Spectrum Estimation and Adaptive Signal Processing Coursework

Sebastian Grubb
sg3510@ic.ac.uk

June 7, 2014

Contents

Part 1

Nonparametric Spectrum Estimation

1.1 Properties of Power Spectral Density (PSD)

Approximation in the definition of PSD We have the first definition of PSD:

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

We also have the second definition of the PSD which is:

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

We aim to show that the two are equivalent under the assumption that the covariance sequence decreases rapidly:

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

From this:

$$P(\omega) = \lim_{N\to\infty} \mathbb{E}\left\{ \frac{1}{N} \left| \sum_{n=0}^{N-1} x(n)e^{-jn\omega} \right|^2 \right\} \qquad = \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} \mathbb{E}\left\{x(n)x^*(m)\right\} e^{-j\omega(n-m)} \tag{1.4}$$

Notice the double summation, which we reduce to one summation:

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

$$\{f(0) + ... + f(N-1)\} + ... + \{f(-(N-1)) + ... + f(0)\}$$

i.e. we have $f(0)$ N times, $f(-1)$ & $f(1)$ $N-1$ times until we reach $f(N-1)$ & $f(-(N-1))$, which only appear once. This gives us:

$$\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} f(n-m) = \sum_{k=\_(N-1)}^{N-1} (N-|k|)f(k) \tag{1.5}$$

Note that:

$$\begin{aligned} r(n) &= \mathbb{E}\left\{x(n)x^*(n-m)\right\} \\ r(n-m) &= \mathbb{E}\left\{x(n-m)x^*(k)\right\} \end{aligned} \tag{1.6}$$

We can combine 1.4, 1.5 and 1.6 to get:

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} r(n-m)e^{-j\omega(n-m)} = \qquad \lim_{N\to\infty} \frac{1}{N} \sum_{k=-(N-1)}^{N-1} (N-|k|)r(k)e^{-j\omega k}$$

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

From this we see that the two definitions are equal under the condition of equation 1.3. From figure 1.1 we can see that for a non-infinite N we can end up with different PSD estimates. In this case the amplitude around 0 is different.

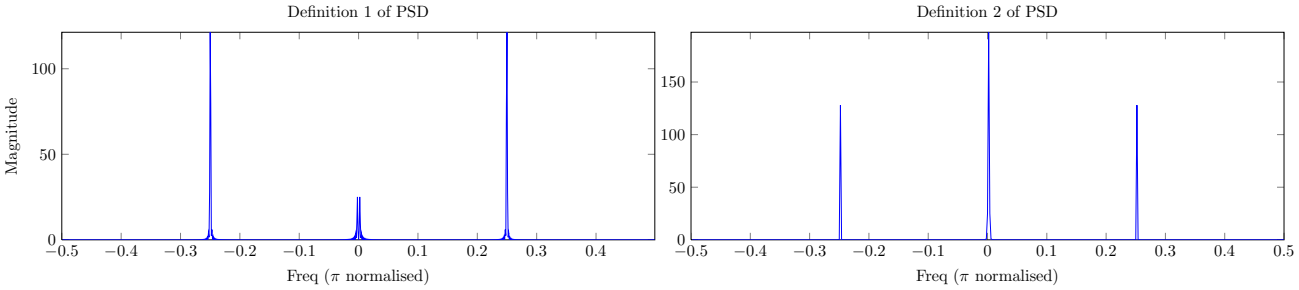


Figure 1.1: Plot of the PSD of `sin(f0*2*pi*n) + sin(f1*2*pi*n)` with `f0 = 0.25 f1 = 0.001` for 2 different definitions.

Using DTFT to determine PSD from ACF In this section we have the autocovariance sequence defined by:

$$r(k) = \begin{cases} \dfrac{M-|k|}{M}, & \text{if } |k| \leq M-1 \\ 0, & \text{otherwise.} \end{cases} \tag{1.7}$$

1.1.a Choice of x

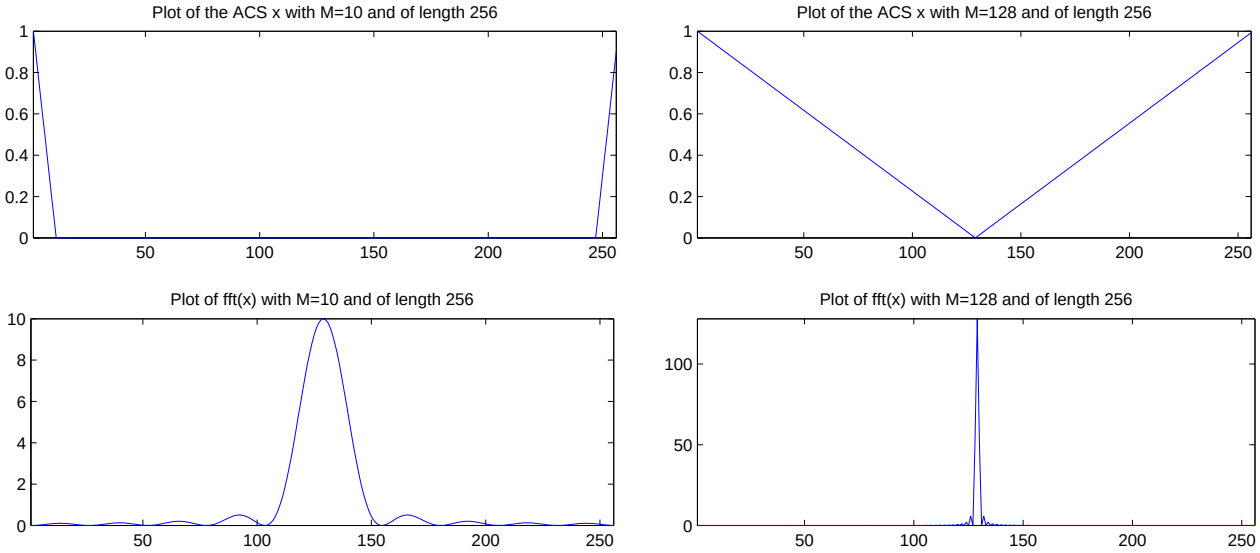


Figure 1.2: Plot of the vector $\mathbf{x}$, note that `fftshift` has been used for the fft plots to center them

We have $\mathbf{x} = [r(0), r(1), \ldots, r(M-1), 0, \ldots, 0, r(-M+1), \ldots, r(-1)]^T$ with zero padding in the middle. This firstly serves to have the length of $\mathbf{x}$ be a power of two as $256 = 2^8$, which creates a more computationally effective FFT, as well as increasing the resolution of the PSD[1]. Additionally the zero padding occurring in the middle which creates a circular sequence (that is on which wraps by the vectors length), meaning negative indices can in fact be accessed. i.e. negative lags$(-k)$ must be placed at $M-1+k$. This is due to the way matlab calculates its FFT, which is explained in section 1.1.c.

1.1.b Imaginary part due to round-off error

Removing the imaginary part via `real(xf)` introduces very little error as the imaginary part is negligible. This is demonstrated by `sum(real(xf)-abs(xf))` which returns `0`. `max(imag(xf))` and `min(imag(xf))` both return

[1] as seen in Slide 10 of Lecture 1

values in the order of `10e-14` which shows that these are negligible values. These values are only due to matlab computational error.

Due to the circular way the indices are accessed it is not expected that an imaginary part appears. If we imagine that the data is spread across a unit circle and that frequency is accessed in a normalised way (that is $0 \to 2\pi$) we can see imaginary parts will cancel out. This is the case for autocorrelation sequences if correctly generated as they will be symmetric, non-negative and real. This can be seen in simulations of figure 1.2 where only positive real values appear.

1.1.c Non-negligible Imaginary part

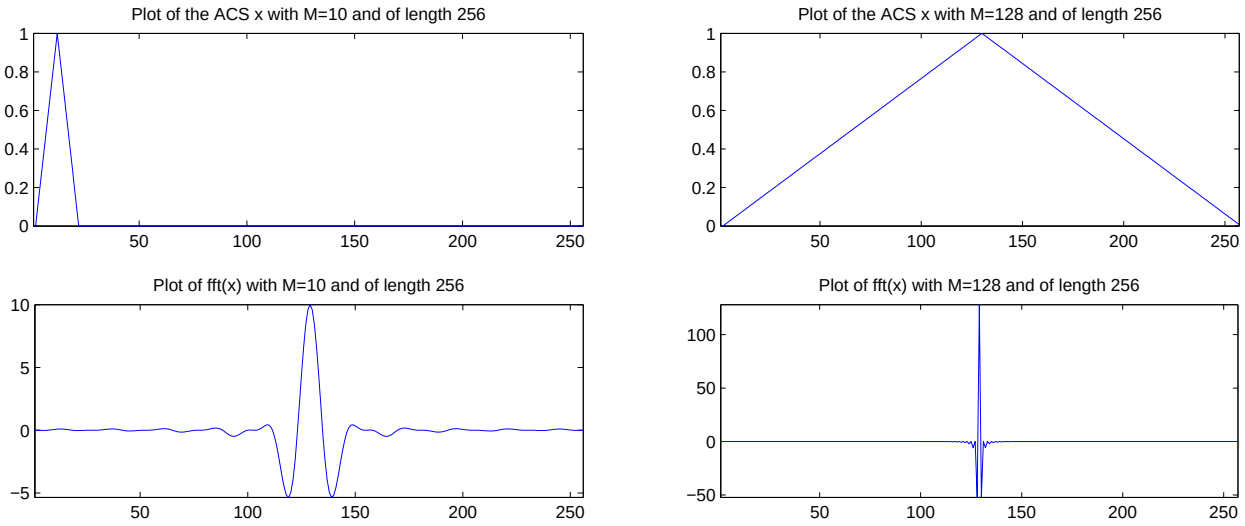


Figure 1.3: Plot of the vector $\mathbf{x}$ with padding at the end, again `fftshift` has been used to center the frequency

This time we have set up $\mathbf{x} = [r(-M-1), \ldots, r(-1), r(0), r(1), \ldots, r(M-1), 0, \ldots, 0]^T$. Thus a vector of length L is made with padding at the end (for `M=10`). We notice that not only the FFT is erronous (from simple comparison to the previous graph, Figure 1.2, as well as the fact that it is negative) but that it contains large, non-negligible, imaginary values. On the other hand for `M=128` the imaginary values are, again negligible, but the FFT is still slightly off.

The first issue of large imaginary values relies in the way matlab does its FFT. It assumes the array effectively has its negative index reflection in the second half of the array. Equation 1.8 shows the way matlab calculates its FFT.

$$X(k) = \sum_{n=1}^{N} x(n) e^{-i2\pi \frac{(n-1)(k-1)}{N}} \qquad \textit{Note:}\text{Matlab starts all its indexing at } \texttt{1} \tag{1.8}$$

To illustrate this, let us look at the matrix of values of $\frac{(n-1)(k-1)}{N}$ across n and k:

$$\exp(-i2\pi \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/8 & 2/8 & 3/8 & 4/8 & 5/8 & 6/8 & 7/8 \\ 0 & 2/8 & 4/8 & 6/8 & 8/8 & 10/8 & 12/8 & 14/8 \\ 0 & 3/8 & 6/8 & 9/8 & 12/8 & 15/8 & 18/8 & 21/8 \\ 0 & 4/8 & 8/8 & 12/8 & 16/8 & 20/8 & 24/8 & 28/8 \\ 0 & 5/8 & 10/8 & 15/8 & 20/8 & 25/8 & 30/8 & 35/8 \\ 0 & 6/8 & 12/8 & 18/8 & 24/8 & 30/8 & 36/8 & 42/8 \\ 0 & 7/8 & 14/8 & 21/8 & 28/8 & 35/8 & 42/8 & 49/8 \end{bmatrix}) \tag{1.9}$$

From this we can see that the values wrap around and are not correctly cancelled out. This is due to the sequence not being symmetric.

For the second issue of erroneous FFT, this is due to the ordering, which is in reverse, again due to the way the indices are perceived. This is because the function is increasing, i.e. $r(0) \geq |R(k)|, \forall k$ is violated, giving rise to real negative imaginary values.

1.1.d `fftshift` use

To get the correct **w** vector to annotate the FFT (when correctly `fftshift`ed) we use the following commands:

```
%L is the length
%even
w = 2*pi*((0:(L-1)) - L/2)/L;
%odd
w = 2*pi*(-((L-1)/2):((L-1)/2))/L;
```

These will placed the axis in the range of $[-\pi, \pi]$.

To get the correct **t** (time) vector to annotate the IFFT (when correctly `fftshift`ed) we use the following commands:

```
%even
t = -L/2:L/2-1;
%odd
t = -((L-1)/2):((L-1)/2);
```

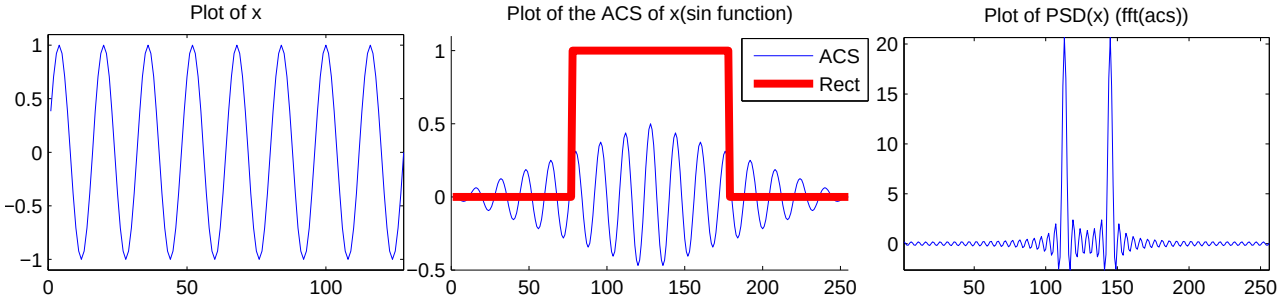
1.2 Properties of Window Functions in Spectral Estimation

1.2.a Negative Spectral Estimates

Negative spectral estimates are impossible if calculating the PSD from the second definition(1.2) due to the presence of the absolute value function. However the first definition, which involves taking the Fourier transform of the auto-covariance sequence, can lead to negative spectral estimates.

One way is to apply a window to a biased ACS:

While applying a window to the original signal will not lead to negative spectral estimates(due to the way a spectogram is calculated), applying a rect window to the ACS will. The typical PSD of a sin function is $\frac{A^2}{4}\delta(|f - f_0|)$, with A being the amplitude. The FFT of a rect function is $\frac{1}{\sqrt{2\pi}}\text{sinc}(\frac{\omega}{2})$, which contains negative values. Thus applying a rect window will make negative values appear when convolved with the delta function arising from the sine wave.



Note: ACS and PSD were appropriately `fftshift`ed for processing purposes.

Figure 1.4: Plot of the vector **x**, with its ACS and PSD functions

The unbiased ACS ($r(k) = \frac{1}{N}\sum_{n=0}^{N-1-k} x(n+k)x(n)$) is also liable to creating negative estimates, as it does not guaranteed the creation of a positive definite sequence. If the ACS is not positive definite this corresponds to a non-positive semi-definite autocorrelation matrix, which will not be able to guarantee a purely positive PSD.

Thus there are two, related, ways of creating negative spectral estimates.

1.2.b Bartlett Window

The Bochner theorem states that any positive-definite function in R^n is the Fourier transform of a positive measure g on the real line with $g(y) \geq 0$, $\forall y$.[1]

$$w_B(k) = \begin{cases} 1 - \dfrac{|k|}{N}, & \text{if } |k| \leq N \\ 0, & \text{otherwise.} \end{cases} \tag{1.10}$$

Thus to prove that the Bartlett window (1.10) is positive we must prove its Fourier transform is positive for all values.

$$w_B(k) = \Pi(k) \star \Pi(k) \text{where } \Pi \text{ is the rect function of width } N \text{ and height } 1$$
$$\mathfrak{F}(w_B(k)) = \mathfrak{F}(\Pi(k)) \times \mathfrak{F}(\Pi(k)) = \frac{1}{\sqrt{2\pi}} \operatorname{sinc}(\frac{\omega}{2}) \times \frac{1}{\sqrt{2\pi}} \operatorname{sinc}(\frac{\omega}{2}) = \frac{1}{2\pi} \operatorname{sinc}^2(\frac{\omega}{2})$$

As can be seen above $\frac{1}{2\pi} \operatorname{sinc}^2(\frac{\omega}{2})$ is both real and always positive(due to the square), which allows to conclude that the Bartlett window is positive definite.

As seen previously it is desirable to have a positive definite window which leasts distorts the out. Thus we want a window which distorts the ACF the least. To show the Bartlett window satisfies this we minimise the following criterion:

$$\min_{\{w(k)\}} \sum_{k=0}^{M-1} |1 - w(k)|$$

We can replace $w(k)$ by $\sum_{j=0}^{M-1} s_j s_{j+k}$ as a positive definite window can be replaced by a convolution of an arbitrary sequence.

We define $w(0) = 1$ as this is a window.

As $w(0) \geq w(k) \forall k$ we can rewrite as:

$$\sum_{k=0}^{M-1} 1 - \left(\sum_{j=0}^{M-1} s_j s_{j+k} \right)$$

We can rewrite the convolution as a vector sum $w(k) = s^T T_k s$ where $T_k = \begin{bmatrix} 0_{k\times(M-k)} & 0_{k\times k} \\ I_{(M-k)\times(M-k)} & 0_{(M-k)\times k} \end{bmatrix}$. While this is practical we will slightly tweak it to get a symmetric matrix which is nicer for our proof. We get $w(k) = \frac{1}{2} s^T (T_k + T_k^T) s$. Let $T_k + T_k^T = P_k$. From this:

$$\sum_{k=0}^{M-1} 1 - \frac{1}{2} s^T P_k s = M - \frac{1}{2} s^T \sum_{k=0}^{M-1} (P_k)\, s$$

To minimise this function is simply to maximise $s^T \sum_{k=0}^{M-1} (P_k)\ s$. Note that:

$$\sum_{k=0}^{M-1} (P_k) = \begin{bmatrix} 2 & 1 & 1 \\ 1 & \ddots & 1 \\ 1 & 1 & 2 \end{bmatrix} = P_{sum}$$

P_{sum} has, ordered by magnitude, eigenvalues of $\big[(M+1), 1, \ldots, 1\big]$. As we know that the maximising s is given by the eigenvector associated with the largest eigenvalue[2] we get the maximising s:

$$s_{max} = [1, \ldots, 1] / \sqrt{M}$$

This vector corresponds to the triangular Bartlett window (because we defined $w(k)$ as the convolution of s on itself i.e. $rect \star rect = w_B$), which concludes our proof.

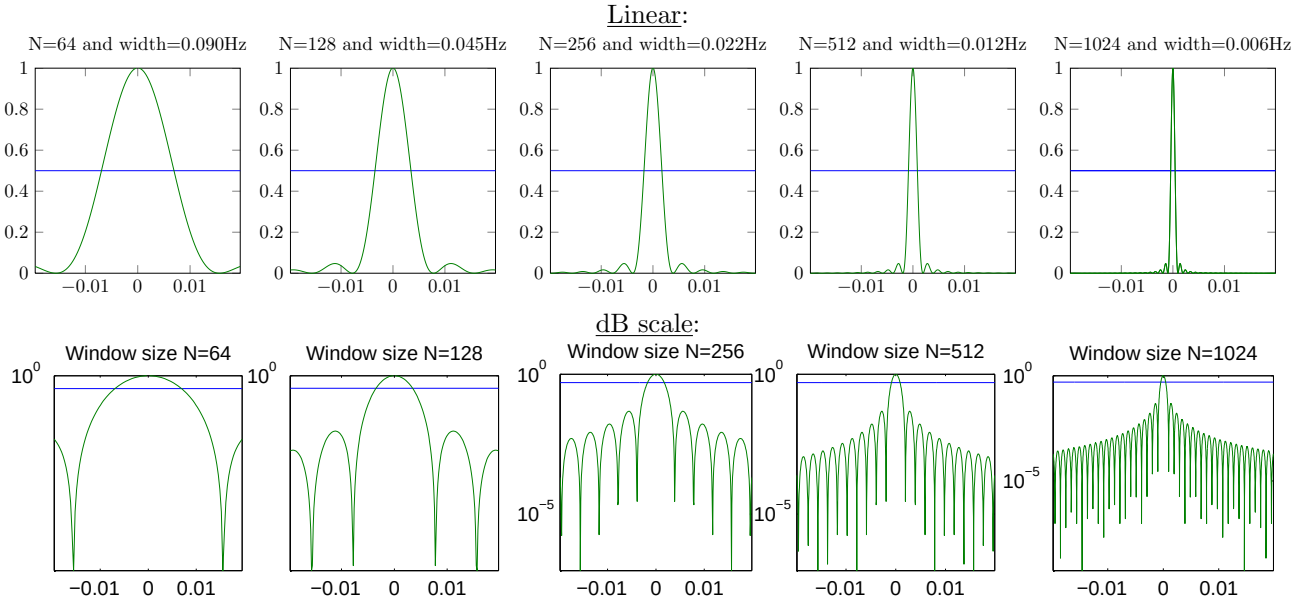
1.3 Resolution and Leakage of Periodogram-based Methods

1.3.a Bartlett window lobes

From figure 1.3.a we can see that the 3db width is just under $0.9 \cdot 2\pi/N$. From observation the side lobe height is at $-6.54dB$ which is independent of N (this is expected from the $W_B(\omega)$ function).

1.3.b α value

For the function $\mathbf{x} = \sin(0.2 \times 2\pi n) + \sin((0.2 + \alpha/N) \times 2\pi n)$ we investigate the resolution of the periodogram for $N = 256$, that is the increase in α needed to distinguish the two sine waves. From this we find that an $\alpha \geq 0.7$ allows the distinction between two frequencies. This is consistent with the FFT covering $0 \to 2\pi$ and being f length N, implying a resolution of $0.7 \cdot 2\pi/N$.



Note: The blue line denotes the 3db line and the green line is $W_B(\omega)$. Additionally the window has been normalised to 1 for practicality. Frequency axis is in Hz.

Figure 1.5: Plot of the Bartlett window and the 3db width of the main lobe

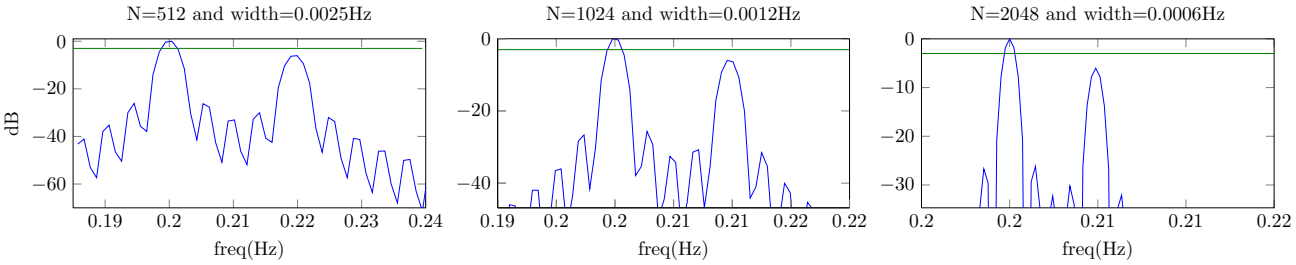


Figure 1.6: Plot of the lobe width for a Bartlett windowed signal with $\alpha = 10$

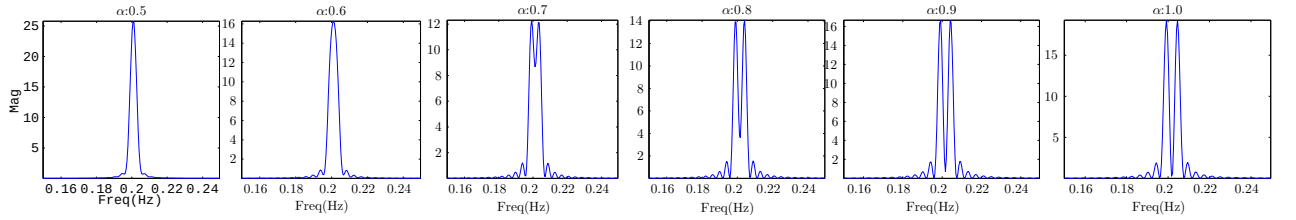


Figure 1.7: Plot of multiple $\mathbf{x}$ for different α values

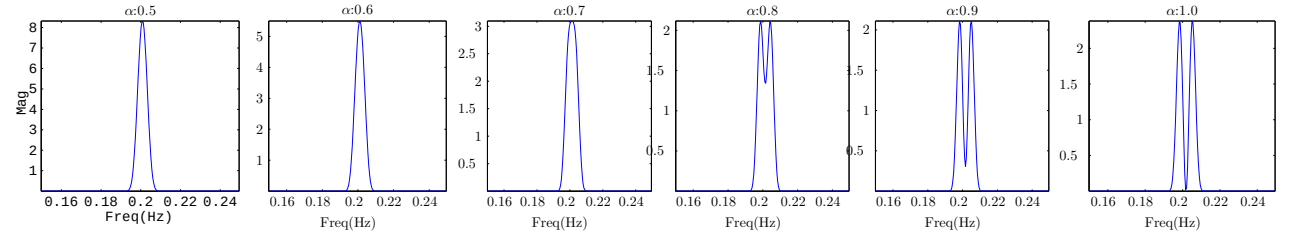


Figure 1.8: Plot of multiple $\mathbf{x}$ for different α values

1.3.c α value with Hamming windows

From Figure 1.8 we see that a value of 0.8 instead of 0.7 works when data is windowed through a Hamming window. This is due to the Hamming window's lobe causing more smearing and a loss in spectral resolution.

1.3.d Amplitude Identification

We define: $\mathbf{x} = a_1 \sin(0.2 \times 2\pi n) + a_2 \sin\left(\left(0.2 + \frac{\alpha}{N}\right) \times 2\pi n\right)$

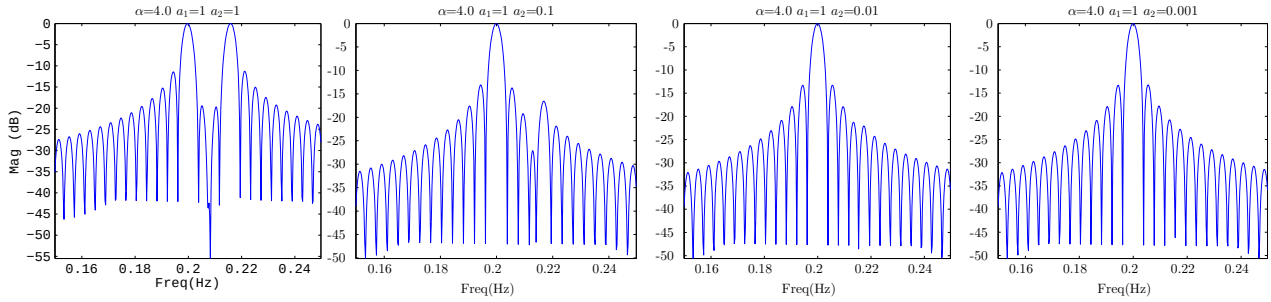


Figure 1.9: Plot of multiple $\mathbf{x}$ for different a_2 values. Decreasing a_2 leads to a smaller magnitude for the second sinewave.

Figure 1.9 shows us that the amplitude is no longer identifiable when $a_2 \leq 0.01$. i.e. the magnitude difference is now more than two magnitudes in difference with the largest signal.

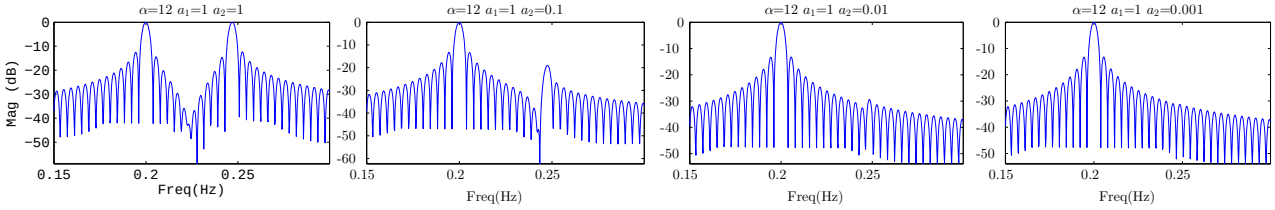


Figure 1.10: Plot of multiple $\mathbf{x}$ for different a_2 values

In Figure 1.10 we now have $\alpha = 12$, increasing the frequency distance between the two signals. This leads to see that increased the distance between the frequency of the two sine waves increases the identification threshold to $a = 0.01$.

These findings highlight the effects of different amplitudes and frequencies in signals causing interference.

1.3.e Effect of Bartlett window

We know that $\mathbb{E}\left\{\hat{P}_{per} = W_B(f) * P_{xx}(f)\right\}$ i.e.the expected value of the periodogram is the convolution of the power spectrum with the Fourier transform of the Bartlett window.

From the shape of the Bartlett window in frequency we can see that a too big difference in frequencies as well as magnitudes can cause issues in the quality of the periodogram.

Looking at 1.11 we see that the height of the sidelobes at $f = \frac{4}{N}$ and $\frac{12}{N}$ (and for another integer by N) are effectively troughs, at about $-100dB$. From this we can see why the lower amplitude sinusoid would be obscured. Further on the amplitude has been reduced enough such that $12/N$ offset is not present when the sinc^2 convolutes over the first frequency, enough for it to effectively act like a delta function.

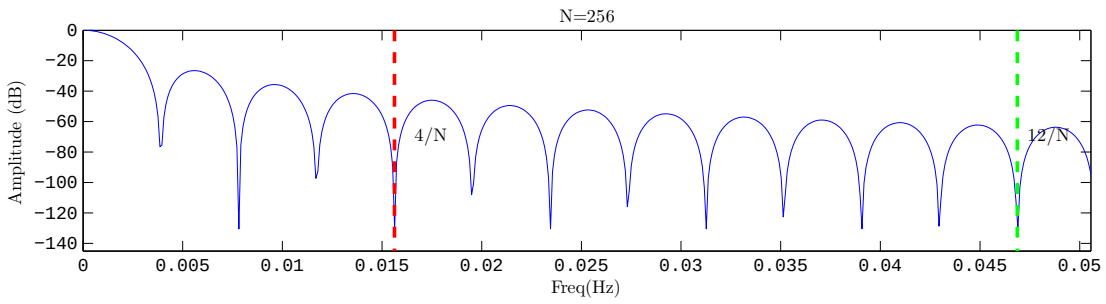


Figure 1.11: Plot of Bartlett window 0 to $0.05Hz$

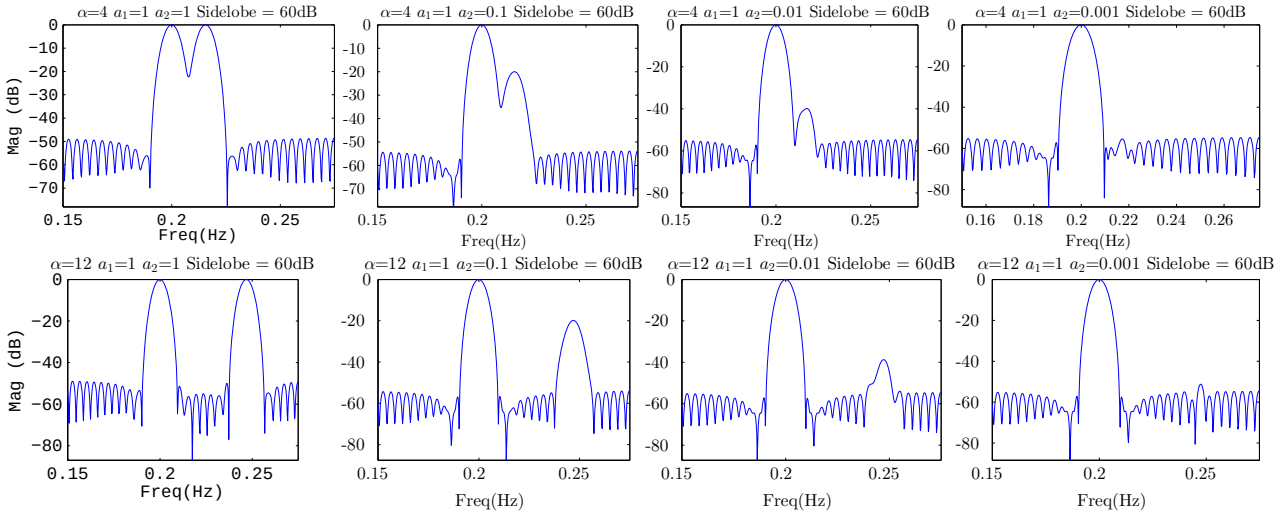


Figure 1.12: Plot of Chebyshev windowed data

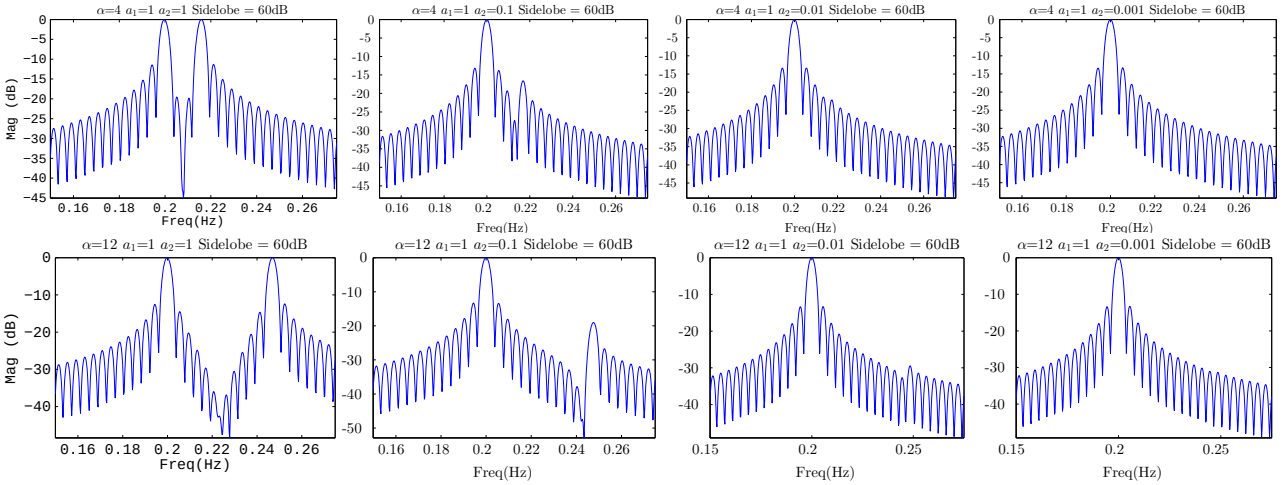


Figure 1.13: Plot of Blackman-Tukey method with a Chebyshev window data

1.3.f Chebyshev and Blackman-Tukey windowing

Here we have windowed the sample data in figure 1.12 and in figure 1.13 the autocovariance function has been windowed (Blackman-Tukey method). In both the Chebyshev window was used. This is done using the `chebwin(L,r)` function in matlab where `L` is the window length and `r` is the depth in dB of the sidelobes, which are constant. Figure 1.12 shows us that the combination of the Chebyshev's window width and second lobe amplitude allows it to detect the weaker sinusoid up until $a_2 = 0.001$. For this case a sidelobe of 60dB was found to work best. This is because the Chebyshev window can sacrifice the width of its central lobe for constant low amplitude sidelobes, leading to less spectral leakage.

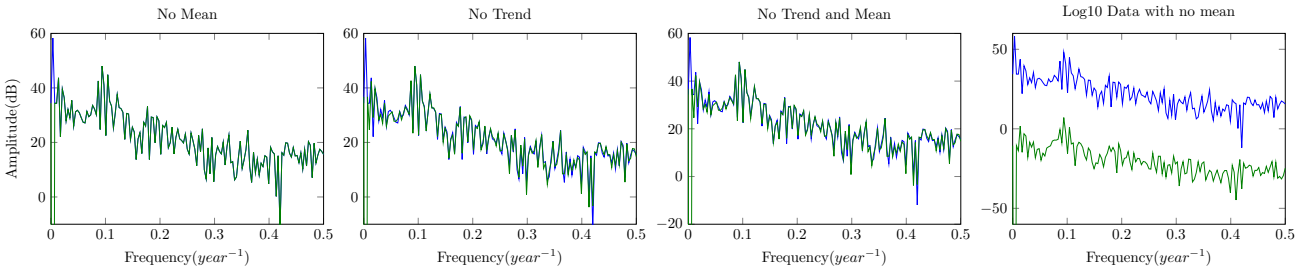
The Blackman Tukey method aims to reduce the variance of the estimator. While giving us a more precise spectrum, it is not as accurate due to trying to reduce the contribution of unreliable estimates to the periodogram. So despite the low amplitude sidelobes (the Bartlett window sidelobes are not as low) the weak amplitude sinusoid is not able to be detected.

1.4 Periodogram-based Methods Applied to Measured Data

1.4.a Sunspot Data Investigation

From figure 1.14 we view little difference to applying pre-processing methods. However removing the mean and the trend does make the amplitude at $0year^{-1}$ be, 0. Logging the data causes the PSD to be slightly "spikier". In all cases, the trends of the data can be seen at about $0.1year^{-1}$ and $0.013year^{-1}$, which fits with the trends we see for the unprocessed data.

The removal of the mean essentially causes the amplitude at $0Hz$ to be 0. This is because it causes the sum



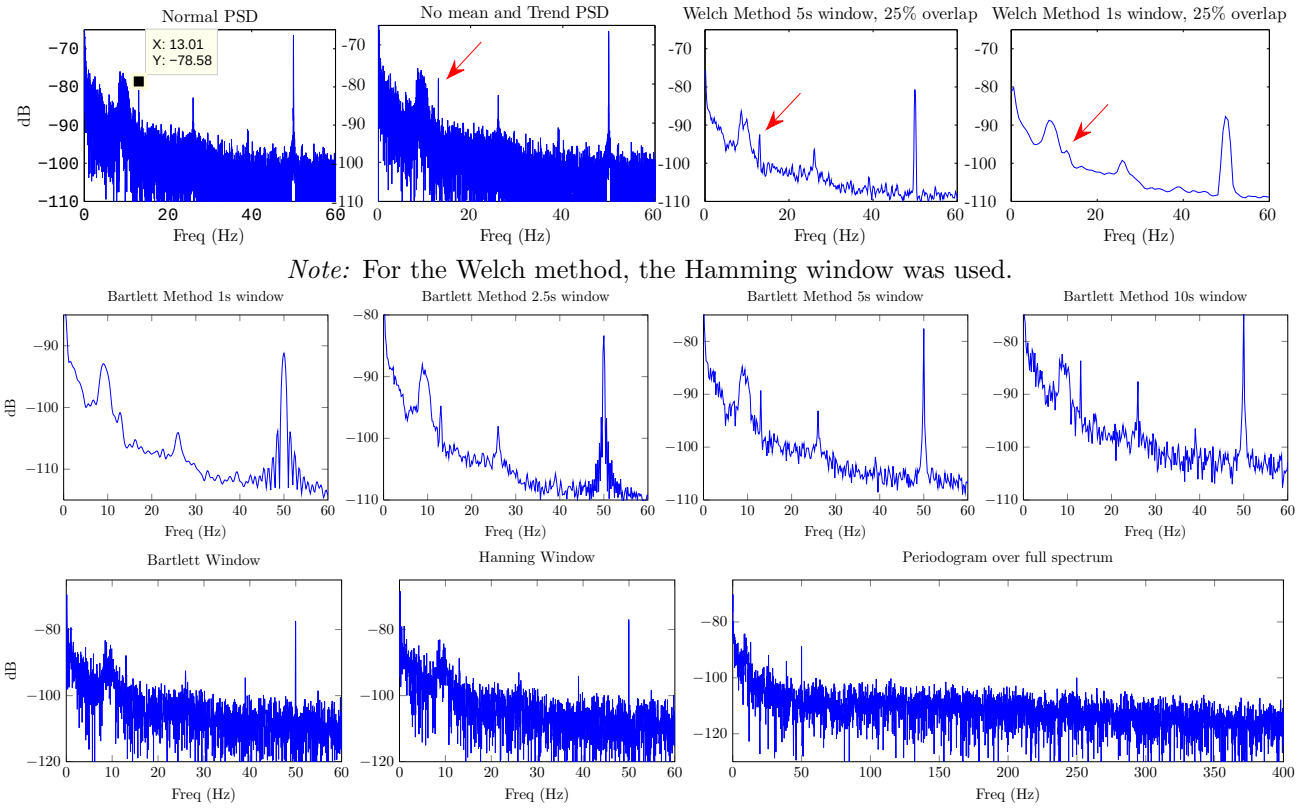
Note: The blue line denotes the non preprocessed PSD and the green one is the pre-processed line.

Figure 1.14: Plot PSD estimates for various pre-processing methods

of all values to equal zero. During the calculation of the exponent for $0Hz$ with $P(\omega) = \frac{1}{N}\left|\sum_{n=0}^{N-1} x(n)e^{-jn\omega}\right|^2$ leads to $e^{-jn\omega} = 0$ being 1, essentially summing up all values which are now zero. This is advantageous as it removes the zero bias making it easier to see spectral components near to $0Hz$.

Taking the log of the data causes the PSD estimate to be slightly more "spikier", allowing a greater distinction of amplitudes at lower frequencies.

1.4.b Investigation on EEG data



Note: For the Welch method, the Hamming window was used.

Figure 1.15: Periodograms of EEG data, where the unmodified PSD is used, with other methods

From figure 1.15 we attempt to look for the SSVEP peak (and its harmonics). The first observation is that a usual PSD is very noisy and identifying a single peak, while possible, is potentially just noise. Next we tried windowing but this had little useful effect - mainly as leakage is not a concern for this dataset. The Welch and Bartlett methods were then tried out. For both it was found that a small time window (such as 1 second) produced PSDs of too low quality. However past a time window of 2 seconds the 13Hz peak become easily viewable. The Welch method with a Hamming window was found to be slightly worse than the Bartlett window, for the same window lengths, due to the more prominent peaks, which the Welch method slightly removes. The lower peaks are due to the windowing the Wlech method applies, rather than the window overlapping.

The use of windowing functions must be carefully weighted against their disadvantages as, due to their wider lobes, the spectral resolution is lower. Additionally time windowing and taking an average (Bartlett and Welch

method) can also lead to decreased quality of signal due to the shorter time segments. This means larger frequency ranges are contained in each bin, decreasing the resolution.

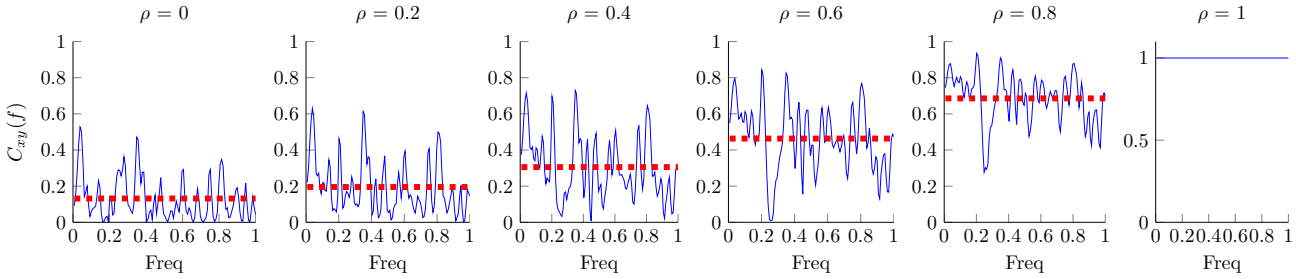
Now that we now that the SSVEP peak is at $13Hz$ we can see from figure 1.15 that the harmonics are present at 26, 39 and $52Hz$, quickly decaying in amplitude.

Part 2

Parametric and Line Spectra

2.1 Coherency Spectrum

2.1.a Bivariate correlated white Gaussian signal



Note: The red line represents the average coherency across the whole spectrum.

Figure 2.1: Plot of the coherence between two signals x and y for different ρ

The coherency spectrum is calculated by:

$$C_{xy} = \frac{P_{xy}^2(\omega)}{P_x(\omega)P_y(\omega)} \tag{2.1}$$

Its values vary between 0 and 1 and are useful in providing information on how much two signals tend to agree for various frequencies, where 1 would imply a perfect agreement between both signals.

Here we have two signals $x(n) = x_1(n)$ and $y(n) = \rho x_1(n) + \sqrt{1-\rho^2}x_2(n)$ where $x_1, x_2 \sim \mathcal{N}(0,1)$. We notice that the more ρ is increased, the more each value of the coherency spectrum is expected to be near 1, being fully 1 for two identical signals. A low ρ (under 0.2) will not cause much change, as seen by the similar averages.

2.1.b Noise and coherence

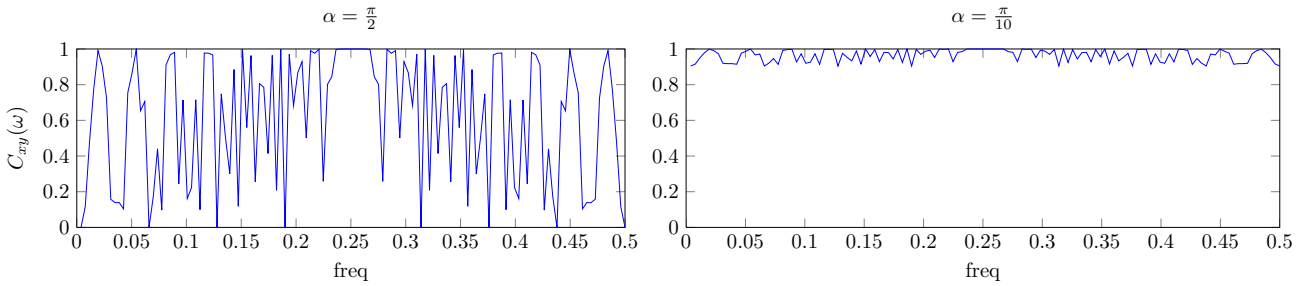
Note: In all situations an average time windowing was used, as it was found to produce workable results.

In this case:

$x = \sin(2\pi f_0 n) + w_1$ and $y = sin(2\pi f_0 n + \alpha) + w_2$

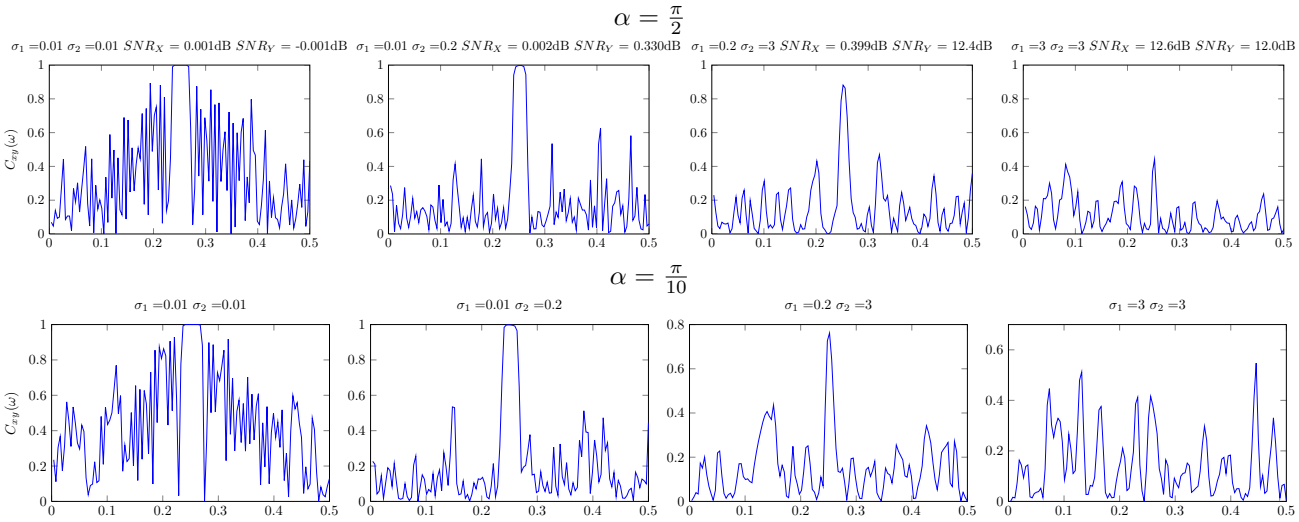
In figure 2.2 we see that, as expected, having $\alpha = \frac{\pi}{10}$ leads to $C_{xy}(\omega)$ being near 1. This is due to the relation between the two signals. A shift of $\frac{\alpha}{2}$ changes the spectral contributors - this shift is still present in $r_{xx}(k)$ and as P_{xx} if the Fourier transform of r_{xx} this change is still reflected. Remember that $\sin(x + \frac{\pi}{2}) = \cos(x)$. However for $\frac{\pi}{10}$, this does not hold and all spectral components are essentially the same. The seemingly random samples not near to 0.1 freq for $\pi/2$ are due to roundoff errors on matlab's part.

In figure 2.3 we see that the coherence is similar for $\alpha = \frac{\pi}{2}, \frac{\pi}{10}$, meaning that the phase shift is no longer relevant. Additionally adding noise actually increases the quality of the coherence spectrum with noise of standard deviation $\sigma \approx \{0.01, \quad 0.2\}$. With values around this range the peak is clearly detected. For excessive amounts of noise, such as $\sigma = 3$ very little information is present as the original signal is too corrupted by the noise. In effect an SNR higher than 10dB seems to be enough to start decreasing the coherence quality. This is consistent with a high SNR signifying a loss of information in the signal.



Note: In both cases there is no noise i.e. $\sigma_1, \sigma_2 = 0$. $f_0 = 0.25$

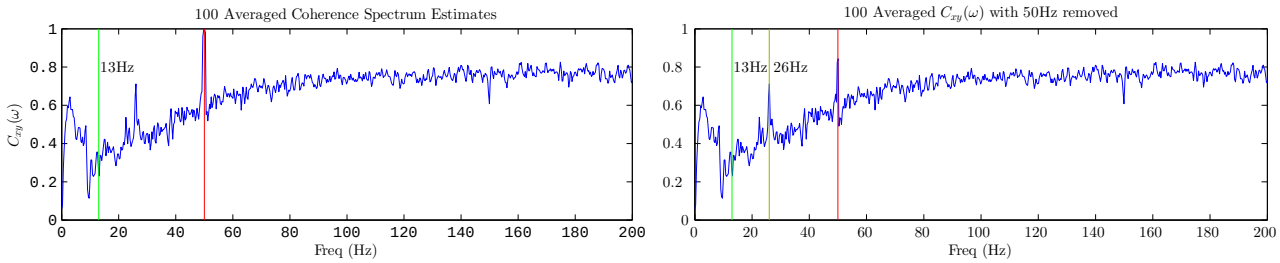
Figure 2.2: Plot of the coherence between two signals x and y for different α



Note: $f_0 = 0.25$

Figure 2.3: Plot of the coherence between two signals x and y for different α

2.1.c EEG Coherency Spectrum



Note: The green line denotes the $13Hz$ line, the dark green is $26Hz$ and the red line denotes $50Hz$ (power-line interference).

Figure 2.4: Plot of the coherence between two-channel EEG data

This time a two channel data is taken and we notice that in the coherency spectrum the expected $13Hz$ signal is present. Upon further investigation and with the help of figure 2.5 it was noticed that the `Cz` channel did not have any clear peaks at $13Hz$. However the first harmonic of the $13Hz$ is noticeable at $26Hz$.In figure 2.4 this is highlighted by the dark green line on the right graph (with the $50Hz$ power-line interference removed). While the 39 and $52Hz$ harmonics would be expected their amplitude has decayed too much to be noticeable.

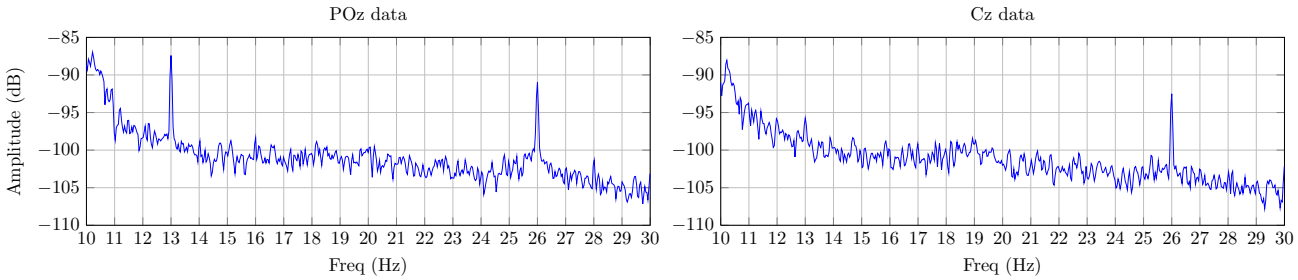


Figure 2.5: Plot of the left and right PSD of the EEG data. This uses the Welch method with a Hamming time window of 20 seconds and a 25 % overlap.

2.2 Correlation Estimation

2.2.a Biased and unbiased effects on spectral estimation

Note: The graphs in dB may not be entirely accurate due to the negative values created by unbiased ACS estimates.

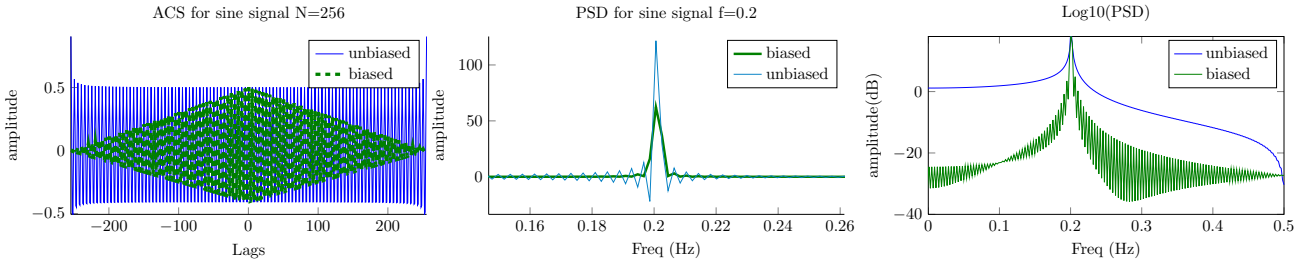


Figure 2.6: Plot of the ACS and PSD for a sine wave.

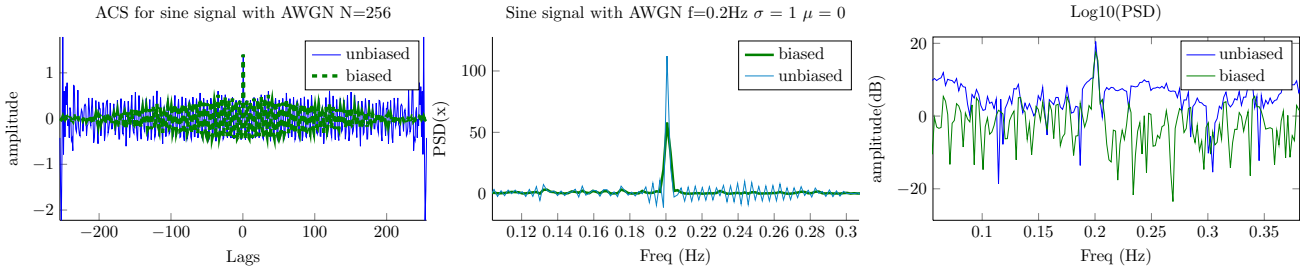


Figure 2.7: Plot of ACS and PSD for sinewave with AWGN

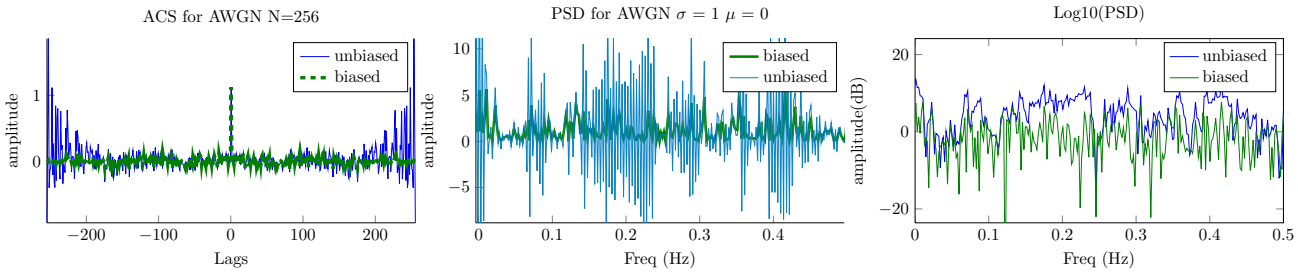


Figure 2.8: White Gaussian noise

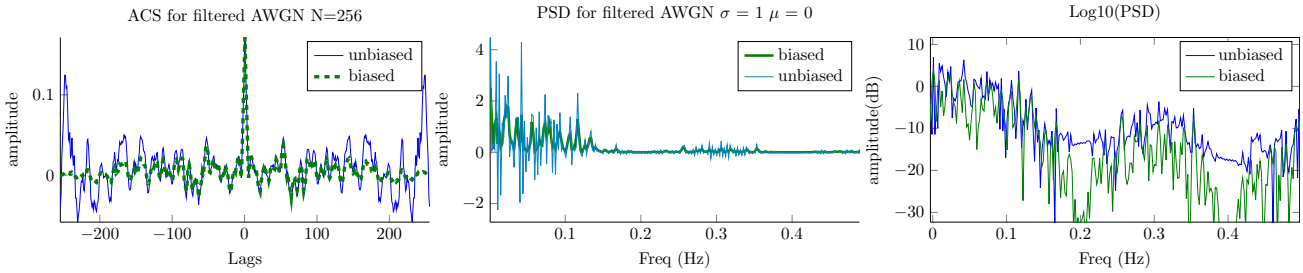
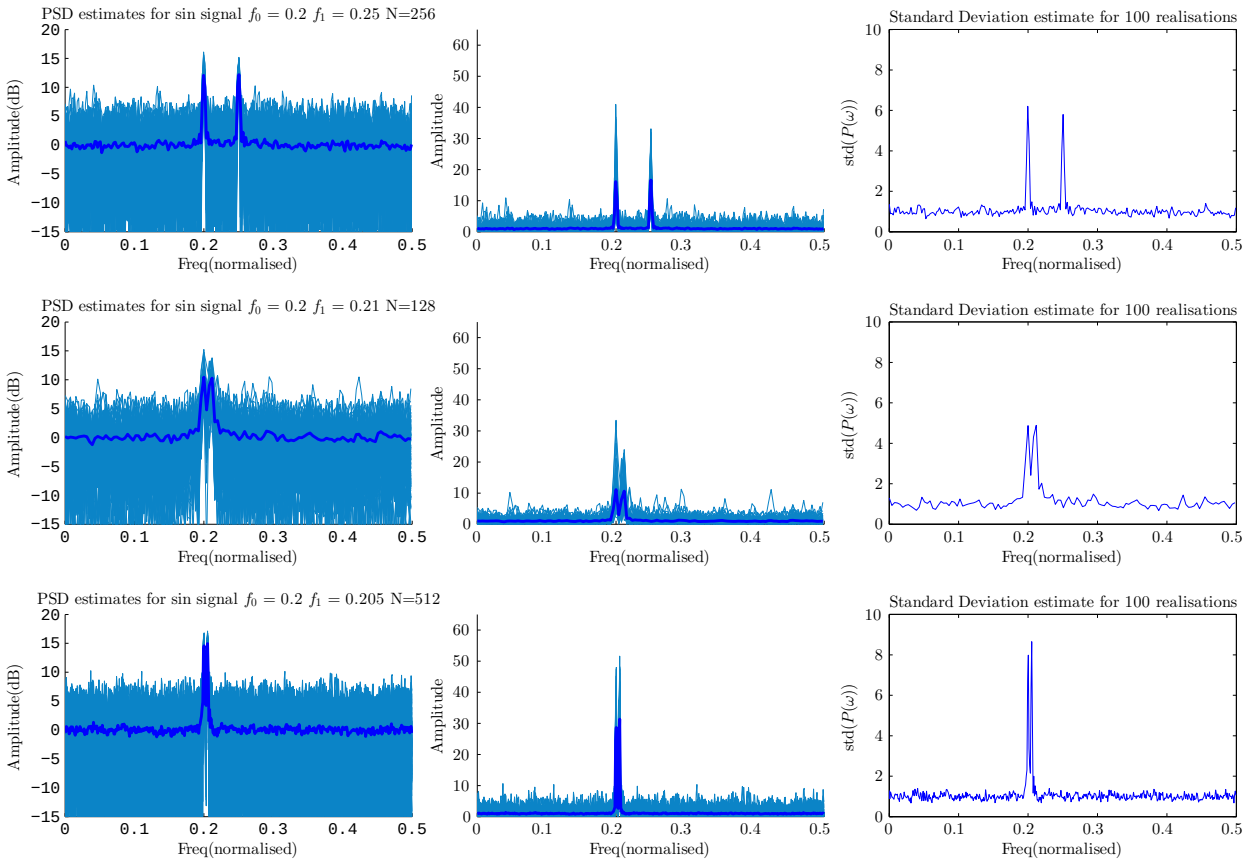


Figure 2.9: Low-pass filtered White Gaussian Noise

From figure 2.6 - 2.9 we see the differences in spectral estimates from biased and unbiased ACS. For the ACS the first difference is the final value as $|k| \to N$, where k is the lag. For a biased ACS we have $\lim_{|k|\to N} r(k) \approx 0$, whereas for an unbiased ACS the estimates become inaccurate or divergent as $|k| \to N$ due to the $\frac{1}{N-k}$ factor being used instead of $\frac{1}{N}$ that is used in the biased one. This leads to an over representation of small segments of data which may not be representative overall, whereas a biased estimate removes these.

The PSD is the result of taking the Fourier Transform of the ACS, i.e. $P(\omega) = \mathscr{F}\{r(k)\}$. The first noted difference is the negative spectral estimates resulting from the unbiased ACS. This is due to it not being a semi-definite positive sequence which, as we have seen in section 1.2.a, can lead to negative values. The second effect on the PSD is that in general $|P_{biased}| \leq |P_{biased}|$ (This was tested in matlab and found to be true for about 75% of samples). This is due to the unbiased ACS that the FFT is being taken of to not penalise unnecessarily high values. This leads to "spikier" estimates, caused by the higher data variance.

2.2.b Different random realisations



The blue line represents the mean.

Figure 2.10: PSD estimate and deviation for different realisations with AWGN of $\sigma = 1$ $\mu = 0$

In figure 2.10 100 different realisations of a 2 sine signals with Gaussian noise were done. From this we see that at the frequency peaks the standard deviation is greatly increased. For the tested data points we find that

the peaks of the variance are roughly equal to $\frac{\sqrt{N}}{2.7}$, that is the variance increases for greater sample lengths. However the standard deviation for frequencies outside of the peaks is 1 and is not related to N. Essentially the standard deviation is also influenced by the PSD value, which is problematic for spectral estimation with only few realisations.

2.2.c Plotting in decibel

In 2.10 the left most graph for each case was plotted in decibels and figure 2.11 is the dB plot of the standard deviation. Here we observe that the main advantage this has is to compare the full frequency spectrum with a smaller range, due to the contraction property of the log function. This is practical for the high variance that arises for large PSD values as they are essentially negated. For the standard deviation estimation taking the log doesn't achieve much except making the resolution for lower std estimates slightly better - which we are not interested in.

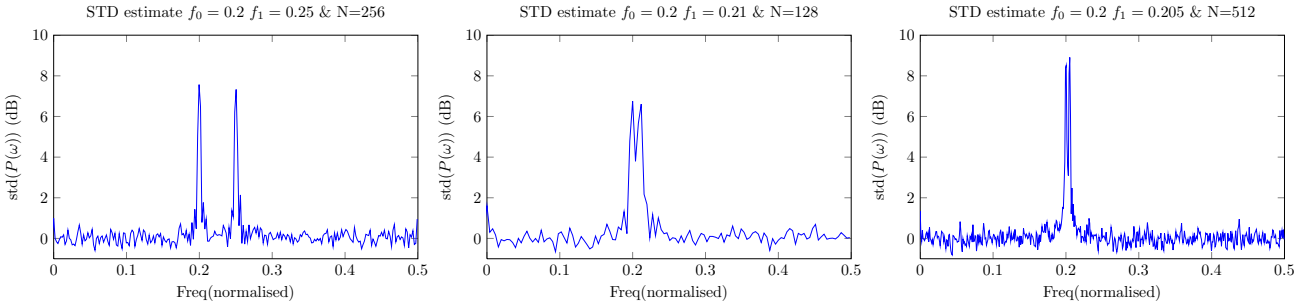


Figure 2.11: STD estimates in decibel

2.2.d Complex signals

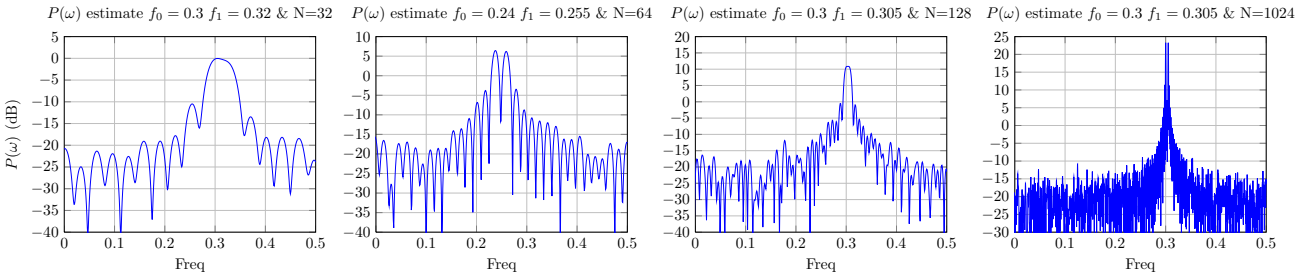


Figure 2.12: PSD estimates of complex signals

In figure 2.12 generation of complex signals was experimented. This was achieved with the following code:

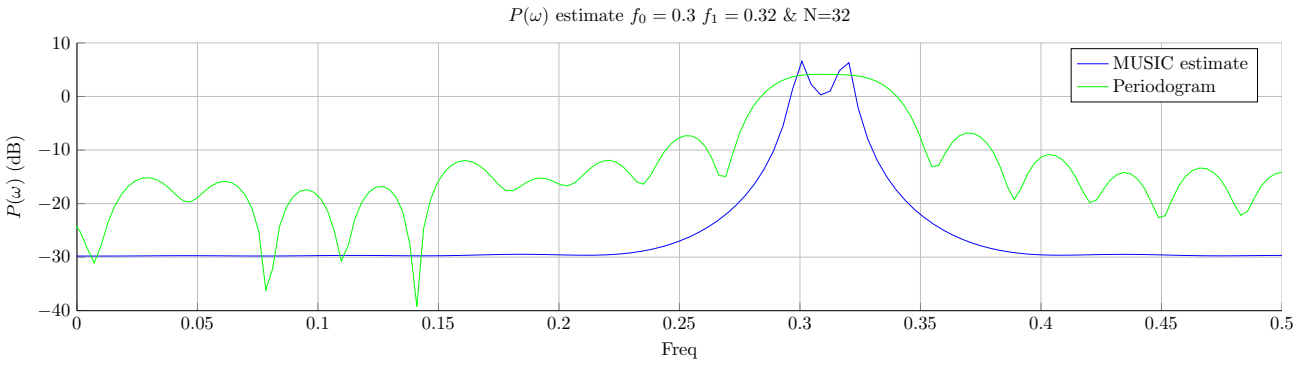
```
%% Complex Signal Generation
%define values
N = 32;
P = 128*8; % amount of extra padding wanted
f0 = 0.3;
f1 = 0.32;
n = 0:(N-1);
%generate sequence
x = exp(1j*2*pi*f0*n) + exp(1j*2*pi*f1*n) + 0.2/(sqrt(2))*(randn(1,N)+1j*randn(1,N));
x = [x zeros(1,P)]; %pad
```

Here we observe again that increasing the number of samples, from 128 to 1024 in this case, increases the resolution and allows the distinction of two, very close, frequencies.

Now since we have a complex signal, we have a complex autocorrelation, meaning that we are not guaranteed a symmetric PSD.

2.2.e MUSIC algorithm

We notice that in figure 2.12 for $f_0 = 0.3$ and $f_1 = 0.32$ with $N = 32$ the two expected spectral peaks are not present but instead merged into one. We thus try out the MUSIC algorithm to give us better estimates. Figure



Note: PSDs have been scaled for comparison purposes

Figure 2.13: PSD estimates of complex signal using MUSIC and periodogram approach

2.13 shows how the MUSIC algorithm performs better by separating out the two peaks. Additionally the lack of peaks makes for a nicely smoothed spectrum. The code used is as follows.

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

The MUltiple SIgnal Classication method works by assuming a signal is structured as such:

$$x(n) = \sum_{i=1}^{p} A_i e^{jn\omega_i} + w(n), \qquad w(n) \sim \mathcal{N}(0, \sigma_w^2) \tag{2.2}$$

That is we split the signal into multiple space and assume the signal is part of p subspaces. These correspond to the p largest eigenvalues of the $\mathbf{R}_{xx}$ matrix (the autocorrelation matrix). Thus for an $\mathbf{R}_{xx}$ of size M we would assumed that the remaining eigenvalues are related to the noise. We can represent the eigenvalue and eigenvector decomposition of the matrix into signal and noise as:

$$\mathbf{R}_{xx} = \underbrace{\sum_{i=1}^{p} \lambda_i v_i v_i^H}_{\text{signal}} + \underbrace{\sum_{i=p+1}^{M} \lambda_i v_i v_i^H}_{\text{noise}} \tag{2.3}$$

The first line `[X,R] = corrmtx(x,14,'mod');` serves to make $\mathbf{X} \in \mathbb{R}^{2(n-m)\times(m+1)}$, where $m = 14$ (the second input argument) and $n = 32$, the length of the input vector x. $\mathbf{X}$ is a Toepliz matrix that allows us to calculate a $(m+1) \times (m+1)$ autocorrelation matrix. The `'mod'` ensures the size of the matrix $\mathbf{X}$. After this we have an autocorrelation matrix $\mathbf{R}_{xx} = \mathbf{X}^T\mathbf{X}$.

The second line `[S,F] = pmusic(R,2,[ ],1,'corr');` simply returns the estimated PSD in $\mathbf{S}$ with $\mathbf{F}$ containing the frequency array. The first argument is simply the autocorrelation matrix, generated previously. After this with the second argument we set $p = 2$, as we know our signal has 2 peaks we can use this information to our advantage. The two subsequent arguments simply keep the default FFT length as well as set the frequency sampling to 1. `'corr'` tells the function the input argument is a correlation matrix and to treat it as such. After eigenvector and eigenvalue analysis

The final line `plot(F,S,'linewidth',2); set(gca,'xlim',[0.25 0.40])` simply serves as plotting the spectrum with a practical x limit (within the range of the expected peaks at 0.3 and 0.32).

The advantages of the MUSIC algorithm to a traditional PSD are the better resolution, the ability to search for a few dominant peaks which can nicely eliminate noise and the clear PSD estimate output that can be easier to analyse. The disadvantages are the assumption that the signal is complex and the fact that it is parametric - prior signal knowledge is desirable to correctly apply it.

Overall a spectrum generated using MUSIC would only be a faithful representation if enough knowledge is know about the signal - which would be the case in digital communications. However for more complex uses such as voice analysis the MUSIC method seems to less appropriate for analysis.

For different realisations the signal represented smaller variance for MUSIC estimate.

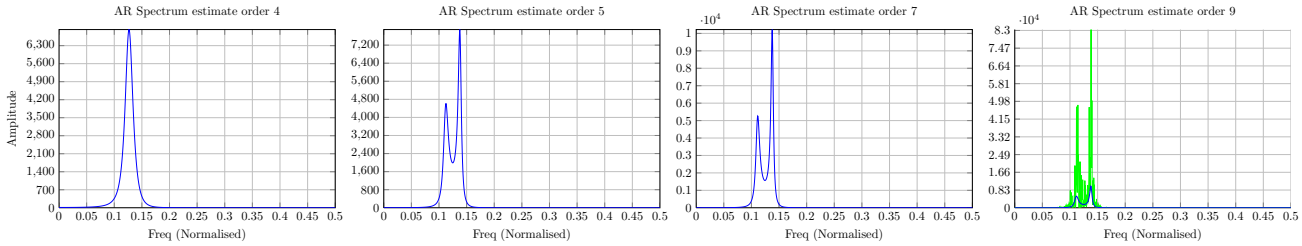
2.3 Spectrum of autoregressive processes

2.3.a AR process estimation via unbiased ACF

Using an unbiased ACF estimate $\mathbf{R}_{xx}$ is not guaranteed to be positive definite, as seen in section 1. This means it can be singular and thus non-invertible. This is needed for the AR process coefficients as $\mathbf{a} = \mathbf{R}_{xx}^{-1}\mathbf{R}_{xx}$. Additionally in cases where it is invertible we would be dealing with estimates of high variance - as has been seen in 2.2.a.

This is due to the estimates at high lags being inaccurate (when $k \to N$) due to little values being able to represent it fully.

2.3.b Order of AR process spectrum



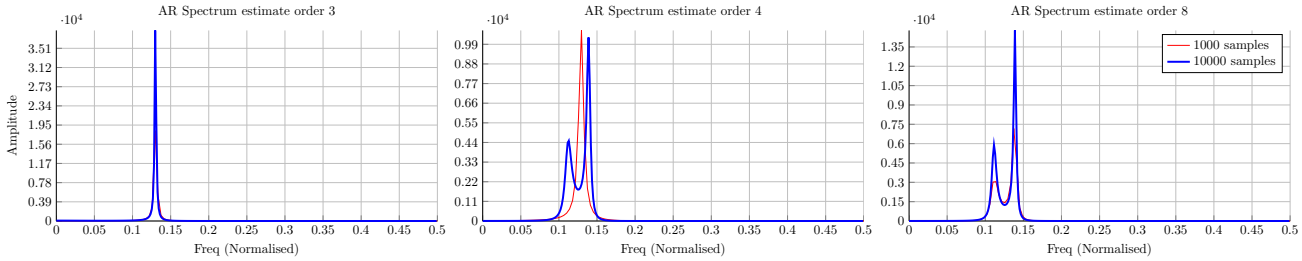
The blue line represents a AR spectrogram estimate and the green line represents a traditional periodogram.

Figure 2.14: PSD estimates from AR coefficients

Here in figure 2.14 we have created an AR process from $x(n) = 2.76x(n-1) - 3.81x(n-2) + 2.65x(n-3) - 0.92x(n-4) + w(n)$ with $w(n) \sim \mathcal{N}(0, \sigma_w^2)$. From this we have used Yule-Walker equations to generate AR coefficients estimates to be used to estimate a spectrum for different assumed orders. The figure uses orders 4,5,7 and 9. We can see that for any order above 5 for this AR4 process the two frequencies becoming increasingly distinguished. Additionally this has the effect of smoothing the spectrum for more legible interpretation.

2.3.c Effect of sample length and model order

Over and under-modelling



The blue line represents a AR spectrogram estimate and the green line represents a traditional periodogram.

Figure 2.15: PSD estimates from AR coefficients for 10000 and 1000 length samples

From figure 2.15 we see the effects of having a large number of samples and what high model orders can do. For high model we see a better definition of the spectrum as seen when passing from 4 to 5 AR coefficients in figure 2.14. Past this the effect of increasing the order only marginally increases the quality of the PSD estimate - at the expense of greater computational power. Additionally a higher model order may start picking up erroneous frequencies that would make interpretation of the data harder.

Increasing the number of samples leads to an even worse estimate of the magnitude of frequencies but does guarantee a sharper distinction between frequencies.

2.4 Time Frequency Estimation

2.4.a Matlab `spectogram` function

We find that the best values for creating a spectrogram of a self generated 'chirp' signal are a FFT length of 128 with a very close overlap (120) to ensure a smooth spectrum. Indeed making the overlap closer to the nfft makes it smoother.

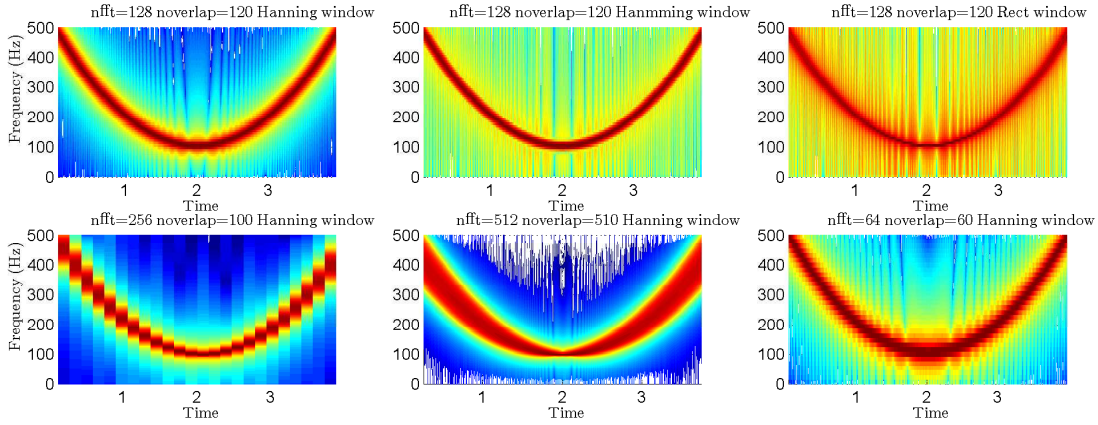


Figure 2.16: Spectogram for various nfft, overlap and window settings for chirp signal

Hanning window was preferred for this example due to the higher contrast it offers to other window types.

For the spectrogram we have the choice of increasing the FFT and window size. The overlap allows more time resolution while being able to maintain a high frequency resolution (nfft). However as this takes larges sample lengths it can lead to some distortion as seen in the accompanying figure.

2.4.b `spectogram` on SSVEP data

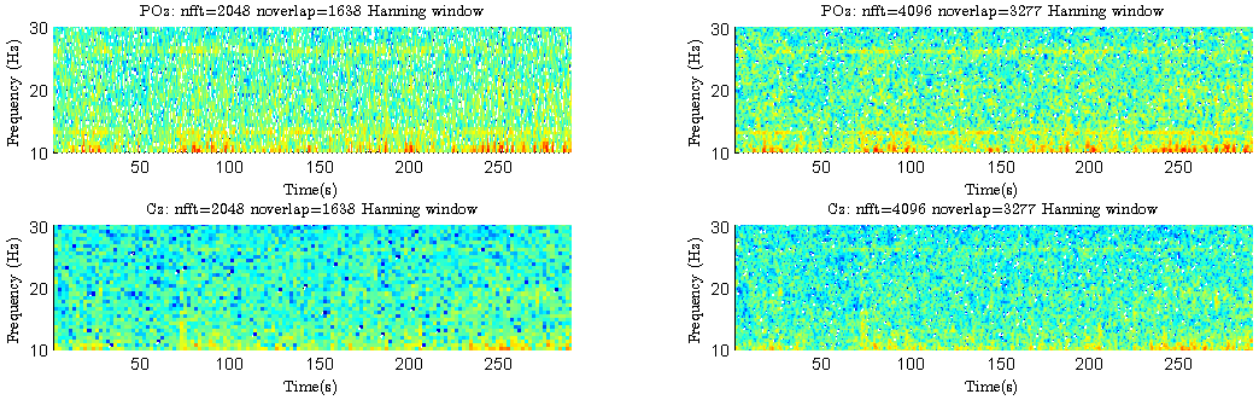


Figure 2.17: Spectogram for various nfft, overlap and window settings for EEG data

In figure 2.17 the SSEVP stimulus was sought in both the right and left channel of the EEG data. From previous tests it is know that this is 13Hz. In the case for the POz data the 13Hz peak can be seen (the yellow line at 13Hz), however this is absent from the Cz data, as earlier noticed. Increasing frequency bins did not help resolve.

It was found that the Hanning window also provides the best contrast here, allowing even the SSVEPs harmonics to be identified.

In choosing `nfft` the compromise was involved between time and frequency resolution - as increasing frequency bins decreases time resolution. Due to sampling frequency being 1200Hz and that we wanted to separate out one specific frequency a starting point was to have more than 1200 frequency bins. However exactly 1200 leads to a range of possible frequencies for one sample - between 13 and 14Hz for example. To pin it more to a specific

frequency the `nfft` used was a higher 2048 and 4096.

Part 3

Adaptive signal processing

3.1 The least mean square (LMS) algorithm

3.1.a Correlation Matrix for $AR(2)$ process

Here we have a system defined by $x[n] = a_1x[n-1] + a_2x[n-2] + \eta[n]$ with $a_1 = 0.1$, $a_2 = 0.8$ and $\sigma_\eta^2 = 0.25$. With the input vector $\begin{bmatrix} x[n-1]x[n-2] \end{bmatrix}^T$ we can form the correlation matrix $\mathbf{R} = \mathbb{E}\left\{\mathbf{x}(n)^T\mathbf{x}(n)\right\}$.

$$\mathbf{R} = \mathbb{E}\left\{\begin{bmatrix} x[n-1]^2 & x[n-1]x[n-2] \\ x[n-2]x[n-1] & x[n-2]^2 \end{bmatrix}\right\} = \begin{bmatrix} \mathbb{E}\left\{x[n-1]^2\right\} & \mathbb{E}\left\{x[n-1]x[n-2]\right\} \\ \mathbb{E}\left\{x[n-2]x[n-1]\right\} & \mathbb{E}\left\{x[n-2]^2\right\} \end{bmatrix} \tag{3.1}$$

Thus we are interested in $\mathbb{E}\left\{x[n-1]^2\right\}$ and $\mathbb{E}\left\{x[n-1]x[n-2]\right\}$. Note that $\mathbb{E}\left\{x[n-1]^2\right\} = \mathbb{E}\left\{x[n]^2\right\} = \mathbb{E}\left\{x[n-2]^2\right\}$ as this sample is taken from the same time series. Thus $\mathbf{R}$ is symmetric.

To find $\mathbb{E}\left\{x[n]^2\right\}$ and $\mathbb{E}\left\{x[n]x[n-1]\right\}$ we carry out the following:

$$\begin{aligned}
x[n] &= a_1x[n-1] + a_2x[n-2] + \eta[n] && \text{AR(2) definition} \\
x[n]x[n-k] &= a_1x[n-1]x[n-k] + a_2x[n-2]x[n-k] + \eta[n]x[n-k] && \text{Multiply by } x[n-k] \\
\mathbb{E}\left\{x[n]x[n-k]\right\} &= \mathbb{E}\left\{a_1x[n-1]x[n-k] + a_2x[n-2]x[n-k] + \eta[n]x[n-k]\right\} = \phi(k) && \text{Take expectation} \\
\phi(k) &= a_1\phi(k-1) + a_2\phi(k-2) && \text{For } k \geq 1
\end{aligned}$$

Thus $\mathbb{E}\left\{x[n]^2\right\} = \phi(0)$ and $\mathbb{E}\left\{x[n]x[n-1]\right\} = \phi(1)$. We can calculate $\phi(0)$,$\phi(1)$ and $\phi(2)$ as such:

$$\begin{aligned}
\phi(0) &= \mathbb{E}\left\{x[n]^2\right\} = a_1\phi(-1) + a_2\phi(-2) + \mathbb{E}\left\{\eta[n]^2\right\} \\
\phi(0) &= a_1\phi(-1) + a_2\phi(-2) + \sigma_\eta^2 \\
\phi(0) &= a_1\phi(1) + a_2\phi(2) + \sigma_\eta^2 && \text{Using symmetry property} && (3.2) \\
\phi(1) &= a_1\phi(0) + a_2\phi(1) && && (3.3) \\
\phi(2) &= a_1\phi(1) + a_2\phi(0) && && (3.4)
\end{aligned}$$

We solve the simultaneous equations 3.2, 3.3 and 3.4 for $a1 = 0.1$ and $a2 = 0.8$ and get:

$$\mathbf{R} = \begin{bmatrix} \frac{25}{27} & \frac{25}{54} \\ \frac{25}{54} & \frac{25}{27} \end{bmatrix} \tag{3.5}$$

Note that these are only expected values.

For the convergence of the LMS in the mean we have:$0 < \mu < \frac{2}{p\phi(0)}$ with $p = \text{trace}(\mathbf{R}_{xx})$. Thus $0 < \mu < (\frac{=759}{625} = 1.1664)$. This is only an estimate and the formal way is that it is bounded by $0 < \mu < \frac{2}{\lambda_{max}}$ where λ is the largest eigenvalue of $\mathbf{R}_{xx}$. This is seen to be $\lambda_{max} = 1.389$ making the bound max at 1.44.

For the convergence of the LMS in the mean square we have: $0 < \mu < (\frac{1}{3p\mathbf{r}_{xx}(0)} = \frac{6561}{15625} = 0.419904)$.

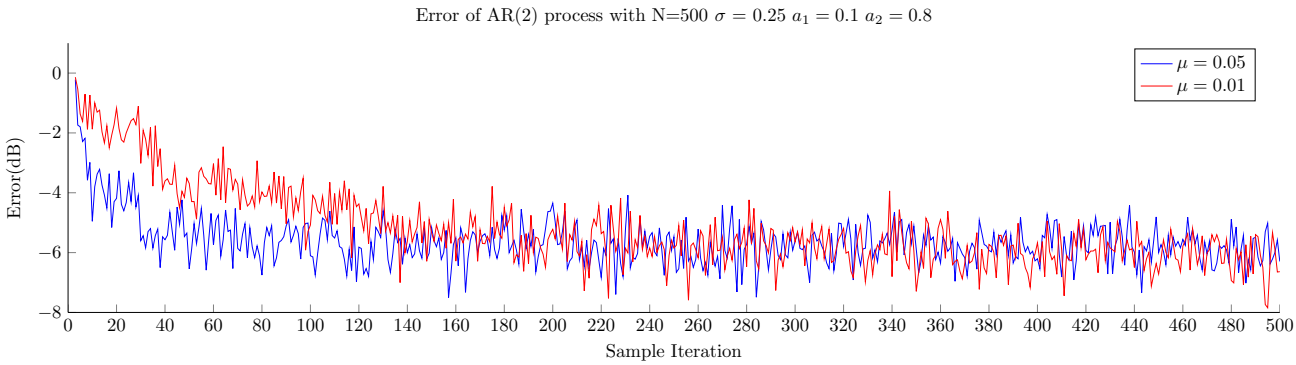


Figure 3.1: PSD estimates from AR coefficients

3.1.b LMS Filter error

Here we have created a LMS filter that learns the ideal **w** vector other 500 iterations. We see that changing μ, the learning rate, leads to a slower decrease rate in error over each iteration. This shows the compromise between wanting a fast convergence (low error) and that of a convergence that is not too fast to avoid overshoot and be stable.

An important observation is the error converging to values near -6.02dB. This is because the filtered AWGN has $\sigma^2 = 0.25$, which is essentially the minimum theoretical error.

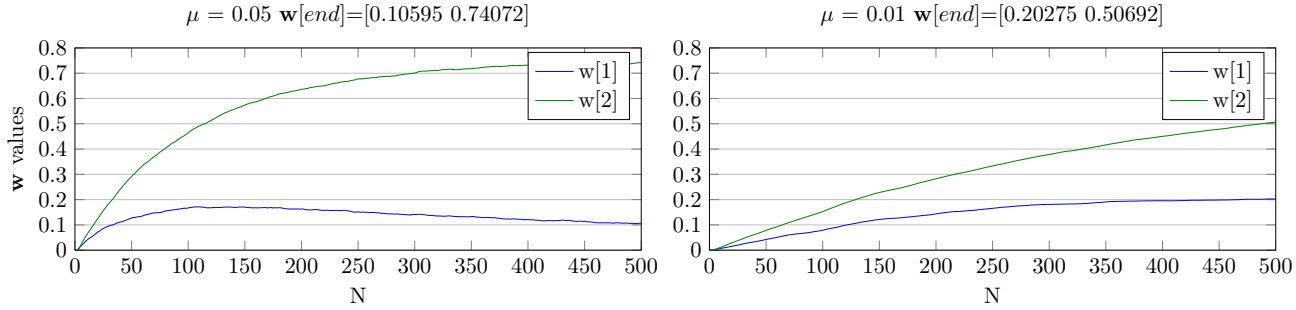
3.1.c Excess Mean Square Error

The theoretical excess error is: $\mathcal{M} = \frac{\mu}{2}\mathbf{Tr}\{\mathbf{R}\}$. Thus for each μ we have $\mathcal{M}_{\mu=0.05} = \frac{5}{54}$ and $\mathcal{M}_{\mu=0.01} = \frac{1}{54}$.

	MSE	**EMSE**	$\mathcal{M}$	$\mathcal{M}_{LMS}$
μ =**0.01**	0.2517	0.0017	0.0069	0.009259
μ =**0.05**	0.2616	0.0116	0.0462	0.046296

Here we see that the $\mathcal{M}$ and the $\mathcal{M}_{LMS}$ are not too disimilar, within the correct order of magnitude. It can be seen how larger μ values cause a larger EMSE due to the the larger oscillations this produces for converged values.

3.1.d Steady State values



Note: Starting AR(2) coefficients are $a_1 = 0.8$, $a_2 = 0.1$ and $w = [a_1 a_2]$

Figure 3.2: AR(2) coefficient estimates average over 100 independent trials

Thus we have, after 500 iterations, $\mathbf{w}_{\mu=0.01}$=[$\mathcal{N}(0.20275, 0.0518)$ $\mathcal{N}(0.50692, 0.0462)$] and $\mathbf{w}_{\mu=0.05}$=[$\mathcal{N}$(0.105950.0495) $\mathcal{N}(0.74072, 0.0500)$] which compared to the original coefficients is acceptable. Indeed a high sampling frequency could mean a convergence in less than a few seconds. We see however that for $\mu = 0.01$ the convergence rate is a bit slow, especially considering $\mu = 0.05$ doesn't suffer from excessive overshoot and converges faster.

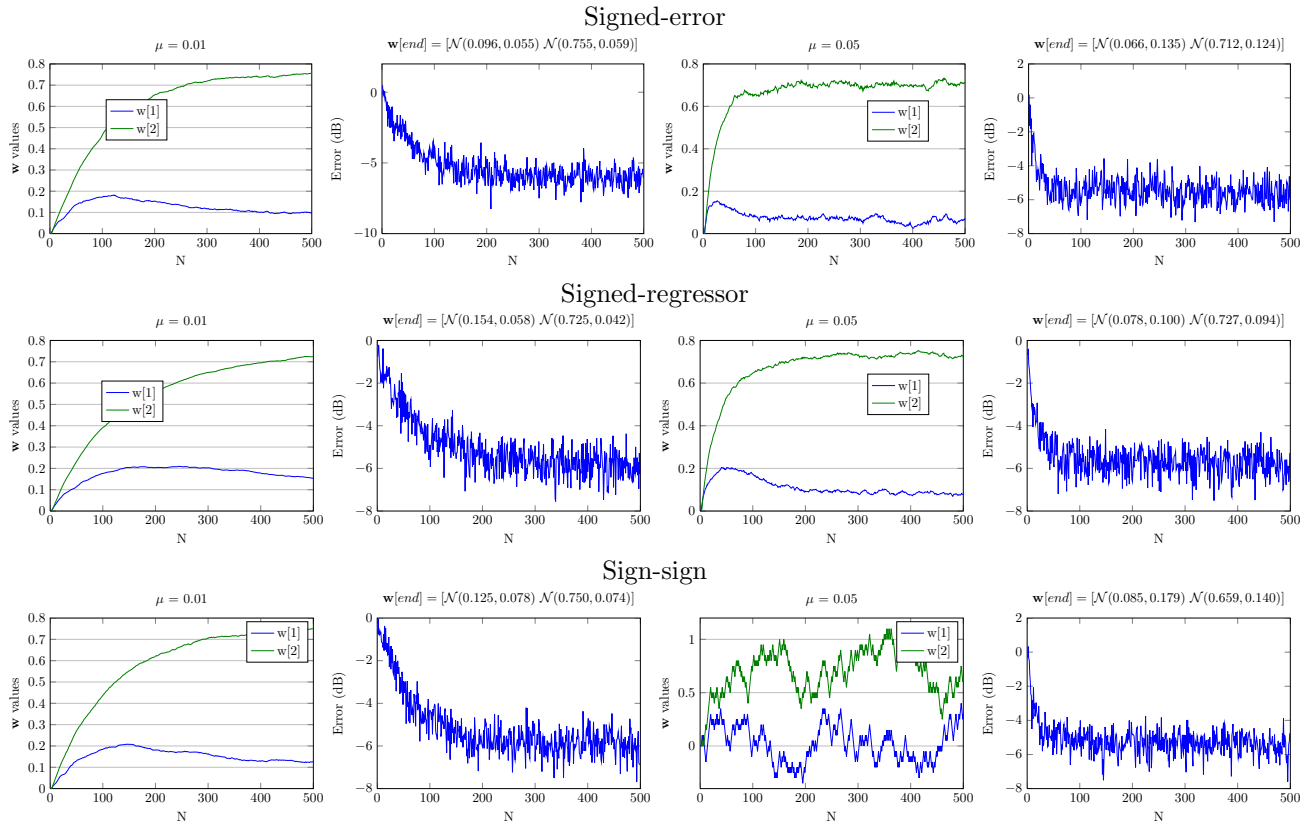
3.1.e Sign based optimisation

From figure 3.3 we can see that increasing the learning rate also (expectedly) changes the variance of the final values **w** takes.

All of the above algorithms cause a change in the error function. For example the signed-error version causes the error to be $J[n] = |e[n]|$. This affects the complexity of the algorithm (time to run each step), as well as the rate of convergence.

The signed-error converges faster than the signed-regressor algorithm, as seen by comparing graphs. However note that this can negatively impact the stable values, having higher variance.

The sign-sign algorithm causes the step change of $\mathbf{w}$ to be μ, meaning convergence takes at minimum $\frac{endvalue-startvalue}{\mu}$. Convergence will typically take longer however due to the rate of convergence slowing as ideal values approach. The variance and stabilisation of the algorithm is directly reliant on μ. This is the least computationally intensive algorithm.



Note: For the sign-sign regressor the graph for $\mu = 0.01$ is for one instance whereas the $\mu = 0.05$ is an averaging for 100 instances. For signed-regressor and signed error algorithms an average of 100 different trials was taken.

Figure 3.3: Error and $\mathbf{w}$ estimates for different LMS algorithms

Signed-Error For the signed error we notice that the error is reasonably small and can lead to slightly faster convergence, though less accurate convergence. This is due to the learning curve only being associated with the sign of the error and not its magnitude, i.e. any correlation to errors magnitude has been removed.

Signed-error	**MSE**	**EMSE**	$\mathcal{M}$
$\mu =$**0.01**	0.25459	0.00459	0.01837
$\mu =$**0.05**	0.28333	0.03333	0.13331

Signed-Regressor For this mechanism only information concerning the current sample is partly ignored. However considering the current sample is taken into consideration while calculating error this is not as bad as the signed error. The variance for final $\mathbf{w}$ values is smaller for signed-regressor than for sign-sign.

Signed-regressor	**MSE**	**EMSE**	$\mathcal{M}$
$\mu =$**0.01**	0.25876	0.00876	0.03503
$\mu =$**0.05**	0.26185	0.01185	0.04741

Sign-Sign Here only μ provides the amplitude of movement and both the error and the current sample combined provide a direction. This does not perofrma as well as other methods - this can be seen by the higher variance in final values.

Sign-sign	**MSE**	**EMSE**	$\mathcal{M}$
μ =**0.01**	0.25740	0.00740	0.02961
μ =**0.05**	0.28309	0.03309	0.13238

3.2 Recursive Least Squares Algorithm

3.2.a Growing window RLS comparison to LMS

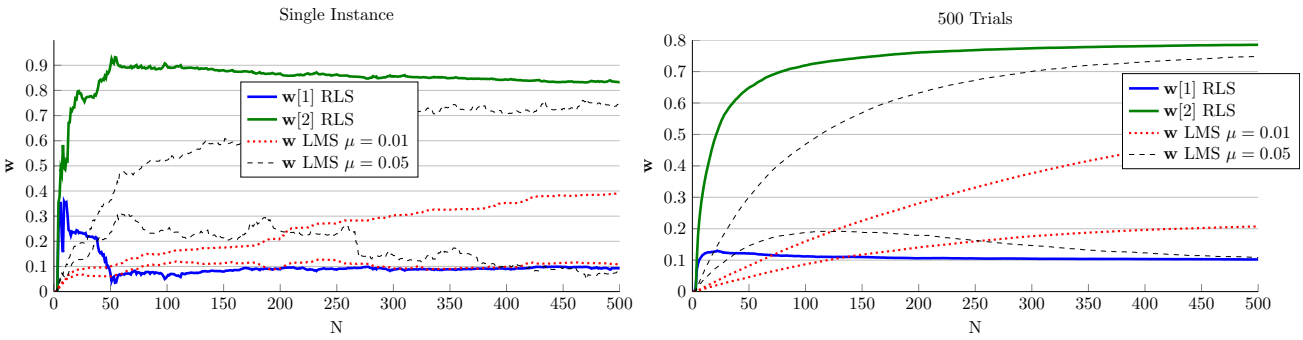
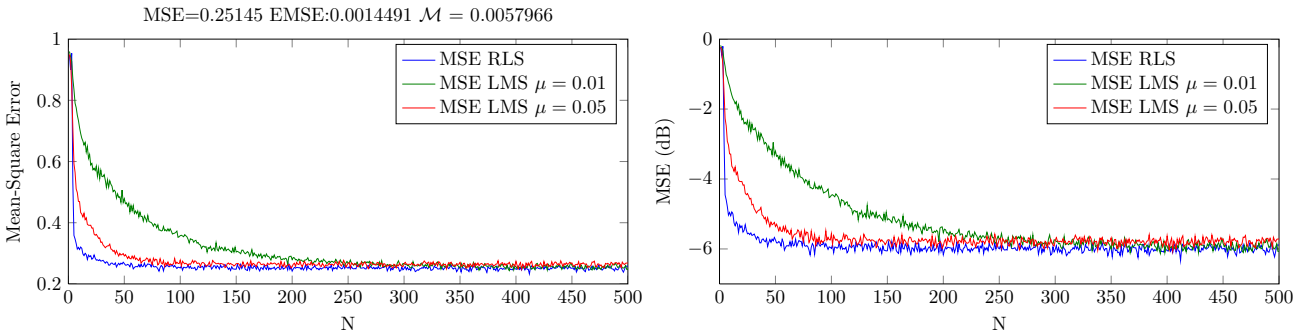


Figure 3.4: Comparison of rate of convergence for RLS compared to LMS with different μ

From figure 3.4 it can be seen that the growing window RLS converges faster than LMS. However the overshoot could be seen as worse than LMS. The final values of the RLS algorithm are also closer to that of the initial filter ($\mathbf{w}_{RLS} = [0.0994\ 0.8085]$ vs $\mathbf{w}_{LMS\mu=0.05} = [0.1301\ 0.7953]$).

This can be explained by the fact that LMS is an adaptive algorithm considering each incoming value as a seperate indicator and converging towards it. The RLS instead considers each incoming one as a whole with the previous ones. This means that once it has collected enough values it can converge quicker and explains its more erratic behaviour at the start - due to having less values to make an estimate from.

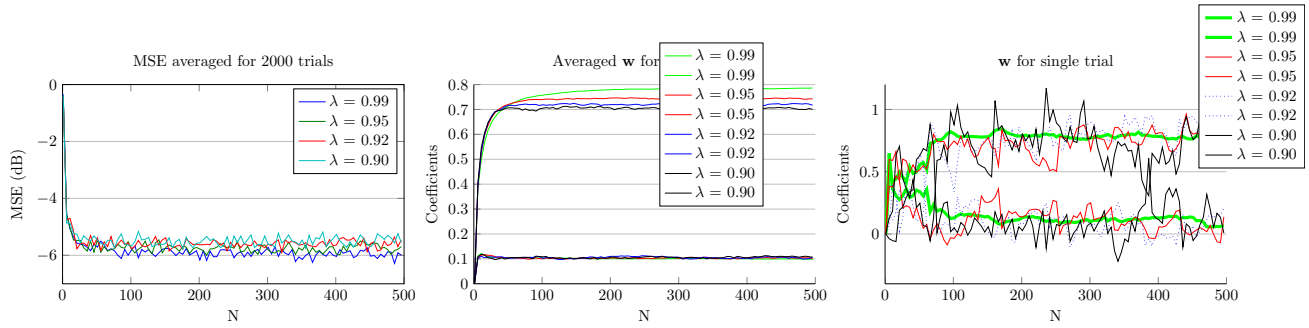
3.2.b Learning Curve



LMS for $\mu = 0.01$ MSE=0.25 EMSE=0.0050 $\mathcal{M}$=0.0199 (this is slightly inaccurate due to values not having fully converged)
LMS for $\mu = 0.05$ MSE=0.26 EMSE=0.0136 $\mathcal{M}$=0.0543

Figure 3.5: Comparison of learning rate for RLS compared to LMS with different μ

In figure 3.5 we can see that the misadjustment is better for an RLS filter than a LMS filter despite the similar MSE. This could be in part due to the RLS filter being more stable for fully converged values, whereas the LMS filter still "jumps" around.



$$\mathbf{w}_{\lambda=0.99} = (\mathcal{N}(0.099\ 0.044), \mathcal{N}(0.786\ 0.045))$$
$$\mathbf{w}_{\lambda=0.95} = (\mathcal{N}(0.101\ 0.104), \mathcal{N}(0.743\ 0.104))$$
$$\mathbf{w}_{\lambda=0.92} = (\mathcal{N}(0.106\ 0.139), \mathcal{N}(0.717\ 0.138))$$
$$\mathbf{w}_{\lambda=0.90} = (\mathcal{N}(0.107\ 0.161), \mathcal{N}(0.701\ 0.164))$$

Figure 3.6: Comparison of learning rate for RLS for different λ

3.2.c Weighting of λ

From investigations in figure 3.6 we see that the error converges to a stable value for each λ. Interestingly the higher the λ value, the higher the stable error value. This is because a higher λ causes $\mathbf{w}$ to be less accurate at its converged state. For example $\mathbf{w}_{\lambda=0.99}$ is closer to the real value than $\mathbf{w}_{\lambda=0.90}$, which also has a larger variance over 2000 trials. This is reflected in the single trial graph where we see that a lower λ causes the RLS to be less stable with the value jittering a large amount. Thus the only benefit of decreasing λ is faster convergence at the cost of worse estimates and less stable converged values.

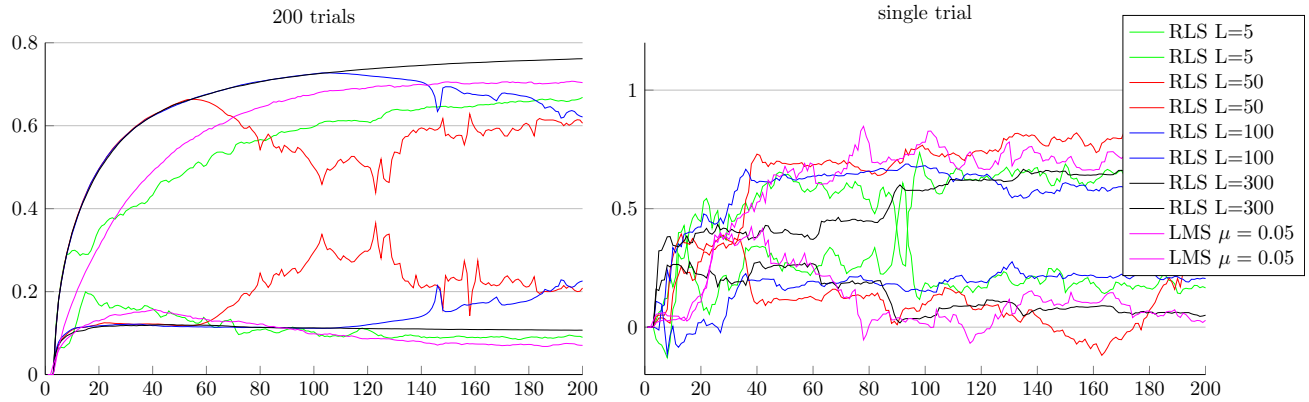
3.2.d Sliding window RLS

In figure 3.7 and 3.8 we see that the coefficients and the error for various RLS lengths converge but that their stability is impacted by this choice. Very short window lengths perform worse than the LMS algorithm in terms of stability and error. For example for RLS of L=5 the two coefficients are slightly better for the LMS - however the variance of the sliding RLS is lower.

We see that for semi short window lengths, i.e. 50 and 100,0 the coefficients can be very unstable. This is especially obvious for the error, where the EMSE is much worse.

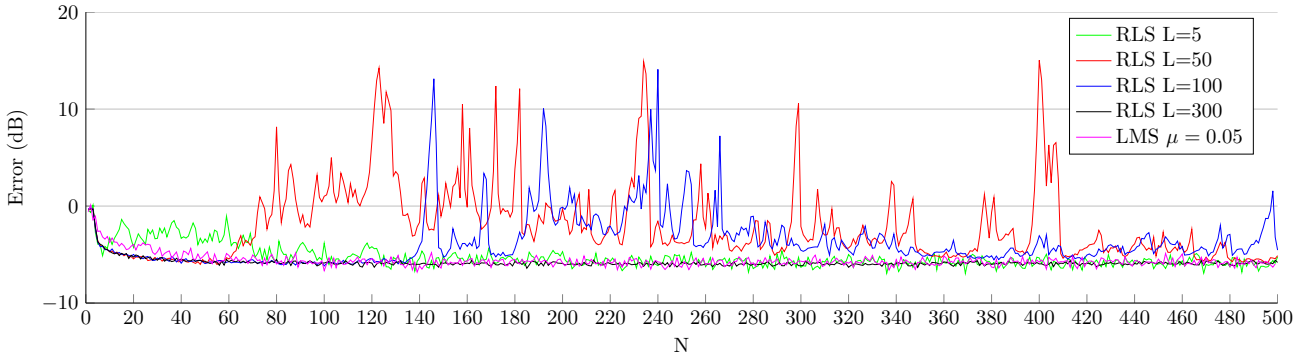
For very long window lengths the sliding window RLS starts resembling a usual RLS. A very long window, 300, produces a better estimate than LMS and approaches that of a traditional RLS.

Thus the window length is a compromise between convergence and variance. We want a quick local convergence - favoured by a shorter window, but also a lower variance, facilitated by a longer window. It is thus up to the engineers choice to decide what is best depending on the application.



L=5 $\mathbf{w}$ = ($\mathcal{N}$(0.08302, 0.063211) $\mathcal{N}$(0.72296, 0.057362))
L=50 $\mathbf{w}$ = ($\mathcal{N}$(0.1262, 0.15037) $\mathcal{N}$(0.71234, 0.14853))
L=100 $\mathbf{w}$ = ($\mathcal{N}$(0.17788, 0.71909) $\mathcal{N}$(0.67797, 0.73032))
L=300 $\mathbf{w}$ = ($\mathcal{N}$(0.12425, 0.093108) $\mathcal{N}$(0.75651, 0.090693))
μ=0.05 $\mathbf{w}$ = ($\mathcal{N}$(0.081341, 0.074355) $\mathcal{N}$(0.71611, 0.076142))

Figure 3.7: Comparison of learning rate for sliding RLS for different window lengths



L=5 MSE:0.2681 EMSE:0.018098
L=50 MSE:1.2123 EMSE:0.96225
L=100 MSE:0.70171 EMSE:0.45171
L=300 MSE:0.25425 EMSE:0.0042499
μ=0.05 MSE:0.2634 EMSE:0.013405

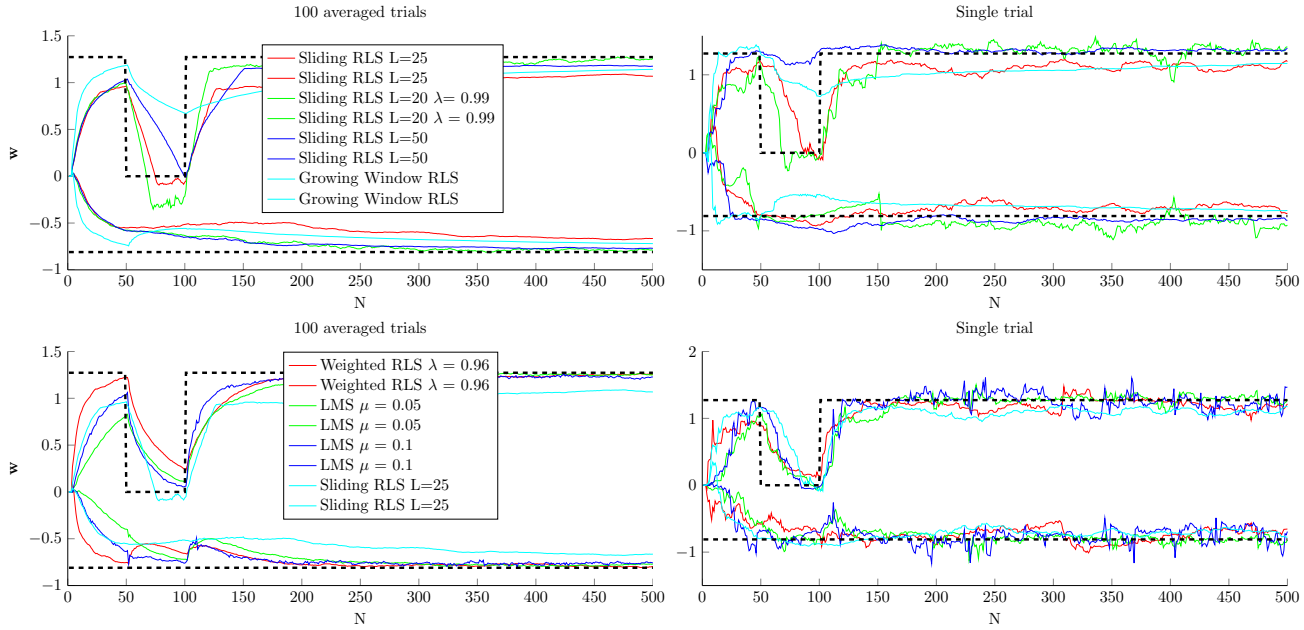
Figure 3.8: Comparison of error for sliding RLS for different window lengths

3.2.e Time varying difference

Here we see from figure 3.9 that the adaptation time of various algorithms for a changing a_1 coefficient is indeed different. The sliding RLS, despite giving more jagged signals does indeed respond better to changing coefficients. In all cases presented it is the first to reach the new a_1 value at 0. A long window length is not especially useful as it does not respond as fast. Window lengths of 20 and 25 were found to perform best for changing coefficients. This is because a shorter window length weights the current samples more so is better at representing the current value. However this means less samples are considered than the typical RLS so it is subject to high variance.

In the second subfigure it was shown that for an LMS of $\mu = 0.1$ the coefficients can also be followed at a similar rate at the expense of a very noisy estimate of coefficients. It can be noticed that for most algorithms the estimate of the coefficient a_2 is impacted upon the change of a_1. This is because of the feedback of $\mathbf{w}$ typically in the feedback form $\mathbf{w}_{n+1} = \mathbf{w}_n + e(n)\mathbf{f}(n)$ where $\mathbf{f}(n)$ is a vector function that is scaled by $e(n)$ that affects the update of both a_1 and a_2. As soon as a_1 changes a large error appears and causes this effect.

If a coefficient changing process $x[n]$ was being processed, a sliding window RLS with $\lambda = 0.99$ would be chosen for linear prediction due to this combination being able to quickly adapt to changes and at the same time not being too noisy for practical purposes.



The black lines represent the a_1 and a_2 coefficients. We have :

$$a_1 = \begin{cases} 1.2728 & 0 \leq \text{n} < 50, \\ 0 & 50 \leq \text{n} < 100, \\ 1.2728 & 100 \leq \text{n} \end{cases}$$

$$\text{and } a_2 = -0.81$$

Figure 3.9: Comparison of learning rate for various algorithms for time varying coefficients

3.3 Adaptive Noise Cancellation

3.3.a Adaptive Line Enhancer delay

An Adaptive Line Enhancer (ALE) consists of a delay operator and a linear predictor. By having a correct delay the underlying noise and the predicted input can be uncorrelated allowing only noise to be suppressed from the signal estimate. Thus we are concerned with finding a predictor input which is uncorrelated with the signal, this is equivalent to choosing an appropriate Δ (i.e. the lag). This can be seen as a problem of estimating the mean square error $\mathbb{E}\{(s[n] - \hat{x}[n])^2\}$. In this case $s[n]$ is the signal and $s[n] = x[n] + \eta[n]$, $x[n]$ is the clean signal and $\eta[n] = v[n] + 0.5v[n-2]$ with $v[n] \sim \mathcal{N}(0,1)$ - white noise. Let us minimise the white noise with the given constraints.

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

We can only optimise $2\mathbb{E}\{(x[n] - \hat{x}[n])\eta[n]\}$ in terms of Δ so we simplify the optimisation criteria to $\mathbb{E}\{\hat{x}[n]\eta[n]\}$. This is due to $x[n]$ being completely uncorrelated from $\eta[n]$. Thus:

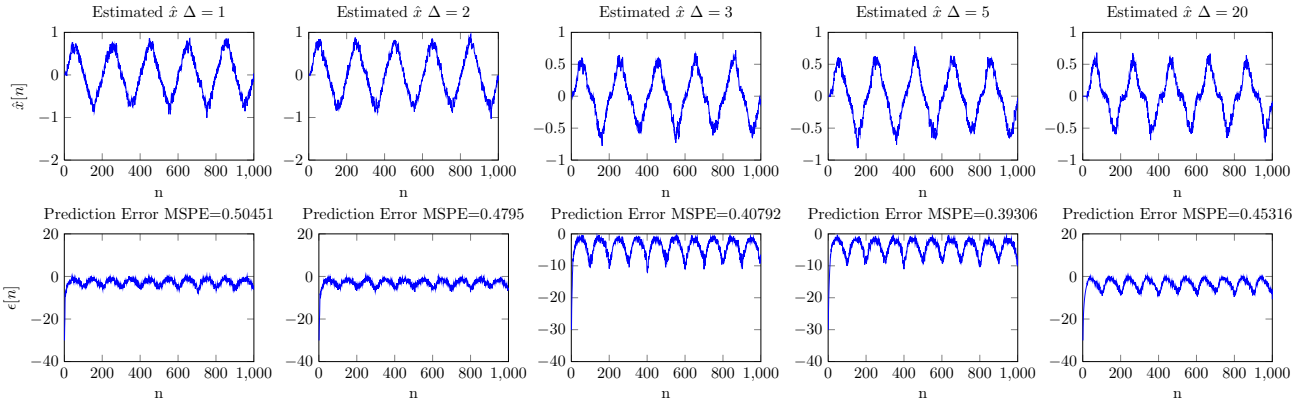
$$\begin{aligned} \mathbb{E}\left\{\hat{x}[n]\eta[n]\right\} &= \mathbb{E}\left\{\mathbf{w}^T\mathbf{u}[n](v[n] + 0.5v[n-2])\right\} && \text{convert } \mathbf{w}^T\mathbf{u}[n] \text{ to sum form} \\ &= \mathbb{E}\left\{\left(\sum_{i=0}^{M} w_k s(n - \Delta - i)\right)(v[n] + 0.5v[n-2])\right\} && \text{Expand } s(n-\Delta-i) \end{aligned}$$

$$= \mathbb{E}\left\{\left(\sum_{i=0}^{M} w_k(x(n-\Delta-i)+\eta(n-\Delta-i))\right)(v[n]+0.5v[n-2])\right\} \qquad x(n-\Delta-i) \text{ uncorrelated}$$

$$= \mathbb{E}\left\{\left(\sum_{i=0}^{M} w_k(\eta(n-\Delta-i))\right)(v[n]+0.5v[n-2])\right\} \qquad \text{subtitute } \eta(n-\Delta-i)$$

$$= \mathbb{E}\left\{\left(\sum_{i=0}^{M} w_k(v(n-\Delta-i)+0.5v(n-\Delta-i-2))\right)(v[n]+0.5v[n-2])\right\}$$

We end up with two instances of sums of white noise, $v[n]$, which at differing indices give an expectation of 0, i.e. $\mathbb{E}\left\{v[i]v[j]\right\} = 0 \; \forall \; i \neq j$ It is pretty self explanatory that we should have $\Delta \geq 3$ to minimise the mean square predicted error for ALE. From figure 3.10 we can see that there is less error for $\Delta \geq 3$ as well as a significantly lower MSPE.



Done for 100 different trials and an order M=2

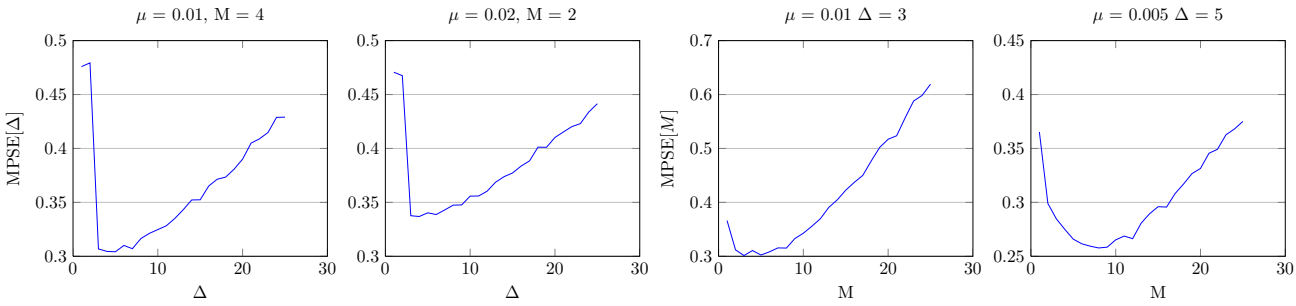
Figure 3.10: Comparison for various time lags

3.3.b Effect of Δ and order M

Here we investigate the performance of the ALE algorithm to denoise a signal based on the Δ (lag) and M (model order) parameters.

In the case of Δ we are interested in decorrelating the noise thus it is best if each sample used is of similar signal amplitude than the one to decorrelate. For example with $s[n] = x[n] + \eta[n]$ we have $x[n]$,a sine wave in our case. Ideally the lag should be at integer multiples of phase shifts to get a similar $x[n]$ amplitude as this would best remove noise. As we cannot immediately estimate accurately the phase of the signal (and it may not even be a sine wave) it is in our interest to choose the smallest possible decorrelating lag. This is seen in figure 3.11, where a lower lag is preferred - in this case 3 performs very well.

For model order we would expect higher the better as it can best describe a process. However there are limitations to this as a higher order is more susceptible to being unstable and not converging - or having bad overshoot. A smaller step size can partially solve this - as was done in figure 3.11. Though a smaller step size does decrease convergence rate and is not ideal. Thus the best model order would be between 3 and 9 as seen in figure 3.11.

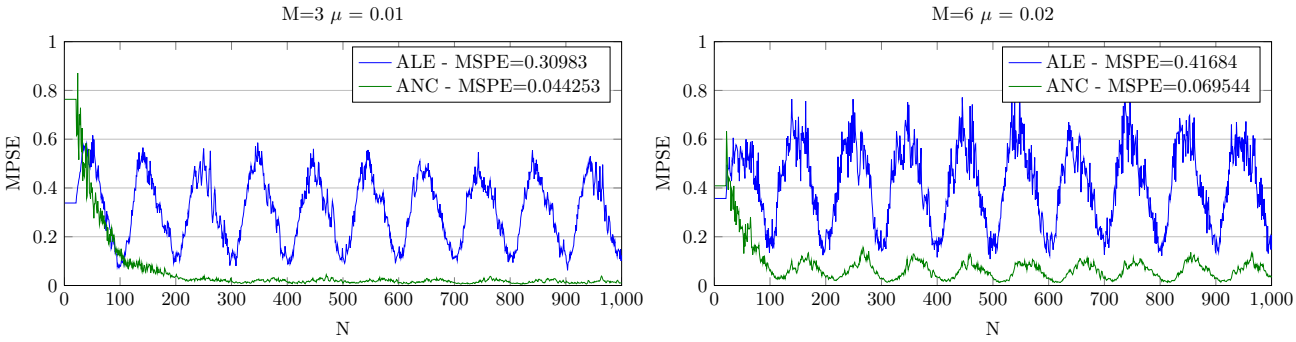


Done for 100 different trials

Figure 3.11: Comparison for Δ and M effects on MSPE

3.3.c Adaptive Line Enhancer compared to Adaptive Noise Cancellation

In the example in figure 3.12 ALE and ANC have been compared. For test purposes the noise was fed into the ANC algorithm, explaining its near perfect performance. This does however demonstrate the potential of ANC compared to ALE, suggesting that it be chosen if a good noise estimate is available.



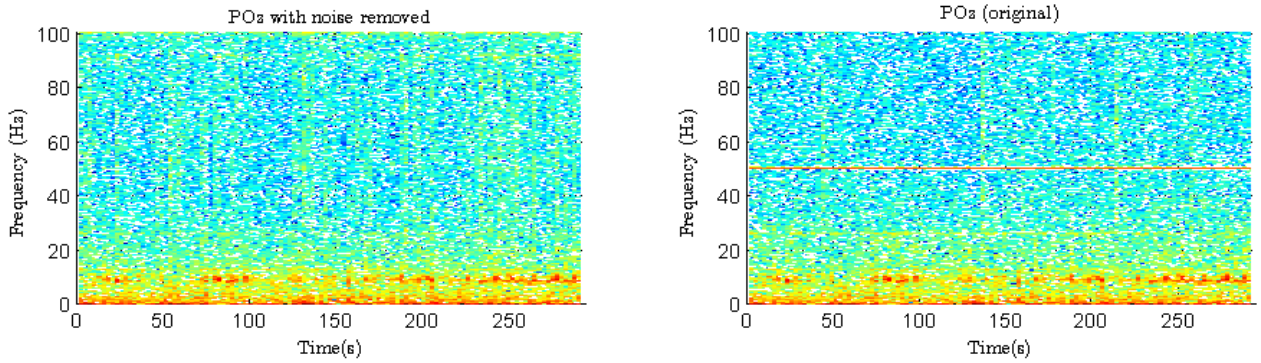
Done for 100 different trials

Figure 3.12: Comparison for ALE and ANC in terms of MSPE

3.3.d Processing EEG data with ANC

From figure 3.13 we see that the 50Hz mains DC noise can easily be removed by generating a synthetic sine signal with added white noise. With a good LMS filter the noise can be removed without affecting the rest of the signal.

Removing the noise for both left and right channels as in 3.14 shows us that the 50Hz can be removed - the coherence spectrum, for `Cz` with `POz` with noise removed, shows different relations than with noise.



M = 10 with a μ = 0.01 was found to produce best results

Figure 3.13: Comparison ANC denoised EEG data (right untouched, left denoised)

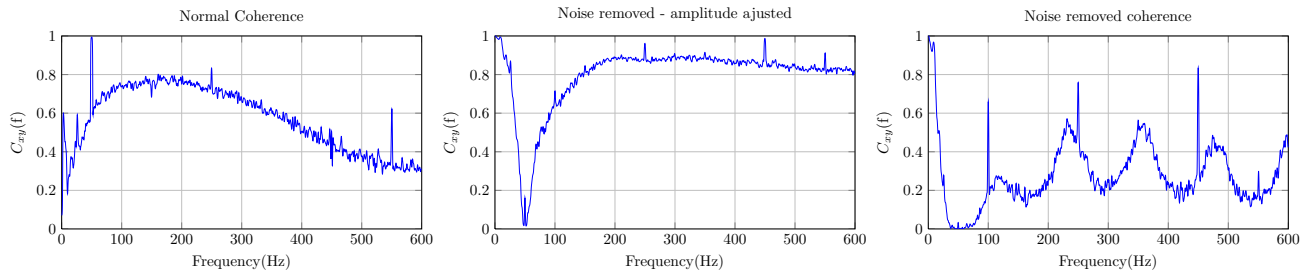


Figure 3.14: Coherence Spectrum comparison for noisy and de-noised left and right EEG channels

Part 4

Complex-valued statistical processing

4.1 Adaptive Step Sizes

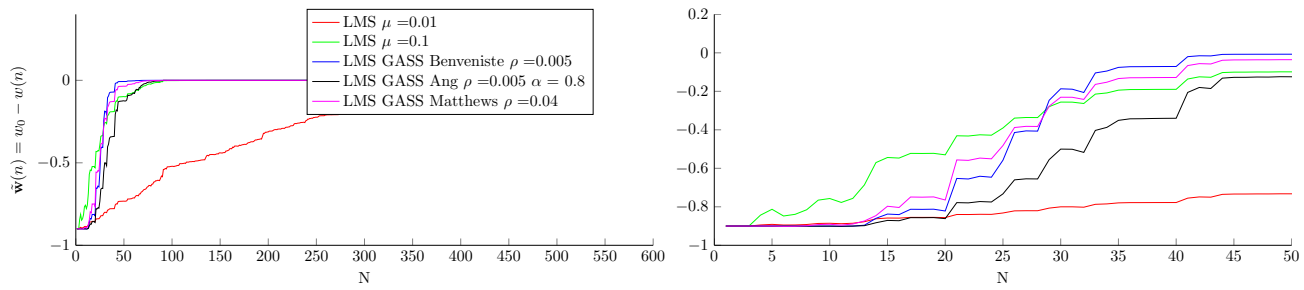
4.1.a LMS variations with adaptive step size

Here we are comparing the coefficient convergence for a MA(1) process. Originally LMS uses a fixed μ (step-size) however as seen this can impact convergence time as well as steady state stability. The idea now is to use gradient adaptive step-size (GASS) that relates the amplitude of the error in some way to the step size - i.e. the larger the error the bigger the step size. Here me minimise $\nabla_\mu J$ such that $\mu \to 0$ for steady state coefficients. This gives a general update equation of the form $\mu(n+1) = \mu(n) + \rho e(n)\mathbf{x}^T(n)\boldsymbol{\psi}(n)$ where $\boldsymbol{\psi}(n)$ is varied for each algorithm type.

The first used GASS algorithm is the Benveniste one which $\boldsymbol{\psi}(n) = [\mathbf{I} - \mu(n-1)\mathbf{x}(n)\mathbf{x}^T(n)]\psi(n-1) + e(n-1)\mathbf{x}(n-1)$. This is essentially correlates the step size to a low pass filtered version of $\nabla e(n-1)\mathbf{x}(n-1)$ (i.e. the gradient). This allows it to ignore small local variations (random noise) of the gradient. The Ang & Farhang algorithm is $\boldsymbol{\psi}(n) = \alpha\psi(n-1) + e(n-1)\mathbf{x}(n-1)$, effectively replacing the adaptive low pass filter coefficient calculation by a fixed α making it computationally lighter. The Matthews & Xie simply sets α to zero.

These results can be seen experimentally in figure 4.1, where the Benveniste algorithm is the first to converge and have a non existing steady state error.

Note that since we are now estimating a MA(1) process it is possible to have a near 0 steady state error, as seen in the accompanying figure.



Coefficients are modelled for a MA(1) process with $x[n] = 0.9\eta[n-1] + \eta[n]$ and $\eta[n] \sim \mathcal{N}(0, 0.5)$

Figure 4.1: Weight error curves for LMS variations

4.1.b NLMS update equation

The NLMS is a variant of the LMS equation that normalises the step size using $\mathbf{x}(n)$, resulting in:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\beta}{\epsilon + \|\mathbf{x}(n)\|^2} e(n)\mathbf{x}(n) \tag{4.1}$$

We are interested in showing that using the *a posteriori* error $e_p(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n+1)$ in the equation

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

is equivalent to the NLMS, equation 4.1. We can do this by rearranging 4.2.

$$\begin{aligned}
\mathbf{w}(n+1) &= \mathbf{w}(n) + \mu e_p(n)\mathbf{x}(n) && \\
\mathbf{x}^T\mathbf{w}(n+1) &= \mathbf{x}^T\mathbf{w}(n) + \mathbf{x}^T(n)\mu e_p(n)\mathbf{x}(n) && \text{Multiply by } \mathbf{x}^T \\
-\mathbf{x}^T\mathbf{w}(n) &= -\mathbf{x}^T\mathbf{w}(n+1) + \mathbf{x}^T(n)\mu e_p(n)\mathbf{x}(n) && \text{Swap two leading terms} \\
d(n) - \mathbf{x}^T\mathbf{w}(n) &= d(n) - \mathbf{x}^T\mathbf{w}(n+1) + \mathbf{x}^T(n)\mu e_p(n)\mathbf{x}(n) && \text{Add } d(n) \text{both sides} \\
e(n) &= e_p(n) + \mathbf{x}^T(n)\mu e_p(n)\mathbf{x}(n) && \text{Substitute for error terms} \\
e(n) &= e_p(n)\left(1 + \mu\|\mathbf{x}(n)\|^2\right) && \text{Factor} \\
e_p(n) &= \frac{e(n)}{1 + \mu\|\mathbf{x}(n)\|^2} && \text{Rearrange}
\end{aligned}$$

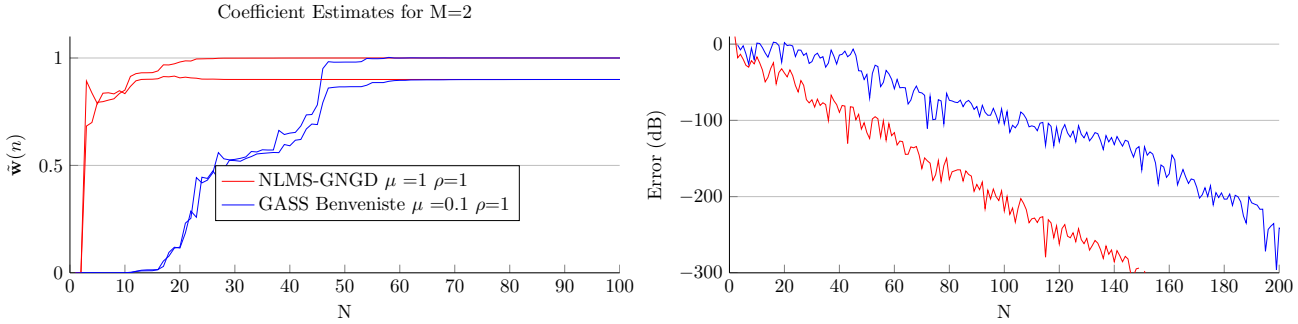
From the expression for $e_p(n)$ we can substitute it back into 4.2.

$$\begin{aligned}
\mathbf{w}(n+1) &= \mathbf{w}(n) + \mu e_p(n)\mathbf{x}(n) && \\
&= \mathbf{w}(n) + \mu\frac{e(n)}{1 + \mu\|\mathbf{x}(n)\|^2}\mathbf{x}(n) && \text{Substitute in } e_p(n) \\
&= \mathbf{w}(n) + \frac{1}{\frac{1}{\mu} + \|\mathbf{x}(n)\|^2} e(n)\mathbf{x}(n) && \text{Multiply nominator and denominator by } \frac{1}{\mu}
\end{aligned}$$

Thus it is equivalent to 4.1 for $\epsilon = \frac{1}{\mu}$ and $\beta = 1$. Note that we also have a non-zero ϵ regulator due to its inverse relation to μ.

4.1.c Generalized Normalized Gradient Descent (GNGD)

For a NLMS-GNGD we see from figure 4.2 that it provides a faster descent than the GASS Benveniste estimate. Parameters that optimise both functions were used for fair comparison.



Coefficients are modelled for a MA(1) process with $x[n] = 0.9\eta[n-1] + \eta[n]$ and $\eta[n] \sim \mathcal{N}(0, 0.5)$
Convergence is expected around 1 and 0.9

Figure 4.2: Comparison of NLMS-GNGD to GASS algorithm

To compare complexities an analysis of a single loop of each algorithm is done.
Complexity for GNGD

```
x_prev(:,i) = x(i:-1:i-M+1); %O(1)
x_e(i)= w_e(i,:)*x_prev(:,i); %O(2M-1)
coef(i) = mu/(epsi(i) + (x_prev(:,i)'*x_prev(:,i))); %O(2M-1+1+1)
e(i) = d(i) - x_e(i); %O(1)
w_e(i+1,:) =  w_e(i,:) + coef(i)*e(i)*x_prev(:,i)'; %O(2M+2)
a = (e(i)*e(i-1)*(x_prev(:,i)')*x_prev(:,i-1)); %O(2*M+1)
b = (epsi(i-1) + (x_prev(:,i-1)')*x_prev(:,i-1)) %O(2M)
epsi(i+1) = epsi(i) - p*mu*(a/b^2); %O(5)
```

Total complexity $= O(10M + 10) = O(10M)$ Complexity for GASS Benveniste

```
x_c = x(i:-1:i-M+1); %O(1)
x_prev = x(i-1:-1:i-M);%O(1)
x_e(i)= w_e(i,:)*x_c; %O(2M-1)
e(i) = d(i) - x_e(i); %O(1)
w_e(i+1,:) =  w_e(i,:) + mu(i)*e(i)*x_c'; %O(2M+1)
phi(i,:) = (eye(M) - mu(i-1)*(x_prev*x_prev'))*phi(i-1,:)'+e(i-1)*x_prev; %O(2M+2)
mu(i+1) = mu(i) + p*e(i)*x_c'*phi(i,:)'; %O(3M^2)+O((2M-1)*M)+O(1+M)
```

Total complexity $= O(4M^2) = O(4M^2 + 8M + 6)$

Note: M is the order and is what the speed mainly scales with.

The advantage of GNCD over GASS Benveniste is that it does not use any matrix multiplication which introduces a term of complexity $O(M^2)$. Thus not only does NLMS GNGD converge better but it is also more efficient.

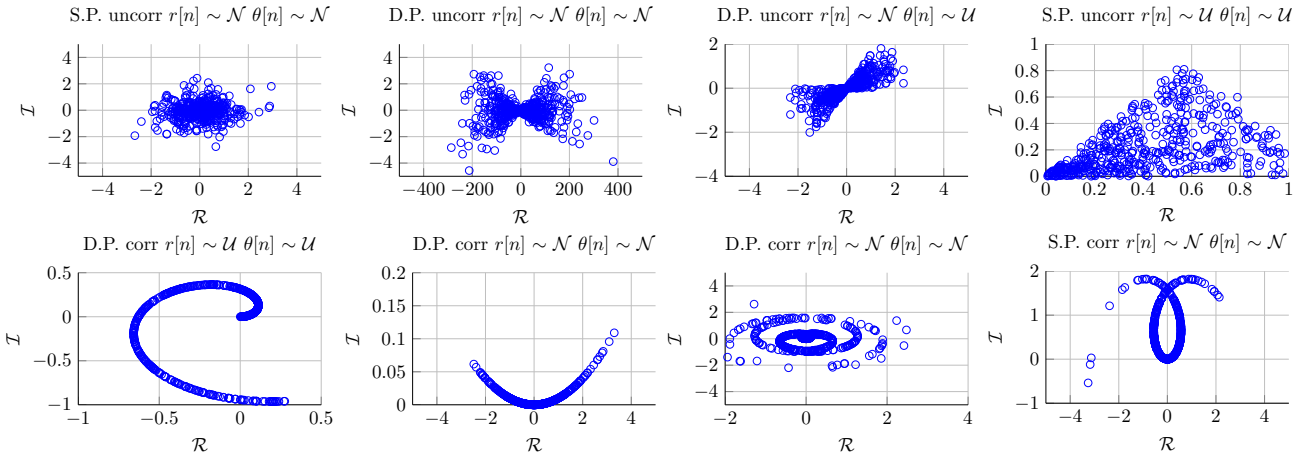
4.2 Complex Random Variables and Adaptive Filters

4.2.a Complex valued signals

Complex random variables can only be processed as generic real valued if they are considered circular. This is the case if $\rho = \frac{\mathbb{E}(zz^*)}{\mathbb{E}(z^2)} = \frac{p}{c} = 0$, or more specifically $p = 0$.

Further inspection of 4.3 shows that a clear visual difference between circular and non circular r.v.'s exists, where the top left graph, for same power uncorrelated, is the only circular r.v. displayed. It is called circular because it is invariant to a rotation of any degrees around the point $(0, 0)$.

For $p = 0$ we have $\sigma_x^2 - \sigma_y^2 + 2j\sigma_{xy} = 0$, where $z = x + iy$. Thus we need the real part to have the same power of the imaginary part and also be uncorrelated to be circular.



Plots of real and imaginary components of $z[n] = r[n] * e^{j\theta[n]}$ for 1000 different trials.
D.P. stands for different power
S.P. stands for same power

Figure 4.3: Comparison of scatter plots for various complex variables

4.2.b Complex Gaussian Noise

Here we generate complex-valued white Gaussian noise $z[n] = x[n] + jy[n]$ with $x[n] = \alpha x_1[n]$, $y[n] = \gamma x_1[n] + \theta x_2[n]$ where $x_1, x_2 \sim \mathcal{N}(0,1)$. We express the covariance and autocovariance in terms of α, γ and θ:

$$\begin{aligned}
\mathbb{E}(zz^*) &= \mathbb{E}\left((x+jy)(x-jy)\right) && \text{Covariance}\\
&= \mathbb{E}\left(x^2+y^2\right) = \mathbb{E}\left((\alpha x_1)^2 + (\gamma x_1 + \theta x_2)^2\right)\\
&= \mathbb{E}\left((\alpha^2+\gamma^2)x_1^2 + 2\gamma\theta x_1 x_2 + \theta^2 x_2^2\right)\\
&= (\alpha^2+\gamma^2)\sigma_{x_1}^2 + 2\gamma\theta\sigma_{x_1x_2} + \theta^2\sigma_{x_2}^2\\
&= \alpha^2+\gamma^2+\theta^2 = c && \text{As } x_1, x_2 \text{ independant}\\
\mathbb{E}(z^2) &= \mathbb{E}\left((x+jy)^2\right) && \text{Pseudo-Covariance}\\
&= \alpha^2-\gamma^2-\theta^2+2j\alpha\gamma = p
\end{aligned}$$

From this we get:

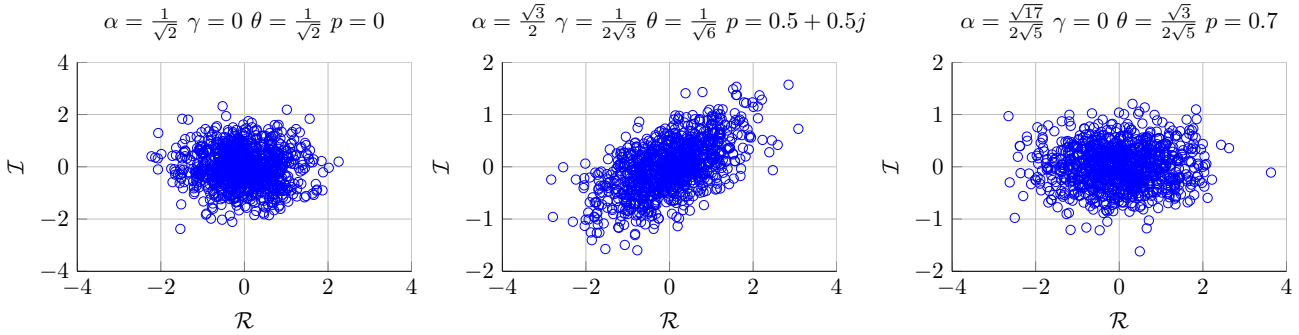
$$\begin{aligned}
\rho &= \frac{p}{c} && \text{Circularity Coefficient}\\
&= \frac{\alpha^2-\gamma^2-\theta^2+2j\alpha\gamma}{\alpha^2+\gamma^2+\theta^2}
\end{aligned}$$

For given c and p we can solve the values in table 4.1. This is possible due to have 3 equations and 3 unknowns.

p	$\boldsymbol{\alpha}$	$\boldsymbol{\gamma}$	$\boldsymbol{\theta}$
0	$\pm\frac{1}{\sqrt{2}}$	0	$\pm\frac{1}{\sqrt{2}}$
$\mathbf{0.5+0.5j}$	$\frac{\sqrt{3}}{2}$	$\frac{1}{2\sqrt{3}}$	$\pm\frac{1}{\sqrt{6}}$
$\mathbf{0.5+0.5j}$	$-\frac{\sqrt{3}}{2}$	$-\frac{1}{2\sqrt{3}}$	$\pm\frac{1}{\sqrt{6}}$
0.7	$\pm\frac{\sqrt{17}}{2\sqrt{5}}$	0	$\pm\frac{\sqrt{3}}{2\sqrt{5}}$

Table 4.1: Values satisfying the requirements of p with $c = 1$

We can make a circularity plot of each in 4.4. As we see a non-zero value of γ creates a correlation between x and y leading to a non-circular variable. Having a different α and θ causes them to have different powers and also being non-circular. This demonstrates how $p = 0$ makes a random variable circular.

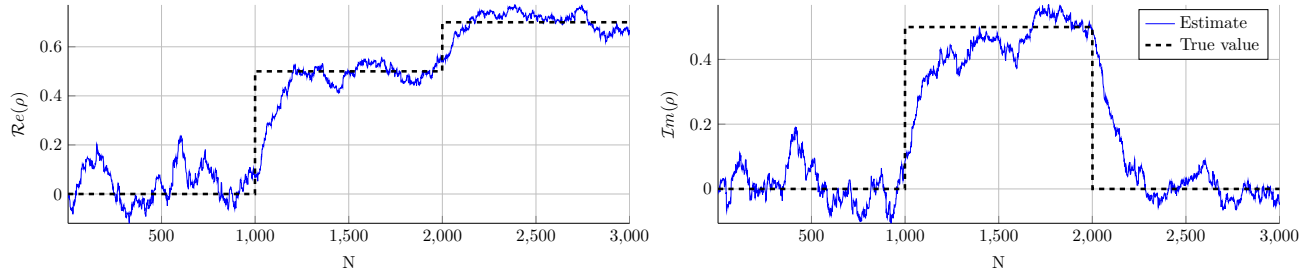


Plots of real and imaginary components of $z[n] = x[n] + iy[n]$ for 1000 different trials.

Figure 4.4: Comparison of scatter plots for various complex variables

4.2.c Circularity coefficient estimate

It is of interest to be able to track the circularity of a signal to see whether processing on it is sensible. Thus an estimate that updates over each new value of a signal is useful. This is done by a simple update mechanism $\hat{\rho}(n+1) = \hat{\rho}(n) + \mu e^*(n) z(n)$. In figure 4.5 we see that this estimate is able to provide a fair estimate in realtime.

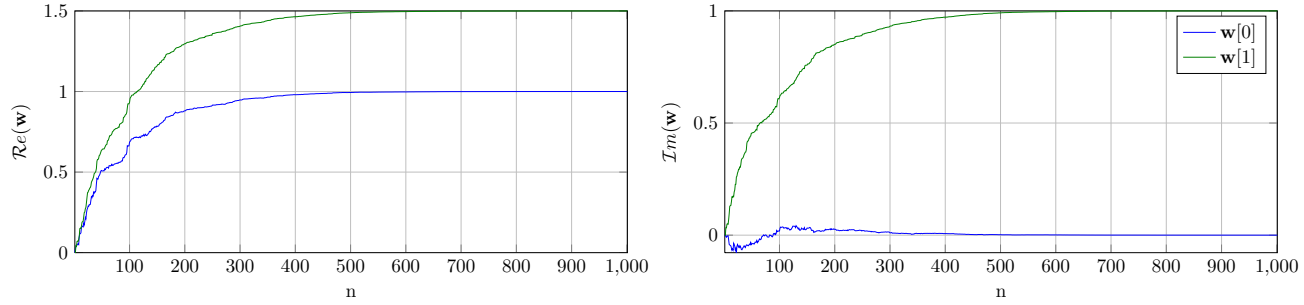


$z[n]$ is a vector of 100 samples each from the r.v.'s found in table 4.1 and figure 4.4. First 1000 for $\rho = 0$, then $\rho = 0.5 + 0.5j$ and $\rho = 0.7$.

Figure 4.5: Circularity estimate function for $\mu = 0.01$

4.2.d CLMS for $MA(1)$ process

In this part we implement a complex LMS function by changing the weight update equation to $\mathbf{w}[n+1] = \mathbf{w}[n] + \mu e^*(n)\mathbf{x}[n]$. A CLMS was implemented and its performance is found in figure 4.6. The $MA(1)$ process coefficient is $b = 1.5 + 1j$ and is correctly found for complex white Gaussian noise passed through this process.



White Complex circular Gaussian noise $z[n]$ was implemented using the coefficients from table 4.1. From this a process defined by $d[n] = b_1 z[n-1] + z[n]$ with $b = 1.5 + 1j$.
The convergence of the coefficients is reflected in the $\mathbf{w}[1]$ vector.

Figure 4.6: CLMS for $\mu = 0.01$

4.2.e Bivariate wind data circularity

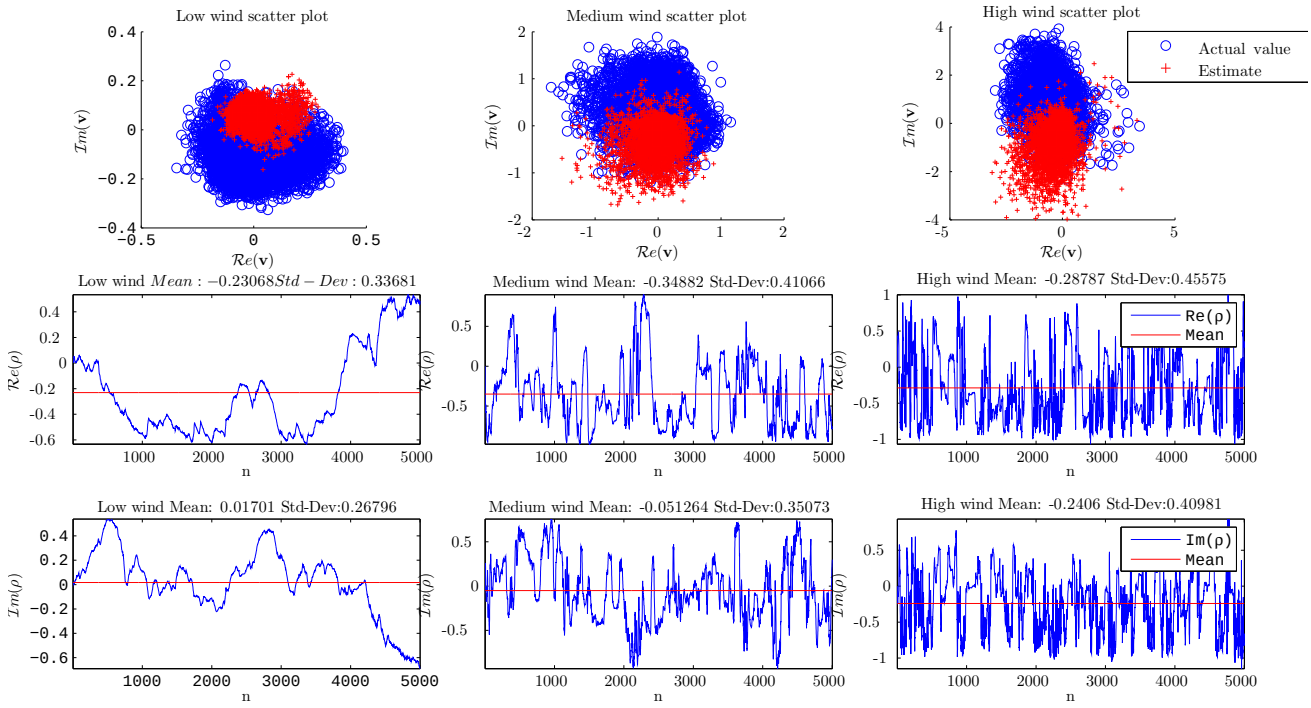
Here wind data for three different type of regimes was investigated for circularity. Before being processed a complex was generated by taking the east and north wind intensities into one variable: $v[n] = v_{east}[n] + jv_{north}$.This get all wind data into a single variable(south wind is simply negative that of north wind). All findings have been compiled to figure 4.7.

For the low wind regime data it can be seen the data looks mostly circular, as the scatter graph would look mostly the same for any rotation of the plot. This is reflected in the circularity estimate which lays mostly around -0.3 and .3 (with a max/min of -0.5/.5).

For the medium wind regime data the variations in the circularity estimate are higher ,the standard deviation is hgiher than before for both complex and real parts and the mean is further away from 0 - where it should be for perfect circularity. This can be seen from the scatter plot which exhibits a certain bias.

The high wind regime scatter plot is heavily biased and any rotation will be noticed. The high frequency of change in the circularity estimate and its high standard deviation values are two other indicators of noncircularity.

Thus overall the circularity tracker seems to reflect the circularity of the variables mainly in relation to is fast change in values and high amplitudes. Additionally we can comment that for low wind there seems to be the trend that it is simply rustling around and that for high regime wind the wind is more headed for just one direction, explaining the non-circular nature of it.



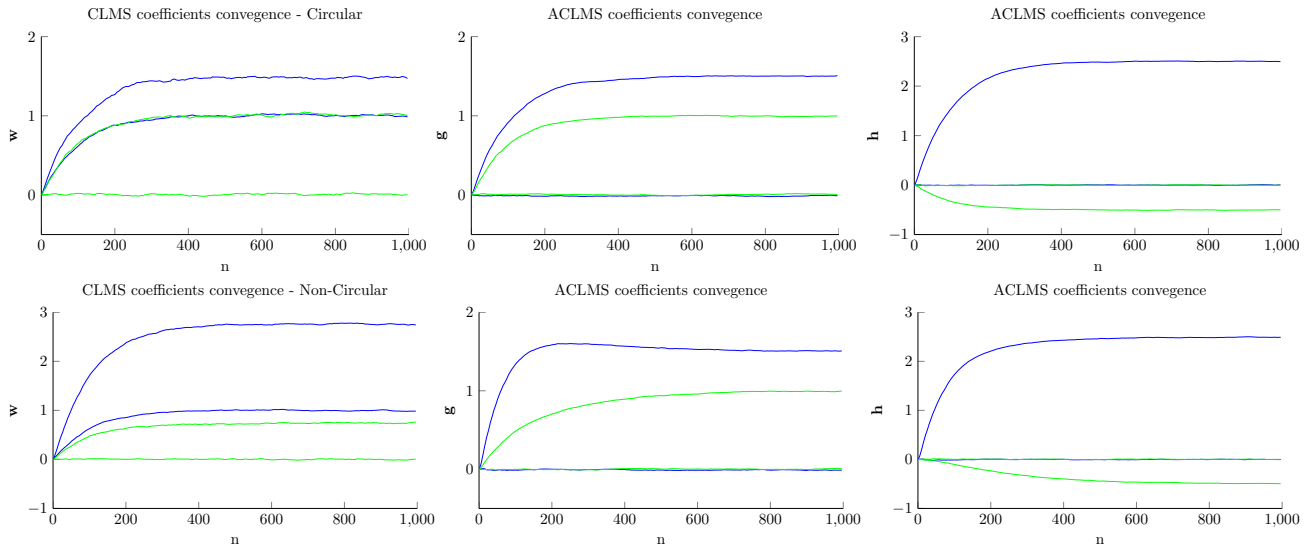
Done for $\mu = 0.1$ circularity

Figure 4.7: Wind scatter plot and circularity estimate (ρ)

4.2.f Comparison of CLMS to ACLMS

Here the Augmented Complex Least Mean Square and CLMS algortihm is used to model the coefficients of a Widely Linear Moving Average process (WLMA). For CLMS we have one weight vector, $\mathbf{w}$, which means we have a strictly linear extension but for ACLMS we have two weight vectors allowing us to take advantage of the second order statistics of the data, $\mathbf{g}$ and $\mathbf{h}$. The convergence of these is found in figure 4.8. We can see that the CLMS struggles to converge to correct values for non-circular data, though it seems to perform similarly to the ACLMS for circular data.

Another way of seeing this is to look at the error curves in figure 4.9 that show the ACLMS (in green) consistently outperforming the CLMS for both circular and non-circular data. For circular data we see that the errors are somewhat similar (steady state error of $-2.14dB$ vs $-9.34dB$, CLMS vs ACLMS)but do not agree for non-circular data (steady state error of $4.51dB$ vs $-3.37dB$, CLMS vs ACLMS).



Done for $\mu = 0.01$ over 100 different trials

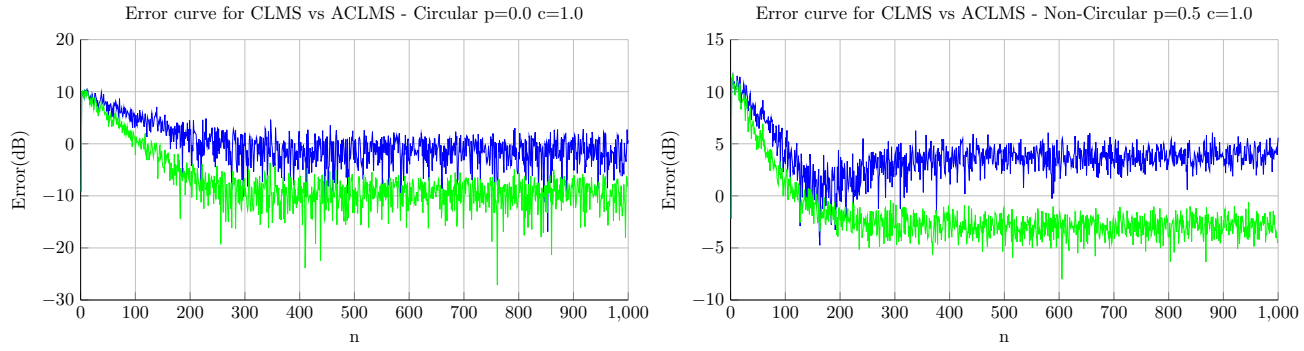
Note: The blue line denotes the real values and the green line the imaginary values.

In these graphs both circular and non circular white gaussian noise was generated and created the WLMA(1) using $y[n] = x[n] + b_1 x[n-1] + b_2 x^*[n-1]$ with $b_1 = 1.5 + 1j$ and $b_2 = 2.5 - 0.5j$

Circular noise was generated for $x \sim \mathcal{N}(0, c, p)$ with $c = 1$, $p = 0$

Non-circular noise was generated for $x \sim \mathcal{N}(0, c, p)$ with $c = 1$, $p = 0.5$

Figure 4.8: Coefficient Convergence comparing CMLS to ACLMS for circular and non-circular variables



Done for $\mu = 0.01$

Note: The blue line denotes the CLMS and the green line the ACLMS.

Figure 4.9: Error curves comparing CMLS to ACLMS for circular and non-circular variables

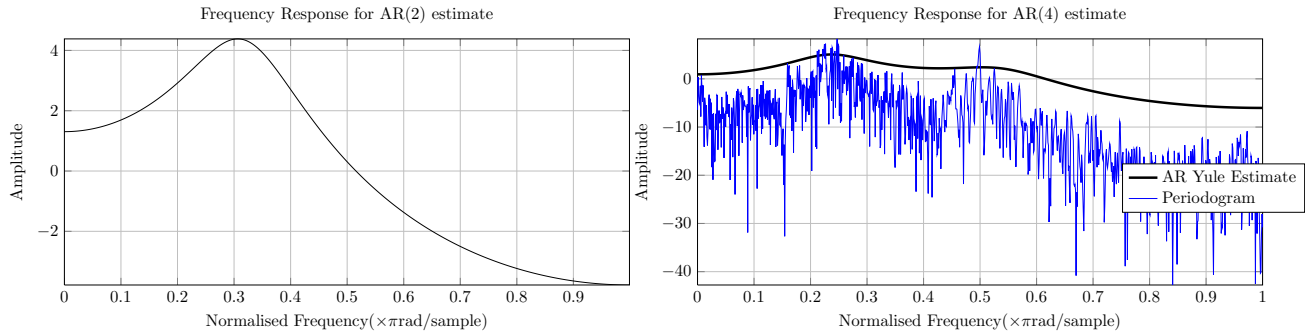
4.3 Adaptive Spectral Estimation

4.3.a Time varying coefficients

Here have a coefficient changing $AR(2)$ process of the form $x[n] = a_1[n]x[n-1] - 0.81x[n-2] + \eta[n]$ with $\eta[n] \sim \mathcal{N}(0, 0.25^2)$ and

$$a_1(n) = \begin{cases} 1.2728 & 0 \leq \text{n} < 400, \\ 0 & 400 \leq \text{n} < 800, \\ 1.2728 & 800 \leq \text{n} < 1200 \end{cases} \tag{4.3}$$

The Yule-Walker equations are usually used to estimate the coefficients for a stationary process and it is of interest to try them on a process with changing coefficients. For this we use the matlab `aryule` command. We estimate the coefficients for $AR(2)$ and $AR(4)$. The choice of $AR(2)$ is natural however it was felt that maybe a $AR(4)$ representation could provide more insight considering the changing coefficient. Figure 4.10 shows the estimate PSD from the Yule-Walker equations coefficient estimates. As seen from the periodogram the $AR(4)$ estimate seems to better match the periodogram, however this is because the signal contains those two frequencies at different times (see 4.3) and is not a correct model for predicting future values. Thus we see the weakness of trying to estimate changing coefficients with Yule-Walker equations.



For $AR(2)$ we have as coefficients: `1.0000 0.8641 −0.5867`
For $AR(4)$ we have as coefficients: `1.0000 1.0344 −1.0425 0.5765 −0.4237`
Notice that neither are accurate.

Figure 4.10: `aryule` PSD of $AR(2)$ process

The spectrogram in figure 4.11 shows the effect of a changing coefficient, from the peak of the signal (in red) over time. This shows why the traditional Yule-Walker equations fail due to the changing nature of the signal, which is not consistent with Yule-Walker equations assuming a single nature.

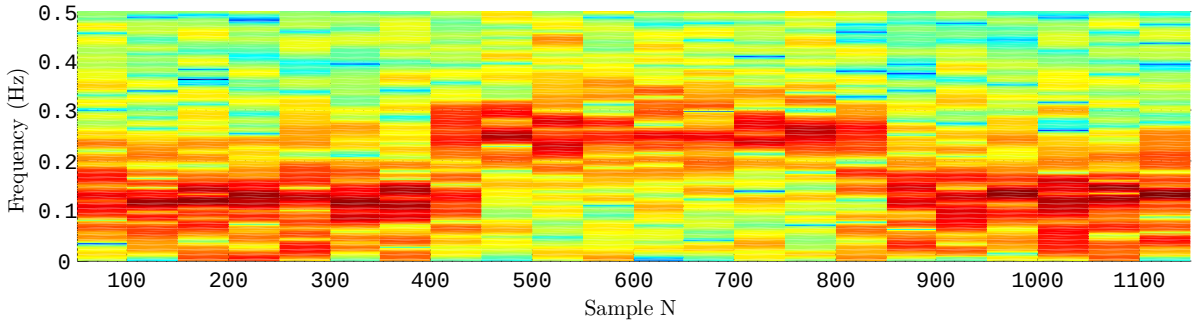


Figure 4.11: Spectrogram

4.3.b RLS estimation

As seen earlier Yule Walker is not a good option when faced with changing estimates. Here the aim is to use a recursive least squares (RLS) algorithm with $\lambda = 0.9$ and for each estimate in time the spectrum is estimated using `freqz`. This gives figure 4.12 which is very similar to the spectrogram in figure 4.11. In fact the RLS spectrogram estimate provides more contrast than the spectrogram and reacts faster to the change in coefficient - though this is also due to the small overlap in the spectrogram. The fast adaptation to changing coefficients is made possible by the low λ value.

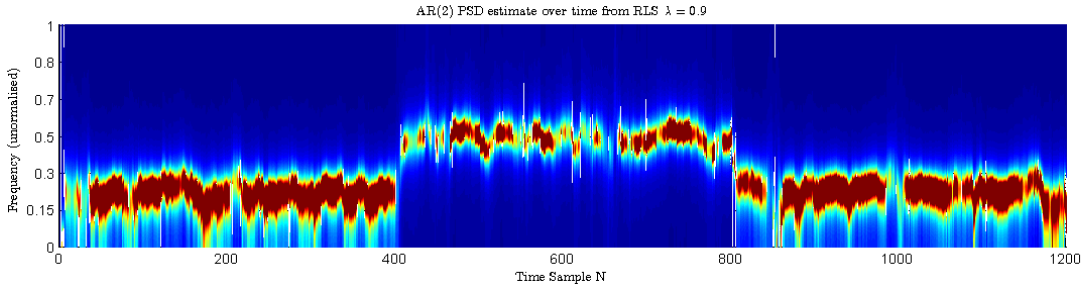


Figure 4.12: Spectrogram estimate using RLS coefficients

Bibliography

[1] Lectures on Fourier Integrals, *Salomon Bochner*, 1959, Princeton University Press

[2] The Rayleigh Quotient Nuno Vasconcelos, ECE Department, UCSD `http://www.svcl.ucsd.edu/courses/ece271B-F09/handouts/Dimensionality3.pdf`