COURSEWORK

IMPERIAL COLLEGE LONDON

DEPARTMENT OF BIOENGINEERING

Adaptive Signal Processing and Machine Intelligence

Author:
Sorenza Bastiaens (CID: 01112757)

Date: April 12, 2019

Contents

1 Classical and Modern Spectrum Estimation **3**
1.1 Properties of Power Spectral Density (PSD) . . . 3
1.2 Periodogram-based Methods Applied to Real-World Data . . . 4
1.3 Correlation Estimation . . . 6
1.4 Spectrum of Autoregressive Processes . . . 8
1.5 Real World Signals: Respiratory Sinus Arrythmia form RR-Intervals . . . 10
1.6 Robust Regression . . . 12

2 Adaptive Signal Processing **14**
2.1 The Least Mean Square (LMS) Algorithm . . . 14
2.2 Adaptive Step Sizes . . . 19
2.3 Adaptive Noise Cancellation . . . 20

3 Widely Linear Filtering and Adaptive Spectrum Estimation **25**
3.1 Complex LMS and Widely Linear Modelling . . . 25
3.2 Adaptive AR Model Based Time-Frequency Estimation . . . 30
3.3 A Real Time Spectrum Analyser Using Least Mean Square . . . 31

4 From LMS to Deep Learning **33**

5 Tensor Decompositions for Big Data Applications **38**

Appendices **39**

A Appendix **39**

1 Classical and Modern Spectrum Estimation

1.1 Properties of Power Spectral Density (PSD)

The PSD is defined by the following equation:

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

The aim is to show analytically that equation (1) is equivalent to the DTFT of the ACF which corresponds to another definition of the PSD under the assumption that the covariance sequence $r(k)$ decays rapidly, such that:

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=-(N-1)}^{N-1} |k||r(k)| = 0 \tag{2}$$

Starting with equation (1), we have:

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

$$P(w) = \lim_{N\to\infty} E\left\{\frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-jnw}\sum_{m=0}^{N-1} x^*(m)e^{jmw}\right\}$$

$$P(w) = \lim_{N\to\infty} \frac{1}{N}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} E\left\{x(n)e^{-jnw}x^*(m)e^{jmw}\right\}$$

$$P(w) = \lim_{N\to\infty} \frac{1}{N}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} E\{x(n)x^*(m)\}e^{-jw(n-m)}$$

The variable k is defined as k=n-m, which gives:

$$P(w) = \lim_{N\to\infty} \frac{1}{N}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} E\{x(n)x^*(n-k)\}e^{-jwk}$$

$$P(w) = \lim_{N\to\infty} \frac{1}{N}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} r(k)e^{-jwk}$$

The double summation is reduced to one summation

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

In conclusion, f(0) appears N times, f(-1) and f(1) appear N-1 times, f(-2) and f(2) appear N-2 times and so on. Finally f(N-1) and f(-(N-1)) appear only once. This is summarized as:

$$\sum_{k=-(N-1)}^{N-1} (N-|k|)f(k) \tag{3}$$

We have then:

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

$$P(w) = \lim_{N\to\infty} \frac{1}{N} \sum_{k=-(N-1)}^{N-1} Nr(k)e^{-jwk} - \lim_{N\to\infty} \frac{1}{N} \sum_{k=-(N-1)}^{N-1} |k|r(k)e^{-jwk}$$

From assumption (2), the right hand side of the equation equates to 0. For the left hand side, the N is put outside the sum and with the N converging to infinity we have:

$$P(w) = \sum_{k=-\infty}^{+\infty} r(k)e^{-jwk} \tag{4}$$

which consists in the DTFT of the ACF.

This equivalence does not hold when assumption (2) is not fulfilled and for a finite number of N. This is due to the fact that computationally it is not possible to implement values of k and N respectively for definition 1 and 2 to tend to infinity . Therefore, the PSDs plotted on MATLAB are only approximations (estimates) of the true PSD resulting in variations between the two definitions for small values of N and k. The following figure is the PSD of two sine functions added together of frequency 10Hz and 20Hz. In this case the number of samples is 200. The first definition presents ripples and the second definition only presents the peaks.

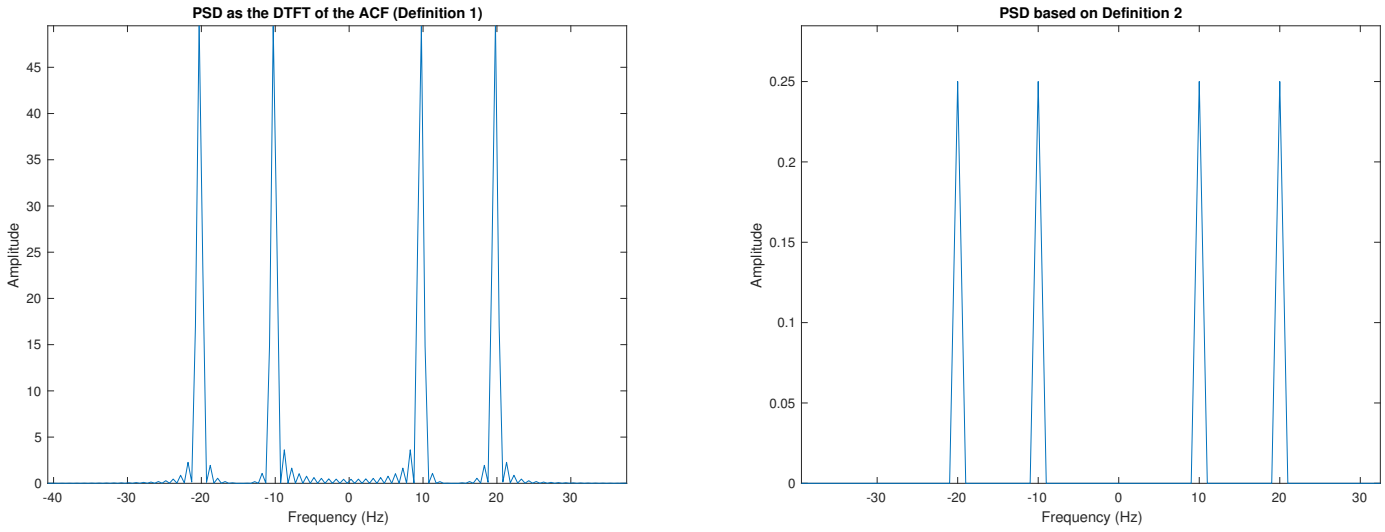

Figure 1: PSD of the function sin(2π20t)+ sin(2π10t) using two different methods

1.2 Periodogram-based Methods Applied to Real-World Data

a) The periodogram of the sunspot time series is estimated for the original data, without the mean and without the trend. This is presented on the left of Figure 2.
Without the mean, the periodogram is equal to 0 at frequency 0 cycle/year. This indicates that subtraction of the mean results in attenuation of the DC component. The aim of using detrend is to remove the linear trend of a function. Therefore, not much difference is observed between the original data and the one without trend as no linear trend is visible. On Figure 2, only removing the mean is leads to on observable

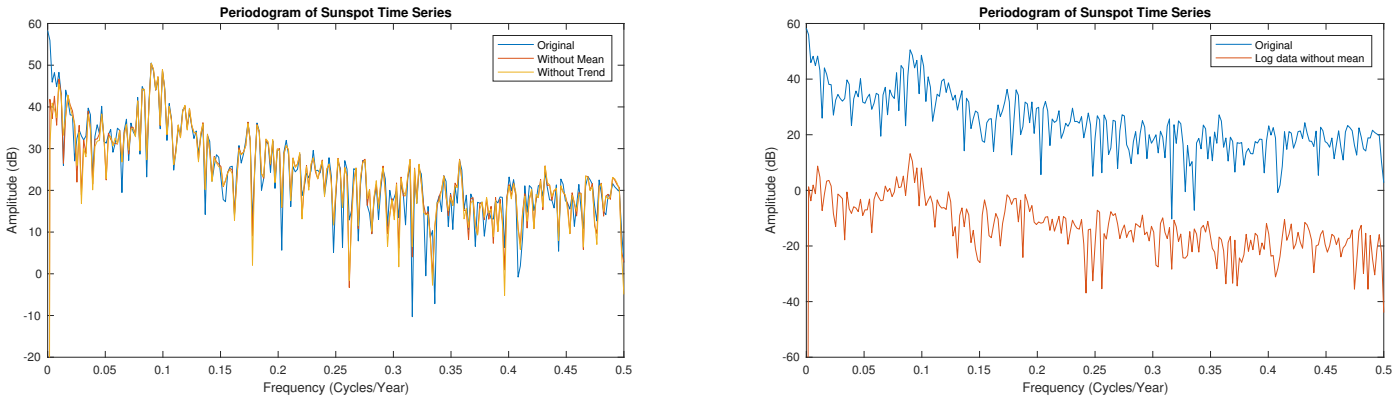

Figure 2: Left: Periodogram of the original Sunspot Time Series, without mean and trend, Right: Periodogram of the logarithm of the data and without the mean

effect at 0 cycle/year. For other frequency values, the three periodograms are very similar and the peak at 0.1 cycle/year is present in all of them. This indicates a period of 10 years.
Applying the logarithm centers the data around 0dB and a somewhat smoother periodogram is observed. The peak at 0.1 cycle/year is then easier to identify. Subtracting the mean from this logarithmic data perform the same function as before, which is removing the DC component.

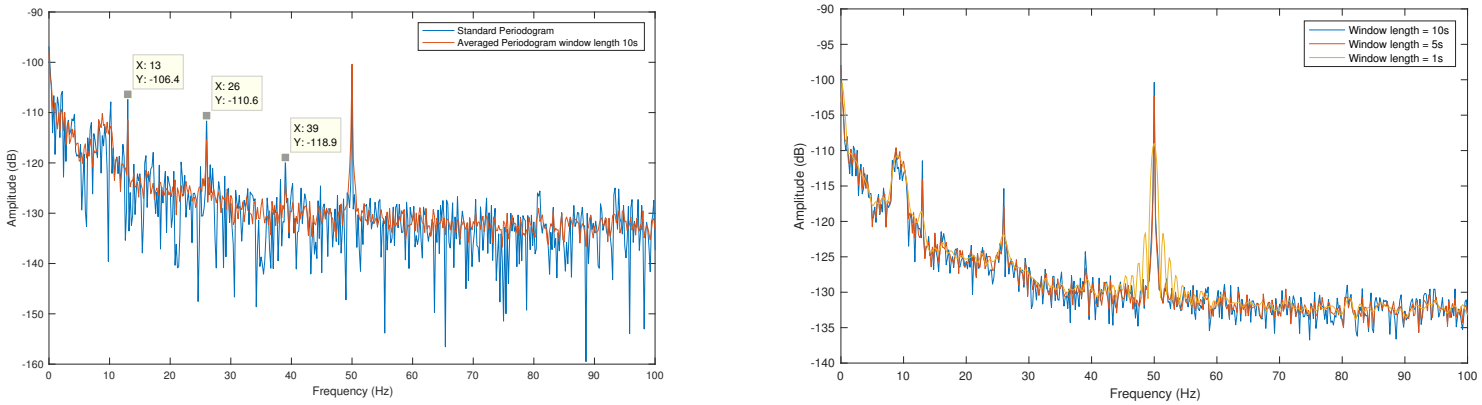

Figure 3: Left: Periodogram of EEG signal with indication of peaks corresponding to the SSVEP; Right: Average periodograms with different window length

b) The steady state visual evoked potential (SSVEP) is the response in the EEG at the same frequency as the flashing visual stimulus. From Figure 3, the fundamental frequency of the SSVEP is observed at 13 Hz. Two harmonics are also present at integer multiples of 13, which are 26Hz and 39Hz. The peak of large amplitude at 50Hz corresponds to the power-line interference. This is the reason why the third harmonic at 52Hz is not identifiable in the periodogram, it is hidden by the interference. The wider peak between 8-10Hz are the alpha-rythm.
With the standard periodogram or the averaged periodogram, the SSVEP peaks are clearly identified. The difference between the two resides in the variance. Indeed, the averaged periodogram has considerably less variability than the standard periodogram. However, the resolution in the averaged periodogram is also reduced since the SSVEP peaks are of smaller amplitude.
As the window length increases the average periodogram has less and less variability. A very small window size, 1s, produces a very smooth and continuous signal with no visible peaks, only a general trend with lobes. The only peak represented as many different lobes is the interference, which is not of interest. A small window size decreases the resolution of closely spaced narrow band components and their

amplitude. The details in the spectrum are averaged out and thus peaks of interest (SSVEP) are not observable anymore. Therefore, there is a trade-off between variability and resolution of the spectrum with the averaged periodogram.

1.3 Correlation Estimation

The code to estimate the unbiased and biased ACF to compute the correlogram is validated with a white Gaussian noise (WGN), a signal made of two sinusoids and random WGN noise and filtered WGN.

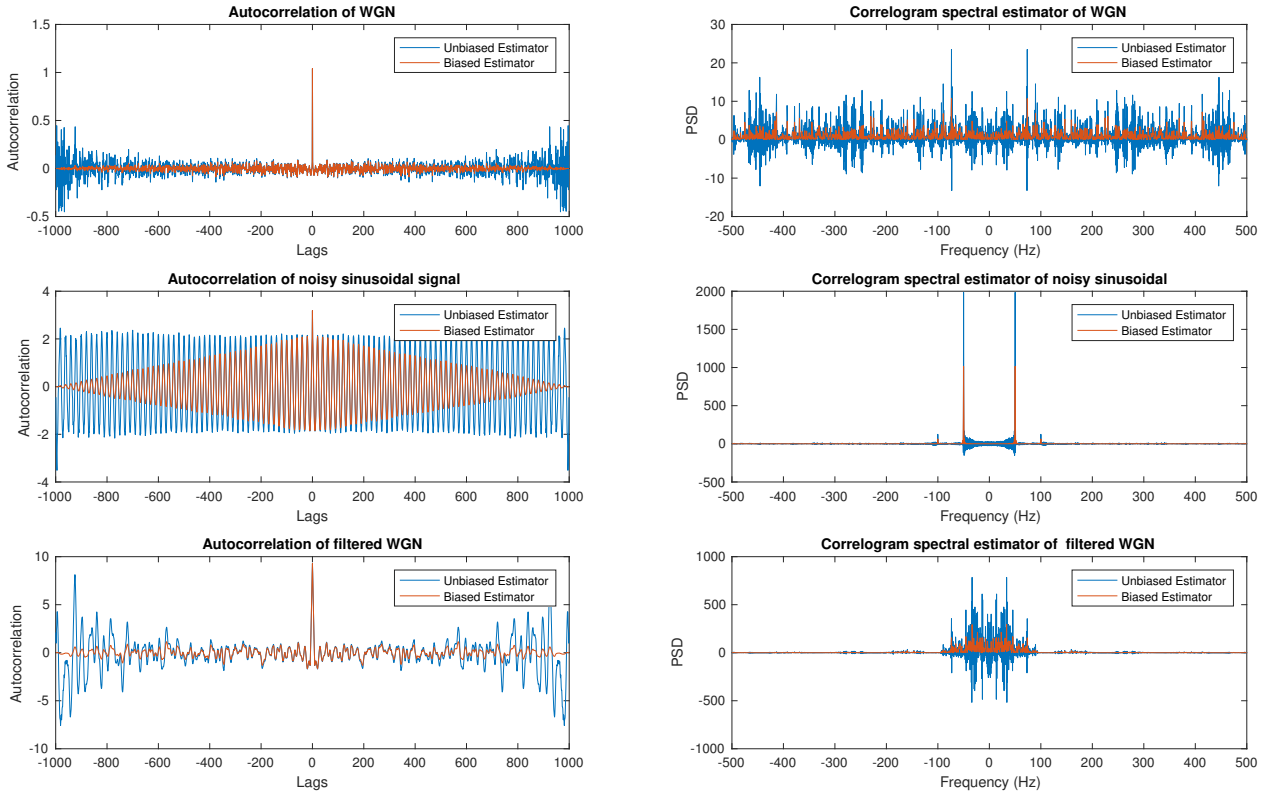

Figure 4: PSD of 3 different signals based on the correlogram spectral estimator with biased and unbiased estimators of the Autocorrelation function

a) Theoretically, the unbiased estimator is more accurate than the biased estimator as its mean matches the true PSD. However, as we can see on Figure 4 where the autocorrelation of WGN or filtered WGN is represented, for large values of k (lags) the ACF becomes more variable and inaccurate as it has fewer samples and diverges. The results is an over representation of small segments resulting in a miss representation of the data. This induces more variability in the PSD estimate but more importantly it leads to the presence of negative values which is not possible for power. For the biased estimator, as k increases the ACF tends towards 0. The signal is shifted over itself. Therefore, as it moves away from the signal the ACF decreases. The maximum amplitude is always encountered at 0 lag. Even though it will never be exactly equal to the true PSD, the biased PSD has less variability than the unbiased one and only generates positive values.

b) Several realisations of a signal composed of two sinusoids of frequency 50Hz and 100Hz corrupted by noise are generated and their individual PSD is estimated. The mean of the PSDs is also determined. From Figure 5 , the disparity of the different realisations is quite visible due to the variance induced by random noise added in each realisation. This variability is around 150 in amplitude. The peaks of interest in the

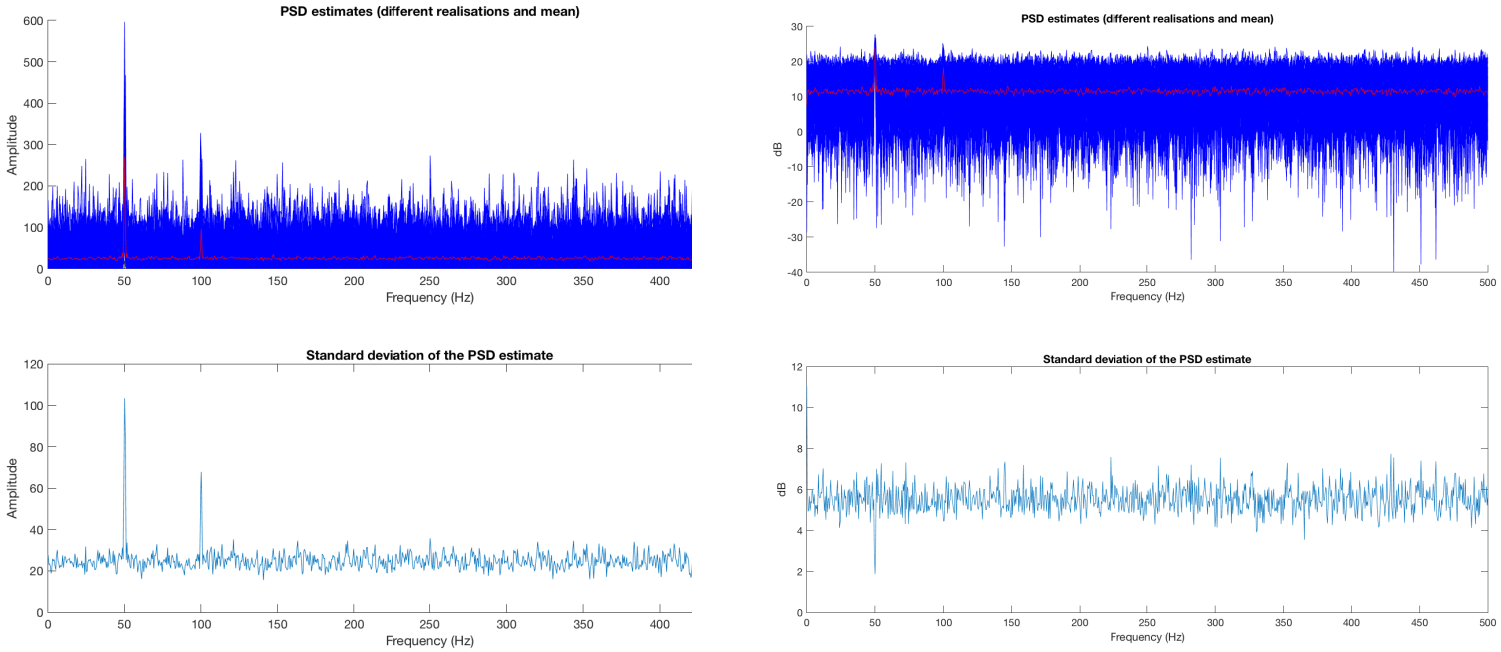

Figure 5: On the left, PSD estimates of 100 different realisations of two sinusoids of frequency 50Hz and 100Hz with WGN. On the right, it is the same only in dB

PSD are then somewhat hidden under the noise and are not always easily identifiable as it is the case here for the 100Hz sinusoid. If the noise was even more present, the signal and the noise would have the same amplitude. However, if the mean between the signals generated is computed (red line in Figure 5), the noise is considerably reduced, revealing the peaks at 50Hz and 100HZ.
The plot of the standard deviation (std) demonstrates the fact that the std variance is proportional to the PSD. It reveals that many realisations are required to gain an accurate PSD estimate.

c) dB is a logarithmic scale which leads to an increased variability around small values and a decreased variance around higher values. This means that the peaks of interest are less visible and blend with the noise, as it is shown on the right side of Figure 5. The standard deviation is roughly constant in dB indicating that the spread between at the peaks between the different PSD realisations is less prominent than in a non-logarithmic scale. This indicated that with one realisation, the PSD estimate value will be more accurate in dB than in a non-logarithmic scale. The main advantage of the logarithmic scale is the contraction property as it enables the comparison of the entire spectrum within a smaller range.

d) When the number of samples in the signal is insufficient, the periodogram might have insufficient resolution as it is proportional to 1/N. Figure 6 on the next page, illustrates this fact. When the number of data samples is equal or bigger than 50, the periodogram shows the correct two line spectra as the difference between 0.3Hz and 0.32Hz is of 0.02Hz which corresponds to 1/50. If N is smaller, the resolution is too low to distinguish between the two and one lobe is observed instead.

e) The MUSIC (Multiple Signal Classification) algorithm estimates the frequency content of an autocorrelation matrix using an eigenspace method. The function `pmusic` in MATLAB implements this algorithm returning a Pseudospectrum. To use the function, the input requires a rectangular array for which each row of x represents a separate observation of the signal such that x'*x is an estimate of the correlation matrix. This is achieved with `corrmtx` since the output is a biased estimate of the autocorrelation matrix for vector x. The output X is a rectangular Toeplitz matrix X, such that X'X. The output of interest is R as

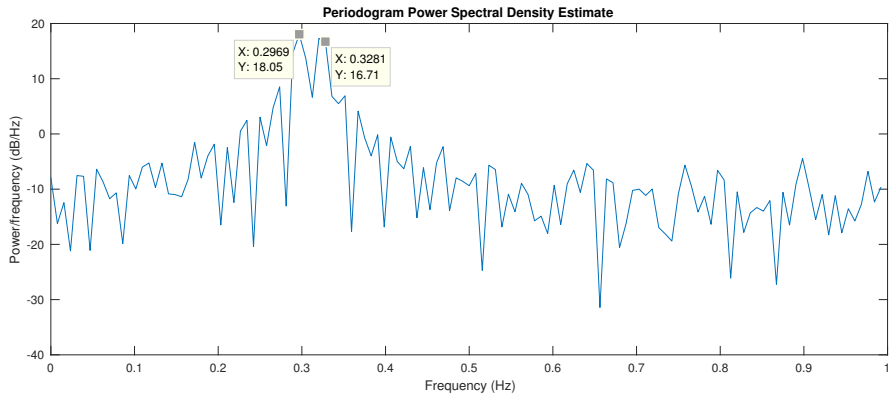

Figure 6: PSD estimates in dB of 100 different realisations of two sinusoids of frequency 50Hz and 100Hz with WGN

the estimate is calculated as X'*X. The choice of getting a modified rectangular Toeplitz matrix is that generates an autocorrelation estimate derived using forward and backward prediction error estimates, based on an a certain order prediction error model, in this case 14. In `pmusic` the second input to the function corresponds to the signal subspace dimension which means the number of peaks that is expected to be identified. The fourth input 1 is the sampling frequency and finally the 'corr' precision is in order to force the input argument x to be interpreted as a correlation matrix rather than matrix of signal data which is the case here.
The advantage of the MUSIC algorithm is that it outperforms the periodogram in the presence of noise, when the number of components is known in advance, as it uses that knowledge to remove the remaining noise producing a smooth function with only the peaks of interest. The disadvantage of this method, even if it has high precision, is that the number of components need to be known in advance.
The MUSIC algorithm does not provide more detailed information of the signal but more precise results as it highlights the components and the rest is set to zero. The variance of the MUSIC algorithm is considerably smaller than the periodogram. However, the periodogram is less biased as the additive noise is still visible whereas the Pseudospectrum does not translate the actual signal but only the signal without white noise. In general the accuracy of the MUSIC method would be high for a general spectrum estimate if the noise is white and uncorrelated with the rest of the data.

1.4 Spectrum of Autoregressive Processes

a) One of the shortcomings of using the unbiased ACF estimate when finding the AR parameters is the fact that the ACF may not be positive definite as mentioned previously, leading to a possible singular $\mathbf{R_{xx}}$ matrix which corresponds to the autocorrelation matrix. This is an issue as to determine the parameters, the following equation needs to be valid:

$$\mathbf{r_x} = \mathbf{R_{xx}}\mathbf{a} \Leftrightarrow \mathbf{a} = \mathbf{R_{xx}^{-1}}\mathbf{r_x} \tag{5}$$

This is only relevant if $\mathbf{R_{xx}}$ is invertible meaning a definite non-singular matrix.
Furthermore, for large lags k close to N, the unbiased ACF has a large variance as is has fewer data points.

b) The power spectrum density of an autoregressive process with parameters a1 = 2.76, a2 = -3.81, a3 = 2.65 and a4 = -0.92 and in gaussian white noise is estimated using model orders (p) ranging from 2 to 14. From Figure 7, for p smaller or equal to 4, the PSD presents only one peak instead of the expected two peaks observed with the ideal PSD. From p=6, the two peaks at different frequencies are distinguishable. Starting from p=8, the error is considerably reduced and closely resembles the behaviour of the true PSD.

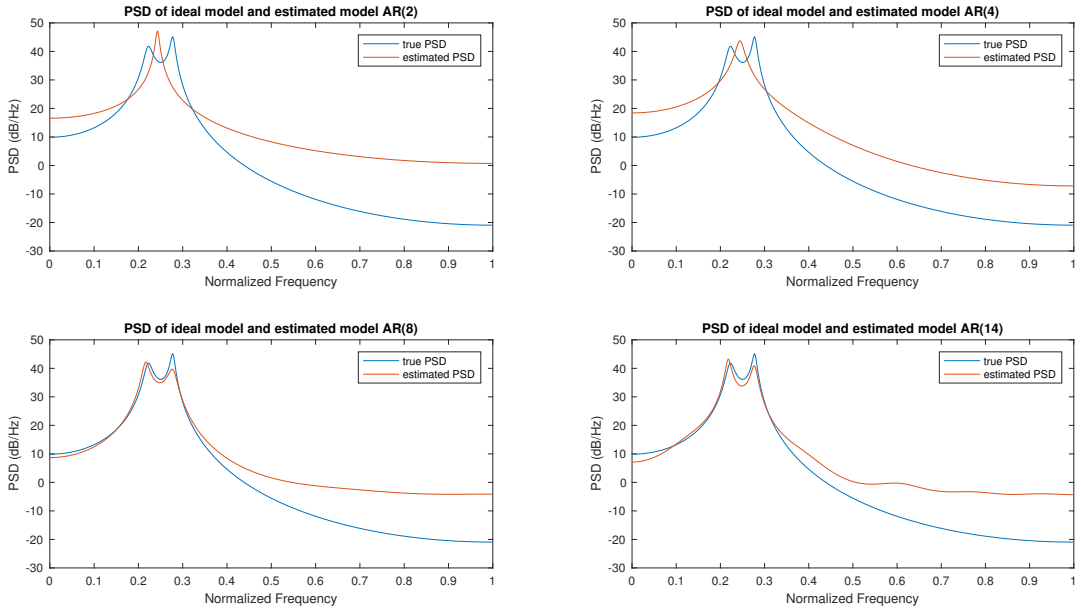

Figure 7: Power spectrum density of the signal for different model orders

Additionally, due to the small number of sample available to estimate the PSD (500 samples) , the fitted model order 4 does not perform well even though the original signal is an AR(4) process. To solve the issue, the number of samples is increased which is done in the next section.

c) For data length 10,000 samples, when the chosen model order is lower than 4, only one peak is observed, thus incorrectly translating the behaviour of the process. Under-modelling leads to an incorrect modelling of the AR leading to the loss of information. For higher order models, the PSD has two peaks at the correct frequencies. The estimated PSD of model order higher or equal to 5 seem to fit the true PSD more accurately. However, higher order model can overfit the data, meaning it will start describing the random error. To choose a model order, different criterion can be used which takes this issue of over-modelling into consideration such as the AIC or MDL criterions. Higher order models are also avoided as they add complexity to the model.

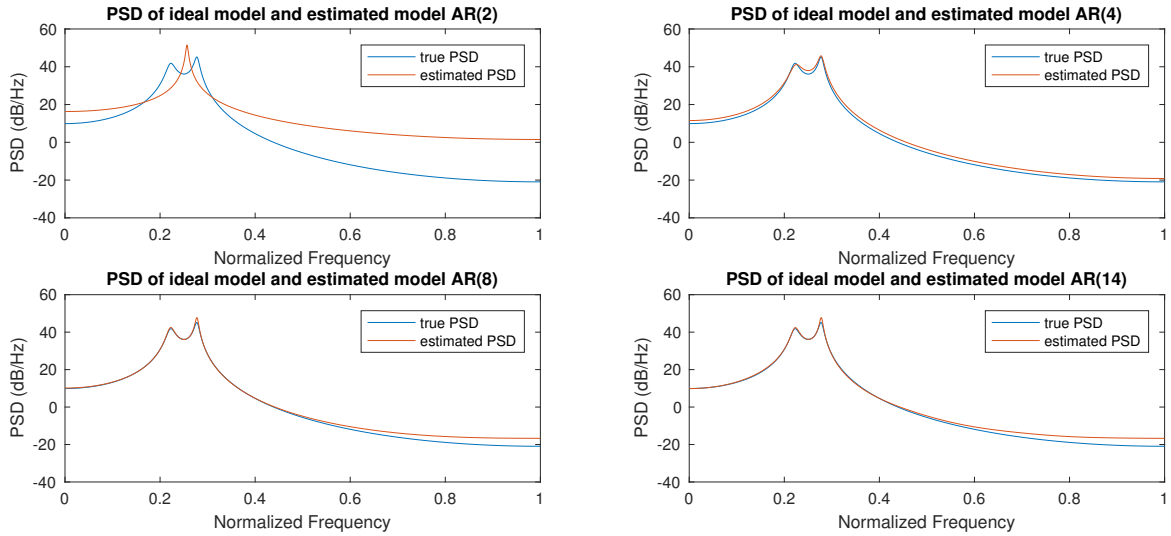

Figure 8: Power spectrum density of the signal for different model orders

1.5 Real World Signals: Respiratory Sinus Arrythmia form RR-Intervals

a) The PSD of three different RRI intervals is computed. For the average periodogram, a hanning window is chosen instead of a rectangular window to yield finer results. Trial 1, trial 2 and trial 3 respectively correspond to normal breathing, fast breathing and slow breathing.

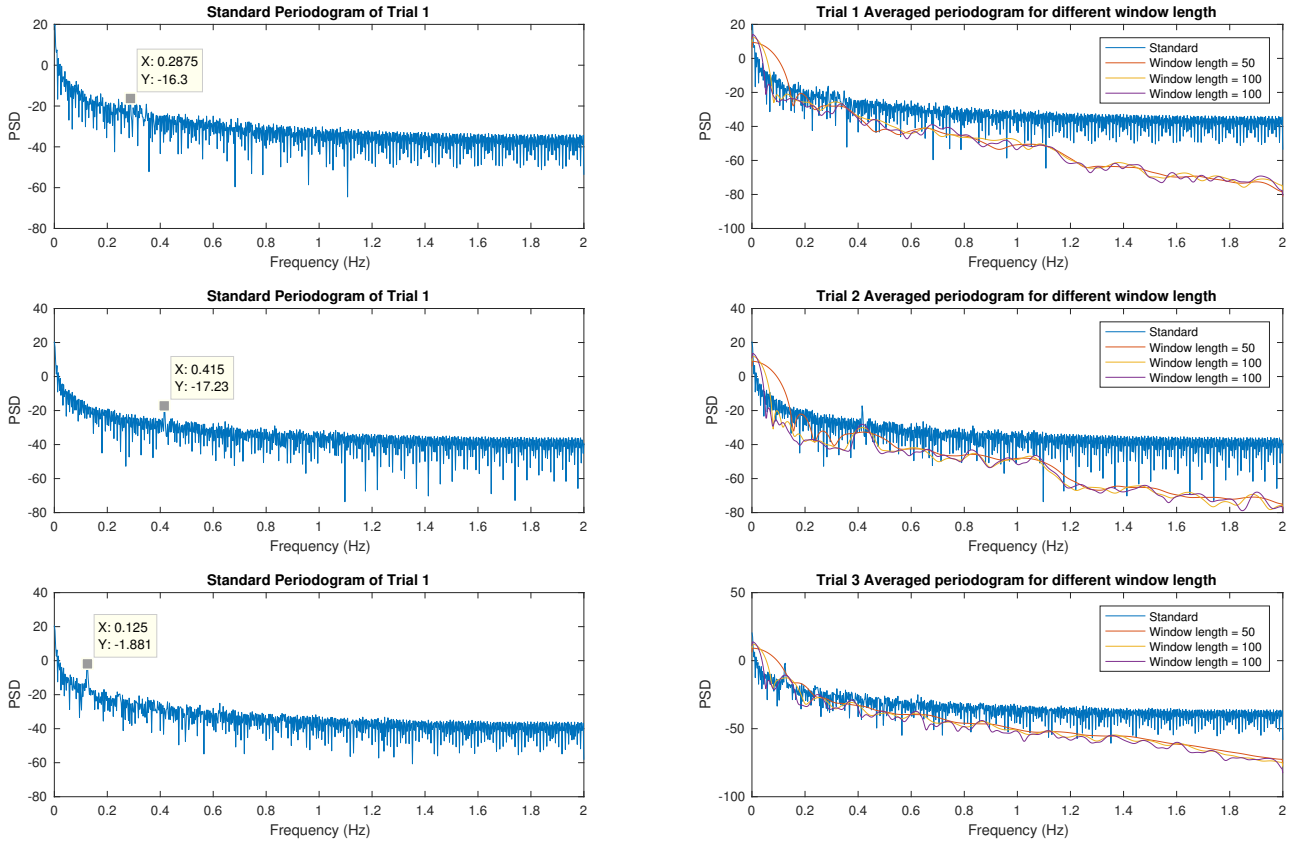

Figure 9: Standard and Averaged periodogram of three different RRI intervals trials

Even though the averaged periodogram has less variability, the peaks are not always captured depending on the window length. For example, in trial 3 with window length 50, the first lobe englobes the peak and fails to show the respiration frequency. With the other window lengths an enlarged peak is observed in the same area as the peak in the standard periodogram. In the case of trial 1, the standard periodogram fails to clearly present a peak and thus the averaged periodogram shows the a range of frequencies instead of a peak resulting in less precision. The standard periodogram is more accurate in this case.

b) Between the three trials, the difference lies in the position of the peak. Indeed, depending on the breathing rate the frequency location of the peak will change. For trial 1 (where the peak is the least identifiable), an approximation of 17.25 breaths per minute is calculated (BPM). For trial 2, the peak is at 0.415Hz which corresponds to 24.9 BPM. Finally, for slow breathing the BPM is 7.5 since the peak is observed at 0.125Hz. These values are close to the expected value as for fast breathing, the rate is 25BPM and for slow breathing 7BPM.

c) The aim is to determine an optimal AR model that would fit the peaks. In other words, the AR spectrum estimate is plotted with the standard periodogram to facilitate comparison and determine at which AR model order a peak is approximately observed at the theoretical respiration rate for each trials.

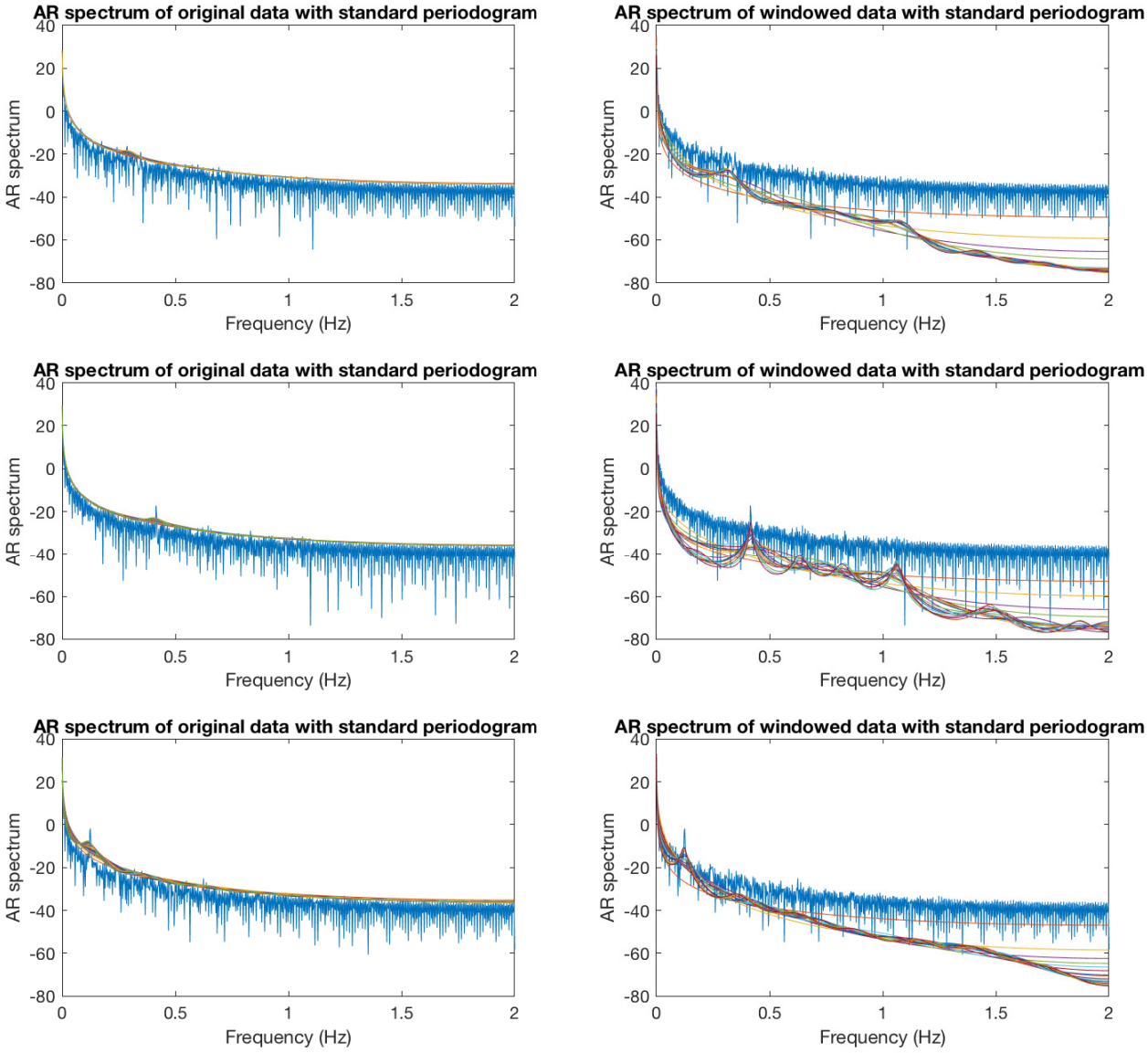

Figure 10: AR spectrum with standard periodogram based on the original data (left) for 40 model orders and the windowed data (right) for the three RRI trials for 20 model orders

On the left side of Figure 10, the AR spectrum estimated with the original data is shown up to model order 40. The model order for trial 1, trial 2 and trial 3 before a peak is observed are respectively 30, 32 and 25. In the case of trial 1 a peak is not really distinguishable but more of a lobe. These are quite high order values. Thus, on the right side of Figure 10, is the AR spectrum estimated with the hanning windowed data up to 20 model order. As we can see lower model orders are then possible as peaks already appear for model orders equal to 10.

The AR spectrum is smoother than the standard periodogram. However very high order models are required for peak detection which involves high complexity and computational load. Indeed, many coefficients are necessary especially with the original data to model to viex respiration frequency accurately. Compared to the averaged periodogram, the AR spectrum does not present the issue of main and side lobes. In conclusion, the AR spectrum is efficient if the peak clearly stands out like in trial 3 and if the data is preprocessed in a way. Otherwise, the AR model requires too many coefficients to be accurate enough and the standard periodogram is sufficient to find the right frequency like in trial 2.

1.6 Robust Regression

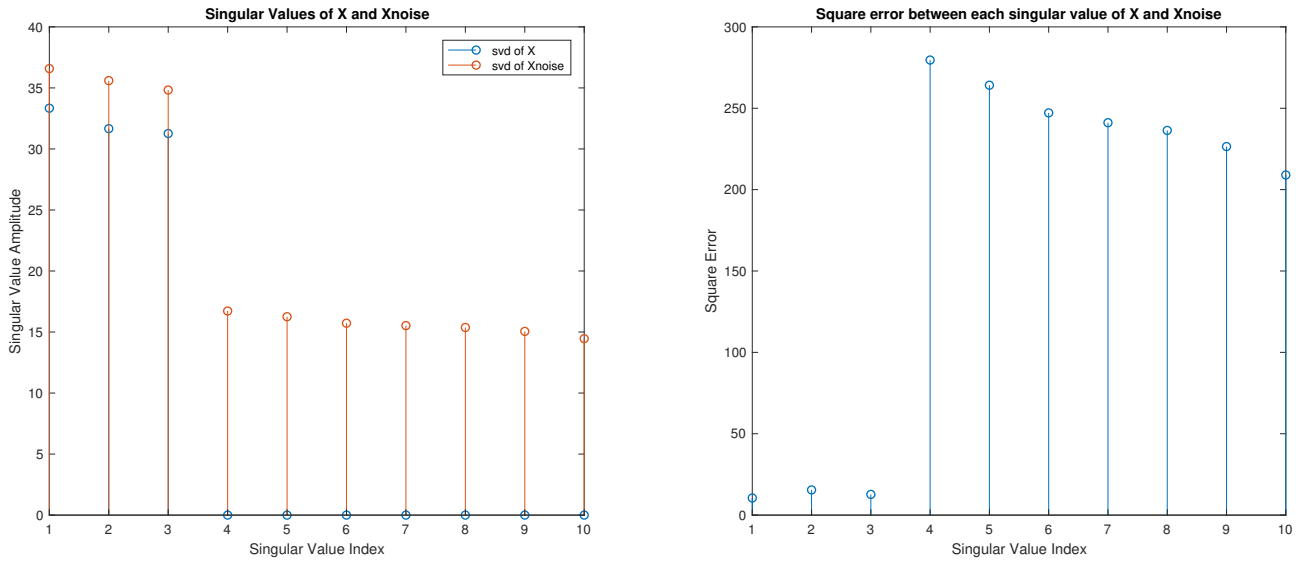

Figure 11: Left: Singular Values of Matrices X and Xnoise; Right: Square error between each singular value of X and Xnoise

a) The singular values of both matrices are plotted in descending order in Figure 11. The singular values of a matrix $\mathbf{X}$ correspond to the square root of the eigenvalues of $\mathbf{X^T X}$. The rank of any square matrix equals the number of nonzero eigenvalues with repetition. Thus, the number of nonzero singular values of $\mathbf{X}$ equals the rank of $\mathbf{X^T X}$. Following the rank-nullity theorem and the fact that $\mathbf{X}$ and $X^T X$ have the same kernel, this means that they also have the same rank. Therefore, the number of nonzero singular values of $\mathbf{X}$ is equal to the rank of the matrix. From Figure 11, $\mathbf{X}$ only had 3 nonzero singular values, which indicates that the rank of the input data $\mathbf{X}$ is equal to 3.
For $\mathbf{X_{noise}}$, the 3 first singular values are of slightly higher amplitude than in the case of $\mathbf{X}$ due to noise corruption. The remaining singular values of $\mathbf{X_{noise}}$ are approximately half the amplitude. These span the noise subspace whereas the initial 3 span the signal subspace.
The square error is relatively small for the 3 first singular values. However, the error is large for the other values as for $\mathbf{X}$, the singular values are equal to 0 which is not the case for $\mathbf{X_{noise}}$. The noise then adds singular values amplitude. In this case, the rank is still distinguishable as the singular values in the noise subspace are half the magnitude and a thresholding process is sufficient. The identification of rank becomes hard if the amount of noise increase and the singular values in the noise and signal subspace are of equal magnitude.

b) To estimate the difference between the variables (columns) of the matrices, the mean error between the absolute value of the difference between the input matrix and either the denoised matrix or the corrupted matrix squared in each column is calculated and presented on the left of Figure 12. To create the low-rank approximation of $\mathbf{X_{noise}}$, the value of r is equal to 3 as to only have the most significant principal components. For every variables, the error between $\mathbf{X}$ and $\tilde{\mathbf{X}}_{\mathbf{noise}}$ is smaller than the error between $\mathbf{X}$ and $\mathbf{X_{noise}}$. This shows the effect of the PCA method to only take the significant component and remove some of the added white Gaussian error. It seems to work well for every variable, expect variable 4 which has 0.2 error. This could be explained by the fact that some information still remains in the other components, which has been eliminated with the noise in the low-rank approximation.

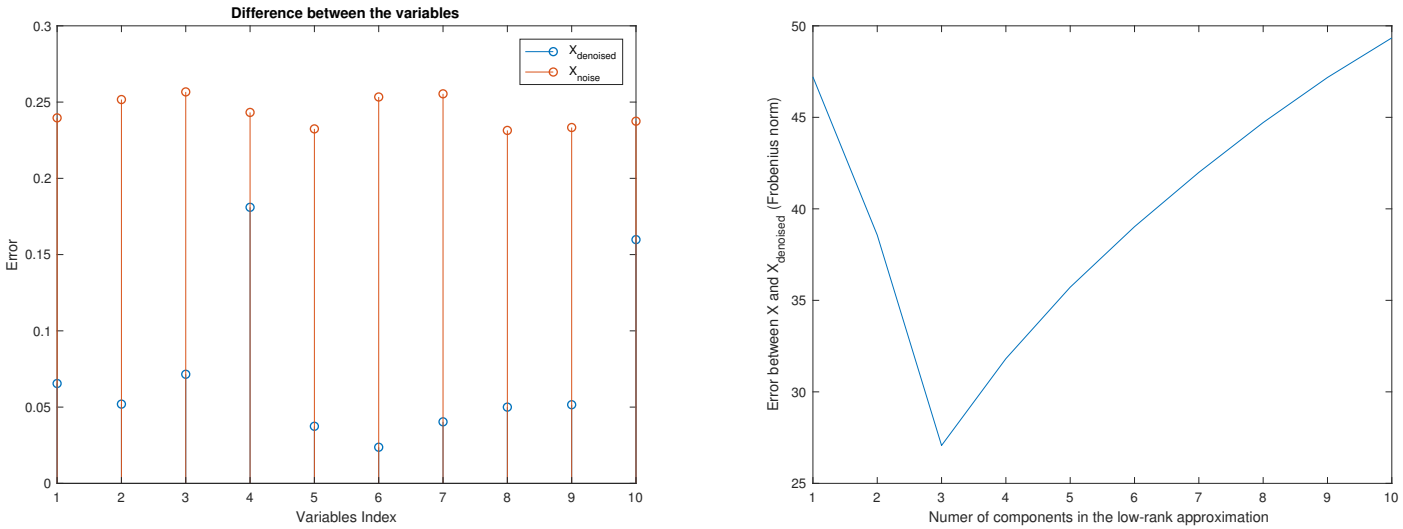

Figure 12: Left: Difference between the variables of $\mathbf{X}$ and $\tilde{\mathbf{X}}_{\mathbf{noise}}$ and $\mathbf{X}_{\mathbf{noise}}$; Right Error between $\mathbf{X}$ and $\tilde{\mathbf{X}}_{\mathbf{noise}}$ for different k values

Another way to calculate the error is with the Frobenius norm to determine the 2-norm of the column vector. The error between $\mathbf{X}$ and the noise corrupted matrix $\mathbf{X}_{\mathbf{noise}}$ is found to be around 49.34 whereas the difference between $\mathbf{X}$ and the denoised matrix is only 27.01.
This shows that the denoising operation with the creation of a low rank approximation $\mathbf{X}_{\mathbf{noise}}$ only using the most significant principal components is able to reflect the initial data fairly adequately. Indeed, on the right of Figure 12, the error for different low rank approximation is shown. The error is minimized when r is equal to 3, highlighting the efficacy of the principal component analysis method to mimic the input data from a noise corrupted input matrix. The remaining error is explained by the fact that the principal component analysis (PCA) relies on the assumption that the non significant principal components are purely noise which is not always the case.

c) The ordinary least squares (OLS) estimates for the unknown regression matrix $\mathbf{B}$ is determined by using the noisy data $\mathbf{X}_{\mathbf{noise}}$ as it is full-rank. However, this might lead to spurious correlations in the calculation of regression coefficients and is based on noisy data. To solve these issues, the other method that exists is the principal component regression (PCR) which performs an initial PCA and thus a low rank approximation matrix $\tilde{\mathbf{X}}_{\mathbf{noise}}$ is used instead to determine the regression coefficient. The aim is to assess the efficacy of each method on the actual output and then the test-set.
The value of r that should be retained is 3, as lower values would be insufficient and higher values would just add noise. However, in order to present the data, the error is estimated for different r values.

The error between $\mathbf{Y}$ and $\hat{\mathbf{Y}}_{\mathbf{OLS}}$ is approximately equal to 59.59. In the case of $\hat{\mathbf{Y}}_{\mathbf{PCR}}$, as expected, the error is high for $r < 3$ and then constant at approximately 59.81. Therefore, the OLS method seem to perform better than PCR by 0.37%. This difference could be explained by the fact that $\mathbf{Y} = \mathbf{XB} + \mathbf{N_Y}$ where $\mathbf{N_Y}$ is zero-mean Gaussian noise. The corrupted matrix $\mathbf{X}_{\mathbf{noise}}$ also has zero-mean Gaussian noise. This menans that the OLS estimate might slightly be more accurate as it has the same added noise whereas with PCR the noise is reduced and would then remove some of the zero-mean Gaussian noise and be less similar than $\mathbf{Y}$.
When the same process is done on the data from the test-set using the regression coefficients determined previously, the performance changes. The error between $\mathbf{Y}_{\mathbf{test}}$ and $\hat{\mathbf{Y}}_{\mathbf{test-OLS}}$ is estimated at 48.76. As before, for values of $r < 3$ the estimated error is high when compared to $\hat{\mathbf{Y}}_{\mathbf{test-PCR}}$ as the rank is insufficient. However, for $r >= 3$ this time the $\hat{\mathbf{B}}_{\mathbf{PCR}}$ seems to perform slightly better than $\hat{\mathbf{B}}_{\mathbf{OLS}}$ as it is approximately equal to 48.18. Thus this time the PCR regression coefficient is closer to the test data result by 1.2%.

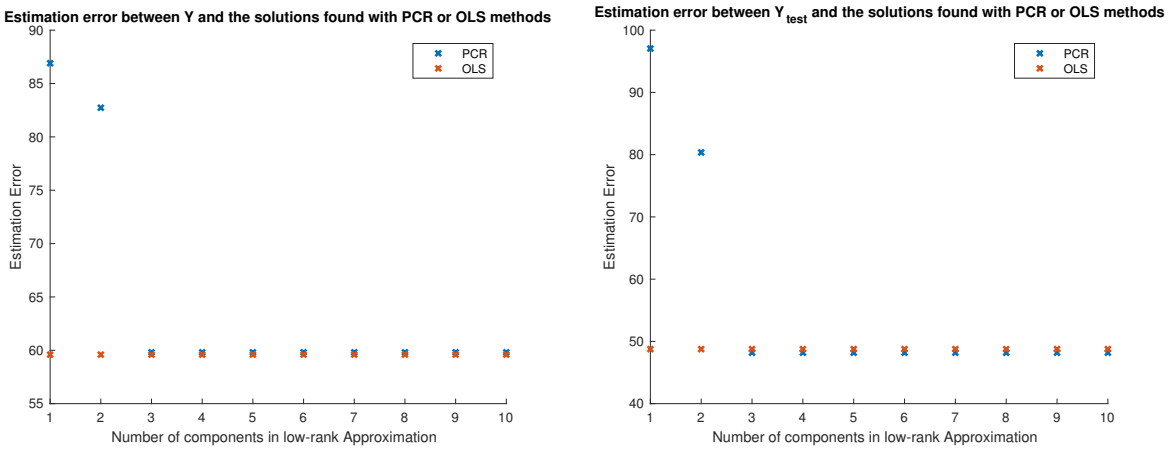

Figure 13: Left: Estimation error between $\mathbf{Y}$ and $\hat{\mathbf{Y}}_{\mathbf{OLS}}$ and $\hat{\mathbf{Y}}_{\mathbf{PCR}}$ for different k; Right: Estimation error between $\mathbf{Y}_{\mathbf{test}}$ and $\hat{\mathbf{Y}}_{\mathbf{test-OLS}}$ and $\hat{\mathbf{Y}}_{\mathbf{test-PCR}}$ for different k

This might be due to the fact that the PCA conserve the maximum amount of information without noise, whereas the OLS method determines a coefficent based on the corrupted matrix, keeping this information. Therefore, the PCA method will be more general and perform better on test data.

d) To assess the effectiveness of PCR compared to OLS solution, the estimated regression coefficients are tested over an ensemble of test data. The mean square error is estimated on MATLAB with the function immse to quantify the prediction errors of each methods. The result for the PCR method (using 3 significant principal components) and OLS methods are respectively 0.46 and 0.49. Thus, the PCR method is approximately 6% more effective than the OLS one on the test data.

2 Adaptive Signal Processing

2.1 The Least Mean Square (LMS) Algorithm

In this section the LMS algorithm is studied to estimate the parameters of an AR(2) process.

a) An AR(2) process is stationary if the solutions for z in the equation $1-a_1z-a_2z^2=0$ are outside the unit circle. Another to prove stationarity is to show stability as stability implies stationarity. The conditions of an AR(2) process are the following:

$$\begin{cases} a_1 + a_2 < 1 \\ a_2 - a_1 < 1 \\ -1 < a_2 < 1 \end{cases} \tag{6}$$

See Appendix A for the derivation to obtain these equations. These conditions are fulfilled for $a_1 = 0.1$ and $a_2 = 0.8$. Therefore, the process is stable and wide sense stationary. The correlation matrix of the input vector $\mathbf{x}(n) = [x(n-1), x(n-2)]^T$ is defined in the following equation.

$$R_{xx} = \mathbb{E}\{\mathbf{x}(n)\mathbf{x}^T\} = \begin{bmatrix} x(n-1)x(n-1) & x(n-1)x(n-2) \\ x(n-1)x(n-2) & x(n-2)x(n-2) \end{bmatrix} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(1) & r_{xx}(0) \end{bmatrix} \tag{7}$$

where $r_{xx}(k)$ corresponds to the autocorrelation of x(n) at lag k. Therefore, the autocorrelation function (ACF) needs to be determined to solve the correlation matrix. For an AR process, the ACF is obtained by first multiplying the equation defining the process by $x(n-k)$. We have then:

$$x(n-k)x(n) = a_1x(n-1)x(n-k) + a_2x(n-2)x(n-k) + x(n-k)\eta(n) \tag{8}$$

The ACF is the expectation of the left term which corresponds to the expectation of each summed term on the right side of the equation. When $k > 0$, the term $\mathbb{E}\{x(n-k)\eta(n)\}$ vanishes. For a general AR(k) process, the following equations are valid:

$$\begin{aligned} r_{xx}(0) &= a_1 r_{xx}(1) + a_2 r_{xx}(2) + ... + a_p r_{xx}(p) + \sigma_\eta^2 \qquad k = 0 \\ r_{xx}(k) &= a_1 r_{xx}(k-1) + a_2 r_{xx}(k-2) + ... + a_p r_{xx}(k-p) \qquad k > 0 \end{aligned} \tag{9}$$

Due to the symmetry property of the ACF, we have $r_{xx}(-k) = r_{xx}(k)$. For an AR(2) process, the following three equations are deduced, taking the symmetry property into account:

$$\begin{aligned} r_{xx}(0) &= a_1 r_{xx}(1) + a_2 r_{xx}(2) + \sigma_\eta^2 \\ r_{xx}(1) &= a_1 r_{xx}(0) + a_2 r_{xx}(1) \\ r_{xx}(2) &= a_1 r_{xx}(1) + a_2 r_{xx}(0) \end{aligned} \tag{10}$$

By substituting $a_1 = 0.1$, $a_2 = 0.8$ and σ_η^2, we get from the 2nd equation of above:

$$r_{xx}(1) = 0.1 r_{xx}(0) + 0.8 r_{xx}(1) \Leftrightarrow r_{xx}(1)(1-0.8) = 0.1 r_{xx}(0) \Leftrightarrow r_{xx}(0) = 2 r_{xx}(1) \tag{11}$$

Using equation (10), we can define $r_{xx}(2)$ depending on $r_{xx}(1)$:

$$r_{xx}(2) = 0.1 r_{xx}(1) + 0.8 \times 2 r_{xx}(1) \Leftrightarrow r_{xx}(2) = (0.1 + 1.6) r_{xx}(1) \Leftrightarrow r_{xx}(2) = 1.7 r_{xx}(1) \tag{12}$$

Implementing all the equivalencies in the first equation, enables the identification of $r_{xx}(1)$:

$$2 r_{xx}(1) = 0.1 r_{xx}(1) + 1.36 r_{xx}(1) + 0.25 \Leftrightarrow 0.54 r_{xx}(1) = 0.25 \Leftrightarrow r_{xx}(1) = \frac{25}{54} \tag{13}$$

Therefore, $r_{xx}(0) = \frac{25}{27}$ and $r_{xx}(2) = \frac{85}{108}$ and the correlation matrix of $\mathbf{x}(n)$ corresponds to:

$$R_{xx} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(1) & r_{xx}(0) \end{bmatrix} = \frac{25}{54} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{14}$$

The condition of convergence in the mean of the weight vector in LMS algorithm is

$$0 < \mu < \frac{2}{\lambda_{max}} \tag{15}$$

where λ_{max} is the largest eigenvalue of the autocorrelation matrix defined above. To determine the range of values of the step size μ when the LMS converges in the mean, λ_{max} is determined by calculating the eigenvalues of $\mathbf{R_{xx}}$. To facilitate calculations the eigenvalues of the matrix [2 1; 1 2] is computed and then multiplied by $\frac{25}{54}$. Therefore, by eigendecompostion we have:

$$(2-\lambda)(2-\lambda) - 1 = 0 \Leftrightarrow \lambda^2 - 4\lambda + 3 = 0 \Leftrightarrow (\lambda - 3)(\lambda - 1) = 0 \tag{16}$$

The largest eigenvalues of the small matrix is 3 and for $\mathbf{R_{xx}}$, the largest eigenvalue is then equal to $3 \times \frac{25}{54} = \approx 1.389$. The range of values for μ is $0 < \mu < 1.439$.

b) An LMS adaptive predictor is implemented for $x(n)$, a 2nd order auto-regressive process with added noise of variance 0.25. The number of samples for each realisation is 1000. The squared prediction error in dB is plotted for 1 realisations and 100 different realisations on two independant graphs. These are also estimated for two different step-sizes: $\mu = 0.05$ and $\mu = 0.01$

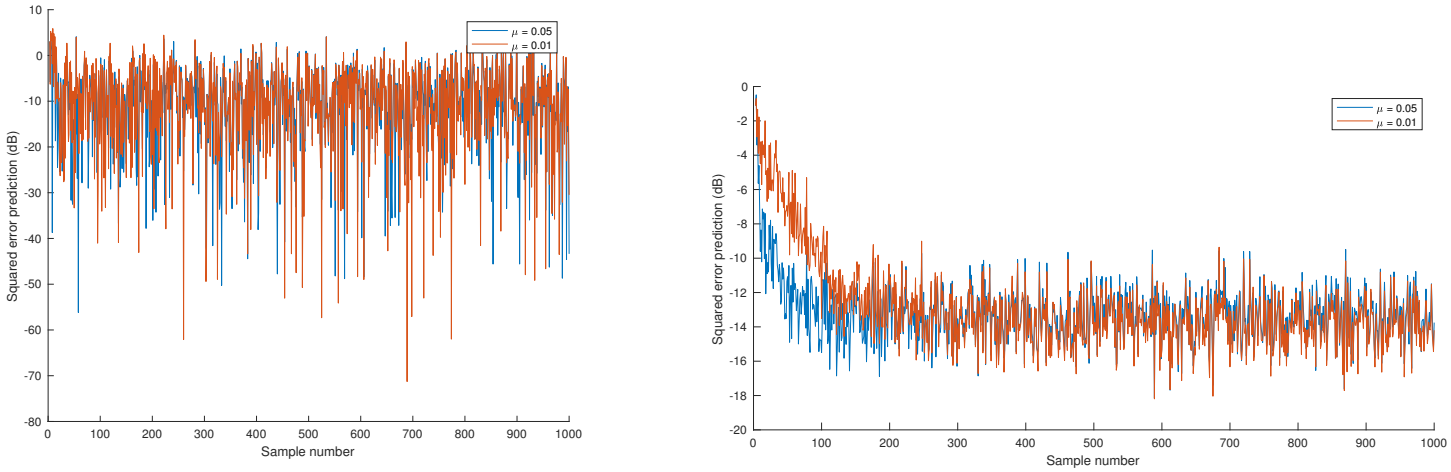

Figure 14: Squared error prediction of LMS adpative predictor in dB. Left: 1 realisation; Right: Average over 100 realisations (Learning curve)

For 1 realisation, the squared prediction error does not vary much with the number of sample. This is due to the fact that with one realization the variance is too high to notice the difference with sample numbers. In other words, the squared prediction error will not change with time as for one realization the value at n is random. Only the mean over different realizations will the variance decrease and the squared prediction error as well.
Therefore, for a 100 realizations an initial considerable decrease is observed before stabilizing approximately around -14dB for both step-size values. For $\mu = 0.05$ and $\mu = 0.01$, this state is respectively reached at n = 90 and n = 210. The convergence is faster for bigger μ values. However, the variance of the squared prediction error with $\mu = 0.01$ reduces more than $\mu = 0.05$. Which means that smaller μ values, the error is reduced and thus yields better results. There is a trade-off between time convergence to the correct value and variance of the error.

c) The misadjustement is a dimensionless quantity used to quantify the accuracy of the convergence of the LMS algorithm. The theoretical LMS misadjustement is approximated with:

$$\mathcal{M}_{LMS} = \frac{\mu}{2}\mathrm{Tr}\{\mathbf{R}\} \tag{17}$$

where **R** corresponds to the auto-correlation matrix of the input determined in part a) (equation (14)). The trace of **R** is calculated and found to be equal to 1.8519.
The steady-state for both is definitely reached from t = 500. Therefore, the estimated values of misadjustement are determined by time-averaging over the last 500 ensemble-averaged mean square error. The MSE is first calculated. The estimated misadjustement is then equal to the MSE subracted and divided by the variance:

$$\mathcal{M}_{est} = \frac{\mathrm{EMSE}}{\sigma_\eta^2} = \frac{\mathrm{MSE} - \sigma_\eta^2}{\sigma_\eta^2} \tag{18}$$

The results are summarized in the following table

μ	$\mathcal{M}_{LMS}$	$\mathcal{M}_{est}$
0.05	0.0463	0.0605
0.01	0.0093	0.0177

Table 1: Estimated and theoretical misadjustement of LMS algorithm for different μ values

Comparing the estimated values with the theoretical LMS misadjustement, we can see that they are in the same order. The estimated values seems to be slightly higher than the theoretical one in both cases. This could be explained by the fact that 500 values for time-averaging is used. If more steady-state values are introduced, it might resemble the theoretical value more closely. However, we are limited by the speed of convergence and the number of samples.
The LMS misadjustement is considerably lower for $\mu = 0.01$, since the variance is lower for smaller μ values inducing larger oscillations as explained previously.

d) The values of the adaptive filter coefficients averaged over 100 independent trials is plotted for 1000 samples and compared to the true coefficients.

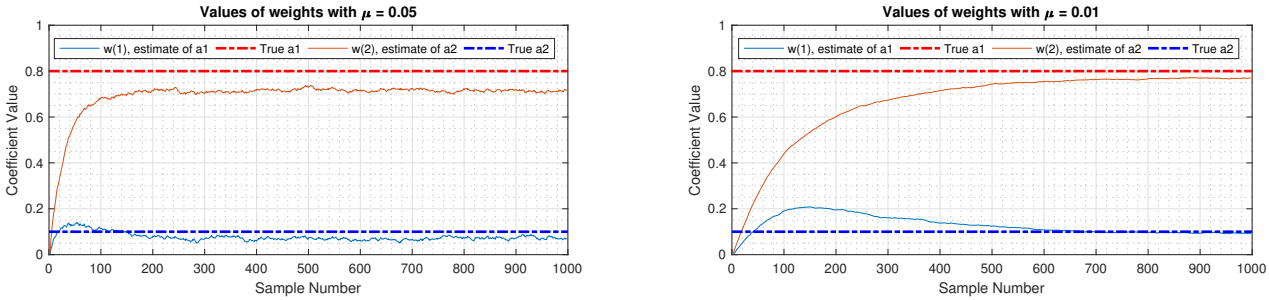

Figure 15: Adaptive filter coefficients for step-sizes $\mu = 0.05$ and $\mu = 0.01$ and true coefficients

For $\mu = 0.05$, the steady state value is reached after n = 200 whereas for $\mu = 0.01$ the estimated coefficients seem only stable after n = 600. Both estimated coefficients are lower than the actual value due to the high variance observed previously.
For $\mu = 0.01$, the coefficients are reached more slowly but the curves is smoother than with the larger μ value. For larger sample number value (n > 450) the coefficient estimates obtained with $\mu = 0.01$ is closer to the true value than $\mu = 0.05$. The percentage error is calculated for each cases and presented in the following table

Coefficient	$\mu = 0.05$ **Percentage error**	$\mu = 0.01$ **Percentage error**
a1	29.62%	1.69%
a2	10.52%	4.78%

Table 2: Percentage error for each weight coefficients obtained with LMS algorithm with different *mu* values

The adaptive filters weights are more accurate for the smaller μ value as expected.

e) To determine the LMS coefficient update for $\mathbf{w}(n)$ that minimizes the cost function $J_2(n) = \frac{1}{2}(e^2(n) + \gamma\|\mathbf{w}(n)\|_2^2$, the gradient of $J_2(n)$ is first calculated.
Using $e^2(n) = (x(n) - \hat{x}(n))^2$ and $\hat{x}(n) = \mathbf{w}^\mathbf{T}(n)\mathbf{x}(n)$, the terms are replaced in the equation and expanded:

$$J_2(n) = \frac{1}{2}((x(n) - \mathbf{w}^\mathbf{T}(n)\mathbf{x}(n))^T(x(n) - \mathbf{w}^\mathbf{T}(n)\mathbf{x}(n)) + \gamma(\mathbf{w}^\mathbf{T}(n)\mathbf{w}(n))) \tag{19}$$

The gradient is then:

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

With the method of steepest descent, the update equation is $\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\Big[-\nabla_{\mathbf{w}} J_2(n)\Big]$ From the previous equation, we have:

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

This final equation corresponds to the leaky LMS.

f) The aim of the LMS algorithm is to converge to the optimum Wiener filter solution if condition (15) is filled. This corresponds to:

$$\mathbf{w}_* = \mathbf{R}^{-1}\mathbf{p} \tag{20}$$

where $\mathbf{R}$ is the auto-correlation matrix of the input and $\mathbf{p}$ is the cross-correlation vector between the desired output and input vector. This equation assumes that $\mathbf{R}$ is invertible which is not the case if one of the eigenvalues is equal to 0. Furthermore, the LMS algorithm is highly dependant on the spread of the eigenvalues of the autocorrelation matrix. Another issue also arises when the input is highly correlated. Therefore, the leaky LMS algorithm is introduced to allow the weights to converge and stabilize the system.

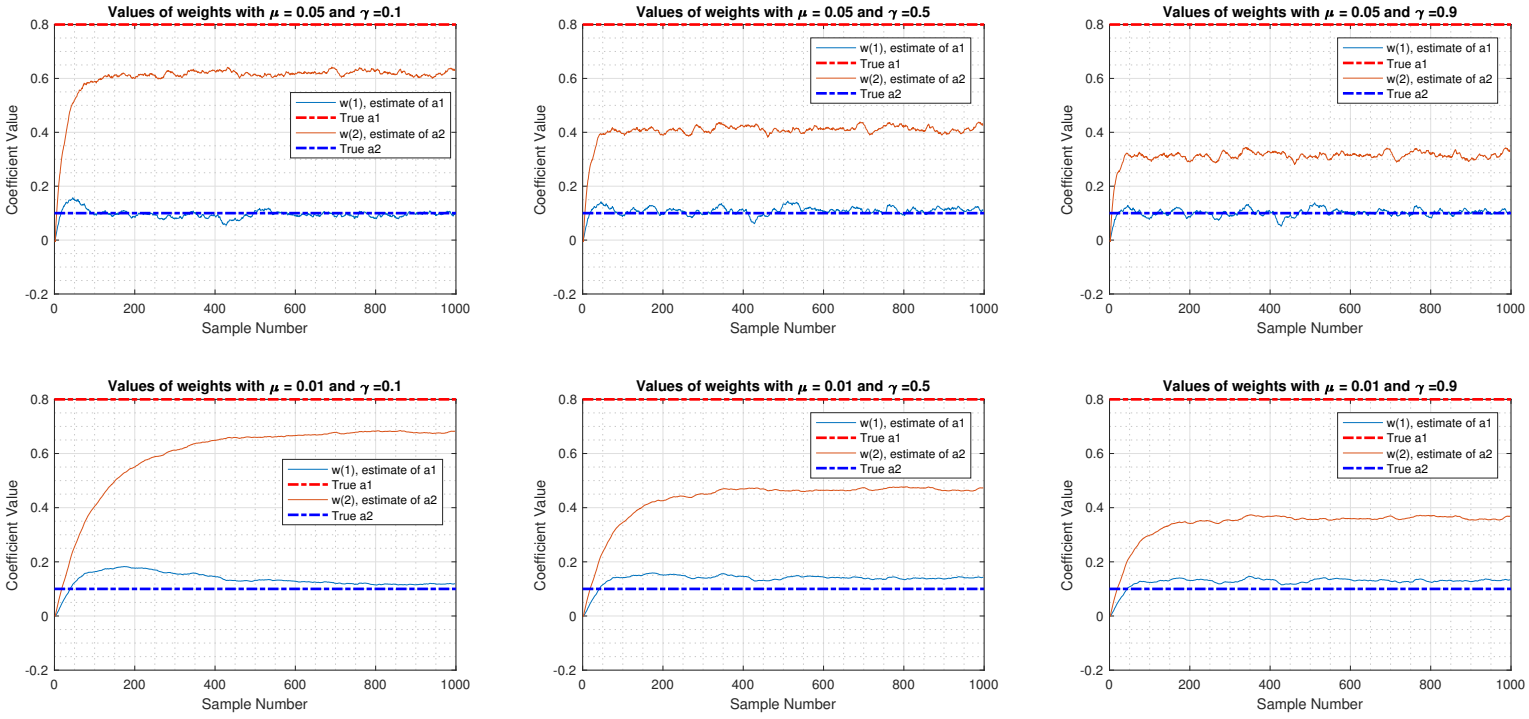

Figure 16: Evolution of the weights estimate with the leaky LMS algorithm for different γ and μ values

It can be shown that the weights for the leaky LMS algorithm converges to:

$$\mathbf{w}_{leaky} = (\mathbf{R}+\mathbf{I})^{-1}\mathbf{p} \tag{21}$$

Therefore, the weight do not converge to the optimum Wiener filter solution and a bias is introduced with this leaky algorithm. The effect of the *leakage* coefficient is the same as adding zero-mean white noise with variance γ. This results in an increase of uncertainty in the input signal holding the coefficients back (phenomenon is referred as *prewithenng*. This is more visible for the weight estimate a2 on Figure 16, as with increasing γ value, the estimate goes from 0.6 to 0.3 approximatively with value of γ ranging from 0.1 to 0.9. Furthermore, as γ increases, the variance increase is also observed as more oscillation appears.

However, γ ensures invertibility of the autocorrelation matrix. It can be seen as a regularization parameter that can be varied between 0< γ <1.
Additionally, with the "new" eigenvalues of the leaky LMS, the spread between them is smaller than the original eigenvalues since $\gamma >= 0$. Therefore, there is a trade-off between achieving the greatest reduction in eigenvalues spread and introducing biases in the solution[1].

2.2 Adaptive Step Sizes

a) In order to deal with the trade-off between convergence speed and steady state error variance when choosing a fixed value of μ, three different variable step-size (VSS) algorithm are implemented: Benveniste (B), Ang & Farhang (AF) and Mattehws & Xie (MX).
The stating point for the three methods is $\mu = 0.1$

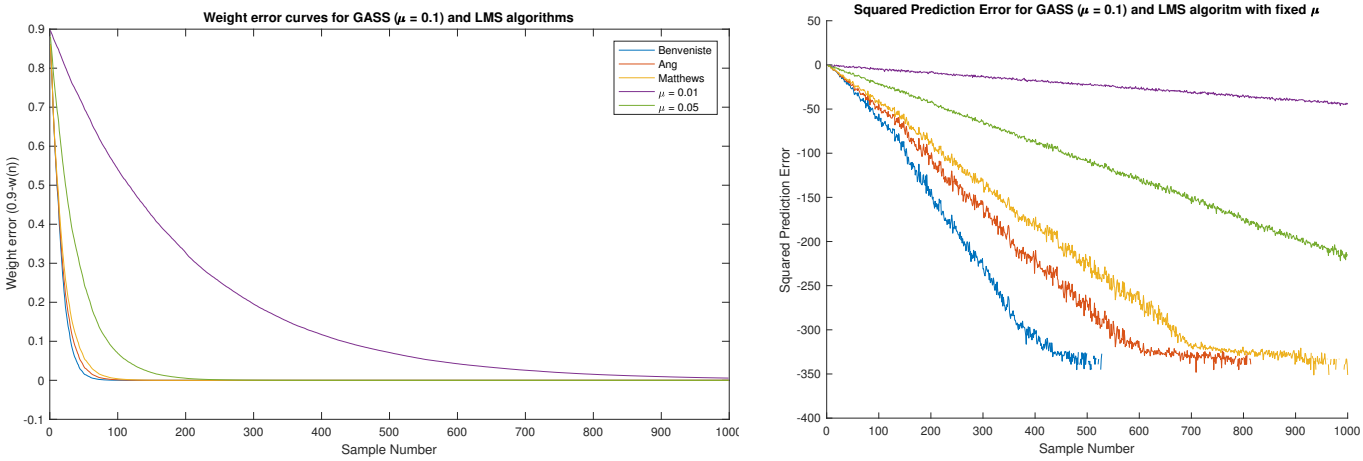

Figure 17: Weight error curve (left) and Squared error prediction (right) of GASS and LMS algorithm

The fastest convergence is achieved with B, followed by AF. The slowest convergence is with MX. Therefore, the best GASS algorithm seems to be Benveniste, which is the fastest and has the smallest steady-state error. However, MX is computationally less complicated than B and AF. The advantage of a standard LMS algorithm is the time taken to compute the result is faster as less computation is done. However, the variance to the steady-state error is fixed and the result is not able to converge more. The advantage of the GASS algorithm is that the steady-state error is reached more quickly and has a small steady-state error. The disadvantage is the computational complexity adding computational time.

b) The update equation based on the *a posteriori* error $e_p(n) = d(n) - \mathbf{x}^\mathbf{T}(n)\mathbf{w}(n+1)$ corresponds to

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e_p(n)\mathbf{x}(n) \tag{22}$$

To introduce $e_p(n)$ on the left hand side of equation (18), it is first multiplied by $-\mathbf{x}^\mathbf{T}(n)$ and then $d(n)$ is added.

$$d(n) - \mathbf{x}^\mathbf{T}(n)\mathbf{w}(n+1) = d(n) - \mathbf{x}^\mathbf{T}(n)\mathbf{w}(n) - \mathbf{x}^\mathbf{T}(n)\mu e_p(n)\mathbf{x}(n) \tag{23}$$

On the right hand side, we notice that the *a priori error* $e(n)$ is introduced $(e(n) = d(n) - \mathbf{x}^\mathbf{T}(n)\mathbf{w}(n))$. It is now possible to express $e_p(n)$ in terms of $e(n)$

$$e_p(n) = e(n) - \mathbf{x}^\mathbf{T}\mu e_p(n)\mathbf{x}(n)$$

$$e_p(n) = e(n)\frac{1}{1 + \mu\mathbf{x}^\mathbf{T}\mathbf{x}(n)}$$

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

Replacing $e_p(n)$ by $e(n)$ in the initial update equation gives:

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

If $\beta = 1$ and $\epsilon = \frac{1}{\mu}$, the update equation based on the *a posteriori* error is equivalent to the normalized LMS (NMLS).

c)The generalized normalized gradient descent (GNGD) algorithm is the adaptive NLMS algorithm which enables the normalized LMS algorithm to not explode for large values of μ

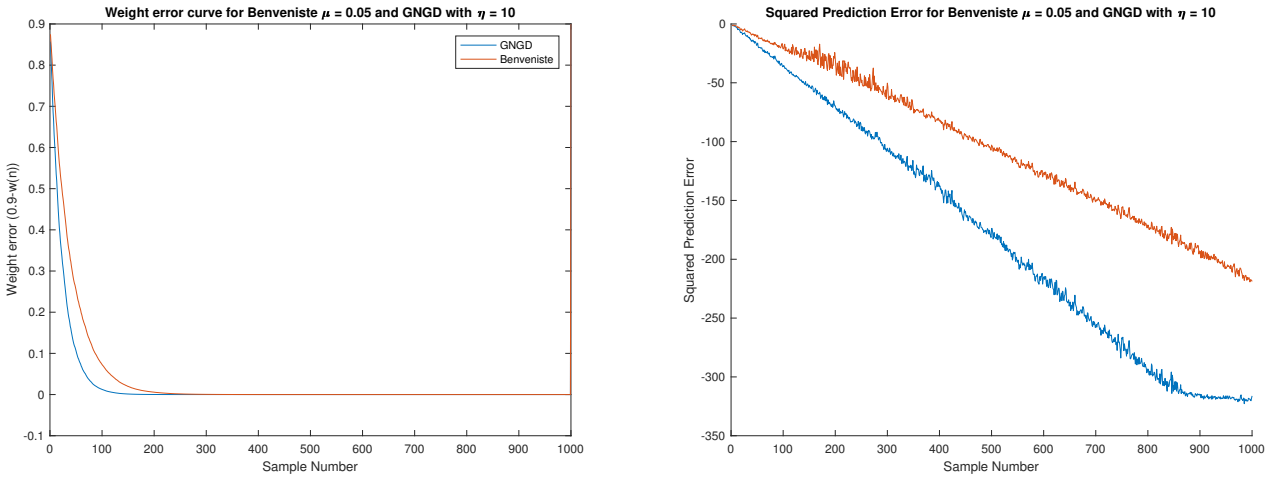

Figure 18: Weight error curve (left) and Squared error prediction (right) of Benveniste and GNGD algorithm

The GNGD has a faster convergence and a smaller prediction error making it a better system identification algorithm than Benveniste's GASS algorithm.
The computational complexity of GNGD is also lower than the Benveniste algorithm. Indeed, the GNGD only has inner product matrix calculations, which means that they are multiplications and additions in the dimensions of the filter length (L). GNGD algorithm is then $\mathcal{O}(L)$. In the case of Benveniste there is also matrix multiplication such as $\mathbf{x}(n-1)\mathbf{x}^T(n-1)$ and $[I-\mu(n-1)\mathbf{x}(n-1)\mathbf{x}^T(n-1)]\varphi(n-1)$ which implies multiplications in the order of L^2. Thus, Benveniste's GASS algorithm is $\mathcal{O}(L^2)$ The GNGD seems a better alternative algorithm to the other GASS algorithms as it is computationally less complex and yields a faster convergence.

2.3 Adaptive Noise Cancellation

a) The delay Δ must be chosen to the noise component in the signal $s(n)$ is uncorrelated to the predictor input $\mathbf{u}(n)$ whereas the periodic signal remains approximately the same (corresponds to a phase shift).
The MSE is expanded to determine how the noise correlation affects the linear predictor:

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

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

By regrouping the terms adequately, we get:

$$\mathbb{E}\{(s(n)-\hat{x}(n))^2)\} = \mathbb{E}\{\eta(n)^2\} + \mathbb{E}\{(x(n)-\hat{x}(n))^2\} + 2\mathbb{E}\{\eta(n)(x(n)-\hat{x}(n))\} \tag{24}$$

The first two terms are not of interest to determine the minimum delay value as the first one corresponds to the noise power independent of Δ and the 2nd one is the mean squared prediction error and there is no multiplication between $\eta(n)$ and Δ which means that it does not affect the correlation. Therefore, the only term of interest is the last one. Here, $\eta(n)$ times $x(n)$ are assumed to be uncorrelated and what is left is: $\eta(n)\hat{x}(n)$.

$$\mathbb{E}\{\ \eta(n)\hat{x}(n)\} = \mathbb{E}\{(v(n)+0.5v(n-2))(\mathbf{w}^T(n)\mathbf{u}(n))\}$$

$$\mathbb{E}\{\ \eta(n)\hat{x}(n)\} = \mathbb{E}\Big\{(v(n)+0.5v(n-2))\Big(\sum_{k=0}^{M+1} w_k(x(n-\Delta-k)+\eta(n-\Delta-k)\Big)\Big\}$$

Since $x(n)$ is uncorrelated the term of interest is η which is replaced by the moving average filter terms:

$$\mathbb{E}\{\ \eta(n)\hat{x}(n)\} = \mathbb{E}\Big\{(v(n)+0.5v(n-2))\Big(\sum_{k=0}^{M+1} w_k(v(n-\Delta-k)+0.5v(n-\Delta-k-2)\Big)\Big\} \tag{25}$$

The terms have expectation equal to 0 when the indices of the white noise multiplied together are different. Therefore, to be uncorrelated the condition that needs to be filled is $\Delta > 2$. Figure 19 shows that if Δ is smaller than 2, the signal has more variance as it accounts the noise as part of the clear signal.

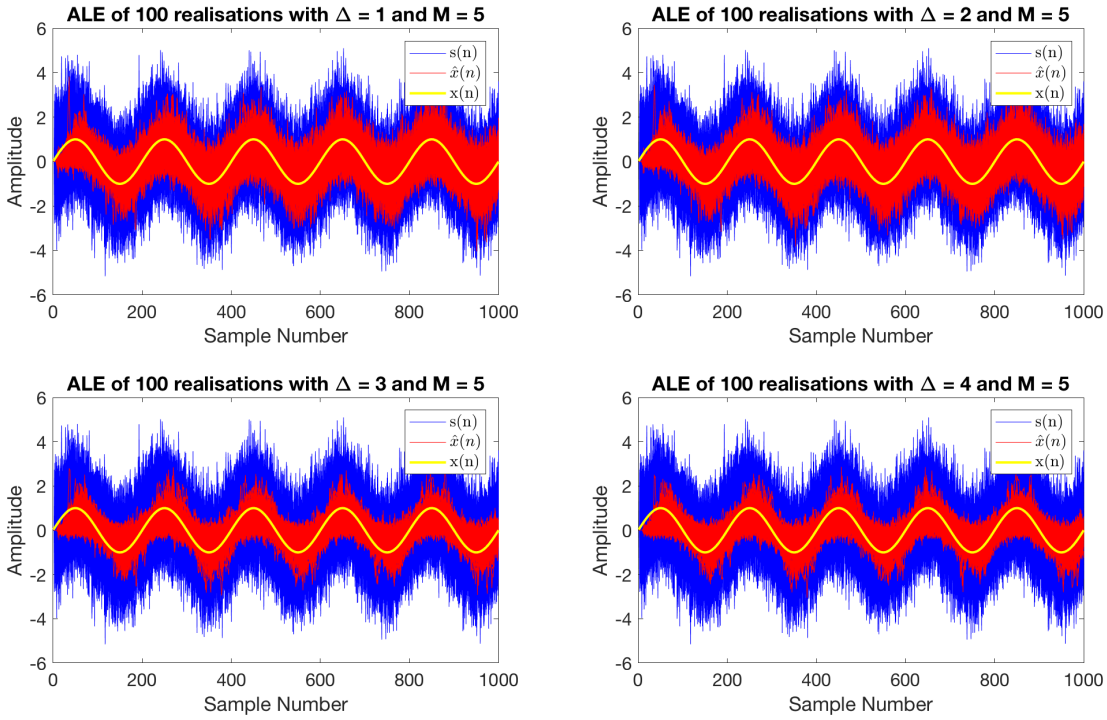

Figure 19: Performance of ALE with different Δ values

b) The mean square prediction error (MSPE) is an estimate of how well the denoising algorithm performs. To analyse the effect of delay Δ and the filter length M, the MSPE for different values of these is calculated.

The results are presented in graphical form to the performance depending on Δ or M. The μ is fixed at 0.01 for the LMS part of the adaptive line enhancement (ALE) in the following simulations.

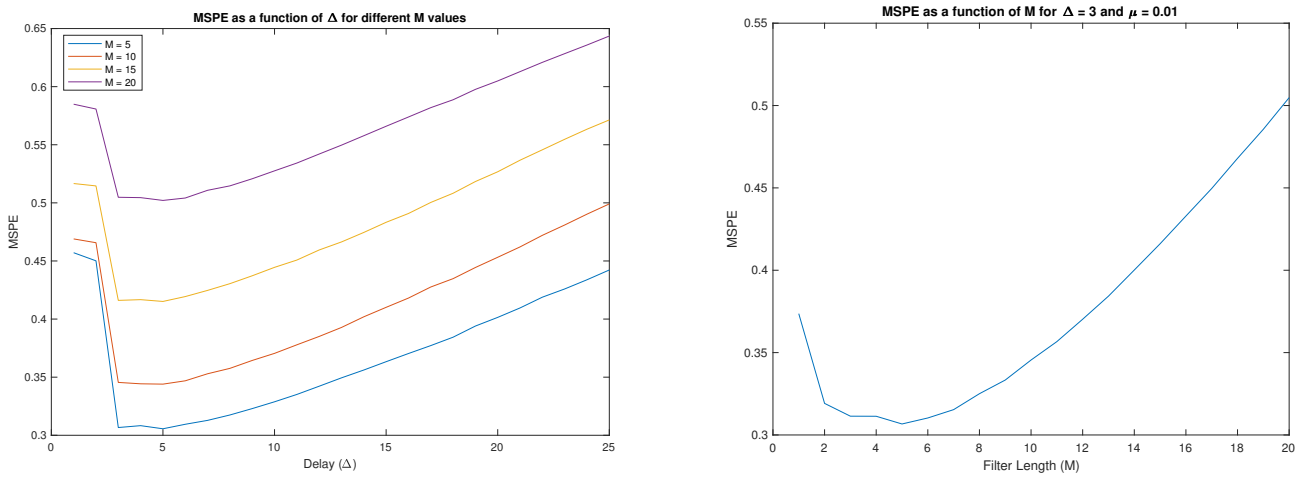

Figure 20: MSPE of ALE with different values of Δ and M

Figure 20 on the left hand side, the MSPE value increases as M increases since for each delay the smaller MSPE is obtained with the smallest M value. For each M, the minimum is either observed at $\Delta = 3$ or $\Delta = 5$. Furthermore as M increases it seems that the minimum Δ is observed more and more for 5. The left figure 20 shows a minimum at M=5 for a $\Delta = 3$.
The delay has a shifting effect. Therefore, if Δ is more than 5 the shift adds more error. The ALE result is then out of phase with the original signal. Therefore, the best delay with minimum error is 3 (As seen previously this is the minimum value possible, because otherwise the noise in the signal and in **u**(n) would be correlated). If M is too big, an overfitting effect is observed and noise is kept, which makes resemble less like x(n) and more like s(n). Thus more variance is observed which explains the increasing curve for the MSPE.
Some further testing showed that smaller μ values the performance is better for longer filter length M, leading to a more accurate result but also a more complex and time consuming model.
Therefore the most efficient and accurate model is with a μ value of 0.01, M and Δ respectively equal to 5 and 3.

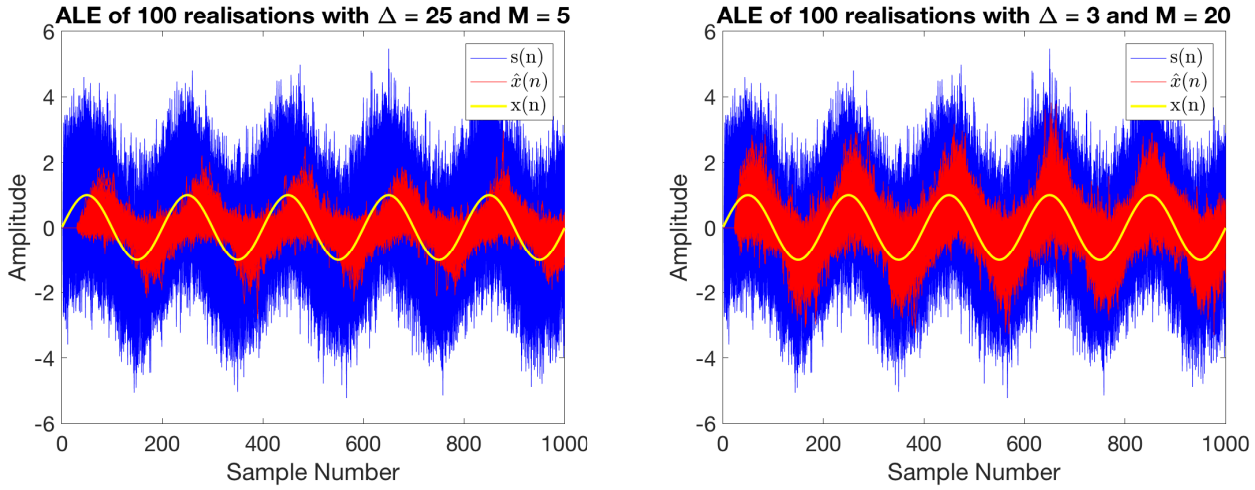

Figure 21: Performance of ALE with big delay (left) and big filter length (right)

c) The adaptive noise cancellation configuration requires a secondary noise input $\epsilon(n)$ which is correlated

to the noise in the corrupted signal. Therefore, $\epsilon(n)$ is equal to the noise signal times 0.9 to be closely correlated and 0.05 is added to it. For comparison purposes, the optimum condition for ALE are chosen such that the $\Delta = 3$ and $\mu = 0.01$. The same value of μ is used in the ANC.

De-noising Adaptive filter	M = 5	M = 10	M = 15	M = 20
ALE	0.2693	0.2620	0.2854	0.3234
ANC	0.0896	0.1161	0.1415	0.1661

Table 3: MSPE value of ALE and ANC with different filter lengths M

From the table, we can see that the ANC filter has a lower MSPE value for every M value tested. Thus, the ANC method is more accurate than ALE in general. However, from Figure 22, we can see that the ANC is only accurate after a certain amount of time, in this case after t = 200. This means that if the number of sample is small, ALE should be used otherwise ANC is more accurate if the number of samples is sufficient.

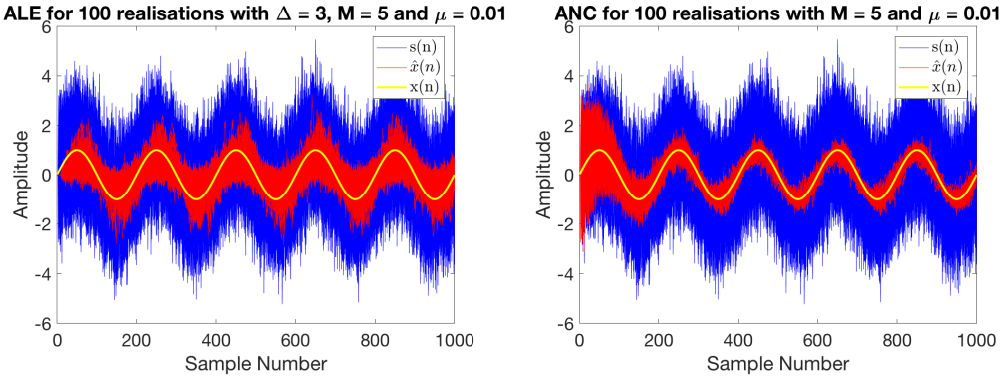

Figure 22: Performance of the ALE and ANC adaptive filters with the original signal, the clean signal and the estimation

d) To remove the 50Hz component introduced by the mains using ANC configuration, a synthetic reference input composed of a sinusoid of 50Hz corrupted by white Gaussian noise. The spectrogram of the corrupted signal and the reference input is presented in Figure 23.

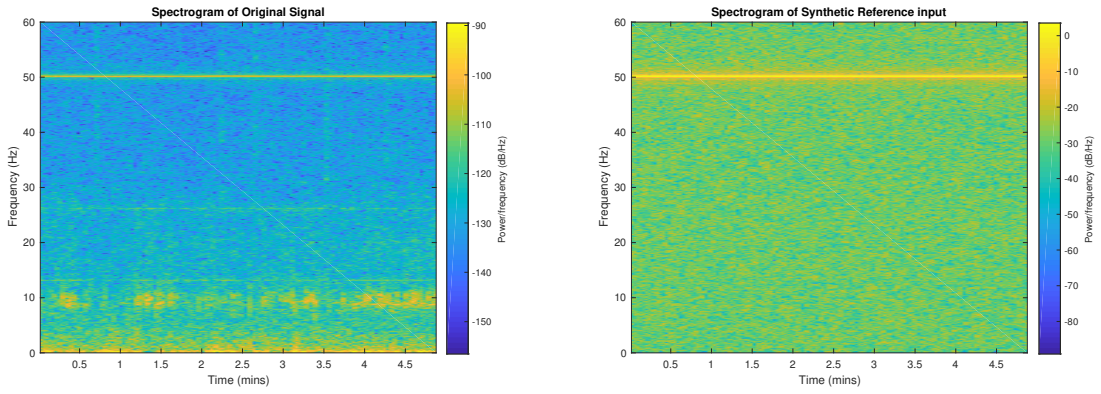

Figure 23: Spectrogram of Original Signal and the Reference Input used to remove the 50Hz component

In both cases, a constant high Power is observed at 50Hz which is expected. The rest of the spectrogram looks different. The aim is to remove the 50Hz component without altering the surrounding frequencies. The spectrogram of denoised signals obtained with different values of M and μ was analysed to determine the optimal parameters to choose. To show the effects of M and μ, only the spectrogram of certain combinations of parameters is shown in the following figure:

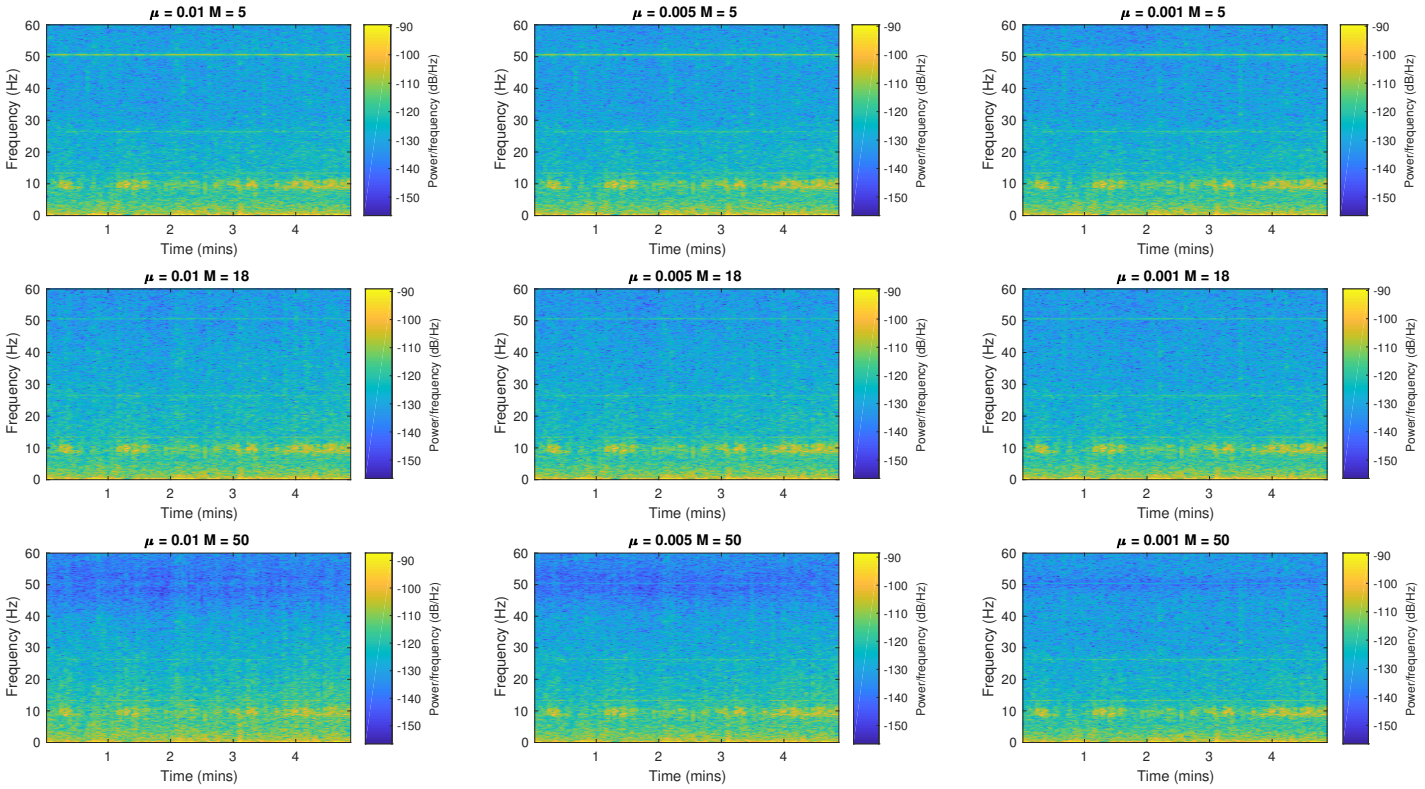

Figure 24: Spectrogram of de-noised signals obtained with ANC using different parameters

M determines how much of the component's intensity is removed. M =5 is insufficient as the component is still visible. However, in the case for M = 50, too much of the surrounding is removed (larger patch of dark). Thus, the M, needs to be as small as possible as to not affect the surrounding frequencies but high enough to actually remove the component. μ seems to minimize the effect to the surrounding frequencies. However, small values of μ means that it takes more time as it is a smaller time step. Therefore, the optimal combination seems to be M = 18 and $\mu = 0.005$. Applying this gives the periodogram of Figure 25. We can indeed notice at 50Hz the absence of spike in the de-noised signal and the presence of the surrounding frequencies.

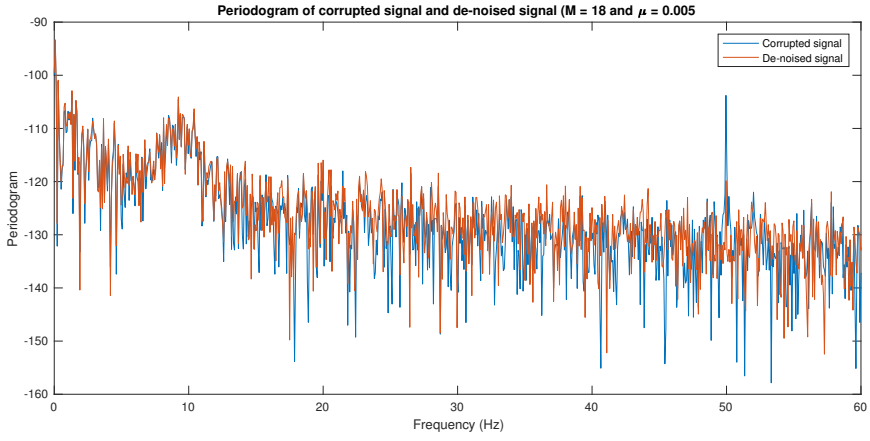

Figure 25: Periodogram of corrupted and de-noised signal with optimal parameters

3 Widely Linear Filtering and Adaptive Spectrum Estimation

3.1 Complex LMS and Widely Linear Modelling

a) The complex LMS (CLMS) is the standard strictly linear model for complex data is valid only for circular random variables. For general complex data, the widely linear framework is used which has sufficient degrees of freedom to capture second-order statistics in the data. This corresponds to the augmented CLMS (ACLMS)

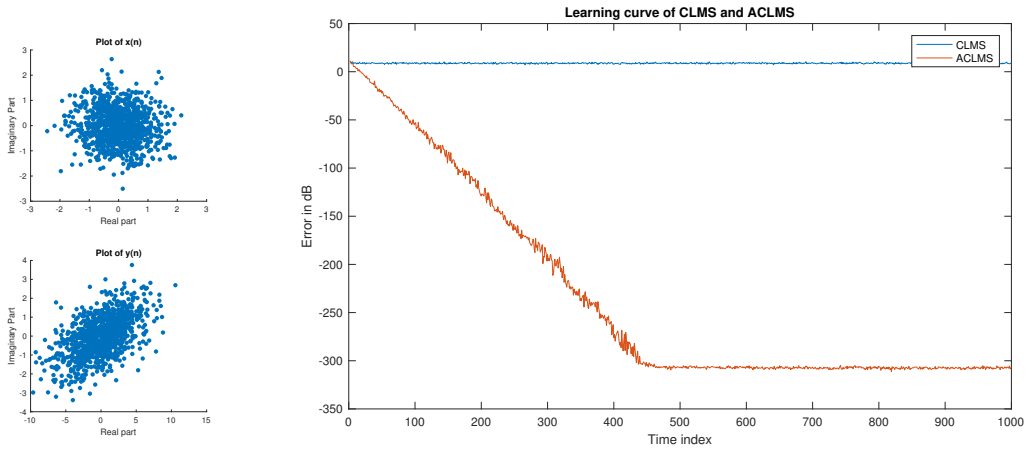

Figure 26: Learning curve of ACLMS and CLMS to identify y(n) driven by x(n)

From the figure above, we can clearly see that x(n) is circular white Gaussian noise. This drives y(n) which is not circular. This means that the CLMS is invalid for y(n) and that ACLMS is needed. This is portrayed above as the steady state error for CLMS is around 8dB and around -309dB for ACLMS. This is due to the fact that CLMS does not have enough degrees of freedom to capture the 2nd order statistics as it has an elliptical shape. The speed of convergence to steady state error of ACLMS can be modulated with the step size μ. Here, μ is equal to 0.1.

b) The circularity plots of the three regimes are presented separately to see their shape and on the same plot for comparison in Figure 27.

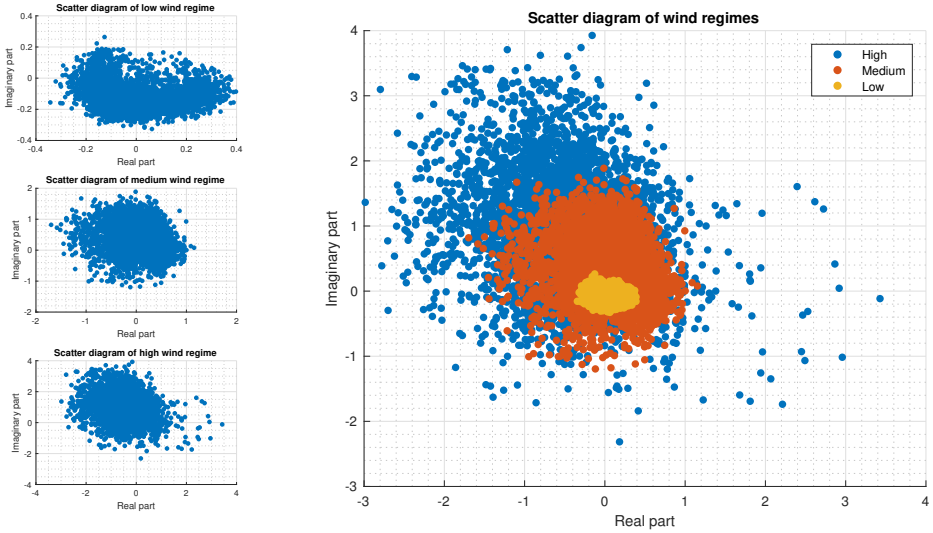

Figure 27: Scatter diagrams (circularity plots) of the three different wind regimes

Even though, low wind looks the least circular in the separated plots, it is actually the most circular compared to the other two as it is less spread out and more centered around 0. To verify this, the circularity coefficient ρ is determined for the three winds and summarized in table 4. This is a degree of non circularity, since the close the value of ρ to 0. A circular data means that the distribution is rotationally invariant: it does not depend on the angle. The smallest value is observed for low wind with ρ equal to 0.159 approximately. As expected The highest number is observed with high wind. Due to the difference in spread, the value of μ is varied for each case: 0.1, 0.01 and 0.001 for low, medium and high wind respectively.

Wind	ρ
Low	0.159
Medium	0.454
High	0.624

Table 4: Table of circularity coefficient values for the wind regimes

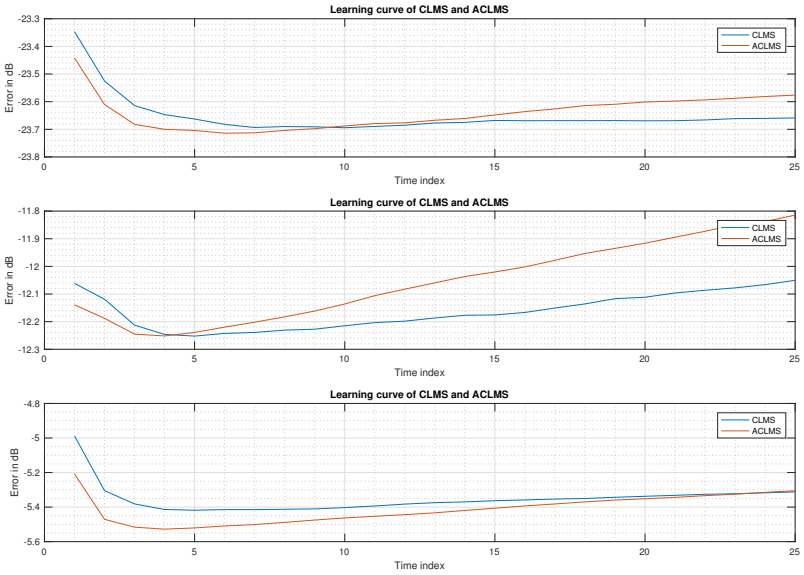

Figure 28: Squared error prediction for different filter lengths

As the distribution becomes more non-circular, the global error increases. The optimal performance for each case is achieved for ACLMS as it deals better with non-circular data. Especially for high wind, the ACLMS outperforms the CLMS by a greater margin. The minimum results are observed for filter lengths M between 3 and 7. As M increases, the error increases again due to overfitting. For those values, CLMS has a smaller error as it has less degrees of freedom and overfits less the noise than ACLMS.

c) The circularity diagram of balanced and unbalanced system of the complex voltages $\alpha - \beta$ is plotted in Figure 29. For unbalanced system, on one plot the voltage magnitude varies and on the other one the phase distortion relative to the balanced three-phase system Δ is varied.
As expected with a balanced system, the shape observed is a circle. This means that with a balanced system, the CLMS system is enough to estimate the nominal frequency. Furthermore the circularity coefficient is equal to 0.000.
For unbalanced system, the shape is elliptical. The voltage value affect the width of the ellipse and the general direction. If $V_c < V_b$, the direction is diagonal up-left and down-right. If $V_c > V_b$, the direction is diagonal up-right and down-left. The Δ affect the angle tilt. If Δ_c increases, the ellipse tilts clockwise, whereas if Δ_b increases, the ellipse tilts anti-clockwise. The circularity diagram can be used to identify if the system is balanced or unbalanced based on the shape. If the system is unbalanced, it is possible to identify the fault in the system based on the direction, width and angle tilt of the ellipse.

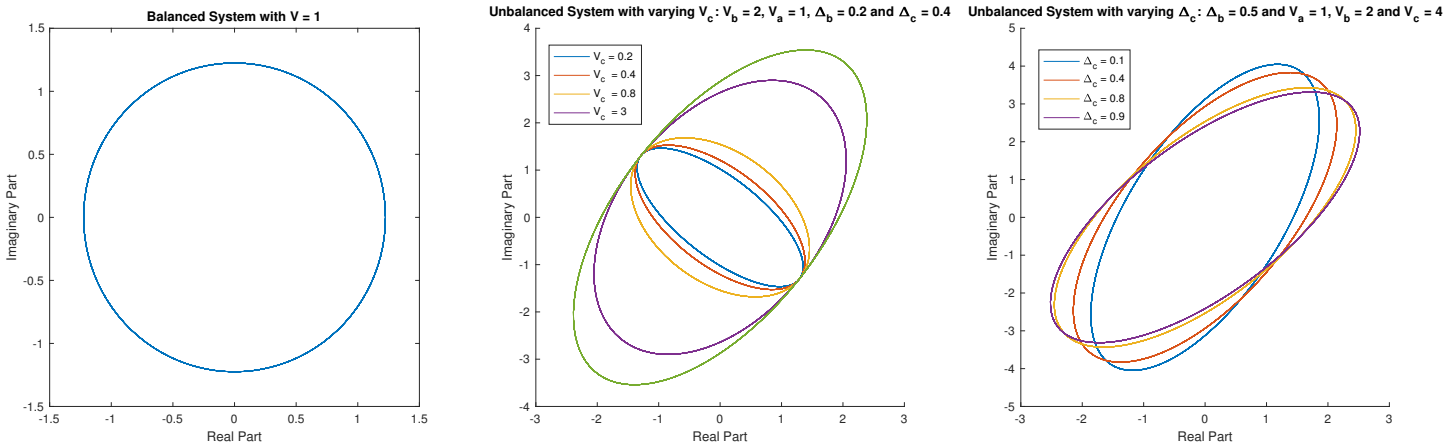

Figure 29: Circularity diagrams of balanced and unbalanced system with different parameter combinations

d) The strictly linear autoregressive models of order 1 is given by $v(n+1) = h^*(n)v(n)$. Under balanced conditions we have:

$$v(n) = \sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_o}{f_s} n + \phi)} \tag{26}$$

which gives:

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

$$v(n+1) = \sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_o}{f_s} n + \phi)} e^{j2\pi \frac{f_o}{f_s}}$$

$$v(n+1) = v(n) e^{j2\pi \frac{f_o}{f_s}}$$

We know have:

$$h^*(n) = e^{j2\pi \frac{f_o}{f_s}}$$

$$h(n) = e^{-j2\pi \frac{f_o}{f_s}}$$

$$|h(n)| e^{jarctan\left\{\frac{\mathcal{I}m\{h(n)\}}{\mathcal{R}e\{h(n)\}}\right\}} = e^{-j2\pi \frac{f_o}{f_s}}$$

Those two complex numbers are equal if their angle and magnitudes are equal. This gives:

$$arctan\left\{\frac{\mathcal{I}m\{h(n)\}}{\mathcal{R}e\{h(n)\}}\right\} = 2\pi \frac{f_o}{f_s} \Leftrightarrow f_o(n) = \frac{f_s}{2\pi} arctan\left\{\frac{\mathcal{I}m\{h(n)\}}{\mathcal{R}e\{h(n)\}}\right\} \tag{27}$$

The frequency of the balanced complex $\alpha - \beta$ voltage can be derived from the coefficients in h(n) as shown in equation (27). With the unbalanced system we have the widely linear autoregressive model $v(n+1) = h^*(n)v(n) + g^*(n)v^*(n)$. Replacing $v(n)$ with the general form of the complex-valued voltage we have:

$$v(n+1) = h^*(n)[A(n)e^{j(2\pi \frac{f_o}{f_s} n + \phi)} + B(n)e^{-j(2\pi \frac{f_o}{f_s} n + \phi)}] + g^*(n)[A^*(n)e^{-j(2\pi \frac{f_o}{f_s} n + \phi)} + B^*(n)e^{j(2\pi \frac{f_o}{f_s} n + \phi)}] \tag{28}$$

$$v(n+1) = e^{j(2\pi \frac{f_o}{f_s} n + \phi)}[h^*(n)A(n) + g^*(n)B^*(n)] + e^{-j(2\pi \frac{f_o}{f_s} n + \phi)}[h^*B(n) + g^*(n)A^*(n)] \tag{29}$$

Another expression for $v(n+1)$ can be derived from the definition:

$$v(n) = A(n)e^{j(2\pi \frac{f_o}{f_s} n + \phi)} + B(n)e^{-j(2\pi \frac{f_o}{f_s} n + \phi)} \tag{30}$$

$$v(n+1) = A(n+1)e^{j(2\pi\frac{f_o}{f_s}(n+1)+\phi)} + B(n+1)e^{-j(2\pi\frac{f_o}{f_s}(n+1)+\phi)} \tag{31}$$

Combining equation (29) and (31) gives:

$$A(n+1)e^{j(2\pi\frac{f_o}{f_s}(n+1)+\phi)} = e^{j(2\pi\frac{f_o}{f_s}n+\phi)}[h^*(n)A(n) + g^*(n)B^*(n)] \Leftrightarrow A(n+1)e^{j(2\pi\frac{f_o}{f_s})} = h^*(n)A(n) + g^*(n)B^*(n) \tag{32}$$

$$B(n+1)e^{-j(2\pi\frac{f_o}{f_s}(n+1)+\phi)} = e^{-j(2\pi\frac{f_o}{f_s}n+\phi)}[h^*(n)B(n) + g^*(n)A^*(n)] \Leftrightarrow B(n+1)e^{-j(2\pi\frac{f_o}{f_s})} = h^*B(n) + g^*(n)A^*(n) \tag{33}$$

Assuming that the changes in voltages are negligible over time, we then have A(n+1) = A(n) and B(n+1) = B(n). Using this and rearranging the equations gives:

$$e^{j(2\pi\frac{f_o}{f_s})} = h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} \tag{34}$$

$$e^{-j(2\pi\frac{f_o}{f_s})} = h^*(n) + g^*(n)\frac{A^*(n)}{B(n)} \tag{35}$$

To get the nominal frequency, we need to determine $\frac{B^*(n)}{A(n)}$. To do this both equations are combined. They are complex conjugate of each other:

$$h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} = h(n) + g(n)\frac{A(n)}{B^*(n)} \Leftrightarrow (h^*(n) - h(n)) + g^*(n)\frac{B^*(n)}{A(n)} - g(n)\frac{A(n)}{B^*(n)} = 0 \tag{36}$$

If equation (36) is multiplied by $\frac{B^*(n)}{A(n)}$, we get a quadratic that we can solve.

$$\frac{B^*(n)}{A(n)} = \frac{-(h^*(n) - h(n)) \pm \sqrt{(h^*(n) - h(n))^2 + 4g^*(n)g(n)}}{2g^*(n)} = \frac{-2i\mathcal{I}m\{h(n)\} \pm \sqrt{(-2i\mathcal{I}m\{h(n)\})^2 + 4|g(n)|^2}}{2g^*(n)} \tag{37}$$

$$\frac{B^*(n)}{A(n)} = \frac{\mathcal{I}m\{h(n)\}i \pm i\sqrt{(\mathcal{I}m\{h(n)\})^2 - |g(n)|^2}}{g^*(n)} \tag{38}$$

This is substituted in (34):

$$e^{j(2\pi\frac{f_o}{f_s})} = h^*(n) + \mathcal{I}m\{h(n)\}i \pm i\sqrt{(\mathcal{I}m\{h(n)\})^2 - |g(n)|^2} \tag{39}$$

$$e^{j(2\pi\frac{f_o}{f_s})} = \mathcal{R}e\{h(n)\} \pm i\sqrt{(\mathcal{I}m\{h(n)\})^2 - |g(n)|^2} \tag{40}$$

Frequencies are positive values, so only the positive answer is valid. The complex number obtained on the right hand side is converted to a polar form. Considering r is equal to the magnitude of the corresponding complex number obtained we have:

$$e^{j(2\pi\frac{f_o}{f_s})} = re^{arctan(\frac{\sqrt{(\mathcal{I}m\{h(n)\})^2 - |g(n)|^2}}{\mathcal{R}e\{h(n)\}})} \tag{41}$$

Equating the angles as with the balanced system we obtain:

$$f_o = \frac{f_s}{2\pi}arctan\left\{\frac{\sqrt{(\mathcal{I}m\{h(n)\})^2 - |g(n)|^2}}{\mathcal{R}e\{h(n)\}}\right\} \tag{42}$$

Therefore, from the coefficients $h(n)$ and $g(n)$ we are able to estimate the frequency of the unbalanced voltage from equation (42).

e) CLMS and ACLMS algorithms are used to estimate the frequency of the $\alpha - \beta$ voltages generated previously. The squared error prediction and the frequency estimates are plotted.

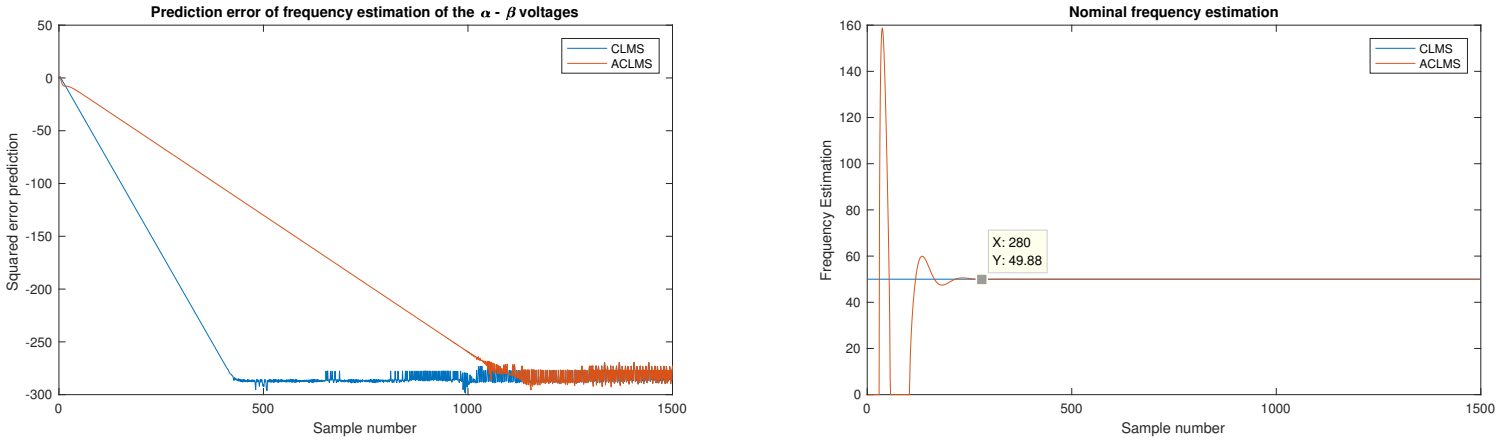

Figure 30: Nominal frequency estimate for a balanced system using CLMS and ACLMS

For a balanced system, CLMS performs better than ACLMS, as the error curve converges faster to steady state. Furthermore, the CLMS directly gets a correct estimate of $f_o = 50Hz$ whereas ACLMS oscillates until n is approximately 280 before reaching a good estimate. The results observed by the ACLMS can be explained by the additional degree of freedom. In this case it is not necessary and even yields a worst result as it is more complex and tries to compensate for the degree of freedom missing. Therefore, for a balanced system CLMS is sufficient.
For unbalanced systems, only one example is shown as the results yield similar global results. The parameters are $V = [1, 0.5, 1.5]$ and $\Delta = [0, 0.2, 0.4]$

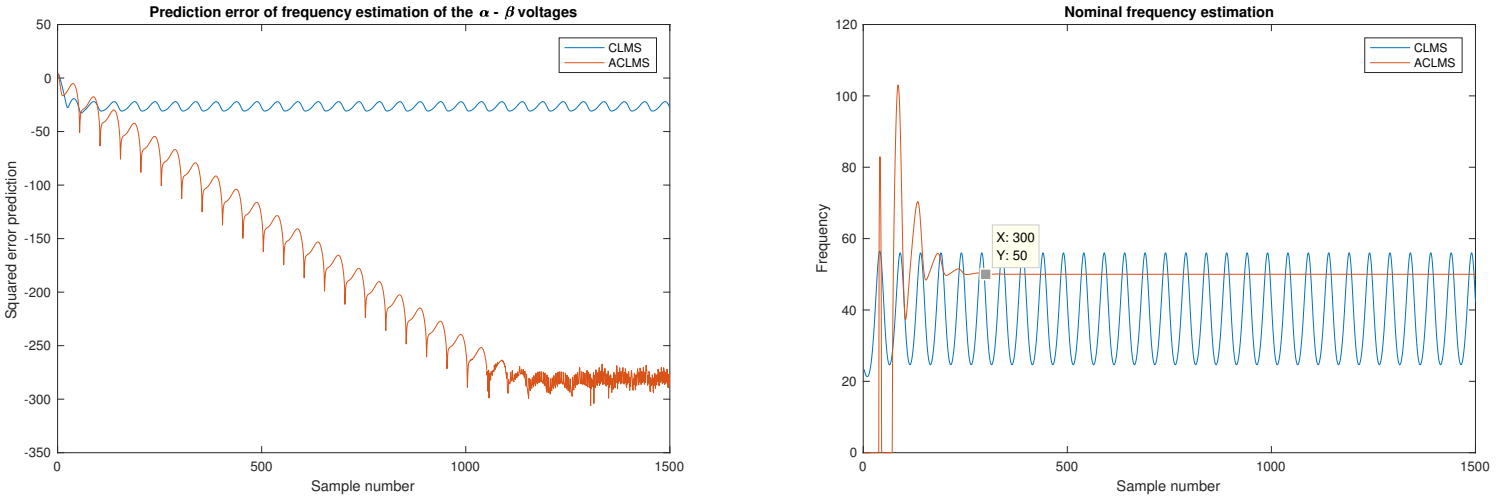

Figure 31: Nominal frequency estimate for an unbalanced system using CLMS and ACLMS

As seen previously, with an unbalanced system the circularity plot is an ellipse instead of a circular. This means there is 2nd order statistical relationships between the input and the output. However, the CLMS is unable to capture such second order statistics in the data. This explains why the CLMS never reaches a correct estimate and oscillates forever like a sinusoid. Therefore, the ACLMS is necessary to acquire a correct nominal frequency. Dampened oscillations are still observed at the beginning, until it stabilizes around 50Hz from n equal to 300.

3.2 Adaptive AR Model Based Time-Frequency Estimation

a) To generate the frequency modulated (FM) signal $y(n)$, the phase $\phi(n)$ is determined from $f(n)$

$$f(n)=\frac{d\phi(n)}{dn}=\begin{cases}100, & 1\leqslant n\leqslant 500\\ 100+\frac{n-500}{2}, & 501\leqslant n\leqslant 1000\\ 100+(\frac{n-1000}{25})^2, & 1001\leqslant n\leqslant 1500\end{cases}\Leftrightarrow\phi(n)=\begin{cases}100n, & 1\leqslant n\leqslant 500\\ -150n+\frac{1}{4}n^2, & 501\leqslant n\leqslant 1000\\ 1700n-1.6n^2+\frac{1}{3\times 25^2}n^3, & 1001\leqslant n\leqslant 1500\end{cases}$$

f_s is chosen to be equal to 1000. It is now possible to compute $y(n)$. The AR(1) coefficients are estimated with `aryule` MATLAB function. The frequency response is then plotted as well as for AR(10) model (to see if the model order affects the performance of the coefficient estimation).

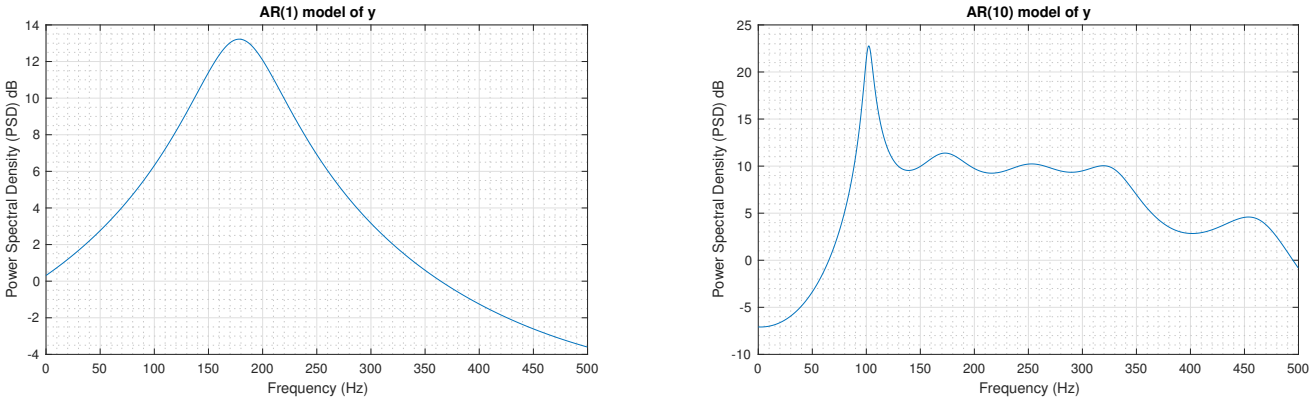

Figure 32: Frequency response for digital filter with AR coefficients of y(n)

AR(1) model is not able to capture the frequency changes. As we can see there is only one peak. This is due to the fact that AR coefficients estimate are used with stationary signals which is not the case here. Increasing the model order does not solve the problem as seen in Figure 33. It tries to find specific frequency value which is not possible. If the estimation is done in batches, the following figure is obtained.

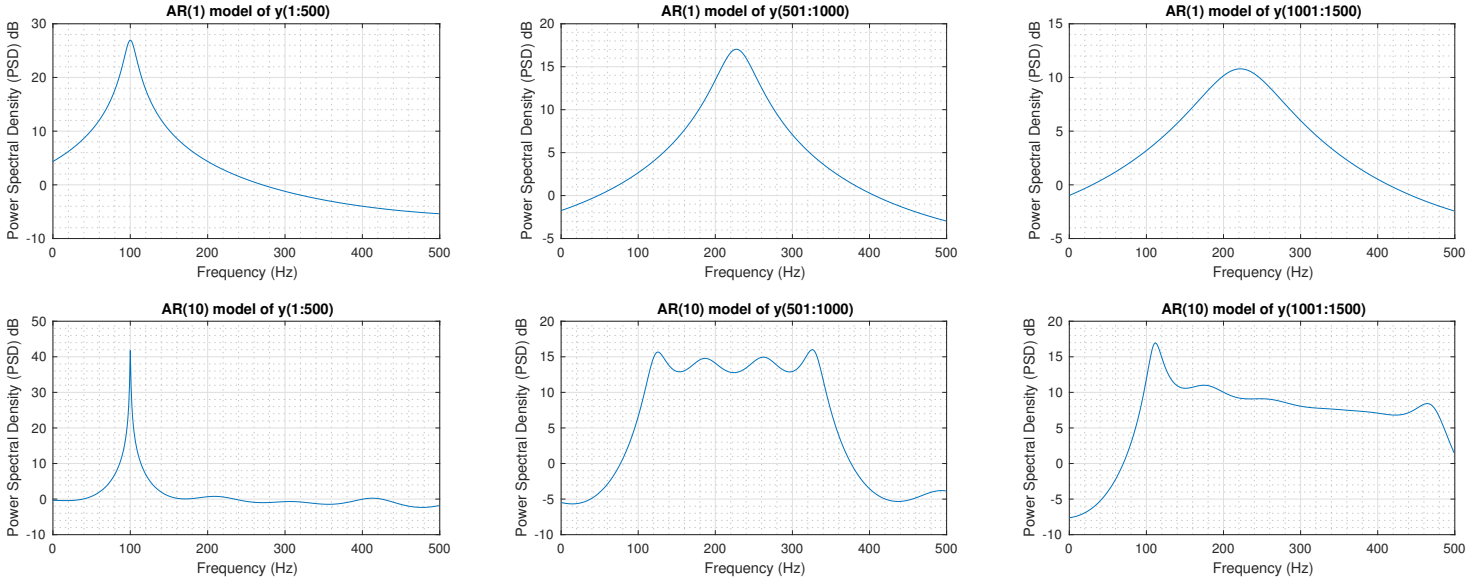

Figure 33: Frequency response for digital filter with AR coefficients of y(n) in batches

If the estimation of the AR coefficients is done in batches, the model AR(1) and AR(10) is only accurate for n=1:500 as the frequency is constant (100Hz). For varying frequencies values, the models are unable

to give an accurate result. Therefore, another method is necessary to estimate the coefficients for non-stationary signals.

b) The CLMS algorithm is used to estimate the AR coefficient of the signal $y(n)$. Different values of μ are tested.

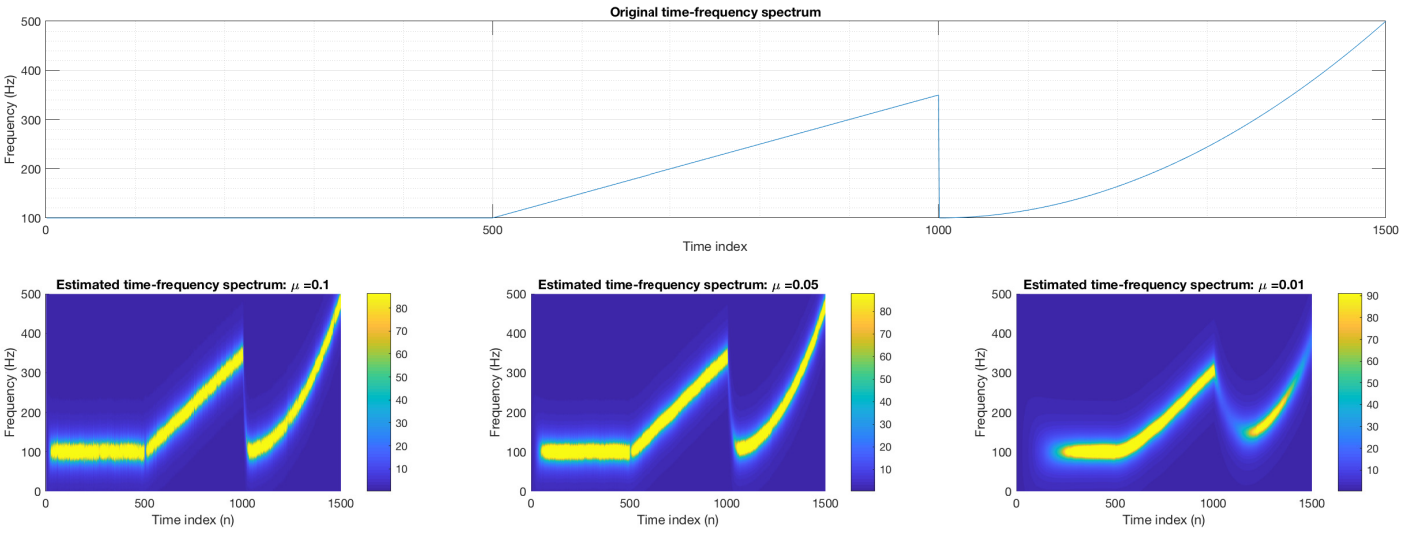

Figure 34: Original and estimated time-frequency spectrum of y(n)

If μ is too small, the CLMS does not have the time to converge to the steady state values which renders an incomplete time-frequency spectrum. If μ is too big, the output variance is higher rendering a time-frequency spectrum with oscillations instead of the actual value. There is a trade-off between speed of convergence and variance. Here, the best case is with μ equal to 0.05. The variance is still higher than the original time-frequency spectrum as a wide yellow band is observed. Averaging over 100 samples could reduce this error. This method is more accurate than the AR coefficient obtained with `aryule` and it is a way to estimate the frequencies of non-stationary signals.

3.3 A Real Time Spectrum Analyser Using Least Mean Square

a) The least square solution of the problem is obtained by minimizing the following equation:

$$J(w) = \|\mathbf{y} - \hat{\mathbf{y}}\|^2 = (\mathbf{y} - \mathbf{F}\mathbf{w})^H(\mathbf{y} - \mathbf{F}\mathbf{w}) = \mathbf{y}^H\mathbf{y} - \mathbf{y}^H\mathbf{F}\mathbf{w} - \mathbf{w}^H\mathbf{F}^H\mathbf{y} + \mathbf{w}^H\mathbf{F}^H\mathbf{F}\mathbf{w} \tag{43}$$

To minimize equation (43), the derivative (determined using matrices properties such as $(AB)^H = B^H A^H$) is set to 0:

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{w}} &= 0 \\ -\mathbf{F}^H\mathbf{y} - \mathbf{F}^H\mathbf{y} + 2\mathbf{F}^H\mathbf{F}\mathbf{w} &= 0 \\ \mathbf{F}^H\mathbf{F}\mathbf{w} &= \mathbf{F}^H\mathbf{y} \end{aligned}$$

If the equation is solved for $\mathbf{w}$ the LS solution for the problem is given by:

$$\mathbf{w} = (\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H\mathbf{y} \tag{44}$$

To relate the equation to the discrete Fourier transform, the inverse discrete Fourier transform (IDFT) is taken:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j2\pi kn/N} \tag{45}$$

F is defined as the matrix of size $K \times N$ with the exponential term of the sum for all n estimates. Collecting all N estimates of the signal x(n) gives: $\mathbf{x} = \mathbf{FX}$
This is of the same form as seen previously for $\hat{\mathbf{y}}$. Therefore, if we want to estimate the IDFT of x(n) using linear combination of N harmonically related sinusoids and determine the coefficients X(n), the equation is:

$$\mathbf{X} = (\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H\mathbf{x} \tag{46}$$

This is how the LS solution can be used in the IDFT.

b) From equation (46), the Fourier coefficients correspond to a linear combination of the columns of **F**. The columns are orthogonal and each column is a basis function. To get the coefficients, **x** is projected onto the basis functions to get the amplitudes of **X** for each k. Therefore, the Fourier transform is a way of decomposing function into a sum of orthogonal basis functions which are $\{e^{-i2\pi kx}, k \in \mathbf{R}\}$

c) The DFT-CLMS algorithm is implemented by generating first the complex phasor which will be the input to the CLMS. The rest is the same as previously for the other CLMS computations.

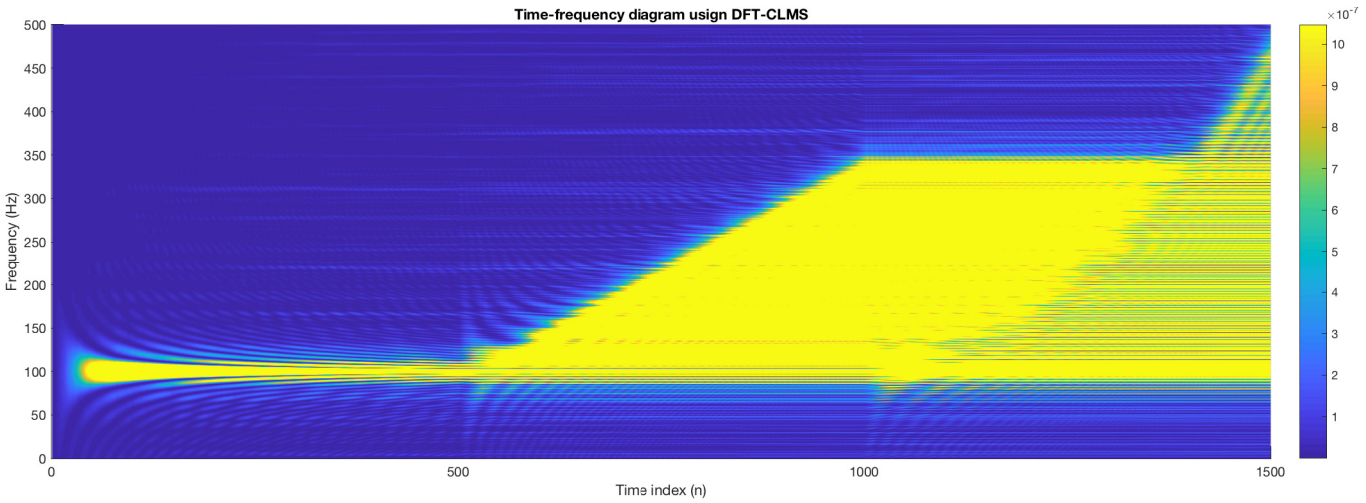

Figure 35: Estimated time-frequency spectrum of y(n) using DFT-CLMS

For n smaller than 500, the DFT-CLMS gives the expected frequency of 100Hz. It seems to perform even better than the CLMS as the variance decreases with time coming closer and closer to 100. However, for varying frequency values, the DFT-CLMS has a poor performance. The general shape is still observable but it presents more frequency than there actually are. Once a frequency value is added with the weights, it stays in time, it does not get updated. This is due to the fact that the weight matrix had K elements instead of 1 or 2 as seen in AR(1) and AR(2) models. Therefore, when you go from n to n+1, the weights have only shifted by one so all the elements seen previously are the same except for the last one. This is implemented in an adaptive way and not block based, this means that backpropagation is slow and that the frequencies picked at a certain point will stay in time and be added with the new values. It takes a while before the old weights are updated. This can be seen as a sort of ”memory”. Therefore, a way to deal with this would be to introduce a leakage factor as seen previously, in order to remove the additional components that are kept due to the size of the weight matrix. With $\mu = 0.05$ the following graph are obtained for different γ values.

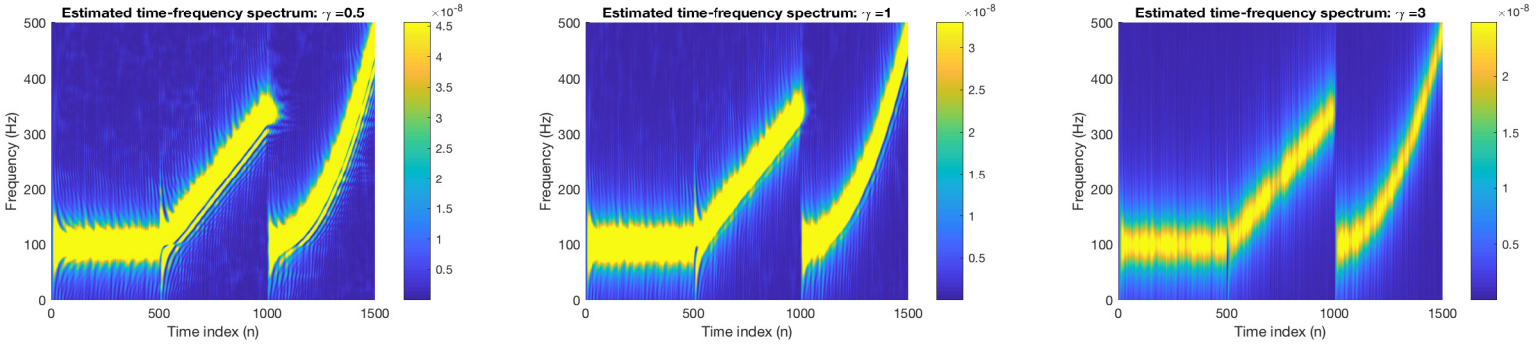

Figure 36: Estimated time-frequency spectrum of y(n) with leakage DFT-CLMS

d) The DFT-CLMS is used on the EEG signal to get a time-frequency diagram.
On the time-frequency diagram of the EEG in Figure 37, the component of the mains artefact at 50Hz is observable as well as the fundamental frequency response peak at 13Hz. The 2nd harmonic at 26 is also visible. The strong luminescence in the lower frequency values can be explained by the alpha-rythm as the subject was tired during recording. The time-frequency diagram presents us the same information as the peaks of the standard periodogram. However, in both cases noise is still present we can see on the spectrum additional frequency components. The third harmonic is also not visible.

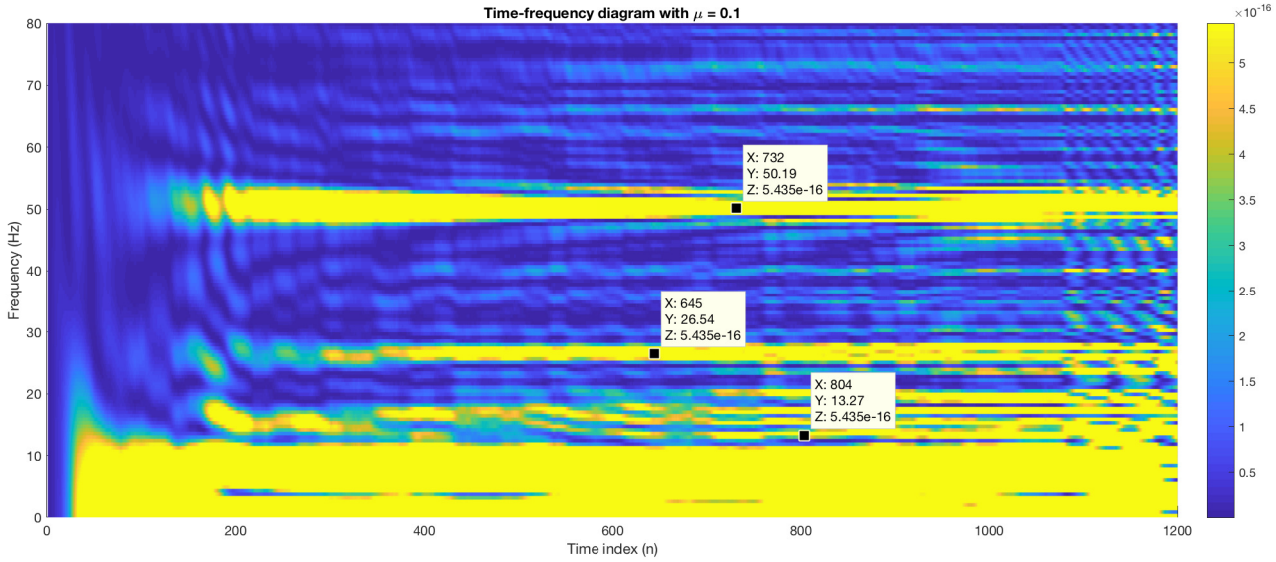

Figure 37: Time-frequency spectrum of EEG signal

4 From LMS to Deep Learning

The values of MSE and Prediction gain R_p calculated for each case is summarized in a table at the end of 4, to be able to compare the results.

1. The mean is removed from the time-series. A prediction of the data is made using the LMS algorithm assuming the data is generated using an AR(4) process.

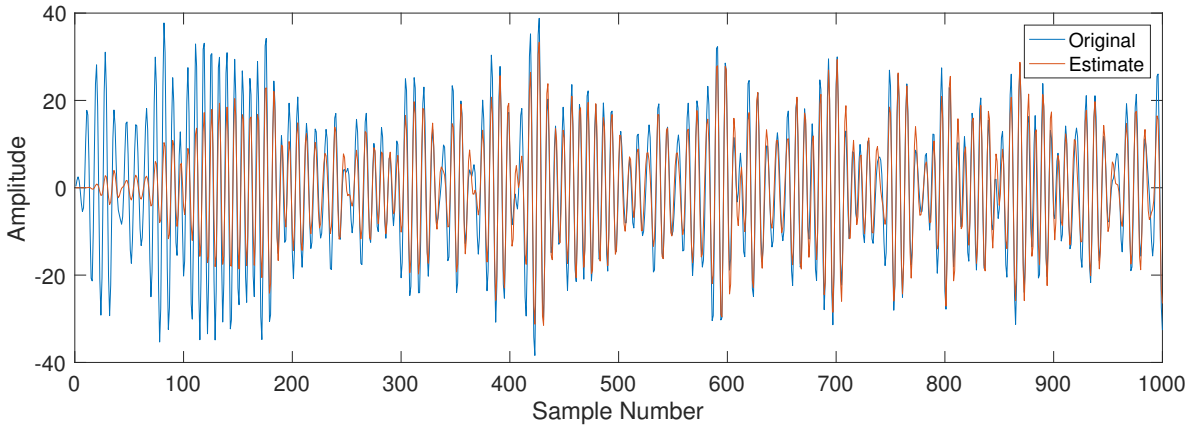

Figure 38: Original and estimated time-series obtained using LMS one-step ahead prediction ($\mu = 1 \times 10^{-5}$)

For the predicted output, a transient period is observed for the first 200 samples, where the output oscillates more and more towards the actual time-series value. This corresponds to the weight values adapating and changing until reaching a relatively stable level anf final convergence. The MSE is calculated to be 40.09 approximately and the prediction gain R_p is found to be 5.20. Therefore, even though stability is reached after 200 samples and we get a relatively good approximate, the result is still quite crude and other elements have to be implemented to reduce the MSE.

2. A non-linearity component is added to the output of the LMS in order for the model to become more expressive. In this case, the non-linearity element is a tanh function in order to implement *dynamical perceptron*.

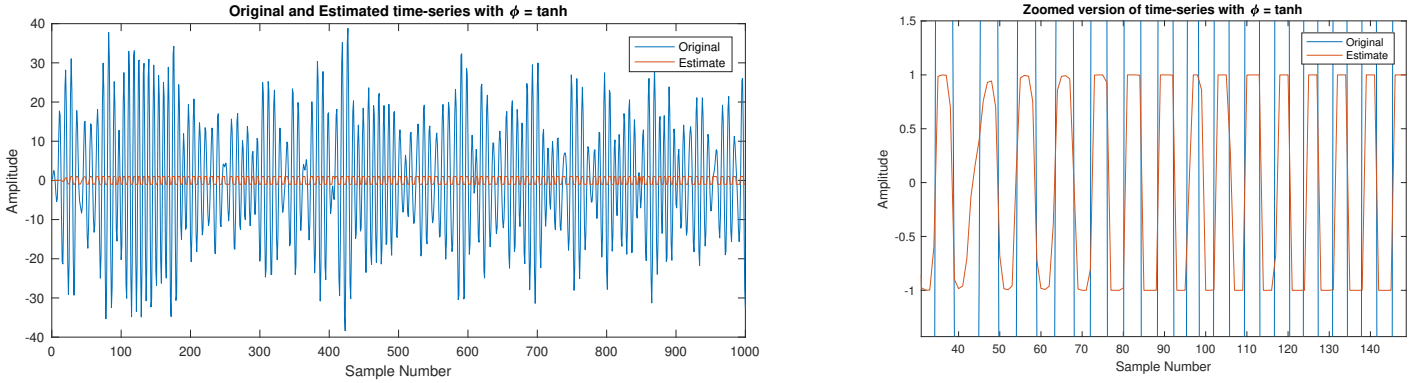

Figure 39: Original and estimated time-series obtained using *dynamical perceptron* ($\mu = 1 \times 10^{-5}$ and $\phi = tanh$)

From Figure 39, it is clear that the tanh function as an activation function is not appropriated. Indeed, the tanh function can only output values between -1 and 1. If the absolute value of the output is higher than 1, the function saturates at 1 or -1. Either the output needs to be scaled down between -1 and 1 or the tanh function is scaled by a certain factor in order to integrate adequately this non-linear component.

3. As mentioned previously, the activation function tanh needs to be scaled in order to capture output values lower and higher than -1 and 1 respectively (a*tanh). The value of a has to be at least greater than the absolute maximum value in the original data. However, since the tanh function gets less sensitive for values further away than 0, the sensitivity decreases as a increases (see Figure 40, right). Therefore, an acceptable range for a is [39 80]. The value of a chosen is 50 and the resulting graph obtained is the left figure of Figure 40.
With a equal to 50, the *dynamical perceptron* is more effective than the simple LMS algorithm. Comparing

the two, we can see that the prediction gain is 3 times bigger. This is due to the fact that no or a very short transient period is observed with the *dynamical perceptron* whereas stability for LMS is only obtained after n = 200. Convergence is quicker with the activation function added.

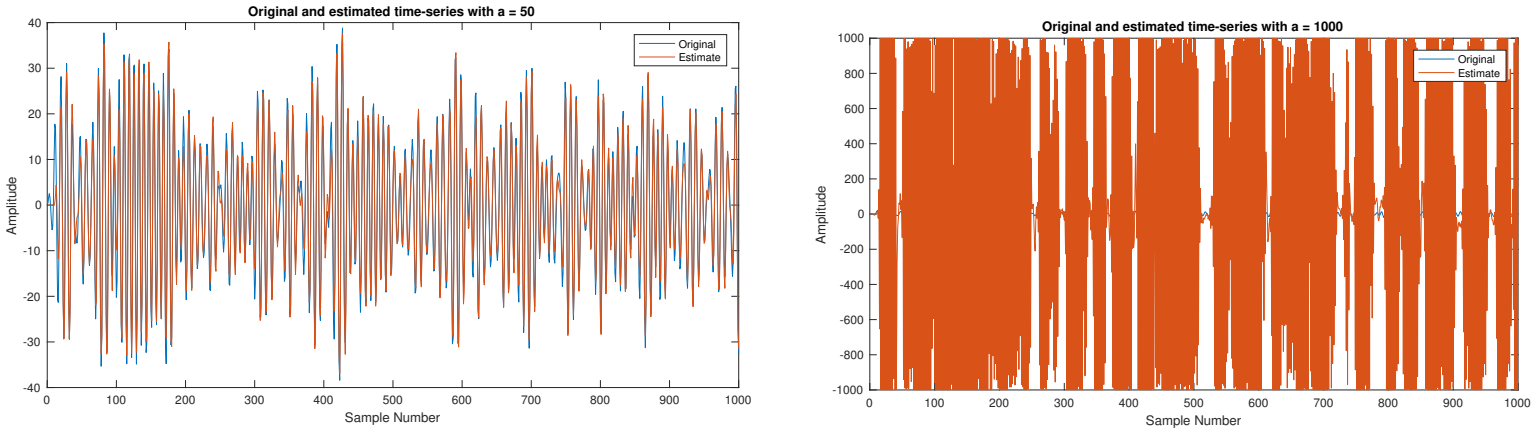

Figure 40: Original and estimated time-series obtained using *dynamical perceptron* with varying a values ($\mu = 1 \times 10^{-5}$, $\phi = tanh$)

4. Now instead of removing the mean from the data, the original data is kept as it is and a bias is added to the non-linearity component to compensate for the non-zero mean.
The estimate for non-zero mean data obtained is presented in Figure 41. However from table 5, we can see that the prediction gain is not as high as the previous case with the zero mean data estimation using a dynamical perceptron. This can be explained by the fact that the length of the signal was not enough for the bias to properly converge to the correct value. The convergence of the bias seems to be quite slow and only reached the value of 0.0066. This might indicate that this weight value for the bias still needs to increase before being sufficiently important to account for the mean.

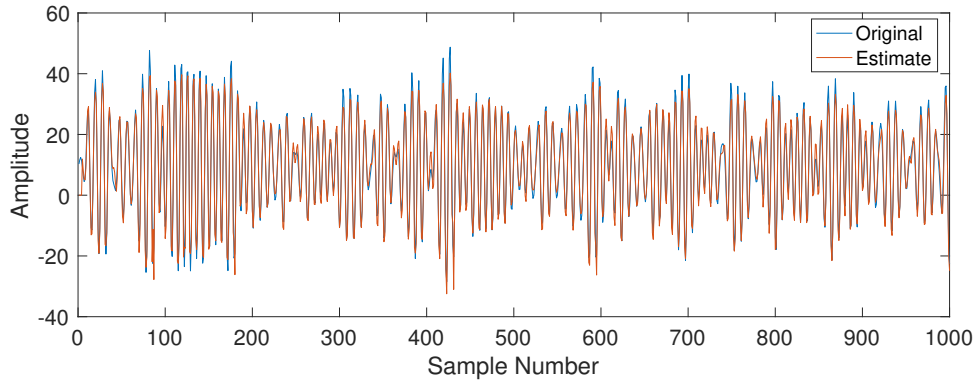

Figure 41: Original and estimated time-series obtained using *dynamical perceptron* ($\mu = 1 \times 10^{-5}$, $\phi = tanh$, a = 50 and b = 1)

5. To deal with this slow convergence of the bias, weights can be pre-trained by only using 20 first samples and iterating the process a 100 times in this case. The weights obtained are used then as initial weights for the algorithm.
Comparing Figure 41 and Figure 42 from n = 100:200, that the bias is better adjusted with pre-trained weights. In this case, the prediction gain is almost identical to the dynamical perceptron without pre-trained weights and zero-mean as the difference is only of 0.2%. The weight of the bias was able to completely converge thanks to the iterations. The pre-trained weights method is a way to get accurate weights value from a sample number in case the data available for training is relatively small.

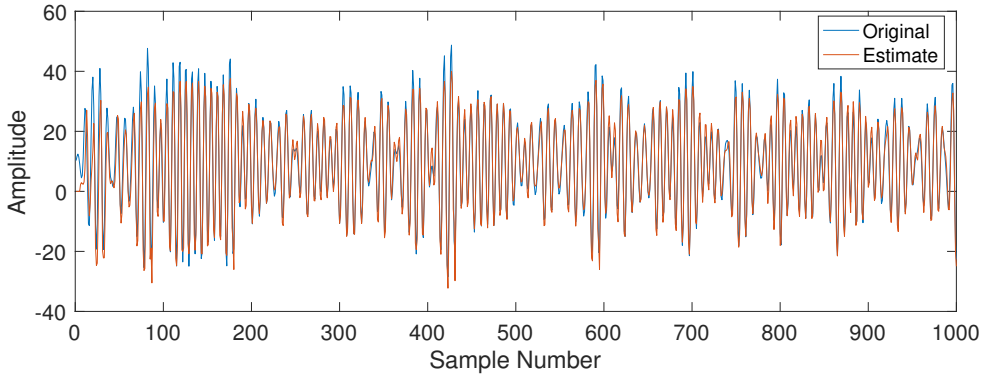

Figure 42: Original and estimated time-series obtained with pre-trained weights ($\mu = 1 \times 10^{-5}$, $\phi = tanh$, a = 50 and b = 1)

The following table summarizes the MSE and prediction gain of the different methods evaluated previously.

Method	**MSE**	**Prediction gain R_p**
LMS (1)	40.09	5.20
Dynamical perceptron (a*tanh) (3)	6.55	15.06
With bias (4)	13.71	12.22
With Pre-trained weights (5)	7.08	15.02

Table 5: MSE and prediction gain for the estimates obtained with the different methods encountered

From the table above, we can see that the most error is observed with the simple LMS algorithm due to the longer transient period. To conclude, the best results are obtained with the dynamical perceptron for zero-mean data and pre-trained weights with bias for non-zero mean data.

6. A deep network is trained using a `backpropagation` algorithm based on gradient descent. Backpropagation corresponds to the algorithm for determining how a single training example would like the weights and biases to change. In other words, the aim is to determine how sensitive the cost function is to the weights and biases in order to know the most efficient adjustement in order to decrease this cost function. This is equivalent to optimizing the weights to correctly map arbitrary inputs to ouputs.
For simplicity of explanation, it is assumed that the deep network is composed of one output. The first step to perform backpropagation is to feed forward one training example. We get then an estimated value of the output. The prediction error is then calculated (actual value - predicted value). This error information is backpropagated through the network updating the weights and biases in each step to reduce the cost function (error). The equations to perform backpropagation are done in this order: the error in the output layer is computed, then the error in the layer l = L-1, L-2 and so on until the first layer is reached (which is why it is called backpropagation) is computed for each layer. This is thanks to an equation that gives the error in a layer in terms of the error in the next layer. The output is then the rate of change of the cost with respect to any bias in the network and the rate of change of the cost with respect to any weight in the network.
The process is repeated for every training data for a certain number of iterations (epochs) in order to minimize the cost function. The calculations performed correspond to the gradient descent method. The final weights and biases obtained are then used to estimate output of the test input data. The number of weights and biases is determined by the number of layers and the number of neurons in each layer.

7. Looking at the test performance against epoch numbers (loss function) for LMS, non-linear LMS and deep network with default parameters, we can see that LMS and non-linear LMS stabilize faster than deep network but converge to a higher value. Deep network has a sharp decrease followed by a considerable

increase and only after 5500 epoch does it stabilizes. non-linear LMS has a slight smaller convergence value than LMS but deep network outperforms both of them. This is also visible in the output plot as deep network seem to follow more closely the curve of the test data than LMS and non-linear LMS which only follow the very general trend. This is due to the fact that deep network enable more weights to be available to approximate better the data reducing the cost function more accurately than with a dynamical perceptron.

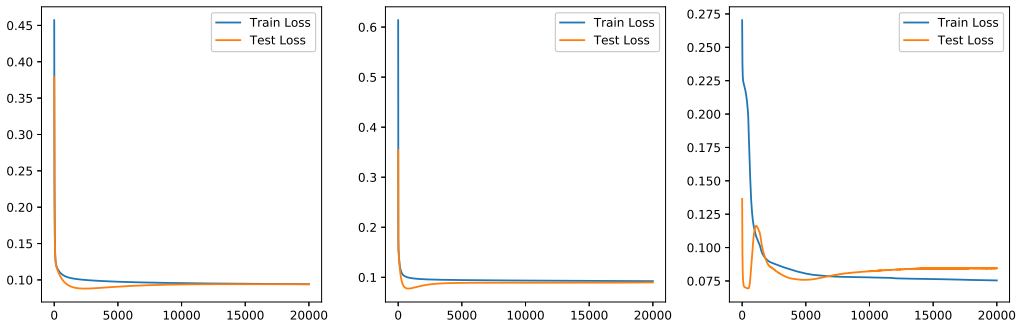

Figure 43: Loss function for dynamical perceptron (LMS and non-linear LMS) and deep network

8. To see the effect of noise power, the value was increased to 0.1 and then 0.5. With 0.1, we can see that for the deep network the convergence loss value of test loss is a bit higher than with 0.05. However it is still a better approximate then the dynamical perceptron. A considerable difference is observed for 0.5 noise power. In this case as the value of epochs increases, the deep network has a high test loss value and LMS and non-linear LMS perform better. This is due to the fact that increasing epoch number can lead to overfitting of the training data especially if the data is noisy which then gives incorrect test data values. It will do well for the training data but not the test data. Therefore, deep network are effective if the training data is not noisy and accurate. If the input data in training is already noisy, then LMS and non-linear will yield better results than the deep network. Deep network are useful if the training data is accurate and not noisy. Another disadvantage observed with deep network is that it is computationally time consuming as the cost function takes longer to compute then with a simple dynamical perceptron

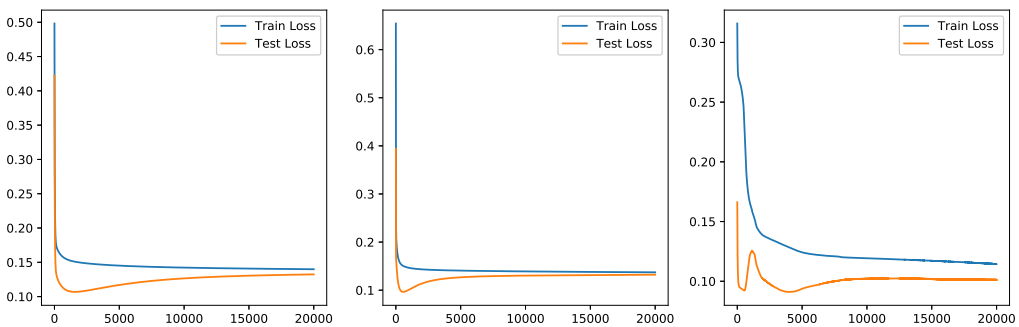

Figure 44: Loss function for LMS, non-linear LMS and deep network with noise power equal to 0.1

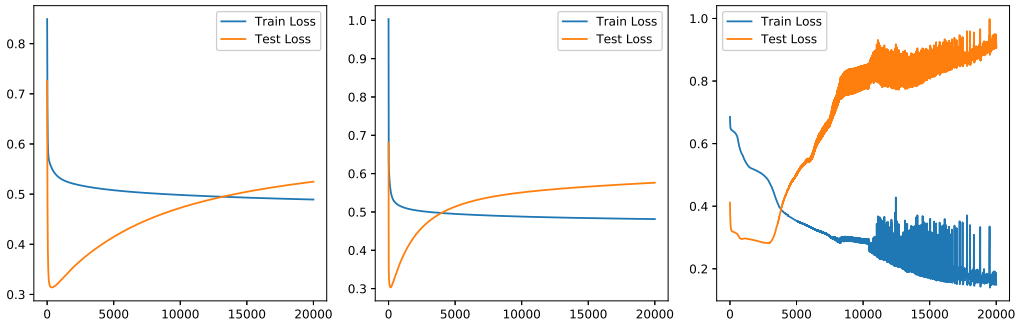

Figure 45: Loss function for LMS, non-linear LMS and deep network with noise power equal to 0.5

5 Tensor Decompositions for Big Data Applications

The notebooks corresponding to Assignments 1, 2 and 3 of this part are in the folder Part 5 available in the zip file.

NOTE: The MATLAB code written for each sections are saved in the folder ”Code”. Their names are ”PART_number_of_section” In the same folder are also the functions used in the different assignments.

Appendices

A Appendix

The following Appendix corresponds to part 2.3.1 of my Coursework Advanced Signal Processing:
The conditions for an AR(2) process to be stable are:

$$\begin{cases} a_1 + a_2 < 1 \\ a_2 - a_1 < 1 \\ -1 < a_2 < 1 \end{cases} \tag{47}$$

These conditions are determined by the correlation coefficient $\rho(1)$ and the variance of an AR(2) process. Indeed, from the Yule-equations for p=2, we know that:

$$\rho(1) = \frac{a_1}{1 - a_2} \tag{48}$$

Since $\rho(1) < \rho(0) = 1$ (as $\rho(0)$ is the normalized Autocorrelation function at 0) we have the following condition:

$$\frac{a_1}{1 - a_2} < 1 \Leftrightarrow a_1 + a_2 < 1 \tag{49}$$

which corresponds to condition 1.
Furthermore, the variance, which is derived from the general AR(2) process equation, is equal to:

$$\sigma_x^2 = \left(\frac{1 - a_2}{1 + a_2}\right) \frac{\sigma_w^2}{(1 - a_2)^2 - a_1^2} = \frac{(1 - a_2)\sigma_w^2}{(1 + a_2)(1 - a_2 - a_1)(1 - a_2 + a_1)} \tag{50}$$

The variance is always positive. Therefore, the numerator and the denominator need to be of the same sign.

$$\text{If} \quad 1 - a_2 > 0 \quad \text{then} \quad (1 + a_2)(1 - a_2 - a_1)(1 - a_2 + a_1) > 0 \tag{51}$$

This happens if in the denominator either all the factors are positive or 2 of them are negative and one positive. Due to the fact that $1 > a_2$ and condition 1, the only possibility for the denominator to be positive is if all the factors are positive. Therefore we have:

$$\begin{aligned} 1 + a_2 > 0 &\Leftrightarrow -1 < a_2 \\ 1 - a_2 + a_1 > 0 &\Leftrightarrow 1 > a_2 - a_1 \\ 1 - a_2 - a_1 > 0 &\Leftrightarrow 1 > a_2 + a_1 \end{aligned} \tag{52}$$

which correspond to our three conditions.