spikes from a zero baseline, the logarithm allows to compensate for the inherent signal asymmetry around the time axis resulting from the physical meaning of the data as a necessarily non-negative number. After logarithm application, variations in x are transformed from a spike-based behaviour to a more sinusoidal shape, and therefore data periodicity, in the sense of the power spectrum defined from the Fourier transform via a decomposition of any signal in a linear combination of harmonically related sinusoids, is reinforced.

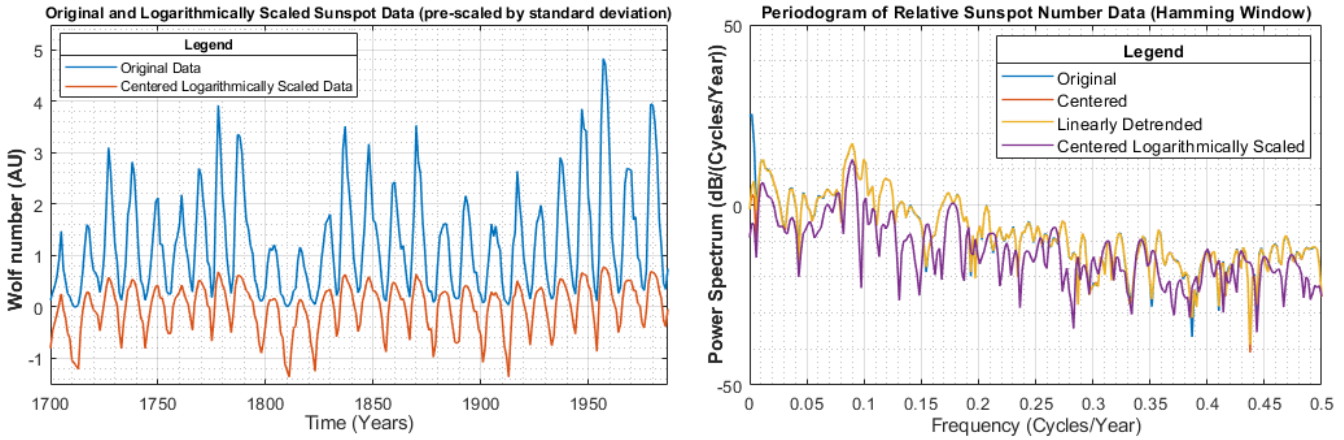

Figure 3: Original and logarithmically scaled sunspot data signals (**left**) and the periodograms of original, centred, linearly detrended and logarithmically scald sunspot data (**right**) under Hamming windowing. The periodogram peaks around 0.09 cycles/year.

b. We now process real-world electroencephalogram (EEG) data, which forms the basis of BCI. The standard periodogram (rectangular windowing) was applied to the entire EEG recording, with 5 DFT samples per Hz, as shown in Figure 4. The performance of the standard periodogram was compared to that of the averaged periodogram, while the effect of varying

window length within the averaged periodogram method was also explored. Specifically, we have chosen to use non-overlapping rectangular windows (Bartlett method was chosen instead of Welch or Blackman-Tuckey). We observe from Figure 4 a spectral peak corresponding to SSVEP at $13Hz$, as well as its associated harmonics.

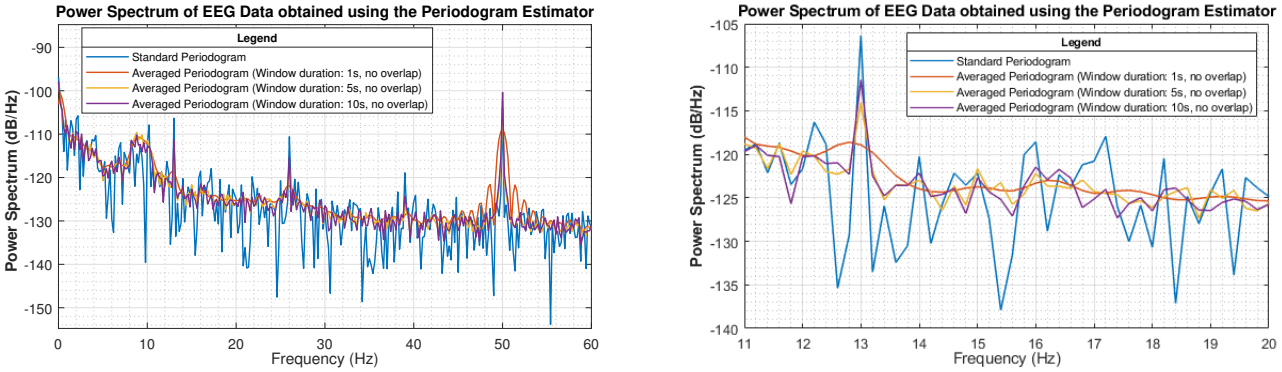

Figure 4: Standard and averaged periodograms obtained from real-world EEG data, showing a spike associated with SSVEP activity at 13Hz (**right**, showing a detail of the power spectrum on the $11-20$Hz frequency range of interest). The 50Hz mains noise and alpha-rhythm response over $8-10$Hz are also visible (**left**).

	Standard Periodogram	**Averaged Periodogram**		
Window Length (s)	NA	1	5	10
Variance (dB)	62.6	31.2	30.3	31.9

Table 1: Variance in the PSD trace shown in Figure 4, comparing standard and averaged periodogram performance.

The averaged periodogram method enables a reduction in PSD estimate variability when compared to the standard periodogram, making it easier to distinguish SSVEP from other random spurious spiking in the surrounding spectrum. This is demonstrated via Table 1, illustrating a halving of trace variance (in dB) when using the averaged periodogram method instead of the standard version. Assuming uncorrelated data windows, it can be shown that the variance in PSD estimate decreases with the number of averages, which is itself inversely proportional to window size, hence reducing window size reduces power spectrum variability. However, the width of the lobes appearing in the periodogram output are also inversely proportional to the window size (Equation 9), thus a trade-off must be met between variance reduction and lobe widening. With regards to our objective of SSVEP identification, design requirements are met for window sizes of $5s$ or $10s$, while for $1s$ the width of the lobe is such that the periodogram peaks at a frequency of 12.8Hz which compares against the true 13Hz. Let us note that conservation of power principles imply that lobe widening causes a decrease in the spectral power value at the peaks. Lastly, the effect of side-lobe spectrum pollution can be clearly observed around the 50Hz mains frequency for a window length of 1s, as shown in Figure 4. This effect causes the trace variance for that window length to exceed that observed for a window length of 5s, as shown in Table 1, while we should otherwise expect trace variance to decrease with window length.

1.3 Correlation Estimation

a. The correlogram is a PSD estimator based on the left-hand side of Equation 1. It can be computed using a biased $\hat{\phi}_b$ or unbiased $\hat{\phi}_u$ autocorrelation estimator, to which `fft` is subsequently applied. We have chosen to define the autocorrelation of a signal x for lags k running from $N-1$ to $1-N$, where N is the length of x, such that our autocorrelation estimators are symmetrical with respect to the zero-lag axis, resulting in purely real-valued correlograms after application of `fft`. While the mean of $\hat{\phi}_b$ matches the true mean of PSD under the assumption of a wide-sense stationary $x(n)$ (Equation 6), $\mathbb{E}[\hat{\phi}_u]$ can be expressed as the true autocorrelation $\phi_{xx}(k)$ multiplied with a triangular window known as the Bartlett window, centred at zero lag and expressed in the scaling term in Equation 7. This triangular windowing ensures stability of the autocorrelation estimator at large lags, preventing it from becoming chaotic as the scaling $N-|k|$ becomes small and the autocorrelation is computed using less samples. The autocorrelation estimators and corresponding correlograms were computed for a deterministic constant $x=1$ signal, such as to illustrate the Bartlett windowing effect, as shown in Figure 5.

$$\mathbb{E}[\hat{\phi}_u] = \mathbb{E}\left[\frac{1}{N-|k|}\sum_{n=|k|+1}^{N} x(n)x^*(n-k)\right] = \frac{1}{N-|k|}\sum_{m=1}^{N-|k|}\mathbb{E}\Big[x(m+k)x^*(m)\Big] = \frac{1}{N-|k|}\sum_{m=1}^{N-|k|}\phi_{xx}(k) = \phi_{xx}(k) \tag{6}$$

$$\mathbb{E}[\hat{\phi}_b] = \mathbb{E}\left[\frac{1}{N}\sum_{n=|k|+1}^{N} x(n)x^*(n-k)\right] = \frac{1}{N}\sum_{m=1}^{N-|k|}\mathbb{E}\Big[x(m+k)x^*(m)\Big] = \frac{1}{N}\sum_{m=1}^{N-|k|}\phi_{xx}(k) = \left[1-\frac{|k|}{N}\right]\phi_{xx}(k) \tag{7}$$

The biased and unbiased correlogram spectral estimators were applied to different random signals including white Gaussian noise (WGN) (Figure 6), a noisy sinusoid at 200Hz corrupted with WGN (Figure 5), and pink noise (Figure 7), in order to identify differences between spectral estimates obtained from biased and unbiased auto-correlations for a variety of test signals. With each signal, the variability of the unbiased correlogram was larger than that of the biased version, with the difference becoming more visible for larger lags k. In addition, as a consequence of the erratic behaviour of $\hat{\phi}_u$ for large lags causing $\hat{\phi}_u$ to be non-positive definite, PSD estimates obtained from the unbiased correlogram can become negative, which conflicts with the physical meaning of PSD as a necessarily non-negative power spectra.

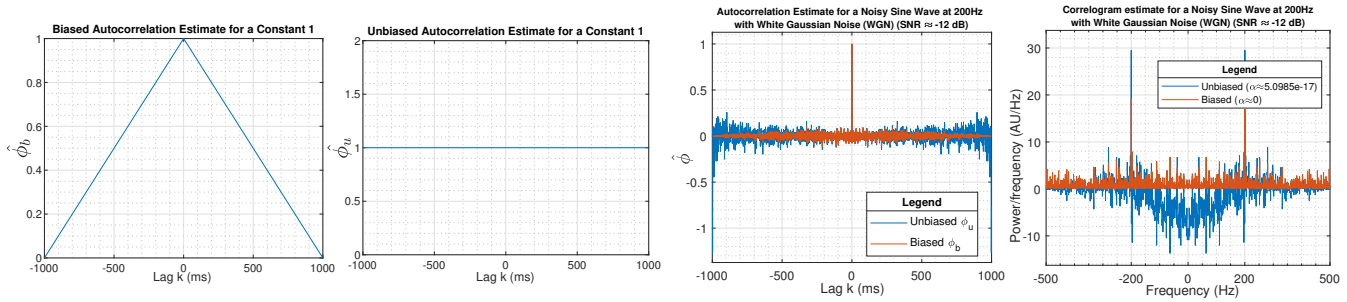

Figure 5: Left: Biased and unbiased autocorrelation computed from the constant signal $x=1$, showing for illustration the Bartlett windowing effect. **Right:** Autocorrelation and correlogram (real-valued) computed using biased and unbiased methods for a sine wave at 200Hz with WGN at SNR of -12dB, showing erratic behaviour of the unbiased autocorrelation at large lags resulting in negative-valued power spectrum. The α coefficient measures the mean value of the imaginary correlogram component: it is negligible in both cases.

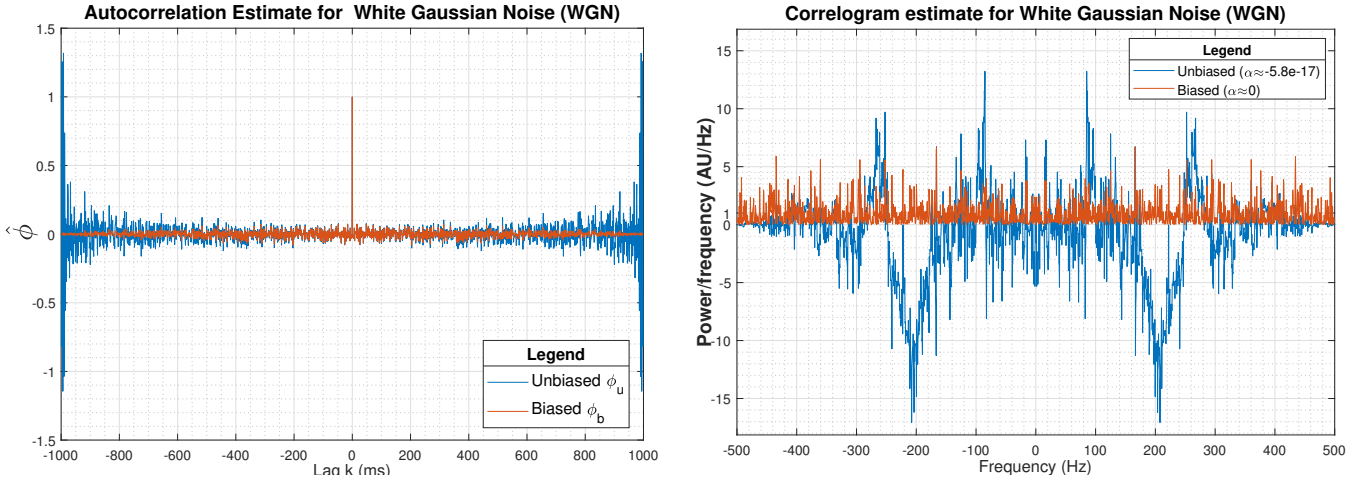

Figure 6: Autocorrelation (**left**) and correlogram (**right**) computed using biased and unbiased methods for white Gaussian noise, showing that the power spectrum under biased estimation oscillates around a power magnitude of 1, as expected. The α coefficient measures the mean value of the imaginary correlogram component: it is negligible in both cases.

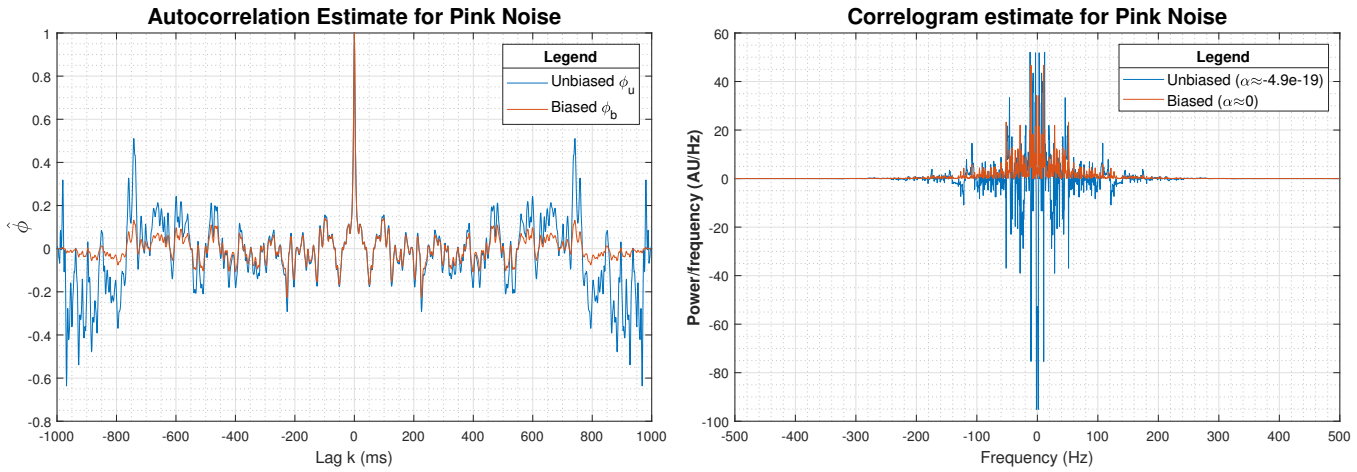

Figure 7: Autocorrelation(**left**) and correlogram (**right**) computed using biased and unbiased methods for pink noise (filtered WGN), showing that the power spectrum decays exponentially away from the zero frequency, as expected. The α coefficient measures the mean value of the imaginary correlogram component: it is negligible in both cases.

As detailed in the legend of the plots shown above, the PSD estimates obtained for each test signal are as expected in the case of the biased correlogram, which validates our method.

b. The biased correlogram was applied to each one of 100 realisations generated from a random process $x(n)$ consisting of two sinusoids corrupted with WGN with a signal-to-noise ratio (SNR) of approximately -18dB, as expressed in Equation 8 where $w \sim \mathcal{N}(0,1)$. Every realisation was z-scored such that the resulting $x(n)$ were zero-mean with a standard deviation of 1. We have chosen to zero-pad the signal by computing `fft` over twice as many frequency-domain samples as time-domain samples in the original realisations, such as to obtain a smooth sinc behaviour in the PSD estimates, as in the coursework. The high SNR value was chosen such as to highlight the power of averaging over realisations. Indeed, Figure 8 demonstrates that although very few of the correlogram realisations matched the true process PSD, the mean correlogram $\hat{P}$ peaks clearly at f_1 and f_2.

$$x(n) = \sin(2\pi f_1 t) + \sin(2\pi f_2 t) + w(n) \qquad f_1 = 100Hz \quad f_2 = 150Hz \tag{8}$$

It can be observed that the standard deviation in the correlogram at a given frequency is proportional to the true PSD value at this frequency, where we approximate the true PSD value with the average correlogram value across realisations. We indeed note that the correlogram realisations are most dispersed at frequencies f_1,f_2 corresponding to peaks in the mean correlogram. As a result, the standard deviation trace peaks at f_1 and f_2, and our confidence interval is widest at those frequencies. The normalised dot product between the mean $\bar{\mathbf{P}}$ and the standard deviation $\sigma_{\mathbf{P}}$ of the correlogram trace across realisations was given by $\frac{\bar{\mathbf{P}}\cdot\sigma_{\mathbf{P}}}{||\sigma_{\mathbf{P}}||||\bar{\mathbf{P}}||}\approx 0.995$, confirming the strong linear dependency of the standard deviation on the correlogram mean.

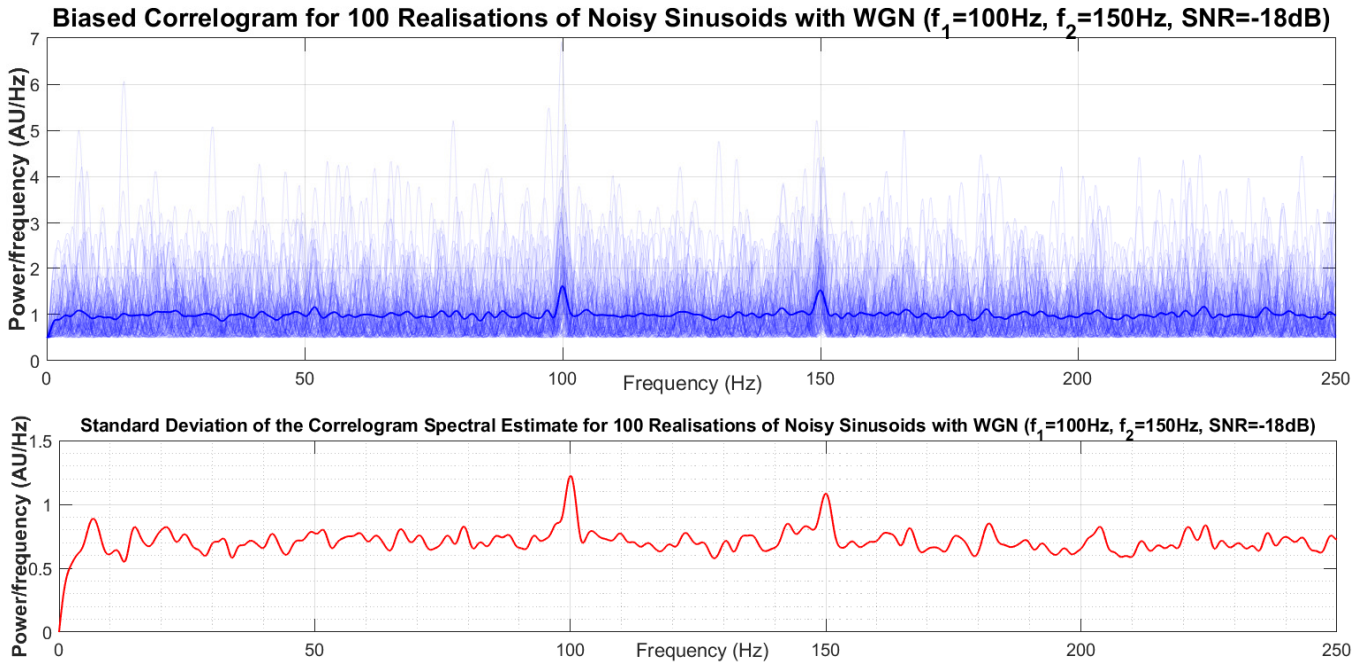

Figure 8: Top: Overlay plot of the biased correlograms obtained from 100 realisations of the process $x(n)$ (Equation 8), with mean correlogram shown as the thick deep blue trace. **Bottom:** Standard deviation of the correlograms obtained from 100 realisations of the process $x(n)$ (Equation 8), showing clear peaks at $f_1 = 100$Hz and $f_2 = 150$Hz.

The above was repeated for the higher SNR of 0dB for comparison, as shown below. As expected, the correlograms peaks more clearly at the frequencies $f_1 = 100$Hz and $f_2 = 150$Hz, both in the PSD trace and in its standard variation across realisations.

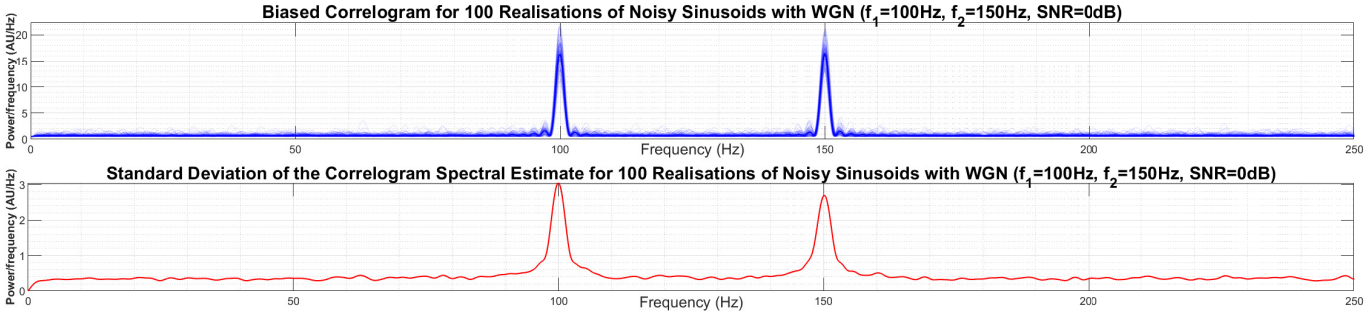

Figure 9: Equivalent of Figure 8, where the SNR was modified from -18dB to 0dB for comparison.

c. Due to the contraction property of the logarithm, by plotting correlogram results in decibels we observe a more condensed representation of the correlogram realisations, as shown in Figure 10. We propose that since the derivative of the logarithm is the inverse function, then variations in PSD estimate around a point are amplified by the inverse of the value of the PSD estimate at that point. Hence, the amplification is sub-linear around the spectral peaks where the power density exceeds 1, and supra-linear elsewhere as the power density is less than 1. As a conclusion, the decibels spectrum representation collapses the confidence bounds around spectral peaks, thereby facilitating its interpretation.

In Figure 10, we observe that since the spectral peaks associated with f_1 and f_2 were not clearly above 1, the standard deviation behaviour is swamped by the large noise component and consequently constantly oscillates around 2.75 across frequencies. In contrast, when increasing the SNR from -18dB to 0dB, the spectral peaks clearly exceed 1 (see Figure 9) and consequently the standard deviation at the frequencies f_1 and f_2 is lowest, as can be seen from Figure 11.

d. We generated a set of stochastic complex signals $x_i(n)$ at a sampling frequency of $f_s = 1$Hz consisting of two complex exponentials at frequencies f_1 and f_2 with added complex white Gaussian noise given by $w \sim \mathcal{N}(0,0.2)$. The frequencies f_1,f_2 as well as the signal length N were varied. The periodogram was applied to this set of complex signals, under rectangular windowing of the signal by its full length and using 128 bins along the frequency axis, as shown in Figure 13. The effect of varying N on the ability of the periodogram to distinguish one exponential component from the other was studied, in order to characterise the frequency resolution of the periodogram.

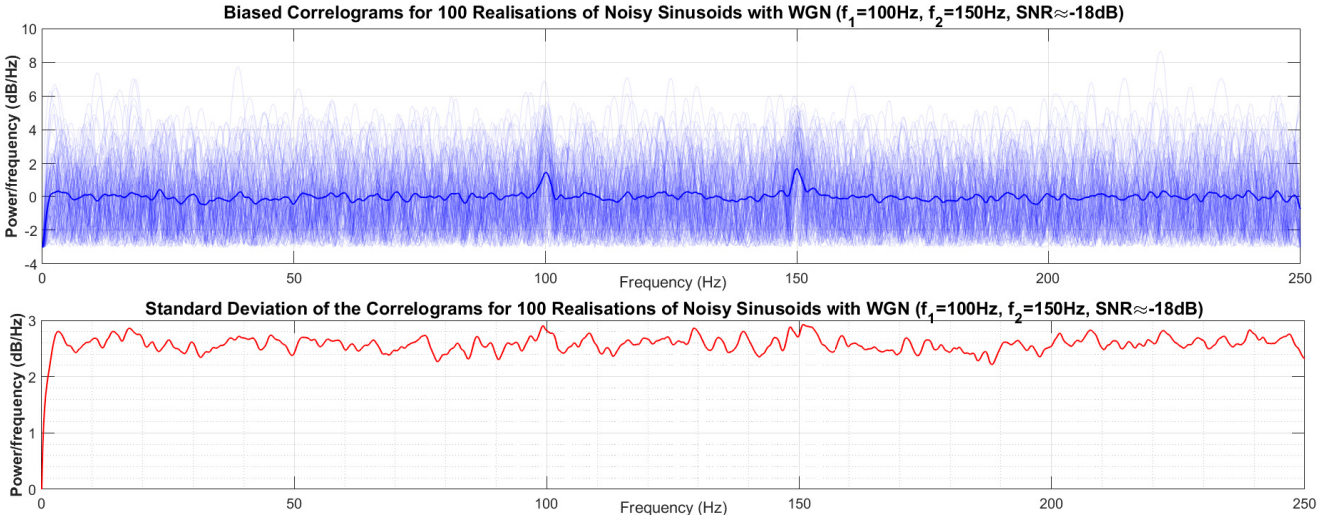

Figure 10: Top: Overlay plot of the biased correlograms obtained from 100 realisations of the process $x(n)$ (Equation 8) shown in Decibels, with mean correlogram shown as the thick deep blue trace. **Bottom:** Standard deviation of the correlograms obtained from 100 realisations of the process $x(n)$ (Equation 8), showing constant standard deviation across frequencies.

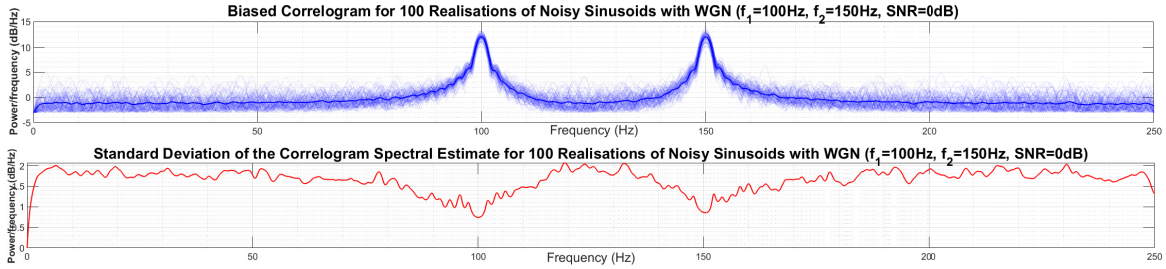

Figure 11: Equivalent of Figure 10, where the SNR was modified from -18dB to 0dB for comparison. It can be seen that the standard deviation is minimal at the peak frequencies f_1 and f_2, which demonstrates the benefits of the decibels representation.

We assume that finite-length complex signals $x_i(n)$ can be represented by the multiplication of the equivalent infinite-time process $x_\infty(n)$ with a rectangular window $h(n)$ with length N: $x_i = x_\infty \times h$. Let X_∞ and H be the DFT products of x_∞ and h respectively, we then know that the DFT spectrum of each signal x_i will be given by Equation 9, where the finite-duration property of $x_i c$ causes the true DFT spectra of the signal to be convolved with a sinc function, resulting in spectral lobes in frequency domain of width $\frac{1}{N}$, or $\frac{2}{N}$ for the main lobe. The lobe width is known as the resolution of the DFT, and hence of the periodogram.

$$X_c = \frac{1}{2\pi}\Big[X_\infty \circledast H\Big] = \frac{1}{2\pi}\left[X_\infty \circledast \mathrm{sinc}\Big(\frac{N}{2}f\Big)\right] \tag{9}$$

As a result, if the difference between the two main frequency components of the complex-valued exponential-noise mixture, denoted by $\delta_f = |f_1 - f_2|$, is smaller than the spectrum resolution N^{-1}, then we cannot guarantee that the two frequency components can be distinguished from one another. This yields the following criterion on N: $N \geq \delta_f^{-1}$, which guarantees the identification of two peaks at f_1 and f_2, and in our case $N \geq 50$Hz. Let us finally note that this is a criterion to guarantee identification of two spectral peaks, while slightly smaller values for N may also allow for the same result.

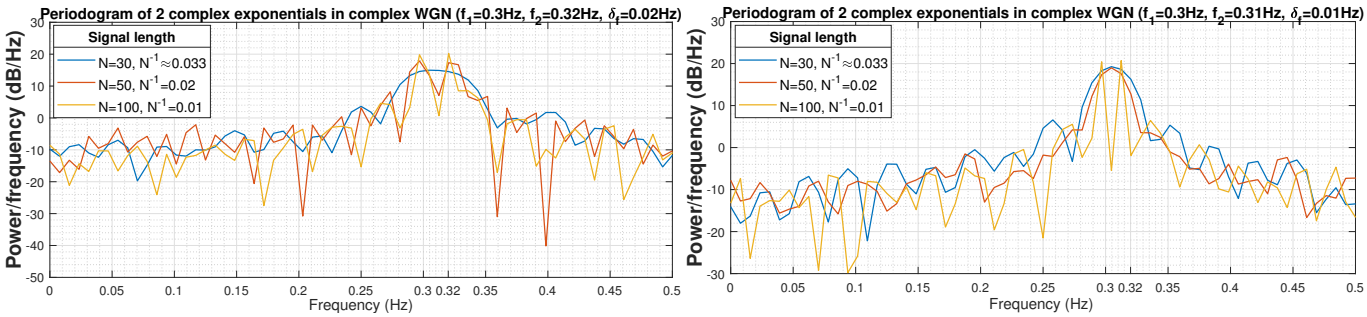

Figure 12: Single-sided periodograms computed from signals x_i, demonstrating that when the periodogram resolution $\frac{1}{N}$ is smaller than or equal to the difference δ_f between the two exponential component frequencies, then those exponentials can be distinguished from one another and the power spectrum estimate exhibits two distinct peaks.

e. The MUltiple SIgnal Classification (MUSIC) algorithm is a subspace method which assumes a harmonic model for a given complex signal x to estimate its power spectrum. The model assumes that x consists of p complex exponentials in additive complex noise, and attempts to generate two orthogonal sub-spaces upon which x is projected, with the objective to separate the complex exponentials and the additive noise across sub-spaces, such as to recover a noise-free power spectrum.

The algorithm sorts the eigenvalues of the autocorrelation matrix $\mathbf{R}_{xx} \in \mathbb{C}^{M\times M}$ of x in decreasing order, where $M \leq N$ (where N is the signal length), such as to extract the eigenvectors corresponding to the p largest eigenvalues of $\mathbf{R}_{xx}$. To this regard, MUSIC is a dimensionality reduction technique used to resolve the time–frequency uncertainty issues in frequency–based Machine Intelligence. These p eigenvectors obtained represent the directions of largest variability in $\mathbf{R}_{xx}$. Since the autocorrelation of white Gaussian noise is constant, we expect these p eigenvectors to span a subspace $\mathbf{R}_s$ upon which x can be projected to extract only its exponential components. Assuming the exponential components and white Gaussian noise are uncorrelated, the remaining $M-p$ eigenvectors should span a subspace $\mathbf{R}_n$ orthogonal to $\mathbf{R}_s$, upon which x can be projected to extract its noise component. Let us define the noise eigenvectors as $\mathbf{v_i^n}$ and the steering vector $\mathbf{e}$ as $\mathbf{e} = \begin{bmatrix} 1 & e^{jw} & e^{j2w} & \cdots & e^{j(M-1)w} \end{bmatrix}^T$, then the MUSIC spectral estimate is given by:

$$\hat{P}_M(e^{jw}) = \frac{1}{\sum_{i=p+1}^{M} \left| \mathbf{e}^H \mathbf{v_i^n} \right|^2} \tag{10}$$

At signal frequencies w_k where $\mathbf{e}^H$ becomes orthogonal to the noise eigenvectors $\mathbf{v_i^n}$ (which happens when $\mathbf{e}^H$ equals one of the p eigenvectors spanning $\mathbf{R}_s$, hence when w_k is at the frequency of one of the p exponential components in x), we will obtain a peak in the spectrum since $\mathbf{e}^H \mathbf{v_i^n} = 0$. Therefore, $\hat{P}_M$ will have p peaks, as expected for a spectrum estimator of p complex exponentials.

The MATLAB command `[X,R] = corrmtx(x,M,'mod')` yields the modified $(M+1)\times(M+1)$ autocorrelation matrix estimation `R` for the signal `x`. This autocorrelation estimation matrix is obtained using forward-backward method, is Toeplitz, unbiased, and displays conjugate symmetry. We have here set $M = 14$ such that `R` will have dimensions 15×15, where in the context of our signal of length $N = 30$ we have cropped the complete autocorrelation to keep only half of its samples, such as to avoid instability at large lags k and ensure minimal variability in the resulting PSD estimate. The line `[S,F] = pmusic(R,2,[],1,'corr')` applies the MUSIC algorithm to the matrix `R`, assuming `R` is an autocorrelation matrix (`'corr'`) and assuming a harmonic model of order `p=2` for x, to fit its definition as two exponentials submerged in complex WGN. the hyperparameter `p` can be tuned in the case where the number of exponential components in a given random signal is unknown.

Due to its nature, the MUSIC algorithm essentially acts as an exponential component detector and offers a less detailed representation of the power spectrum where a large portion of the information present in the data is discarded in order to better highlight the presence of complex exponential components. As a result, its response is highly stereotypical and its variance is minimal, which is a characteristic feature of model-based approaches. While the periodogram offers an unbiased estimation of the true spectral density of the data but suffers from a dependency of spectral resolution on the inverse of data length, the MUSIC algorithm offers a strongly biased spectral representation where only complex exponential components are expressed in the power spectrum, but also exhibits a spectral resolution independent from data length. We propose that the accuracy of the MUSIC algorithm in identifying complex exponential components in real-world data would be high in an ideal setting, but would rely upon the whiteness of the corrupting noise and the absence of correlation between the data and the noise. In other terms, the MUSIC performance is highly dependant upon how data fits the assumed model, and relies upon our ability to decide on how many exponential components p are present in the data provided. Nevertheless, where the assumed model is appropriate, MUSIC mitigates the variance and resolution limitations of classical spectrum estimation methods.

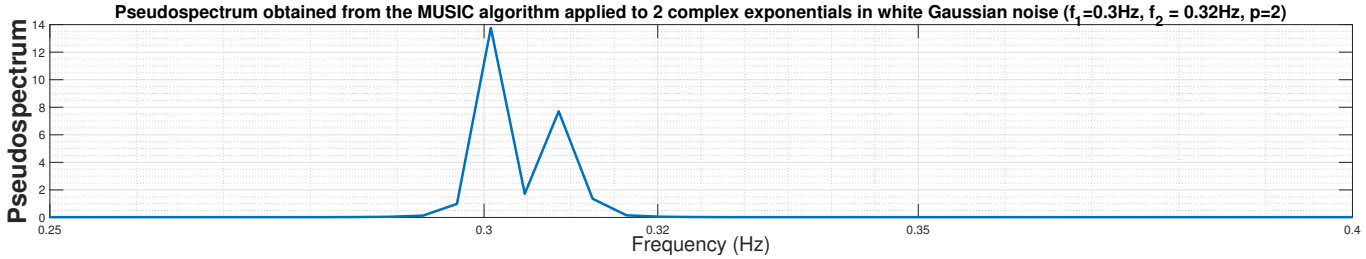

Figure 13: MUSIC power spectrum density obtained from a single realisation of the process $x(n)$ described in Equation 10, where the hyperparameter p was set as $p = 2$. Both frequency components at f_1 and f_2 could be identified.

1.4 Spectrum of Autoregressive Processes

a. We have shown that classical spectrum estimation techniques, including the periodogram and correlogram methods, can suffer from high trace variability, non-zero bias and poor frequency resolution when confronted to short data lengths. In order to solve this problem, we use modern estimation techniques which assume an underlying pole-zero model for the data. The problem of spectral estimation is then transferred to the problem of estimation of the model parameters. For an AR process in particular, the power spectrum is the output of an all-pole filter, where the parameters σ_w^2 and the vector $\mathbf{a}$ can be estimated using the Yule-Walker equations from biased or unbiased autocorrelation estimates, by inverting the autocorrelation matrix $\mathbf{R}_{xx}$: $\mathbf{a} = \mathbf{R}_{xx}^{-1}\mathbf{r}_{xx}$. We require the ACF matrix $\mathbf{R}_{xx}$ to be positive definite in order to guarantee its inversion. Given the erratic behaviour of the unbiased ACF estimate $\hat{\phi_u}$ for large lags k could result in $\hat{\phi_u}$ to be non positive definite and possibly compromise the invertiblity of $\mathbf{R}_{xx}$, we suggest that the biased ACF estimate should be used in order to guarantee that the Yule-Walker equation can be solved reliably, resulting in accurate coefficients for the auto-regressive model.

b. We have generated $N = 1000$ samples of data in `MATLAB`, following the auto-regressive equation given in the coursework from a random initial condition. In order to limit the dependency of our analysis upon the random initial condition generated, the transient filter output consisting of the first 500 samples of the trace were discarded. The true and estimated power spectrum were compared when performing parameters estimators assuming models of order p ranging from 2 to 14, where the true spectrum was obtained from the true AR(4) parameters defined as $\mathbf{a} = \begin{bmatrix} 2.76 & -3.81 & 2.65 & -0.92 \end{bmatrix}$.

It is observed that increasing the model order p from $p=2$ up to $p=9$ results in a reduction of the mean error between the estimated and true power spectra (see Figure 14), while increasing the model order beyond $p=9$ results in increased model complexity, with no error reduction benefit, and a risk of over-fitting. Consequently, for models of order $p \geq 9$, the power spectrum estimate exhibits the expected two-peak behaviour (see Figure 15), while for model orders $p<5$ it fails to do so. In addition, for the high model order $p=14$, unwanted oscillations in the PSD trace are introduced in the high-frequency range, which results from over-fitting. Although we may expect optimal results to be obtained when the assumed underlying model matches the true model $p=4$, factors such as the short signal length and the choice of a biased autocorrelation estimator require the underlying model to exhibit additional degrees of freedom in the AR model in order to capture the true two-peak spectral behaviour.

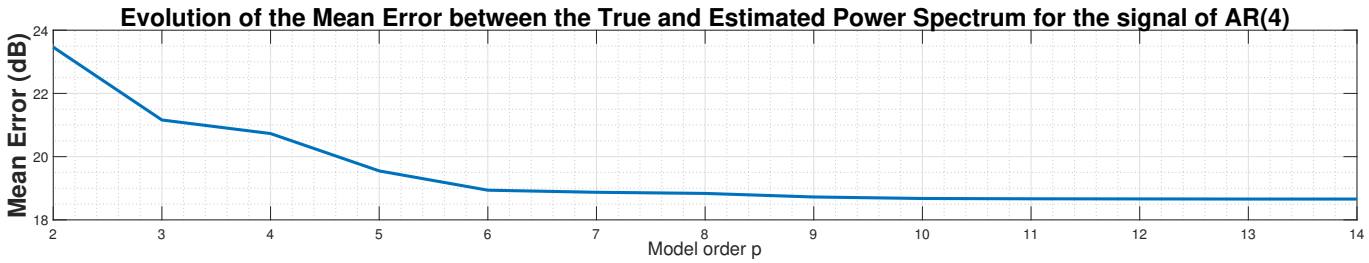

Figure 14: Evolution of the mean error (in decibels) between the true and estimated power spectra for the signal generated from an AR(4) process with parameters **a**. It can be seen that the error plateau after $p=9$. We have kept $N=1000$.

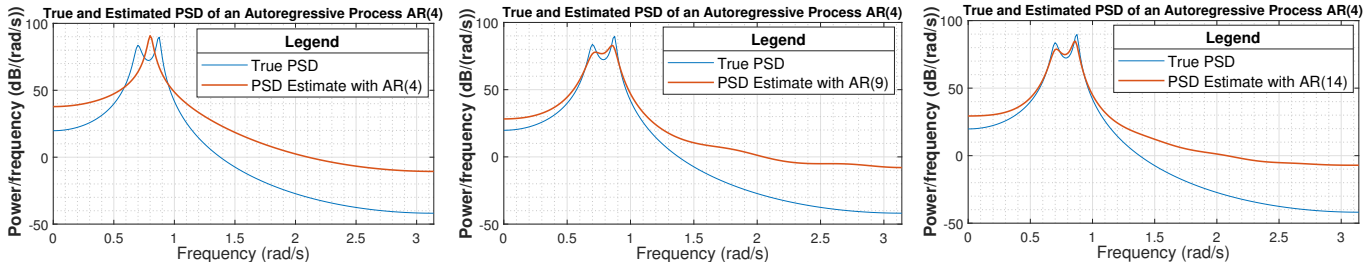

Figure 15: Left: True and estimated power spectra for the signal generated from an AR(4) process modelled with an AR(4) model. **Middle:** True and estimated power spectra for the signal generated from an AR(4) process modelled with the optimal AR(9) model. **Right:** True and estimated power spectra for the signal generated from an AR(4) process modelled with an AR(14) model.

c. The previous experiment was repeated for a signal length $N=10^4$, which highlighted that the required assumed model for optimal power spectrum estimation converges towards the true underlying model when the amount of available data increases. Overall, increasing the signal length increases the accuracy in parameters estimation, yielding a reduced mean error between the true and estimated power spectra. With $N=10^4$, we observe that under-modelling systematically results in the inability of the spectrum to exhibit two distinct peaks, while over-modelling above AR(5) results in added ripples in the power spectra as the Yule-Walker algorithm fits model parameters to the additive WGN. The corresponding curve of mean error against model order p is shown below. Modelling at the true model order 4 can still occasionally fail in the identification of two distinct peaks, as shown in Figure 17. When setting $N=10^5$, the Yule-Walker method consistently yields two spectral peaks when assuming an underlying model of order 4. In the case of an unknown model order, some criteria including the Akaike Information Criterion (AIC), the Bayesian Information Criterion or MDL can be used to determine the model order most appropriate to the data, via penalsiation of high model orders.

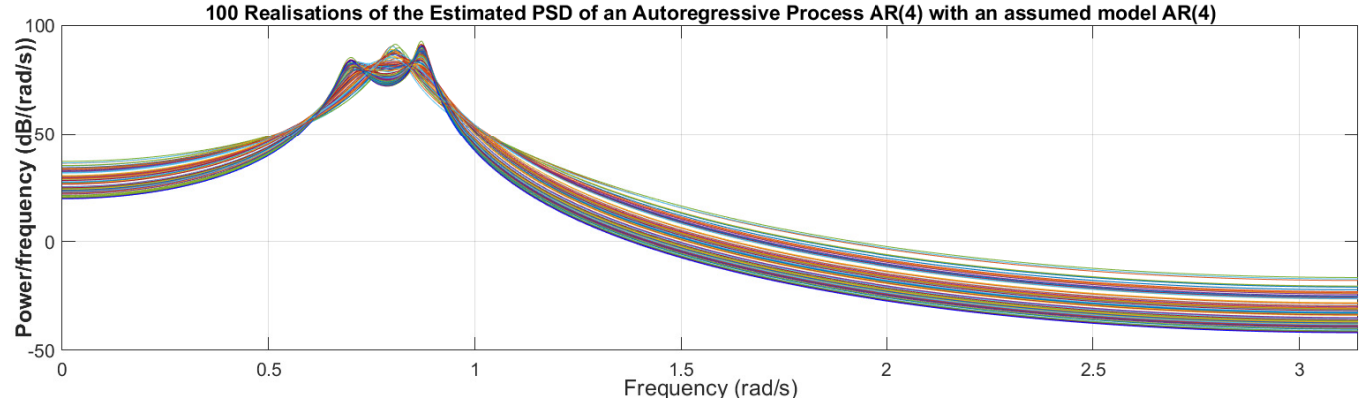

Figure 16: Overlay plot of 100 realisations of the estimated PSD of the signal generated from an AR(4) process of length $N=10^4$ and modelled with an AR(4) model, showing occurrences where the Yule-Walker algorithm fails to identify two distinct spectral peaks. The realisations differ by their initial starting conditions for the process $x(n)$.

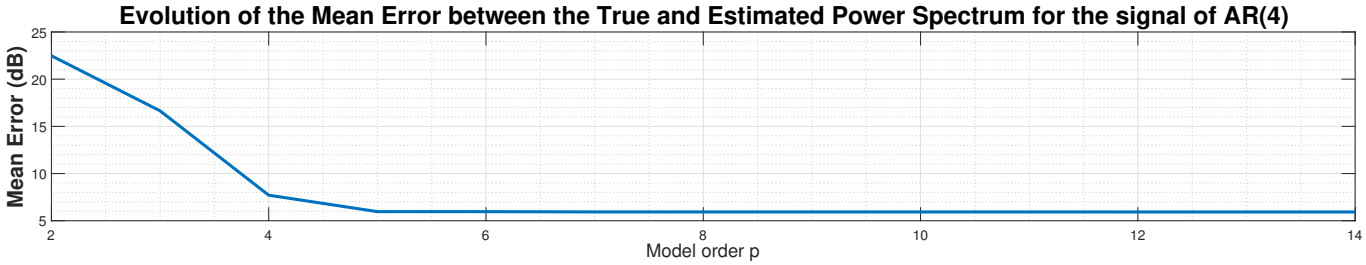

Figure 17: Evolution of the mean error (in decibels) between the true and estimated power spectra for the signal generated from an AR(4) process with parameters **a**. It can be seen that the error plateau after $p=5$. We have kept $N=10^4$.

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

a. Respiratory sinus arrhythmia (RSA) refers to the modulation of cardiac function by respiratory effort. This section aims to use our knowledge of this effect to identify breathing frequency from the R-R Interval data (RRI) of an ECG. The standard periodogram (`MATLAB` function `periodogram`) and averaged periodogram (MATLAB function `pwelch`) with window lengths 50s and 150s were applied to the RRI data shown for the three trials separately in Figure 18, where each trial corresponds to a breathing frequency. Since we expect the spectral peaks associated with breathing to be at low frequencies, the RRI data was preprocessed through centring, detrending and linear scaling by its standard deviation, such as to eliminate any artificial spurious spiking not associated with breathing in the low-frequency range. The frequency axis shown was obtained by multiplying by 60 the Hz-axis to obtain a BPM-axis, where BPM stands for Breaths Per Minute.

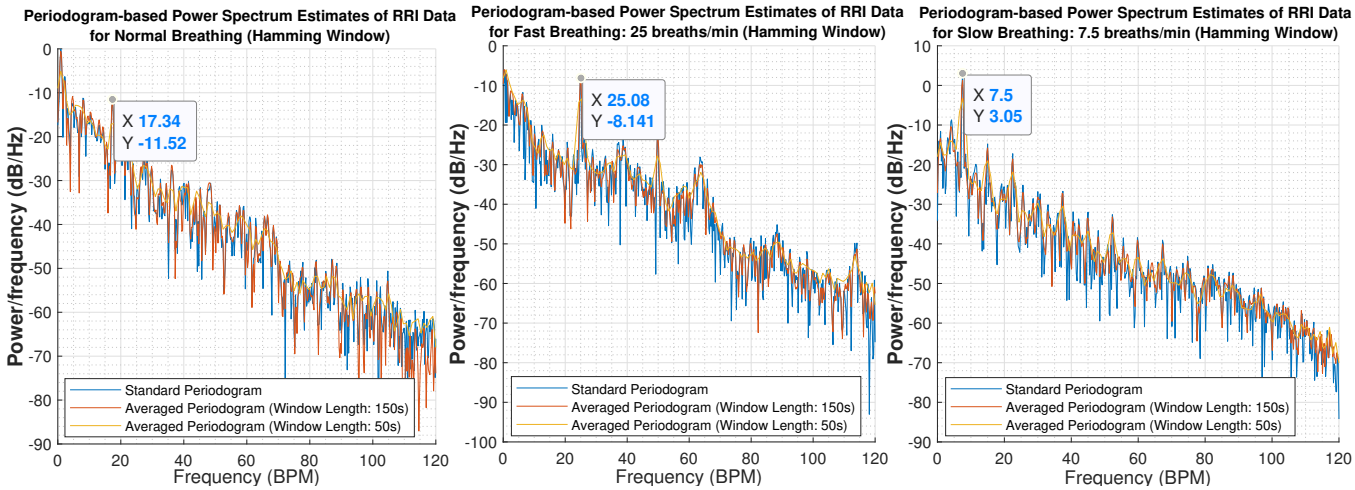

Figure 18: Periodogram-based PSD estimates of the RRI data using Hamming windowing: the standard and averaged periodograms with window lengths 50s and 150s are shown for each of the three trials (**left:** normal, **middle:** slow, **right:** fast). The spectral peaks associated with breathing frequency have been highlighted.

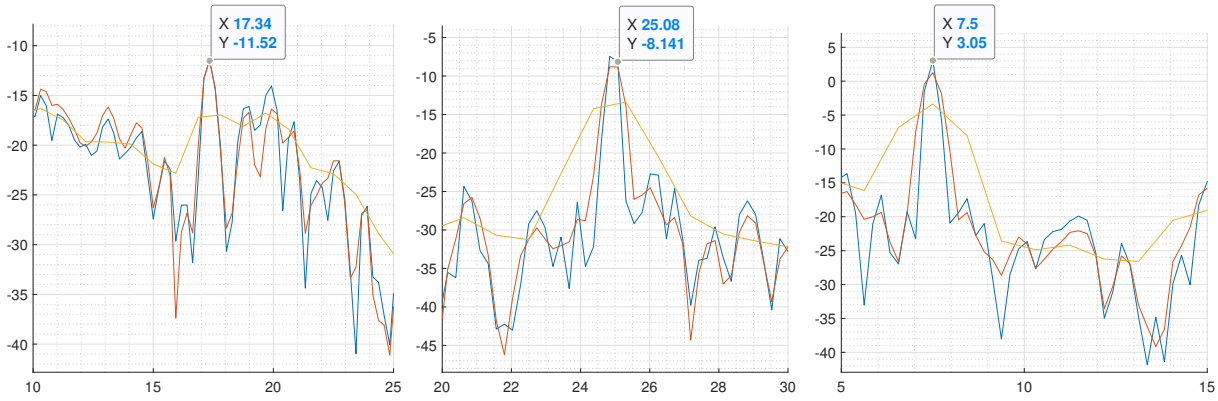

Figure 19: Detailed view of the spectral peaks associated with breathing frequency in each trial of Figure 18 shown above.

b. Respiratory sinus arrhythmia (RSA) is observed in each of the three trials: the RR-Intervals signal has been modulated by breathing frequency, such that the periodogram-based PSD estimates of the RRI data contain spectral peaks at the breathing frequencies of 17.34 breaths/minute, 25.08 breaths/minute and 7.5 breaths/minute for unconstrained, fast and slow breathing respectively, as shown in Figure 19. In the case of fast breathing, up to the 10th harmonic of the breathing frequency can be observed, while in the case of slow breathing only the second harmonic can be detected. We propose that this results from the fact that the subject was breathing with increased vigour during the fast trial to sustain the fast breathing rhythm, which is translated into more spectral power at the breathing frequency in the PSD estimate shown.

The standard periodogram estimate exhibits higher variability and smaller spectral lobe width, while in comparison the averaged periodogram exhibits smaller variability at the cost of wider lobes. In fact, since the averaged periodogram is computed from averaging over M windows of data each of length $\frac{N}{M}$ where N is the original data length, and using Equation 9, we state that sinc lobe width increases inversely proportionally with the window length. The benefit of reducing window length within the averaged periodogram method is a reduction in trace variance as the number of averages increases, as observed. Hamming windowing was selected to further enforce the smoothness of the PSD trace.

c. To compare classical and modern spectral estimation methods on real-world data, the AR spectrum estimate of order p was fitted to the RRI data using the Yule-Walker equation for parameters estimation. The optimal model order p was selected by visual comparison of the AR(p) spectrum estimate with the Hamming-windowed averaged periodogram for a window length of 50s. The latter was used as reference since, as previously established, small window lengths result in smooth and simplified PSD traces, which renders visual comparison easier. Specifically, for each trial, the low model order $p=1$ was shown, along with the high $p=20$ model order demonstrating fitting of the breathing frequency harmonics. Lastly, a third optimal order p_{opt} is shown, and was chosen as the smallest order (least complexity argument, Occam's razor) for which the spectral spike associated with breathing frequency could be clearly identified.

AR modelling can provide a robust estimation of spectral peaks with minimal power spectrum trace variability and no dependency of the spectral lobe width on data length, in contrast with the periodogram-based estimates where the variance-resolution trade-off is expressed. While under-modelling ($p<p_{opt}$) can result in omission of spectral peaks, over-modelling ($p>p_{opt}$ causes the model to fit the data noise, resulting in false spectral peaks. In conclusion, the efficacy of AR modelling relies on our ability to select an appropriate model order as well as on the ratio of signal autocorrelation to noise autocorrelation. In the case of white noise, with poor noise autocorrelation, the auto-regressive approach is thus effective. As a conclusion, modern model-based spectral estimation methods do not suffer from resolution, variance and bias limitations encountered by classical spectral estimators, but rely upon our ability to select appropriate hyper-parameters for an *a-priori* unknown signal.

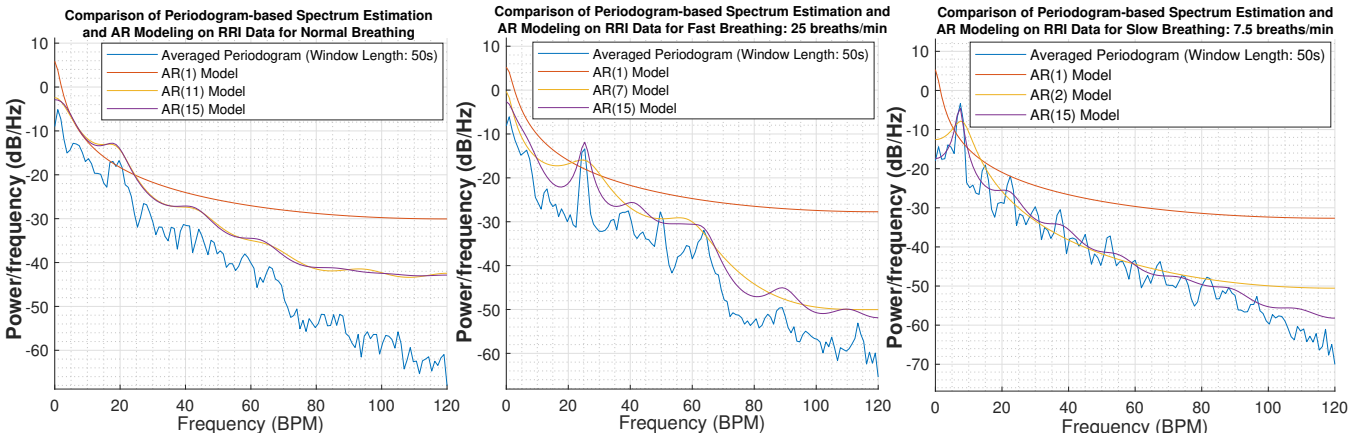

Figure 20: AR model-based PSD estimates of the RRI data using different model order p: AR(p) models are shown and compared to the averaged periodogram estimate with 50s window length for each of the three trials (**left:** normal, **middle:** slow, **right:** fast). The minimum model order at which the frequency spectral peak of the AR model matches that of averaged periodogram is shown.

1.6 Robust Regression

a. Single value decomposition was performed on the noiseless and noisy data matrices $\mathbf{X}$ and $\mathbf{X_{noise}}$. Their respective singular values are shown in Figure 24, where it can be seen that $\mathbf{X}$ has 3 non-zero singular values and hence has rank $n=3$, while $\mathbf{X_{noise}}$ is full rank with 3 dominant singular values corresponding to the eigenvectors spanning the signal subspace, and 7 non-dominant singular values spanning the noise subspace. Since $\mathbf{X}$ spans a lower-dimensional subspace than its noisy counterpart, which suggests that some input variables in $\mathbf{X}$ are co-linear, we propose that a subspace de-noising method could be employed in this context. The squared difference between the singular values of $\mathbf{X}$ and $\mathbf{X_{noise}}$ are also shown.

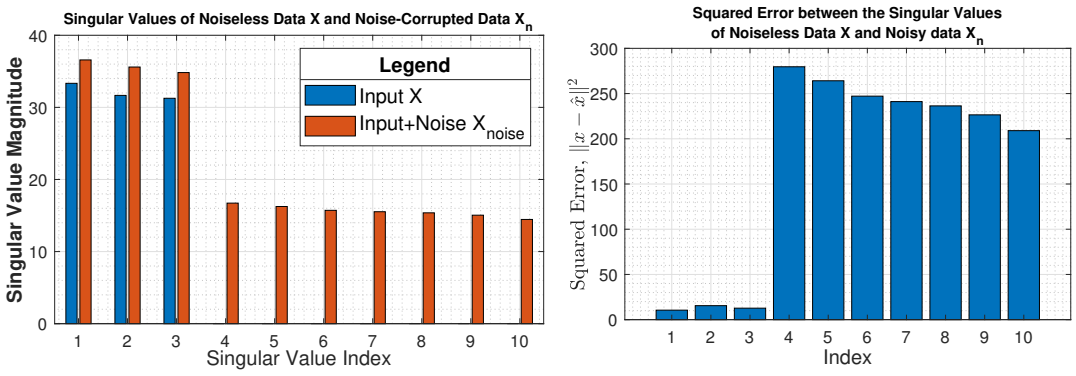

Figure 21: Left: Singular values of the matrices $\mathbf{X}$ and $\mathbf{X}_{noise}$. **Right:** Squared difference in the singular values of $\mathbf{X}$ and $\mathbf{X_{noise}}$.

The squared difference (see Figure 24) shows that under eigenanalysis, the corruption caused by the addition of Gaussian noise to the input data $\mathbf{X}$ modifies by a larger amount the non-dominant singular values of $\mathbf{X}$ than the dominant singular values. Since the squared error is non-zero for the eigenvalues associated with the signal subspace (indexes 1 to 3 here), we propose that the additive noise may not be fully orthogonal to the signal. Due to the non-homogeneous corruption of the signal singular values by the noise singular values, there must exist a noise power level P_w such that the noise singular values are comparable to the signal subspace singular values and the rank of $\mathbf{X}$ becomes hard to identify by simple observation of $\mathbf{X_{noise}}$ via thresholding.

b. We performed dimensionality reduction on $\mathbf{X_{noise}}$ to create its low-rank approximation denoted by $\tilde{\mathbf{X}}_{\mathbf{noise}}$, using its r most significant components, where r is the rank of $\mathbf{X}$. The Frobenius norm, which gives a measure of the 2-norm of the column vectors (variables) of the matrix, was computed for the difference between $\mathbf{X}$ and $\mathbf{X_{noise}}$ and between $\mathbf{X}$ and $\tilde{\mathbf{X}}_{\mathbf{noise}}$. This measure of error was also selected because it has become a standard in tensor decomposition applications, as well as within popular Deep Learning platforms including TensorFlow and PyTorch.

Since the squared error between the singular values of $\mathbf{X}$ and $\mathbf{X_{noise}}$ is concentrated in the non-dominant singular values of index greater than n, we propose that $\tilde{\mathbf{X}}_{noise}$ should approximate the noiseless matrix $\mathbf{X}$ more closely than $\mathbf{X_{noise}}$. Indeed, we can generally define the Frobenius norm of the difference between the noiseless matrix $\mathbf{X}$ and another matrix $\mathbf{Z}$ containing data and noise as the total noise contribution in $\mathbf{Z}$. It follows that we observe that while reducing r from 10 up to n (where we recall that n is the rank of the matrix $\mathbf{X}$) reduces the noise contribution in $\tilde{\mathbf{X}}_{noise}$ and hence renders $\tilde{\mathbf{X}}_{noise}$ more similar to $\mathbf{X}$, further reducing r produces no further denoising and results in an increase in the error between matrices $\mathbf{X}$ and $\tilde{\mathbf{X}}_{noise}$, given that we are now not only suppressing the noise component in $\mathbf{X_{noise}}$ but also the signal component concentrated on the subspace defined by the first three singular values indexed 1 to 3.

Besides, while we have used the Frobenius norm as a compact scalar representation of the total error between two matrices, it is also possible to measure the relative error as the 2-norm of each column in the difference matrix $\mathbf{X}-\tilde{\mathbf{X}}_{noise}$ over the 2-norm of the corresponding column in the original matrix $\mathbf{X}$, to define a relative error per variable. This shows that the denoising effect of PCA results in a reduction and homogenisation of the relative error measure across variables. We suggest that this could be beneficial to some applications, given that no variable is biased relative to another.

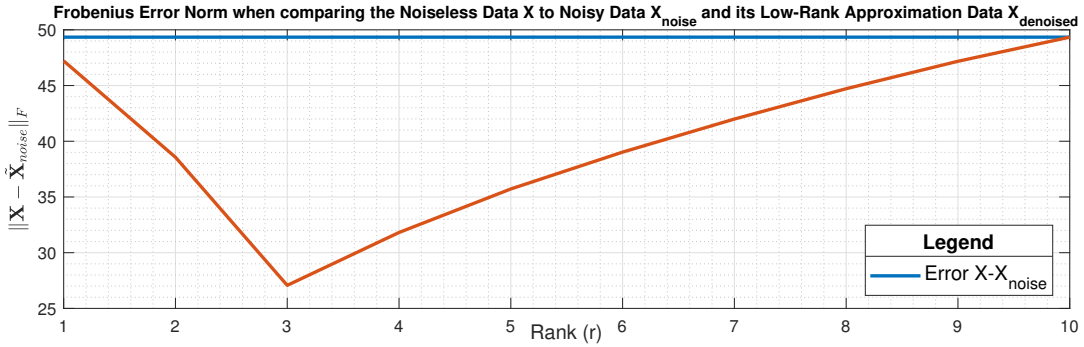

Figure 22: Evolution of the Frobenius norm error between the noiseless matrix $\mathbf{X}$ and the low-rank approximation matrix $_{noise}$, showing minimal error at $r=n$. The Frobenius norm error between $\mathbf{X}$ and $\mathbf{X}_{noise}$ is shown to demonstrate the noise-reduction property of PCA.

c. We perform linear regression to estimate the output matrix $\mathbf{Y}$ from the input matrix $\mathbf{X}$ through the estimation of the parameters $\mathbf{B}$ where we have $\mathbf{Y}=\mathbf{XB}+\mathbf{N}$, where the elements of $\mathbf{N}$ were drawn from a zero-mean Gaussian distribution. The ordinary least square (OLS) and principal component regression (PCR) methods were implemented and compared with respect to the Frobenius norm of the estimation errors $\mathbf{Y}-\mathbf{Y}_{OLS}$ and $\mathbf{Y}-\mathbf{Y}_{PCR}$, as shown from Figure 23. While the OLS solution yields a matrix equation guaranteeing the least squared error between the true and predicted matrices $\mathbf{Y}$ and $\tilde{\mathbf{Y}}$ and involves a possibly intractable matrix inversion, PCR is a dimensionality reduction method which aims to project noisy data on a subspace orthogonal to the noise component.

We observe that for $r \geq n$, the performance of both methods becomes approximately equivalent. At $r=3$, while OLS achieves a 0.35% reduction in estimation error relative to PCR when computed from the training dataset, PCR generalises better as it achieves a 0.49% reduction in estimation error relative to OLS on the testing dataset. Clearly, this suggests PCR performs better, since in a real-world application the test performance is of prime interest.

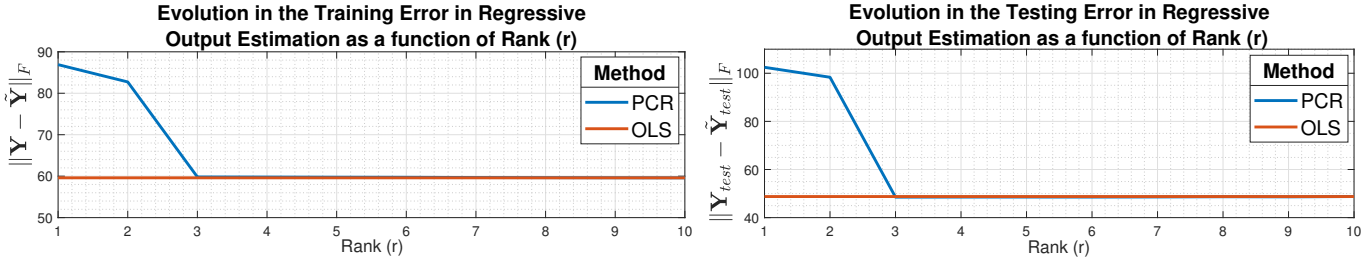

Figure 23: Left: Evolution of the Frobenius norm error between the true data matrix $\mathbf{Y}$ and the predicted matrix $\tilde{\mathbf{Y}}$ across ranks r for the OLS and PCR methods under training data. **Left:** Evolution of the Frobenius norm error between the true data matrix $\mathbf{Y}$ and the predicted matrix $\tilde{\mathbf{Y}}$ across ranks r for the OLS and PCR methods under testing data.

d. Since the performance differences observed between PCR and OLS on training and testing datasets are relatively small, we have chosen to evaluate the regressors performance on a testing dataset consisting of 100 realisations of the process $\mathbf{X}_{noise}$, in order to demonstrate statistical significance of the improved performance of PCR on testing datasets. We indeed observe a maximum MSE reduction of 2.43% when using PCR against OLS for a rank $r=4$. This allows to confirm that PCR outperforms OLS for this noisy regression problem.

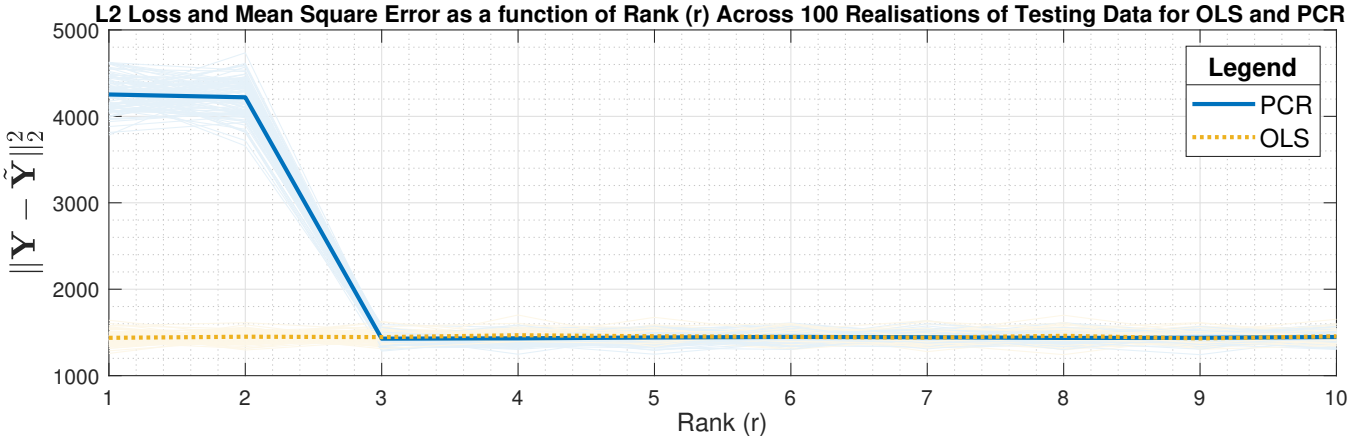

Figure 24: Evolution of the L2 loss error between the true and estimated outputs $\mathbf{Y}$ and $\tilde{\mathbf{Y}}$ under OLS and PCR algorithms as a function of rank (r) over 100 realisations of test data, showing convergence of the methods at $r=n$ and a maximum relative decrease in MSE for $r=4$.

Besides, PCR also avoids the OLS issue of possibly non-tractable solutions, as well as the expensive inverse matrix computation required by the OLS form. Indeed, this inversion would be particularly expensive in a Big Data context with a large number of input variables, while PCR represents a dimensionality reduction method particularly desirable in such Big Data contexts.

2 Adaptive Signal Processing

2.1 The Least Mean Square (LMS) Algorithm

a. Adaptive filters shape white noise into an output $y(n)$, which follows a desired signal $d(n)$ under a criteria of minimisation of the power of the prediction error $e(n)=d(n)-y(n)$ expressed in the form of a cost function $J(n)$. While traditional block filters operate on a whole set of data, dynamic filters offer sequential solutions operating in a real-time online fashion on streaming data, hence adaptive filters can be implemented on hardware constrained in memory resources such as FPGAs, additionally offer promising potential for Big Data applications, and do not rely upon data stationarity.

The general difference equation for a second-order auto-regressive process is given in Equation 11, where $\eta \sim \mathcal{N}(0,\sigma_n^2)$. In the case of a stationary signal $z(n)$, the AR parameters $\mathbf{a}$ can be obtained by the Yule Walker solution in the form $\mathbf{a}=\mathbf{R}_{zz}^{-1}\mathbf{r}_{zz}$. Conversely, in the case of a non-stationary process, $\mathbf{a}$ becomes a vector of time-varying parameters $\mathbf{a}(n)$, prompting the need for adaptive filtering.

$$x(n)=a_1x(n-1)+a_2x(n-2)+\eta(n) \tag{11}$$

In the setting of a system identification configuration, the dynamic filter weights $\mathbf{w}(n)$ are estimates for the AR parameters, $\mathbf{w}(n)=\hat{\mathbf{a}}(n)$, and the desired output $d(n)$ can be the signal $x(n)$, such that at any time point, the filter recursively updates the weights $\mathbf{w}$ to minimise J hence the output y is an estimate for x, $y(n)=\hat{x}(n)$. In the simple case of the Least-Mean-Square (LMS) algorithm, the cost function is half the square of the error $e(n)=d(n)-y(n)$ given by Equation 12, and its minimisation is achieved under a gradient-based method to yield the weight update equation given in Equation 13, where μ is the adaptation gain.

$$J(n)=\frac{1}{2}\Big[e^2(n)\Big]=\frac{1}{2}\Big[d(n)-y(n)\Big]^2 \tag{12}$$

$$\mathbf{w}(n+1)=\mathbf{w}(n)-\mu\nabla_{\mathbf{w}}J=\mathbf{w}(n)+\mu e(n)\mathbf{x}(n) \tag{13}$$

We define the vector $\mathbf{x}$ as $\begin{bmatrix}x(n-1) & x(n-2)\end{bmatrix}^T$ and the parameters $\mathbf{a}$ as $\begin{bmatrix}a_1 & a_2\end{bmatrix}$ where $a_1=0.1$ and $a_2=0.8$, yielding the output equation $y(n)=\hat{x}(n)=\mathbf{a}(n)\mathbf{x}(n)$. Since the roots of the polynomial $P(z)=1-a_1z-a_2z^2$ are given by $[z_1,z_2]\approx[1.0573,-1.1823]$ and lie outside the unit circle ($|z_1|,|z_2|>1$), the AR(2) process $x(n)$ is wide-sense stationary. Although this property of stationarity allows us to derive a criteria on the adaptation gain μ to guarantee LMS convergence, let us recall that data stationarity is not an underlying assumption of the class of adaptive filters to which LMS belongs. We now derive the correlation matrix of $\mathbf{x}$, $\mathbf{R}_{xx}$, starting from the definition of correlation and defining the autocorrelation of $x(n)$ as $r_{xx}(k)=E\{x(n)x(n-k)\}$, where from the wide-sense stationarity property of $x(n)$ we know that r_{xx} is only a function of the time lag k.

$$\mathbf{R}_{xx}=E\Big\{\mathbf{x}(n)\mathbf{x}^T(n)\Big\}=E\Big\{\begin{bmatrix}x(n-1)x(n-1) & x(n-1)x(n-2)\\ x(n-1)x(n-2) & x(n-2)x(n-2)\end{bmatrix}\Big\}=\begin{bmatrix}r_{xx}(0) & r_{xx}(1)\\ r_{xx}(1) & r_{xx}(0)\end{bmatrix} \tag{14}$$

In Equation 15 below, we derive an expression for the process autocorrelation r_{xx}, taking advantage of the linearity property of the expectation operator. Equations 16 derives an expression for a sub-term of Equation 15.

$$\begin{aligned}r_{xx}(k)&=E\Big\{x(n)x(n-k)\Big\}=E\Big\{\Big[a_1x(n-1)x(n-k)+a_2x(n-2)x(n-k)+\eta(n)x(n-k)\Big]\Big\}\\ r_{xx}(k)&=a_1E\Big\{x(n-1)x(n-k)\Big\}+a_2E\Big\{x(n-2)x(n-k)\Big\}+E\Big\{\eta(n)x(n-k)\Big\}=a_1r_{xx}(k-1)+a_2r_{xx}(k-2)+r_{x\eta}(k)\end{aligned} \tag{15}$$

We assume $x(n)$ is uncorrelated with the noise term $\eta(n)$, such that $r_{x\eta}(k)=0$ for $k>0$. In addition, since η is white, its autocorrelation will be equal to the variance of the process η at zero lag, and zero for lags $k>0$, as shown in Equation 16.

$$\begin{aligned}r_{x\eta}(k)&=E\Big\{\Big[a_1x(n-1-k)\eta(n)+a_2x(n-2-k)\eta(n)+\eta(n-k)\eta(n)\Big]\Big\}\\ r_{x\eta}(k)&=a_1r_{x\eta}(k+1)+a_2r_{x\eta}(k+2)+r_{\eta\eta}(k)=\begin{cases}\sigma_n^2 & \text{for } k=0\\ 0 & \text{for } k>0\end{cases}\end{aligned} \tag{16}$$

The symmetry property of the autocorrelation function, under wide-sense stationarity, is expressed as $r_{xx}(k)=r_{xx}(-k)$. We obtain a recursive expression for r_{xx}, given in Equation 17, where $a_1=0.1$, $a_2=0.8$, $\sigma_n^2=0.25$.

$$\begin{aligned}r_{xx}(k)&=a_1r_{xx}(1-k)+a_2r_{xx}(2-k)+r_{x\eta}(k)\\ r_{xx}(2)&=a_1r_{xx}(1)+a_2r_{xx}(0)+r_{x\eta}(2) \;:\; r_{xx}(2)=a_1r_{xx}(1)+a_2r_{xx}(0)\\ r_{xx}(1)&=a_1r_{xx}(0)+a_2r_{xx}(1)+r_{x\eta}(1) \;:\; r_{xx}(1)=\frac{a_1}{1-a_2}r_{xx}(0)=\frac{25}{54}\\ r_{xx}(0)&=a_1r_{xx}(1)+a_2r_{xx}(2)+r_{x\eta}(0) \;:\; r_{xx}(0)=\sigma_n^2\Big[1-a_1^2\Big(\frac{1+a_2}{1-a_2}\Big)-a_2^2\Big]^{-1}=\frac{25}{27}\end{aligned} \tag{17}$$

As a result, the autocorrelation matrix is given in Equation 18. Let us note that one use of the autocorrelation matrix is defining the set of optimum weights satisfying the Wiener-Hopf equations $\mathbf{w_o}=\mathbf{R_{xx}^{-1}r_{dx}}$, where $\mathbf{r_{dx}}$ is the cross-correlation vector between the input and desired signals $x(n)$ and $d(n)$. In addition, by performing the eigenvalue decomposition of $\mathbf{R_{xx}}$, a criteria for convergence of the weights vector under LMS can be formulated in Equation 19, where λ_{max} is the dominant eigenvalue of $\mathbf{R_{xx}}$.

$$\mathbf{R}_{xx}=\begin{bmatrix}r_{xx}(0) & r_{xx}(1)\\ r_{xx}(1) & r_{xx}(0)\end{bmatrix}=\frac{25}{54}\begin{bmatrix}2 & 1\\ 1 & 2\end{bmatrix} \tag{18}$$

From Equation 18, the eigenvalues of $\mathbf{R_{xx}}$ are $\lambda_1,\lambda_2=0.4630,1.3889$, yielding $\lambda_{max}=1.3889$, which is substituted in Equation 19.

$$0<\mu<\frac{2}{\lambda_{max}} \qquad \rightarrow \qquad 0<\mu<1.44 \tag{19}$$

In the range for μ defined above, we have used the fact that in the expression $\mathbf{w}(n+1)=\mathbf{w}(n)-\mu\nabla_{\mathbf{w}}J$, the direction $-\nabla_{\mathbf{w}}J$ is by definition a descent direction for J, hence the line search parameter μ should be positive such that the weight update equation guarantees a reduction in J at every instant n.

b. The LMS adaptive predictor was implemented using $N=1000$ samples of the AR(2) process $x(n)$ defined in Equation 11 for two different adaptation gains $\mu_1=0.01$ and $\mu_2=0.05$. The effect of the hyperparameter μ, also known as the learning rate, was assessed through the learning curves shown in Figure 25, showing the evolution in prediction error power $e^2(n)=(x(n)-\hat{x}(n))^2$ over time steps n obtained for 1 realisation of the process $x(n)$. The large variance initially observed was reduced by averaging learning traces over 100 realisations of $x(n)$, to reveal the underlying learning dynamics.

Since μ_1 and μ_2 satisfy the criteria of Equation 19, both learning traces converge. Furthermore, we observe that the LMS rate of convergence is inversely proportional to the learning rate μ: the learning curve with μ_2 converges on the scale of $n_{c_1}\approx 100$ time steps while with the smaller μ_1 it converges around $n_{c_2}\approx 250$ steps, where $\mu_2>\mu_1$. However, in the steady-state region of the learning curves, the mean $\overline{e^2}$ and standard deviation σ_{e^2} in error power are smallest for the small learning rate μ_1, as summarised in Table 2. This suggests a trade-off in μ between convergence speed and steady-state mean error power and error power variance.

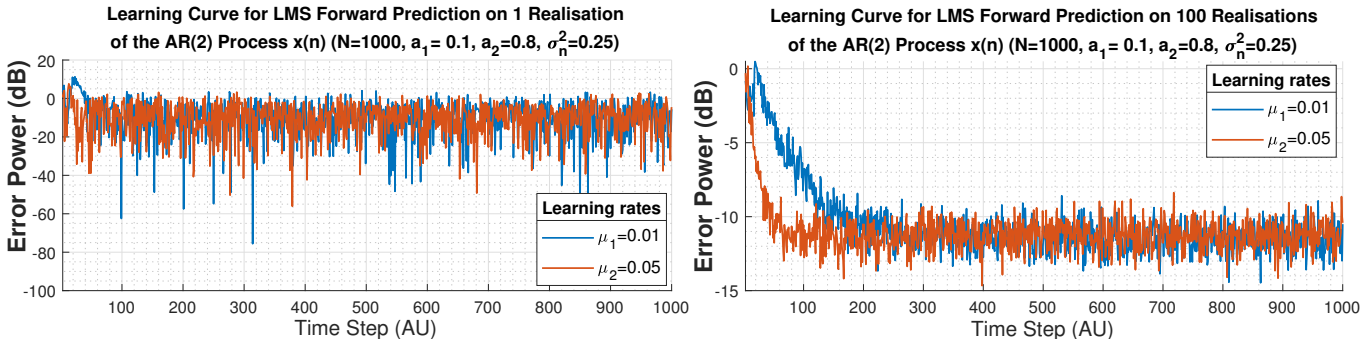

Figure 25: Left: Learning curve for LMS in forward prediction configuration for 1 realisation of the AR(2) process $x(n)$ of Equation 11. **Right:** Learning curve for LMS in forward prediction configuration for 100 realisation of the AR(2) process $x(n)$ of Equation 11.

Learning rate	**Steady-state**		**Transient**
	Mean error (dB)	*Error standard deviation (dB)*	*Convergence (steps)*
$\mu_1=0.01$	$\overline{e^2}_{\mu_1}\approx -11.47$	$\sigma_{e^2}^{\mu_1}\approx 0.86$	$n_{c_1}\approx 100$
$\mu_2=0.05$	$\overline{e^2}_{\mu_2}\approx -11.26$	$\sigma_{e^2}^{\mu_2}\approx 0.93$	$n_{c_2}\approx 250$

Table 2: Measured results for the statistics of the LMS learning curves computed across 100 realisations of the process $x(n)$, as shown in Figure 25. The results illustrate the trade-off between speed of convergence and steady-state performance.

c. We suggest that the increase in steady-state error power observed with increasing learning rates μ is a consequence of the fluctuations of the LMS weights around the optimal values which are directly related to μ (see Equation 13). The excess error introduced by the model instability increasing with μ is quantified via the misadjustment $\mathcal{M}$.

In Table 3, the theoretical LMS misadjustment $\mathcal{M}_t$ is compared to the empirical misadjustment $\mathcal{M}_e$ (see Equation 20), where MSE was obtained from time-averaging the squared error $e^2(n)$ (ensemble-averaged over 100 realisations) over the time steps n from Kn_c to N, where $K=2$ is a safety factor ensuring that we are averaging in the plateau region some steps away from the apparent convergence. We observe that experimental results exceed the theoretical misadjustment $\mathcal{M}_t$ by an amount proportional to the value of the learning rate μ. We propose that the relative error between $\mathcal{M}_e$ and $\mathcal{M}_t$ may be due to the finite machine precision of `MATLAB` computations, resulting in additional MSE error.

Given that $\mathcal{M}$ is the ratio of the excess mean square error to the minimum achievable mean square error, $\mathcal{M}\ll 1$ indicates that the excess error introduced to the model due to weight fluctuations is negligible. As expected, we observe that $\mathcal{M}$ (empirical and theoretical) is smallest for the smaller rate μ_1, hence we conclude that reducing the adaptation gain causes a reduction in misadjustment.

$$\mathcal{M}_t\approx\frac{\mu}{2}\mathrm{Tr}\{\mathbf{R_{xx}}\}=\mu\frac{50}{54}\qquad\text{and}\qquad\mathcal{M}_e=\frac{\mathrm{MSE}}{\sigma_n^2}-1 \tag{20}$$

	$\mathcal{M}_\mathbf{t}$	$\mathcal{M}_\mathbf{e}$	**Relative Error (%)**
μ_1	0.0093	0.0108	16.1
μ_2	0.0463	0.0589	27.2

Table 3: Comparison of the theoretical and empirical misadjustment factors computed from 100 independent trials of the experiment

d. We now formally define the steady-state temporal region for each value of the hyperparameter μ_i as $S_i:\{Kn_{c_i}\to N\}$, and note that in system identification, the filter weights correspond to estimates of the AR parameters a_1 and a_2 such that $\mathbf{w}=\begin{bmatrix}\hat{a}_1 & \hat{a}_2\end{bmatrix}$. As a result, the steady-state value of the filter coefficients was estimated by time-averaging the filter weights over the corresponding interval S_i for each setting μ_i. Results are shown in Figure 26, along with the evolution of the filter weights along time steps n. We observe that both sets of weights converge (again, as expected, since both μ satisfy Equation 19) with different rates of convergence. We again observe a trade-off in terms of μ, given that the smaller step size μ_1 offers reduced steady-state bias in the AR parameter estimates $\hat{a_1}$ and $\hat{a_2}$, while the larger rate μ_2 offers an increased rate of convergence and reduced variance in the estimates (see Table 4).

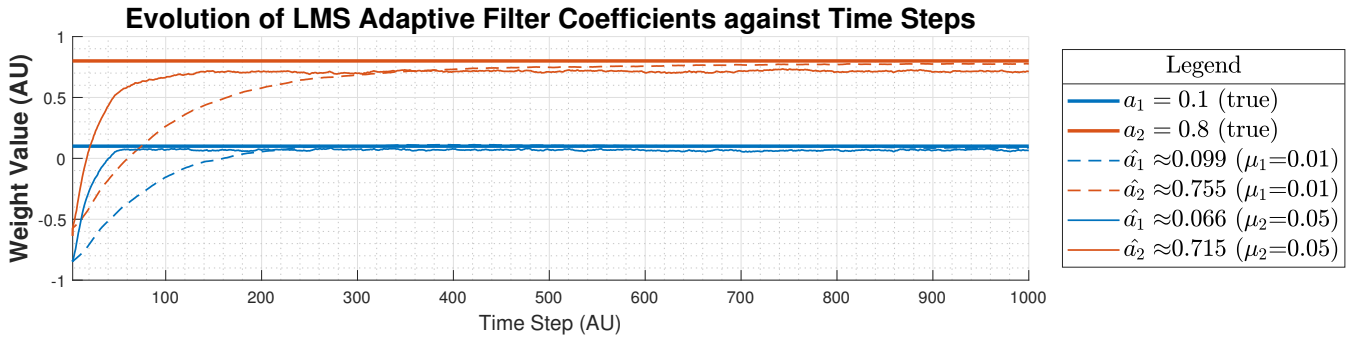

Figure 26: Evolution of the adaptive filter weights across time steps using the LMS algorithm, along with the steady-state AR parameters estimates $\hat{a_1}$ and $\hat{a_2}$ for different learning rates μ. The true AR parameters a_1 and a_2 are shown.

Learning rate	AR parameter	Bias (AU)	Standard deviation (AU)
$\mu_1 = 0.01$	$\hat{a_1}$	$b_1 = \lvert a_1 - \hat{a_1} \rvert \approx 0.001$	$\sigma_{\mu_1, a_1} \approx 0.008$
	$\hat{a_2}$	$b_2 = \lvert a_2 - \hat{a_2} \rvert \approx 0.045$	$\sigma_{\mu_1, a_2} \approx 0.022$
$\mu_2 = 0.05$	$\hat{a_1}$	$b_1 \approx 0.044$	$\sigma_{\mu_2, a_1} \approx 0.005$
	$\hat{a_2}$	$b_2 \approx 0.085$	$\sigma_{\mu_2, a_2} \approx 0.007$

Table 4: Measured statistics characterising the AR parameters estimates obtained from the LMS filter weights shown in Figure 26. The trade-off in μ between speed of convergence and steady-state performance is illustrated.

To conclude from the above, we now have an example of bias-variance trade-off depending on the paramter μ. Besides, we note that the non-zero bias in AR parameter estimates is surprising, given that LMS weights should converge to the Wiener-Hopf solution $\mathbf{w_o} = \mathbf{R_{xx}^{-1}} \mathbf{r_{dx}}$, where $\mathbf{r_{dx}}$ is the cross-correlation vector between the input vector and desired signals $\mathbf{x}$ and $d(n)$, given by Equation 21 where we have used $d(n) = x(n)$ to yield $\mathbf{a} = \mathbf{w_o}$.
Intuitively, we propose that the AR process is defined with 3 independent parameters a_1, a_2 and σ_n^2, while the LMS filter is constructed from only 2 independent weights, therefore the LMS filter does not exhibit sufficient degrees of freedom to completely model the process $x(n)$, resulting in biased estimates $\hat{a_1}$ and $\hat{a_2}$. Besides, we also observe that both the bias and standard deviation in AR parameter estimates are proportional to their corresponding true values.

$$\mathbf{r_{dx}} = \begin{bmatrix} r_{xx}(1) \\ r_{xx}(2) \end{bmatrix} = \begin{bmatrix} \frac{25}{54} \\ \frac{85}{108} \end{bmatrix} \quad \rightarrow \quad \mathbf{w_o} = \mathbf{R_{xx}^{-1}} \mathbf{r_{dx}} = \left(\frac{25}{54} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \right)^{-1} \begin{bmatrix} \frac{25}{54} \\ \frac{85}{108} \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.8 \end{bmatrix} = \mathbf{a} \tag{21}$$

e. We introduce the Leaky LMS algorithm, which attempts to guarantee model stability when the input signal $\mathbf{x}$ has an autocorrelation matrix with eigenvalues equal to zero. The Leaky LMS minimises the cost function J_2 expressed in Equation 23, where the error signal $e(n)$ is expressed in Equation 22. The gradient descent algorithm yields the weight update expression shown in Equation 25, where $\nabla_{\mathbf{w}} J_2$ denotes the gradient with respect to the weight vector $\mathbf{w}$ of the cost function J_2, expressed in Equation 24 (where the time index n has been omitted for simplicity). Equation 25 verifies the Leaky LMS weight update equation.

$$e(n) = d(n) - y(n) = x(n) - \hat{x}(n) = x(n) - \mathbf{w}^T(n)\mathbf{x}(n) \tag{22}$$

$$J_2(n) = \frac{1}{2}\left(e^2(n) + \gamma ||\mathbf{w}(n)||_2^2\right) = \frac{1}{2}\left(\left(x(n) - \mathbf{w}^T(n)\mathbf{x}(n)\right)^2 + \gamma ||\mathbf{w}(n)||_2^2\right) \tag{23}$$

$$\nabla_{\mathbf{w}} J_2 = \frac{1}{2}\left(-2\mathbf{x}(x - \mathbf{w}^T\mathbf{x}) + 2\gamma\mathbf{w}\right) = \gamma\mathbf{w} - e\mathbf{x} \tag{24}$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\nabla_{\mathbf{w}} J_2(n) = \mathbf{w}(n) - \mu\left(\gamma\mathbf{w}(n) - e(n)\mathbf{x}(n)\right) = \left(1 - \mu\gamma\right)\mathbf{w}(n) + \mu e(n)\mathbf{x}(n) \quad \texttt{QED} \tag{25}$$

f. The Leaky LMS algorithm was implemented by modifying the original LMS weight update expression (Equation 13) with the newly derived update equation (Equation 25). Different values of hyper-parameters μ (adaptation gain) and γ (leakage coefficient) were explored, as shown in Figure 27. While μ adjusts the rate of convergence of the learning curves obtained as previously observed, increasing γ widens the bias (undershoot) in AR parameters estimation to an extent proportional to the true parameter value.

Let us numerically investigate the above claims. Let $\mathbf{b}(\gamma) = \begin{bmatrix} b_{1,\gamma} & b_{2,\gamma} \end{bmatrix}$ give the bias vector for a_1 and a_2 for $\mu_2 = 0.05$ as a function of the leakage coefficient γ, we have $\mathbf{b}(0.1) = \begin{bmatrix} 0.01 & 0.18 \end{bmatrix}$, $\mathbf{b}(0.5) = \begin{bmatrix} 0.013 & 0.385 \end{bmatrix}$ and $\mathbf{b}(0.9) = \begin{bmatrix} 0.004 & 0.482 \end{bmatrix}$. This shows that the bias increase caused by an increase in hyperparameter γ is only consistent for the large estimate $\hat{a_2}$. Besides, it can be seen mathematically from Equation 25 and graphically from Figure 27 that increasing the learning rate μ also increases the negative bias in the AR parameter estimates. Similarly, the standard deviation vectors σ are given, for μ_2, by $\sigma(0.1) = \begin{bmatrix} 0.009 & 0.007 \end{bmatrix}$, $\sigma(0.5) = \begin{bmatrix} 0.011 & 0.012 \end{bmatrix}$ and $\sigma(0.9) = \begin{bmatrix} 0.013 & 0.01 \end{bmatrix}$. Increasing the leakage term γ hence also increases steady-state variance of the estimates.

Let us now analytically explain our observations. The offline, block-filter optimal solution to the problem of AR parameter estimation is given by the Wiener-Hopf filter, where the optimal weight vector $\mathbf{w_o}$ is given by Equation 26, where $\mathbf{R}$ is the autocorrelation matrix of the input signal $\mathbf{x}$ and $\mathbf{p}$ is the cross-correlation between the input signal $\mathbf{x}$ and the desired signal $\mathbf{d}$. If the learning rate μ satisfies the condition of Equation 19, then the LMS filter weights will converge to the Wiener solution, provided that $\mathbf{R}^{-1}$ exists, that is, provided that all the eigenvalues of $\mathbf{R}$ are non-zero. It can be shown that the Leaky LMS algorithm can be characterised by a modified autocorrelation matrix $\mathbf{R}' = \mathbf{R} + \gamma\mathbf{I}$ which has no zero eigenvalue, such that the

modified filter converges to the optimal set of weights $\mathbf{w}'_{\mathbf{o}}$. Again, referring to Equation 21 it can be shown that $\mathbf{w}'_{\mathbf{o}} \neq \mathbf{a}$, and furthermore that for every index i we have $w'_{o,i} < a_i$, which is confirmed in Figure 27.

$$\mathbf{w_o} = \mathbf{R}^{-1}\mathbf{p} \;\rightarrow\; \mathbf{w}'_{\mathbf{o}} = \mathbf{R'}^{-1}\mathbf{p} = \Big(\mathbf{R}+\gamma\mathbf{I}\Big)^{-1}\mathbf{p} \tag{26}$$

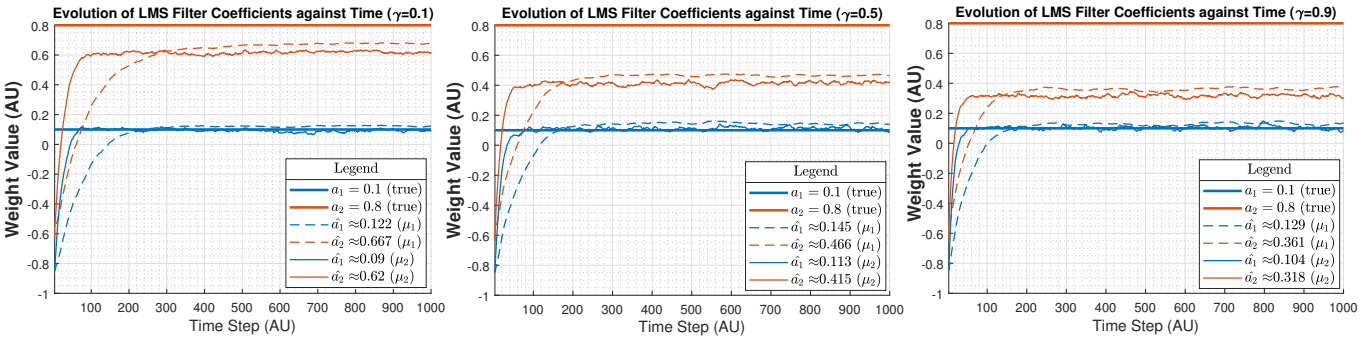

Figure 27: Evolution of the adaptive filter weights across time steps under the LMS algorithm for different values of learning rate μ and leakage coefficient γ, along with the steady-state AR parameters estimates $\hat{a_1}$ and $\hat{a_2}$ for different learning rates μ.

2.2 Adaptive Step Sizes

a. The step-size trade-off between convergence speed and reduction in steady-state error and variance previously identified prompts the need for an adaptive learning rate μ, initially large and progressively decreasing to achieve fast convergence and minimal steady-state error. Within the class of variable step-size (VSS) algorithms, we will implement the gradient adaptive step size (GASS) technique. To minimise $\nabla_\mu J$, we obtain the learning rate update expression given by Equation 27, where the term ψ can take several form which approximate the true $\psi = \nabla_\mu \mathbf{w}$. We will study three GASS algorithms: Benveniste, Ang & Farhang, and Matthews & Xie.

$$\mu(n+1) = \mu(n) + \rho e(n)\mathbf{x}^T(n)\psi(n) \tag{27}$$

The GASS algorithms were tested in system identification setting on a real-valued MA(1) process given by Equation 28. In this setting, we have a teaching signal $d(n) = x(n)$, a filter output $y(n) = \hat{x}(n)$ and a design vector $\mathbf{x} = \begin{bmatrix}\eta(n) & \eta(n-1)\end{bmatrix}$, yielding $y(n) = \hat{x}(n) = \mathbf{w}^T\mathbf{x}$. We note that in the setting of system identification, the input weight vector includes the value of the driving input at the present instant n, $\eta(n)$, as opposed to the previously considered forward prediction example where the input vector could only contain past values of the driving input $x(n-k)$ with $k > 0$ from which the present value $x(n)$ was estimated. We suggest that this justifies why in the present case of an MA(1) process identification the filter weights exhibit zero steady-state bias, as illustrated in Figure 28.

The performance of each GASS algorithm is assessed via the study of its associated learning curve and the evolution of the bias in the estimate of the MA parameter a_1, as shown in Figure 28, where for each GASS algorithm, the learning rate was initialised at μ_2, to allow for fair comparison with the fixed-learning-rate-LMS. The learning curves observed confirm that adaptive learning rates allow for faster convergence than static LMS algorithms. This is also confirmed by the evolution of the filter weight error defined as $\tilde{w} = w_o - w(n)$: while all weight traces converge to the true MA(1) parameter a_1, GASS algorithms offer an increased rate of convergence. We conclude that GASS algorithms outperform their fixed-learning-rate counterpart by solving the trade-off between convergence speed and steady-state error power reduction and yield faster convergence at reduced steady-state error powers, as summarised in Table 5. Besides, given that $\mathcal{M}$ is proportional to μ, we also expect GASS algorithms to deliver a reduced steady-state misadjustment caused by the progressive learning rate decrease.

$$x(n) = a_1\eta(n-1) + \eta(n) \;\texttt{where}\; a_1 = 0.9 \;\texttt{and}\; \eta \sim \mathcal{N}(0, 0.5) \tag{28}$$

Algorithm	**LMS ($\mu=0.05$)**	**LMS ($\mu=0.01$)**	**Benveniste**	**Ang**	**Matthews**
Convergence (steps)	200	100	40	50	80
Steady-state error (dB)	−322.4	−313.1	−330.6	−327.7	−325.8

Table 5: Summary of measured statistics from the learning curves and weight error traces from Figure 28, illustrating the GASS algorithms outperform static LMS algorithms both in rate of convergence of the weights and reduced steady-state error power.

We now compare different GASS algorithms via the learning curves obtained. While Benveniste gives the exact expression for ψ, Ang & Farhang uses a low-pass filter with a fixed coefficient α and Matthews & Xie further simplifies ψ by assuming $\alpha = 0$. Unsurprisingly, the rate of convergence offered by each GASS algorithm is proportional to the computational complexity and exactness in the expression for ψ: Beneviste (convergence after $n_b \approx 600$ time-steps) outperforms Ang & Farhang ($n_{af} = 650$), which itself outperforms Matthews & Xie. Increased complexity also offers reduced error power, as illustrated in Table 5.

To conclude, GASS algorithms offer reduced steady-state error and increased rate of convergence at the cost of high computational cost and with an extra hyperparameter ρ to define. Within GASS algorithms, performance is similarly proportional to algorithmic complexity, with Benveniste offering optimal results.

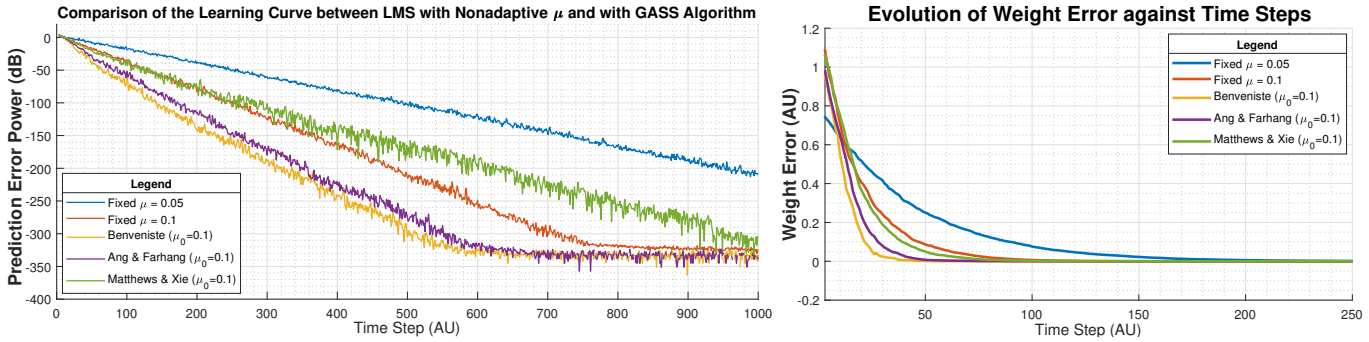

Figure 28: Right: Learning curves for standard LMS and Beneviste, Ang & Farhang and Matthews & Xie GASS algorithms implemented in system identification on the MA(1) process of Equation 28. **Left:** Weight error evolution for standard LMS and Beneviste, Ang & Farhang and Matthews & Xie GASS algorithms implemented in system identification on the MA(1) process of Equation 28. The results were averaged over 100 realisations of signals $x(n)$ of $N=1000$ samples each. For all GASS algorithms we have selected $\rho=5\times10^{-3}$ while for Ang & Farhang we have set the low-pass filter parameter $\alpha=0.8$.

b. We wish to compare the LMS weight update equation given in Equation 30, based on the *a posteriori* error defined as $e_p(n)=d(n)-\mathbf{x}^T(n)\mathbf{w}(n+1)$, with the NLMS update equation given in Equation 29, to identify the relationship between β, ϵ and μ. We start with the LMS weight update equation, where we pre-multiply both sides by $-\mathbf{x}^T(n)$ and add $d(n)$, such as to construct the *a posteriori* error on the LHS. The second part of Equation 29 is obtained by visual comparison of the LMS and NLMS weight update equations.

$$\mathbf{w}(n+1)=\mathbf{w}(n)+\frac{\beta}{\epsilon+\mathbf{x}^T(n)\mathbf{x}(n)}e(n)\mathbf{x}(n) \;\rightarrow\; \frac{\beta}{\epsilon+\mathbf{x}^T(n)\mathbf{x}(n)}e(n)=\mu e_p(n) \tag{29}$$

$$\mathbf{w}(n+1)=\mathbf{w}(n)+\mu e_p(n)\mathbf{x}(n) \;\rightarrow\; e_p(n)=d(n)-\mathbf{x}^T(n)\mathbf{w}(n+1)=d(n)-\mathbf{x}^T(n)\mathbf{w}(n)-\mu e_p(n)\mathbf{x}^T(n)\mathbf{x}(n) \tag{30}$$

In Equation 30, the *a priori* error is defined as $e(n)=d(n)-\mathbf{x}^T(n)\mathbf{w}(n)$ and since $\mathbf{x}$ is a column vector we have $\mathbf{x}^T(n)\mathbf{x}(n)=||\mathbf{x}(n)||^2$, therefore Equation 30 yields the *a posteriori* error $e_p(n)$ as a function of the *a priori* error $e(n)$, as shown in Equation 31. Now substituting Equation 31 in Equation 29, we obtain Equation 32, where by visual inspection we have $\beta=1$ and $\epsilon=\mu^{-1}$.

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

$$\frac{\beta}{\epsilon+||\mathbf{x}(n)||^2}e(n)=\mu\frac{e(n)}{1+\mu||\mathbf{x}(n)||^2} \;\rightarrow\; \frac{\beta}{\epsilon+||\mathbf{x}(n)||^2}=\frac{1}{\mu^{-1}+||\mathbf{x}(n)||^2} \quad \texttt{QED} \tag{32}$$

c. The normalised LMS (NLMS) filter was implemented with an adaptive regularisation factor $\epsilon(n)$ whose dynamics are controlled by a generalised normalised gradient descent (GNGD) algorithm. The NLMS filter was evaluated in system identification configuration on the process $x(n)$ of Equation 28 and compared to the LMS filter with Benveniste GASS by assessing the evolution of the weight error. Figure 29 shows that the hyper-parameters μ_0 and ρ for each filter were set to approximate the optimal hyper-parameter values ensuring model stability, such as to compare the best-case-scenario performance of each algorithm. Indeed, a small variation away from these optimal hyper-parameters results in model instability. It can be seen that the NLMS method with GNGD algorithm outperforms LMS with Benveniste GASS both in terms of rate of convergence, and in terms of steady-state weight error. We measured steady-state weight errors of -247dB for NLMS with GNGD and -151dB for Benveniste GASS LMS.

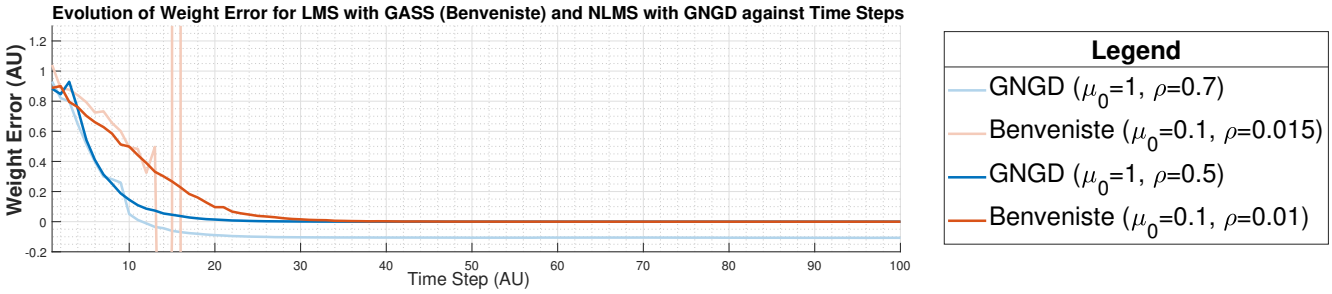

Figure 29: Weight error evolution for NLMS and LMS with Benveniste GASS implemented in system identification on the MA(1) process of Equation 28. The results were averaged over 100 realisations of signals $x(n)$ of $N=1000$ samples each. The less opaque traces illustrate the unstable weight behaviour when a small variation is applied to one of the parameters μ_0 and ρ.

Let us note that from an optimisation perspective, the normalised LMS and LMS with adaptive μ only differ in the line search parameter and exhibit identical search directions at every instant n. While the normalised LMS uses a line search parameter normalised to the input vector $\mathbf{x}$, the GASS-LMS method solves an exact line search problem at each instant n, and we therefore expect the latter to be computationally heavier. This is confirmed in the derivations below.

The NLMS GNGD and the Benveniste's GASS algorithm only differ by their learning rate (line search parameter) component, given respectively by $\frac{\beta}{\epsilon+\mathbf{x}^T(n)\mathbf{x}(n)}$, which involves the computation of $\epsilon(n)$, and $\mu(n)$, which involves the computation of $\psi(n)$.

Although expressing the exact number of operations required by each algorithm would be tedious, we can easily compare their computational complexity by noting that updating ψ involves the computation of an outer product $\mathbf{x}\mathbf{x}^T$, as well as a matrix product give by the expression $\big[I-\mu(n-1)\mathbf{x}(n-1)\mathbf{x}(n-1)^T\big]\psi(n-1)$. Let P be the model order used by the LMS/NLMS filter (that is, the number of delays in the AR or MA underlying expression) such that $\mathbf{x}$ has length P, then the outer product consists of P^2 operations while the matrix multiplication represents $P(2P-1)$ operations, which both have quadratic complexity. In contrast, the update equation for ϵ simply involves the computation of inner products of vectors of size P requiring $2P-1$ operations, and scalar multiplications and additions, which cost one operation each. As a result, while for Benveniste's algorithm we obtain $\mathcal{O}(\mathbf{P^2})$, for GNGD we have $\mathcal{O}(\mathbf{P})$. Assuming a high model order $P\gg 1$, GNGD would result systematically less computationally expensive than Benveniste's while also offering an improved performance in terms of convergence speed and steady-state weight error reduction at the optimal hyper-parameter values.

2.3 Adaptive Noise Cancellation

a. We now consider de-noising adaptive filtering configurations, starting with the Adaptive Line Enhancer (ALE), which consists of a delay operator and a linear predictor which in our case will use the LMS algorithm. To investigate what is the minimum value for the delay Δ that may be used in ALE for a filter length of $M>1$, we first expand the expression of the filter Mean Square Error (MSE), defined in Equation 33, where the noise-corrupted signal $s(n)$ is the sum of a clean sine wave $x(n)$ and a coloured noise term $\eta(n)$.

$$\begin{aligned}
&\mathbb{E}\Big\{e^2(n)\Big\}=\mathbb{E}\Big\{\big(s(n)-\hat{x}(n,\Delta)\big)^2\Big\}=\mathbb{E}\Big\{\big(x(n)+\eta(n)-\hat{x}(n,\Delta)\big)^2\Big\}\\
&\mathbb{E}\Big\{e^2(n)\Big\}=\mathbb{E}\Big\{\eta^2(n)\Big\}+2\mathbb{E}\Big\{\eta(n)\big(x(n)-\hat{x}(n,\Delta)\big)\Big\}+\mathbb{E}\Big\{\big(x(n)-\hat{x}(n,\Delta)\big)^2\Big\}\\
&\mathbb{E}\Big\{e^2(n)\Big\}=\mathbb{E}\Big\{\eta^2(n)\Big\}+2\mathbb{E}\Big\{\eta(n)x(n)\Big\}-2\mathbb{E}\Big\{\eta(n)\hat{x}(n,\Delta)\Big\}+\texttt{MSPE}
\end{aligned} \tag{33}$$

The delay Δ must be chosen such that noise in the signal $s(n)$ and the predictor input $\mathbf{u}(n)$ are uncorrelated, so that only the noise is suppressed by the linear predictor. In other terms, we wish to find the minimum delay Δ for which $\eta(n)$ and $\mathbf{u}(n)$, the Δ-delayed version of η, are uncorrelated, such that the MSE depends only on the noise η, since $e(n)=\eta(n)$ yields $\hat{x}=x$. We therefore wish to study how varying Δ affects the contribution of delayed noise samples $\eta(n-\Delta)$ to MSE, and minimise this contribution. The first MSE term in Equation 33 is the noise power, which is unaffected by Δ, and hence can be ignored. Similarly, the second MSE term vanishes assuming the signal $x(n)$ and the noise $\eta(n)$ are uncorrelated. The last MSE term is the Mean Square Prediction Error (MSPE), which depends on $\eta(n-\Delta)$ through $\hat{x}(n,\Delta)$. However, this term is not a function of the noise autocorrelation and hence we cannot express the effect of changing Δ on this term. We therefore wish to minimise the third MSE term, which contains $\eta(n)$ and a Δ-delayed version of the noise $\eta(n,\Delta)$ through $\hat{x}(n,\Delta)$.

$$\begin{aligned}
&\mathbb{E}\Big\{\eta(n)\hat{x}(n,\Delta)\Big\}=\mathbb{E}\Big\{\eta(n)\mathbf{w}^T\mathbf{u}(n,\Delta)\Big\}=\mathbb{E}\Big\{\eta(n)\sum_{i=0}^{M-1}w_i(n)s(n-\Delta-i)\Big\}\\
&\mathbb{E}\Big\{\eta(n)\hat{x}(n,\Delta)\Big\}=\mathbb{E}\Big\{\eta(n)\sum_{i=1}^{M-1}w_i(n)x(n-\Delta-i)\Big\}+\mathbb{E}\Big\{\eta(n)\sum_{i=0}^{M-1}w_i(n)\eta(n-\Delta-i)\Big\}
\end{aligned} \tag{34}$$

We assume that the signal $x(n)$ is uncorrelated with the noise $\eta(n)$, such that the correlation cross-terms $x(n)\eta(n+m)$ vanish. In addition, since $v(n)$ is white Gaussian noise, its autocorrelation $r_{vv}(m)=\mathbb{E}\{v(n)v(n-m)\}$ will vanish for $n>0$. Let us take the worst case scenario $i=0$ which we substitute in $r_{vv}(\Delta+i-2)$, yielding $\Delta-2>0$ where we finally have $\Delta>2$.

$$\begin{aligned}
&\mathbb{E}\Big\{\eta(n)\hat{x}(n,\Delta)\Big\}=\mathbb{E}\Big\{\eta(n)\sum_{i=0}^{M-1}w_i(n)\eta(n-\Delta-i)\Big\}=\mathbb{E}\Big\{\sum_{i=0}^{M-1}w_i(n)\eta(n)\eta(n-\Delta-i)\Big\}\\
&\mathbb{E}\Big\{\eta(n)\hat{x}(n,\Delta)\Big\}=\mathbb{E}\Big\{\sum_{i=0}^{M-1}w_i(n)\big[v(n)+0.5v(n-2)\big]\big[v(n-\Delta-i)+0.5v(n-2-\Delta-i)\big]\Big\}\\
&\mathbb{E}\Big\{\eta(n)\hat{x}(n,\Delta)\Big\}=\sum_{i=0}^{M-1}w_i(n)\big[1.25\times r_{vv}(\Delta+i)+0.5\times r_{vv}(\Delta+i+2)+0.5\times r_{vv}(\Delta+i-2)\big]\\
&\min_{\Delta>0}\mathbb{E}\Big\{\eta(n)\hat{x}(n,\Delta)\Big\}\ \rightarrow\ \Delta>2
\end{aligned} \tag{35}$$

Since η is given by an MA(2) process, the result $\Delta_{min}=3$ is expected. This theoretical result is confirmed by the simulations displayed in Figure 30, showing an overlay plot of $R=50$ realisations of the noisy sinusoids $s(n)$ and the clean sinusoids $x(n)$, along with the filter output $\hat{x}(n)$, which in the current configuration is an estimate for the noiseless $x(n)$. A clear performance improvement, marked by a reduction of noise in $\hat{x}(n)$, can be observed for $\Delta\geq\Delta_{min}$. The results were obtained from a `MATLAB` program for the adaptive line enhancement using the LMS algorithm which can be found in the zip file attached to this coursework.

b. The effect of hyper-parameters M (filter order) and Δ (delay parameter) on the Mean Squared Prediction Error (MSPE) of the ALE was investigated, where we take MSPE as a measure of ALE de-noising performance. Overall, MSPE was minimised for the $\Delta-M$ pair [3,5], and we observe a convex dependency of the MSPE both on the delay Δ and on the filter order M, as can be seen from the error surface shown in Figure 32.

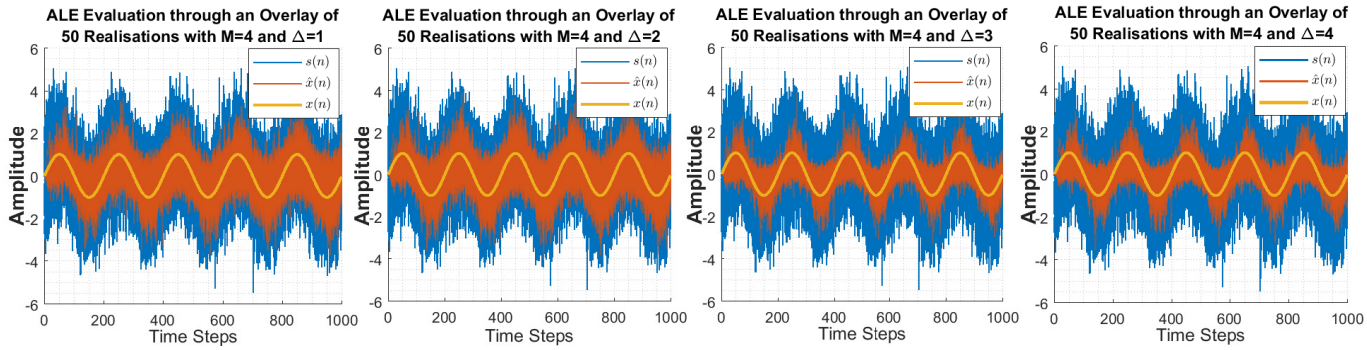

Figure 30: Overlay plots of ALE output $\hat{x}(n)$ as implemented on 50 realisations of the noisy signal $s(n)$, where the clean signal $x(n)$ is shown for reference, for different values of hyperparameter Δ. The results demonstrate an improvement in noise suppression for $\Delta \geq \Delta_{min}$.

As expected, MSPE is minimised for $\Delta = \Delta_{min}$, given that at lower delays the coloured noise in the noisy input signal $s(n)$ is still correlated the delayed version of itself in the filter input vector $\mathbf{u}$. In contrast, for larger delays, when Δ becomes of the order of magnitude of the inverse of the instantaneous signal frequency, which is equivalent to Δ becoming comparable to the instantaneous signal period (i.e. when Δ is large relative to how fast the signal is changing), then further increasing Δ causes an increase in MSPE, denoting a decrease in de-noising performance. Indeed, at large delays, the resulting time-shift between the input and output of the ALE filter results in an increase in MSPE, since the output at an instant n is in fact an estimate for $x(n-\Delta)$. An overlay plot and the averaged equivalent for 50 realisations of the noisy input $s(n)$ are shown in Figure 31, illustrating the time-shift at $\Delta = 25$. The optimal $\Delta = 3$ ensures de-correlation of the noise with itself and minimal time-shift.

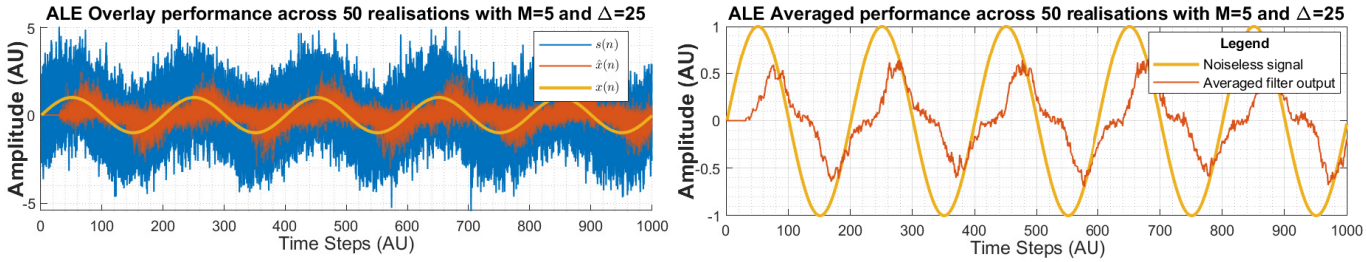

Figure 31: Left: Overlay plots of ALE output $\hat{x}(n)$ as implemented on 50 realisations of the noisy signal $s(n)$, where the clean signal $x(n)$ is shown for reference, for $\Delta = 25$. **Right:** Averaged plot of ALE output $\hat{x}(n)$ as implemented on 50 realisations of the noisy signal $s(n)$, where the clean signal $x(n)$ is shown for reference, for $\Delta = 25$.

At small filter order $M < 3$, the filter fails to offer sufficient degrees of freedom to model the signal of interest $x(n)$, while similarly at large model orders $M > 5$ over-fitting occurs, causing an increase in MSPE as the linear predictor attempts to model the noise component of $s(n)$, $\eta(n)$. The curve of MSPE against M appears to plateau in the region $M : [3,5]$, and since computational complexity increases with model order, we suggest that the optimal filter order is the one that guarantees low MSPE at the lowest possible M, yielding $M_{opt} = 3$. Although the absolute minim MSPE is achieved for $M = 5$, we argue that the error reduction is not enough to justify the additional computational complexity required by the larger model order.

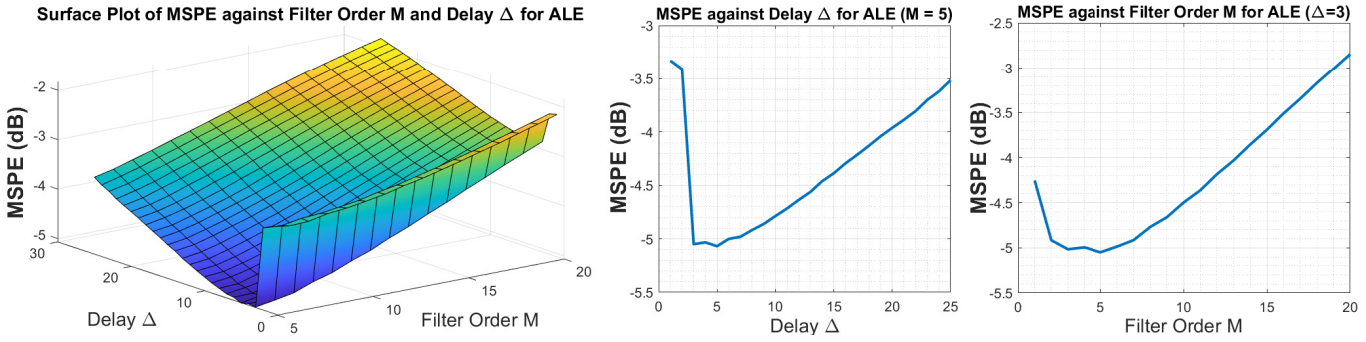

Figure 32: Left: Surface plot of MSPE for ALE against Filter Order M and Delay Δ. **Middle:** Plot of MSPE against delay Δ for a fixed filter order M. **Right:** Plot of MSPE against filter order M for a fixed delay Δ. MSPE values are averaged over 50 realisations.

c. We now similarly write a `MATLAB` code for the Adaptive Noise-Cancelling (ANC) filter, which can be found in the zip file attached to this coursework. We compare the relative performance of ANC and ALE in de-noising the noisy sinusoid $s(n)$, where the MSPE is used as performance metric for comparison. The simulations displayed in Figure 33 show an overlay plot of $R = 50$ realisations of the noisy sinusoids $s(n)$ and the clean sinusoids $x(n)$, along with the filter outputs $\hat{x}(n)$ obtained using ALE and ANC algorithms. We stress that the same inout signals $s(n)$ were fed to both filters, to enable unbiased comparison.

In the case of the ALE filter, the hyper-parameters were set at the optimal $\Delta_{opt} = 3$ and $M_{opt} = 3$. In the case of ANC, the filter inputs are the noise-corrupted signal $s(n) = x(n) + \eta(n)$, where η is the primary noise, and the secondary noise $\epsilon(n) = v(n)$, where $v(n)$ is white Gaussian noise with zero mean and variance 1. We know the primary and secondary noise are related

through $\eta(n)=v(n)+0.5v(n-2)$, and will therefore select a filter order for ANC of $M=3$, since η is a second-order MA process. We note that, in practice, such prior knowledge of the relationship between the primary and secondary noise is not available, hence a method must be established to estimate a suitable filter order M. In addition, to enable meaningful comparison between ANC and ALE, the learning rate for both algorithms was set to $\mu=0.01$.

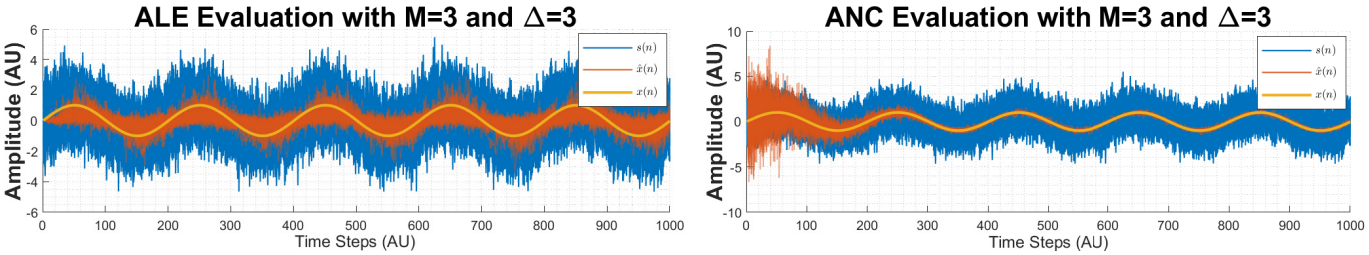

Figure 33: Left: Overlay plots of ALE output $\hat{x}(n)$ as implemented on 50 realisations of the noisy signal $s(n)$, where the clean signal $x(n)$ is shown for reference. **Right:** Overlay plots of ANC error $\hat{x}(n)$ as implemented on 50 realisations of the noisy signal $s(n)$, where the clean signal $x(n)$ is shown for reference. It can be seen that after a transient period, ANC outperforms ALE.

In Figure 33 above, the estimates $\hat{x}(n)$ are obtained from the output of ALE and the error signal from ANC. The MSPE for ALE and ANC in the aforementioned setting are measured as -5.1dB and -6.5dB respectively, demonstrating an improved performance under ANC. While during its transient response, on the first 300 steps, ANC performs more poorly than ALE, it progressively converges and exhibits a steady-state error power much inferior to that of ALE. This is illustrated in time domain in Figure 33, where it can be seen that after a transient period ANC manages to efficiently track $x(n)$. Lastly, Figure 34 shows the estimates $\hat{x}$ for ALE and ANC, averaged over 50 realisations, which further demonstrates that the averaged de-noising performance of ANC is superior to that of ALE.

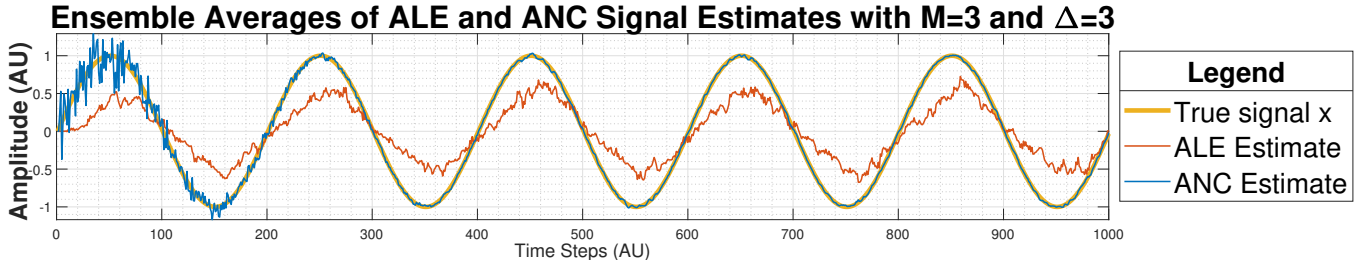

Figure 34: Ensemble averaged plots of ALE output and ANC error $\hat{x}(n)$ as implemented on 50 realisations of the noisy signal $s(n)$, where the clean signal $x(n)$ is shown for reference. ANC outperforms ALE in de-noising performance and tracking of $x(n)$.

d. We aim to remove the 50Hz mains disturbance from a single-channel EEG recording `POz` denoted by $s(n)$ using the ANC filtering configuration with a synthetic reference input $\eta(n)$ composed of a sinusoid of 50Hz corrupted by white Gaussian noise. Since the scale of the amplitude of η should ideally approximately match the scale of the amplitude of the 50Hz mains component in the EEG signal which is unknown, we pre-process the EEG data via standardisation to zero mean and variance 1 with the help of the function `zscore`. As a result, we can select an amplitude of 1 for $\eta(n)$, and we have selected for η a signal-to-noise ratio of the 50Hz sine wave to WGN of 20dB. The method described in this section can be used in a wide variety of situations, given that 50Hz noise perturbation is extremely common.

Let us now select the parameters for `spectrogram` computation. We recall that Equation 9 demonstrates that the width of the main spectral lobe equals twice the inverse of the duration of the time-domain window, while reducing window size also allows for averaging over more windows, which in turn reduces PSD variability. Let us assume that we require a frequency resolution (i.e. main lobe width) of $0.5Hz$, then the window length must be $4f_s$, where f_s is the signal sampling frequency. This yields a window length of $L=4800$ samples. The frequency resolution was chosen such that we could easily characterise the effect of the ANC filter in the vicinity of the 50Hz mains frequency.

Now, let K be the number of DFT points used to compute the `spectrogram`, and let N be the length of s, then we know that selecting $K>N$ requires zero-padding of $s(n)$ in time domain, while for $K<N$ the frequency resolution of the PSD estimate will be given by $\frac{f_s}{2K}$. To avoid incoherent sampling, we set $\frac{f_s}{2K} \ll 0.5$Hz, such that the we represent many points per spectral lobe, and choose $K=12000$, where a trade-off between computational cost and frequency resolution is met (indeed, increasing K increases computational cost). Lastly, Hamming windows were selected with 50% overlap, which is the default setting in the `MATLAB` function `spectrogram`.

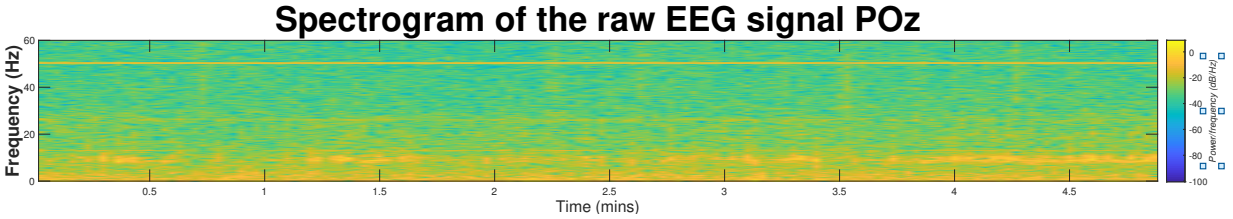

Figure 35: Spectrogram showing the time-frequency spectrum of `POz`, with a clear 50Hz component present for all the signal duration.

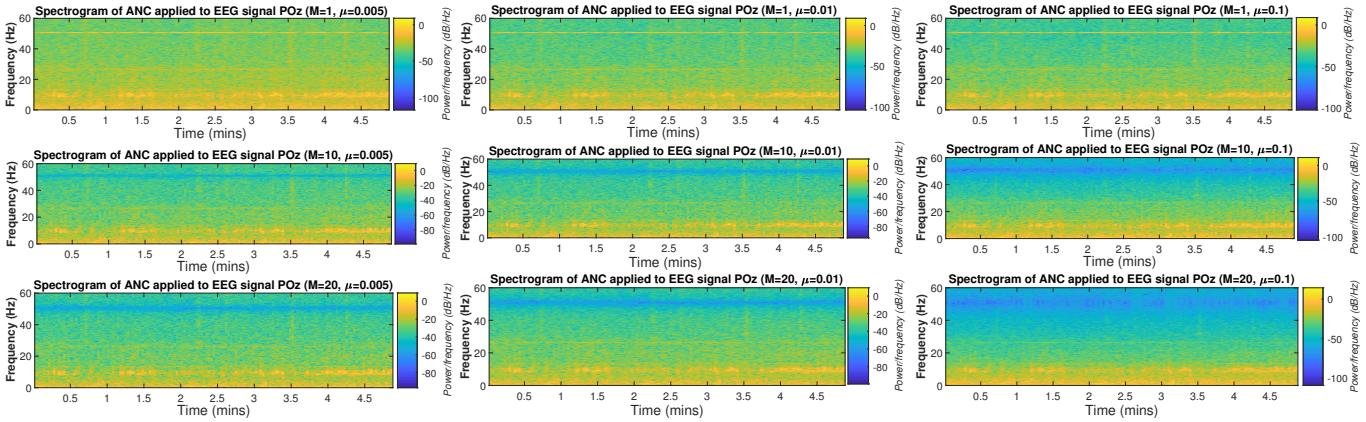

Figure 36: Spectrogram showing the time-frequency spectrum of the output of the ANC filter with the EEG signal `POz` as input, for different combinations of hyper-parameters μ, M. Different degrees of removal of the 50Hz noise are illustrated, with an optimal selective noise removal at $\mu = 0.01$ and $M = 10$.

The spectrogram of the original is shown in Figure 35, where the 50Hz noise component to be eliminated is clearly present for all the signal duration. The spectrogram of the output of the ANC filter are shown in Figure 36 for different values of the hyper-parameters μ (learning rate) and M (filter order).

We observe that while increasing the learning rate causes the ANC filter to suppress more efficiently (smaller power density) and within a shorter delay (in time domain) the 50Hz hum, it also causes it to eliminate more spectral components in the neighbourhood of the mains frequency $50Hz$, which is undesirable. Conversely, while setting a smaller $\mu = 0.005$ increases the delay before steady-state convergence, it also ensures that only the 50Hz component is selectively removed. With respect to the filter order M, we observe that under-modelling ($M = 1$) leads to poor noise cancelling performance, given that the 50Hz noise component is poorly attenuated, while over-modelling ($M = 25$) causes the filter to suppress spectral components in a wide region around the 50Hz frequency, as the in-built LMS filter attempts to model the wide-band white Gaussian noise component of $\eta(n)$. As a result, in order to selectively suppress the 50Hz mains without affecting the other frequency components of $s(n)$, we select $\mu = 0.01$ and $M = 10$.

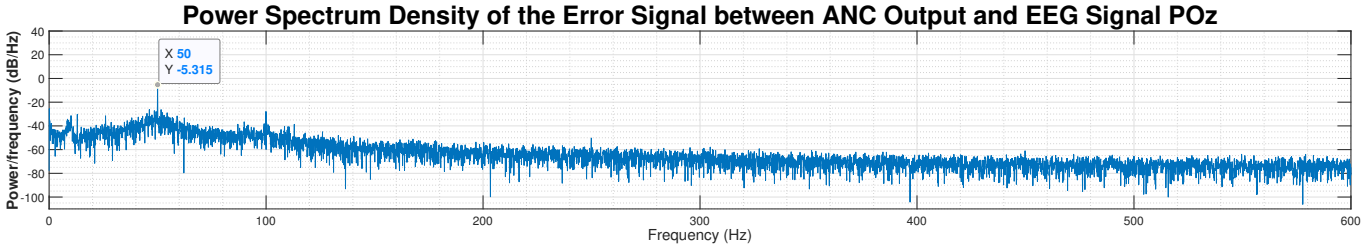

Figure 37: Power spectrum of the difference signal between the ANC output and the `POz` EEG signal, which primarily contains the 50Hz mains component, as expected. The hyper-parameters were set at the optimal values $\mu = 0.01$ and $M = 10$.

Figure 37 above depicts the power spectrum of the difference signal between the ANC output and the `POz` EEG signal for the optimal hyper-parameters $\mu = 0.01$ and $M = 10$, where it can be seen that the 50Hz noise component is clearly removed, while all other spectral components are virtually non-affected by the ANC filter, which validates our method.

3 Widely Linear Filtering and Adaptive Spectrum Estimation

3.1 Complex LMS and Widely Linear Modelling

a. In this section, our interest shifts to complex signals, which can be complex by design or by convenience of representation. In order to process complex data, we introduce the Complex-LMS (CLMS) algorithm, which assumes a strictly linear data model. As a result, one limitation of CLMS is that it only completely captures the second-order statistical behaviour of circular data. The Augmented CLMS (ACLMS) algorithm is therefore introduced: in contrast, it builds upon a widely linear data model, offering sufficient degrees of freedom to fully identify the second order statistics in non-circular data.

We have implemented both the CLMS and ACLMS algorithm in system identification configuration to identify the WLMA(1) model given by Equation 36 where $x \sim \mathcal{N}(0,1)$. Figure 38 demonstrates that while the complex white Gaussian noise driving the WLMA(1) process is by definition circular, the WLMA(1) data $y(n)$ generated is itself non-circular. We hence expect the CLMS configuration to fail to completely model $y(n)$, while ACLMS on the other hand should offer the sufficient degrees of freedom to learn the WLMA(1) parameters, which we denote by $\mathbf{b}$ (see Equation 36). This is confirmed by the learning curves obtained, averaged over 100 realisations of the process $y(n)$, as the steady-state error powers for CLMS and ACLMS are measured to be -13.5dB and -325.5dB respectively. The large steady-state error power reduction obtained under ACLMS when compared to CLMS demonstrates that widely linear data models are required to process non-circular data. Conceptually, the ACLMS filter has been learning $\mathbf{b}$ over 360 time steps, while CLMS has failed to do so.

$$y(n) = x(n) + b_1 x(n-1) + b_2 x^*(n-1) \qquad \texttt{where} \qquad \mathbf{b} = [b_1, b_2] = [1.5+1i, 2.5-0.5i] \tag{36}$$

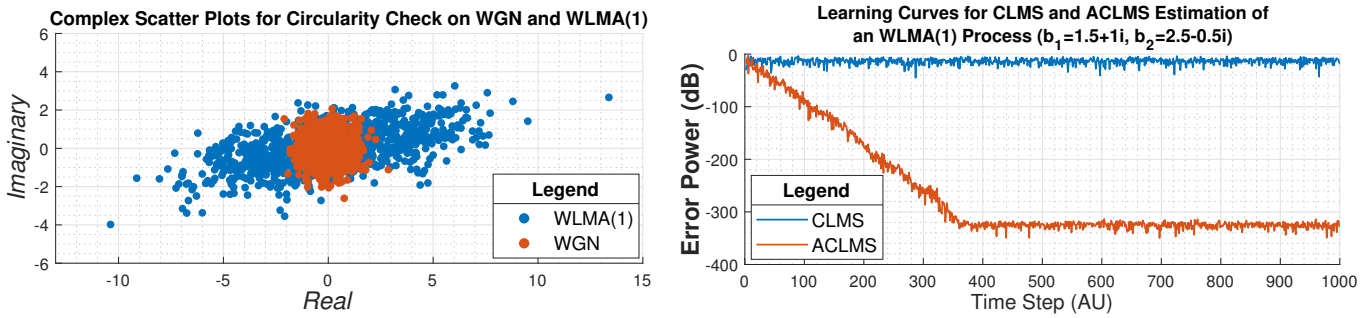

Figure 38: Left: Circularity plot for the WGN data $x(n)$ (filter input) and WLMA(1) data $y(n)$ (filter output), demonstrating non-circularity of $y(n)$. **Right:** Learning Curves for CLMS and ACLMS obtained from 100 realisations of the process $y(n)$ defined in Equation 36, demonstrating effective learning only in the case of ACLMS, owing to data non-circularity.

b. Complex-valued signals were created from 2D wind speed data in the North-South and East-West directions for low, medium and high dynamic regimes. The circularity plots, along with the circularity coefficients ρ, are shown in Figure 39 for each wind regime. Data circularity indicates rotational invariance of the probability distribution of the data in the complex plane, and the circularity coefficient ρ indicates the degree of non-circularity of the data, therefore Figure 39 demonstrate that despite the circular appearance of the circularity plots for higher regimes, ρ indicates that lower regimes result in improved data circularity. We therefore expect ACLMS to outperform CLMS by a larger extent for high wind regimes.

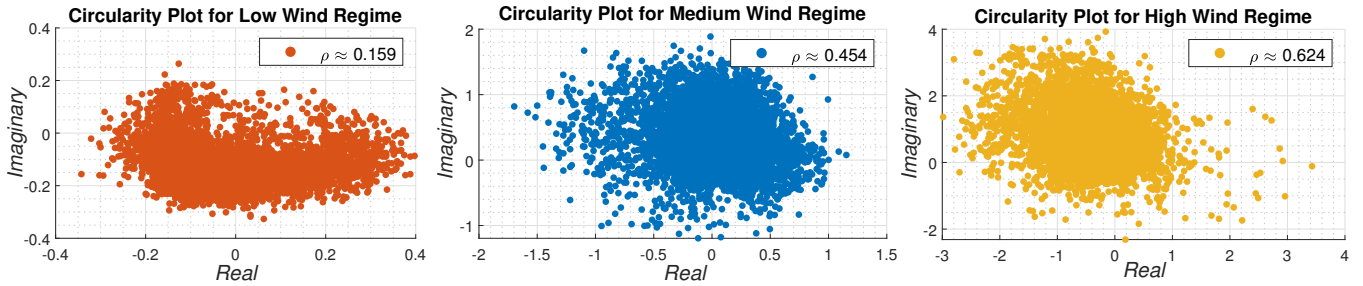

Figure 39: Left: Circularity Plot for Low Wind Regime, showing a circularity coefficient $\rho \approx 0.159$. **Middle:** Circularity Plot for Medium Wind Regime, showing a circularity coefficient $\rho \approx 0.454$. **Right:** Circularity Plot for High Wind Regime, showing a circularity coefficient $\rho \approx 0.624$. The metric ρ indicates that higher wind regimes exhibit stronger non-circularity.

The CLMS and ACLMS filters were implemented in a one-step-ahead prediction setting on the wind data. The effect of filter length on the Mean Squared Prediction Error (MSPE) power was studied in Figure 40 for different wind regimes, where MSPE was taken as the filter performance metric. Our proposed definition of filter performance attempts to yield a standard error metric enabling significant comparison across algorithms.

Regime	**Low**		**Medium**		**High**	
Algorithm	*ACLMS*	*CLMS*	*ACLMS*	*CLMS*	*ACLMS*	*CLMS*
Optimal M	7	7	4	5	4	5

Table 6: Summary of optimal model order M yielding minimal MSPE for different wind regimes, using the data shown in Figure 40.

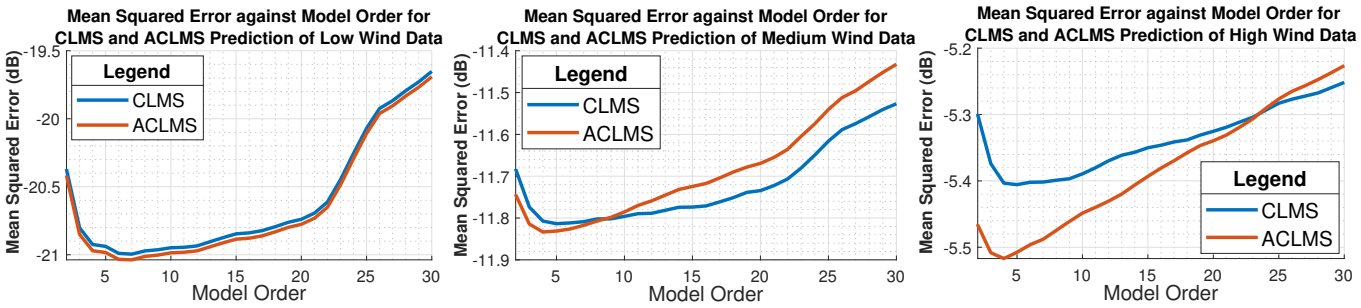

Figure 40: Left: CLMS and ACLMS MSPE against filter length for Low Wind Regime. **Middle:** CLMS and ACLMS MSPE against filter length for Medium Wind Regime. **Right:** CLMS and ACLMS MSPE against filter length for High Wind Regime.

The results are summarised in Table 6 and demonstrate that for the more non-circular data associated with medium and high wind regimes, ACLMS outperforms CLMS at low model orders. At higher model orders ($M \geq 8$ for medium regime and $M \geq 23$ for high regime), the additional degrees of freedom offered by ACLMS result in increased over-modelling and over-fitting as the filter coefficients attempt to model the data noise, ultimately compromising the filter performance. Since CLMS exhibits less degrees of freedom, it suffers less from possible over-fitting, and as a result outperforms ACLMS for high model orders. In contrast, in the case of the more circular data obtained under low wind regime, ACLMS consistently outperforms CLMS.

The results also reveal that minimum MSPE can be achieved for the optimal filter lengths in the range $[3 \rightarrow 7]$. At the optimal filter length, the degree to which ACLMS outperforms CLMS is proportional to the extent of data non-circularity. Indeed, the MSPE reduction offered by ACLMS compared to CLMS at the optimal filter length is approximately 0.04dB and 0.1dB for low and high regimes respectively, where the circularity coefficients are $\rho \approx 0.16$ and $\rho \approx 0.62$ respectively, and we recall that a high ρ indicates strong data non-circularity.

c. We now consider a three phase power system. Four sets of three voltages (each set representing a three-phase power system) were synthesised for balanced and unbalanced networks. Specifically, let $\mathbf{V}$ be the vector of the peak magnitude of each component of the three-phase system, and let the term $\mathbf{\Delta}$ be the vector of phase distortions relative to the balanced three-phase system, we have defined balanced and unbalanced vectors $\mathbf{V_b}$, $\mathbf{V_u}$ and $\mathbf{\Delta_b}$, $\mathbf{\Delta_u}$, given in Equation 37. The voltage signals were generated using a sampling frequency $f_s = 5$kHz. Each three-voltage set, constructed from a unique combination of $\mathbf{V_b}$, $\mathbf{V_u}$ and $\mathbf{\Delta_b}$, $\mathbf{\Delta_u}$, was represented, via the Clarke transform, through the compact $\alpha - \beta$ complex Clarke voltage representation $v(n) = v_\alpha(n) + jv_\beta(n)$. The circularity plot and the circularity coefficient ρ for each complex voltage synthesised are shown in Figure 41: as expected, the purely balanced system exhibits zero degree of non-circularity ($\rho = 0$), while phase or magnitude distortion results in unbalanced non-circular systems with $\rho > 0$. We also note that imbalance resulting from phase distortion results in elliptic circularity plot where the major and minor axis of the ellipse coincide with the real and imaginary axes. In contrast, an elliptic circularity plot where the major and minor axes do not coincide with the real and imaginary axes would lead us to suggest the existence of a magnitude mismatch failure in the system, possibly combined with phase distortion.

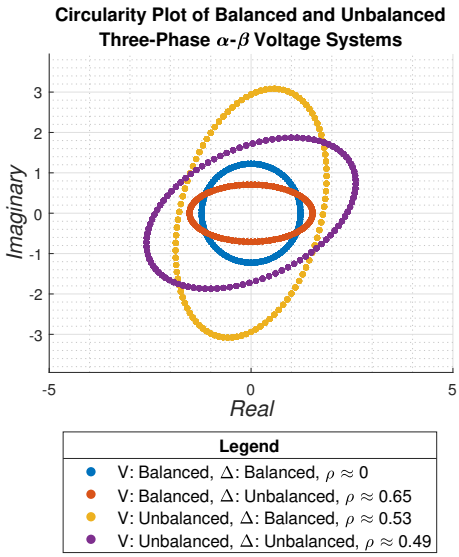

Figure 41: Circularity plots for balanced and unbalanced complex Clarke voltages

$$\begin{aligned} \mathbf{V_b} &= \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T \quad \mathbf{V_u} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T \\ \mathbf{\Delta_b} &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \quad \mathbf{\Delta_u} = \begin{bmatrix} 0 & \frac{\pi}{2} & -\frac{\pi}{2} \end{bmatrix}^T \end{aligned} \tag{37}$$

d. Under balanced conditions, the complex Clarke voltage is given by the left-hand side of Equation 38, which can be substituted in Equation 39 expressing the strictly linear auto-regressive model of order 1. Conversely, under unbalanced conditions, the complex Clarke voltage is given by the right-hand side of Equation 38, which can be substituted in Equation 41 expressing the widely linear auto-regressive model of order 1.

$$v(n) = \sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_0}{f_s} n + \phi)} \quad \texttt{(balanced)} \qquad v(n) = A(n) e^{j(2\pi \frac{f_0}{f_s} n + \phi)} + B(n) e^{-j(2\pi \frac{f_0}{f_s} n + \phi)} \quad \texttt{(unbalanced)} \tag{38}$$

$$v(n+1) = h^* v(n) \quad \texttt{where} \quad v(n+1) = \sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_0}{f_s}(n+1) + \phi)} = \sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_0}{f_s} n + \phi)} \times e^{j(2\pi \frac{f_0}{f_s})} = e^{j(2\pi \frac{f_0}{f_s})} v(n) \tag{39}$$

By identification from Equation 39, we obtain $h^*(n) = e^{j(2\pi \frac{f_0}{f_s})}$, which can be deconstructed into a phase equality and a magnitude equality, as shown in Equation 40, where $\Re\{h(n)\}$ and $\Im\{h(n)\}$ denote the real and imaginary parts of $h(n)$ respectively.

$$
\begin{aligned}
&h^*(n)=e^{j(2\pi\frac{f_0}{f_s})}\ \texttt{therefore}\ \ h(n)=e^{-j(2\pi\frac{f_0}{f_s})}\ \texttt{then}\\
&\big|h(n)\big|=\big|e^{-j(2\pi\frac{f_0}{f_s})}\big|=1\ \texttt{and}\ \angle\big\{h(n)\big\}=\angle\big\{e^{-j(2\pi\frac{f_0}{f_s})}\big\}\ \texttt{therefore}\\
&\arctan\left(\frac{\Im\big\{h(n)\big\}}{\Re\big\{h(n)\big\}}\right)=-2\pi\frac{f_0}{f_s}\ \ \texttt{therefore}\ \ f_0(n)=-\frac{f_s}{2\pi}\arctan\left(\frac{\Im\big\{h(n)\big\}}{\Re\big\{h(n)\big\}}\right)\ \ \texttt{QED}
\end{aligned}
\tag{40}
$$

In an unbalanced system, we can similarly derive the voltage frequency $f_0(n)$ as a function of the parameters $h(n)$ and $g(n)$. We again start by computing $v(n+1)$ from the right-hand side of Equation 38, as shown in Equation 41 below.

$$
\begin{aligned}
&v(n+1)=h^*(n)v(n)+g^*(n)v^*(n)\ \ \texttt{where}\ \ v(n+1)=A(n+1)e^{j(2\pi\frac{f_0}{f_s}(n+1)+\phi)}+B(n+1)e^{-j(2\pi\frac{f_0}{f_s}(n+1)+\phi)}\ \ \texttt{with}\\
&A(n)=\frac{\sqrt{6}}{6}\Big(V_a(n)+V_b(n)e^{j\Delta_b}+V_c(n)e^{j\Delta_c}\Big)\ \ \texttt{and}\ \ B(n)=\frac{\sqrt{6}}{6}\Big(V_a(n)+V_b(n)e^{-j(\Delta_b+2\pi/3)}+V_c(n)e^{-j(\Delta_c-2\pi/3)}\Big)
\end{aligned}
\tag{41}
$$

Given that $V_a(n)$, $V_b(n)$ and $V_c(n)$ are defined as the peak value of each voltage sinusoid, we can assume that their change over one time step are negligible, yielding $V_a(n+1)\approx V_a(n)$, $V_b(n+1)\approx V_b(n)$ and $V_c(n+1)\approx V_c(n)$.

$$
\begin{aligned}
&v(n+1)=A(n)e^{j(2\pi\frac{f_0}{f_s}n+\phi)}\times e^{j(2\pi\frac{f_0}{f_s})}+B(n)e^{-j(2\pi\frac{f_0}{f_s}n+\phi)}\times e^{-j(2\pi\frac{f_0}{f_s})}\\
&v(n+1)=h^*(n)\Big(A(n)e^{j(2\pi\frac{f_0}{f_s}n+\phi)}+B(n)e^{-j(2\pi\frac{f_0}{f_s}n+\phi)}\Big)+g^*(n)\Big(A^*(n)e^{-j(2\pi\frac{f_0}{f_s}n+\phi)}+B^*(n)e^{j(2\pi\frac{f_0}{f_s}n+\phi)}\Big)
\end{aligned}
\tag{42}
$$

Now, collecting the $e^{j(2\pi\frac{f_0}{f_s}n+\phi)}$ and $e^{-j(2\pi\frac{f_0}{f_s}n+\phi)}$ terms together, we obtain the following expression, where by identification we can obtain an equation quadratic in $\frac{B^*(n)}{A(n)}$ by noting that $e^{j(2\pi\frac{f_0}{f_s})}$ and $e^{-j(2\pi\frac{f_0}{f_s})}$ are complex conjugates.

$$
\begin{aligned}
&v(n+1)=\Big(h^*(n)A(n)+g^*(n)B^*(n)\Big)e^{j(2\pi\frac{f_0}{f_s}n+\phi)}+\Big(h^*(n)B(n)+g^*(n)A^*(n)\Big)e^{-j(2\pi\frac{f_0}{f_s}n+\phi)}\ \ \texttt{thus}\\
&A(n)e^{j(2\pi\frac{f_0}{f_s})}=h^*(n)A(n)+g^*(n)B^*(n)\ \ \texttt{and}\ \ B(n)e^{-j(2\pi\frac{f_0}{f_s})}=h^*(n)B(n)+g^*(n)A^*(n)\ \ \texttt{then}\\
&e^{j(2\pi\frac{f_0}{f_s})}=h^*(n)+g^*(n)\left[\frac{B^*(n)}{A(n)}\right]\ \ \texttt{and}\ \ e^{-j(2\pi\frac{f_0}{f_s})}=h^*(n)+g^*(n)\left[\frac{A^*(n)}{B(n)}\right]\ \ \texttt{are complex conjugates hence}\\
&h^*(n)+g^*(n)\left[\frac{B^*(n)}{A(n)}\right]=\left[h^*(n)+g^*(n)\left[\frac{A^*(n)}{B(n)}\right]\right]^*=h(n)+g(n)\left[\frac{A(n)}{B^*(n)}\right]\ \ \texttt{multiplying both sides by}\ \ \frac{B^*(n)}{A(n)}\\
&g^*(n)\left[\frac{B^*(n)}{A(n)}\right]^2-2j\Im\Big\{h(n)\Big\}\left[\frac{B^*(n)}{A(n)}\right]-g(n)=0\ \ \texttt{where we have used}\ \ h^*(n)-h(n)=-2j\Im\Big\{h(n)\Big\}
\end{aligned}
\tag{43}
$$

The above quadratic equation is solved for $\frac{B^*(n)}{A(n)}$ in Equation 44, where we have used $g^*(n)g(n)=|g(n)|^2$.

$$
\frac{B^*(n)}{A(n)}=\frac{2j\Im\Big\{h(n)\Big\}\pm\sqrt{\Big(-2j\Im\Big\{h(n)\Big\}\Big)^2+4g^*(n)g(n)}}{2g^*(n)}=\frac{j\Im\Big\{h(n)\Big\}\pm j\sqrt{\Im^2\Big\{h(n)\Big\}-|g(n)|^2}}{g^*(n)}
\tag{44}
$$

The above result can be substituted in $e^{j(2\pi\frac{f_0}{f_s})}=h^*(n)+g^*(n)\frac{B^*(n)}{A(n)}$ where we ultimately solve for the frequency term f_0.

$$
\begin{aligned}
&e^{j(2\pi\frac{f_0}{f_s})}=h^*(n)+g^*(n)\left[\frac{j\Im\Big\{h(n)\Big\}\pm j\sqrt{\Im^2\Big\{h(n)\Big\}-|g(n)|^2}}{g^*(n)}\right]=h^*(n)+j\Im\Big\{h(n)\Big\}\pm j\sqrt{\Im^2\Big\{h(n)\Big\}-|g(n)|^2}\\
&e^{j(2\pi\frac{f_0}{f_s})}=\Re\Big\{h(n)\Big\}\pm j\sqrt{\Im^2\Big\{h(n)\Big\}-|g(n)|^2}\ \ \rightarrow\ \ \angle\Big\{e^{j(2\pi\frac{f_0}{f_s})}\Big\}=\angle\Big\{\Re\Big\{h(n)\Big\}\pm j\sqrt{\Im^2\Big\{h(n)\Big\}-|g(n)|^2}\Big\}\\
&2\pi\frac{f_0}{f_s}=\arctan\left\{\frac{\pm\sqrt{\Im^2\Big\{h(n)\Big\}-|g(n)|^2}}{\Re\Big\{h(n)\Big\}}\right\}\ \ \texttt{where}\ \ f_0>0,\ f_s>0\ \ \texttt{therefore}\\
&2\pi\frac{f_0}{f_s}=\arctan\left\{\frac{\sqrt{\Im^2\Big\{h(n)\Big\}-|g(n)|^2}}{\Re\Big\{h(n)\Big\}}\right\}\ \ \rightarrow\ \ f_0(n)=\frac{f_s}{2\pi}\arctan\left\{\frac{\sqrt{\Im^2\Big\{h(n)\Big\}-|g(n)|^2}}{\Re\Big\{h(n)\Big\}}\right\}\ \ \texttt{QED}
\end{aligned}
\tag{45}
$$

In the above, we have used the fact that $\arctan(x)>0\forall x>0$, to obtain an expressing for the non-stationary frequency $f_0(n)$ as a function of the ACLMS filter weights. As a result, we have constructed an adaptive filter for spectral estimation of non-stationary data. We will demonstrate the advantages of this online implementations over classical subspace techniques in the next section.

e. The first-order auto-regressive AR(1) CLMS and ACLMS filters were trained in system identification configuration on balanced and unbalanced complex Clarke voltages. Specifically, the unbalanced voltage was constructed from combined phase

distortion and magnitude imbalance using the vectors $\mathbf{V_u}$ and $\mathbf{\Delta_u}$ previously defined in Equation 37, resulting in the data shown in purple in Figure 41. The resulting learning curves are shown in Figures 42 and 43, averaged over 100 realisations, and measures of interest obtained from these curves are summarised in Table 7 shown below.

In the case of circular data resulting (i.e. balanced network), CLMS outperforms ACLMS in rate of convergence and both filters exhibit similar steady-state error power (see Table 7). This is coherent with our previous observation that the steady-state performances of CLMS and ACLMS are equivalent when implemented on circular data. We note that the additional degrees of freedom offered by ACLMS cause the latter to converge more slowly than CLMS, due to the additional computational burden.

In contrast, in the case of non-circular data (i.e. unbalanced network), CLMS fails to capture the second-order statistics of the data and the filter saturates at an error power of -23.8dB. In contrast, ACLMS makes use of the data pseudo-statistics and successfully models the non-circular data as it attains a largely inferior steady-state error power of -280.3dB over the scale of 1000 time steps, as summarised in Table 7.

We now wish to demonstrate that CLMS and ACLMS filters can be used to track the main frequency component of complex data. To that end, from Equations 40 and 44, estimates for the system nominal frequency f_0 were constructed, using the CLMS and ACLMS filters parameters $h(n)$ and $g(n)$, as shown in Figures 42 and 43. Again, for circular data, both filters converge to the true value $f_0 = 50$Hz, and CLMS outperforms ACLMS given that the former immediately converges to f_0 while the latter displays (over the first 250 time steps) a transient behaviour oscillating around f_0. In the case of non-circular data, ACLMS maintains a transient behaviour over the same time scale as with circular data, and similarly converges to the exact value f_0. In contrast, the CLMS nominal frequency estimate $\hat{f}_0$ fails to converge and oscillates around the erroneous frequency of 37Hz. This again demonstrates that ACLMS outperforms CLMS when applied to non-circular data as CLMS lacks the degrees of freedom to model non-circular complex data distributions, where the circularity coefficient is $\rho \approx 0.5$. ACLMS, on the other hand, makes use of additional degrees of freedom to completely capture the second-order data statistics (i.e. pseudo-covariance).

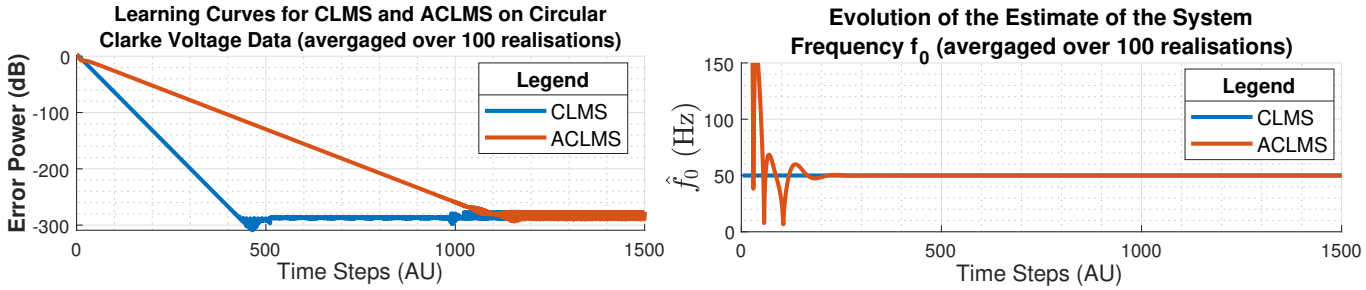

Figure 42: Left: Learning curves for CLMS and ACLMS on circular complex Clarke voltage data. **Right:** Evolution of the estimate $\hat{f}_0$ under CLMS and ACLMS for the circular complex Clarke voltage data. The learning rate was set to $\mu = 0.05$.

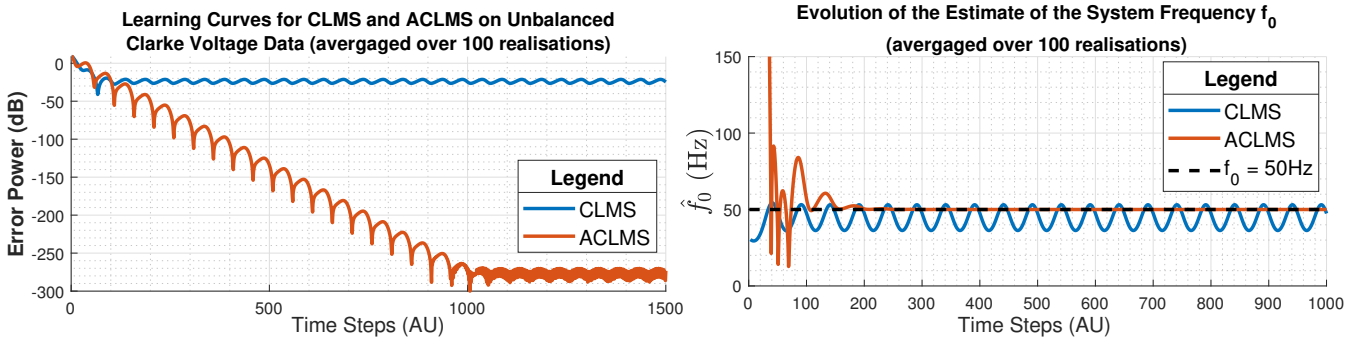

Figure 43: Left: Learning curves for CLMS and ACLMS on non-circular complex Clarke voltage data. **Right:** Evolution of the estimate $\hat{f}_0$ under CLMS and ACLMS for the non-circular complex Clarke voltage data. The learning rate was set to $\mu = 0.01$.

Data Circularity	Algorithm	Convergence (# steps)	Steady-State Error Power (dB)
Non-Circular	CLMS	500	-286.3
	ACLMS	1200	-285.1
Circular	CLMS	NA	-23.8
	ACLMS	1000	-280.3

Table 7: Rate of convergence and steady-state error power obtained from the learning curves corresponding to CLMS and ACLMS algorithms applied to circular and non-circular complex voltage data, as shown in Figures 42 and 43.

3.2 Adaptive AR Model Based Time-Frequency Estimation

a. This section explores non-stationary spectrum estimation where online frequency tracking is required, as would be the case in Big Data applications. The non-stationary frequency-modulated (FM) signal $y(n)$ was generated to exemplifies such a case where signal information is carried by the time-varying frequency $f(n)$, making online frequency estimation necessary. The time-variation of the frequency $f(n)$ is shown in Figure 44 to illustrate data non-stationarity.

Initially, the Yule-Walker method is applied to the signal $y(n)$: it is an AR model-based block filter which assumes data stationarity and therefore fails to capture the time-varying behaviour of $f(n)$, as shown in Figure 44 where the estimated power spectrum obtained from the application of the Yule-Walker method to the FM signal $y(n)$ does not reflect the frequency variations of $y(n)$. Indeed, the Yule-Walker method attempts to fit the data to a filter model represented via a rational function with constant coefficients, and hence cannot model data with time-varying statistical characteristics.

As shown in Figure 44, increasing the assumed AR model order p within the Yule-Walker framework only enables the identification of the constant 100Hz frequency component characterising $y(n)$ from $n=0\rightarrow500$. Furthermore, when the Yule-Walker method is applied to each segment of the piece-wise function $f(n)$ (constant, linear, and quadratic), the resulting spectrum is only valid when the data is stationary, that is, when the frequency $f(n)$ equals a constant. In the case of linearly/quadratically varying frequency, it appears that the model converges approximately to the mean frequency over the associated time range (see Table 8).

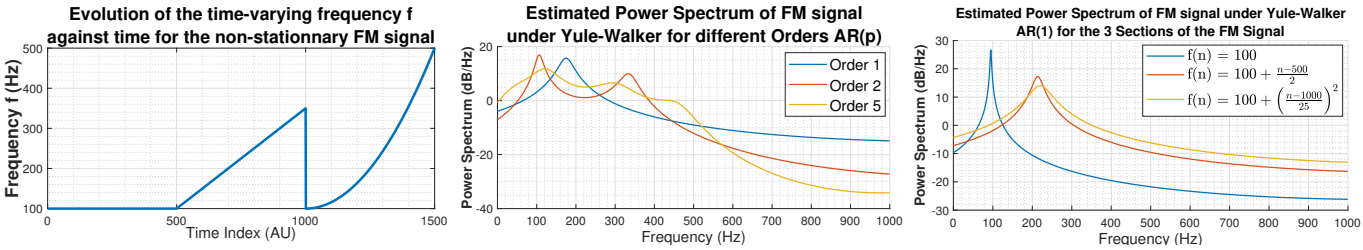

Figure 44: Left: Time-varying frequency $f(n)$ characterising the FM signal $y(n)$. **Middle:** Estimated power spectrum obtained from fitting the FM signal $y(n)$ to an AR(p) model using Yule-Walker method, for different model orders p. **Right:** Estimated power spectrum for each section of the piecewise-defined FM signal $y(n)$ obtained from fitting the data to an AR(1) model using Yule-Walker method.

Segment (step range)	[1:500]	[501:1000]	[1001:1500]
Mean frequency (Hz)	100	225.25	233.73
Spectral peak (Hz)	100	216.7	222.7

Table 8: Summary of measured frequency associated with spectral peaks in Figure 44 illustrating the estimated power spectrum for each section of the piecewise-defined FM signal $y(n)$ obtained from fitting the data to an AR(1) model using Yule-Walker method, along with the time-averaged frequency $\overline{f}(n)$ associated with the corresponding time-step range.

b. In order to overcome the limitations of the Yule-Walker block filter, a dynamic filtering approach was preferred. The CLMS algorithm was implemented to estimate the AR coefficient of the signal $y(n)$ and track the time-varying frequency $f(n)$ under the assumption of a first-order model AR(1). At each time instant n, the AR(1) coefficient is estimated from the dynamic weight of the adaptive CLMS filter and the corresponding power spectrum associated with that AR(1) parameter is constructed. This method yields a time-frequency description of the FM signal $y(n)$ shown in Figure 46, where the evolution of the frequency $f(n)$ can be characterised by tracking the frequency of the main spectral peak computed at each instant n, as shown in Figure 47. As a consequence, this dynamic CLMS-AR(1) filter overcomes the limitations proposed by the static Yule-Walker method and successfully models the non-stationary data $y(n)$. Let us note that by the nature of this technique, there is no constraint on the assumed model order p, and we could equivalently define a more general CLMS-AR(p) framework enabling online spectral estimation for any order p. Besides, let us also note that the use of CLMS instead of ACLMS is here justified since the FM data $y(n)$ is circular, as shown in Figure 45.

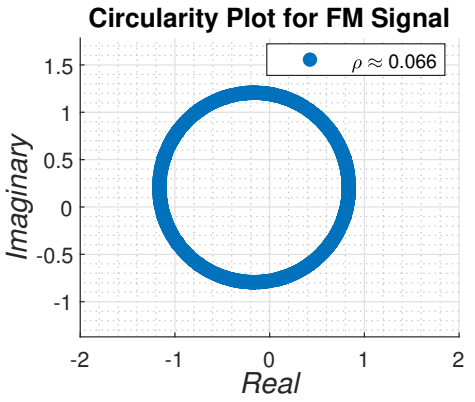

Figure 45: Circularity plot for FM signal $y(n)$.

The effect of varying the learning rate μ upon the frequency tracking performance of CLMS was assessed. At small learning rates ($\mu=0.001$), the filter weights adapt too slowly to track the fast variations in frequency $f(n)$, causing the frequency estimate to systematically exhibit a lower slope than the true frequency value $f(n)$, as seen in the linear segment for $n:500\rightarrow1000$ in Figure 47. While the higher learning rate ($\mu=0.1$) allow CLMS to successfully track variations in $f(n)$, the resulting estimate exhibits a higher variance as stability was traded for improved dynamic behaviour and reduced estimation bias. This is yet another example of bias-variance trade-off depending on the hyper-parameter μ.

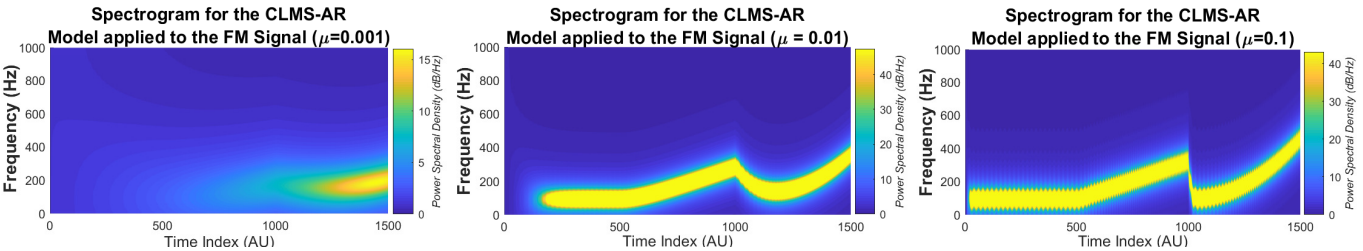

Figure 46: Spectrogram of the CLMS-AR(1) model applied to the non-stationary FM data $y(n)$ for different learning rates μ.

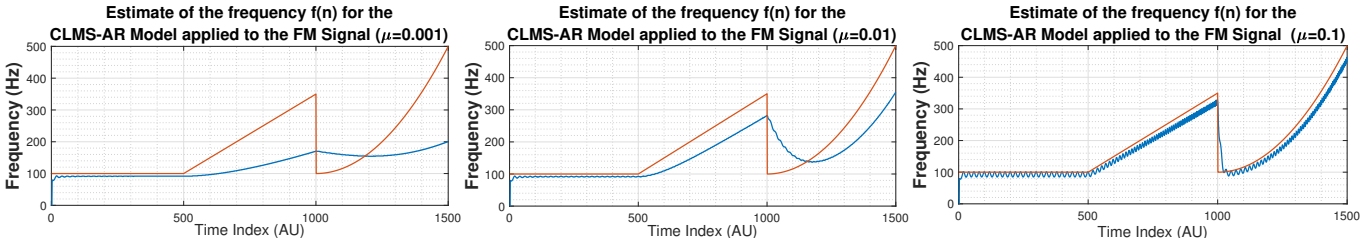

Figure 47: Frequency estimate obtained from the CLMS-AR model applied to the non-stationary FM data $y(n)$ for different learning rates μ. The trade-off between dynamic range and model stability is illustrated.

3.3 A Real-Time Spectrum Analyser Using Least Mean Square

a. The least square solution for estimating an arbitrary signal $y(n)$ using a linear combination of N harmonically related sinusoids is now derived. The cost function $J(n)$ to be minimised is expressed in Equation 46, where $\mathbf{y}$ is the true signal and $\hat{\mathbf{y}}$ is the estimate of $\mathbf{y}$ given by $\hat{\mathbf{y}}=\mathbf{F}\mathbf{w}$, where $\mathbf{F}$ is the matrix of basis complex sinusoids. We have used the fact that by definition $J(n)$ is a scalar, therefore when expressing $J(n)$ as a sum of sub-terms, each sub-term is also a scalar and necessarily equal to its own transpose, as shown in Equation 46. The minimisation of $J(n)$, defined from an optimisation perspective as an objective function where the design variable is the column vector $\mathbf{w}$ containing the filter weights, can be achieved by setting the first derivative of $J(n)$ with respect to $\mathbf{w}$ to zero to obtain the optimal weight vector $\mathbf{w_o}$, as shown in Equation 47.

$$\begin{aligned} &J=||\mathbf{y}-\hat{\mathbf{y}}||^2=||\mathbf{y}-\mathbf{F}\mathbf{w}||^2=\Big[\mathbf{y}-\mathbf{F}\mathbf{w}\Big]^H\Big[\mathbf{y}-\mathbf{F}\mathbf{w}\Big]=\Big[\mathbf{y}^H-\mathbf{w}^H\mathbf{F}^H\Big]\Big[\mathbf{y}-\mathbf{F}\mathbf{w}\Big]=\mathbf{y}^H\mathbf{y}-\mathbf{y}^H\mathbf{F}\mathbf{w}-\mathbf{w}^H\mathbf{F}^H\mathbf{y}+\mathbf{w}^H\mathbf{F}^H\mathbf{F}\mathbf{w} \\ &J=\mathbf{y}\mathbf{y}^H-2\mathbf{w}^H\mathbf{F}^H\mathbf{y}+\mathbf{w}^H\mathbf{F}^H\mathbf{F}\mathbf{w} \end{aligned} \tag{46}$$

The dot product rule from $\mathbb{CR}$ calculus gives $\nabla_{\mathbf{w}}\big(\mathbf{w}^H\mathbf{F}^H\mathbf{F}\mathbf{w}\big)=\nabla_{\mathbf{w}}\big(\mathbf{w}^H\big)\mathbf{F}^H\mathbf{F}\mathbf{w}+\mathbf{w}^H\nabla_{\mathbf{w}}\big(\mathbf{F}^H\mathbf{F}\mathbf{w}\big)=2\mathbf{F}^H\mathbf{F}\mathbf{w}$, and we obtain Equation 47 yielding the optimal weight vector $\mathbf{w_o}$, where for the solution to exist, we require $\mathbf{F}^H\mathbf{F}$ to be invertible. Intuitively, since $\mathbf{F}$ characterises a set of orthogonal complex sinusoids basis, hence $\mathbf{F}$ is full rank and since $\texttt{rank}(\mathbf{F}^H\mathbf{F})=\texttt{rank}(\mathbf{F})$ (from $\mathbb{CR}$ calculus), then $\mathbf{F}^H\mathbf{F}$ is also full rank and therefore it is invertible, where both the existence and uniqueness of the least square solution $\mathbf{w_o}$ was demonstrated.

$$\begin{aligned} &\nabla_{\mathbf{w}}J\big|_{\mathbf{w}=\mathbf{w_o}}=0 \;\rightarrow\; \nabla_{\mathbf{w}}\big(\mathbf{y}\mathbf{y}^H\big)-2\nabla_{\mathbf{w}}\big(\mathbf{w}^H\mathbf{F}^H\mathbf{y}\big)+\nabla\mathbf{w}\big(\mathbf{w}^H\mathbf{F}^H\mathbf{F}\mathbf{w}\big)=0-2\mathbf{F}^H\mathbf{y}+2\mathbf{F}^H\mathbf{F}\mathbf{w_o}=0 \\ &\mathbf{w_o}=(\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H\mathbf{y} \;\; \texttt{QED} \end{aligned} \tag{47}$$

We now wish to compare the optimal weight expression, corresponding to the least-square solution to the problem of approximating a signal $y(n)$ as a linear combination of complex basis sinusoids, against the Inverse Discrete Fourier Transform (IDFT) formula, defined in Equation 48 where $X(k)$ is the Discrete Fourier Transform (DFT) coefficients of the signal $x(n)$ obtained from projecting $x(n)$ on a set of N orthogonal complex harmonically related basis sinusoids.

$$x(n)=\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}X(k)e^{2\pi k\frac{n}{N}} \tag{48}$$

Let us define $\omega(n)=e^{i\frac{2\pi n}{N}}$, and let the column vector $\mathbf{x}$ contain the elements $x(n)$ with index n, while the column vectors $\mathbf{X}$ and $\mathbf{W_n}$ contain the elements $X(k)$ and $\omega(n)^k$ with index k. We can then express Equation 48 in matrix form as shown in Equation 49, where we have $\mathbf{F}=\frac{1}{\sqrt{N}}\begin{bmatrix}\mathbf{W_0} & \mathbf{W_1} & \dots & \mathbf{W_{N-1}}\end{bmatrix}^T$, as defined in Equation 50.

$$\begin{aligned} &x(n)=\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}X(k)\omega(n)^k=\frac{1}{\sqrt{N}}\begin{bmatrix}1 & \omega(n)^1 & \dots & \omega(n)^{N-1}\end{bmatrix}^T\cdot\begin{bmatrix}X(0) & X(1) & \dots & X(N-1)\end{bmatrix}=\frac{1}{\sqrt{N}}\mathbf{W_n}^T\mathbf{X} \\ &\mathbf{x}=\begin{bmatrix}x(0) & x(1) & \dots & x(N-1)\end{bmatrix}^T=\frac{1}{\sqrt{N}}\begin{bmatrix}\mathbf{W_0}^T\mathbf{X} & \mathbf{W_1}^T\mathbf{X} & \dots & \mathbf{W_{N-1}}^T\mathbf{X}\end{bmatrix}^T=\mathbf{F}\mathbf{X} \end{aligned} \tag{49}$$

The IDFT matrix equation formulated in Equation 49 is analogous to the equation formulated for the problem of estimating a signal $y(n)$ using a linear combination of N harmonically related sinusoids, given by $\hat{\mathbf{y}}=\mathbf{F}\mathbf{w}$. We indeed identify $\mathbf{x}\equiv\hat{\mathbf{y}}$ and $\mathbf{X}\equiv\mathbf{w}$. This demonstrates that a real-time spectrum analyser can be implemented from a least mean square algorithm where the adaptive LMS filter weights are the coefficients of the DFT of the input signal $y(n)$.

Now, considering the Discrete Fourier Transform (DFT), it can be shown that the DFT transformation consists of projecting a given signal $x(n)$ on a set of basis complex exponentials with base element $w^*(n)$, and which can be represented in matrix form as the conjugate transpose of the matrix $\mathbf{F}$, such that $\mathbf{X}=\mathbf{F}^H\mathbf{x}$. Substituting $\mathbf{X}$ in Equation 49, we obtain $\mathbf{x}=\mathbf{F}\mathbf{F}^H\mathbf{x}$ which implies that $\mathbf{F}$ is unitary as $\mathbf{F}\mathbf{F}^H=I$ (**this is not a formal proof**, however it can formally be shown that $\mathbf{F}$ is indeed unitary, see Problem Sets associated with the course). Finally, using Equation 47, we note that since $\mathbf{w_o}=(\mathbf{F}^H\mathbf{F})^{-1}\mathbf{y}^H\mathbf{F}=\mathbf{F}^H\mathbf{y}=\mathbf{Y}$, where $\mathbf{Y}$ contains the coefficients of the DFT transform of the arbitrary signal $\mathbf{y}$, we have demonstrated that the IDFT yields the least-square solution to the approximation of the signal $\mathbf{y}$ using a set of complex harmonically related sinusoids, minimising the error term J.

$$\mathbf{F}=\frac{1}{\sqrt{N}}\begin{bmatrix}\mathbf{W_0}\\ \mathbf{W_1}\\ \vdots\\ \mathbf{W_{N-1}}\end{bmatrix}=\frac{1}{\sqrt{N}}\begin{bmatrix}\omega(0)^0 & \omega(0)^1 & ... & \omega(0)^{N-1}\\ \omega(1)^0 & \omega(1)^1 & ... & \omega(1)^{N-1}\\ \vdots & \vdots & \ddots & \vdots\\ \omega(N-1)^0 & \omega(N-1)^1 & ... & \omega(N-1)^{N-1}\end{bmatrix}=\frac{1}{\sqrt{N}}\begin{bmatrix}1 & 1 & ... & 1\\ 1 & e^{i\frac{2\pi}{N}} & ... & e^{i\frac{2\pi(N-1)}{N}}\\ \vdots & \vdots & \ddots & \vdots\\ 1 & e^{i\frac{2\pi(N-1)}{N}} & ... & e^{i\frac{2\pi(N-1)^2}{N}}\end{bmatrix} \tag{50}$$

b. We have previously demonstrated that the IDFT yields the least-square solution to the approximation of a signal $\mathbf{y}$ using a set of N complex harmonically related sinusoids. Equivalently, from Equation 49 we have shown that $\mathbf{X}=\mathbf{w_0}=\mathbf{F}^H\mathbf{x}$. In other terms, the DFT coefficients are a linear combination of the rows of the transformation matrix $\mathbf{F}^H$, or equivalently of its columns, since $\mathbf{F}$ is a Toeplitz matrix. Using Equation 50, the rows of $\mathbf{F}^H$, denoted by $\mathbf{r_n}=\bar{\mathbf{W}}_\mathbf{n}$ (where $\bar{\mathbf{W}}_\mathbf{n}$ denotes the complex conjugate of the vector $\mathbf{W_n}$), are orthonormal vectors, as seen from Equation 51. Therefore, $\mathbf{F}^H$ spans a space where the vector basis are complex harmonically related sinusoids, where each sinusoid is at a multiple of the frequency $\frac{f_s}{N}$. As a consequence, the DFT transformation is a projection of a given signal vector $\mathbf{s}$, defined in a euclidean space spanning time domain, to the aforementioned space defined by $\mathbf{F}^H$ where the vector basis are complex harmonically related sinusoids spanning a frequency domain.

$$\mathbf{r_k}\bar{\mathbf{r}}_\mathbf{l}=\frac{1}{\sqrt{N}}\bar{\mathbf{W}}_\mathbf{k}\frac{1}{\sqrt{N}}\mathbf{W_l}=\frac{1}{N}\sum_{n=0}^{N-1}[w^*(n)]^k[w(n)]^l=\frac{1}{N}\sum_{n=0}^{N-1}e^{-i\frac{2\pi n}{N}k}e^{-i\frac{2\pi n}{N}l}=\frac{1}{N}\sum_{n=0}^{N-1}e^{i\frac{2\pi n}{N}(l-k)}=\begin{cases}0 & l\neq k\\ 1 & l=k\end{cases} \tag{51}$$

In Equation 51, and using a graphical interpretation within the framework of the complex plane, we have used the fact that if $l-k\neq 0$ then we are summing complex numbers sampled regularly all around the unit circle, which necessarily must yield zero. However, if $l-k=0$, then we have $\mathbf{r_k}\bar{\mathbf{r}}_\mathbf{l}=\frac{1}{N}\sum_{n=0}^{N-1}1=1$, which concludes the proof that the columns of $\mathbf{F}^H$ are orthonormal, and hence that $\mathbf{F}^H$ spans a subspace upon which any complex signal can be projected.

c. We have constructed the K-points DFT-CLMS algorithm, which was applied to the FM signal $y(n)$ for evaluation. Again, the use of the CLMS algorithm instead of ACLMS is justified since the FM signal $y(n)$ is circular (see Figure 45). The time-frequency spectrogram obtained demonstrates that while the DFT-CLMS filter successfully tracks the non-stationary frequency $f(n)$, the filter appears to keep some "memory" of the dominant frequencies identified. To formulate this mathematically, let f_d be the dominant frequency component in the frequency spectra at time n_d and n be the time variable, then for all $n>n_d$ the power associated with frequency f_d remains high in the DFT-CLMS spectrogram, as seen from Figure 48. We suggest that possible causes for this phenomenon may include the following: **(a)** since at every instant n the weight vector has length K where $K\gg 1$, then the degrees of freedom it offers is such that the filter output can track the signal $y(n)$ with only a small change applied to each weight, hence the weight vector varies very little at each time step, resulting a large weight memory; **(b)** since frequency can only be physically defined on a finite time period rather than in an instantaneous fashion, and since the DFT-CLMS filter uses a gradient-descent adaptive algorithm which dynamically corrects its weights at every instant n to minimise the cost function J rather than in a block-filter mode, then there is no clear mechanism for error back-propagation to correct previously learnt weights; and **(c)** since both the filter input vector $\mathbf{x}$ and the signal $e(n)$ contain complex elements with small norm ($e(n)$ scales with $y(n)$), then from the weight equation it can be seen that from a numerical perspective, the weight change per time step will be small, resulting in the visible "lagging" spectrum which differs from the true power spectrum.

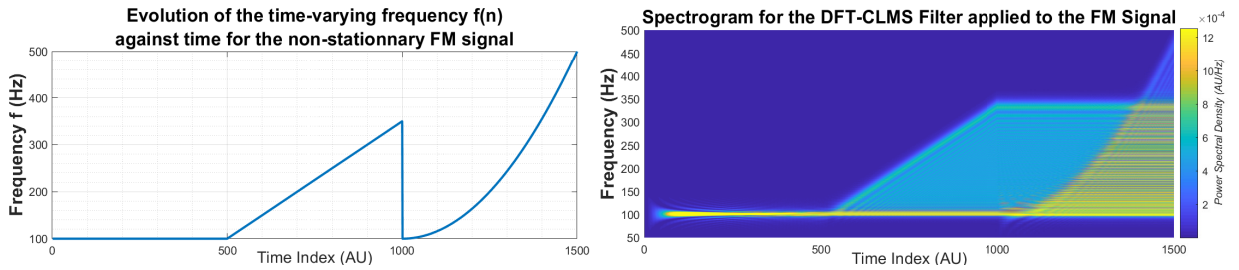

Figure 48: Left: Time-varying frequency $f(n)$ characterising the FM signal $y(n)$. **Right:** Spectrogram of the FM signal $y(n)$ obtained from the DFT-CLMS real-time spectrum analyser, illustrating system weight memory.

We now compare DFT-CLMS to the adaptive AR-spectrum analyser previously implemented. This comparison is similar to comparing classical spectrum estimation against model-based methods as discussed in Section 1.5 of this report. While the AR-spectrum analyser method yields a power spectrum estimate which is characterised by no spectral side-lobes, minimal trace variability and no dependence of the spectral resolution on the inverse of the input signal length, its performance can only be as good as how fitting the assumed underlying model is for the particular input. This requires the analyst to estimate fitting values of model hyper-parameters, including in this case the model order p or the learning rate μ. To conclude, the efficacy of the adaptive AR-spectrum analyser relies on our ability to select an appropriate model and model order for a given dataset.

On the other hand, the DFT-CLMS algorithm offers a model-free approach where no assumption is made on the given data, hence its applicability is unlimited since any signal can be approximated as a linear combination of N complex harmonically-related sinusoids with least squares residual. However, this is traded against the existence of spectral lobes, increased trace variability, and dependence of the frequency resolution on the inverse of the signal length. Most importantly, as can be seen from Figure 48, the DFT-CLMS yields a spectrogram representation which, although it can be accurately interpreted by the human observer who can identify that the "lagging" spectral peaks should be ignored, is highly biased and not suitable as such for machine interpretation. The following section details a proposed method for improving the DFT-CLMS algorithm.

Specifically, we propose that regularisation by a leakage term γ may allow the weight vector at instant n to efficiently "forget" its old value, such that the resulting DFT-CLMS spectrogram should be more accurate. We now consider the weight update equation given by $\mathbf{w}(n+1)=(1-\gamma\mu)\mathbf{w}(n)+\mu e^*(n)\mathbf{x}(n)$ and in this case, the leakage coefficient γ can be interpreted as a "forgetting rate". The results are shown below in Figure 49 for different values of γ: we observe that for small γ ($\gamma=0.001$), regularisation results in an insufficient loss of memory in the spectrogram. Conversely, for large values of γ ($\gamma=0.5$), regularisation bias is excessive and spectral leakage is such that the spectrogram performance becomes poor: while we would expect a narrow spectral peak tracking $f(n)$, the actual spectrogram exhibits a wide trace. Lastly, for the leakage coefficient $\gamma=0.01$, the Leaky DFT-CLMS performance is largely improved with respect to the DFT-CLMS filter without regularisation, as filter memory has been suppressed and we observe a narrow spectral peak efficiently tracking $f(n)$.

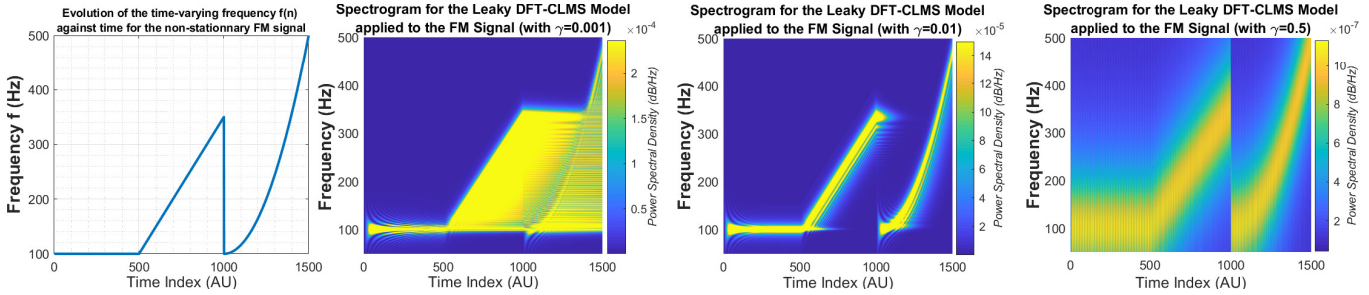

Figure 49: Left: Time-varying frequency $f(n)$ characterising the FM signal $y(n)$. **Right:** Spectrogram of the FM signal $y(n)$ obtained from the Leaky DFT-CLMS real-time spectrum analyser for different values of leakage parameter γ.

The above is confirmed by Figure 50, where the frequency associated with the maximum value in the power spectrum at each instant n is tracked for $\gamma=0.1$ and $\gamma=0.5$ to yield an estimate $\hat{f}(n)$ of the non-stationary frequency $f(n)$. It can be seen that for γ values in the order of 0.01, the resulting frequency estimate closely follows the true frequency $f(n)$, while for high leakage coefficients ($\gamma=0.5$), model stability is compromised as the estimate $\hat{f}(n)$ oscillates around the true $f(n)$. In fact, for $\gamma=0.1$, the Leaky-DFT-CLMS method yields an estimate $\hat{f}(n)$ tracking the true $f(n)$ with more accuracy than was achieved under the CLMS-AR(1) method (compare with Figure 46).

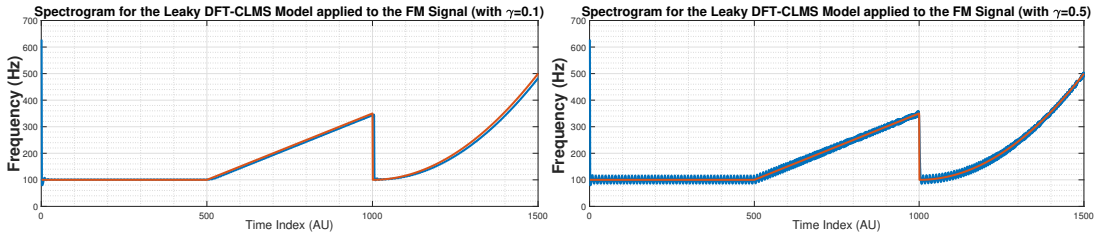

Figure 50: Frequency estimate obtained from the DFT-CLMS model applied to the non-stationary FM data $y(n)$ for different leakage coefficients γ. It can be seen that suppression of trailing spectra comes at the cost of model instability.

d. The DFT-CLMS filter is now applied to a segment of length $N=1200$ of the `POz` EEG signal. Similarly to previous sections, the `POz` signal was pre-processed via standardisation to zero mean and variance 1 through the application of the `zscore` function. This was done in order for the DFT-CLMS input vector $\mathbf{x}$ and the desired filter output $y(n)$ to be expressed on the same scale to prevent numerical instability and weights explosion. The resulting spectrograms are shown in Figure **??** for the cases of no regularisation, acceptable regularisation $\gamma=0.01$, and excessive regularisation $\gamma=0.1$.

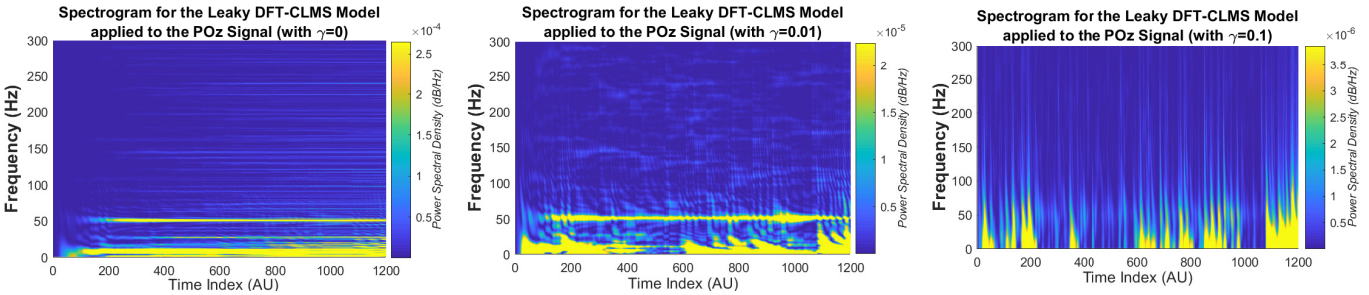

Figure 51: Spectrogram of a section of length $N=1200$ of the EEG signal `POz` obtained from the Leaky DFT-CLMS real-time spectrum analyser for different values of leakage parameter γ.

It can be observed that in the absence of regularisation, the trailing of spectral peaks across time steps n causes some frequency component in the low-frequency EEG activity range ($[0:40]$Hz) to be masked. As a result, only the stationary 50Hz mains component is clearly visible. In contrast, for excessive regularisation, low-frequency bias is introduced to the model, and masks not only the low-frequency EEG range but also the stationary 50Hz hum. Lastly, the trade-off between reduced regularisation bias and improved trailing spectrum suppression is met for $\gamma=0.01$, where both the stationary 50Hz mains component and the non-stationary EEG signal component are visible. Specifically, the 13Hz SSVEP response as well as its first harmonic at 26Hz can be observed, which validates this method.

4 From LMS to Deep Learning

a. The behaviour of the LMS filter in one-step-ahead prediction setting is now investigated for non-stationary streaming data. Specifically, the zero-mean version of the non-stationary time-series $y(n)$, which we denote by $y_0(n)$, was fed as input to the LMS auto-regressive predictor of order $M=4$ and with learning rate $\mu=1\times10^{-5}$, yielding an output prediction $\hat{y}(n)$ which is an estimate for the true $y_0(n)$.

To evaluate the LMS prediction performance, y_0 was plotted against its estimate $\hat{y}$, as shown in Figure 52, where the equation of the corresponding best fit line is given in the form $y_0=a\hat{y}+b$. In the ideal prediction setting we expect $y_0=\hat{y}$ (hence $a=1$ and $b=0$), however in the experimental case we obtain the coefficients $a\approx1.2$ and $b\approx0.03$ where $a>1$ indicates that the LMS filter output $\hat{y}$ undershoots the true signal $y_0(n)$. This could result from an insufficient learning rate μ constraining $\hat{y}$ to small variations, which was verified by selecting $\mu=5\times10^{-4}$, under which the best fit line equation was given by $a\approx1.0$ and $b\approx0.03$, where $a=1$ demonstrates efficient tracking of the input signal y_0. This is further confirmed by Figure **??**, where the time-variation of y_0 and $\hat{y}$ are shown for the aforementioned values of μ. It can be seen that increasing μ prevents undershooting and as expected reduces the duration of the transient response in $\hat{y}(n)$.

The mean-square error (MSE) and prediction gain, given by Haykin and Li (1995) as $R_p=10\log_{10}\left(\frac{\sigma^2_{\hat{y}}}{\sigma^2_e}\right)$, were taken as performance measures in the prediction setting and computed for 200 random weights initialisation, for the two selections of hyper-parameter μ aforementioned. Since MSE expresses the time-averaged error power while R_p is the ratio of the output variance to the prediction error variance expressed in decibels, performance increase is denoted by a reduction in MSE and an increase in R_p. The probability distributions of the MSE and R_p measures are shown in Figures 52 and 54 to assess the effect of random weight initialisation on the filter performance. In all cases, the values are normally distributed around the indicated mean value, which confirms that MSE and R_p are unbiased performance measures. Overall, increasing the learning rate from $\mu=1\times10^{-5}$ to $\mu=5\times10^{-4}$ causes a MSE reduction from 19.7dB to 9.24dB and a prediction gain increase from 3.36dB to 14.08dB, which illustrates the improved performance under $\mu=5\times10^{-4}$.

NOTE: To save computing time, in all subsequent exercises the results shown were obtained from one unique realisation where all weights were initialised at 0. In addition, MSE and R_p were consistently computed on the steady-state step range $[200:N]$.

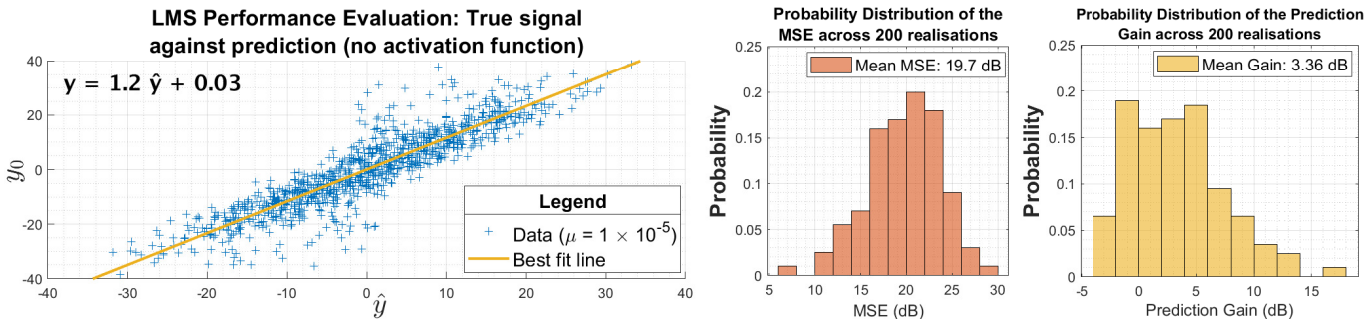

Figure 52: Left: True signal y_0 against LMS prediction $\hat{y}$, where $\hat{y}$ was averaged over 200 random initialisation of the filter weights. **Middle:** Histogram of MSE obtained for LMS prediction under 200 random initialisation of the filter weights. **Right:** Histogram of prediction gain obtained under 200 random initialisation of the filter weights. The learning rate is $\mu=1\times10^{-5}$.

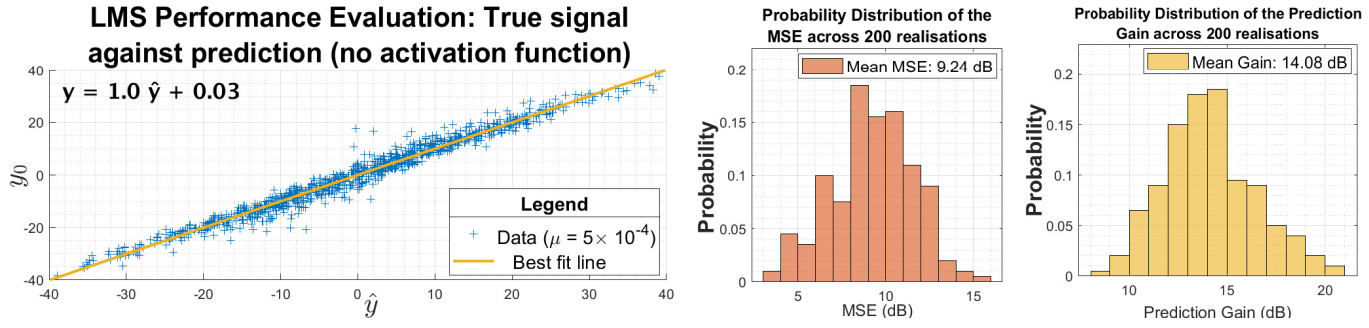

Figure 53: This figure is equivalent to Figure 52, where the learning rate is $\mu=5\times10^{-4}$. The prediction performance is improved.

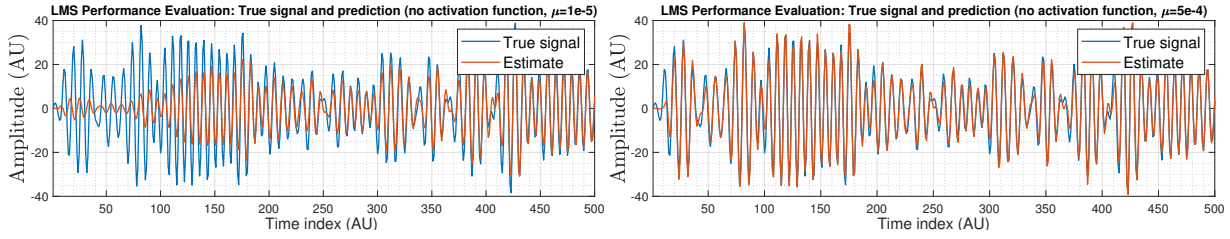

Figure 54: Overlay plot of the true signal $y_0(n)$ and the estimate $\hat{y}$ obtained for different learning rates μ and averaged over 200 realisations showing a detail for the first 500 time steps.

b. We now consider that typically, the generating process for the data is unknown and non-linear. Therefore, to capture data non-linearity, we apply a non-linear activation function $\mathbf{\Phi}$ to the weighted output $net(n)$ of the LMS filter. The resulting algorithmic structure is knwon as the dynamical perceptron. In this setting, the weight update equation is given by Equation 52, where $\mathbf{\Phi}'$ denotes the derivative of the activation function $\mathbf{\Phi}$ with respect to the weight vector $\mathbf{w}$.

The activation function selected is the hyperbolic tangent function `tanh`, which is highly non-linear away from zero and bounded between ± 1. Since y_0 oscillates approximately between ± 40, the application of the `tanh` activation function to the output of the LMS filter causes the dynamic perceptron to saturate at ± 1 and fail to track y_0.

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu \nabla_{\mathbf{w}} J = \mathbf{w}(n) + \mu \mathbf{\Phi}'[net(n)] e(n) \mathbf{x}(n) = \mathbf{w}(n) + \mu (1 - \tanh^2[net(n)]) e(n) \mathbf{x}(n) \tag{52}$$

Although the `tanh` activation function is inappropriate in this case, we could down-scale the signal y_0 such that its variations are comprised within the bounds ± 1. As an example, the data y_0 could be standardised to zero mean and constrained within the range $[-1,1]$ via the min-max normalisation shown in Equation 53. Additionally, for the LMS output $\hat{y}$ to slide across its full possible range, we increase the learning rate from $\mu = 1 \times 10^{-5}$ to $\mu = 1 \times 10^{-1}$. We suggest that the same effect could be achieved by introducing an activation gain β such that activation function is defined as $\tanh(\frac{\beta}{2} net(n))$, instead of modifying μ.

$$y_0'(n) = 2 \frac{y_0(n) - y_{0,min}}{y_{0,max} - y_{0,min}} - 1 \tag{53}$$

The plots of y_0 and y_0' against LMS prediction are shown in Figure 55 for the first 500 steps, where it can be seen that with the raw data y_0, the perceptron output saturates, MSE is high at 22.22dB and prediction gain is poor at -23.55dB. The latter is a consequence of the fact that $\sigma_{\hat{y}}^2$ will be small given that $\hat{y}$ is constrained within $[-1,1]$, while as discussed previously σ_e^2 will be large. Conversely, in the case of the standardised input y_0', saturation is avoided: in this case, the reduction in MSE from 22.22dB to -20.86dB is not only indicative of a performance improvement, but also of the scaling down of the error resulting from the scaling down of the data y_0 to y_0'. Here, R_p is particularly useful as it is defined as a ratio of output variance to error variance, and hence is scaling-independent. The increase in R_p from -23.55dB to 11.46dB demonstrates an improved LMS prediction performance under data min-max standardisation, which is visually confirmed by the time-evolution plots of Figure 55 as the LMS predictor manages to track amplitude variations in y_0 more efficiently under application of Equation 53. Lastly, let us note that constraining the data to $[-1:1]$ also prevents weight saturation; indeed, in the case of large data inputs, we have $\mathbf{\Phi}'[\mathbf{net}(\mathbf{n})] \approx 0$ and as a result the weights remain unchanged, as expressed from Equation 52.

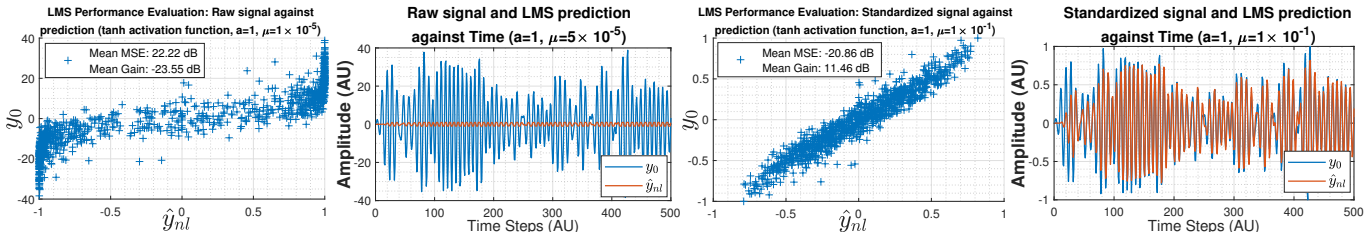

Figure 55: Left: True signal y_0 against dynamical perceptron output $\hat{y}$ under the application of a `tanh` activation function, along with the time evolution of $\hat{y}$ and y_0, for raw data. **Right:** Standardised signal y_0' against dynamical perceptron output $\hat{y}$ under the application of a `tanh` activation function, along with the time evolution of $\hat{y}$ and y_0, for standardised data. The weights were initialised at zero.

c. To prevent output saturation of the dynamical perceptron, we scale the `tanh` activation function by a factor a. The factor a is applied as an output scaling term only and was ignored in the $\mathbf{\Phi}'(n)$ term in Equation 52, where it would act as a learning rate gain. The validity of this approach was confirmed by experimental results. We propose that the dynamic range of the perceptron output should include the range $[y_{0,min}, y_{0,max}]$. Since the `tanh` function varies between ± 1, $a \times$`tanh` has range $\pm a$, where the condition on a is given by $a \geq \max(|y_{0,min}|, |y_{0,max}|) \rightarrow a \geq 38.9$. On the other hand, increasing a also causes the output of the perceptron to be more sensitive to variations in the scalar product $net(n) = \mathbf{w}^T \mathbf{x}$, which in turn could compromise weight convergence as well as the overall model stability. We therefore propose that the range of acceptable values for a is $[40:150]$.

This is confirmed in Figure 56, where saturation occurs only for $a < 38.9$, and where for the large $a = 200$ the predictor becomes unstable. In fact, since `tanh(x)` tends towards ± 1 for $x \rightarrow \pm\infty$ and is never exactly ± 1, we suggest that selecting a such that the perceptron output dynamic range extends past the minimum range ± 38.9 further improves the predictor performance. This is confirmed as increasing a from 40 to 80 causes a MSE reduction from 7.78dB to 5.32dB and a prediction gain increase from 14.66dB to 17.3dB. More generally, both performance metrics indicate that the nonlinear perceptron outperforms the linear perceptron, since for the same learning rate $\mu = 1 \times 10^{-5}$ the latter scored a higher MSE of 19.7dB and a lower prediction gain $R_p = 3.36$dB. In fact, the nonlinear perceptron also outperforms the linear perceptron with $\mu = 5 \times 10^{-4}$, as shown in Figure 55. The performance metrics for different versions of the LMS perceptron are summarised in Table 9.

d. The original data $y(n)$ is now considered: a constant bias input is added to our LMS perceptron model to account for the non-zero mean of $y(n)$. The criteria on a remains $a \geq \max(|y_{0,min}|, |y_{0,max}|)$, which now yields $a \geq 48.8$. We select $a = 60$, and maintain a filter order $M = 4$ and $\mu = 1 \times 10^{-5}$ to enable comparison with previous results. Here, we have specially chosen to compute the MSE and prediction gain R_p on the full time-step range $[0:N]$, as opposed previous sections where these performance metrics were computed over the steady-state range $[200:N]$, as shown in Figure 57. This was done such as to include the transient period in our performance metric, where reduction of that transient response becomes part of our design objective. Figure 57 also includes the plots of true signal y against prediction $\hat{y}$, as well as weight convergence plots, learning curves and time-domain comparison of y and $\hat{y}$.

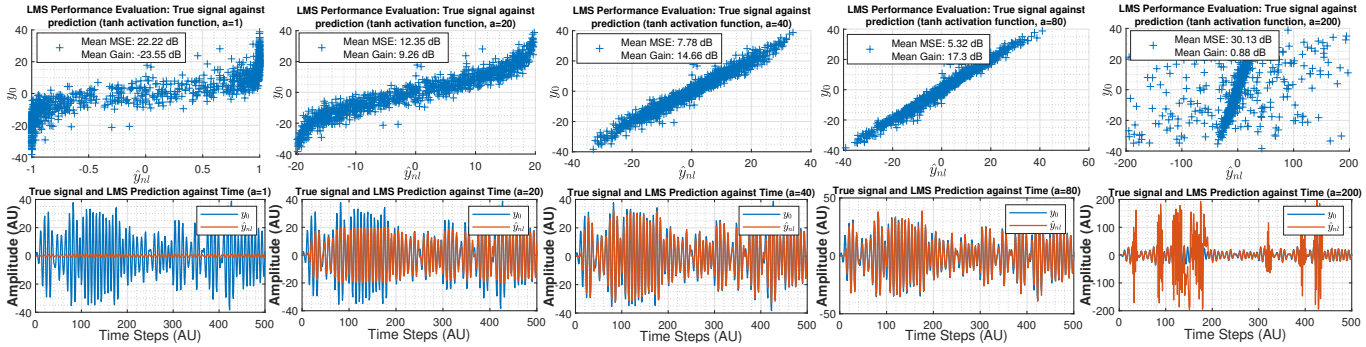

Figure 56: True signal y_0 against dynamical perceptron output $\hat{y}$ under the application of a `tanh` activation function, along with the time evolution of $\hat{y}$ and y_0, for different selections of gain a. The weights were initialised at zero. The learning rate is $\mu = 1\times10^{-5}$.

	MSE (dB)	R_p **(dB)**
Linear perceptron with $\mu = 1\times10^{-5}$	19.7	3.36
Linear perceptron with $\mu = 5\times10^{-4}$	9.24	14.08
Nonlinear perceptron with $a = 1$ **and** $\mu = 1\times10^{-5}$	22.22	-23.55
Nonlinear perceptron with $a = 40$ **and** $\mu = 1\times10^{-5}$	7.78	14.66
Nonlinear perceptron with $a = 80$ **and** $\mu = 1\times10^{-5}$	5.32	17.3
Nonlinear perceptron with $a = 20$ **and** $\mu = 1\times10^{-5}$	12.35	9.26
Nonlinear perceptron with $a = 200$ **and** $\mu = 1\times10^{-5}$	30.13	0.88

Table 9: Summary of performance metrics (MSE and R_p) for different versions of the LMS perceptron, as shown in the Figures of this section. It can be seen that when a is in the selected range, the nonlinear perceptron outperforms the linear perceptron.

It can be observed that the filter weights and learning curves converge over the scale of 200 steps. This represents a transient response of 20% of the full signal duration. As can be seen from Table 10, the performance metrics MSE and R_p illustrate that the prediction performance is improved during steady-state $[200:N]$ as opposed to the transient-state $[0:200]$. Consequently, the dynamical perceptron performance averaged over the full signal length can be improved by reducing the proportion of the transient response to N. While we could instinctively propose to simply increase the learning rate μ, the current setting is such that a slight increase in μ would make the model unstable, prompting the need for an alternative solution described next.

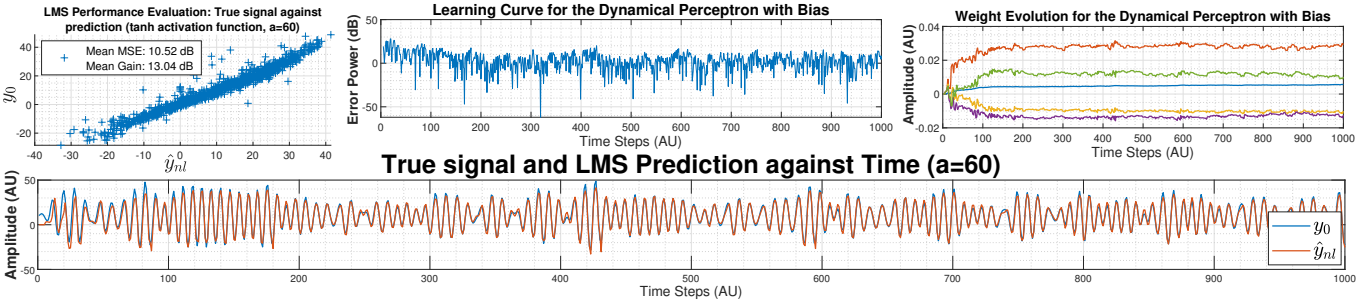

Figure 57: Top: True signal y against dynamical perceptron output $\hat{y}$ under the application of a `tanh` activation function. Learning curve and weight evolution for the dynamical perceptron output. **Bottom:** Time evolution of $\hat{y}$ and y. The weights were initialised at zero. The learning rate is $\mu = 1\times10^{-5}$, and filter order is $M = 4$.

	$[1:N]$	$[1:200]$	$[200:N]$
MSE (dB)	10.52	15.74	6.69
$\mathbf{R_p}$ **(dB)**	13.04	10.24	16.07

Table 10: MSE and prediction gain R_p computed over the full-range, stead-state and transient-state ranges of time-steps for the results shown in Figure 57. The results demonstrate that, as expected, the prediction performance is improved in the steady-state region.

e. In the previous exercise, it was highlighted that we are only performing a singe weight update per time-step with a potentially small number of time-steps N, which causes the transient response of the LMS perceptron to represent a large portion of the total signal length. To overcome this limitation, we will pre-train the perceptron weights $\mathbf{w}$, starting from $\mathbf{w} = \mathbf{0}$, by over-fitting to the first 20 samples of $y(n)$. This operation was repeated for 100 epochs to yield the set of weights $\mathbf{w_{init}}$, which can be used as initial condition for the weight vector $\mathbf{w}$.

The pre-training of the weights is shown in Figure 58, while $\mathbf{w_{init}}$ is used as initial weigth condition to predict the entire time-series $y(n)$ in Figure 59. As expected, steady-state is reached almost immediately and the overall performance metrics MSE and R_p indicate an improved prediction performance when compared to the case of no weight pre-training presented in Figure 57. Indeed, pre-training results in an MSE reduction from 10.52dB to 7.16dB and a prediction gain increase from 13.04dB to 16.31dB, confirming that weight pre-training is efficient. Clearly, the performance improvement yielded by weight pre-training relies upon how representative is the training dataset of any arbitrary dataset independently generated from the same process.

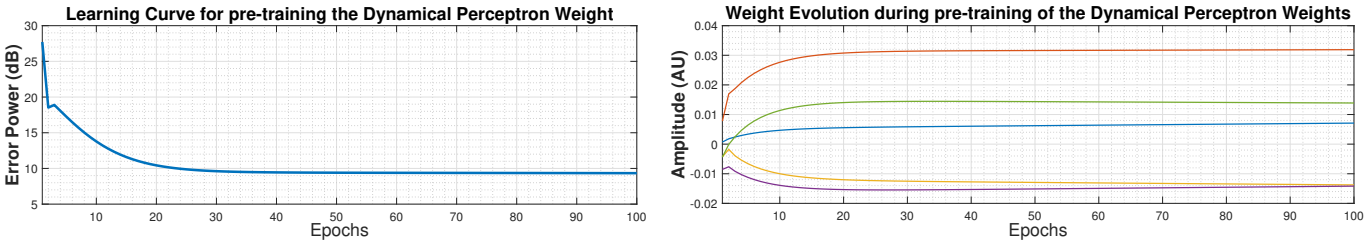

Figure 58: Right: Learning curve for pre-training the weights $\mathbf{w}$ using the first 20 samples of $y(n)$ under 100 epochs. **Left:** Training of the filter weights $\mathbf{w}$ across epochs, yielding $\mathbf{w_{init}}$ after 100 epochs. The weights were initialised at zero. The learning rate is $\mu = 1\times10^{-5}$, and filter order is $M = 4$.

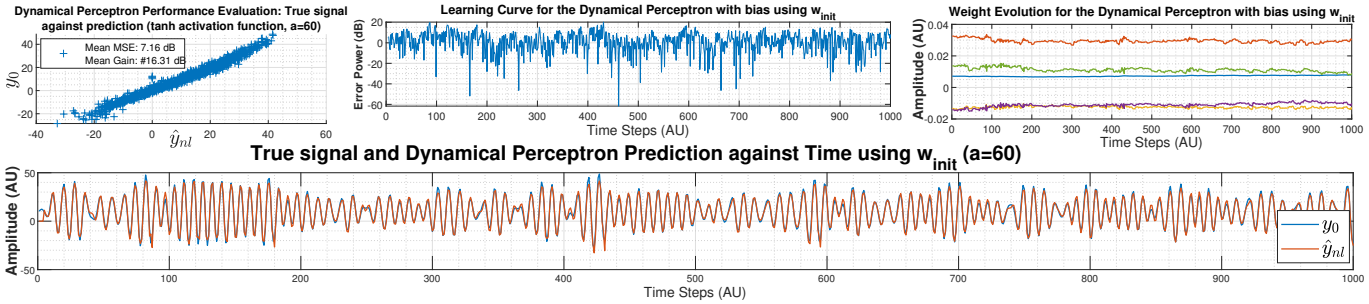

Figure 59: Top: Equivalent of Figure 57, where the initial weights $\mathbf{w_{init}}$ were obtained from pre-training the model (shown in Figure 58).

f. In a deep network, many simple "dynamical perceptron" models are arranged together in a sequence of layers to offer the degrees of freedom required to model real-world problems. In this framework, the output of each layer is used as input to each component of the subsequent layer, where data propagates from the input layer, which is analogous to the dynamical perceptron input vector $\mathbf{x}$, via one or many hidden layers (typically many in a "deep" network), to the output layer, which in one-step-ahead prediction setting is a scalar estimate $\hat{y}(n)$ of the signal $y(n)$, where $y(n)$ is the teaching or desired signal, that is, the signal to be predicted. Ultimately, let $J(n)$ be an error function quantifying the difference between the actual and desired network output, our goal is to update the weights connecting each layer of dynamical perceptrons to the next such that $J(n)$ is minimised. This is a typical case of credit assignment problem, where we aim to determine how each weight element contributes to the output error expressed as $J(n)$.

The backward propagation training algorithm offers a solution to this problem by propagating the errors from output layer to input layer, accordingly with the instantaneous network weights. The weight update equation for each weight w_{ij} connecting the i-th element of one layer to the j-th element of the next is given by $w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}$, where $\Delta w_{ij} = \mu\delta_i^m \mathbf{\Phi}_j^{m-1}$.

In this expression, μ is the learning rate and $\mathbf{\Phi}_j^{m-1}$ denotes $\mathbf{\Phi}[\mathtt{net}_j^{m-1}(n)]$, the output of the j-th dynamical perceptron on layer $m-1$ at instant n after application of the nonlinear activation function $\mathbf{\Phi}$ to the weighted sum (including bias) formed from the output of the preceding layer $m-2$. The weight update equation proposed uses the so-called delta-rule, where δ_i^m is the error measure from the i-th element on the m-th perceptron layer. Starting for the output layer M, where $\delta_i^M = \mathbf{\Phi'}[\mathtt{net}_i^M(n)]e_i(n)$, we propagate backwards as $m = M, M-1, \ldots, 2$ using the expression $\delta_i^{m-1} = \mathbf{\Phi'}[\mathtt{net}_i^{m-1}(n)]\sum_j w_{ij}^m \delta_j^m$. This ensures that each weight is corrected to an extent proportional to the error generated on the subsequent layer weighted by the contribution of the source perceptron to that error. As a conclusion, at each time instant n, data is propagated forward through the deep network, then δ errors are propagated backwards to subsequently update all the network weights and solve the credit assignment problem. Let us note that we have used above $e_i(n) = y_i(n) - \hat{y}_i(n)$ in the context of a prediction setting.

g. We now consider an example of highly non-linear prediction problem tackled with a simple perceptron, a nonlinear perceptron, and a deep network. In this setting, the signal $y(n)$ to be predicted is a non-linear combination of 10 signals $x_i(n)$, which can be collected in a vector $\mathbf{x}$ such that $y(n)$ is a nonlinear function of $\mathbf{x}$, where $y(n)$ appears to be zero-mean and has length $N = 100$. The activation function for this network is the `relu` function. The signal $y(n)$ is divided into two sections of equal length for $n: 0 \rightarrow 50$ and $n: 51 \rightarrow 100$, where one section is used for training the predictor and learning the optimal weights, while the second is used for testing purposes where we assume the weights are no longer allowed to change and the output prediction error is computed to assess performance and training efficacy. Training and testing are performed along 20000 epochs, with a learning rate $\mu = 0.01$ and noise power equal to 0.05. We also note that $y(n)$ is bounded between ± 1, which suggests that there is no risk for output saturation for the dynamical perceptron with a potential `tanh` activation function.

As expected, the learning curves of training loss against epochs are monotonously decreasing and converging, both in the case of the dynamical perceptron (linear and nonlinear) and the deep network, as weights are learned to minimise output prediction error. In addition, Table 11 illustrates that the deep network converges to a smaller training loss than the dynamical perceptron, which may indicate superior learning performance or suggest model over-fitting. The non-linear dynamical perceptron with `tanh` activation function also converges to a lower training loss than its linear counterpart, which may equally indicate improved learning or over-fitting.

Across the first epochs, the perceptron and deep network perform better under testing than under training in terms of loss minimisation. This is because while during training the network weights were initialised randomly, during testing they were initialised at the converged pre-trained weights, yielding a better initial prediction performance. Interestingly, while the testing

loss for the simple dynamical perceptron converges to the training loss as the number of epochs increases, in the case of the deep network the testing loss increases along epochs and exceeds the converged training loss.

This clearly suggests over-fitting, where the deep network not only captures the dynamics of the noise-free component of $y(n)$, but also converges to fit the noise component to the model during the training phase. This results in an increasingly poor testing performance in terms of loss, given that while the dynamics of the noise-free component of $y(n)$ are conserved from the training signal section to the testing signal section, the noise dynamics will be constantly changing. Lastly, we also note that with application of the `tanh` activation function, the nonlinear perceptron achieves a smaller minimum testing loss than the linear perceptron, which further demonstrates that the application of a non-linear activation function improves perceptron performance.

	Minimum Training Loss	**Minimum Testing Loss**
Simple Dynamical Perceptron (linear)	0.1	0.09
Simple Dynamical Perceptron (tanh)	0.09	0.08
Deep Network (relu)	0.03	0.05

Table 11: Measured results from the learning curves of the simple dynamical perceptron with linear output and `tanh` activation function, compared to a deep network within t he framework of prediction of $y(n)$.

h. In previous results, the noise power was set at $n_p = 0.05$. We now investigate the effect of the signal-corrupting noise on the performance of both the simple linear perceptron and the deep network, as shown below. We initially consider the case of a noiseless input signal where $n_p = 0$. The corresponding learning curves are shown in Figure 60 and the converged training and testing losses are shown in Table 12. We observe, as expected, that all algorithms converge to smaller training losses than in the noisy $n_p = 0.05$ case (see Table 11). We suggest that training and testing losses are bounded from below by the noise power n_p.

We now compare the dynamical perceptron to the deep network. In the case of the simple perceptron, the training loss never exceeds the testing loss. This is expected given that during the training stage, the weights are over-fitted to the data: in the ideal scenario of a noiseless data input, the weights necessarily converge to the optimal solution capturing completely the underlying dynamics of the signal $y(n)$. As a result, when tested on previously unseen testing data, the loss exhibited by both the simple perceptron and the deep network must necessarily equal or exceed the training loss, where the latter defines the minimum achievable loss.
In comparison, the deep network training error initially exceeds the testing error for a short transient period: one suggested explanation is that the computational burden represented by the large number of weights characterising a deep network causes the latter to exhibit some transient response as back-propagation fully adjusts all network weights. Conceptually, the complexity of the deep network, offering additional degrees of freedom, also allows the latter to score zero training loss as the weights converge to the exact solution fully characterising the signal $y(n)$. Similarly, the deep network also scores smaller testing loss than the simple perceptron, and therefore generalises better. To conclude, for low noise power n_p, the deep network outperforms the simple perceptron.

We now compare the case $n_p = 0.05$ to the case $n_p = 0$. We observe that while in the noiseless case the simple perceptron scores a minimum testing loss exceeding that from the noisy case, the deep network on the other hand achieves a lower testing loss in the noiseless case. This further demonstrates that the adaptability of the deep network when trained from noiseless datasets is significantly improved. We will now consider the other extreme where the noise corrupting the input data is relatively large.

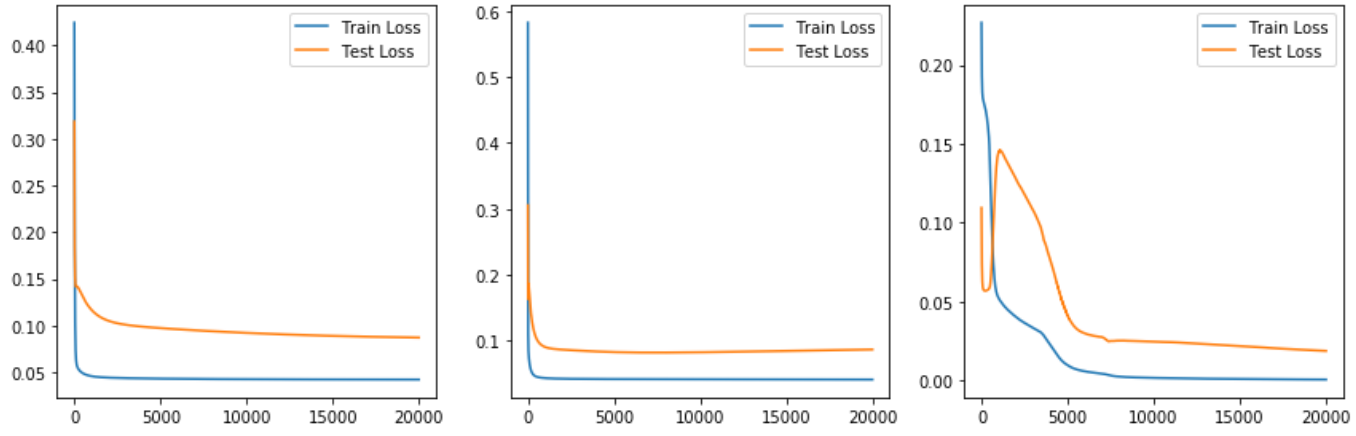

Figure 60: Leaning curves illustrating training and testing loss across epochs for the problem of predicting the highly non-linear signal defined in this section with $n_p = 0$.

	Minimum Training Loss	**Minimum Testing Loss**
Linear perceptron	0.05	0.09
Nonlinear perceptron	0.03	0.10
Deep network	0.00	0.03

Table 12: Summary of the minimum testing and training losses obtained for simple dynamical perceptrons and a deep network in the problem of predicting the highly non-linear signal defined in this section with $n_p = 0$.

We now consider the scenario of relatively large corrupting noise with $n_p = 0.5$, and similarly compare the performance of the simple dynamical perceptron with that of the deep network. The resulting learning curves are shown in Figure 61. We interestingly note that in the case of the deep network, the training loss fails to decrease monotonously and oscillates with an amplitude of approximately 0.2. Moreover, the long-term behaviour of the testing loss across epochs displays a slow rapidly oscillating increase. This unstable behaviour can result from over-fitting: in this case, the deep network attempts to fit the noise to the model weights, and assuming the noise has no underlying structure, the over-specific set of weights will randomly succeed or fail to explain and predict the data, resulting in large loss oscillations. More generally, the long-term increase of testing loss after 4000 epochs is globally indicative of over-fitting, as previously explained.
In contrast, in the case of the dynamical perceptron, the behaviour observed is similar to that under $n_p = 0.05$. Training loss decreases monotonously across epochs, initially exceeds the testing loss and ultimately converges to a reduced loss. Again, we note that the converged training loss of the dynamical perceptron increases with the noise power n_p, as summarised in Table 13. Critically, the long-term testing loss of the deep network after 20000 epochs approximately equals 0.9, and exceeds the long-term testing loss of the dynamical perceptron at 0.55. This suggests that for highly noisy data, and for a large number of epoch iterations, the simple perceptron outperforms the deep network.

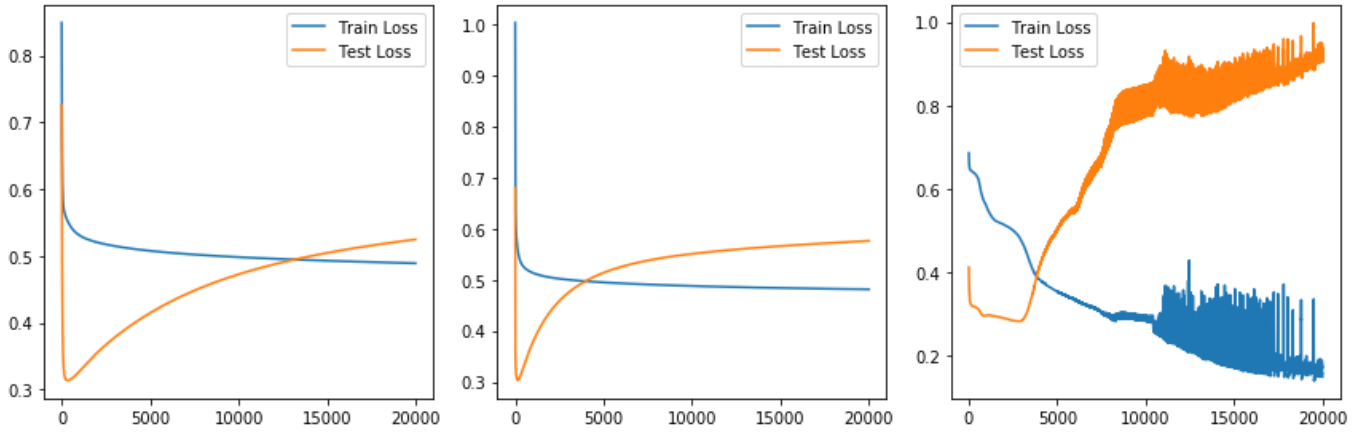

Figure 61: Leaning curves illustrating training and testing loss across epochs for the problem of predicting the highly non-linear signal defined in this section with $n_p = 0.5$.

	Minimum Training Loss	**Minimum Testing Loss**
Linear perceptron	0.31	0.5
Nonlinear perceptron	0.30	0.45
Deep network	0.15	0.3

Table 13: Summary of the minimum testing and training losses obtained for simple dynamical perceptrons and a deep network in the problem of predicting the highly non-linear signal defined in this section with $n_p = 0.5$.

To conclude, in the case where near-deterministic training data is available, with minimal noise power n_p, we expect deep networks to perform better than the simple dynamical perceptron as the former exhibits sufficient degrees of freedom to fully fit the data y to the model weights. This should result in superior prediction performance on newly seen testing data. In contrast, the perceptron weights will only reflect a crude approximation of the true dynamics of y, and training will be less efficient, resulting in inferior testing-data performance.
Conversely, when only highly noisy data is available, the simple perceptron may outperform deep networks. Indeed, the restricted degrees of freedom offered by the simple perceptron may be sufficient to capture some underlying dynamics of the noise-free component of y, while deep networks present the possibility of over-fitting the noise component of y to the model, resulting in an over-specific set of trained weights which perform well on training data but fail to generalise and perform poorly on newly-seen testing data. This is exacerbated for a high number of epoch iterations, as expected.

Therefore, one major drawback of using deep networks for prediction is that the performance of deep network predictor strongly relies upon the quality of the training dataset. This ultimately prompts the need for efficient data-mining and pre-processing methods which aim to guarantee the extraction of rich and clean training data. Many techniques introduced in this report, including ALE/ANC filtering, subspace and dimensionality reduction methods, may be used as part of a data pre-processing stage implemented previous to feeding the data to a deep network. Lastly, adjusting the deep network hyper-parameters accordingly, and selecting relevant activation functions, may improve the overall performance of the deep network.