Imperial College London

IMPERIAL COLLEGE LONDON

PROFESSOR DANILO MANDIC

Adaptive Signal Processing and Machine Intelligence Coursework

Manuela Giraud (CID: 01513890)

Date: August 7, 2022

1 Classical and Modern Spectrum Estimation

1.1 Properties of Power Spectral Density (PSD)

It is possible to demonstrate that the definition of PSD given in equation (1) is equivalent to the one given in (2) under the assumtion that the covariance sequence r(k) decays rapidly (equation (3))

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

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

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

By expanding (2), we get:

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

Assuming that the process is stationary (ie $E\{x(n)x^*(m)\} = r(n-m)$):

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

The expression inside the paranthesis can be decomposed into three sums (First term, second term, third term):

$$\sum_{n=0}^{N-1}\sum_{m=0}^{n-1} f(n-m) + \sum_{n=0}^{N-1} f(0) + \sum_{n=0}^{N-1}\sum_{m=n+1}^{N-1} f(n-m) \tag{5}$$

■ First term ($\tau = n - m > 0$):

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

$$\Rightarrow \sum_{\tau=1}^{N-1}\sum_{n=\tau}^{N-1} f(\tau) = \sum_{\tau=1}^{N-1} f(\tau)\sum_{n=\tau}^{N-1} 1 = \sum_{\tau=1}^{N-1}(N-1-\tau+1)f(\tau) = \sum_{\tau=1}^{N-1}(N-\tau)f(\tau) = \sum_{\tau=1}^{N-1}(N-|\tau|)f(\tau)$$

Since $\tau > 0$, we can replace τ by $= |\tau|$.

■ Second term:

$$\sum_{n=0}^{N-1} f(0) = Nf(0) = \sum_{\tau=0}^{0}(N-|\tau|)f(\tau)$$

■ Third term ($\tau = n - m < 0$):

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

$$\Rightarrow \sum_{\tau=-(N-1)}^{-1}\sum_{n=0}^{\tau+(N-1)} f(\tau) = \sum_{\tau=-(N-1)}^{-1}(\tau+N-1+1)f(\tau) = \sum_{\tau=-(N-1)}^{-1}(\tau+N)f(\tau) = \sum_{\tau=-(N-1)}^{-1}(N-|\tau|)f(\tau)$$

Since $\tau < 0, \tau = -|\tau|$.

Replacing the sum of the three terms back into equation (4):

$$\begin{aligned} P(\omega) &= \lim_{N\to\infty}\frac{1}{N}\left(\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} r(n-m)e^{-j\omega(n-m)}\right) = \lim_{N\to\infty}\frac{1}{N}\left(\sum_{\tau=-(N-1)}^{N-1}(N-|\tau|)r(\tau)e^{-j\omega\tau}\right) \\ &= \lim_{N\to\infty}\left(\sum_{\tau=-(N-1)}^{N-1} r(\tau)e^{-j\omega\tau}\right) - \lim_{N\to\infty}\frac{1}{N}\left(\sum_{\tau=-(N-1)}^{N-1}|\tau|r(\tau)e^{-j\omega\tau}\right) \end{aligned} \tag{6}$$

The triangle inequality states that:

$$0 \le \lim_{N\to\infty}\left|\frac{1}{N}\sum_{\tau=-(N-1)}^{N-1}\right|\tau\left|r(\tau)e^{-j\omega\tau}\right| \le \lim_{N\to\infty}\frac{1}{N}\sum_{\tau=-(N-1)}^{N-1}|\tau|\left|r(\tau)e^{-j\omega\tau}\right|$$

Using the rapid decay assumption in (3):

$$0 \le \lim_{N\to\infty}\left|\frac{1}{N}\sum_{\tau=-(N-1)}^{N-1}\right|\tau\left|r(\tau)e^{-j\omega\tau}\right| \le 0$$

Hence, the second term in (6) is 0 and we get:

$$P(\omega) = \lim_{N\to\infty}\left(\sum_{\tau=-(N-1)}^{N-1} r(\tau)e^{-j\omega\tau}\right)$$

We have thus demonstrated that equations (1) and (2) are equivalent under the assumtion that the covariance sequence r(k) decays rapidly.

A simulation of a sinusoid (slowly decaying covariance sequence) and a delta function were run on Matlab, and the periodograms for the two different definitions of PSD are plotted for the two signals in figure (1). We can see that the equality derived above holds true for the impulse signal since it meets the condition of a fastly decaying covariance sequence, but not for the sinusoid which has a slower decay.

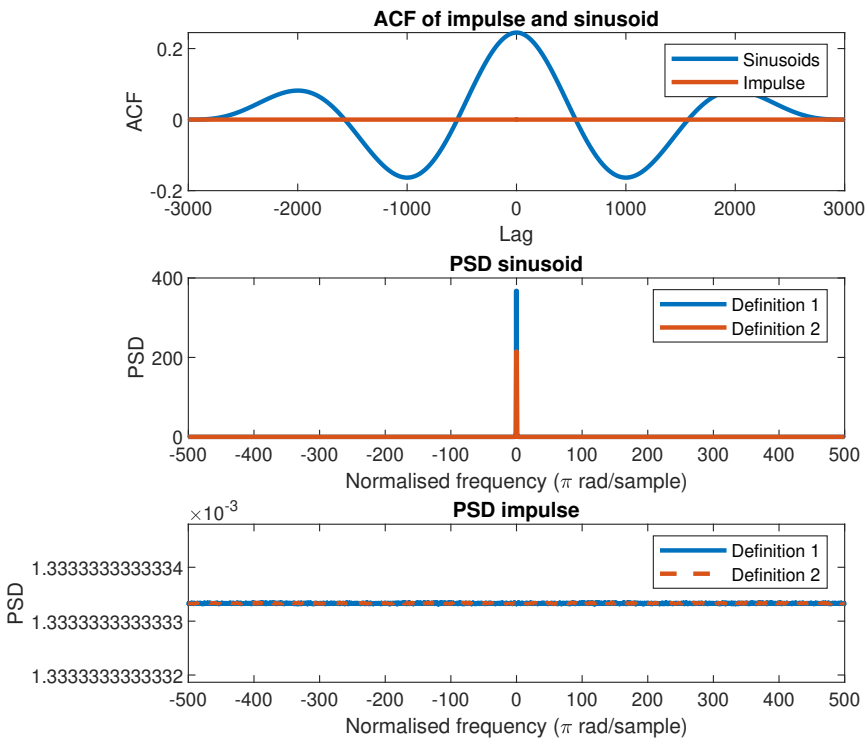


Figure 1: Periodograms

1.2 Periodogram-based Methods Applied to Real-World Data

(a) A periodogram-based spectral estimation was applied to the sunspot time series (figure 2). After centering and de-trending the series, the PSD of the data remains similar to that of the data with DC offset. At w = 0 however, the centered data has a PSD of 0 while the raw data PSD is non-zero due to a DC offset.

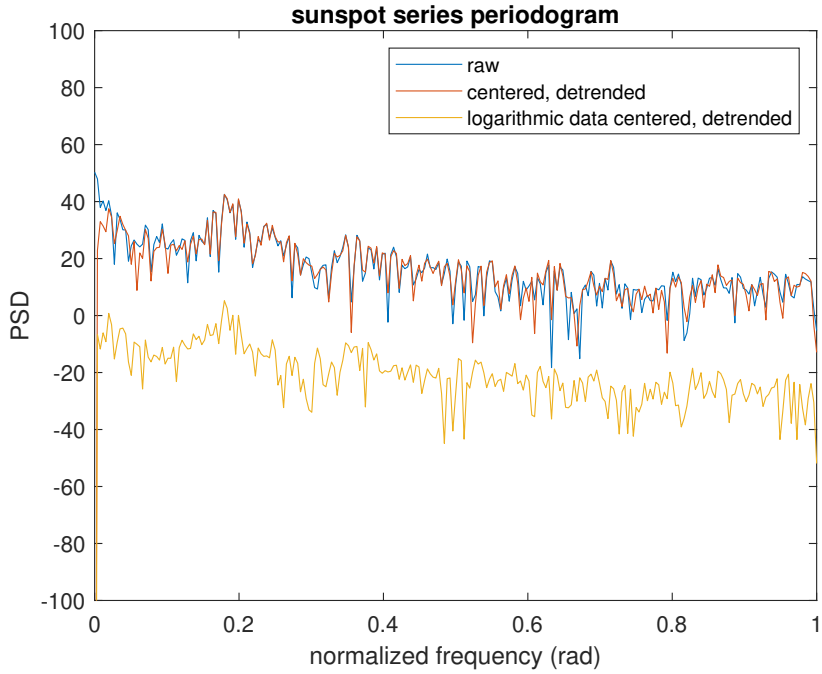


Figure 2: Periodograms of the raw and preprocessed sunspot time series

b) Given EEG recordings of the POz region of the head, we can perform spectral analysis on the data to identify the fundamental frequencies of the SSVEP response. The standard periodogram approach using Welch's method with a Hamming window and zero overlap was applied to the data for varying window lengths (1s, 5s, 10s) and the results are shown in figure (3). From the plots, we can identify peaks at frequencies 9Hz, 13Hz, 26 Hz and 50 Hz. The lower frequency peak corresponds to the alpha-rhythm and the 50 Hz peak is due to power-line interference in the recording apparatus. Therefore the 13 Hz peak corresponds to the SSVEP fundamental frequency f_0, with harmonics at frequencies $k*f_0$ (for all integer k).

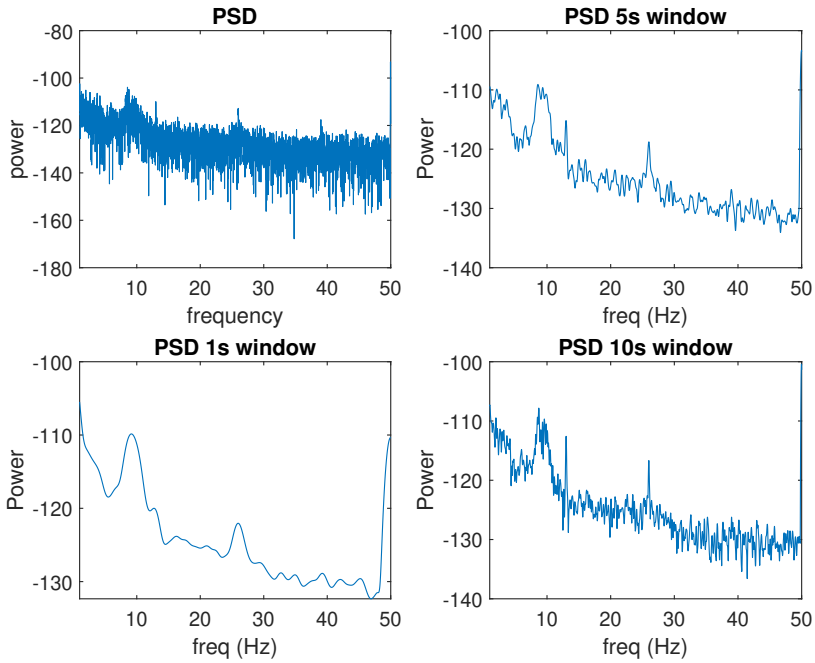


Figure 3: Periodograms of EEG data for the standard method and for Welch's method at different window sizes

1.3 Correlation Estimation

(a) A correlogram for biased and unbiased estimates of autocorrelation for various signals (noisy sinusoid, filtered and un-filtered WGN) was computed and implement into a spectral density estimator. The results are plotted in figure (4). The biased estimates display less variance than unbiased ones due to ACF estimation errors for large lags in the unbiased case.

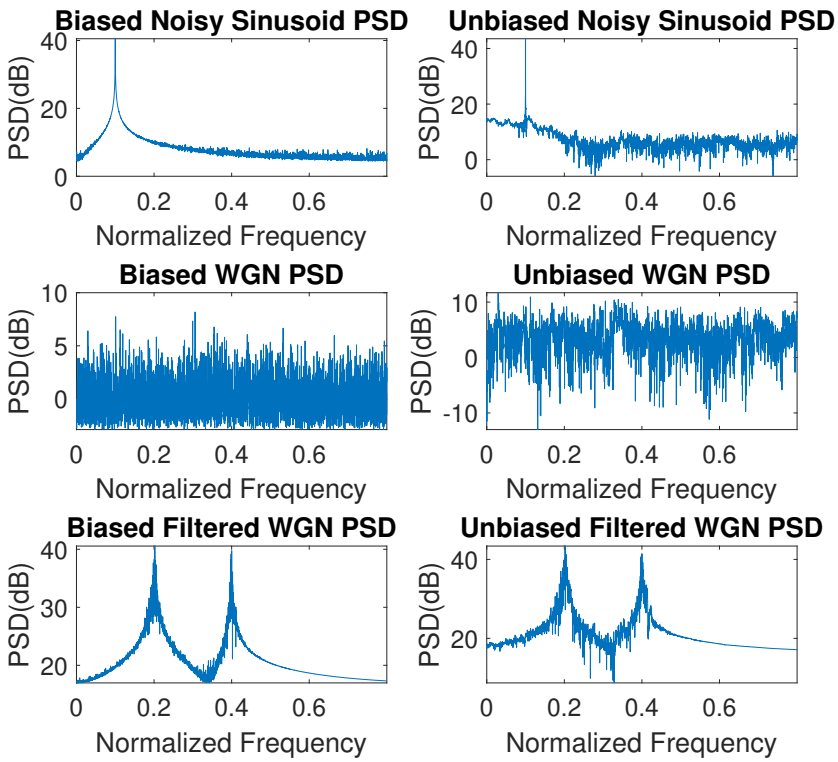


Figure 4: PSD estimates

(b and c) 100 realisations of a sinusoid with random noise were generated and their PSD was plotted in figure (5) along with the mean and standard deviation. When scaling the process in dB, peaks and noise patterns become more visible.

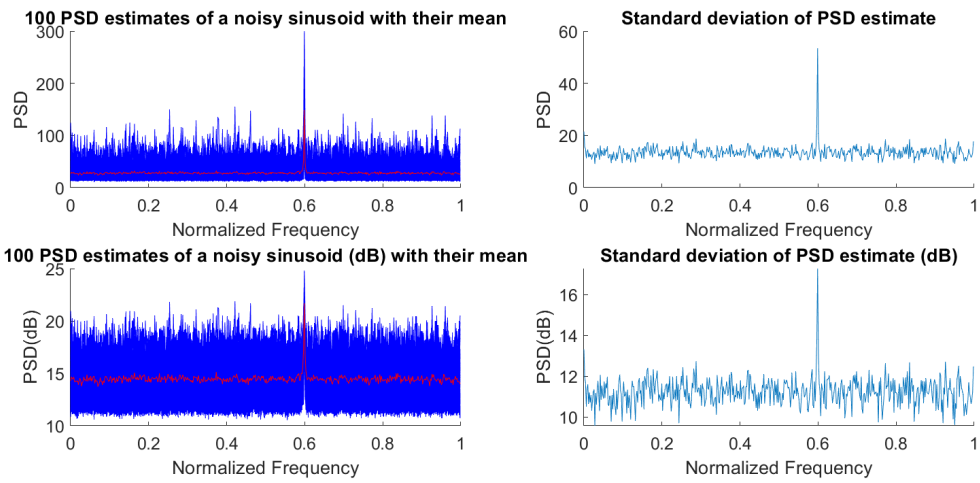


Figure 5: PSD estimates

(d) PSD estimations of complex exponentials were implemented for varying numbers of samples and plotted in figure (6). For 10 data samples, the estimation fails

to detect close-by frequency peaks. A sufficient number of samples is therefore necessary.

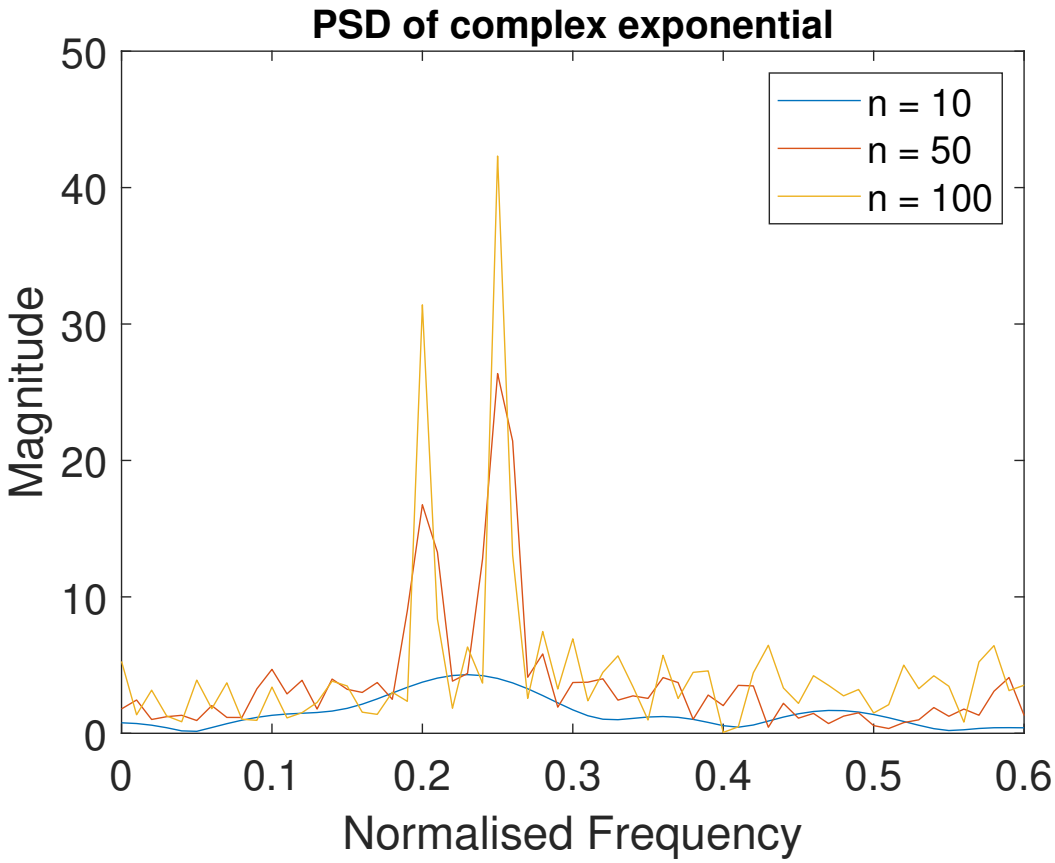


Figure 6: PSD estimates of 100 realisations of a noisy sinusoidal signal with the mean (left) standard deviation of the estimates (right)

(e) The multiple signal classification (MUSIC) algorithm uses subspace classification to perform PSD estimates. It was implemented in matlab using the functions corrmtx and pmusic.
Corrmtx(x, 14, 'mod') finds the biased autocorrelation matrix of signal x with an AR model of order 14.The pmusic function will then use the found autocorrelation matrix to perform the music algorithm. The function assumes knowledge of the subspace dimension which here is 2, to decompose the data.
MUSIC and standard spectrum estimations for the complex exponential introduced in the previous section are compared in figure (7). The resolution of the MUSIC algorithm is greater than the standard method for a given number of samples equal to 30. Indeed the two frequency peaks of the signal are detected with MUSIC but not with the standard periodogram. Such precision comes at a computational cost, the first estimation taking almost 4 times as much time to compute.

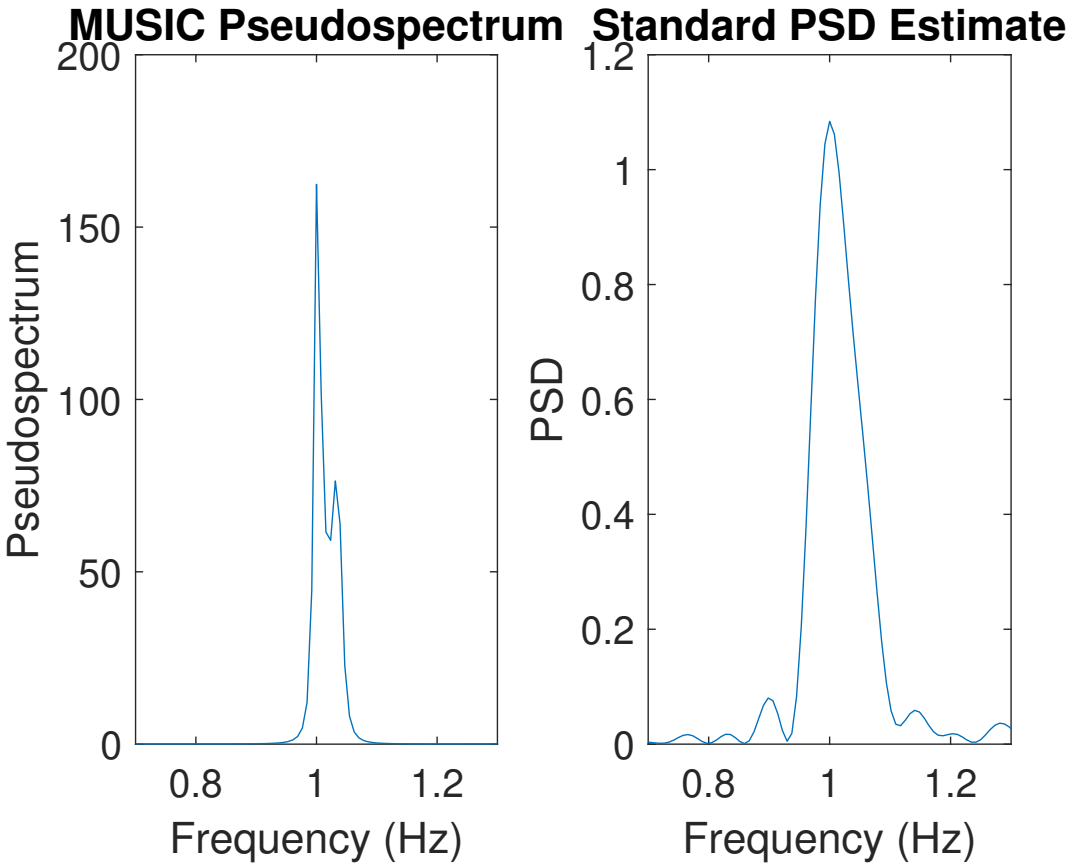


Figure 7: PSD estimates of a complex exponential for samples numbers 10,50 and 100.

1.4 Spectrum of Auto-Regressive Processes

(a) The AR parameters can be found using the inverse Yule-Walker equation: $a = R_{xx}^{-1} r_{xx}$ which holds only for invertible matrices Rxx. The unbiased ACF estimate does not assure an invertible Rxx so the AR parameters might not always be computable. (b and c) An autoregressive process defined by x(n) in equation (7) was generated over 1000 and 10,000 samples.

$$x(n) = 2.76x(n1)3.81x(n2) + 2.65x(n3)0.92x(n4) + w(n) \tag{7}$$

The power spectrum density of the steady-state signal (discarding the transient response of the first 500 samples) was estimated using model orders p =2,...,14. Some of these estimates are plotted in figure (8). Increasing the model allows the distinction of frequency peaks that are close-by, but it can lead to model overfitting, as testified by the oscillations at higher frequencies for p = 14. When we increase the number samples to 10,000, estimations perform better for lower orders: a model with p=4 can detect the two frequency peaks present in the true PSD whereas it only detected one when using 1000 samples. The choice of model orders therefore depends on the number of samples. All model orders for longer simulations also provide better estimates in general, and lower model orders perform as well as higher ones.

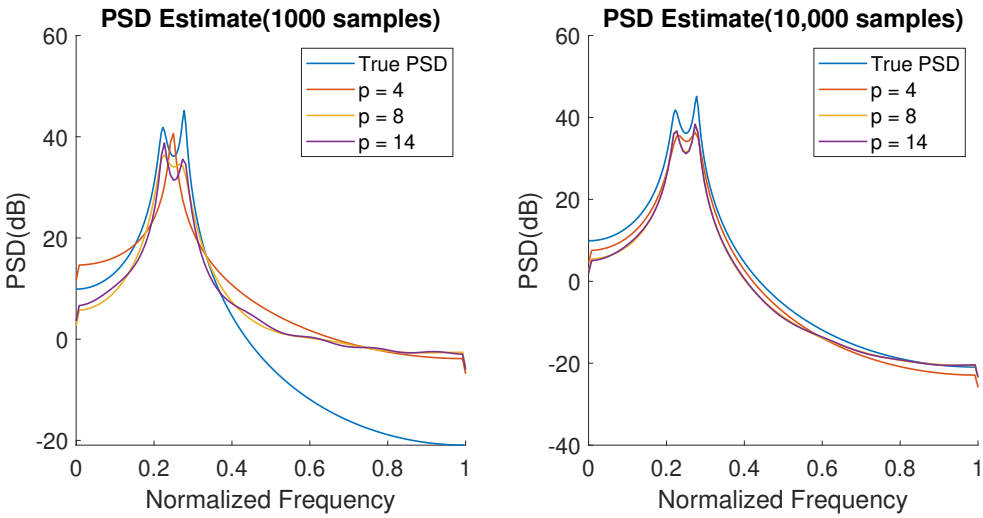


Figure 8: PSD estimates for 1000 and 10,000 samples

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

(a) We were provided with three sets of ECG data corresponding to different trials of a patient breathing. The Power spectrum density of the three trials were estimated using the standard periodogram and averaged periodograms, and are plotted in figure (**??**).

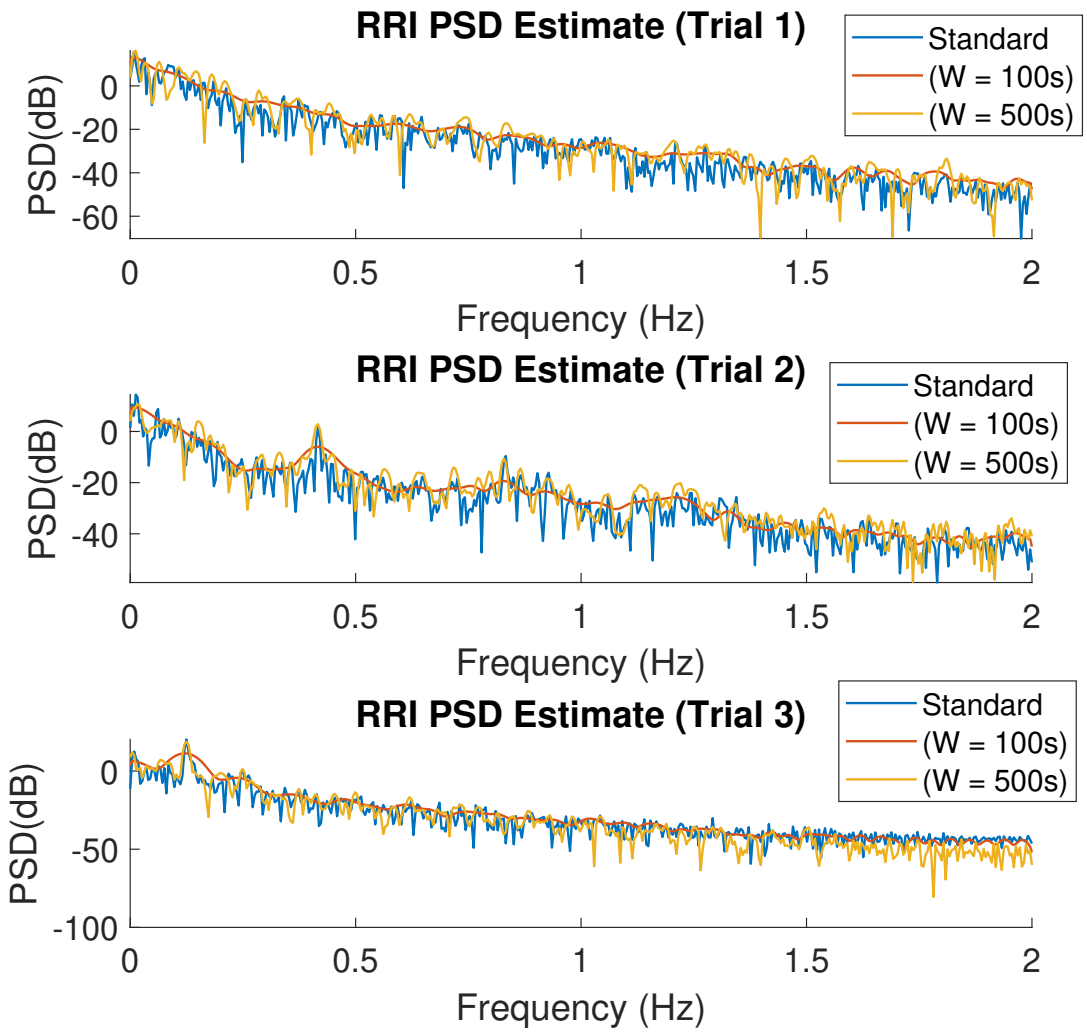


Figure 9: PSD estimations of EEG data for 3 different trials

(b) Frequency peaks for trials 2 and 3 can be observed at 0.42Hz and 0.12 Hz respectively, corresponding to the breathing rate of the patient during the trials. No clear peak is visible for the first trial, suggesting that the patient may have an irregular breathing pattern.

(c) After determining optimal model orders for the three trials, AR spectral estimates were found with the pyulear function in matlab, and plotted in figure (**??**).

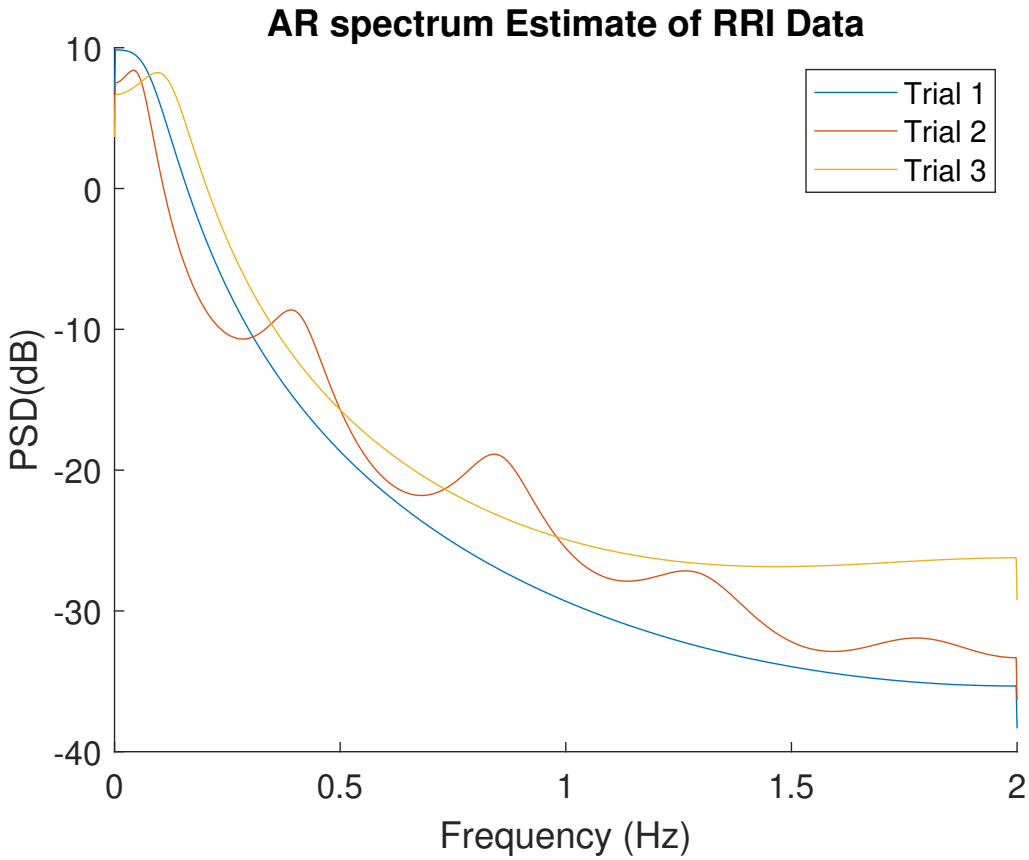


Figure 10: PSD estimations of EEG data for 3 different trials

1.6 Robust Regression

(a) The singular values of a training data matrix $\mathbf{X}$ and the noise corrupted matrix $\mathbf{X}_{noise}$ are shown in figure (**??**). The noise-free matrix has three non-zero Singular values and is therefore of rank 3. Comparatively for the noisy data all subspaces are non-zero. Its rank is still identifiable because the magnitude of the SVs in the signal subspace is significantly larger than the SVs in the noise subspace. When the noise increases, the difference between the subspaces will be blurred and it will not be possible to find the rank of $\mathbf{X}$ from $\mathbf{X}_{noise}$.
The square error between the two signals for each subspace dimension plotted in (**??**) is greater for dimensions higher than the identified rank for $\mathbf{X}$.

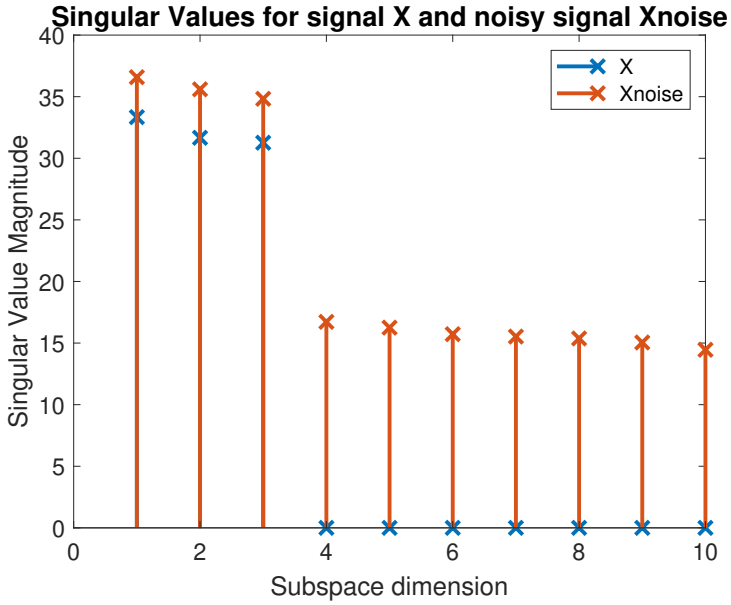


Figure 11

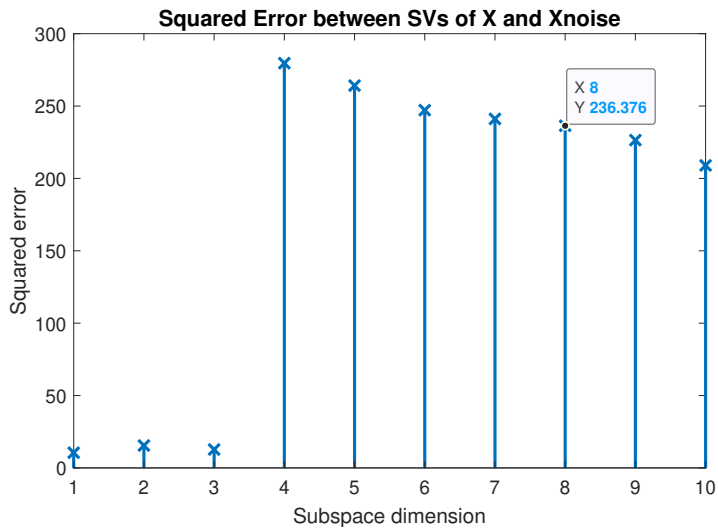


Figure 12

(b) To remove the noise introduced in $\mathbf{X}_{noise}$, a low-rank approximation $\tilde{\mathbf{X}}_{\text{noise}}$ was made using only the 3 most significant components of the matrix. The MSE between the clean input $\mathbf{X}$ and $\tilde{\mathbf{X}}_{\text{noise}}$ is compared to that with $\mathbf{X}_{noise}$ in figure (**??**). As expected, the error decreases when the noisy subspaces have been removed.

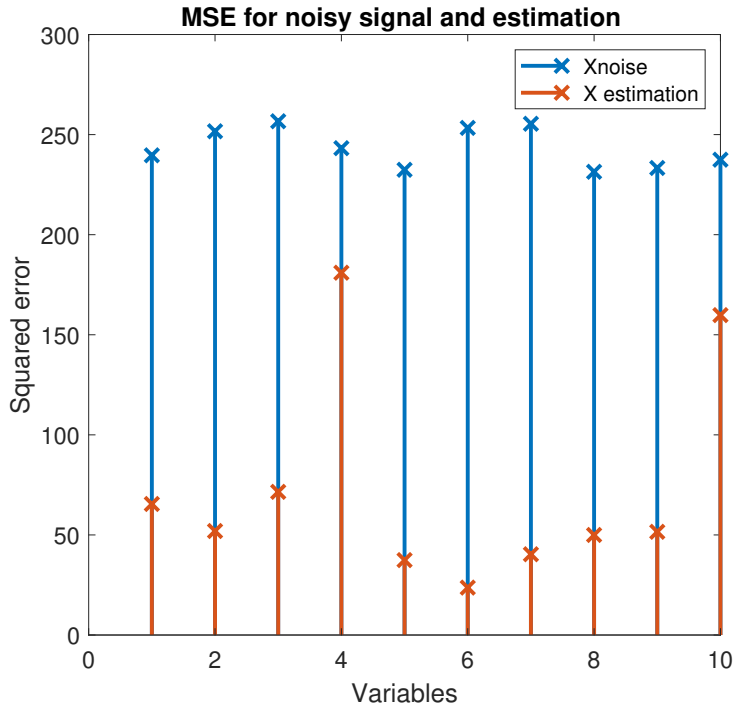


Figure 13: Squared Error between noiseless input matrix noise corrupted matrix or denoised matrix

(c) The ordinary least squares (OLS) and principal component regression (PCR) solutions for the parameter matrix $\mathbf{B}$ (equation (8)) were computed on matlab and were used to find the estimation error between $\mathbf{Y}$ and $\hat{\mathbf{Y}}_{\text{OLS}} = \mathbf{X}_{\text{noise}}\,\hat{\mathbf{B}}_{\text{OLS}}$ and $\hat{\mathbf{Y}}_{\text{PCR}} = \tilde{\mathbf{X}}_{\text{noise}}\,\hat{\mathbf{B}}_{\text{PCR}}$. The sults are plotted in figure (8), and show negligible differences between OLS and PCR. When data from the test set is estimated with regression coefficients computed from the training set, the square error is generally less than when the same training set is used to find an estimation.

$$\begin{aligned} \hat{\mathrm{B}}_{\text{OLS}} &= \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Y} \\ \hat{\mathbf{B}}_{\text{pcr}} &= \mathbf{V}_{1:\mathrm{r}}\left(\Sigma_{1:\mathrm{r}}\right)^{-1}\mathrm{U}_{1:\mathrm{r}}^{\mathrm{T}}\mathbf{Y} \end{aligned} \tag{8}$$

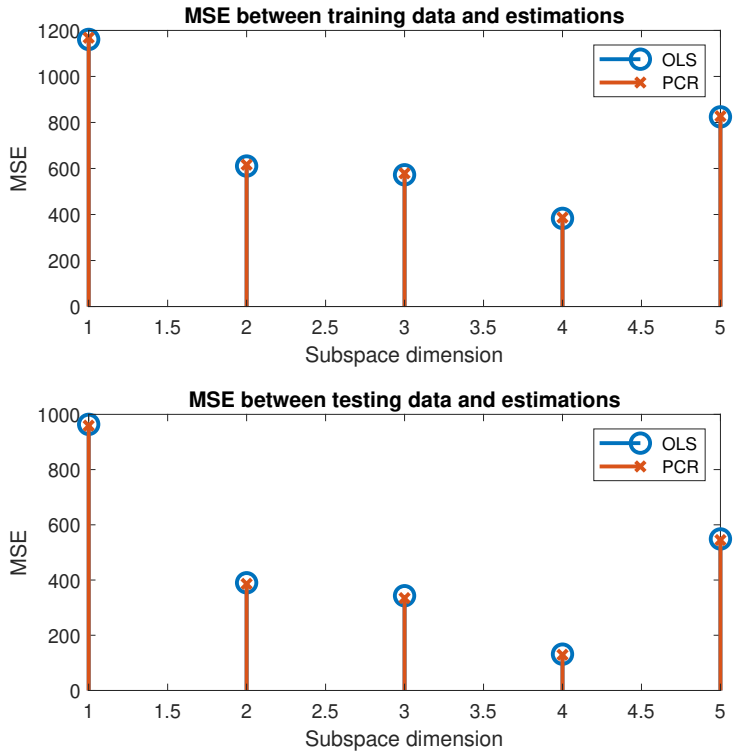


Figure 14: MSE of training and test data for OLS and PCR methods

(d) A broader set of data provided by the regval script was used to generate new realisations of estimates. The MSE for each output using the OLS and PCR methods is plotted in figure (**??**), and shows little difference between the two algoritms.

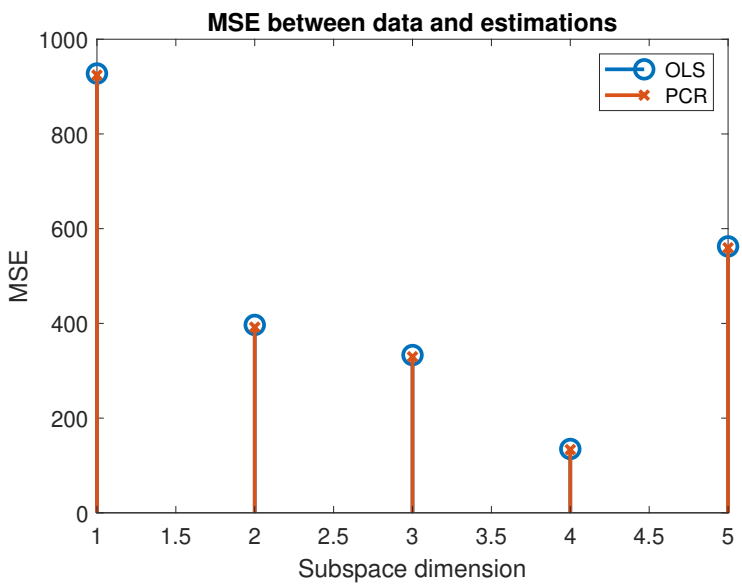


Figure 15: MSE of realisations of data and their estimates computed with regval from the OLS and PCR methods

2 Adaptive Signal Processing

2.1 The Least-Mean Square Algorithm (LMS)

(a) The correlation matrix is defined in (9)

$$R = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(1) & r_{xx}(0) \end{bmatrix} = \begin{bmatrix} \sigma_x^2 & r_{xx}(1) \\ r_{xx}(1) & \sigma_x^2 \end{bmatrix} \tag{9}$$

$$\sigma_x^2 = \frac{1-a_2}{1+a_2} \frac{\sigma_\eta^2}{(1-a_2^2)-a_1^2} \tag{10}$$

$$\rho_1 = \frac{r_{xx}(1)}{r_{xx}(0)} = \frac{r_{xx}(1)}{\sigma_x^2} = \frac{a_1}{1-a_2} \tag{11}$$

With

$$a_1 = 0.1, a_2 = 0.8 \tag{12}$$

and

$$\sigma_\eta^2 = 0.25 \tag{13}$$

we get:

$$\sigma_x^2 = 0.926, r_{xx}(1) = 0.463 \tag{14}$$

$$R = \begin{bmatrix} 0.926 & 0.463 \\ 0.463 & 0.926 \end{bmatrix} \tag{15}$$

The eigenvalues of R are $\lambda_1 = 0.463$ and $\lambda_2 = 1.389$
The LMS will converge to the mean for values of the step size μ in the range $0 < \mu < \frac{2}{\lambda_{max}}$, where λ_{max} is the largest eigenvalue of the correlation matrix R. Therefore we get:

$$0 < \lambda < 1.44 \tag{16}$$

(b) An LMS adaptive predictor filter was applied to the signal x(n)= $a_1 x(n-1) + a_2 x(n-2) + \eta(n)$. The filter updates the weights according to equation (17).

$$w(n+1) = w(n) + e(n)x(n), \quad n = 0, 1, ... \tag{17}$$

The squared prediction error $e^2(n)$ was calculated with step sizes $\mu = 0.05$ and $\mu = 0.01$ and is plotted in dB in figure (16). The learning curve was obtained by averaging the squared prediction error over 100 realizations of x(n). We can observe from the learning curves, that the filter estimation converges faster for larger step sizes but has greater error variability in the steady state.
(c) The misadjustment of the LMS can approximated by equation (18). Applying the formula with $\mu_1 = 0.05$ and $\mu_2 = 0.01$ gives $M_1 = 0.0463$ and $M_2 = 0.0093$. Estimates

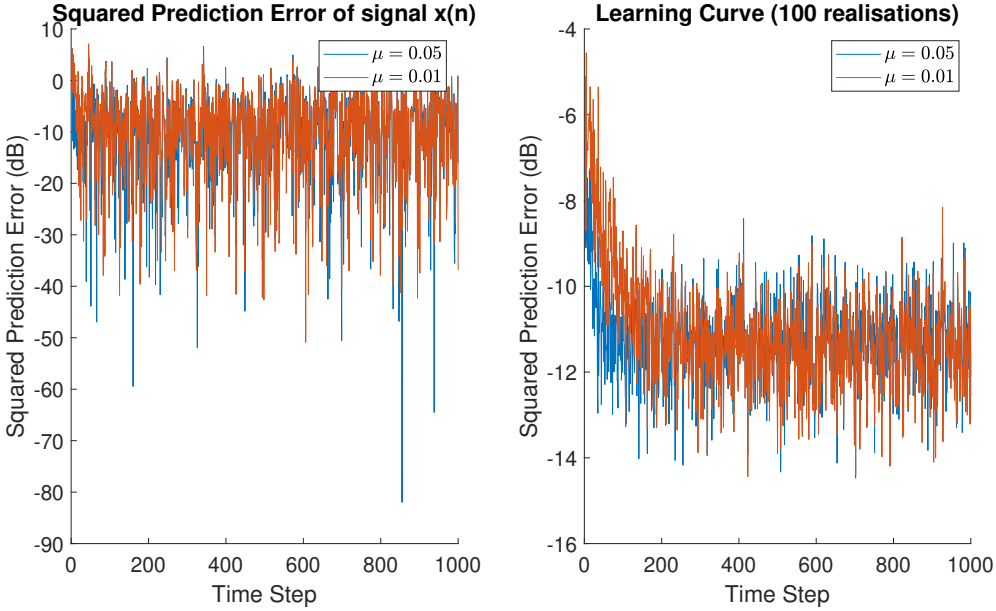


Figure 16: Squared error prediction (left) and learning curve over 100 realizations (right)

of M_1 and M_2 were also calculated by taking the time average of the steady state of the ensemble-averaged learning curves over 100 independent trials. The estimates obtained were $M_1 = 0.0532$ and $M_2 = 0.0087$, giving an average estimation error of 10.7%. The error for the estimation of M_2 however is only of 6.5% since the approximation in equation (refeqn2.1.c.1) holds for smaller step sizes so M_1 will be better approximated than M_2.

$$M \simeq \frac{\mu}{2} Tr(R) \quad \text{For small step sizes} \tag{18}$$

(d) Estimates of values of the adaptive filter coefficients a_1 and a_2 were obtained by averaging over the steady state of 100 trials of the LMS iterations. The results and corresponding errors for the different step sizes are presented in table 2.1.

	μ_1	μ_2	**real**
$\mathbf{a}_1$	0.0725	0.1139	0.1
$\mathbf{a}_2$	0.7030	0.7044	0.8
error a_1	27.%	13.9%	0
error a_2	12.1%	12.0%	0

(e) The leaky LMS algorithm forces the filter weights to converge by introducing a leakage coefficient γ into the weight update. The cost function J_2 (equation (19)) needs to be minimized to find the expression for the leaky LMS weight update given by the steepest gradient descent rule (equation (20)).

$$J_2(n) = \frac{1}{2}\left(e^2(n) + \gamma \|\mathbf{w}(n)\|_2^2\right) \tag{19}$$

with

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

Defining $d(n)$ as the desired signal and $x(n)$ as the input.

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu \nabla_{\mathbf{w}} J_2(n) \tag{20}$$

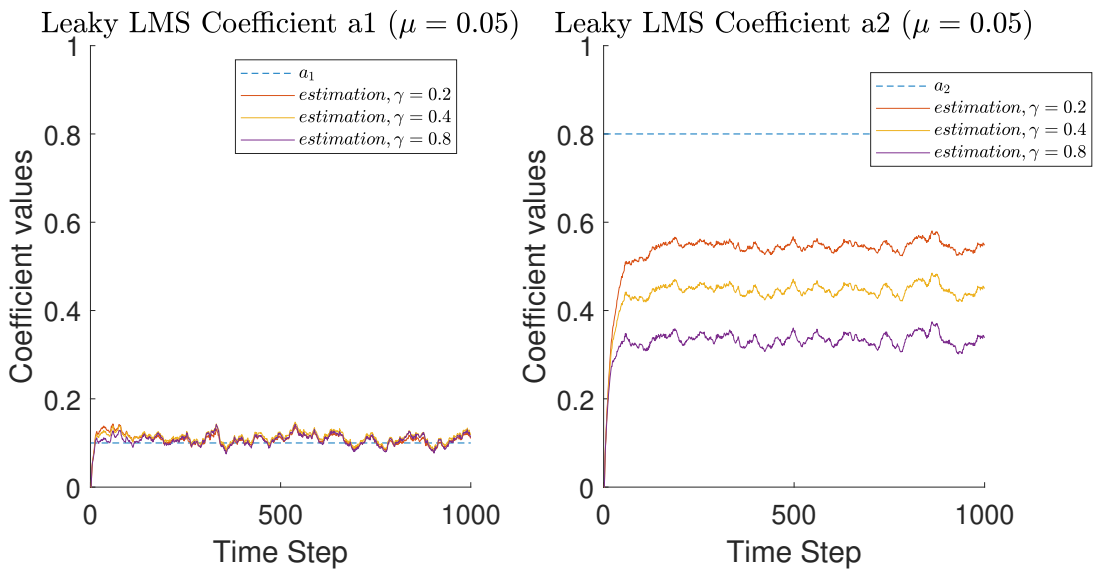
We can find the the gradient of the cost function J_2 with respect to w by replacing the expression of $\|w(n)\|_2$ and can then substitute this gradient in the weight update equation:

$$\|w(n)\|_2 = \sqrt{\mathbf{w}^T(n)\mathbf{w}(n)}$$

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

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

(f) The leaky LMS algorithm was implemented in Matlab with different step sizes and leakage coefficients to estimate AR coefficients of the signal x(n) from part (b), and the results are plotted in figure (17). The estimates get further away from the true coefficient values for increasing values of the leakage coefficient.



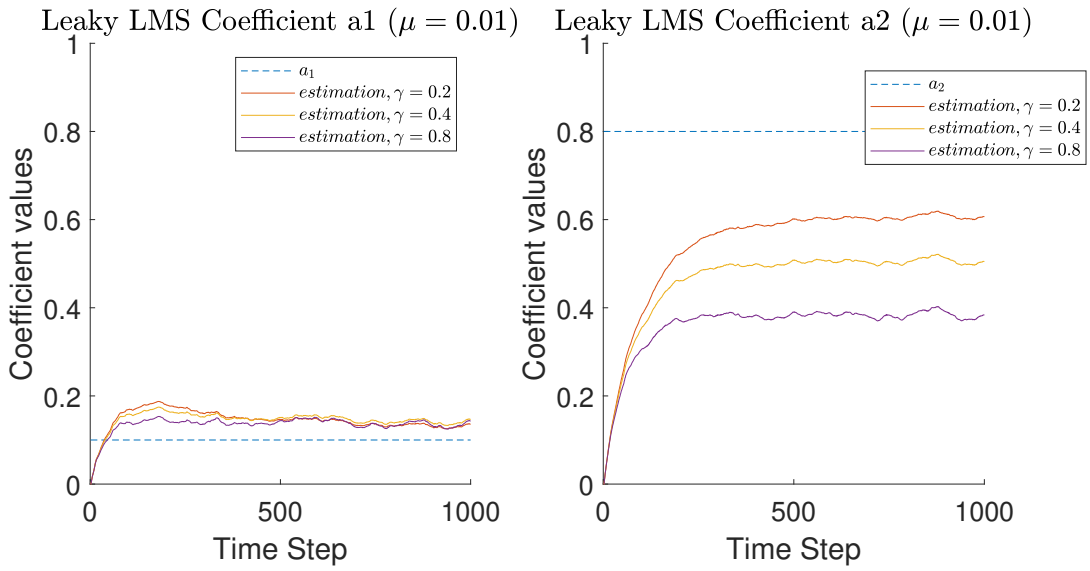


Figure 17: Leaky LMS coefficients estimations with varying values of γ and μ

2.2 Adaptive Step Sizes

(a) Adaptive learning rates can be used to combine fast approach to steady-state and minimal steady-state error. The learning rate $\mu(n)$ is made adaptive by the Gradient Adaptive Step-Size (GASS) algorithm (21). The value of $\psi(n)$ can be determined by the Benveniste model (22), the Ang Farhang model (23), or the Matthews Xie model (24).

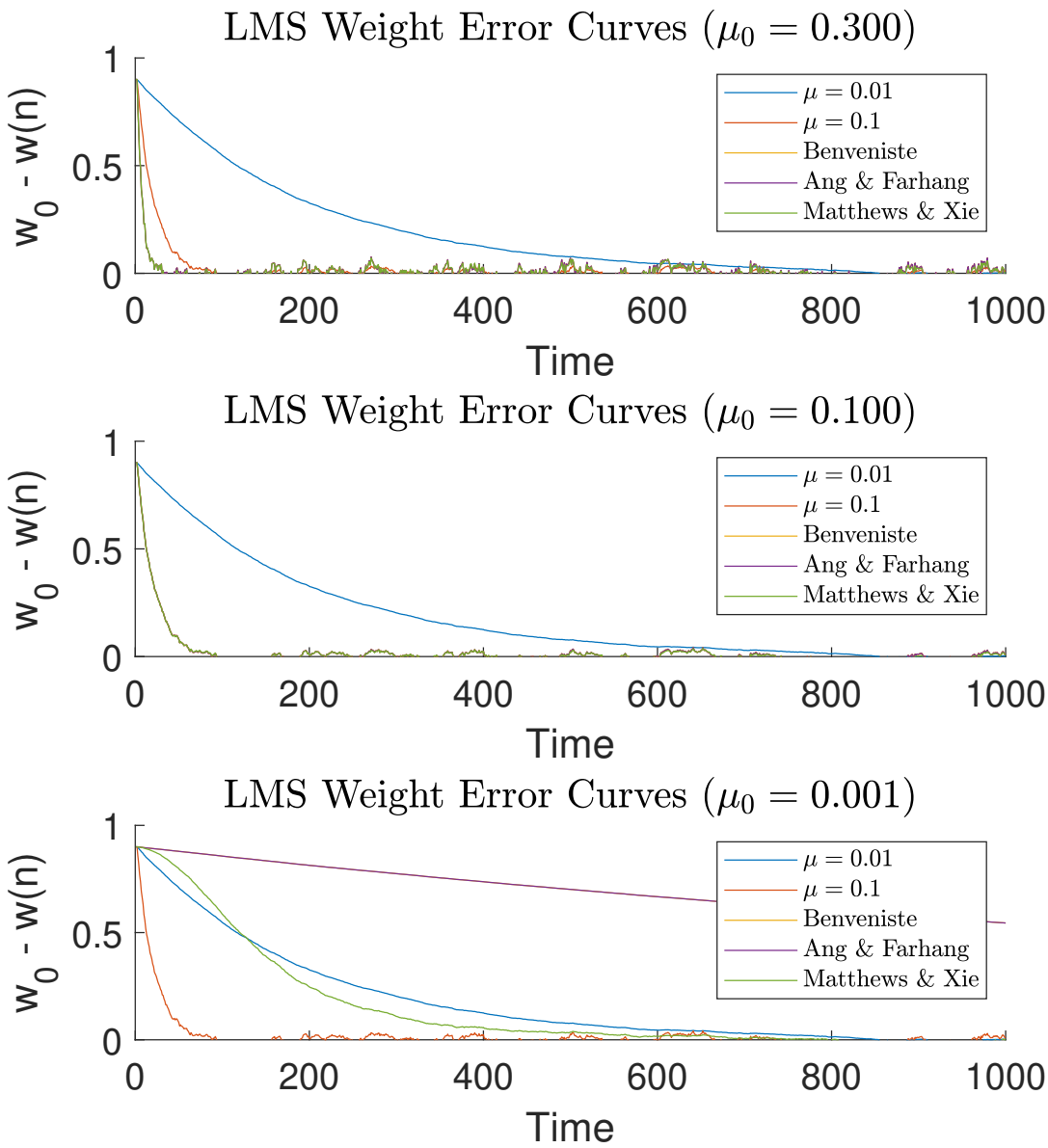
$$\mu(n+1) = \mu(n) + \rho e(n)\mathbf{x}^T(n)\psi(n) \tag{21}$$

$$Benveniste: \psi(n) = [\mathbf{I} - \mu(n-1)\mathbf{x}(n-1)\mathbf{x}^T(n-1)]\psi(n-1) + e(n-1)\mathbf{x}(n-1) \tag{22}$$

$$Ang\&Farhang: \psi(n) = \alpha\psi(n-1) + e(n-1)\mathbf{x}(n-1), 0 < \alpha < 1 \tag{23}$$

$$Matthews\&Xie: \psi(n) = e(n-1)\mathbf{x}(n-1) \tag{24}$$

These 3 algorithms were implemented in Matlab to estimate the weights of a real-valued MA(1) system x(n), over 1000 timesteps $x(n) = 0.9\eta(n1) + \eta(n)$ The weights error curves over time for different starting values of the step size are plotted in figure (**??**) along with the weight error curves of estimations obtained with non-adptive step sizes $\mu_1 = 0.01$ and $\mu_2 = 0.1$. The GASS algorithms all converger faster than the standard LMS for $\mu_0 = 0.3$, but this feature seems to disappear as the starting learning rate gets smaller. The steady-state error of the adaptive algorithms are also smaller than for a standard LMS of step size 0.01, but this difference reduces with the step size. The performance of an LMS with adaptive step sizes is therefore a reduced error, but it comes at the cost of computational complexity.



(b) The NLMS equation is given by:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\beta}{\epsilon + ||\mathbf{x}(n)||^2} e(n)\mathbf{x}(n) = \mathbf{w}(n) + \frac{\beta}{\epsilon + \mathbf{x}T(n)\mathbf{x}(n)} e(n)\mathbf{x}(n) \tag{25}$$

We want to verify that equation (25) is equivalent to:

$$w(n+1) = w(n) + \mu e_p(n)x(n) \tag{26}$$

with $e_p(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n+1)$ the a posteriori error.

We can start by using the update rule to replace $\mathbf{w}$(n+1) in the expression for $e_p(n)$:

$$\begin{aligned} e_p(n) &= d(n) - \mathbf{x}^T(n)(\mathbf{w}(n) + \mu e_p(n)\mathbf{x}(n)) \\ e_p(n)(1 - \mu\mathbf{x}^T(n\mathbf{x}(n))) &= d(n) - \mathbf{x}^T(n)\mathbf{w}(n) \\ e_p(n)(1 - \mu\|\mathbf{x}(n)\|_2^2) &= e(n) \\ e_p(n) &= \frac{e(n)}{(1 - \mu\|\mathbf{x}(n)\|_2^2)} \end{aligned} \tag{27}$$

Substituting equation (27) into equation (26) yields:

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

Equations (25) and (26) are equivalent for $\epsilon = \frac{1}{\mu}$ and $\beta = 1$. .

(c) Weight estimates of signal x(n) are obtained after implementing the GNGD and Benveniste algorithms on matlab, and the results are plotted over time in figure (**??**). The GNGD algorithm converges faster than the Benveniste implementation, however it has a higher steady-state error.

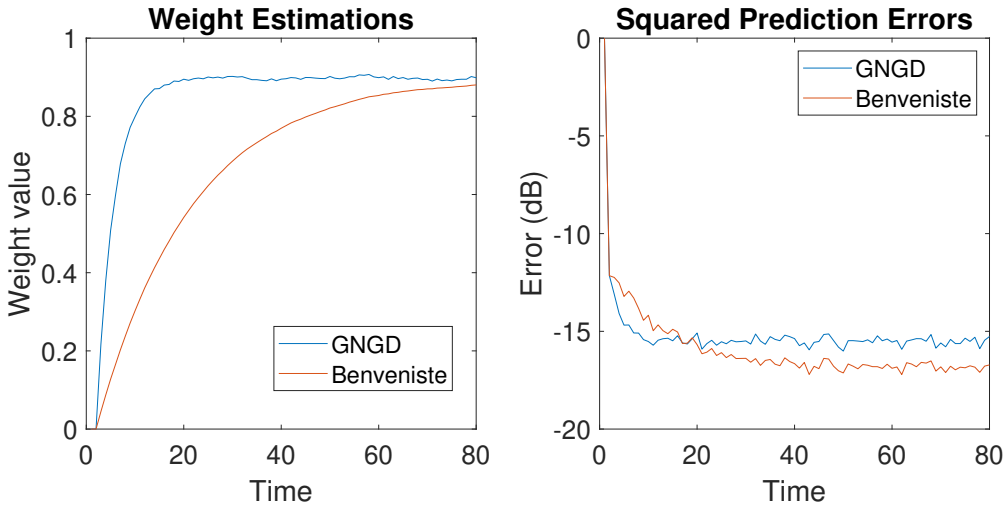


Figure 18: Weight estimations of signal x(n) for a GNGD and a Benveniste algorithm

2.3 Adaptive Noise Cancellation

(a) The adaptive line enhancer (ALE) algorithm acts as a FIR filter on noisy signals. It is a linear predictor following eqn.(28) where the predictor input $\mathbf{u}(n)$ is defined by eqn.(29).

$$\hat{x}(n) = \mathbf{w}^T(n)\mathbf{u}(n) \tag{28}$$

$$\mathbf{u}(n) = [s(n-\Delta), ..., s(n-\Delta-M+1)]^T \tag{29}$$

Given a signal s(n) (30) with noise $\eta(n)$ (31), we want to find the minimum delay Δ used in an ALE filter of length M > 1.

$$s(n) = x(n) + \eta(n) \tag{30}$$

$$\eta(n) = v(n) + 0.5v(n-2), \quad \textit{v(n)} \text{ white noise with unit variance} \tag{31}$$

The MSE of the filter estimation is given by eqn.(32).

$$\begin{aligned} E\{[s(n)-\hat{x}(n)]^2\} &= E\{[x(n)-\hat{x}(n)+\eta(n)]^2\} \\ &= E\{(x(n)-\hat{x}(n))^2\} + 2E\{\eta(n)x(n)\} - 2E\{\eta(n)\hat{x}(n)\} + E\{\eta^2(n)\} \end{aligned} \tag{32}$$

$E\{\eta^2(n)\} = Var(\eta(n)) = 1.25$ and $2E\{\eta(n)x(n)\} = 0$, since x(n) is deterministic therefore:

$$E\{[s(n)-\hat{x}(n)]^2\} = E\{(x(n)-\hat{x}(n))^2\} - 2E\{\eta(n)\hat{x}(n)\} + 1.25 \tag{33}$$

The first term of the MSE (equation (33)) is the true mean square prediction error and is not affected by Δ. Therefore to find the minimum value of Δ, we need to focus on the second term of the equation, which can be expanded by replacing $\hat{x}(n)$ with equation (28).

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

$$= -2E\{\eta(n)(\sum_{i=0}^{M} w_i(x(n-\Delta-i+1) + \eta(n-\Delta-i+1)))\}$$

since x(n) is deterministic:

$$= -2E\{\eta(n)(\sum_{i=0}^{M} w_i(v(n-\Delta-i+1) + 0.5v(n-\Delta-i-1)))\}$$

Since *v(x)* exists for all x $\geq$ 0, we get that $(n-\Delta-M-1) \geq 0$. For the minimum values of n = 0 and M = 2 we have $\Delta_{min} = 3$.

(b) An ALE filter with step size $\mu = 0.01$ was used on a signal *s(n)* (equation (30)) over 1000 samples n. The output represented in fig.(19) reduces SNR and performs better for smaller filter lengths. This is supported by the plot in figure (20) displaying the dependence of MPSE on filter length M. We can also see the negative dependence between the delay and MPSE, that is probably due to model overfitting.

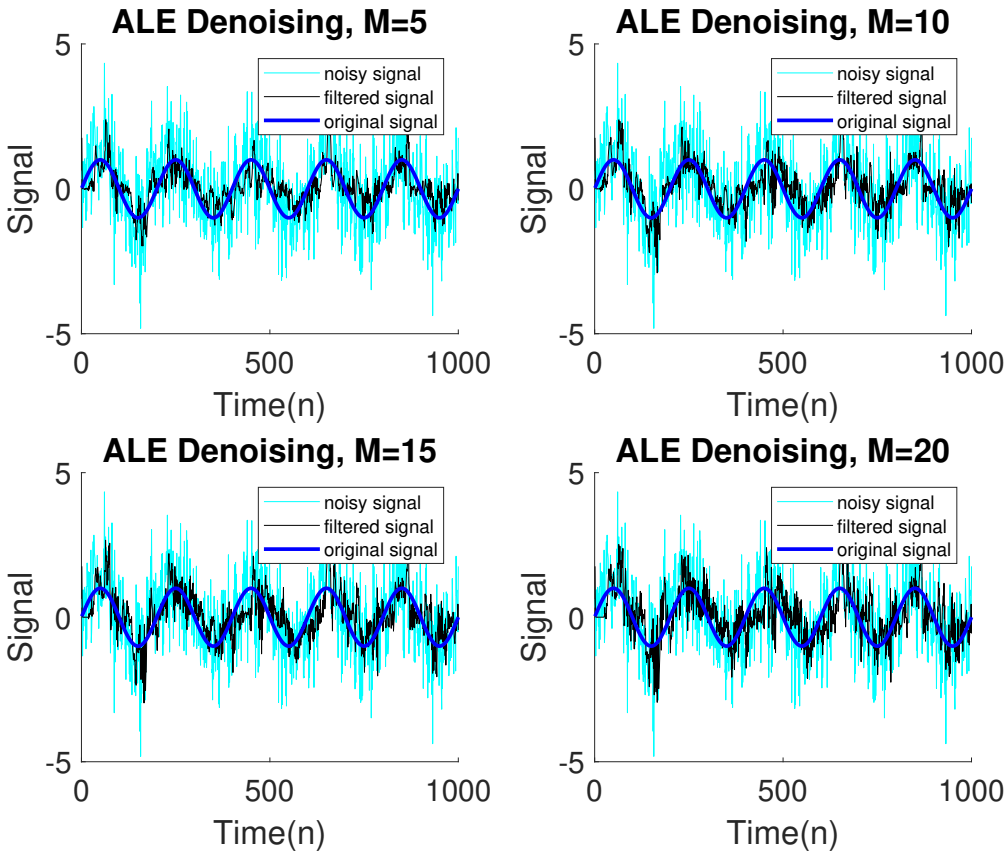


Figure 19: noisy signal vs signal denoised with ALE filter of step size $\mu = 0.01$ and lengths M=5, M=10, M=15 and M=20.

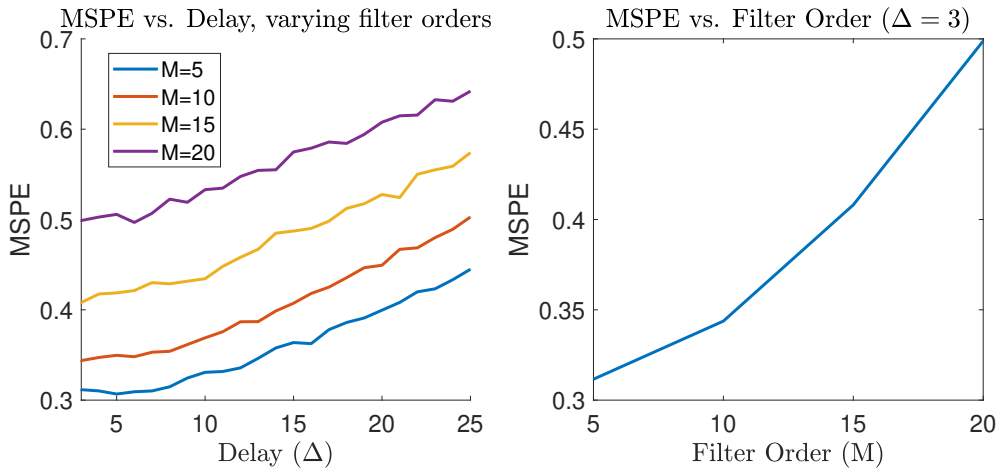


Figure 20: MSPE of signal x(n) for varying filter lengths M and delays Δ

(c) The adaptive noise cancellation (ANC) system was also tested to filter a noisy signal, and its performance is compared with an ALE filter in figure (21). We can see that the ALE filter has the advantage of converging faster, but after enough samples,

the steady-state error for an ANC filter is lower by an order of 10 dBs. This higher performance of the ANC filter is credited by a secondary noise input providing more information on the noise, but this information is not always available.

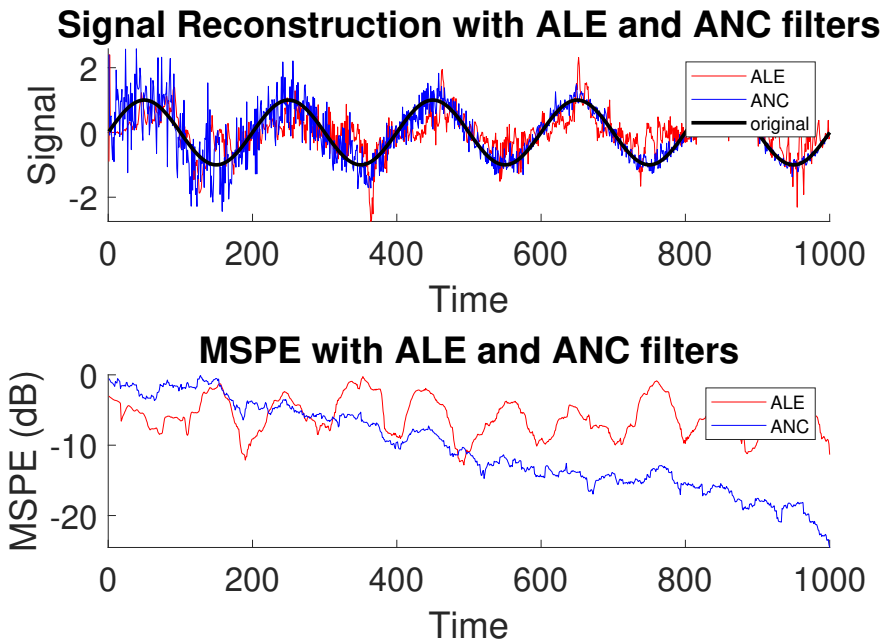


Figure 21: Comparison of ALE and ANC filters' performance over 1000 samples of signal s(n)

(d) The ANC configuration was also tested on EEG data to remove a 50Hz noise introduced by the mains. The secondary noise created for the filter is a sinusoid of 50Hz corrupted by WGN. The spectrum of the filtered data is shown in figure (22); a successful removal of the 50Hz component was obtained for a filter length of 10. Lower filter orders do not remove the the frequency band, and filters that have too high orders can remove too much of the 50 Hz component. This is visible for a filter of M = 20, where a dark band at 50 Hz is in mismatch with the baseline of the spectrogram.

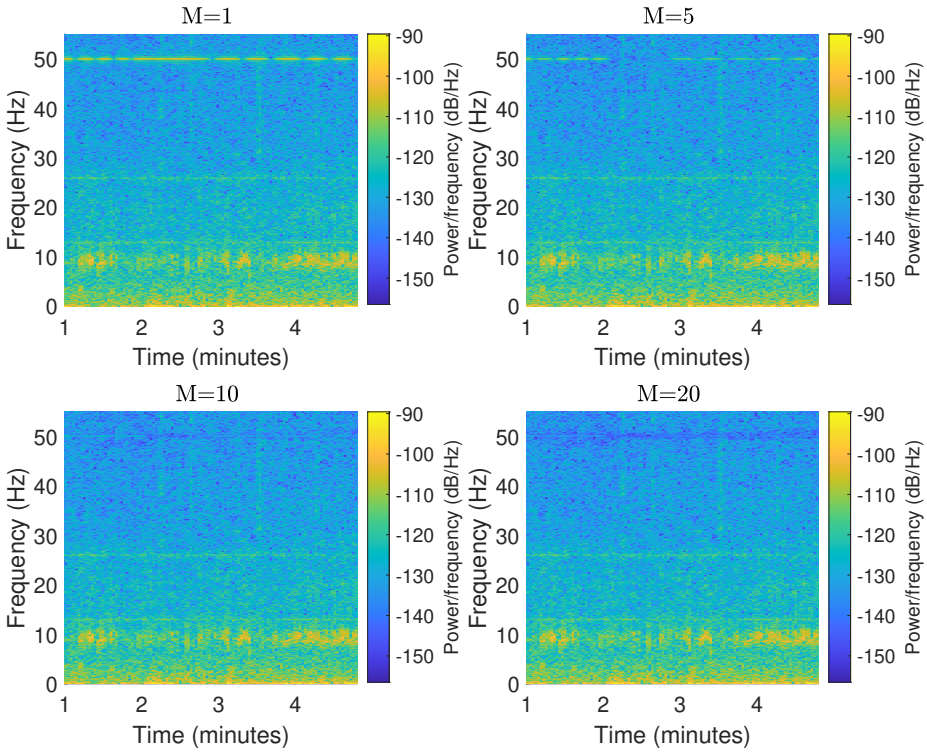


Figure 22: Spectrogram of filtered POz data from ECG recordings for different learning rates μ and filter lengths M.

3 Widely Linear Filtering and Adaptive Spectrum Estimation

3.1 Complex LMS and Widely Linear Modelling

(a) Y(n) defined in equation **??**: is a first-order widely-linear-moving-average process, WLMA(1), driven by circular white Gaussian noise x(n). It can be estimated by a complex LMS (CLMS) filter (equation (35)) or an augmented version of CLMS (ACLMS, equation (36)).

$$y(n) = x(n) + b_1 x(n-1) + b_2 x^*(n-1) \quad x \simeq \mathcal{N}(0,1) \tag{34}$$

where $b_1 = 1.5 + j$ and $B_2 = 2.5 - 0.5j$.

$$\begin{aligned} y(n) &= \mathbf{h}^H(n)\mathbf{x}(n) \\ e(n) &= y(n) - \hat{y}(n) \\ \mathbf{h}(n+1) &= \mathbf{h}(n) + \mu e^*(n)\mathbf{x}(n) \end{aligned} \tag{35}$$

$$
\begin{aligned}
\hat{y}(n) &= \mathbf{h}^H(n)\mathbf{x}(n) + \mathbf{g}^H(n)\mathbf{x}^*(n) \\
e(n) &= y(n) - \hat{y}(n) \\
\mathbf{h}(n+1) &= \mathbf{h}(n) + \mu e^*(n)\mathbf{x}(n) \\
\mathbf{g}(n+1) &= \mathbf{g}(n) + \mu e^*(n)\mathbf{x}^*(n)
\end{aligned}
\tag{36}
$$

The distribution of y(n) over 1000 samples is represented in the complex plane in figure (23) (top), along with that of complex circular WGN. We can see that y(n) is highly non-circular.

Estimations of y(n) were completed with both ACMLS and CMLS, and the learning curves over 1000 samples is shown in figure (23) (bottom). The CMLS algorithm does not learn from the WLMA data as it only caters for strictly circular data. Its learning curve is therefore constant, close to 0dB. On the other hand, estimations with the augmented input have a decreasing error over time, stabilizing around -300 dB

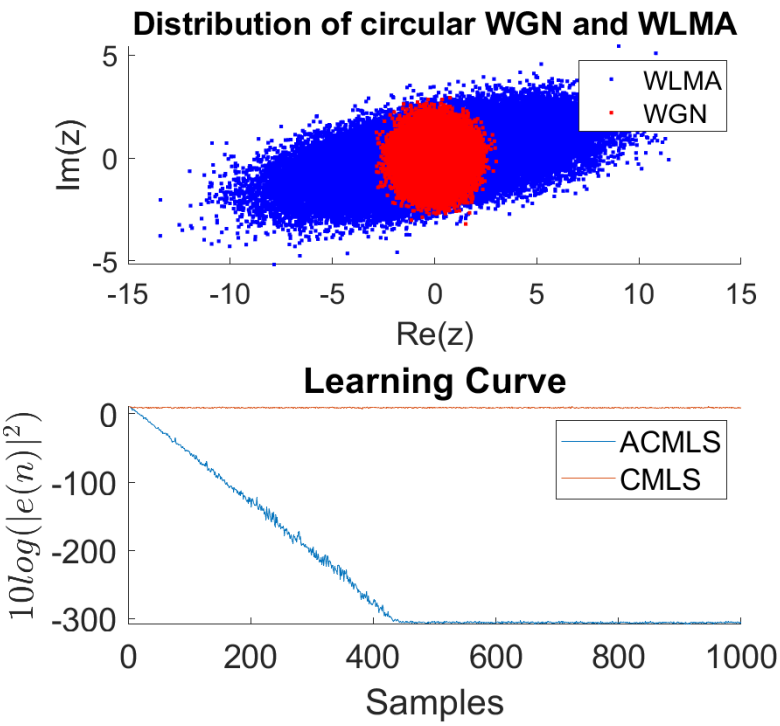


Figure 23: WGN and WLMA signals over 1000 samples (top) and Learning curves for ACMLS and CMLS (bottom)

(b) The circularity plots for the bivariate data v(n) of wind speeds in the East-West direction, v_{east}, and North-South direction, v_{north} are shown in figure (**??**) for three wind regimes (high, medium, low). The data v(n) is modelled according to equation (37):

$$v(n) = v_{east}(n) + jv_{north}(n) \tag{37}$$

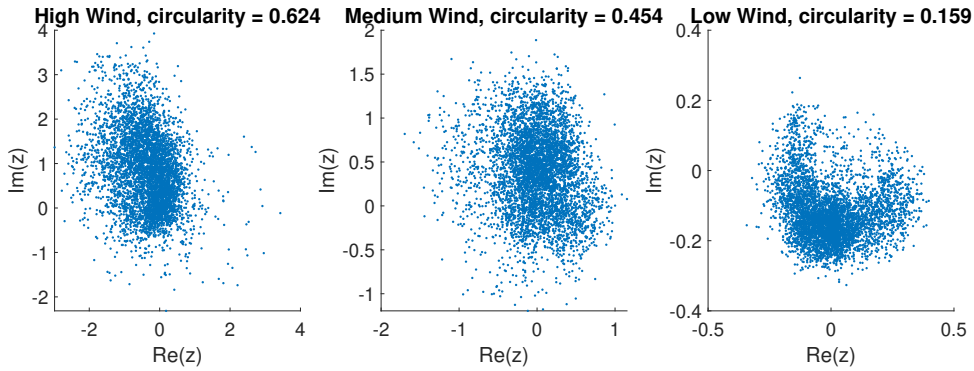


Figure 24: Circularity plots for the three wind regimes

Predictions of these 3 datasets with ACLMS and CLMS filters were made for filter lengths varying from 0 to 20. The relation btween MSPE and filter length for both algorithms is plotted in figure (25). Different learning rates (lr) were tested, and the one giving the smallest error for the optimum filter order was chosen for each wind regime. We can see that ACLMS performs better than standard CLMS for filter lengths lower than 5 for all three wind regimes. This is because the complex datasets are not circular so they are better estimated with ACLMS. As filter length increase, the ACLMS algorithm tends to overfit the data and the error becomes larger than that of the CLMS.

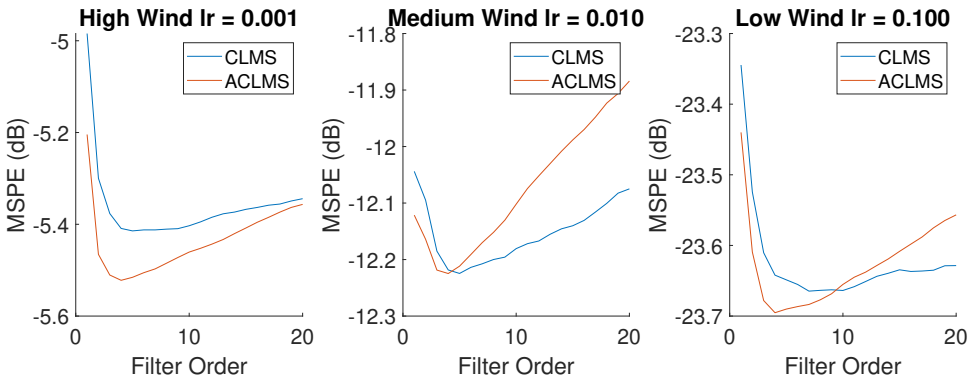


Figure 25: CLMS and ACLMS filters' performances for one-step-ahead prediction of complex wind data, for three different wind regimes

(c) A three-phases power system composed of voltages v_a, v_b,v_c represented by equation (38) can be projected onto a Clarke transform representation given in equation (3.1).

$$\begin{bmatrix} v_a(n) \\ v_b(n) \\ v_c(n) \end{bmatrix} = \begin{bmatrix} V_a(n)\cos\left(2\pi\frac{f_o}{f_s}n + \phi\right) \\ V_b(n)\cos\left(2\pi\frac{f_o}{f_s}n + \phi + \Delta_b - \frac{2\pi}{3}\right) \\ V_c(n)\cos\left(2\pi\frac{f_o}{f_s}n + \phi + \Delta_c + \frac{2\pi}{3}\right) \end{bmatrix} \tag{38}$$

$$\begin{bmatrix} v_0(n) \\ v_\alpha(n) \\ v_\beta(n) \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} v_a(n) \\ v_b(n) \\ v_c(n) \end{bmatrix}$$

(39)

Under balanced conditions, $V_a(n) = V_b(n) = V_c(n) = V$ and $\Delta_b = \Delta_c = 0$, making v_0 go to zero in the Clark transform of the system. v(n) can be represented with complex exponentials:

$$v(n) = \sqrt{\frac{3}{2}} V e^{j\left(2\pi \frac{f_0}{f_s} n + \phi\right)} \quad \text{under balanced conditions} \tag{40}$$

$$v(n) = A(n) e^{j\left(2\pi \frac{f_0}{f_s} n + \phi\right)} + B(n) e^{-j\left(2\pi \frac{f_0}{f_s} n + \phi\right)} \quad \text{general formula} \tag{41}$$

$$\text{where } A(n) = \frac{\sqrt{6}}{6}\left(V_a(n) + V_b(n) e^{j\Delta_b} + V_c(n) e^{j\Delta_c}\right)$$

$$\text{and } B(n) = \frac{\sqrt{6}}{6}\left(V_a(n) + V_b(n) e^{-j(\Delta_b + 2\pi/3)} + V_c(n) e^{-j(\Delta_c - 2\pi/3)}\right).$$

It can be observed in figure (**??**) that a balanced system is circular. When the magnitudes or phases of the system are different between the voltages, the system becomes unbalanced and is no longer circular. The less circular (higher circularity coefficient) the system, the more unbalanced it becomes, so a circularity diagram can be used to identify how unbalanced is a system.

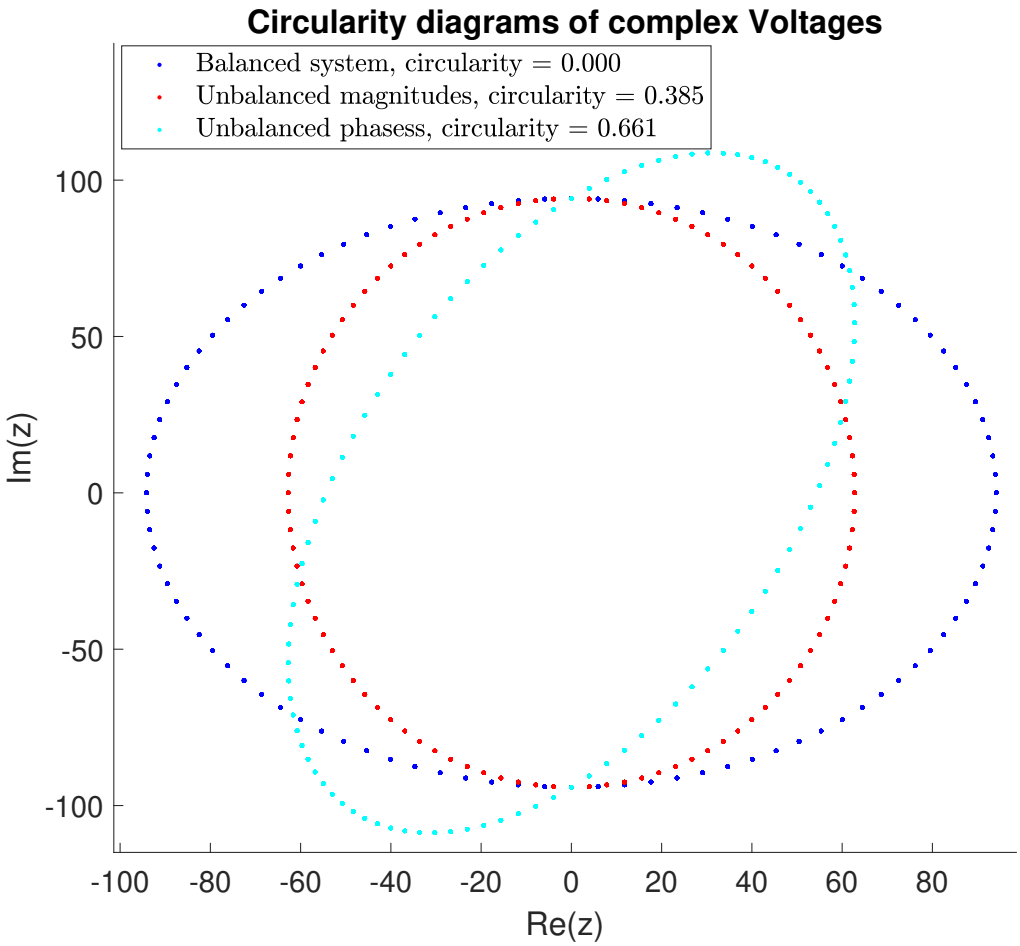


Figure 26: Circularity plot of a balanced system of voltages vs an unbalanced system

(d) The frequency f_o of the balanced complex $\alpha - \beta$ voltage in (40) can be derived from the coefficient h(n) of a strictly linear AR model (equation **??**):

$$v(n+1) = \sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_o}{f_s}(n+1)+\phi)} = h^*(n)v(n) \tag{42}$$

$$\Leftrightarrow \quad \sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_o}{f_s}(n)+\phi)} e^{j(2\pi \frac{f_o}{f_s})} = h^*(n)\sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_o}{f_s}(n)+\phi)} \tag{43}$$

$$\Leftrightarrow \quad e^{j(2\pi \frac{f_o}{f_s})} = h^*(n) \tag{44}$$

$$\Rightarrow \quad \angle e^{j\left(2\pi \frac{f_o}{f_s}\right)} = 2\pi \frac{f_o}{f_s} = arctan\left(\frac{-Im(h(n))}{Re(h(n))}\right) \tag{45}$$

$$\Leftrightarrow \quad f_o = \frac{f_s}{2\pi} arctan\left(\frac{-Im(h(n))}{Re(h(n))}\right) \tag{46}$$

Similarly for an unbalanced system, using the widely linear AR model:

$$v(n+1) = h^*(n)v(n) + g^*(n)v^*(n) \tag{47}$$

Substituting the expression for $v(n)$ in (41) into the widely linear model:

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

By comparing coefficients of the two complex exponentials, we get:

$$\begin{aligned} A(n+1)e^{j\left(2\pi\frac{f_0}{f_s}\right)} &= h^*(n)A(n) + g^*(n)B^*(n) \\ B(n+1)e^{-j\left(2\pi\frac{f_0}{f_s}\right)} &= h^*(n)B(n) + g^*(n)A^*(n) \end{aligned} \tag{48}$$

We can assume that the coefficients A(n) and B(n) do not change much between two consecutive time-steps of the filtering process: $A(n) \approx A(n+1), B(n) \approx B(n+1)$.

$$\begin{aligned} \Rightarrow e^{j\left(2\pi\frac{f_0}{f_s}\right)} &= h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} \\ e^{-j\left(2\pi\frac{f_0}{f_s}\right)} &= h^*(n) + g^*(n)\frac{A^*(n)}{B(n)} \end{aligned} \tag{49}$$

$$\begin{aligned} \Rightarrow & h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} = h(n) + g(n)\frac{A(n)}{B^*(n)} \\ \Rightarrow & h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} - h(n) - g(n)\frac{A(n)}{B^*(n)} = 0 \\ \Rightarrow & g^*(n)\left(\frac{B^*(n)}{A(n)}\right)^2 + (h^*(n) - h(n))\frac{B^*(n)}{A(n)} - g(n) = 0 \\ \Rightarrow & g^*(n)\left(\frac{B^*(n)}{A(n)}\right)^2 - 2j\,\mathrm{Im}\{h(n)\}\frac{B^*(n)}{A(n)} - g(n) = 0 \end{aligned}$$

Solving for $\frac{B^*(n)}{A(n)}$:

$$\begin{aligned} \frac{B^*(n)}{A(n)} &= \frac{2j\,\mathrm{Im}\{h(n)\} \pm \sqrt{(-2j\,\mathrm{Im}\{h(n)\})^2 + 4}}{2g^*(n)} \\ &= j\frac{\mathrm{Im}\{h(n)\} \pm \sqrt{\mathrm{Im}^2\{h(n)\} - |g(n)|^2}}{g^*(n)} \end{aligned}$$

Substituting into (48):

$$\begin{aligned} e^{j\left(2\pi\frac{f_0}{f_s}\right)} &= h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} = h^*(n) + g^*(n)j\frac{\mathrm{Im}\{h(n)\} \pm \sqrt{\mathrm{Im}^2\{h(n)\} - |g(n)|^2}}{g^*(n)} \\ &= \mathrm{Re}\{h(n)\} - j\,\mathrm{Im}\{h(n)\} + j\,\mathrm{Im}\{h(n)\} \pm j\sqrt{\mathrm{Im}^2\{h(n)\} - |g(n)|^2} \end{aligned}$$

$$\Rightarrow \angle e^{j\left(2\pi \frac{f_o}{f_s}\right)} = 2\pi \frac{f_o}{f_s} = \arctan\left(\frac{\mathrm{Im}}{\mathrm{Re}}\right) = \arctan\left(\frac{\sqrt{\mathrm{Im}^2\{h(n)\} - |g(n)|^2}}{\mathrm{Re}\{h(n)\}}\right)$$

(e) Using the CLMS and ACLMS algorithms given in (**??**) (**??**), estimations of the nominal frequency (f_o = 50 Hz) of $\alpha - \beta$ voltages are plotted in figure (27) under balanced and unbalanced conditions. For unbalanced systems, the CLMS does not converge to a frequncy value

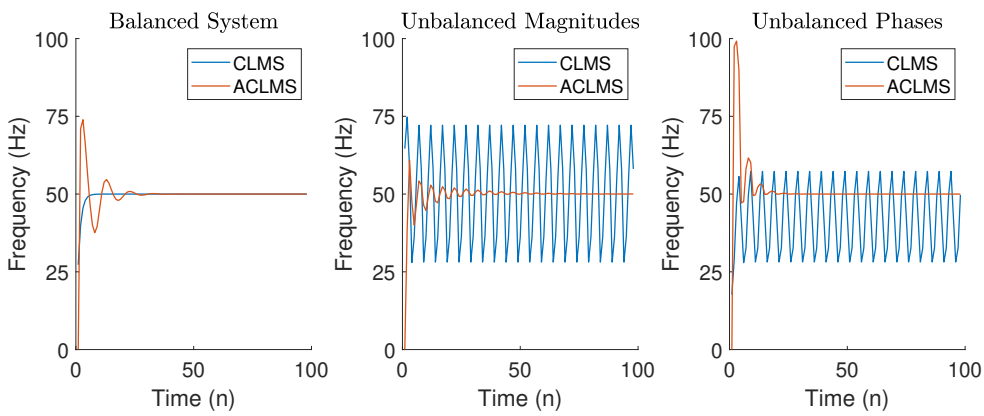


Figure 27: Frequency estimates of $\alpha - \beta$ voltages using CLMS and ACLMS algorithms on balanced and un-balanced voltage systems

3.2 Adaptive AR Model Based Tim-Frequency Estimation

(a) A frequency modulate signal (FM) $y(n) = e^{j()\frac{2\pi}{f_s}\phi(n)} + \eta(n)$ was generated with $\eta(n)$ a circular complex valued white noise and ϕ defined by:

$$\frac{d\phi(n)}{dn} = \begin{cases} 100 & 1 \leq n \leq 500 \\ 100 + \frac{n-500}{2} & 501 \leq n \leq 1000 \\ 100 + \left(\frac{n-1000}{25}\right)^2 & 1001 \leq n \leq 1500 \end{cases} \tag{50}$$

The estimations of the signal with AR models of varying order (figure (28)) are computed under the assumption of a stationary signal and therefore fail to detect changes in frequency. The spectrums only show frequency averages over window spans.

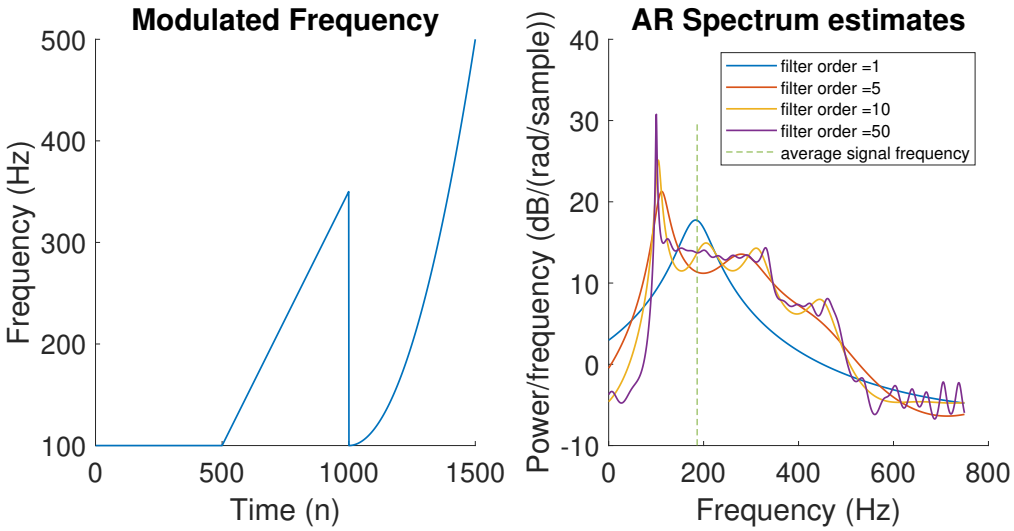


Figure 28: Frequency estimates of the signal y(n) (left) and power spectra of signal estimations (Yule-Walker AR method) for different filter orders.

(b) The CLMS algorithm does not assume stationarity and can be used to compute a frequency spectrum estimation of y(n) over time (figure (29)). The choice of learning rate will affect this estimation: if it is to low, such as 0.005 in figure (29), the CLMS does not learn appropriately.

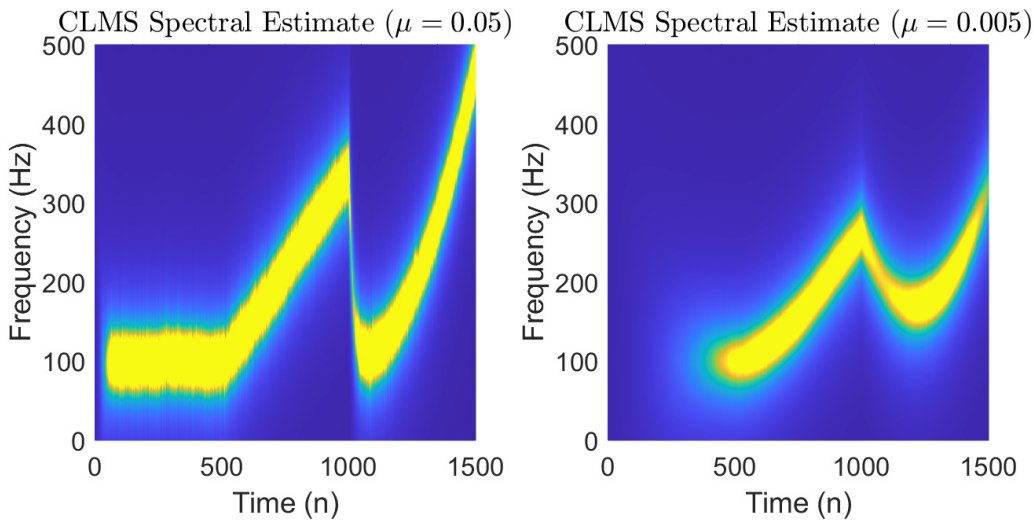


Figure 29: CLMS spectral estimation of y(n)'s frequency over time for different learning rates

3.3 A Real-Time Spectrum Analyser Using Least-Mean Square

(a) For a signal y(n) that uses a linear combination of N harmonically related sinusoid (eqn.(3.3)), the cost function for N samples is given by eqn(52).

$$\hat{y}(n) = \sum_{n=0}^{N-1} w(k) e^{j2\pi kn/N} \Leftrightarrow \quad \hat{\mathbf{y}} = \mathbf{F}\mathbf{w}$$

(51)

$$\text{with } \mathbf{F} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & e^{j\frac{2\pi}{N}(1)(1)} & \dots & e^{j\frac{2\pi}{N}(1)(N-1)} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ 1 & e^{j\frac{2\pi}{N}(N-1)(1)} & \dots & e^{j\frac{2\pi}{N}(N-1)(N-11)} \end{bmatrix} \tag{52}$$

$$||\mathbf{y} - \hat{\mathbf{y}}||^2 = \sum_{n=0}^{N-1} |y(n) - \hat{y}(n)|^2$$

By minimizing the cost function, we can find the optimal weights $\mathbf{w}$ (eqn(54)):

$$\begin{aligned} min_w \quad ||\mathbf{y} - \hat{\mathbf{y}}||^2 &= min_w \quad (\mathbf{y} - \mathbf{F}\mathbf{w})^H (\mathbf{y} - \mathbf{F}\mathbf{w}) \\ &= min_w \quad (\mathbf{y}^H\mathbf{y} - \mathbf{w}^H\mathbf{F}^H\mathbf{y} - \mathbf{y}^H\mathbf{F}\mathbf{w} + \mathbf{w}^H\mathbf{F}^H\mathbf{F}\mathbf{w} \\ &= min_w \quad (\mathbf{y}^H\mathbf{y} - 2\mathbf{w}^H\mathbf{F}^H\mathbf{y} + \mathbf{w}^H\mathbf{F}^H\mathbf{F}\mathbf{w}) \end{aligned} \tag{54}$$

$$\Leftrightarrow \quad \nabla||\mathbf{y} - \hat{\mathbf{y}}||^2 = -2\mathbf{F}^H\mathbf{y} + 2\mathbf{F}^H\mathbf{F}\mathbf{w} = 0 \tag{55}$$

$$\Leftrightarrow \quad \mathbf{w} = (\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H\mathbf{y} \tag{56}$$

The equation in (3.3) is the expression of an inverse DFT, therefore the weights $\mathbf{w}$ can be seen as DFT coefficients.
(c) The DFT-CLMS for signal y(n) was implemented with the filter input at time n given by: $\mathbf{x}(n) = \frac{1}{N}\left[1, e^{j\frac{2n\pi}{N}}, e^{j\frac{4n\pi}{N}}, \dots, e^{j\frac{2n\pi(N-1)}{N}}\right]$. The spectral estimations of the frequency are plotted in figure (30). For a standard DFT-CLMS (no leakage coefficient), the spectrum does not resemble the true power spectrum, because it superposes the learned weights instead of updating them. When introducing leakage, the algorithm can adapt to time variations of the frequency. If γ is chosen to high it can lead to instable estimations and distort the spectrum.

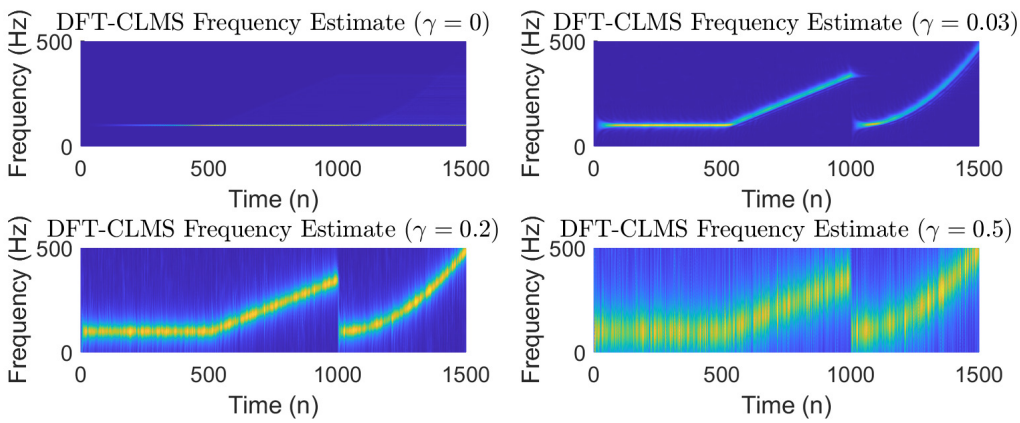


Figure 30: DFT-LMS estimations of the frequency f(n) for different leak coefficients, learning rate of 1, w(0) = 0

(d) The DFT-CLMS was then used for spectral frequency estimations over 1200 samples of EEG data from the POz region of the brain. Responses around 50Hz, 9Hz and fainter harmonics are visible after an initial learning period of 500 samples approximately. These frequencies correspond to the mains and the SSVEP of the signal, proving the efficiency of the DFT-CMLS algorithm in this case.

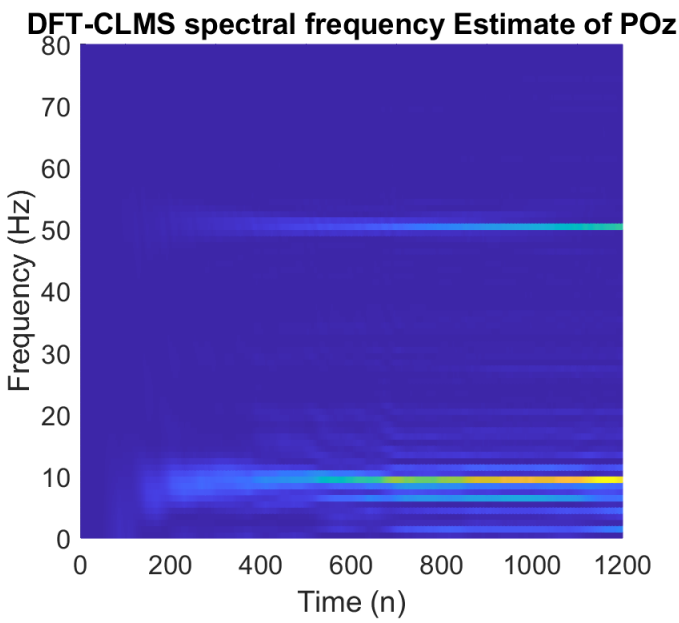


Figure 31: DFT-LMS spectral estimations of the frequency of EEG data POz over time

4 From LMS to Deep Learning

(1) A mean-centered time series is plotted with its AR[4] estimation based on the LMS algorithm in figure (32). The system has a low prediction accuracy during the first timesteps, and starts converging after 200 samples approximately. The MSE of

the prediction after running the simulation on matlab for 1000 samples is 40.1083 dB and the prediction gain $R_p = 10log_{10}(\frac{\hat{\sigma_y}^2}{\sigma_e^2}) = 5.1953$ dB where $\hat{\sigma_y}^2$ is the variance of the output of LMS, and σ_e the variance of the prediction error. After the estimation signal converges, it seems to match the original signal quite well as can be observed in the plot, but the algorithm dos not account for non-linearities of the signal, which leaves room for improvement.

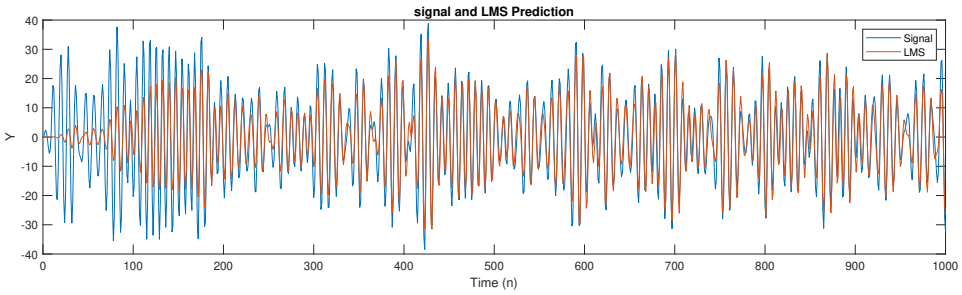


Figure 32: Signal and its linear LMS estimation

(2) When adding a non-linear activation function to the output of the LMS , the model can become better at matching non-linearities of a series. The same signal as in part (1) was plotted with its dynamical perceptron prediction (using $\Phi = tanh()$). We can observe a clear mismatch of amplitudes between the two signals, which can be solved by adapting the amplitude of the activation function to the amplitude of the original signal.

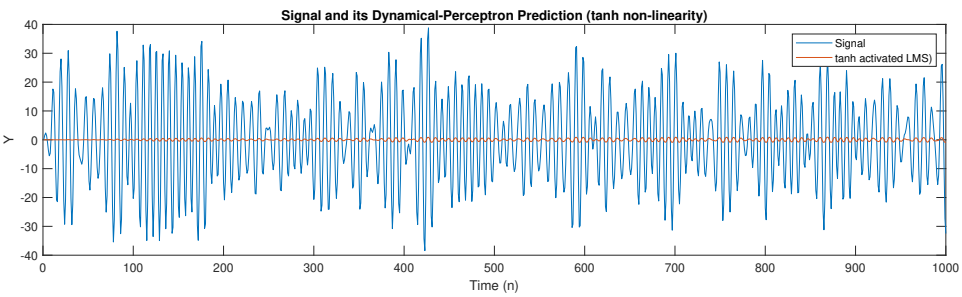


Figure 33: Signal and its non-linear LMS estimation with activation function tanh

(3) By scaling the amplitude of the activation function to 50 we obtain the plot in figure (34). The estimation matches closely the signal, with a significantly lower adaption time than in part (1), and the MSE dropped to 6.6449 dB. The prediction gain is higher than with a standard LMS, at 14.9816 dB.

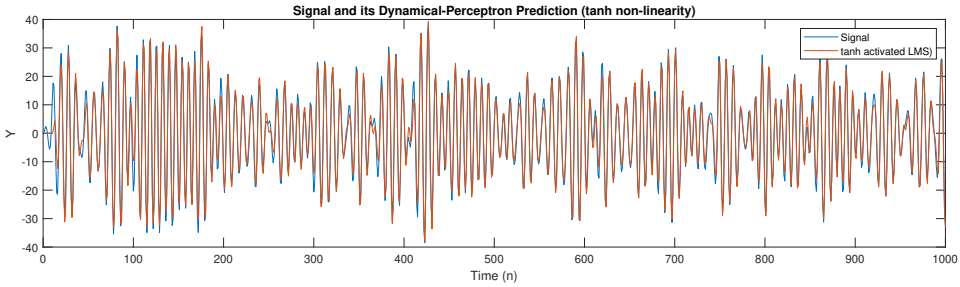


Figure 34: Signal and its non-linear LMS estimation with activation function 50*tanh

(4 and 5) The previous Estimations were conducted on a mean-centered signal. For a time-series with a non-zero mean, the prediction model can be augmented with a bias. The resulting MSE is 12.3918 and Rp = 12.5859, making a worse performance than in the previous section. This is due to an initial offset between the signal which no longer has a mean of 0, and the prediction that starts at 0. These observations are presented in figure (**??**). In the same figure is plotted as well a tanh activated LMS prediction with bias that had its first 20 weight predictions trained over 100 iterations before implementing the AR filter. This pre-training avoids the inital offset observed before, and reduces the error of the prediction to an MSE of 7.2712 and Rp of 14.6425.

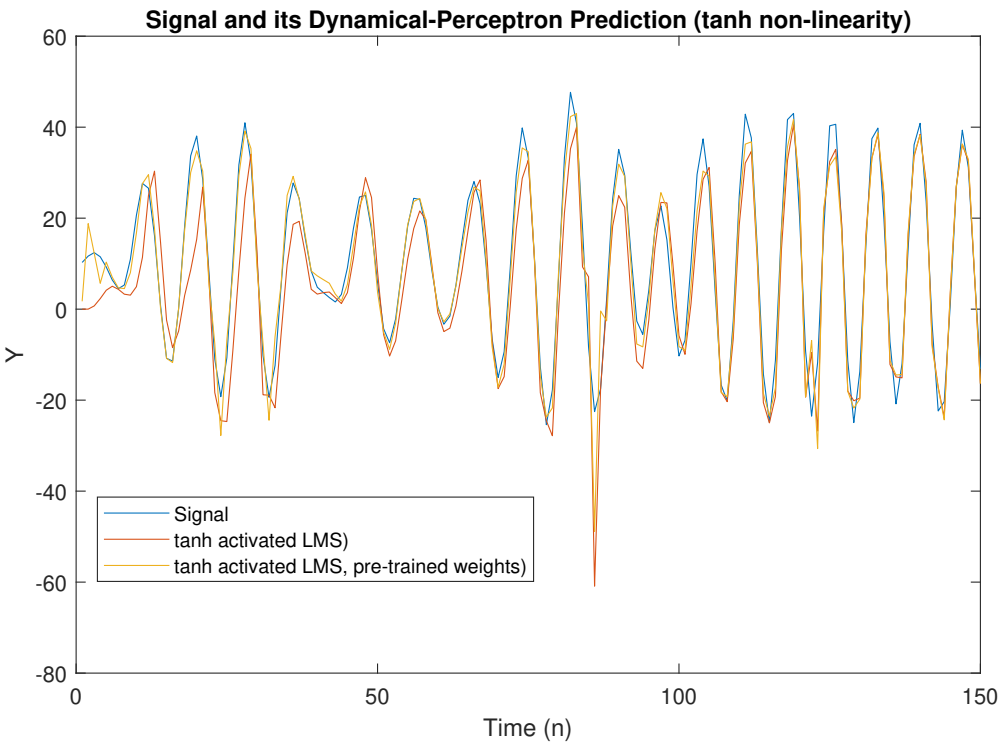


Figure 35: initial segment of Signal and its non-linear LMS estimations with activation function 50*tanh, for pre-trained and non-trained weights

(6) A backpropagation algorithm computes weight estimations on each neuron of

deep network layer by feeding from previous layers. At the output, the prediction error from all paths are compared, and this affects how weights are updated in each node (a larger error will lead to a more significant change). The updates are carried backwards until the input layer.
(7) A sinusoidal time series with added noise of power 0.05 was estimated by a linear LMS algorithm, a dynamical perceptron and a deep Network with 4 hidden layers (Figure(36)). The deep network provides better predictions for training data as the the weights are trained to perceive non linearities in the signal. Indeed the steady-state loss for deep network estimations of train data is of 0.05 while it is at 0.1 for a dynamical perceptron. For test data, the steady-state loss for the neural network is comparable to the other algorithms. It does not converge as fast because the network's complexity makes the estimation fit into noise as well as the signal.

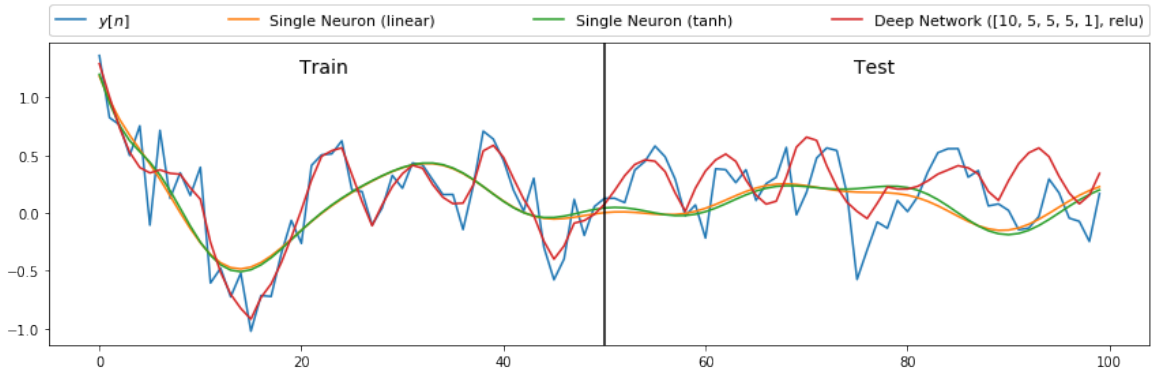


Figure 36: Prediction of a noisy sinusoid for different algorithms after 20,000 epochs and noise power 0.05.

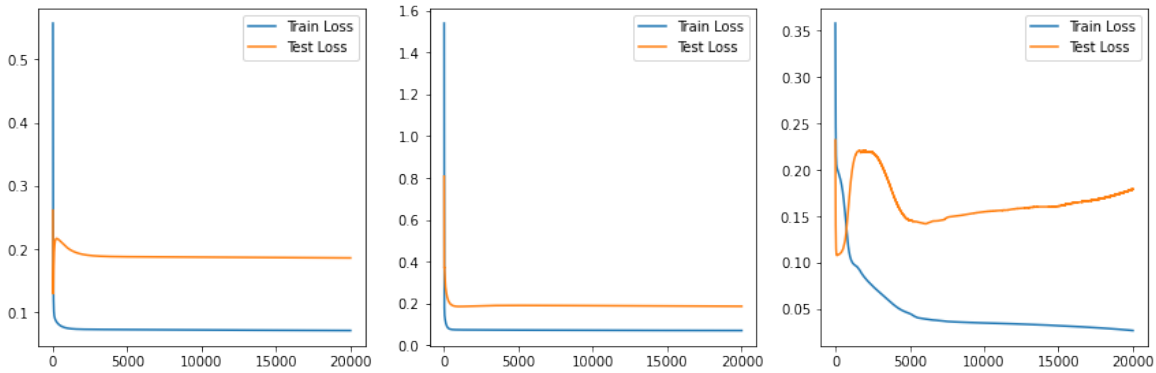


Figure 37: Learning curves over epochs for training and testing sets. Left: standard LMS. Middle: dynamical perceptron. Right: deep network. Noise power = 0.05.

(8) Figures (38) and (39) provide estimation of the same signal and the corresponding learning curves, for a noise power of zero. Without noise, the deep network outperforms the other implementations with a steady state loss of practically 0 for training data, and around 0.025 for test data, compared to 0.05 and 0.1 respectively for the other methods.

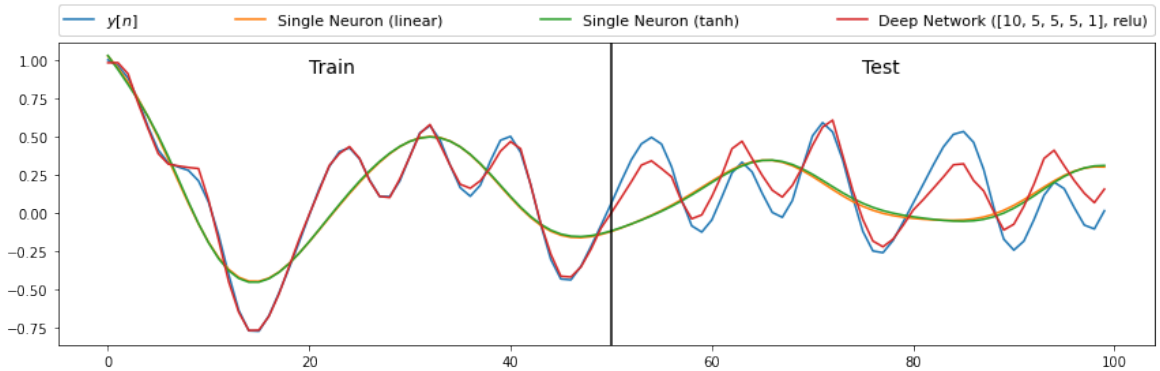


Figure 38: Prediction of a noisy sinusoid for different algorithms after 20,000 epochs and noise power 0.

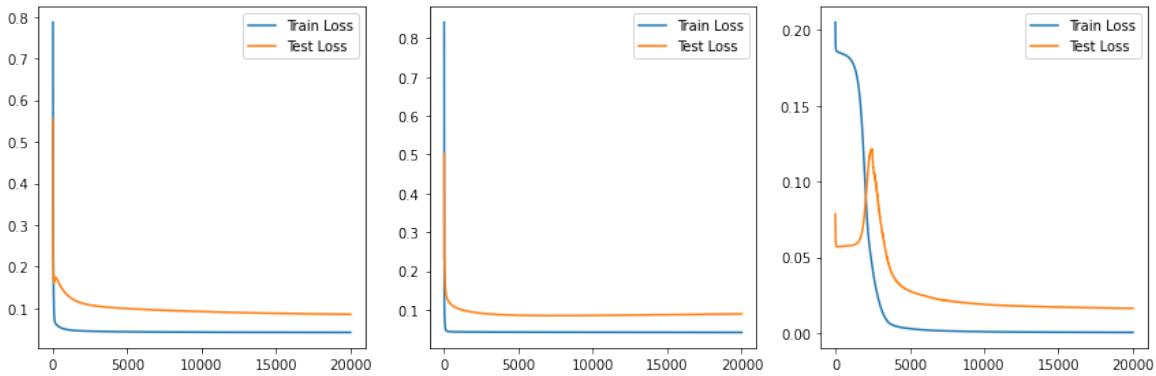


Figure 39: Learning curves over epochs for training and testing sets. Left: standard LMS. Middle: dynamical perceptron. Right: deep network. Noise power = 0.

Figures (40) and (41) provide the same estimations and learning curves, with an increased noise power of 0.5. This time, the complexity of the deep network algorithm over-fits the data to account for noise, and the system does not converge. This is one of the drawbacks of deep learning for prediction of noisy signals.

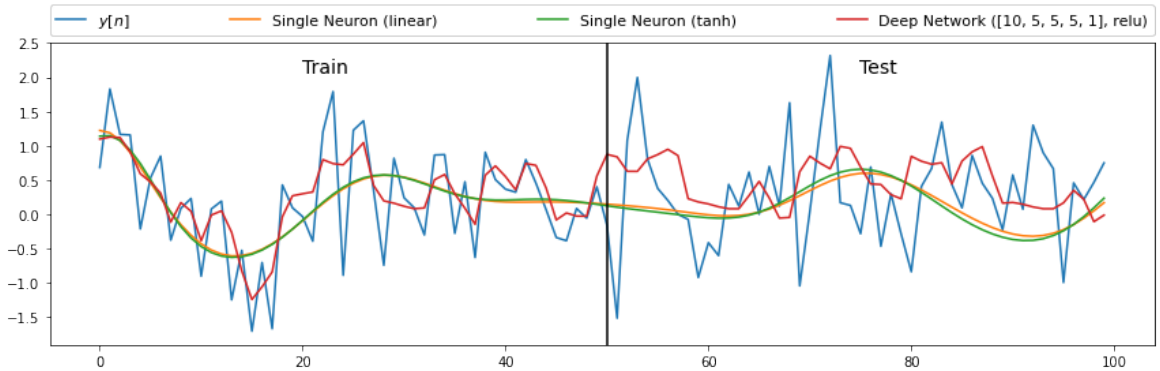


Figure 40: Prediction of a noisy sinusoid for different algorithms after 20,000 epochs and noise power 0.5.

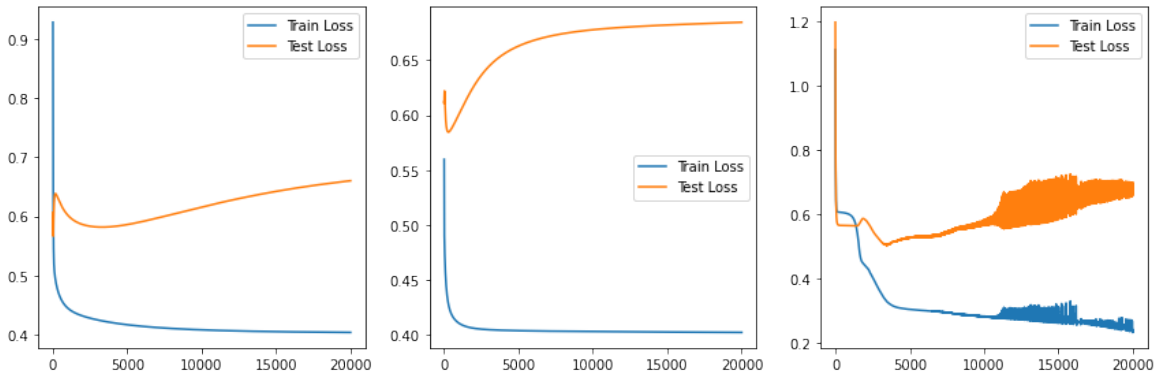


Figure 41: Learning curves over epochs for training and testing sets. Left: standard LMS. Middle: dynamical perceptron. Right: deep network. Noise power = 0.5.

5 Tensor Decompositions for Big Data Applications

The code and answers for this section were submitted along the matlab files in html form.